Skoltech

Skolkovo Institute of Science and Technology

MASTER'S THESIS

Integrating Concurrent Conceptual Systems Design with 3D Modeling

Master's Educational Program Space and Engineering Systems

Student: Nikita Letov

signature, name

Research Advisor: Clément Fortin, Professor

signature, name, title

Moscow 2018

Skoltech

Skolkovo Institute of Science and Technology

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Интегрирование параллельного концептуального проектирования систем с помощью трёхмерного моделирования

Магистерская образовательная программа «Космические и инженерные системы»

Студент: Летов Никита Николаевич
подпись, ФИО

Научный руководитель: Фортин Клемент, Профессор
подпись, ФИО, должность

Москва, 2018

Integrating Concurrent Conceptual Systems Design with 3D Modeling

Nikita Letov

Submitted to the Skolkovo Institute of Science and Technology on June 1 2018

ABSTRACT

Nowadays, one of system-wide approaches to facilitate a product realization process is Concurrent Engineering which can be applied owing to being enable to choose the best practice to improve product introduction process, being capable to improve cross functional integration and communication, and being empowered to apply a set of comprehensive methods for design analysis so that designers can select the most optimal design solution which is not only considering the design constraints, but also taking the constraints of production system, logistics and distribution into account. Hence, it can cover majority of problems in conceptual design phase which are generated due to lack of empathy between design and manufacturing.

Data used in initial phases of product development is predominantly behavioral in nature, that is, a large part of this data does not refer to the geometrical parameters of the system. However, current PLM-systems are based on the geometrical master model concept and thus work best for the detailed design, where the data is mostly geometrical. Thereunder, there is still a gap between parametric and geometric modelling which has to be eventually filled.

The purpose of this study was the implementing of a risk representing methodology that could enhance development of complex system architectures and technology planning based on 3D modeling. It was proposed to develop a software prototype which should consist of a parametric modeling tool, a geometric modeling tool, and a system modeling methodology embedded in it. It is proposed to fill the need in the parametric modeling tool by CEDESK – a concurrent conceptual design tool (Knoll and Golkar, 2016). C3D Modeler is proposed to be a geometrical modeling tool for the software prototype. The estimation of the technology integration risk proposed by Garg et al. (2017) was studied, adapted and implemented in a software prototype of the demonstrator as a way to represent interfaces of a system and risk they possess.

As a result, the methodology was implanted into a software prototype which could be used for concurrent conceptual design of complex systems such as space systems. The software prototype allows to represent a 3D model of a system being developed. Impact scores, failure likelihood, and risk scores of interfaces between subsystems of the system are represented as a 3D design structure matrix (DSM) histogram. The user has an ability to retrieve information for each interface. The results indicate that the developed software prototype has a potential to enhance demonstrator feasibility assessment by representing large amounts of interfaces in 3D and to ensure a successful development of a product.

Research Advisor:
Name: Clément Fortin
Degree: ing. PhD, Mechanical Engineering, Queen's University at Kingston, Canada (1985)
Title: Professor

Интегрирование параллельного концептуального проектирования систем с помощью трёхмерного моделирования

Летов Никита Николаевич

Представлено в Сколковский институт науки и технологий 1 июня 2018 г.

РЕФЕРАТ

Ныне, одним из всесистемных подходов по содействию процессу реализации изделия является параллельное проектирование, которое можно подобающим образом применять с целью получить выборку наилучшей практики чтобы улучшить процесс внедрения изделия; улучшить межфункциональные интеграцию и связь; а также иметь возможность применить исчерпывающие методы для анализа проектирования так, что проектировщики могли выбрать наиболее оптимальное проектировочное решение, рассматривая не только проектировочные ограничения, но и ограничения производственной системы, логистики и распределения. Следовательно, это может решить большинство проблем на этапе концептуального проектирования, возникающие из-за нехватки взаимосвязи между проектированием и производством.

Данные, используемые на начальных фазах разработки изделия, преимущественно поведенческие, бóльшая часть этих данных не связана с геометрическими параметрами системы. Однако, существующие PLM-системы основаны на концепте ведущей геометрической модели, а следовательно, лучше всего применяются на этапе детального проектирования, где данные в большинстве геометрические. А значит, ныне существует разрыв между параметрическим и геометрическим проектированием, который должен быть преодолён.

Цель данной работы состоит во внедрении методологии для отображения интерфейсов, которая могла бы усовершенствовать разработку архитектур сложных систем и технологическое планирование на основе трёхмерного моделирования. Было предложено разработать программный прототип, состоящего из инструмента параметрического моделирования, инструмента геометрического моделирования и методологии для моделирования систем. В качестве инструмента параметрического моделирования было предложено использовать CEDESK – инструмент для параллельного концептуального проектирования (Knoll and Golkar, 2016). C3D Modeler был предложен как инструмент геометрического моделиро-вания. Оценка рисков интегрирования технологий (Garg et al., 2017) была изучена, адаптирована и внедрена в программный прототип демонстрирующей модели в качестве способа отображения интерфейсов системы и рисков, которые они заключают.

Как результат, эта методология была внедрена в программный прототип, который может использоваться для параллельного концептуального проектирования таких сложных систем, как космические системы. Программный прототип позволяет отображать 3D модель разрабатываемой системы. Оценка влияния, вероятность отказа и оценка рисков интерфейсов между подсистемами системы отображаются в виде 3D гистограммы проектировочной структурной матрицы (DSM). Пользователь может получить информацию о каждом интерфейсе. Результаты указывают на то, что разработанный программный прототип обладает потенциалом для усовершенствования оценки осуществимости демонстрирующей модели посредством отображения большого количества связей в 3D и обеспечения успешной разработки изделия.

Научный руководитель:
Ф.И.О.: Клемент Фортин
Учёное звание, степень: ing. PhD, Машиностроение, Университет Куинс в Кингстоне, Канада (1985)
Должность: Профессор

ACKNOWLEDGEMENTS

Firstly, I specially want to thank Clément Fortin for supervising me throughout the whole period of the study, for believing in me, for his full support, insightful discussions and mentorship throughout the project, and giving me the opportunity to work with Airbus, and getting the chance to do my Master's thesis at Skoltech. Clément's constant enthusiasm for research fueled me during my time at Skoltech, and I suspect that most of the good ideas I had were secretly planted there by Clément, and he convinced me they were my own.

I am truly indebted to Rob Vingerhoeds, who hosted me for two months at ISAE–SUPAERO and I want to thank him for the interest shown in my work and the input he provided. He balanced his careful, thorough skepticism of every word, graph, and thought I presented him with a strong undercurrent of encouragement and support, which improved me greatly as a researcher and a person.

I would like to thank Dominik Knoll, who provided insight and excellent recommendations. Without him developing CEDESK, the whole project couldn't be possible. Thanks to Ilya Yuskevich and Ksenia Smirnova, who, alongside with me, worked hard in Toulouse, which was truly difficult considering the nice weather of the South of France. Thanks to Rustam Akhtyamov, Anastasia Stelvaga, Simone Briatore, Olga Danko, Nikolay Groshkov, Nicola Garzaniti, Carolina Moreno Aguirrem and Ignacio Hernandez Arroyo for working alongside with you all at CEDL.

I would like to thank Alessandro Golkar for his patience and allowing me to visit the Airbus R&T campus and see the way the Concurrent Design Faculty of Airbus works. It was always a pleasure talking with him about the project and everything else in the world. I am very thankful for his advice and support even though I wasn't his problem.

I would like to thank many other people at the Airbus R&T campus who influenced me as well. Thanks to Olivier de Weck and Fabienne Robin for being amazingly supportive. Thanks to Jyotsna Budideti, Sandro Salgueiro, Ulkar Alakbarova, Jonathan Karl Landolt, Vasco Guilherme Pesquita, Raffaele Nova Gradini and Nitin Ramchand Lalwani for having great discussions over lunches.

I couldn't have made it through without the staff of C3D Labs. Thanks to Oleg Zykov for providing me the internship opportunity. Thanks to Eduard Maximenko, Alexey Kozyrev, and Nikolay Golovanov, I have learned the theoretical foundations of geometric modeling and gained practical experience of applying a solid modeling kernel.

I would like to thank Dzmitry Tsetserukou for helping me out in my time of need when I had to transfer to the Space CREI and for teaching me robotics and control theory better than anyone else

could possibly do. Thank you to Ighor Uzhinnsky and his research group members – Sergei Nikolaev, Daniel Kekere and Eldar Shakirov – for teaching me PLM, which helped me to consider a system, both as whole and in the details. It helped me in writing this thesis and I will always be grateful for that.

And those are just the people who influenced my research directly. There's no way I can ever list all of the amazing people in my life who got me this far. My friends kept me sane, grounded, and laughing throughout everything. My time as an undergraduate with Zakhar Dmitriev, Artem Elin, Sergei Koltunov, Alexey Atanov, Oleg Naumov, Egor Plishchenko at BMSTU was crucial, and I was lucky to fall into their orbits when I was so impressionable. Andrey Semyonkin had been by my side ever since our high school and can't be thanked enough for that. Thanks to all my friends at Skoltech. Thanks to Dmitry Ermachenkov and Andrey Sartison – good luck with your start-up, guys! Thanks to Oleg Sudakov, Alexander Morozov, Sophia Salas, Natalia Glazkova, Alexandra Tyurlikova, Alexey Kovalov, Artem Volokhaty, Nina Mazyavkina and many others for sharing some courses with me and having a nice time with you in between the courses and being a great distraction to my research sometimes. Thanks to all Skoltech students, professors and staff for making the institute the way it is. Thanks to my mother for being nothing but supportive of me in everything I do, even now when we don't get along sometimes.

Finally, thank you to my Anastasia, my once and future closer, your patience with my Skoltech experience has been unending. We made it together. I've been lucky to have your confidence, patience, and love. Without you by my side through the most difficult times of this journey, the accomplishment wouldn't be half as sweet.

Thank you, everyone.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CDF	Concurrent Design Facility
CE	Concurrent Engineering
CEDESK	Concurrent Engineering Data Exchange SKoltech
CEDL	Concurrent Engineering Design Laboratory
CEF	Concurrent Engineering Facility
CIC	Centre d'Ingénierie Concourante (French: *Center for Concurrent Engineering*)
CNES	Centre National d'Etudes Spatiales (French: *National Centre for Space Studies*)
DISC	Le Département d'ingénierie des systèmes complexes (French: *Department of Complex Systems Engineering*)
DLL	Dynamic-Link Library
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V. (German: *German Aerospace Center*)
DSM	Design Structure Matrix
ESA	European Space Agency
ESTEC	European Space Research and TEChnology Cenctre
GUI	Graphical User Interface
HMI	Human Machine Interface
IDM	Integrated Design Model
ISAE-SUPAERO	École nationale supérieure de l'aéronautique et de l'espace (French: *National School of Aeronautics and Space*)
JDBC	Java DataBase Connectivity
JPL	Jet Propulsion Laboratory
MBSE	Model-Based Systems Engineering
MDI	Multiple Document Interface
NASA	National Aeronautics and Space Administration
OCDT	Open Concurrent Design Tool

ODBC	Open DataBase Connectivity
OO	Object Oriented
OMG	Object Management Group
OPD	Object-Process Diagram
OPL	Object-Process Language
OPM	Object-Process Methodology
OpenMBEE	Open Model Based Engineering Environment
PD	Product Development
PDP	Product Development Process
PLM	Product Lifecycle Management
QA	Quality Assurance
R&T	Research and Technology
RFI	Request for Information
S. E.	Societas Europaea (Latin: *European Company*)
SAPPhIRE	State-Action-Part-Phenomenon-Input-oRgan-Effect
Skoltech	Skolkovo Institute of Science and Technology
SysML	Systems Modeling Language
TRL	Technology Readiness Level
UML	Unified Modeling Language
VirSat	Virtual Satellite
VR	Virtual Reality

CHAPTER 1

INTRODUCTION

1.1 Background and context

Innovation and product development are being the cornerstone of structural growth within the market environment nowadays. But for new product development, the stakes are high, the requirements are increasing and there is a new requirement of delivering more rapidly so as to beat the competition to the market.

New product development is a complex endeavor, which can typically be troublesome to handle and difficult to check beforehand what the end result will be. There is a large uncertainty and sudden things happening along the way, that affect the scope and direction of a product development project. Therefore, these projects can often be difficult to arrange, and plans become obsolete shortly after they are created.

Nowadays, in a turbulent market, developing and launching a new product into the market is one of competitive strategies considered by many large and small enterprises. This strategy enables a company to earn larger market penetration than competitors; consequently, achieving a shorter time-to-money period and increasing the rate-of-return. Establishing this strategy demands that all functions within a supply chain – such as marketing, design, procurement, manufacturing, and distribution – to perform as a unique body of a system. The economic success of most firms depends on their ability to identify the needs of customers and quickly create products that meet these needs and can be produced at low cost. Achieving these goals is not solely a marketing problem; it is a product development problem involving all of these functions (Ulrich and Eppinger, 2008). Product development and Production development are two important processes, which are playing critical role in achieving this competitive capability. One of system-wide approaches to facilitate a product realization process is Concurrent Engineering which can be applied owing to being enable to choose the best practice to improve product introduction process, being capable to improve cross functional integration and communication, and being empowered to apply a set of comprehensive methods for design analysis so that designers can select the most optimal design solution which is not only considering the design constraints, but also taking the constraints of production system, logistics and

distribution into account. Hence, it can cover majority of problems in conceptual design phase, which are generated due to lack of empathy between design and manufacturing.

Whereas the center of gravity is in design engineering function (Wheelwright, 1985), meaning that a design must satisfy various and dynamic customer requirements; the competence of manufacturing must be able to produce a designed product rapidly. Product realization process involves both product development and production development processes as two integrated and dependent processes for achievement of efficient development and realization process (Bellgran and Säfsten, 2009). Thereupon, it is essential to manage product realization process, from concept development to manufacturing of the commercial product, efficiently and effectively. The ultimate purpose of the company is achieving high degree of quality in the shortest time and with as lowest cost as possible. Hence, a central area is the collaboration between product developers (i.e. designers) and production developers (i.e. production engineers) in order to generate the fitness between product design and manufacturing competence.

It is believed that the Industry 4.0 concept has potential to solve many product lifecycle issues. Industry 4.0 is the current trend of automation and data exchange in manufacturing technologies. It includes cyber-physical systems, the Internet of things and cloud computing (Hermann et al., 2016).

Industry 4.0 creates what has been called a “digital factory”. Within the modular structured smart factories, cyber-physical systems monitor physical processes, produce a virtual copy of the physical world and make decentralized decisions.

However, the reality still appears too far from Industry 4.0 nowadays. A way for reducing the complexity of systems should be developed in order to bring better understanding of a system to systems engineers. One of the potential ways to achieve that is to fill the gap between parametric and geometric modeling.

The investigation was conducted within the Concurrent Engineering Design Laboratory (CEDL) at Skoltech, the Complex Systems Engineering Department (DISC) at ISAE-SUPAERO, and the R&T Campus of Airbus S. E. This thesis is focused on development of a software prototype that could potentially enhance conceptual design of complex systems.

1.2 Research statement

Based on data analysis of space projects, the data used in initial phases of product development are predominantly behavioral in nature, that is, a large part of this data does not refer to the geometrical

parameters of the system. However, current PLM-systems are based on the geometrical master model concept and thus work best for the detailed design, where the data is mostly geometrical. Thereunder, there is still a gap between parametric and geometric modelling which has to be eventually filled.

This study is focused on applying a model-based methodology for representing interfaces and impacts, likelihood, and risks they possess. That could enhance development of complex system architectures and technology planning by filling the gap between parametric and geometric modeling based on 3D modeling. Since models provide the basis for rigorous management of future technology investments and allow for identification of synergies across multiple technology areas (Knoll and Golkar, 2017), it is proposed to develop a software prototype which should consist of a parametric modeling tool, a geometric modeling tool, and a system modeling methodology embedded in it. The software prototype is supposed to be used for concurrent conceptual design of complex, reliable, or complex and reliable systems such as space systems. The software prototype shall allow representing a 3D model of a system being developed and interfaces between its subsystems in 3D in such way, that it could enhance demonstrator feasibility assessment by representing large amounts of interfaces in 3D and to ensure a successful development of a product. The software prototype is proposed to be validated on a selected use case.

1.3 Research questions

The main goal of the study is to propose an efficient way to analyze a complex system and interfaces in it, including impact, likelihood, and risk they might have on conceptual design stage of a system's lifecycle.

One of the efficient methods of designing a complex system is Concurrent Engineering (CE) – a systematic technique for integrated product development that emphasizes the reaction to customer expectations. It embodies team values of co-operation, trust and sharing in that kind of way that decision-making is by consensus, concerning all perspectives in parallel, from the beginning of the product life cycle (ESA, 2012). In its conventional use, concurrent design is used to lessen development cost and schedule in integrated product development (Di Domizio and Gaudenzi, 2008).

It is proposed to fill the need in a concurrent conceptual design tool by CEDESK developed by Knoll and Golkar (2016). CEDESK brings concurrency to conceptual design and helps to solve the problem of designing complex systems composed of multiple subsystems referring to different disciplines. Conceptual stage of life cycle is one of the most crucial ones. Costs committed on the

conceptual design stage of lifecycle are equal to about 70%, while only 8% are spent as illustrated in Figure 1.1. It is proposed to use CEDESK in the conceptual design stage of lifecycle in order to lower costs being spent on it.

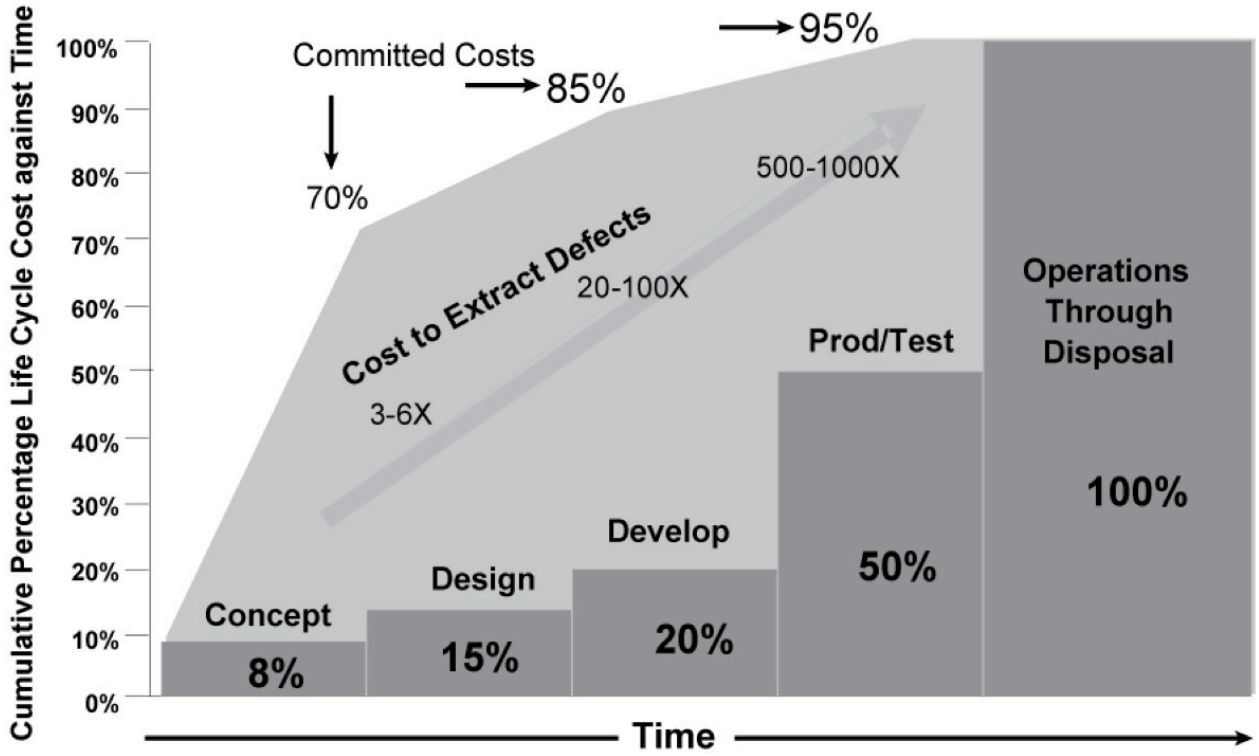

Figure 1.1
Committed life cycle cost versus time, adapted from INCOSE (2010)

It is proposed to use NASA Technology Readiness Levels (TRLs) represented in Figure 1.2 as a method for technology maturity assessment of the demonstrator. TRL is, at its most basic, a description of the performance history of a given system, subsystem, or component relative to a set of levels. The TRL essentially describes the state of the art of a given technology and provides a baseline from which maturity is gauged and advancement defined. Even though the concept of TRL has been around for almost 20 years, it is not well understood and frequently misinterpreted. It is impossible to understand the magnitude and scope of a development program without having a clear understanding of the baseline technological maturity of all elements of the system (NASA, 2007).

The estimation of the technology integration risk proposed by Garg et al. (2017) was studied, adapted and implemented in a software prototype of the demonstrator.

Since the gap between parametric and geometric modelling has to be filled, it was proposed to integrate a solid modeling kernel with MySQL databases used in CEDESK in a software prototype which has a potential to enhance demonstrator feasibility assessment by representing DSM matrices

in 3D. For this purpose, it is proposed to use C3D Geometric Kernel (C3D Labs, 2017) and the principles of the development of geometric modeling systems proposed by Golovanov (2014) in order to address more efficient 2D and 3D modelling in developing the software.

It is proposed to use Nano-satellite design as a use case to concretely demonstrate the applicability of the principles and solutions developed during the project.

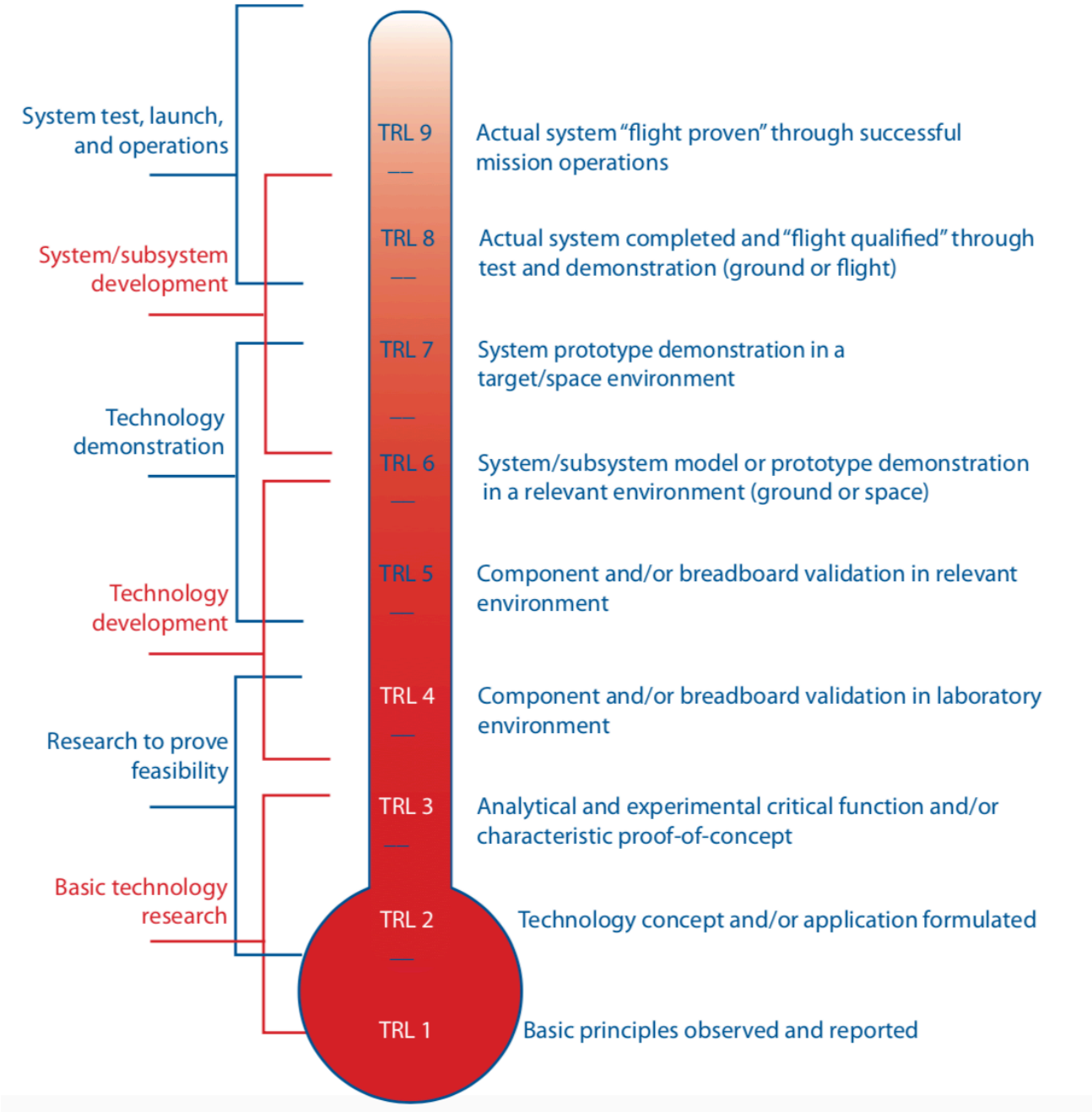

Figure 1.2
Technology Readiness Levels, adapted from NASA (2007)

1.4 Structure of the Thesis

The outline of the thesis is as follows:

- In Chapter 2, the *literature review* for the study and research questions will be presented. The scope of the literature review, including the research areas covered, is listed.
- In Chapter 3, the approach and *methods* to answer the main questions of the study will be introduced.
- In Chapter 4, the *results* achieved during the study will be covered and the demonstration of the application performance will be given.
- In Chapter 5, the results will be *analyzed* and *discussed* and the *conclusions* will be given, along with suggestions for *further research*.

CHAPTER 2

LITERATURE REVIEW

A literature review was carried out in order to explain the context and reasoning behind choosing the papers covered in it.

The literature review is structured as follows:

- Section 1 of Chapter 2 will introduce literature sources relevant to understanding the problem definition and research questions. It consists of the Request for Information which is basically data from the survey of respondents from the system engineering community, as well as the theoretical framework of product design, product realization, and development process.
- Section 2 of Chapter 2 will introduce several approaches to represent links and interconnection in systems. The description of Object Process Methodology and DSM^{3D} are presented and proposed to be implemented in a software prototype.
- Section 3 of Chapter 2 will review various applications with the purpose similar to the ones covered in this study.

2.1 Theoretical framework

2.1.1 Request for Information

In 2010 the School of Systems and Enterprises of the Stevens Institute of Technology published a report with the complete set of textual responses to the questions contained in the responses to the SysML Request for Information (RFI) (Cloutier and Bone, 2010). The report includes over 50 questions (including open-ended responses) that were provided by a sample of respondents from the system engineering community. The study was conducted in response to the OMG SysML Request for Information in an attempt to develop the SysML standard. The RFI is relevant to this research since it provides many viewpoints on Systems Engineering itself. Many of the responses covered in

the RFI, if taken into consideration, could be used to better understand the state of the art of systems modeling.

The RFI responses were submitted via an on-line survey that was accessible from the OMG SysML web site. The intent of the RFI is to assist with guiding the roadmap for future evolution of SysML, by understanding, what is operating well, the issues, proposed solutions, and additional capabilities that are desired of the language. The RFI has two parts, where part I includes 22 questions related directly to the language, and part II includes 38 additional questions related to how SysML is used with model-based systems engineering (MBSE) methods, tools, training, and metrics.

Although there is a number of MBSE approaches and methods developed, many people are familiar with the company specific ones as illustrated in Figure 2.1.

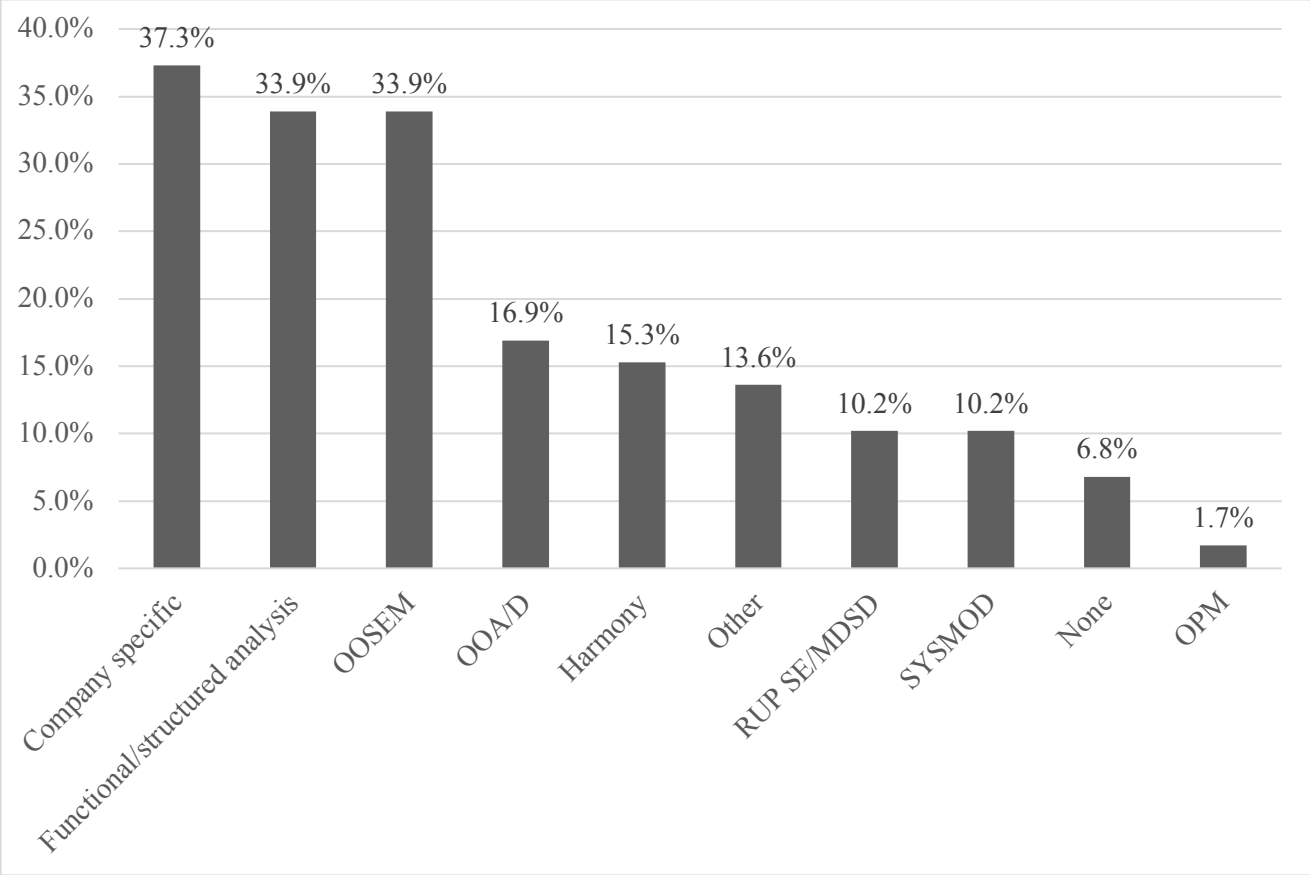

Figure 2.1
What modeling approach/method did you use? (Note: The following methods are mostly identified in the Survey of MBSE Methodologies). Based on data from Cloutier and Bone (2010)

Figure 2.2 represents the primary purposes of the model in systems engineering according to the report and shows the significance of the model.

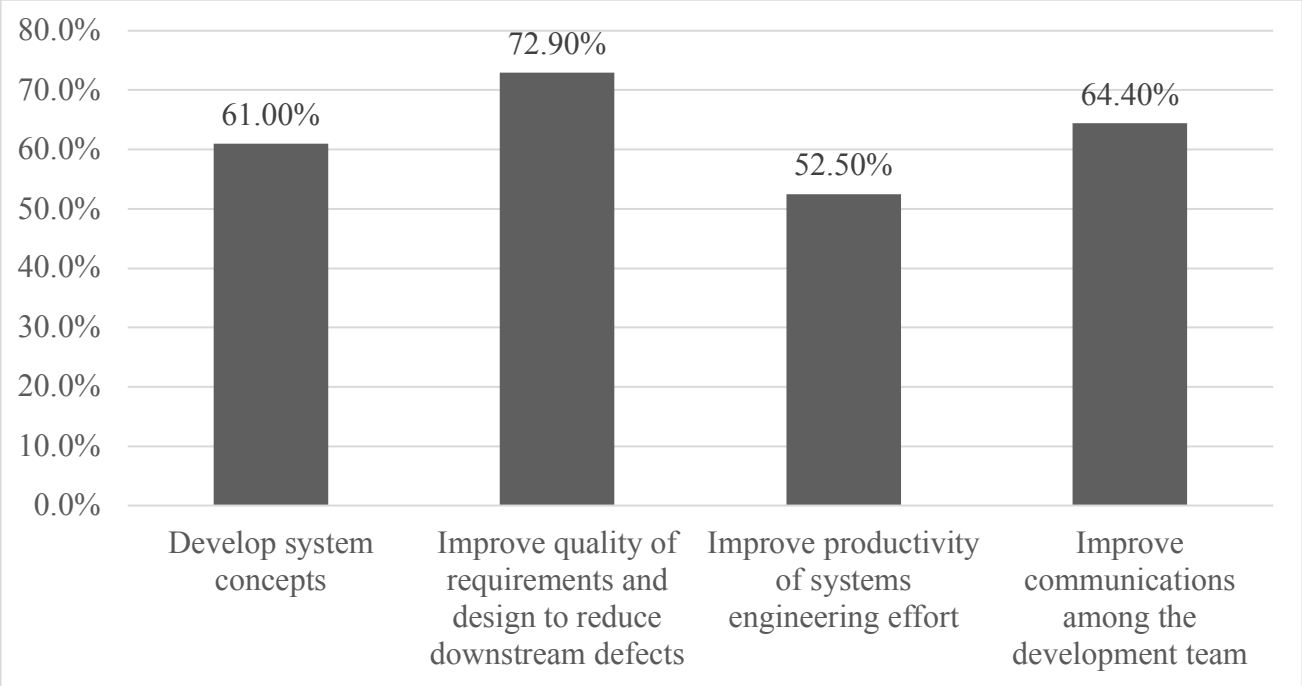

Figure 2.2
What was the primary purpose of the model? Based on data from Cloutier and Bone (2010)

A chart diagram provided in Figure 2.3 shows that SysML is mostly applied to the Space, Aircraft, and Defense types of systems, i.e. the most complex types of systems.

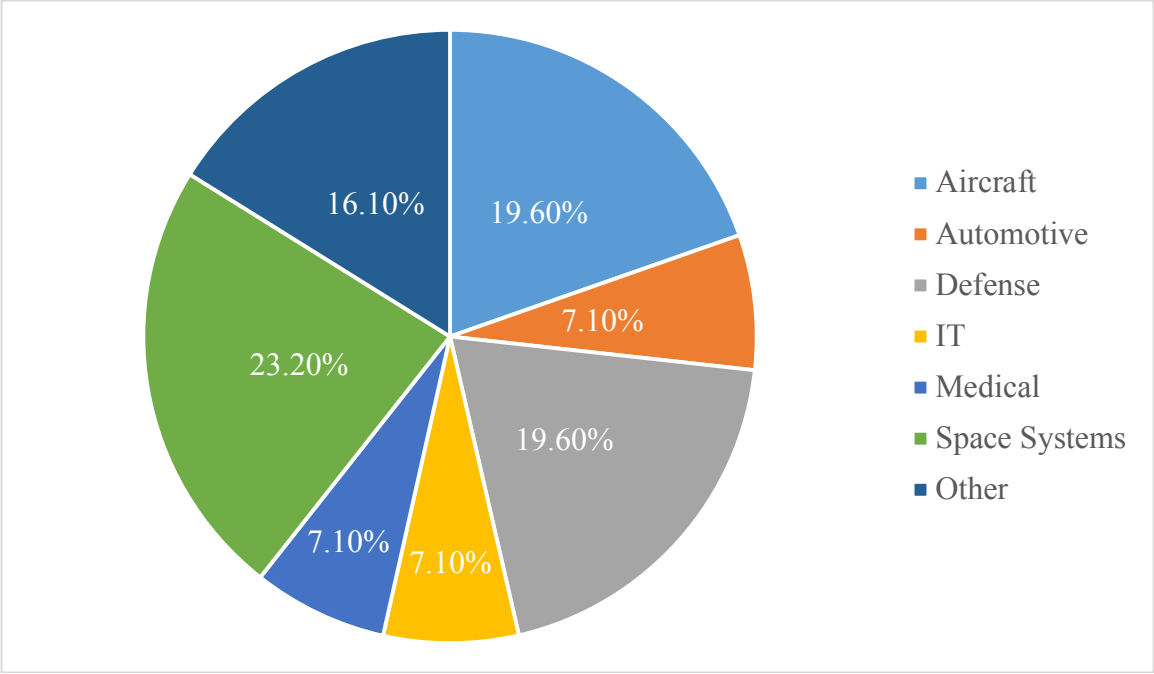

Figure 2.3
What type of system was SysML applied to? Based on data from Cloutier and Bone (2010)

As presented in Figure 2.4, the most popular modeling tools according to the data are IBM Rhapsody (45.0%), NoMagic (35.0%), and Sparx Systems Enterprise Architect (18.3%).

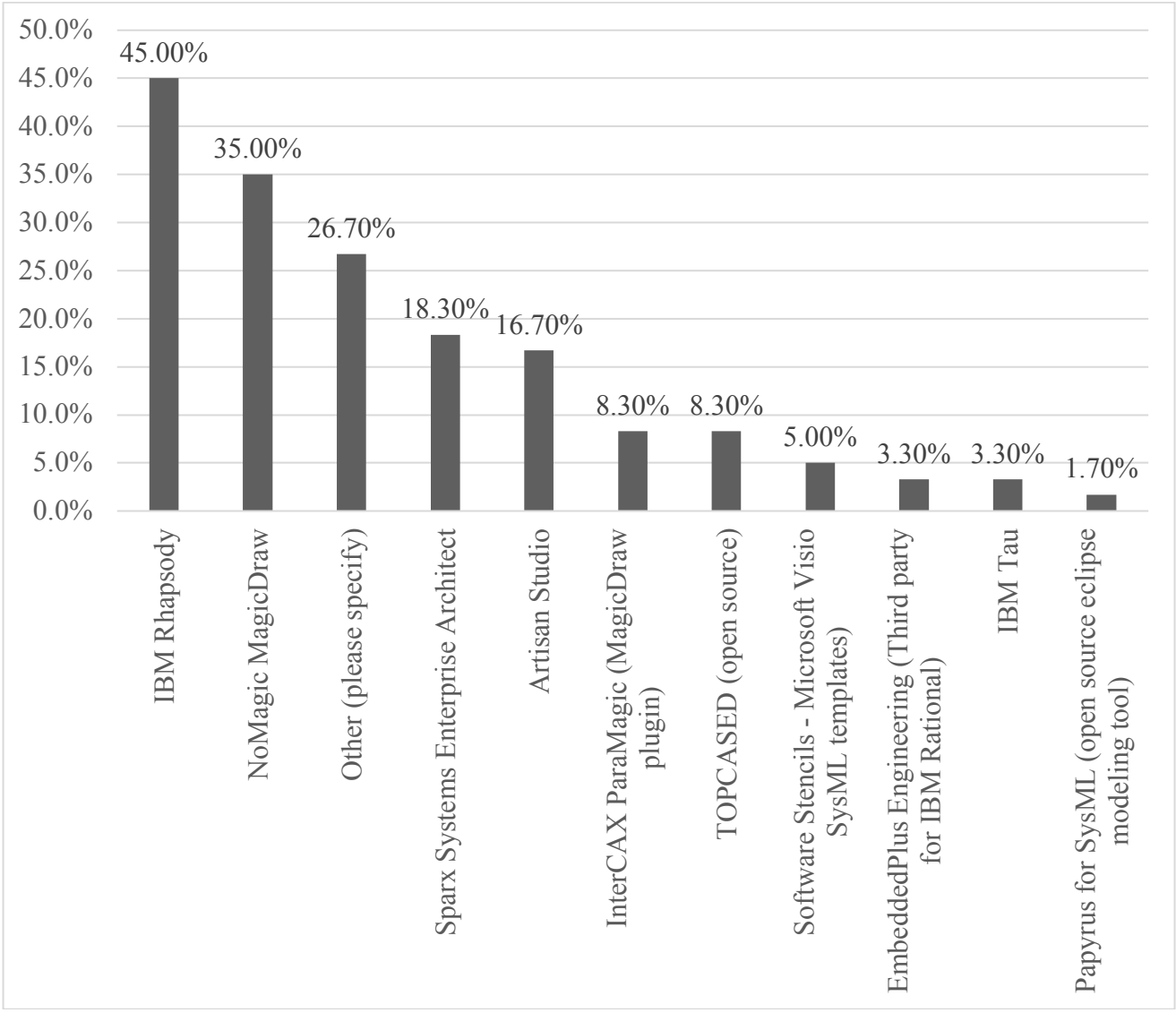

Figure 2.4
What Modeling tools were used on the project? Based on data from Cloutier and Bone (2010)

As shown in Table 2.1 and Figure 2.5, people are satisfied the most with InterCAX ParaMagic (MagicDraw plugin) having 4.21 out of 5.0 score, although it is not the most popular modeling tool as could be seen in Figure 2.4. At the same time, IBM Rhapsody has a low score of 3.57 out of 5.0, while being the most popular modeling tool according to Figure 2.4. This implies that IBM Rhapsody provides some crucial for systems engineering feature, while being not as user-friendly as other systems modeling tool.

Table 2.1
Overall value average of all diagrams. Based on data from Cloutier and Bone (2010)

	Artisan Studio	IBM Rhapsody	InterCAX ParaMagic (MagicDraw Plugin)	NoMagic MagicDraw	Sparx Systems Enterprise Architect	Rating Average
Overall value average of all diagrams	3.88	3.57	4.21	3.95	3.87	3.82

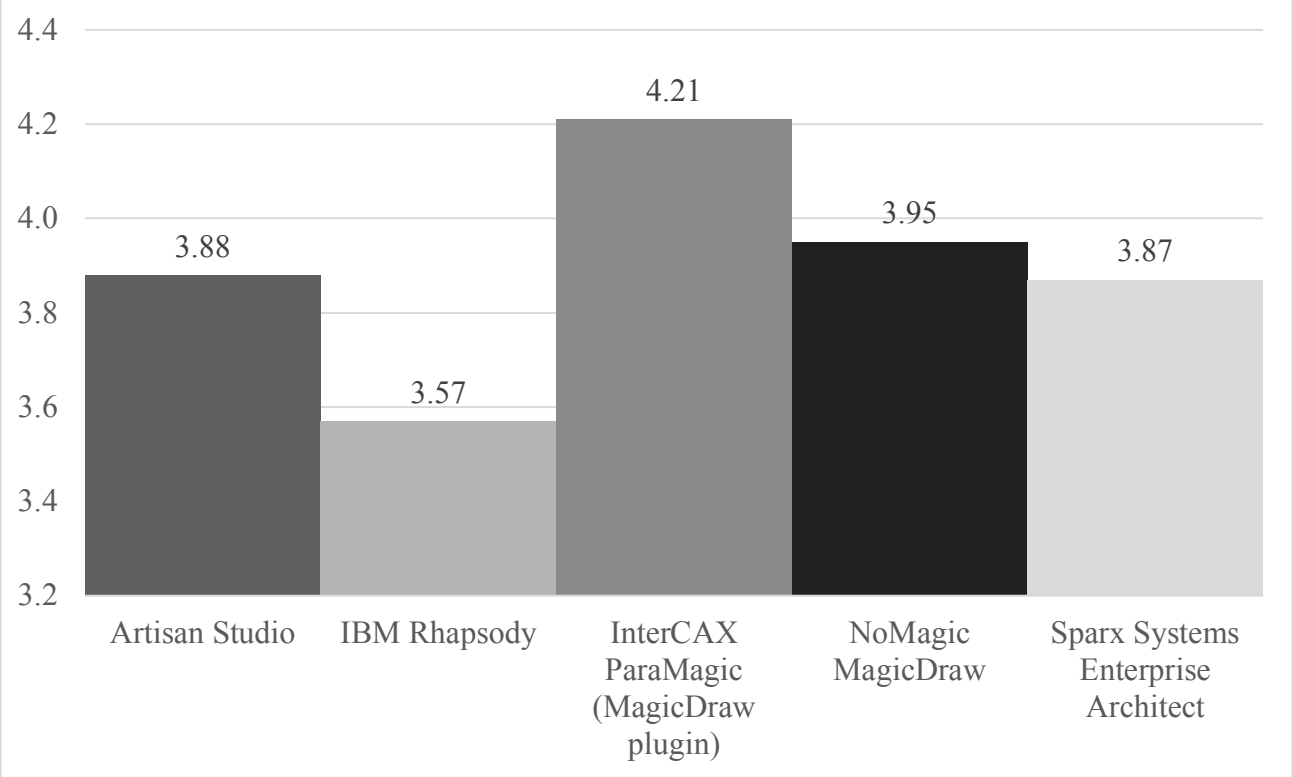

Figure 2.5
How satisfied are you with primary SysML tool used on this project? Based on data from Cloutier and Bone (2010)

From the RFI some key modeling tools can be identified. In this part of the literature review a few of them will be analyzed:

- IBM Rhapsody
- NoMagic MagicDraw
- Sparx Systems Enterprise Architect

They are selected due to their high popularity among systems engineers shown in Figure 2.4.

All of them are based on UML and applied mostly to software development, yet some lessons can be learned by analyzing the tools.

IBM Rational Rhapsody, is a modeling environment based on UML, is a visual development environment for systems engineers and software developers creating real-time or embedded systems and software. Rational Rhapsody uses graphical models to generate software applications in various languages, including C, C++, Ada, Java and C#. The main IBM Rational Rhapsody interface components are illustrated in Figure 2.6.

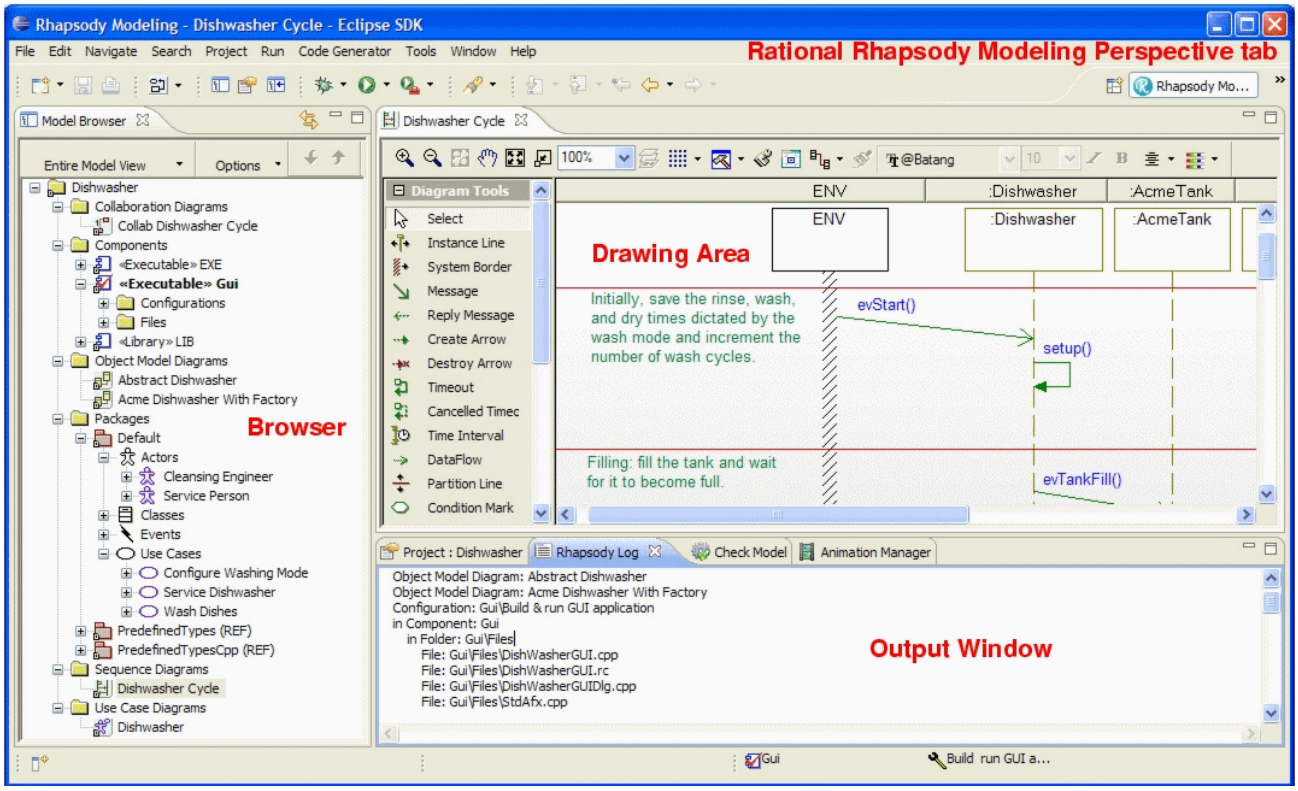

Figure 2.6
The main IBM Rational Rhapsody interface components, which include Browser (Model Browser tab in Eclipse), Diagram Drawing Area, Output Window and Features Window, adapted from IBM Knowledge Center (2016)

MagicDraw is a visual UML, SysML, BPMN, and UPDM modeling tool with team collaboration support. Designed for business analysts, software analysts, programmers, and QA engineers, this dynamic and versatile development tool facilitates analysis and design of object oriented (OO) systems and databases. It provides the code engineering mechanism (with full round-trip support for J2EE, C#, C++, CORBA IDL programming languages, .NET, XML Schema, WSDL),

additionally with database schema modeling, DDL generation and reverse engineering facilities (Davis, 2010). Figure 2.7 represents the user interface of the MagicDraw modeling tool.

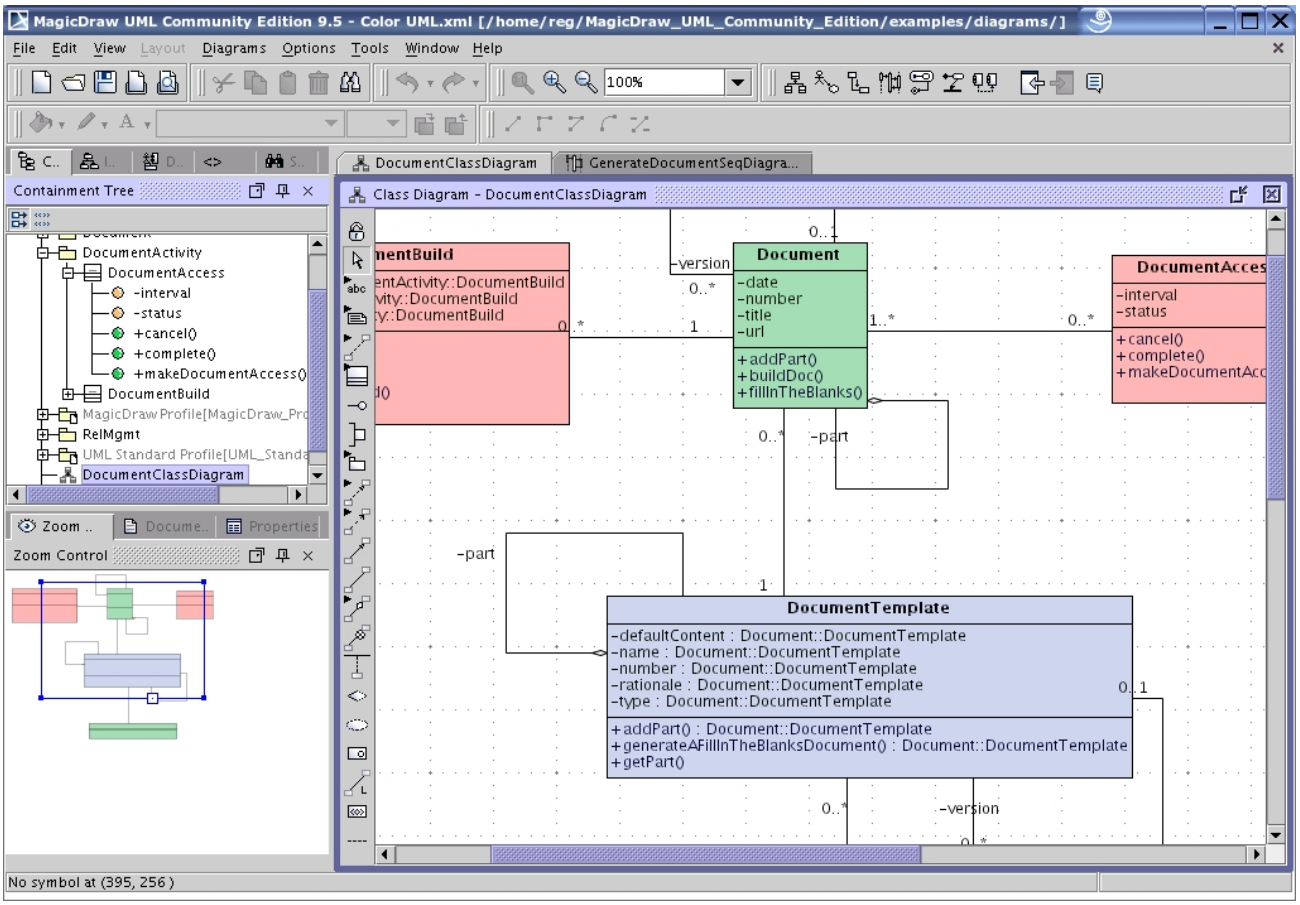

Figure 2.7
User interface of the MagicDraw modeling tool, adapted from Charney (2005)

Sparx Systems Enterprise Architect is a visual modeling and design tool based on the OMG UML. The platform supports: the design and construction of software systems; modeling business processes; and modeling industry-based domains. It is used by businesses and organizations to not only model the architecture of their systems, but to process the implementation of these models across the full application development lifecycle (Sparx Systems, 2018). The user interface of Sparx Systems Enterprise Architect is provided in Figure 2.8.

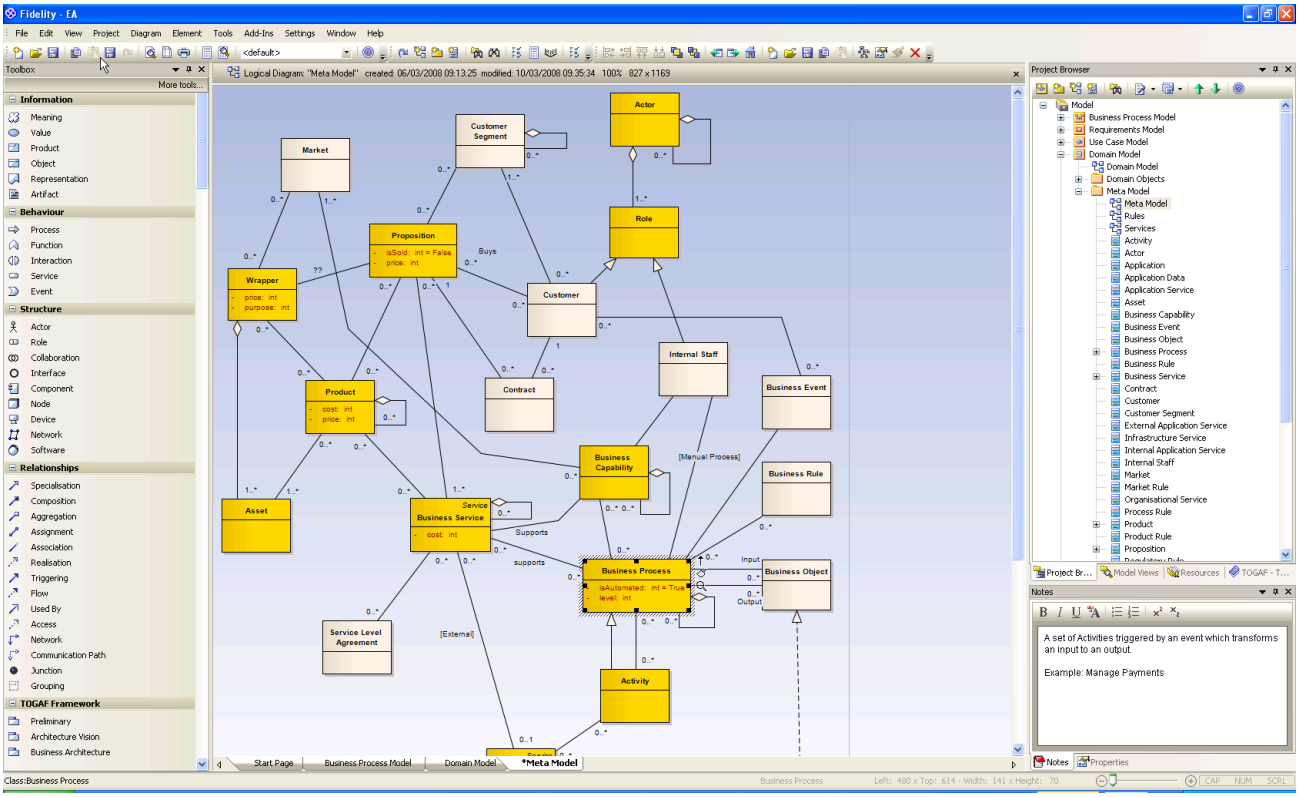

Figure 2.8
User interface of Sparx Systems Enterprise Architect, adapted from Enterprise Architecture (2009)

All those modeling tools are being used for development and quality assurance of large software projects. They are considered to be highly collaborative and agile, yet many users consider its user interface ‘clunky’ (TrustRadius, 2014). The interface links between subsystems intersect each other on an often basis, which makes the process of analyzing them complex. This implies that although a tool can have a great functionality, its GUI can spoil the whole experience and thus affect the result. As seen in Figures 2.6, 2.7, and 2.8, the UML system modeling tools have a similar user interface. The interface is not interactive for new users, which means that they have to spend time, which is crucial in conceptual design stage, on learning.

It can also be noticed, that modeling tools are crucial for industries developing complex and reliable systems, such as the space industry.

2.1.2 Production Design and Development Process

Product Development (PD) is a transformation of customers' needs/desires or market opportunities into what can be sold in available markets for a logical price and reasonable production cost; "he set of activities beginning with the perception of the market opportunity and ending in the production, sale, and delivery of the product (Ulrich and Eppinger, 2008). Product Development Process (PDP) is a sequence of steps or activities which an enterprise employs to conceive, design, and commercialize a product (Ulrich and Eppinger, 2008).

Many of the steps within a PDP are intellectual and organizational rather than physical. The conclusion of the product development process is the product launch meaning; when a product becomes available for distribution and procurement in a marketplace (Ulrich and Eppinger, 2008).

There are two types of product development process – stage-gate and spiral processes. Each one of them constitutes the generic product development phases, but they differ in the arrangement of the sequence of phases. The stage-gate product development process is comprised of distinct stages or phases as well as a review or gate at the end of each phase in order to evaluate whether the previous phase is successfully completed. If the review fulfills the requested conditions the project proceeds to the next phase, otherwise the project will iterate through a former phase. Sometimes this iteration can be difficult and costly (Unger and Eppinger, 2009). The spiral product development process includes several planned iterations that span various phases of product development process. It is mainly implemented by software industry (Unger and Eppinger, 2009).

The generic product development process consists of six phases represented in Table 2.2 which based on their chronological sequence are as follows: planning, concept development, system-level design, detail design, testing and refinement, and production ramp-up.

Table 2.2
Phases of the generic product development process, adapted from Ulrich and Eppinger (2008)

Phase	Description
Planning	This phase includes three overall dimensions. The basic approach to markets and products with respect to the competitor's activities should be determined. This approach is called corporate strategy. Hence the assessment of technology development and the evaluation of marketing objectives should be accomplished in this phase. The output of this phase is named as a mission statement.

Table 2.2 continuation
Phases of the generic product development process, adapted from Ulrich and Eppinger (2008)

Phase	Description
Concept Development	A concept is a description of the form, function, and features of the product which are accompanied by a set of specification, an analysis of competitive products, and justification of project. This phase needs more coordination among different functions.
System-level Design	This phase pertains a definition of the product architecture and the decomposition of the product into subsystems. The architecture is usually presented as a geometric layout. The final assembly scheme for production system and a preliminary process flow diagram for the final assembly process are other outputs of system-level design phase.
Detailed Design	Two important issues are addressed in this phase; the production cost and the robust performance of product/process design. In addition, the complete specification of geometric value, materials metrics, and tolerances of all of the unique parts in the products as well as the identification of the all of the parts that should be provided by supplier are determined. The outputs of this phase are process plan for fabrication and assembly, tooling design, control documentation for the product.
Testing and Refinement	In this phase, multiple preproduction prototypes are constructed and evaluated. The various types of prototypes constructed through different phases of product development process. There are different kinds of prototypes to identify: whether the product satisfies the customer needs, whether it is working as designed, as well as to test product's reliability and performance in order to figure out necessary engineering changes.
Production ramp-up	In the production ramp-up phase intended production system will be implemented in order to train workforces and identify any remaining flaws and the solution to resolve the problems.

2.1.3 Concurrent engineering

Concurrent engineering (CE) is a systematic approach to integrated product development that emphasizes the response to customer expectations. It embodies team values of co-operation, trust and sharing in such a manner that decision-making is by consensus, involving all perspectives in parallel, from the beginning of the product lifecycle (ESA, 2012).

Essentially, CE provides a collaborative, cooperative, collective and simultaneous engineering working environment. The concurrent engineering approach is based on five key elements:

- a process
- a multidisciplinary team
- an integrated design model
- a facility
- a software infrastructure

In its traditional use, concurrent design is used to reduce development cost and schedule in integrated product development (Di Dominzo, 2008). Applying concurrent engineering to a product lifecycle results in a time compression comparing to classical sequential (waterfall) model as sketched in Figure 2.9, which results in a faster start of production

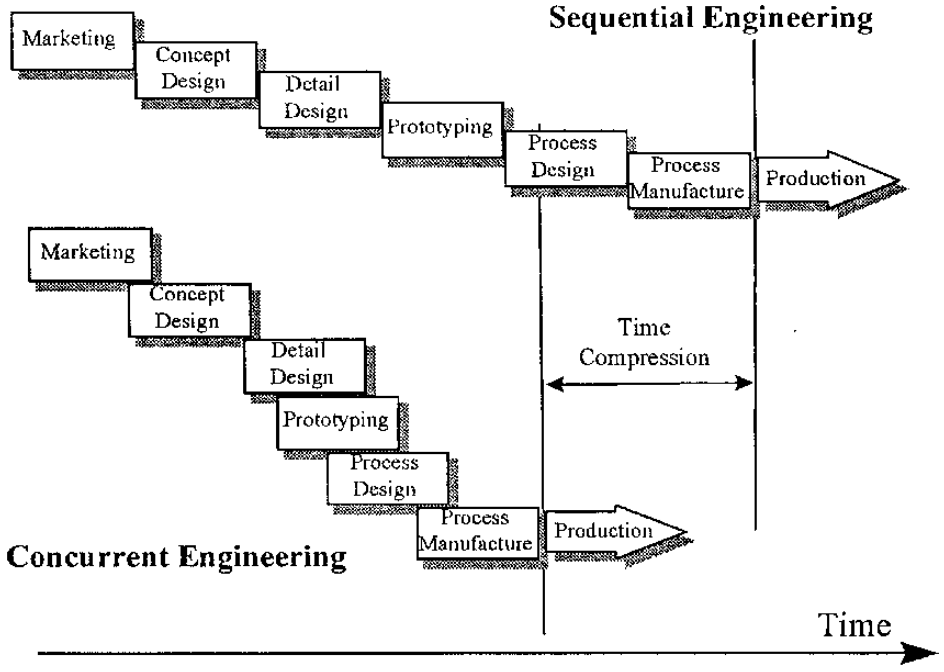

Figure 2.9
Sequential Engineering vs Concurrent Engineering, adapted from Yazdani (1999)

2.1.4 Concurrent engineering for space systems

In the space sector, it is defined that concurrent engineering at the conceptual stage is carried out in a very different way from the conventional design of manufacturing interface. Concurrent engineering is the simultaneous and integrated engineering of all design, manufacturing, and operational aspects of a project from the conceptual formulation of the project through project completion. It is a team-engineering process in which all of the specialists who normally get involved in a project combine into a multi-disciplinary task force to carry out a project. They work together, trading ideas, and ensuring what they do early in the project (like major design decisions or changes) will not adversely affect what they do later (like "manufacturing in" quality or supporting flight operations). All disciplines are addressed simultaneously.

The use of concurrent engineering practices, coupled with the application of current state-of-the-art three-dimensional solid modeling and analysis tools, has proven to dramatically reduce new project development times while maintaining or further improving quality, reliability, and safety (NASA JPL, 2001).

For example, NASA Team X, a cross-functional multidisciplinary team of engineers at NASA JPL, utilizes concurrent engineering methodologies to complete rapid design, analysis and evaluation of mission concept designs. This advanced design team of experienced flight-project engineers is co-located in the Project Design Center to complete architecture, mission, and instrument design studies in real time (NASA JPL, 2015).

The Concurrent Design Facility (CDF), the European Space Agency main assessment center for future space missions and industrial review, uses concurrent engineering methodology to perform effective, fast and cheap space mission studies (ESA, 2014b).

2.2 System Modeling

Some of the UML based system modeling tools has already been covered above in Section 1.1 of Chapter 2. In this section other tools and methodologies that find their application in conceptual design phase of lifecycle will be presented.

2.2.1 OPM

The unnecessary complexity and software orientation of UML calls for a simpler, formal, generic paradigm for systems development. Object Process Methodology (OPM) proposed by Dori (2011) satisfies the essential need for a universal modeling, engineering, and lifecycle support approach under condition of the inherent complexity and interdisciplinary nature of systems. OPM has a potential to be integrated into a next generation MBSE tool since it provides a complete overview a system with objects, processes, and connections it has.

OPM advocates the integration of a system's structure and behavior is a single, graphic and textual model. OPM is used in companies such as Airbus for the roadmap creation process (Roussel et al., 2017), meaning that it is able to help grasping systems as complex as airplanes.

In an essence, OPM is a modeling method describing which design activities to perform, what engineering artifacts to produce, and how they are denoted. Unification of function, structure and behavior in a single model, as well as bi-modal expression of the model via intuitive, yet formal graphics and equivalent natural language makes OPM a good candidate to any future MBSE solution, being the reason the section about OPM is included in this thesis.

OPM is a comprehensive patented systems modeling, engineering, and lifecycle support paradigm (Dori, 2003). The main features of OPM are:

- unification of function, structure and behavior in a single model;
- bi-modal expression of the model via intuitive, yet formal graphics and equivalent natural language.

Figure 2.10 illustrates a simple example of a system represented with OPM for Foundation Constructing of a house. Here, Constructor, who is physical (has a shade) performs Foundation Constructing, physical Raw Materials include Steel and Concrete, steel is consumed by the Steel Frame Forming process and by having Plan as an environment for that, etc.

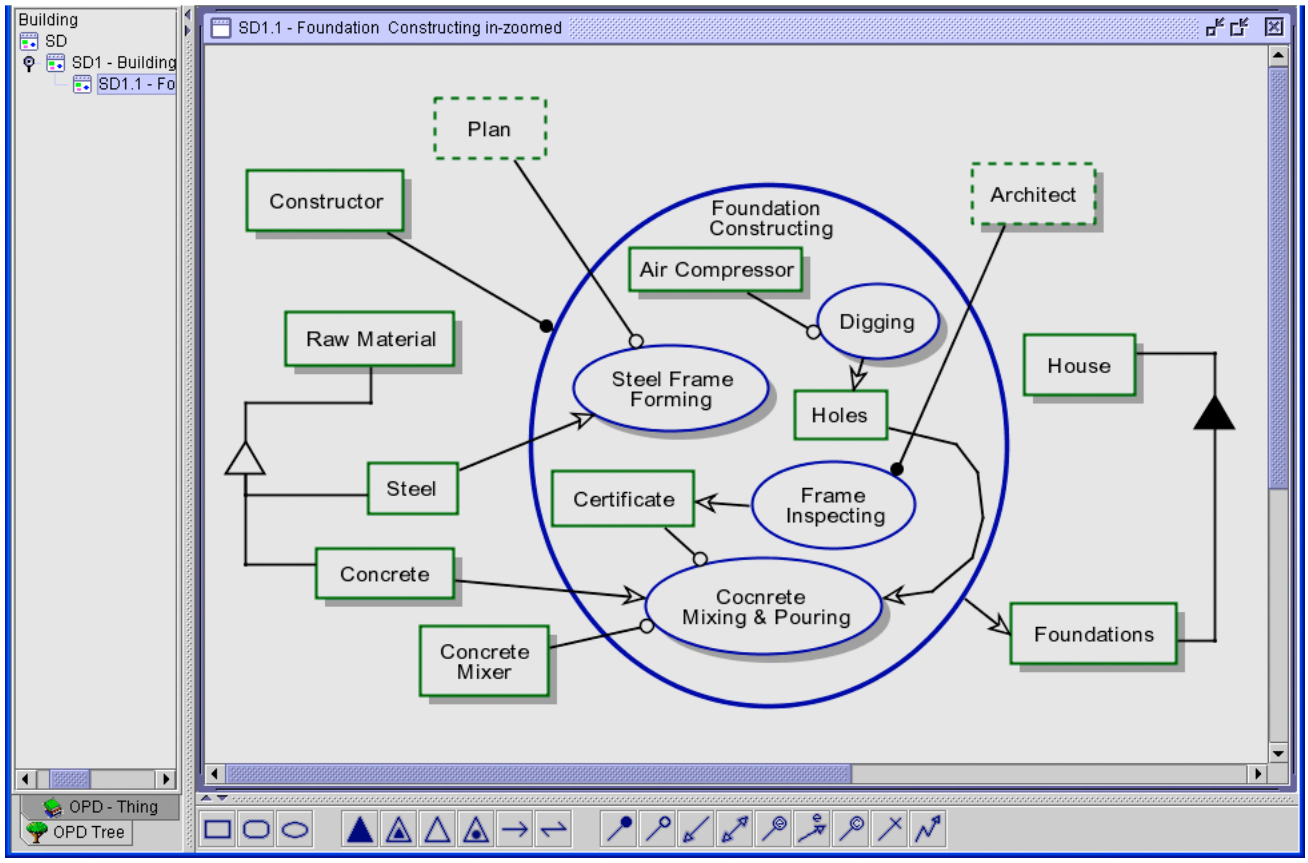

Figure 2.10
An OPM diagram example. Adapted from Dori (2003)

2.2.2 Technology integration risks

Risk estimation is a key interest for product development and technology integration programs. There are many decision assist tools that help project managers discover and mitigate risks in a project, but few explicitly take into account the outcomes of architecture on risk. A novel risk estimation framework was proposed by Garg et al. (2017) that consists from considerations of the system architecture. By way of starting with conventional project management literature, risk is described as a mixture of likelihood and impact.

Technology Readiness Levels (TRLs) proposed by NASA (2007) are used as the measure for likelihood, and for the reason that change propagates via interfaces, measures that relate to connectivity are used to estimate impact. This framework became implemented with an industry example and the data was visualized in different formats to aid in analysis.

The general technique that was applied by Garg et al. (2017) is illustrated and summarized in Figure 2.11. In this method the technology integration risk of each component i is estimated using a common risk metric – the product of likelihood and impact as seen in Equation 1 (Project Management Institute, 2008).

$$Risk_i = L_i \cdot I_i \qquad (1)$$

L_i is the likelihood that the element technology needs an alternation to fulfil its function. This is estimated through the usage of TRLs, that have been proven to be proper estimators of uncertainty in the technology integration process (United States Government Accountability Office, 2007).

I_i is the severity of impact if the element is forced to alternate. The general architecture and the element interfaces must be examined specifically to estimate the impact through the context of change propagation.

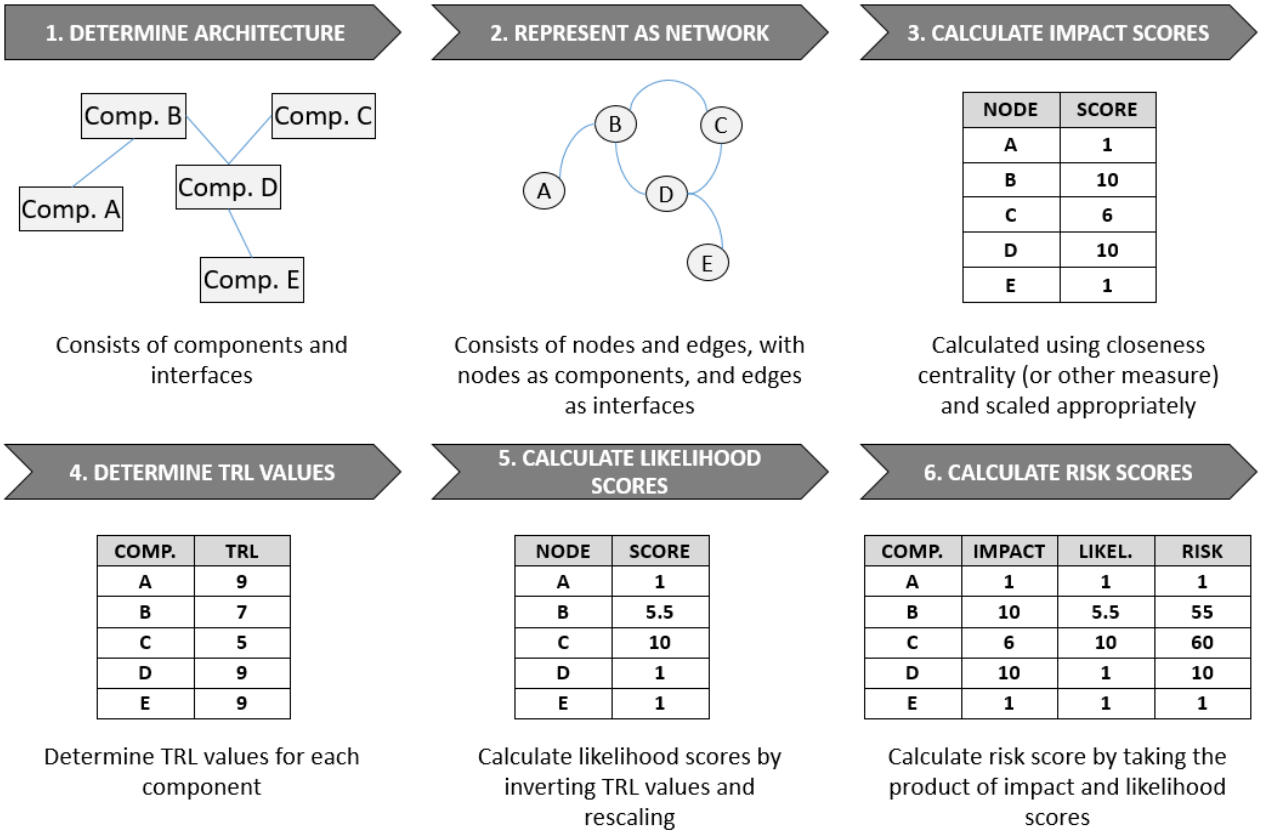

Figure 2.11
Summary of risk calculation method, adapted from Garg et al. (2017)

The technique was implemented with an industry use-case within Analog Devices Inc. Data obtained from Analog Devices have been used to build a view of the system architecture and develop a network representation of the system as illustrated in steps (1) and (2) from Figure 2.9. Assoon as

all of the data has been gathered, impact and likelihood vectors had been calculated as in steps (3), (4) and (5) of Figure 2.11 to obtain final risk scores (step 6). The inputs and final risk calculation are shown in Figure 2.12 with bars in each cell to symbolize the magnitudes.

		Likelihood		Impact		Risk
Subsystem	**Component**	**TRL**	**Risk Input**	**Degree**	**Risk Input**	
Package	Die attach	8	2.5	13	10.0	25
	Leadframe	7	4	5	2.8	11.2
	Wirebond	9	1	6	3.7	3.7
	Plastic Mold	9	1	13	10.0	10
ASIC for Sensor 1	Sensor 1 Analog Front End	7	4	4	1.9	7.6
	Sensor 1 Analog to Digital Converter	3	10	5	2.8	28
	Sensor 1 Calibration	9	1	5	2.8	2.8
	Sensor 1 Processor	9	1	5	2.8	2.8
ASIC for Sensor 2	Sensor 2 Analog Front End	9	1	4	1.9	1.9
	Sensor 2 Analog to Digital Converter	3	10	5	2.8	28
	Sensor 2 Calibration	9	1	3	1.0	1
	Sensor 2 Processor	9	1	5	2.8	2.8
ASIC	ASIC Input/Output	9	1	6	3.7	3.7
	ASIC Non-volatile Memory	6	5.5	7	4.6	25.3
	ASIC Regulator	9	1	7	4.6	4.6
	ASIC Oscillator	9	1	10	7.3	7.3
	ASIC Analog Front End	9	1	4	1.9	1.9
	ASIC Analog to Digital Converter	9	1	6	3.7	3.7
Sensor 1	Sensor 1 Design/layout	5	7	7	4.6	32.2
Sensor 2	Sensor 2 Design/layout	5	7	6	3.7	25.9

Figure 2.12
Vector representation of the components and their scores, adapted from Garg et al. (2017)

The data is graphed on a scatter plot in Figure 2.13, with the two-axis corresponding to likelihood and severity to better visualize the risk scores.

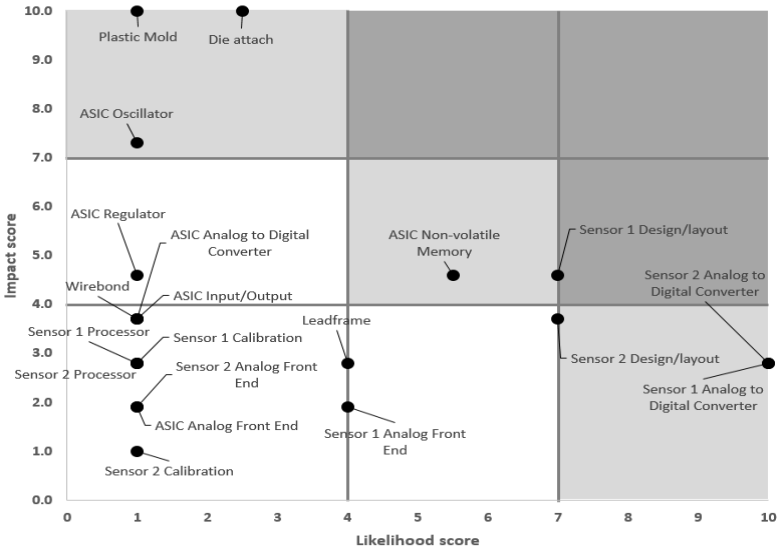

Figure 2.13
Two-axis view of likelihood and impact, adapted from Garg et al. (2017)

In order to preserve information about interfaces, the risk score information was combined with a Design Structure Matrix (DSM) view of the system (Eppinger and Browning, 2012). The

Design Structure Matrix (DSM) is a data exchange model as illustrated in the example in Figure 2.14. In DSMs information flows are easier to capture than work flows, and inputs are simplier to capture than outputs (de Weck, 2012).

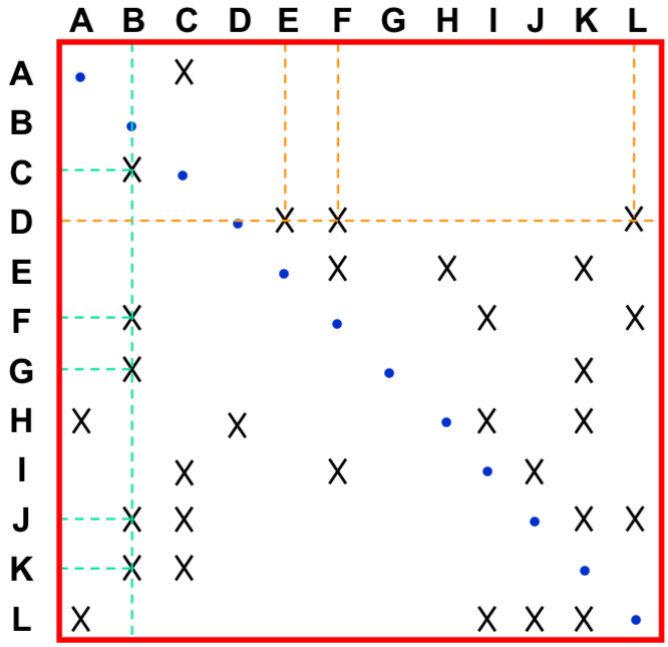

Figure 2.14
An example of a DSM matrix of interfaces. Interpretation: task D requires information from tasks E, F, and L; task B transfers information to tasks C, F, G, J, and K, adapted from de Weck (2012)

In order to use the DSM view with the proposed technique, each off-diagonal mark inside the matrix is selected to represent a risk score composed of the two interfacing components. The calculation is performed according to Equation 2:

$$Interface\ risk_i = \max(L_i, L_j) \cdot \max(I_i, I_j) \qquad (2)$$

Where L_i and L_j represent the likelihood scores for the two interfacing components; I_i and I_j represent the impact scores for each element(Garg et al., 2017). Figure 2.15 allows to see the results of this analysis. The component-level risk calculations are left as a vector in the "risk" column as an additional reference.

Subsystem	COMPONENT	RISK	TRL	8	7	9	9	7	3	9	9	9	3	9	9	9	6	9	9	9	9	5	5
Package	Die attach	25	8		40	25	25	40	100			25	100					25	25	25	25	70	70
	Leadframe	11	7	40		15	40															32	26
	Wirebond	4	9	25	15		10									4						32	26
	Plastic Mold	10	9	25	40	10		40	100			10	100					10	10	10	10	70	70
ASIC for Sensor 1	Sensor 1 Analog Front End	8	7	40			40		28													32	
	Sensor 1 Analog to Digital Converter	28	3	100			100	28		28									73				
	Sensor 1 Calibration	3	9						28		3						25				4	32	
	Sensor 1 Processor	3	9							3						4	25		7			32	
ASIC for Sensor 2	Sensor 2 Analog Front End	2	9	25			10						28										26
	Sensor 2 Analog to Digital Converter	28	3	100			100					28		28					73				
	Sensor 2 Calibration	1	9										28		3		25						
	Sensor 2 Processor	3	9											3		4	25		7				26
ASIC	Input/Output	4	9			4					4				4		25	5	7				
	Non-volatile Memory	25	6							25	25			25	25	25		25	40				
	Regulator	5	9	25			10									5	25		7	5	5		
	Oscillator	7	9	25			10		73		7		73		7	7	40	7			7		
	Analog Front End	2	9	25			10											5			4		
	Analog to Digital Converter	4	9	25			10			4								5	7	4			
Sensor 1	Sensor 1 Design/layout	32	5	70	32	32	70	32		32	32												
Sensor 2	Sensor 2 Design/layout	26	5	70	26	26	70					26			26								

Figure 2.15
DSM view of the system risk, adapted from Garg et al. (2017)

2.2.3 DSM^{V} and DSM^{3D}

Many companies that struggle with product variety and configuration management issues turn to a module-based design approach. Although this approach is well-known to be efficient for managing variety of a product family, current methods do not enable designers to handle both modularity and variety within a product family. The Design Structure Matrix (DSM) has been widely used to identify modules within a product, but its use to identify modules across a family of products has been limited. In this context two more tools were proposed by Alizon et al. (2007) based on an extension of the basic DSM to manage variety of an entire product family. The Variety Design Structure Matrix, DSM^{V}, handles variety of the product family and 3D Design Structure Matrix, DSM^{3D}, enables visual analysis of interfaces across the entire product family. These two tools, combined into a single approach, enable analysis of the product family at many levels — family product, module, and interfaces — to better specify modules and interfaces across all of the products in the family. A case study involving a family of Kodak single-use cameras is used to demonstrate the application of these new DSMs and accompanying cross-module and cross-interface analyses. This approach can be applied during detailed studies as well as in the early stages of the design process.

Kodak, a manufacturer of photographic equipment and systems, successfully led the market of single-use cameras by producing a product family that addressed multiple market segments. Kodak offered a wide range of products that included combinations of key features such as waterproof, panoramic format, flash, and high definition. Product platforming enables companies to cut costs while offering tailored products, yet it also brings the challenge of managing variety within the family.

The DSM application proposed by Alizon et al. (2007) demonstrates two DSM techniques to identify modules across a product family: the DSM variety (DSM^{V}) and the three-dimensional DSM (DSM^{3D}). Using these two DSM techniques, one is able to study families of products, modules, and interfaces.

The model works in two main stages using two original DSM techniques: DSM^{V} and DSM^{3D}. The DSM^{V}, shown in Figure 2.16, uses a static, binary, product architecture DSM to specify the modules in each product containing components that have either common, variant, or unique interfaces. The process is repeated for all the products of the family and then all these DSM^{V}-s are stacked to obtain the DSM^{3D}. The DSM^{3D}, shown in Figure 2.17, is a three-dimensional DSM gathering all products of the family and highlighting the differences.

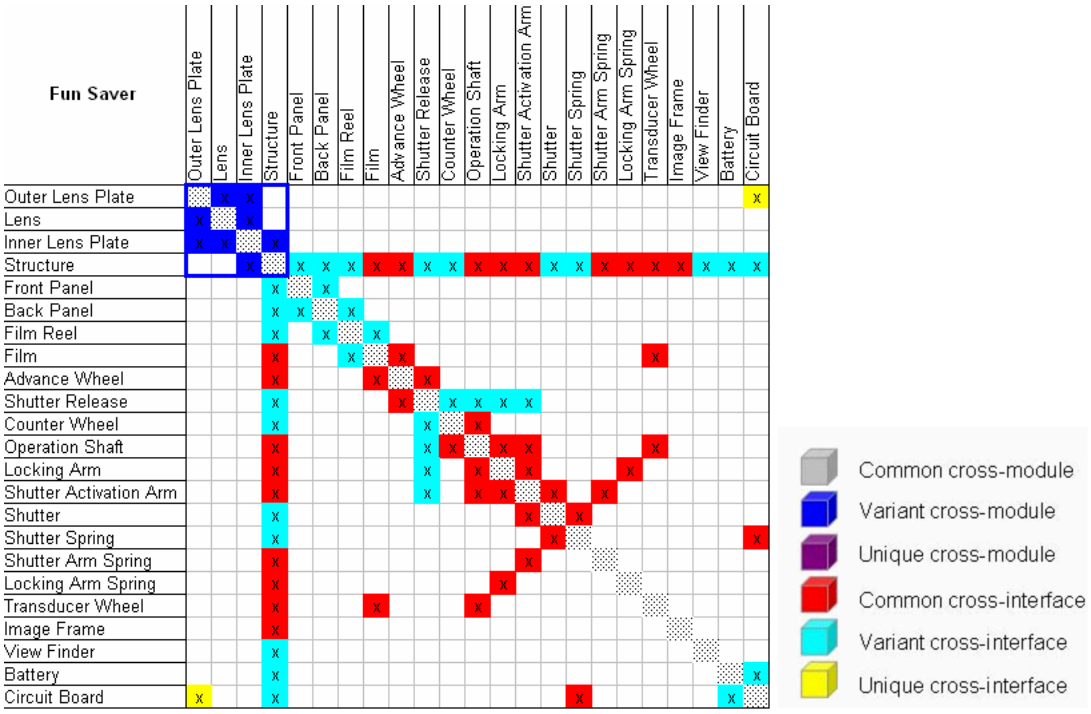

Figure 2.16
Clustered DSM^{V} of *Kodak Fun Saver* camera with a legend of modules and interfaces, adapted from Alizon et al. (2007)

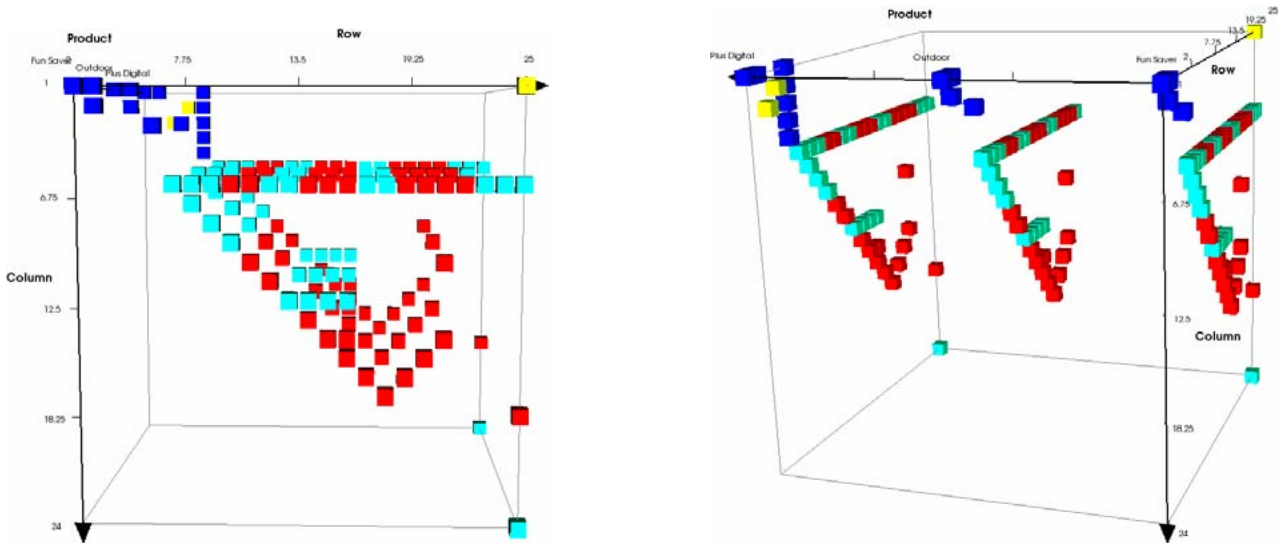

Figure 2.17
Two views of the camera family DSM^{3D}, showing several Kodak single-use camera DSM-s overlapping in 3D, adapted from Alizon et al. (2007)

2.3 Software Review

This section of the literature review will cover various applications already implemented for systems engineering applications with purposes similar to the ones proposed in the study.

2.3.1 Virtual Satellite

In October 2008, the German Aerospace Center (DLR) inaugurated the new Institute for Space Systems located in Bremen, Germany. This concentrates the competences in space engineering, enabling the DLR to build space systems in-house. Furthermore, a Concurrent Engineering Facility (CEF) was established according to ESA's Concurrent Design Facility (CDF) (Bandecchi et al., 2000) to offer the very effective approach of concurrent systems engineering. Additionally, the need for a tool supported process for the simulation-based space system development based on a modern and flexible software infrastructure was identified. The Virtual Satellite project aims at the definition of this process and the implementation of the needed infrastructure (Schumann et al., 2008).

There are general issues of inter-domain communication and understanding of the Virtual Satellite tool. It is believed that a tool which uses CEDESK as a parametric modeling tool could have the potential to overcome those issues, since CEDESK can interact with domain-specific models and tools. There is also an issue of understanding: before an expert starts using Virtual Satellite, he or she has to spend some time training since the GUI does not provide intuitive interaction.

Figure 2.18 illustrates the user interface of VirSat and Figure 2.19 describes a system architecture of Virtual Satellite, which visualize the system model and a connection to a virtual reality environment. The system proposed has high potential since it could improve the inter-domain communication, facilitate feasible design phase and provide more detailed and concrete models for the next design phases (Tsykunova, 2016).

The VirSat Client operates with documents called Visualization Models. These documents are being transferred to the VR environment of VirSat throughout the process of satellite design, providing a tool for both 3D Visualization and 3D Interaction.

VirSat has certain disadvantages: it has no integration of third-party tools in the architecture and is not an open-source software.

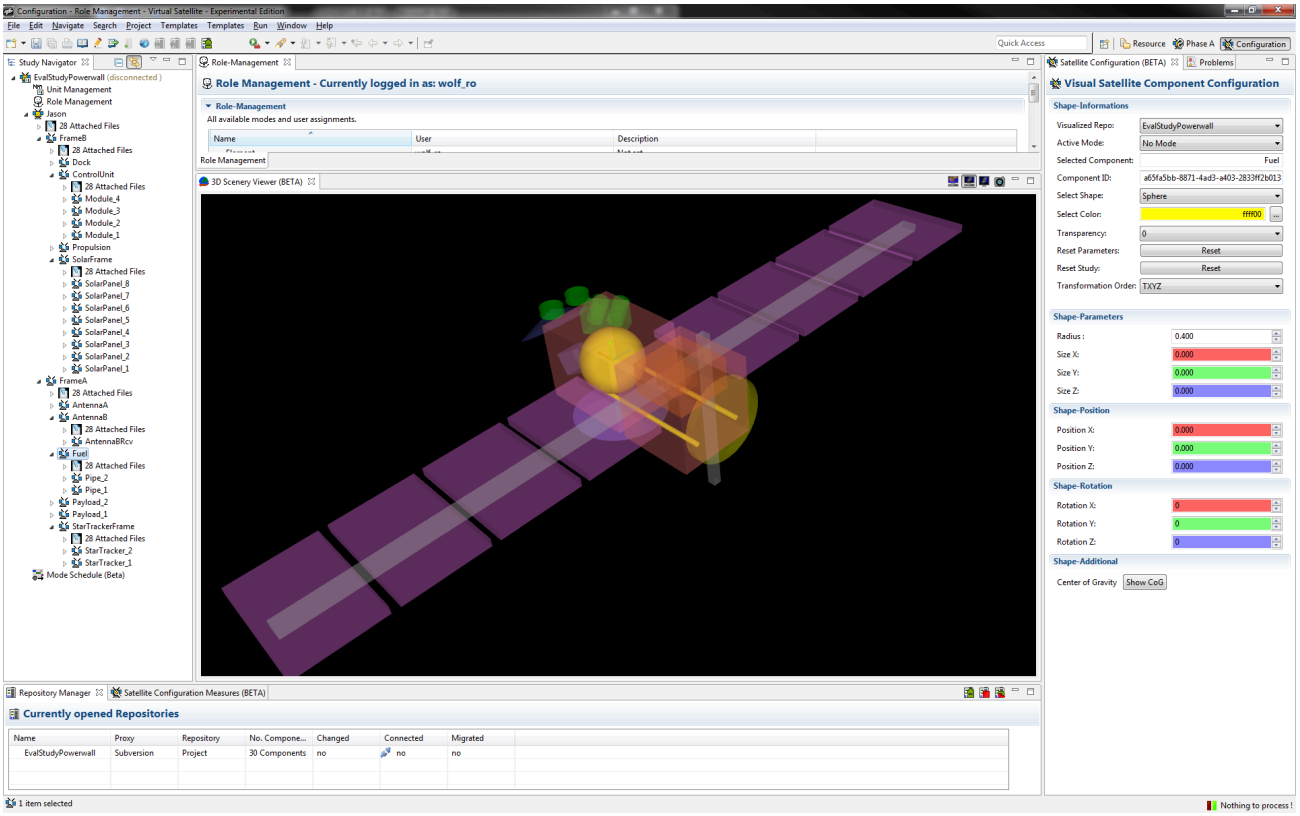

Figure 2.18
3D Visualization and interaction of the system model in the software Virtual Satellite, adapted from Deshmukh et al. (2015)

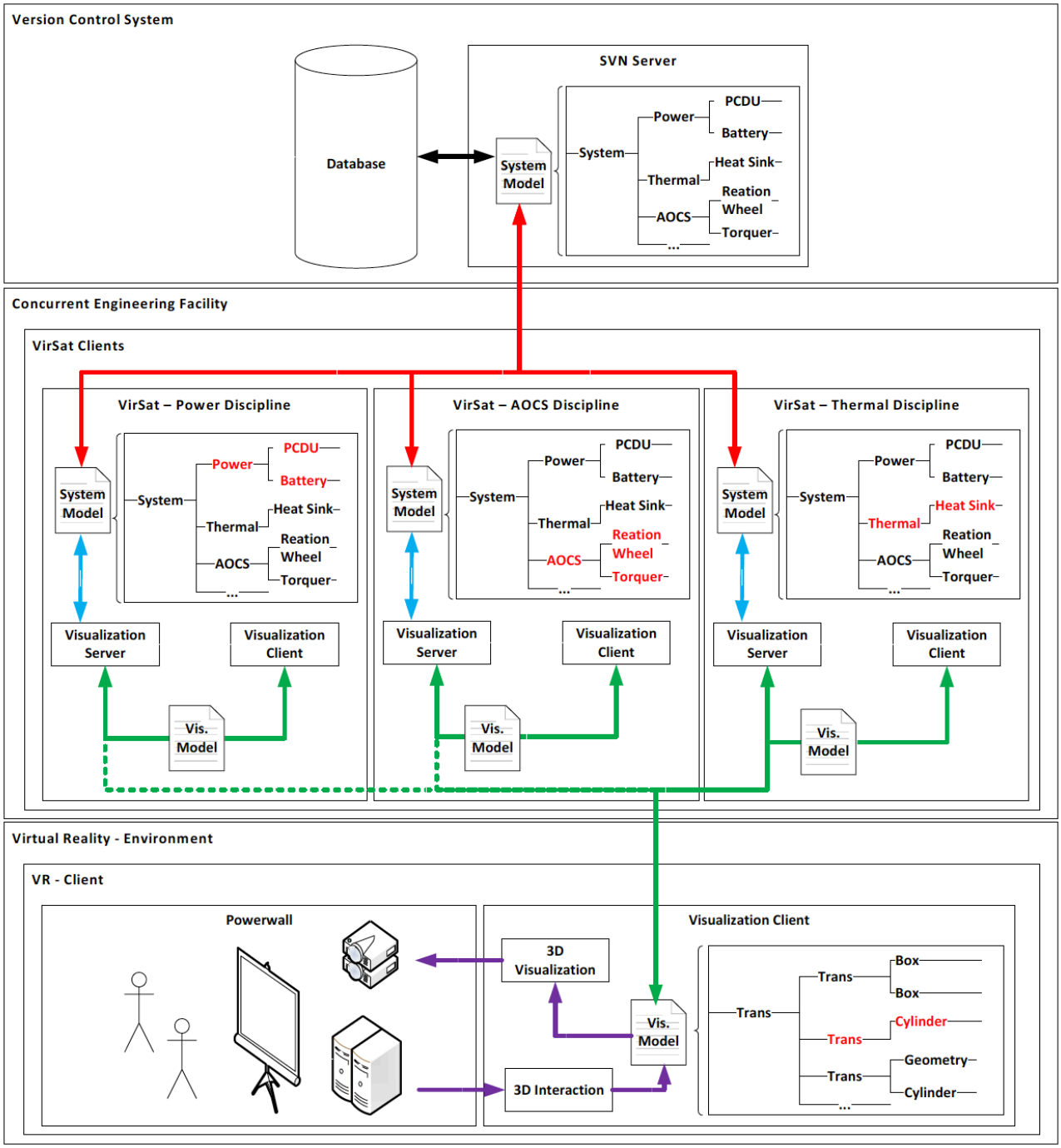

Figure 2.19
Architecture of Virtual Satellite exchanging system model information within the Concurrent Engineering Facility as well as data interaction in a VR environment, adapted from Deshmukh et al. (2015)

2.3.2 IDM

The French National Centre for Space Studies has its own concurrent design facility called Centre d'Ingénierie Concourante (CIC). For the build-up of system budgets and the exchange of parameters between disciplines, CIC makes good use of the Integrated Design Model (IDM) tool provided by the Concurrent Design Facility of ESA and resulting from close cooperation between both agencies (Bousquet et al., 2005).

Figure 2.20 represents the IDM architecture. The IDM is the central system budget tool originally developed by ESTEC and based on Excel® spreadsheets. All discipline specific tools gravitate around IDM with the Data Exchange process being a core of it.

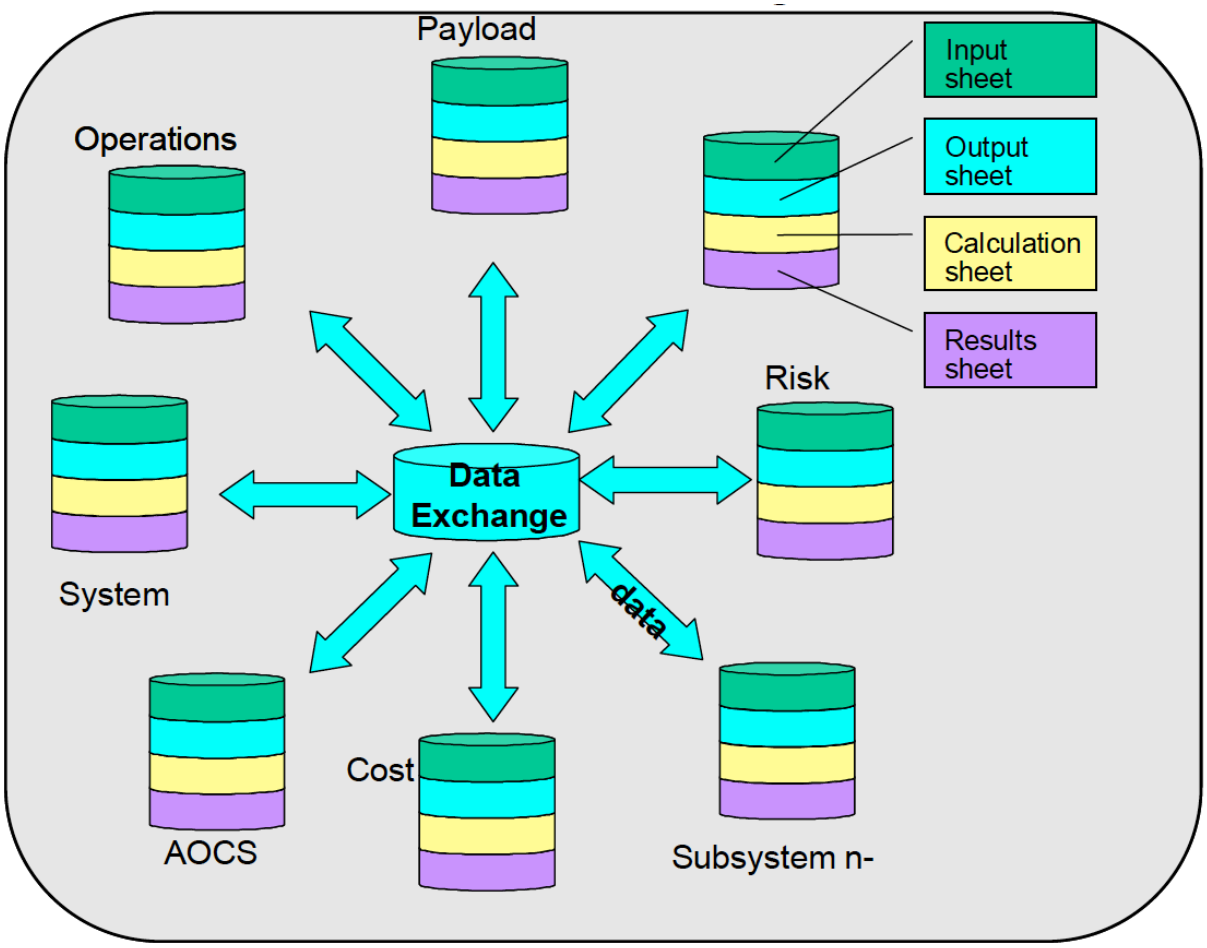

Figure 2.20
IDM architecture, adapted from Bousquet et al. (2005), courtesy of ESA

Table 2.3 lists the disciplines generally involved in a study within CIC, and their principal design tools. The name of the tool is underlined when a direct link has or should be established with the IDM.

Table 2.3
CIC's discipline and tool list, adapted from Bousquet et al. (2005)

System	IDM
AOCS	Matlab
CAD design	Catia V5
Comms	Access Database
Data Handling	Obade
Mission analysist & simulation	Opale, STK, Excel
Power	Saber
Propulsion	Excel database
Risk	Failcab, Cabtree, Supercab, Gencab
Structures	Patran, Nastran
Thermal	Thermica

Since IDM is based on Excel, there might be some interconnection issues relevant to all Excel-based software products. There are many reports on Excel being crashed, frozen, corrupting files, etc. Most of those problems require the user to switch to Excel and to resolve the issues manually (Microsoft Support, 2016). IDM has no integration with third party tools and is not open-source.

2.3.3 Cameo Systems Modeler

It was decided to review Cameo Systems Modeler as a cross-platform collaborative MBSE environment, which provides smart, robust, and intuitive tools to define, track, and visualize all aspects of systems in the most standard-compliant SysML models and diagrams. The environment enables systems engineers to: Run engineering analysis for design decisions evaluation and requirements verification Continuously check model consistency Track design progress with metrics System models can be managed in remote repositories, stored as standard XMI files, or published to documents, images, and web views to address different stakeholder concerns (NoMagic).

The architecture of Cameo Systems Modeler provided in Figure 2.21 shows that the tool is capable of bringing many other tools around it. Cameo Systems Modeler uses many tools of the MagicDraw origin, yet it has many third-party components in it.

Figure 2.21
Architecture of Cameo Systems Modeler, adapted from NoMagic (2017)

NoMagic provides many tools of their own design, but still depends a lot on external software. There is a disadvantage in this: the more external tools are being used, the more licenses should be provided and more version control needed (Spangelo et al., 2013).

2.3.4 CEDESK

As was covered above, one of the efficient approaches to design a complex system is concurrent engineering. It is proposed to find a tool for concurrent conceptual design. One of such tools is CEDESK developed by Knoll and Golkar (2016).

CEDESK is an open-source tool to facilitate co-located collaborative model-based conceptual design of complex engineering systems. This type of tool is also known as data exchange for concurrent engineering studies. Multidisciplinary design teams can use CEDESK to facilitate their work together by building shared parametric models of their system of interest (Knoll and Golkar, 2016).

CEDESK aims to bring concurrency to conceptual design and to solve the problem of designing complex systems composed by multiple subsystems referring to different disciplines. Costs committed on the conceptual design stage of lifecycle are equal to 70%, while only 8% are spent as illustrated in Figure 1.1.

CEDESK mostly focuses on its primary function: exchange parametric model information between discipline experts. However, visualization of basic three-dimensional geometry is not embedded into CEDESK.

CEDESK allows multiple users to work concurrently on the design of a system, while distributing design authority over subsystems among discipline experts. Figure 2.22 shows the user interface used by discipline experts to collaborate on a system model. The structure sub-window represents the systems as a tree model. The parameters sub-window lists all parameters in a selected subsystem. It is also possible to see all linkages for each parameter and subsystem, making it a good tool for system representation.

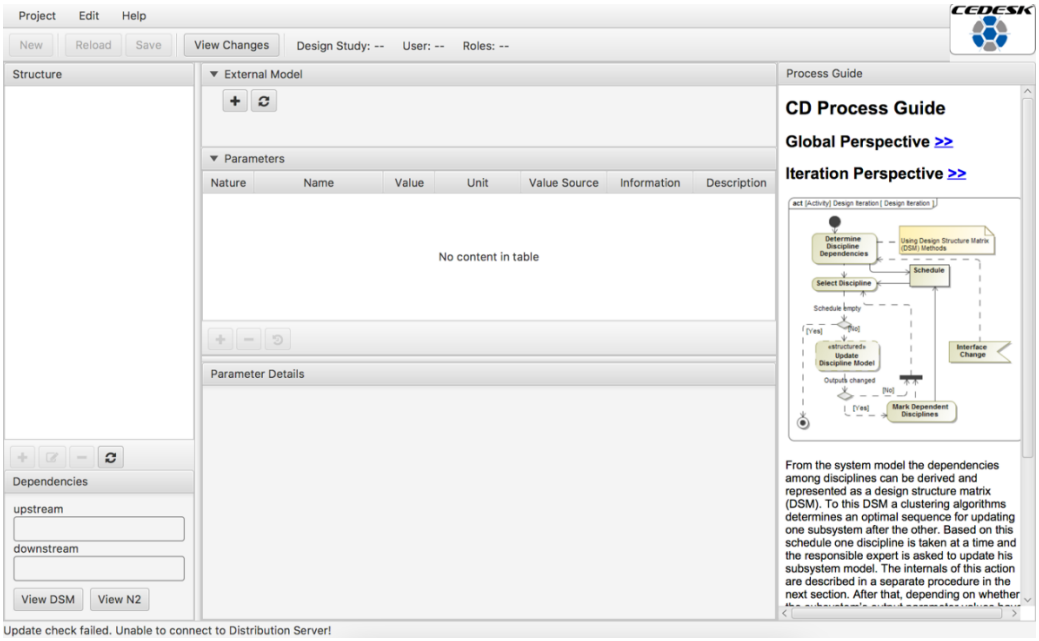

Figure 2.22
Starting screen of CEDESK

CHAPTER 3

APPROACH

It is proposed to develop a tool that could represent interfaces in a 3D DSM view of impact, likelihood, and risk scores using 3D Modeler. Interfaces are proposed to be inherited from MySQL databases used in CEDESK, which makes the representation to be based on a parametric model of a system. C3D Modeler is capable of representing geometry of a system being developed. It is believed that the proposed software prototype has the potential to become a next generation MBSE platform, capable of representing a geometrical model of the system in 3D with supporting tools orbiting the model. Figure 3.1 represents the structure proposed for the next generation MBSE software which could eventually arise from the results of the study. It is proposed that this software should be represented as a Concurrent Conceptual Design and Demonstrator Assessment Platform consisting of a geometric, parametric, and system modeling tool. In the proposed software prototype, it was suggested to implement CEDESK as a parametric modeling tool, C3D Modeler as a geometric modeling tool, and DSM methodology proposed by Garg et al. (2017) as a way to represent a system and interfaces inside of it. Other tools could include many other tools to facilitate concurrent conceptual design, such as Microsoft Excel. However, it must be noted that a variety of tools could bring problems with versioning and licensing. This kind of platform could potentially become in the next generation MBSE software since the proposed tool is supposed to be able to represent the system being developed with modeling tools orbiting it.

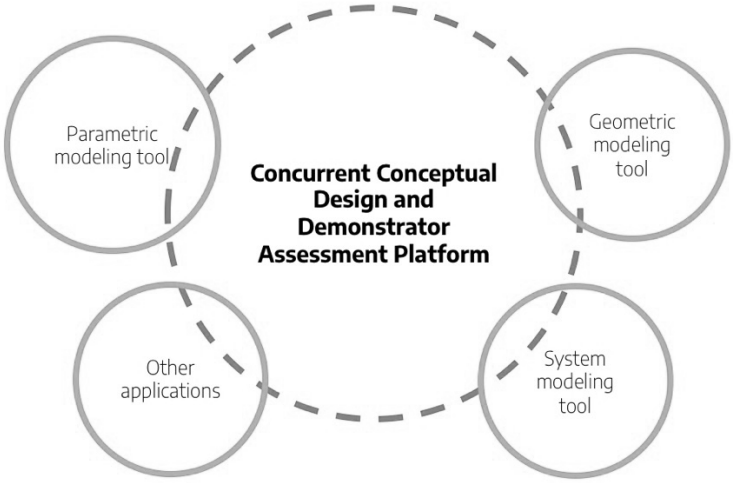

Figure 3.1
Proposed next generation MBSE platform structure

Figure 3.2 represents the approach used in this thesis project. Firstly, a tool for parametric modeling has to be chosen. It is proposed to fill the need in a concurrent conceptual design tool by CEDESK developed by Knoll and Golkar (2016). CEDESK is a tool, which is aimed to support the concurrent conceptual phase and can most effectively work with behavioral data.

Data used in initial phases of product development is predominantly behavioral in nature, that is, a large part of this data does not refer to the geometrical parameters of the system. However, current product lifecycle management (PLM) systems are based on the geometrical master model concept and thus work best for the detailed design, where the data are mostly geometric. Thereunder, there is still a gap between parametric and geometric modelling which has to be eventually filled.

Thus, a tool for geometric modeling is required. As a way to fill the gap between parametric and geometric modelling, it is proposed to integrate the C3D solid modeling kernel with MySQL databases used in CEDESK in a software prototype which has a potential to enhance demonstrator feasibility assessment by representing DSM matrices in 3D.

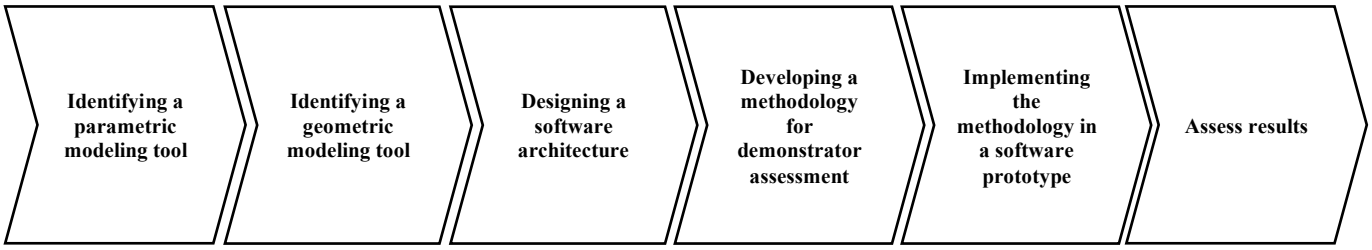

Figure 3.2
The approach proposed to achieve the target of this thesis project

3.1 Parametric modeling tool

It is proposed to fill the need in concurrent conceptual design tool by CEDESK developed by Knoll and Golkar (2016) which was briefly covered in Section 5.4 of Chapter 2.

The architecture shown in Figure 3.3 has been implemented for the European Space Agency (ESA) in the Open Concurrent Design Tool (OCDT) (ESA, 2014a), which use is limited to ESA member states. The commercial version of it called Concurrent Design Platform (CDP) (Fijneman and Matthyssen, 2010). For a similar purpose, the German Aerospace Agency (DLR) developed Virtual Satellite (VirSat) (Schaus et al., 2010), and the Jet Propulsion Laboratory developed Open

Model Based Engineering Environment (OpenMBEE) (NASA JPL, 2016) (see Table 3.1 for a comparison of commonly used tools for conceptual design studies in space agencies).

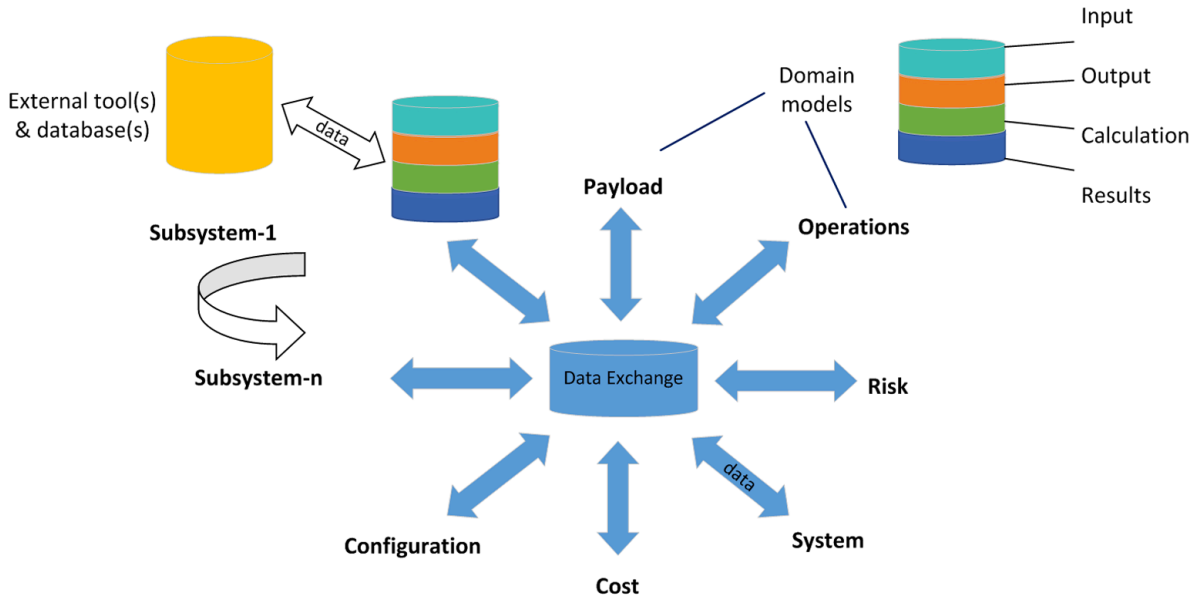

Figure 3.3
Tool architecture – a central data exchange connecting all domain models, adapted from Bandiccheri et al. (2000)

Table 3.1
Comparison of tools for conceptual design in aerospace, adapted from Knoll and Golkar (2016)

Group	PLM tool	MDO tool	Systems engineering tools		Concurrent conceptual design tools					
Tool	ENOVIA (2014)	Model Center 11.2	MagicDraw 18	OpenMBEE	IDM	VirSat 3	OCDT	\CDP 3	Valispace	CEDESK
References	(Dassault Systems, 2016)	(Phoenix Integration, 2015)	(No Magic, 2015)	(Kulkarni et al., 2016; NASA JPL, 2016)	(Bousquet et al., 2005)	(DLR, 2016; Schaus et al., 2010)	(Braukhane, 2015; ESA, 2014)	(Fijneman and Matthyssen, 2010; RHEA-Group, 2015)	(Valispace, 2017)	(Fortin et al., 2017)
Aspect										
Multi-user support	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Lifecycle phase focus	Design, manufacturing	Design	Conceptual design	Design	Conceptual design	Conceptual design	Conceptual design	Conceptual design	Conceptual design	Conceptual design
Parametric modeling focus	Geometry and behavior	Analysis and optimization	Description	Description	Behavior	Behavior and geometry	Behavior	Behavior	Behavior	Behavior
Version control	Yes	No	Limited	Yes	Limited	Yes	Yes	Yes	No	Yes
Primary user interface	Own client	Own client	Own client	MagicDraw	Excel®	Own client	Excel®	Excel®	Own Web	Own client
Integration with third party tools	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Limited
Availability	Commercial	Commercial	Commercial	Open Source	ESA community	Free	ESA community	Commercial	Commercial	Open Source

3.2 Geometric modeling tool

Since geometrical representation is needed to represent a model in the center of a system and interfaces between subsystems, it is proposed to use C3D Modeler as a tool for it. C3D Modeler can be used to create stand-alone applications which is crucial for size-reduction of any future MBSE software. It has the potential to be a base for a VR tool as well. The objects, methods, and algorithms used by the C3D modeling kit are described by Golovanov (2014).

C3D Labs is a company aimed for developing and promoting its geometrical kernel. Being the most popular geometrical modelling kernel in Russia, C3D Toolkit is gaining customers worldwide as well. Customers of C3D Labs are CAD/CAM/CAE developers and various institutions, such as the Skolkovo Institute of Science and Technology. Today, C3D Labs is a part of ASCON group and a resident of the Skolkovo Innovation Center.

It was proposed to implement the C3D Kernel and C3D Vision in order to develop a web-based application for representation and assessment of a demonstrator. The following tasks are accomplished:

1. Building the software architecture.
2. Development of a framework with the C++ programming language on programming level.
3. Development of a methodology for demonstrator assessment.
4. Implementation of the developed methodology in a software prototype using the C3D geometrical kernel.

In C3D Toolkit, a geometrical object describes the form of the modeled object. Geometric objects include curves, surfaces, bodies as well as topological objects that describe geometric properties that don't depend on quantitative features and describe permanently interconnected points in 3D space. There are two-dimensional and three-dimensional geometric objects. Two-dimensional objects are used to work in definition areas of surface parameters, as well to work with planes of local 3D coordinate systems.

The C3D geometric kernel operates with geometric model objects shown in Figure 3.4 (C3D Labs, 2017). Such variety of operable geometric objects implies in the high potential for MBSE software as well. For instance, MbPlaneInstance can be used to represent a three-dimensional plot, diagram, or DSM.

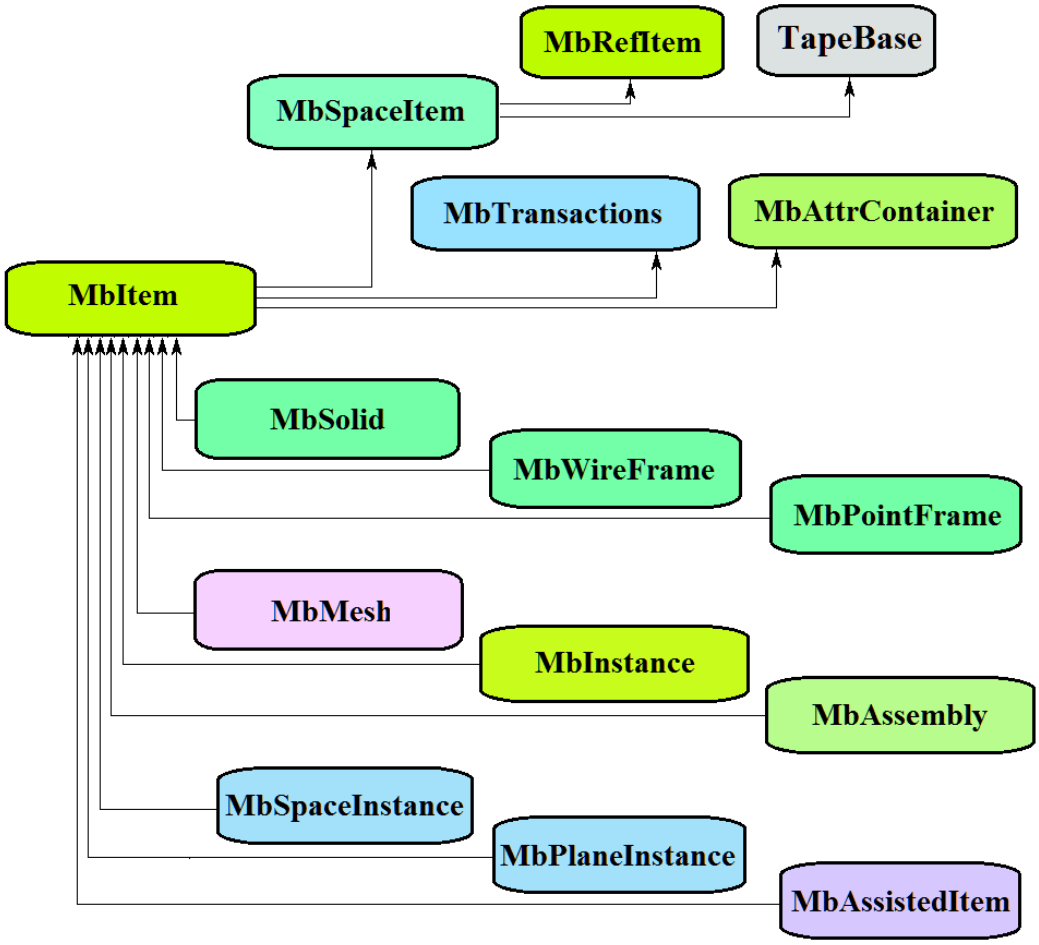

Figure 3.4
Geometric model object operated by C3D geometric kernel, adapted from C3D Labs (2017)

CHAPTER 4

RESULTS

4.1 Proposed software prototype architecture

Firstly, it was decided to develop a software architecture, in order to build the bridge between design requirements and technical software requirements by understanding use cases, and then finding ways to implement those use cases in the software. The goal of an architecture is to identify the requirements that affect the structure of the application (NASA, 2014).

It is possible that a user might need to work with multiple documents, so it was chosen to use the multi-document interface (MDI) structure as a base of the application. All documents are proposed to be accessed within a single framework on the top level of application architecture as illustrated in Figure 4.1.

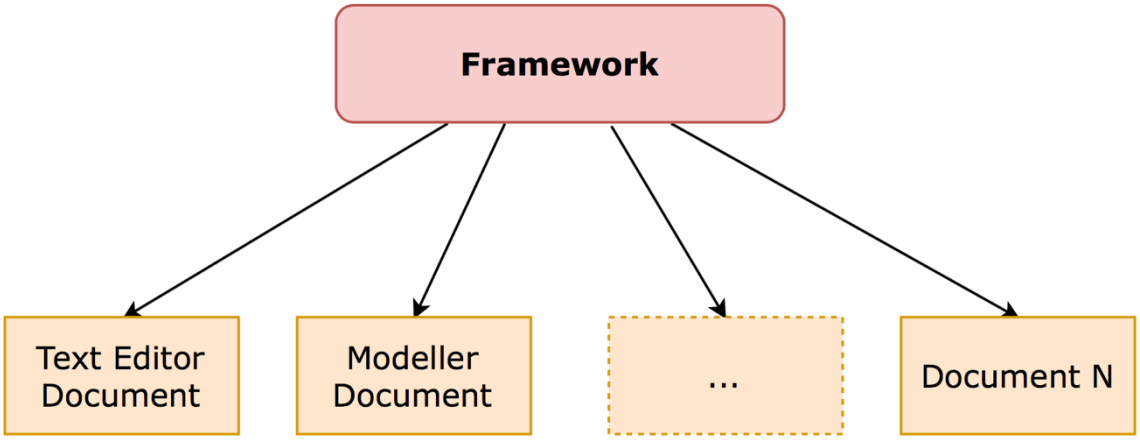

Figure 4.1
The top level of the architecture

Figure 4.2 represents the second level of architecture, which is important to make links inside the application clearer. Inside the architecture, there is a DLL with a GUI and Template Plugin. The GUI is responsible for HMI, while the Template Plugin allows creation of new documents with different types. The GUI, being adapted from the C3D testing application, consists of the Framework with various tools such as menu, statusbar, toolbar, etc. The Template Plugin is controlled by the Framework and able to store and change data in the document. It also interacts back with the

Framework through the Interface. The Interface sends signals to the Registrator Plugins, which sends signals with the Framework to the application itself. The Python script interacts with the Document through the Template Plugin and retrieves data from MySQL databases. The reason for implementing the Python script instead of coding purely in C++ is covered in Section 2.4 of this chapter.

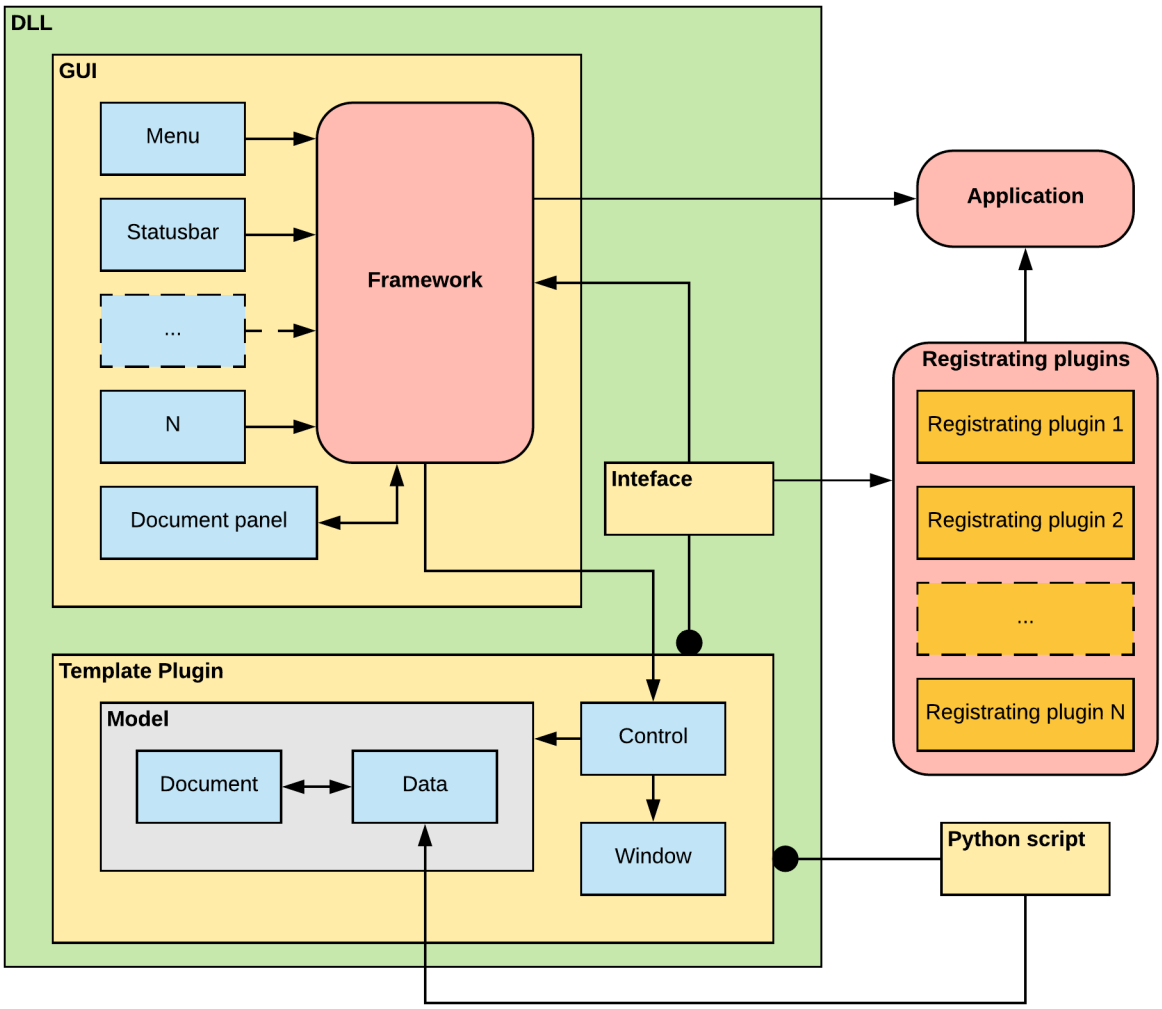

Figure 4.2
The second level of the architecture

The block diagram provided in Figure 4.3 shows the interfaces between the Window and the Manager. The Model, which basically is the data used in projects, may be represented as the Document, which could be seen in the Window by the user. The Manager Editor sends signals to the General Manager in order to make changes in the Document.

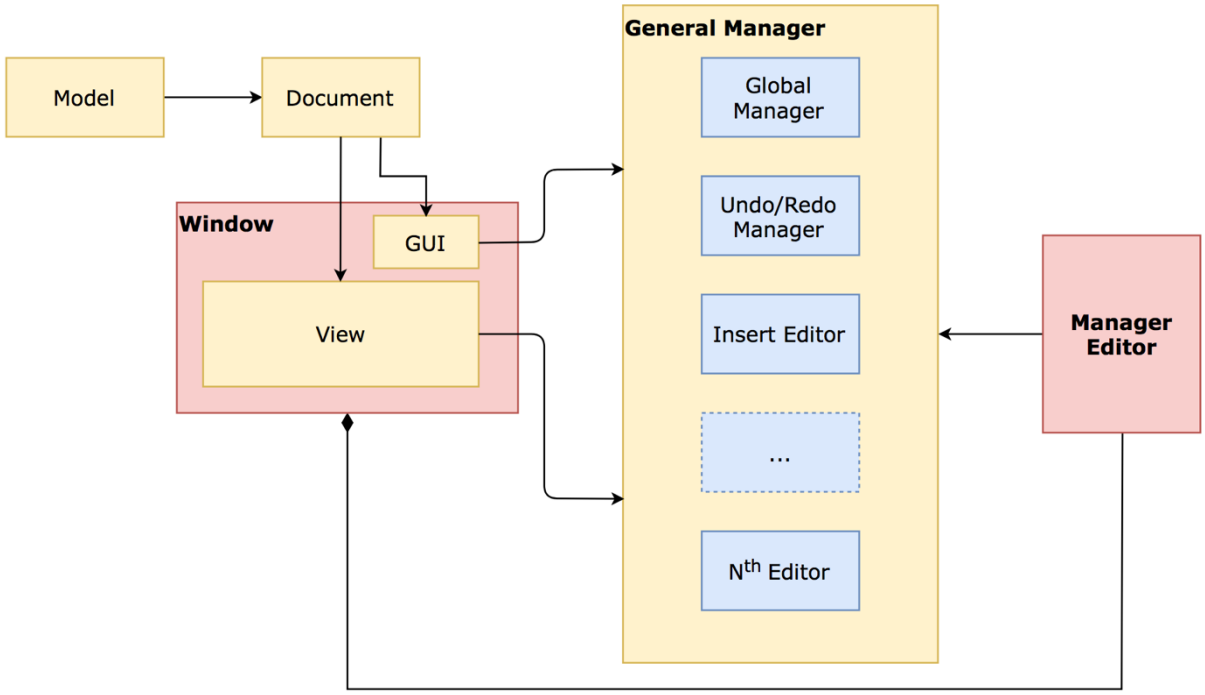

Figure 4.3
Block diagram for representing the Manager Editor interface with the Window

4.2 Development environment

4.2.1 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code (Webster, 2017).

C3D kernel is written in C++ and Microsoft Visual Studio was chosen since it allows use the C++ programming language in a professional way.

4.2.2 CMake

CMake is an extensible, open-source tool that constructs the build procedure in an operating system and in a compiler-independent way. Unlike many cross-platform systems, CMake is designed fo use in conjunction with the native build environment, which is Microsoft Visual Studio in this case. Simple configuration documents located in each source directory (called CMakeLists.txt documents) are used to generate standard build documents (e.g., makefiles on Unix and projects/workspaces in Windows MSVC) which are used in the usual way. CMake can generate a native build environment that will compile source code, create libraries, generate wrappers and build executables in arbitrary combinations. CMake supports in-place and out-of-place builds and may therefore support multiple builds from a single source tree. CMake additionaly supports static and dynamic library builds. Another function of CMake is in generating a cache document that is designed for use with a graphical editor. For instanse, when CMake runs, it locates documents, libraries, and executables, and can encounter optional build folders. This information is accumulated into the cache, which may be modified by the user before the generation of the native build documents (CMake, 2017).

CMake is designed to assist with complex directory hierarchies and applications dependent on numerous libraries. For instance, CMake aids projects consisting of more than one toolkits (i.e., libraries), in which each toolkit could contain numerous directories, and the application relies upon on the toolkits plus extra code. CMake also can manage conditions in which executables ought to be built with the intention to generate code which is then compiled and linked right into a final application. Since CMake is open source, and has a simple, extensible design, CMake may be extended as required to support new features. The build procedure is managed through creating one or more CMakeLists.txt documents in each folder (which includes subfolders) that make up a project. Each CMakeLists.txt includes one or more commands. Each command has the form COMMAND (args…) in which COMMAND is the name of the command, and args is a white-space separated list of arguments. CMake offers many pre-defined commands and presents an interface for including user-defined commands. Furthermore, the advanced user can upload other makefile generators for a specific compiler/OS combination.

Figure 4.4 shows the user interface of CMake used in the development of the software prototype, where the source code is a folder with C++ and header files used to build the solution; binaries are the resulting software prototype which is the result of compiling the source code. The CMake GUI has a table for user-defined variables which have a name and a value. The connection to

the C3D libraries are done through this variable interface, as well some general configuration properties of the program.

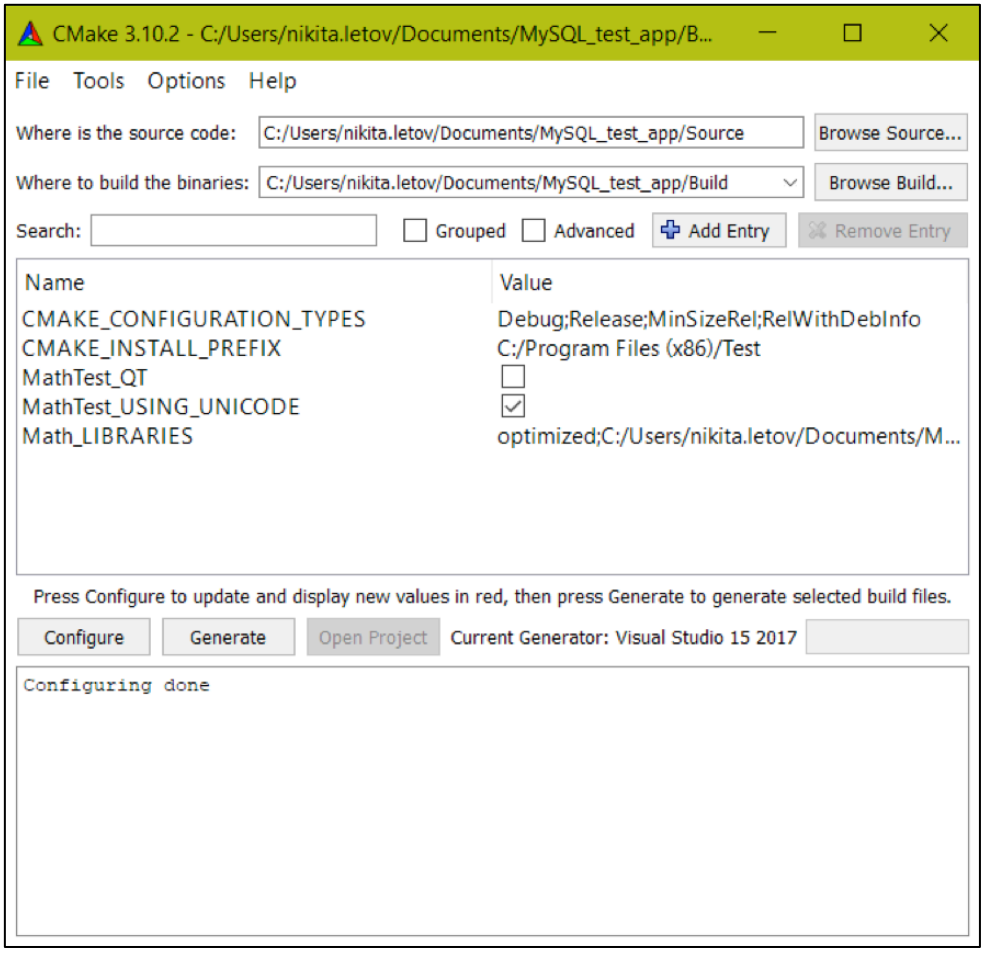

Figure 4.4
User interface of CMake

4.2.3 SourceTree

Agile methods grew out of the real-life project experiences of leading software professionals who had experienced the challenges and limitations of traditional waterfall development on project after project. The approach promoted by agile development is in direct response to the issue associated with traditional software development both in terms of overall philosophy as well as specific processes (McLaughlin, 2005).

Agile development, in its simplest form, offers a lightweight framework for helping teams, given a constantly evolving functional and technical landscape, maintain a focus on the rapid delivery

of business value (i.e., bang for the buck). As a result of this focus, the benefits of agile software development are that organizations are capable of significantly reducing the overall risk associated with software development.

In particular, agile development accelerates the delivery of initial business value, and through a process of continuous planning and feedback, is able to ensure that value continues to be maximized throughout the development process. As a result of this iterative planning and feedback loop, teams are able to continuously align the delivered software with desired business needs, easily adapting to changing requirements throughout the process. By measuring and evaluating status based on the undeniable truth of working, testing software, much more accurate visibility into the actual progress of projects is available. Finally, as a result of following an agile process, at the conclusion of a project is a software system that much better addresses the business and customer needs (VersionOne, 2005).

Figure 4.5 displays the differences between agile and waterfall development processes. By delivering working, tested, deployable software on an incremental basis, agile development delivers increased value, visibility, and adaptability much earlier in the life cycle, significantly reducing project risk.

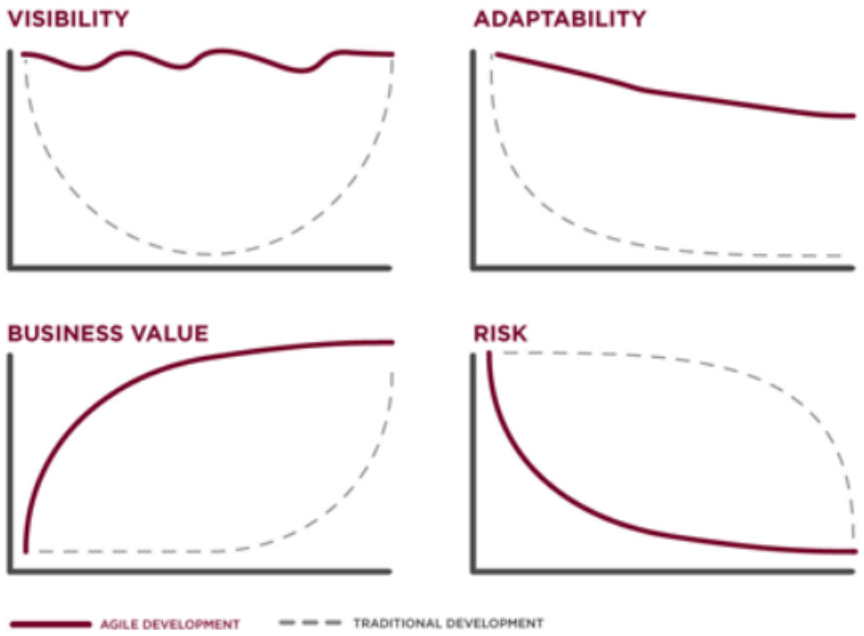

Figure 4.5
Agile development value proposition, adapted from VersionOne (2005)

It was decided to use SourceTree as a tool providing agile development to the study. SourceTree is a free Git client for Windows and Mac for programmers working with Git in development. It provides a visual interface between a user and Git avoiding a command line. The

reason for choosing SourceTree as a Git client for the study is in the desire to make the study agile by branch management, working copies, and branch history (Donnelly, 2015).

Figure 4.6 shows the user interface of SourceTree used in the development of the software prototype with a graph of commits with branches and commit descriptions. Each commit has a date stamp, author, commit ID, its parents. On the bottom right-hand side of the user interface, there is a code viewer which allows a user to see and analyze changes in each file in the commit. Deleted rows of code are highlighted with red color and has a ‘minus’ sign at the beginning of each deleted row, while added rows of code are highlighted with green color and has a ‘plus’ sign at the beginning of each added row.

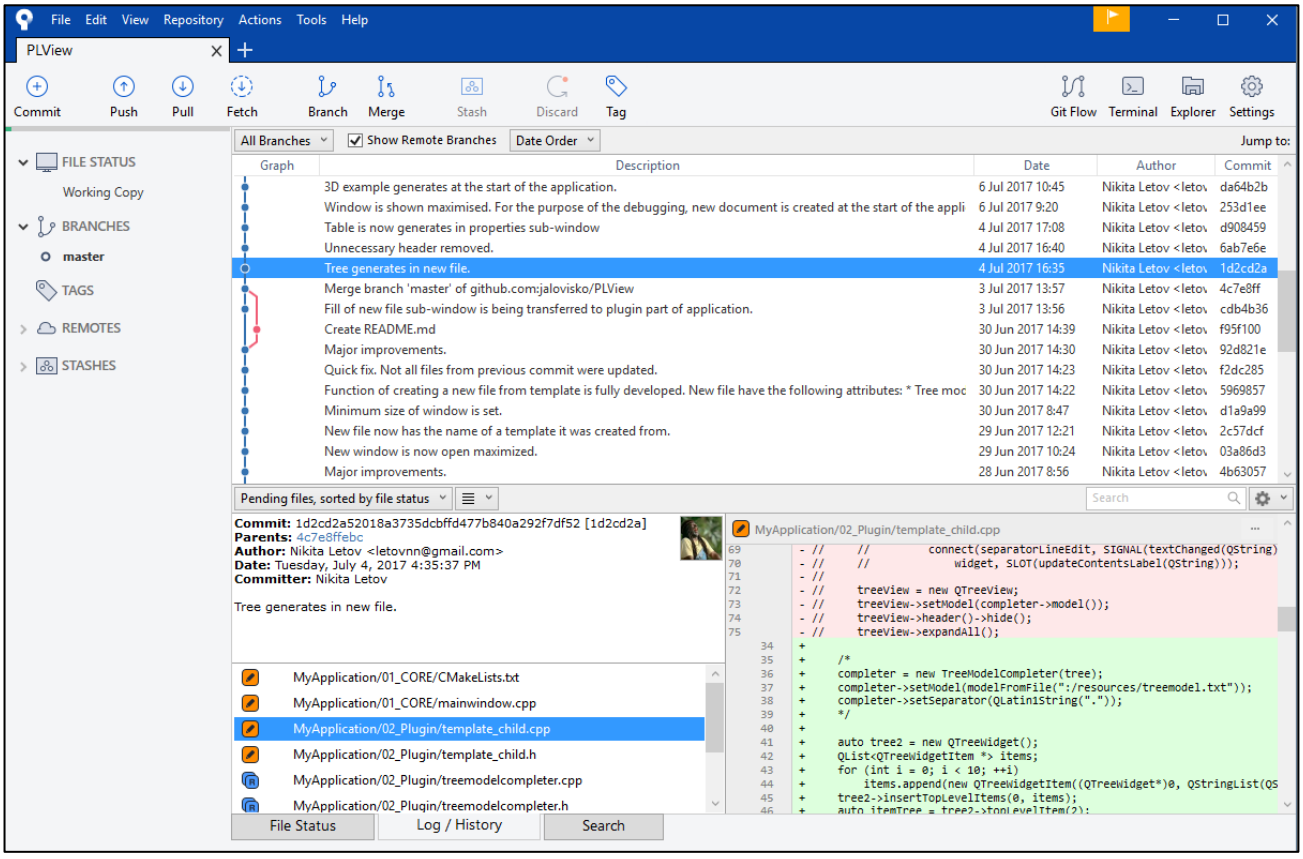

Figure 4.6
User interface of SourceTree

4.2.4 MySQL Connector/Python

CEDESK operates with MySQL databases and is written in the Java programming languages while C3D Modeler is written in the C++ programming language. Since it was decided to develop a software prototype in the scope of this thesis, there has to be an interface between them. MySQL provides standards-based drivers for JDBC, ODBC, and .Net enabling developers to build database applications in their language of choice. In addition, a native C library allows developers to embed MySQL directly into their applications (MySQL, 2018). MySQL developed MySQL Connector drivers for the following several programming languages and environments:

- ADO.NET Driver for MySQL (Connector/NET)
- ODBC Driver for MySQL (Connector/ODBC)
- JDBC Driver for MySQL (Connector/J)
- Node.js Driver for MySQL (Connector/Node.js)
- Python Driver for MySQL (Connector/Python)
- C++ Driver for MySQL (Connector/C++)
- C Driver for MySQL (Connector/C)
- C API for MySQL (mysqlclient)

At first, from this list the C++ Driver for MySQL appeared to be the most appropriate option to be implemented in the software prototype since C3D Modeler is written in C++. However, a preliminary study discovered, that the C++ Driver requires a significant amount of external dependencies other than the C++ standard library and developing the software prototype with the C++ Driver would take a significant amount of time to adjust and implement it.

From the list of the drivers above, the Python Driver for MySQL stands out as one the few drivers without any external dependencies other than the standard required library – the standard Python library. Moreover, Python has a large community of developers, resulting in a lot of documentation available online. Thus, it was decided to develop a Python script which would be called by the program whenever a connection to a MySQL database requires to be established.

4.3 Use case

In order to analyze the performance of the developed application, it is important to perform a use case analysis. For this purpose, a dataset from a feasibility study of a project supported by the CEDESK application: the LaserNaut satellite project. This representative data of a concurrent conceptual design study, was generated by students and researchers from Skoltech participating in Satellite Engineering projects. The satellite design included the following conventional subsystems: Attitude Determination and Control System (ADCS), Communication, Power, Orbit, Thermal, and Structure as well as an Optical payload. Input and output data of each subsystem were extracted from the CEDESK database for the analysis (Fortin et al., 2017). A solid model of the satellite made in SolidWorks is presented in Figure 4.7 and Figure 4.8.

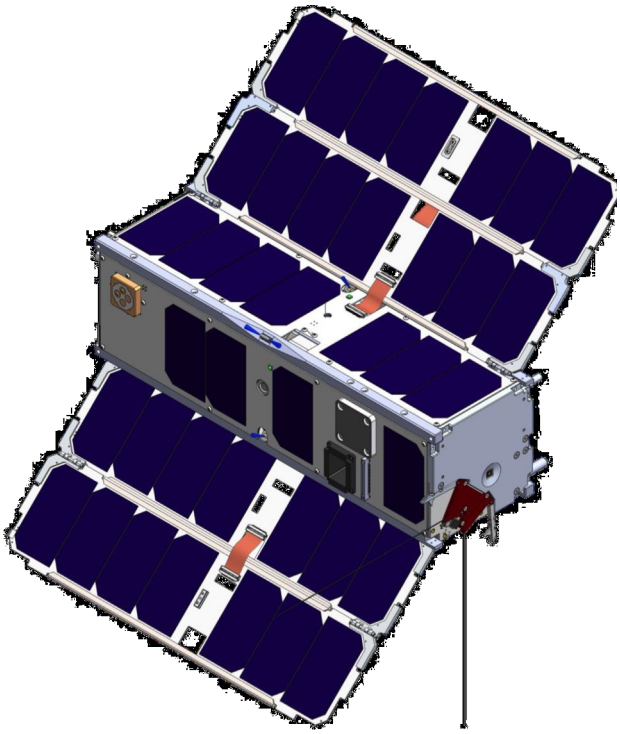

Figure 4.7
3U CubeSat – Tyvak Endeavor, adapted from Knoll et al. (2016).

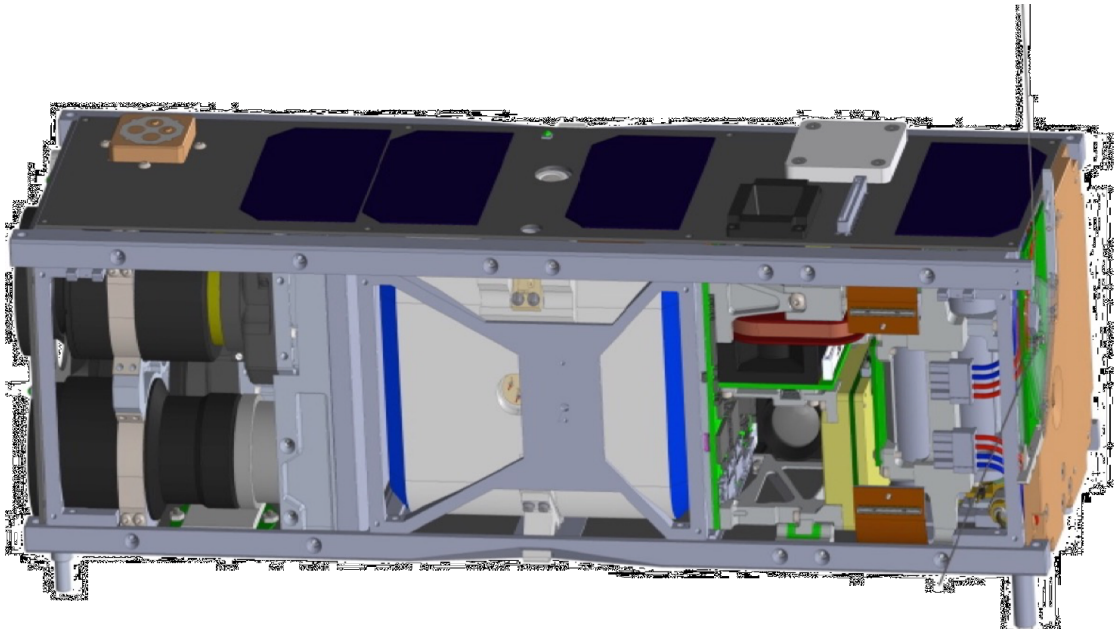

Figure 4.8
3U CubeSat – Tyvak Endeavor without the side panels and the solar panels, adapted from Knoll et al. (2016)

4.4 Application

4.4.1 Implementation

It was suggested develop the prototype application using the C3D geometrical modeling kernel and basing on the architecture presented in Section 1 of Chapter 4. Appendix A of this thesis includes information and an example relevant to using C3D Kernel in the study.

After the software and all its sub-functions are specified, the implementation phase starts. Here coding is a straightforward process when the specifications and designs are well made. Coding process usually includes some code reviews, mainly intended for analyzing that the code is a well commented and follows the good programming practices.

As covered in Section 2.4 of Chapter 4, it was decided to implement the Python script for establishing the connection between the software and MySQL databases. Appendix B of this thesis includes the code for the Python script used in the software prototype with relevant comments and an example of data retrieved.

For the implementation, a new folder in a comfortable working position was created to contain all the source and built code of the software prototype.

It was decided to use the 2017 version of Microsoft Visual Studio as the development environment. The C3D Modeler version was chosen to correspond to the development environment.

Include, Debug, and Release files of the application were adapted from the testing application of C3D Modeler. Source files were adapted and expanded with new functionality relevant for the study.

CMake files were adapted and expanded with new functionality relevant for the study. Appendix C of this thesis includes the code of one of the few CMake files used for the development of the software prototype. Considering that the software prototype is located in directory \SOFTWARE_PROTOTYPE, the directories for the source code and for the built binaries are \SOFTWARE_PROTOTYPE\Source and \SOFTWARE_PROTOTYPE\Build respectively. After that the project is configured using CMake. The generator for this project is specified to be Microsoft Visual Studio 2017. After that, the project files are generated using CMake and can be opened directly from CMake.

In the project, there have been developed various expansions of the original C3D code that were relevant for the study. Appendix D of this thesis includes excerpts the C++ code used for 3D DSM plotting using C3D Modeler.

4.4.2 Testing

As the result of the study, the software prototype was made. The initial screen is shown in Figure 4.9. It has a toolbar with quite common functions, such as creation of a new file, showing all windows, etc. View window can show one or more sub-windows since the MDI structure was chosen. It was decided to build this application on top of the test application provided by C3D Labs. Still, the main idea of this application is to represent the possibility to represent data from MySQL databases in form of DSM matrices. It is not supposed to be a commercial software product ready to be sold and implemented.

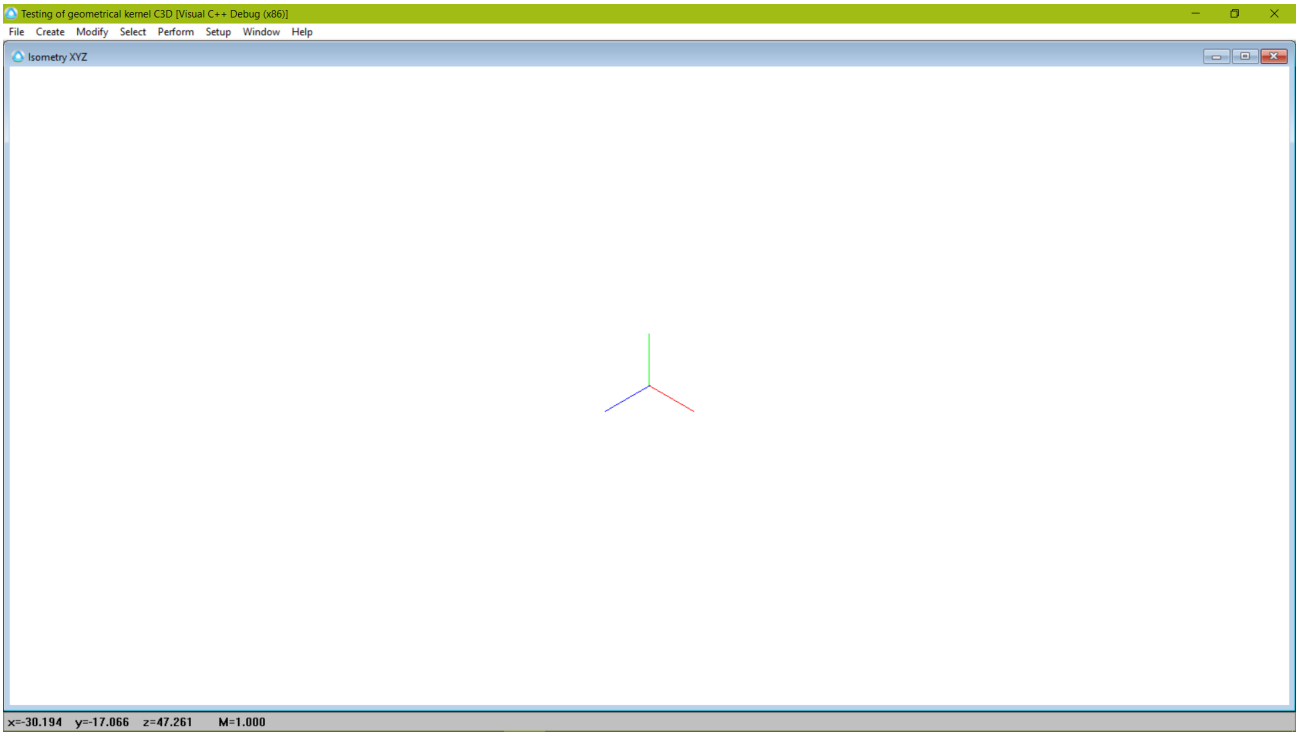

Figure 4.9
Initial screen of the developed software prototype.

In order to provide the user with an access to the developed features of the software prototype, some new elements of the GUI were developed. As illustrated in Figure 4.10 the **File** menu has two specific buttons: **New MySQL connection** and **New risk assessment**. Clicking on **New MySQL connection** launches the process of connecting to a MySQL database using user credentials for it. Those credentials consist of a regular set of parameters needed to access a MySQL database – a username, a password, a host name, and a database name (Figure 4.11, a – d).

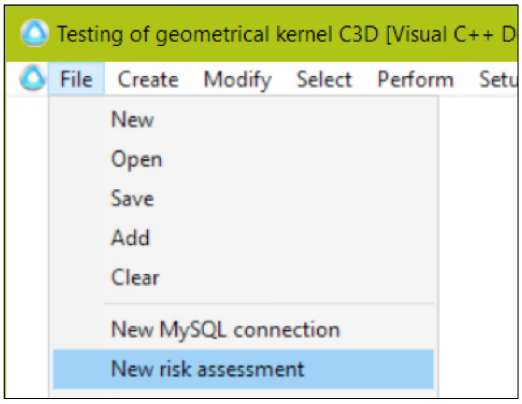

Figure 4.10
Part of the File menu of the developed software prototype

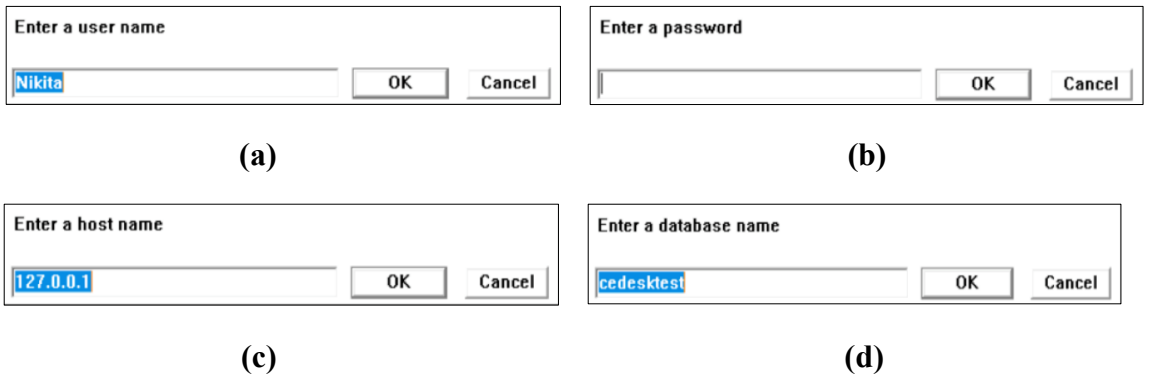

Figure 4.11
Windows for entering user credentials. (a) Window for entering a username. (b) Window for entering a password. (c) Window for entering a host name. (d) Window for entering a database name.

After performing the MySQL access procedures, the software prototype performs access the Python script which accesses a MySQL database using the entered credentials and retrieves data from the database. An example SQL query for retrieving data consisting of parameter dependencies in the LaserNaut cubesat is provided below:

```
CREATE OR REPLACE VIEW parameter_dependencies AS
SELECT
y.id AS system_id,
y.`name` AS system_name,
s1.id AS target_subsystem_id,
s1.`name` AS target_subsystem_name,
p1.id AS target_parameter_id,
p1.`name` AS target_parameter_name,
p1.`value` AS target_parameter_value,
u.`name` AS target_parameter_unit,
p2.id AS source_parameter_id,
p2.`name` AS source_parameter_name,
s2.id AS source_subsystem_id,
s2.`name` AS source_subsystem_name
FROM parametermodel p1
JOIN unit u ON p1.unit_id = u.id
JOIN subsystemmodel s1 ON p1.parent_id = s1.id
JOIN systemmodel y ON s1.parent_id = y.id
JOIN parametermodel p2 ON p2.id = p1.valueLink_id
JOIN subsystemmodel s2 ON s2.id = p2.parent_id
WHERE p1.valueLink_id IS NOT NULL;
```

```
SELECT
`target_subsystem_id`,
`target_subsystem_name`,
`source_subsystem_id`,
`source_subsystem_name`,
COUNT(source_parameter_id) AS linked_parameters
FROM parameter_dependencies
WHERE system_name = \"LaserNaut\"
GROUP BY `target_subsystem_name` , `source_subsystem_name`;
```

The query above returns a set of data that can be interpreted as provided in Table 4.1. The number of linked parameters between subsystems can be interpreted as the impact score used in the method proposed by Garg et al. (2017) which was discussed above in Section 2.2 of Chapter 2 with the literature review.

Table 4.1
Example dataset used for the case study

#	Target Subsystem		Source Subsystem		Linked parameters
	ID	Name	ID	Name	
1	17840	Bio Payload	17838	Mission + Programmatics	1
2	17840	Bio Payload	17842	Power + Thermal	2
3	17840	Bio Payload	17841	Structure	7
4	17845	OBDH	17840	Bio Payload	1
5	17845	OBDH	17846	Optical Comms	1
6	17846	Optical Comms	17843	AOCS	1
7	17846	Optical Comms	17877	Orbit	2
8	17842	Power + Thermal	17846	Optical Comms	2
9	17842	Power + Thermal	17877	Orbit	4
10	17844	RF Comms	14845	OBDH	1
11	17844	RF Comms	17846	Optical Comms	1
12	17844	RF Comms	17877	Orbit	2

Table 4.1 continuation
Example dataset used for the case study

#	Target Subsystem		Source Subsystem		Linked parameters
	ID	Name	ID	Name	
13	17841	Structure	17840	Bio Payload	4
14	17841	Structure	17838	Mission + Programmatics	1
15	17841	Structure	17846	Optical Comms	4
16	17841	Structure	17877	Orbit	3
17	17841	Structure	17842	Power + Thermal	1

After importing all the data, the program uses it to represent a DSM on the screen as a three-dimensional histogram. This 3D histogram uses data and creates a 3D bar graph of the number of linked parameters between subsystems in a two-dimensional grid. This 3D histogram is considered to be a 2D array of integer amount linked parameters as shown in Table 4.2. Figure 4.12 illustrates the 3D DSM view of the impact scores plotted by using the data from Table 4.2.

Table 4.2
DSM view of the impact scores (or linked parameters) between 9 subsystems

Subsystem	#	1	2	3	4	5	6	7	8	9
Mission + Programmatics	**1**		1	1						
Bio Payload	**2**			4				1		
Structure	**3**		7							
Power + Thermal	**4**		2	1						
AOCS	**5**								1	
RF Comms	**6**									
OBDH	**7**						1			
Optical Comms	**8**			4	2		1	1		
Orbit	**9**			3	4		2		2	

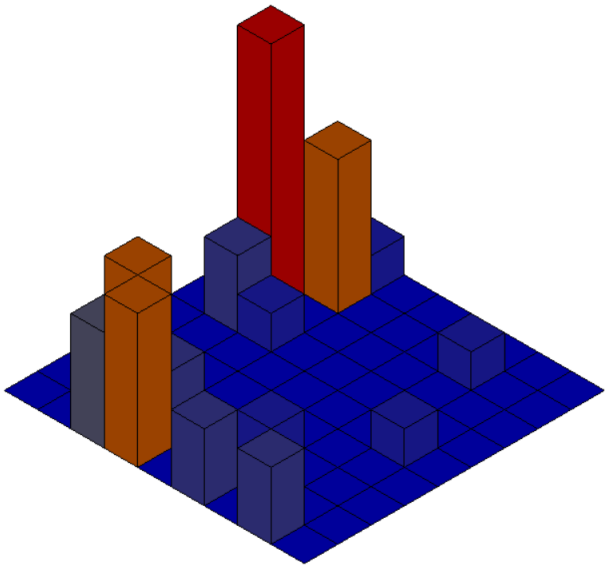

Figure 4.12
3D DSM view of the impact scores (or linked parameters) between 9 subsystems

User can perform a **New risk assessment** from the **File** menu after performing all required manipulations with the impact scores. The result of this operation is a new DSM view of the interfaces between subsystems, now with the risk scores calculated by the method proposed by Garg et al. (2017) which was discussed above in Section 2.2 of Chapter 2 of the literature review and can be described by Equation 2:

$$Interface\ risk_i = \max(L_i, L_j) \cdot \max(I_i, I_j) \qquad (2)$$

The risk scores are being normalized afterwards to the score of 100 being corresponding to the highest interface risk. This new 3D histogram is considered to be a 2D array of the risk scores as shown in Table 4.3. Figure 4.13 illustrates the 3D DSM view of the risk scores plotted by using the data from Table 4.3. In this particular example, all the technologies of the subsystems are considered to have TRL 9.

A potential user has a possibility to retrieve information about each one of the interfaces by clicking on it in the DSM view with right mouse button. For instance, if the user decides to retrieve information from an interface between Bio Payload being the target subsystem and Structure being the source subsystem, they could click with a right mouse button on it and a sub-window with the information appears as shown in Figure 4.14. This information includes names of the target and source subsystems with their IDs in brackets, its impact score, TRL, and risk score.

Table 4.3
DSM view of the risk scores

Subsystem	#	1	2	3	4	5	6	7	8	9
Mission + Programmatics	**1**		14	14						
Bio Payload	**2**			57				14		
Structure	**3**		100							
Power + Thermal	**4**		29	14						
AOCS	**5**								14	
RF Comms	**6**									
OBDH	**7**						14			
Optical Comms	**8**			57	29		14	14		
Orbit	**9**			43	57		29		29	

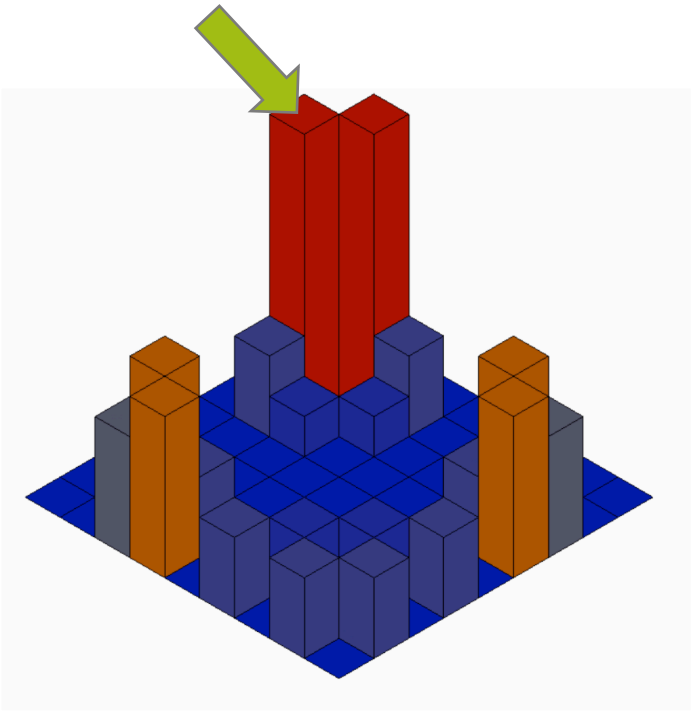

Figure 4.13
3D DSM view of the risk scores between 9 subsystems. The arrow here highlights one of the interfaces for the case study

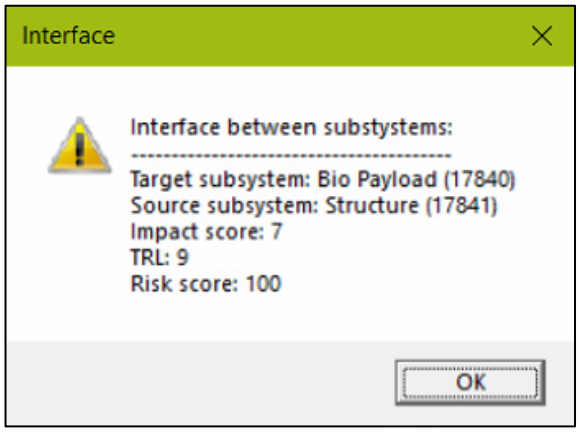

Figure 4.14
Information about the interface highlighted in Figure 4.11 in a separate sub-window

Since it was decided to implement an MDI architecture of the software prototype, user can open several windows with DSM views of impact scores and risk scores. C3D Converter uses the following formats to exchange geometric model data with other systems: STEP, IGES, SAT (ACIS), X_T, X_B (Parasolid), STL, VRML, and JT. STEP, IGES, SAT, X_T, X_B formats transmit the boundary representation of the geometric model. STL and VRML formats transmit the polygonal representation of the geometric model. JT format transmits the hybrid representation (both) of the geometric model. STEP format supports transmit of product and manufacturing information (PMI) (C3D Labs, 2017). Therefore, the user can also open a 3D model of the systems. A user interface in this case appears as illustrated in Figure 4.15.

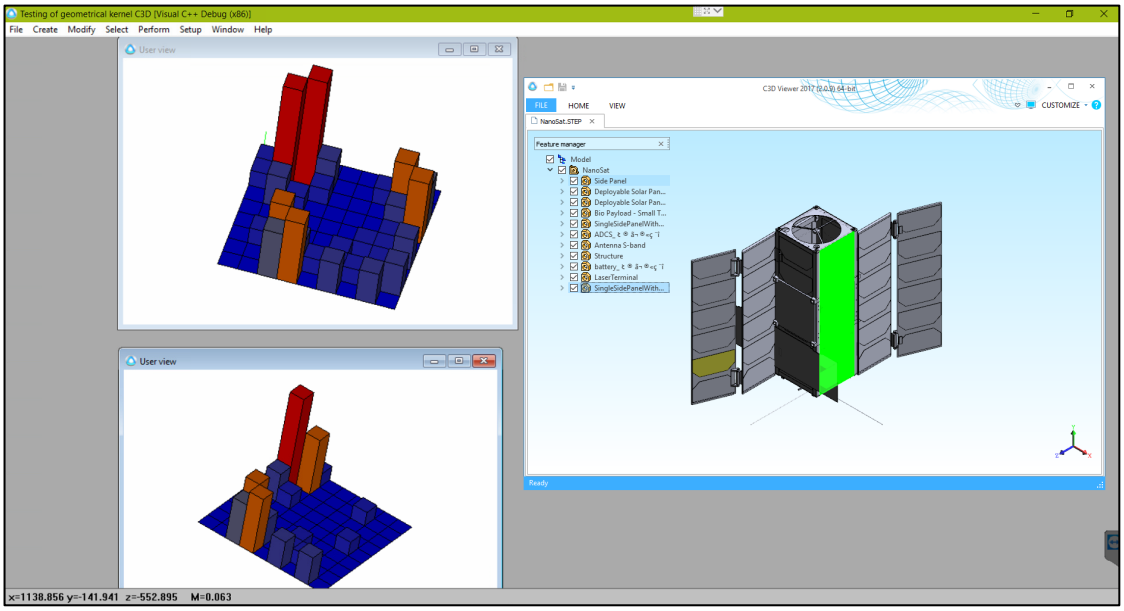

Figure 4.15
User interface of the developed software prototype showing the MDI structure implemented in it

Chapter 5

FUTURE WORK AND CONCLUSIONS

5.1 Avenues of future work

In this section, further research ideas and rooms for improvement will be presented, some of which had to be excluded from this thesis.

Firstly, it would be interesting to see further case studies for the software prototype developed.

It would be interesting to further develop the software prototype to enhance concurrent conceptual design process by using C3D Kernel and C3D Vision features.

It was suggested by one of the R&T team members at Airbus, that the frustration and issues the concurrent design team faced, where things that people do not have appropriate and convenient tools to visualize a system with all the parameters included in it. Thus, it would be interesting to develop a new interface and methodology for data visualization.

Since DSM matrices become larger and more complex, it would be interesting to implement big data analysis and data sciences in order to better identify, capture, and manage information.

It would be interesting to implement DSM^{V} and DSM^{3D} for in the software prototype, since they provide a way to embrace a whole family of products.

It would be interesting to make a full real-time integration of CEDESK with C3D Modeler. The difference in the programming languages between them is an obstacle to achieving that, yet it is feasible in a longer perspective.

It would be interesting to switch to VR to make using DSMs in the conceptual design stage of life cycle more interactive.

It would be interesting to implement the software prototype in a web-based tool/add-on in order to make it available all over the globe.

Figure 5.1 represents a concept of a Next Generation MBSE platform that could emerge from the developed software prototype. This concept has a 3D model of a demonstrator model under development with CEDESK, Opcat, DSM viewer and a plot viewer orbiting it. It would be interesting to see this platform operational.

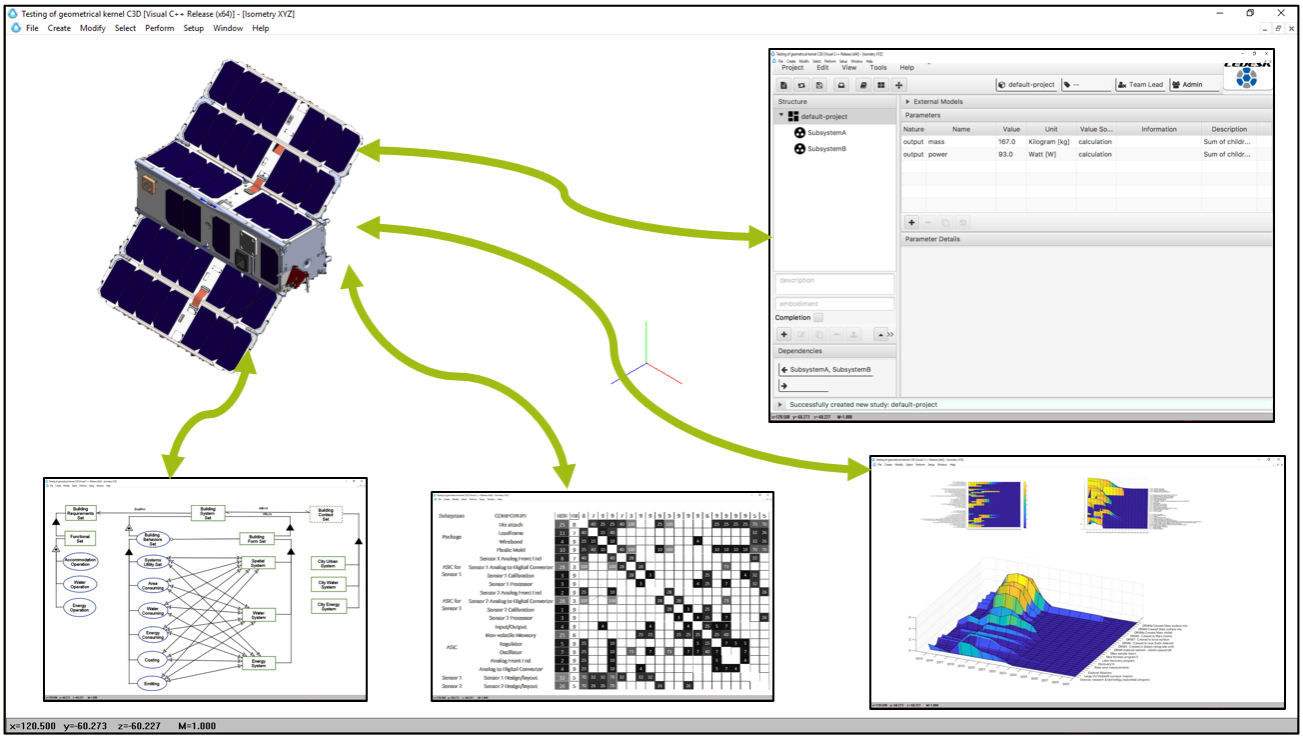

Figure 5.1
A concept of a Next Generation MBSE platform

The developed software prototype allows modelling and evaluation of a demonstrator by representing a 3D model of a system being developed and interfaces in it as 3D DSM matrices. Implementing the OPM methodology proposed by Dori (2003) and covered in Section 2.1 of Chapter 2 could allow a systems designer to even better analyze systems with large amounts of interfaces on their conceptual design stage. This could be crucial for enhancing the development of a software prototype of Digital Factory – a new generation of adaptable factory engineered with knowledge-based engineering systems, which is believed to play a significant role in establishing of the 4th Industrial Revolution as a cyber-physical system. The Digital Factory concept could result to extensively configuration, model, simulate, assess and evaluate items, procedures and system before another industrial facility is constructed or any alteration is really completed on a current framework, keeping in mind the end goal to enhance quality and lessen the time (Canetta et al, 2011).

5.2 Conclusions

The main goal of the research was to study, adapt, and implement some of the current DSM techniques in a software prototype to allow a concurrent engineering design team to represent data from MySQL

databases from CEDESK in a view of three-dimensional DSM matrices. A case study was conducted using the developed software prototype.

The software prototype is not supposed to be a commercial software product ready to be sold and implemented yet, although there might be a potential for that which has to be identified.

The delimitations of the study are that only one main use-case is studied in detail. The results might therefore not be universally applicable to other use-cases. The timeframe of the study was 9 months, resulting in a great amount of data and ideas arising, but not all aspects and perspectives could be accommodated within the scope of the master thesis study.

As a result, it is believed that adding the third dimension to DSM has the potential to allow a systems engineer to comprehensively analyze interfaces in the system, especially in our world where systems become more and more complex. Moreover, a concurrent conceptual design team using the software prototype can experience a more interactive process by developing a parametric model of a system using the CEDESK interface and being able to see a 3D geometrical model of results of their cooperative work, along with DSMs to have a better comprehension of system interfaces and risks in them. It is believed that the developed software prototype has the potential to become the next generation MBSE platform, capable of representing a geometrical model of the system in 3D with supporting tools orbiting the model.

REFERENCES

Alizon, F., Moon, S., Shooter, S. B., and Simpson, T. W. (2007). *Three-Dimensional Design Structure Matrix with Cross-Module and Cross-Interface Analyses*. In: ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 6: 33rd Design Automation Conference, Parts A and B ():941-948. doi:10.1115/DETC2007-34510. Available at: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1604805

Bandecchi, M., Melton, B., Gardini, B., et al. (2000). *The ESA/ ESTEC concurrent design facility*. In: Proceedings of the EUSEC, pp. 329–336. Available at: http://swe.ssa.esa.int/TECEES/spweather/esa_initiatives/spweatherstudies/CDF_study/cdf_paper.pdf (accessed 3 April 2018)

Bellgran, M., and Säfsten, K. (2009). *Production Development: Design and Operation of Production Systems*. Springer, London, UK, ISBN 9781848824942

Bousquet, P. W., Benoist, J., Gonzalez, Fr., Gillen, Ph., Pillet, N., Sire, J.– P., and Vigeant, F. (2005) *Concurrent Engineering at CNES*, pp 1–11. doi: 10.2514/6.IAC–05–D1.3.06

C3D Labs (2017) *C3D Developer Manual*, p. 412.

C3D Labs. *C3D Modeler*. Available at: http://c3dlabs.com/ru/products/modeler/ (accessed 5 November 2017)

Canetta, L., Redaelli, C., and Flores, M. (2011). *Digital Factory for Human-oriented Production Systems* (1st ed.). London: Springer-Verlag London Limited.

Chakrabarti, A., and Srinivasan, V. (2009). *SAPPhIRE – an Approach to Analysis and Synthesis.* In: International Conference on Engineering Design (ICED09), Palo Alto, DS 58–2.

Charney, R. (2005). Programming Tools: UML Tools. Available at: https://www.linuxjournal.com/article/8334 (accessed 26 May 2018)

Cloutier, R. and Bone, M. (2010). *Compilation of SysML RFI – Final Report*. Stevens Institute of Technology. Available at: http://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:omg_rfi_final_report_02_20_2010-1.pdf

CMake. (2017). *CMake*. Available at: https://cmake.org/ (accessed 31 July 2017)

Davis, T. G. (2010). *Java and Mac OS X*. Indianapolis, IN: Wiley.

Deshmukh, M., Wolff, R., Fischer, P., and Gerndt, A. (2015). *Interactive 3D Visualization to Support Concurrent Engineering in the Early Space Mission Design Phase*, CEAS 2015, paper no. 020, pp. 1–8.

Di Domizio, D., and Gaudenzi, P. (2008). *A Model for Preliminary Design Procedures of Satellite Systems*. Concurr. Eng., vol. 16, no. 2, pp. 149–159, 2008.

Donnelly, J. (2015). *5 reasons to use SourceTree for Git*. Available at: https://sagittarius.agency/blog/5-reasons-to-use-sourcetree-for-git (accessed 13 May 2018)

Dori, D. (1995). *Object-Process Analysis: Maintaining the Balance between System Structure and Behavior.* Journal of Logic and Computation. 5 (2): 227–249. doi:10.1093/logcom/5.2.227

Dori, D. (2003). *Object-Process Methodology and Its Application to the Visual Semantic Web*. Presentation, Chicago.

Dori, D. (2011). *Object-process Methodology: A Holistic Systems Paradigm*. New York: Springer.

Ebrahimi, S. M. (2011). *Concurrent Engineering Approaches within Product Development Processes for Managing Production Start-up phase.* (Master). Jönköping University.

Enterprise Architecture. (2009). *Sparx Systems Enterprise Architect*. Available at: http://iea.wikidot.com/sparxsystems-enterprise-architect (accessed 26 May 2018)

Eppinger, S., and Browning, T. (2016). *Design Structure Matrix Methods and Applications* (pp. 74-78). Cambridge: MIT Press.

European Space Agency (ESA). (2012). *What is concurrent engineering?* Available at: http://www.esa.int/Our_Activities/Space_Engineering_Technology/CDF/What_is_concurrent_engineering (accessed 13 April 2018)

European Space Agency (ESA). (2014a). *Open concurrent design tool*. Available at: https://ocdt.esa.int/ (accessed 20 June 2016).

European Space Agency (ESA). (2014b). *CDF*. Available at: http://www.esa.int/Our_Activities/Space_Engineering_Technology/CDF (accessed 26 May 2018)

Fortin, C., McSorley, G., Knoll, D., Golkar, A., and Tsykunova, R. (2017). *Study of Data Structures and Tools for the Concurrent Conceptual Design of Complex Space Systems*. In: Proceedings of the 14th IFIP WG 5.1 International Conference, PLM 2017 Seville, Spain, July 10–12.

Fijneman, M. and Matthyssen, A. (2010) *Application of concurrent design in construction, maritime, education and other industry fields*. In: Proceedings of the 4th international workshop on system & concurrent engineering for space applications (SECESA 2010), Lausanne, 9 June.

Garg, T., Eppinger, S., Joglekar, N., and Olechowski, A. (2017). *Using TRLs and system architecture to estimate technology integration risk.* In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 3: Product, Services and Systems Design, Vancouver, Canada, 21.–25.08.2017.

Golovanov, N. (2014). *Geometric Modeling.* CreateSpace Independent Publishing Platform.

Hermann, M., Pentek, T., and Otto, B. (2016). *Design Principles for Industrie 4.0 Scenarios*. In: System Sciences (HICSS). Hawaii: IEEE.

IBM. (2018). *Rational Rhapsody – Overview – United States.* Available at: https://www.ibm.com/us-en/marketplace/rational-rhapsody (accessed 19 February 2018)

IBM Knowledge Center. (2016). *Rational Rhapsody modeling perspective*. Available at: https://www.ibm.com/support/knowledgecenter/en/SSB2MU_8.1.5/com.ibm.rhp.integ.ides.doc/topics/rhp_r_int_rhp_modeling_perspective.html (accessed 26 May 2018)

INCOSE. (2010). *INCOSE Systems Engineering Handbook v3.2.* Idaho Falls, Idaho.

Knoll, D., Briatore, S., Moreno Aguirre, C., Fursova, A., Akhtyamov, R., Gonzalez, A., Poghosyan, A., and Golkar, A. (2016). *LaserNaut Feasibility Study. Conceptual system study of Skoltech / Harvard Federated "Astronaut-on-a-Chip" Nanosatellite Mission*. Presentation.

Knoll, D. and Golkar, A. (2016). *A coordination method for concurrent design and a collaboration tool for parametric system models*. SECESA 2016, Madrid, (October), pp. 1–11. doi: 10.1177/1063293X17732374

Le Gal, JL. (2013). *Concurrent engineering approach to design mission feasibility studies at CNES.* Presentation, Centre National d'Etudes Spatiales.

McLaughlin, M. (2005). Agile Methodologies for Software Development. Available at: https://www.versionone.com/agile-101/agile-methodologies/ (accessed 16 May 2018)

McSorley, G., Fortin, C., and Huet, G. (2014). DS 77: Proceedings of the DESIGN 2014 13th International Design Conference. In: *International Design Conference* (pp. 1843–1852).

Microsoft Support. (2016). *Excel not responding, hangs, freezes or stops working.* Available at: https://support.office.com/en-us/article/excel-not-responding-hangs-freezes-or-stops-working-37e7d3c9-9e84-40bf-a805-4ca6853a1ff4 (accessed 7 April 2018)

MySQL. (2018). *MySQL Connectors*. Available at: https://www.mysql.com/products/connector/ (accessed 15 May 2018)

NASA. *NASA's Do-It-Yourself Podcast: Rocket Science*. (2014). Available at: https://www.nasa.gov/audience/foreducators/diypodcast/rocket-science-index-diy.html (accessed 28 July 2017)

NASA. (2007). *Systems Engineering Handbook*. Washington, D.C.: National Aeronautics and Space Administration.

NASA JPL. (2001). *Concurrent Engineering Guideline for Aerospace Systems.* Practice No. GD-ED-2204. Available at: https://oce.jpl.nasa.gov/practices/2204.pdf (accessed 26 May 2018)

NASA JPL. (2015). *JPL Team X*. Available at: https://jplteamx.jpl.nasa.gov/ (accessed 26 May 2018)

NASA JPL. (2016). *Open Model Based Engineering Environment*. Available at: http://www.openmbee.org/

NoMagic. (2017). *NoMagic's MBSE Solution*. Available at: http://www.nomagic.com/mbse/ (accessed 14 February 2018)

Project Management Institute. (2008). *Project Management Body of Knowledge (PMBOK)*, Vol. 4.

Roussel, JC., Golkar, A., and Knoll, D. (2017). *Roadmap Creation Process. Airbus Technology Planning and Roadmapping*. Presentation, Airbus.

Schaus, V., Fischer, P., Lüdtke, D., et al. (2010). *Concurrent engineering software development at German aerospace center-status and outlook*. In: 4th international workshop on system & concurrent engineering for space applications, Lausanne, 13–15 October.

Schumann, H., Berres, A., Maibaum, O., and Röhnsch, A. (2008). *DLR's Virtual Satellite approach*. In: 10th International Workshop on Simulation on European Space Programmes, Noordwijk, 7–9 October. Available at: http://elib.dlr.de/56030/1/SESP_2008_Schumann_VirtualSatellite.pdf

Spangelo, S., Kim, H., and Soremekun, G. (2013). *Modeling & Simulation of CubeSat Mission*. Presentation. Available at: http://www.nomagic.com/mbse/images/casestudies/Modeling-and-Simulation_of_CubeSat_Mission.pdf

Sparx Systems. (2018). *UML tools for software development and modelling – Enterprise Architect UML modeling tool.* Available at: http://www.sparxsystems.com

TrustRadius. (2014). *IBM Rational Team Concert Review: Powerful tool but a little clunky.* Available at: https://www.trustradius.com/reviews/ibm-rational-team-concert-2014-06-17-11-45-02 (accessed 19 February 2018)

Tsykunova, R. (2016). *Study of data structures, tools and processes to support concurrent conceptual design of space products* (Master). Skolkovo Institute of Science and Technology.

Ulrich, K. T., and Eppinger, S. D. (2008). *Product Design and Development*. McGraw–Hill, 5th edition.

Unger, D. W., and Eppinger, S. D. (2009). *Comparing product development process and managing risk*. International Journal of Product Development, Vol. 8, No. 4

VersionOne. (2005). The Benefits of Agile Software Development. Available at: https://www.versionone.com/agile-101/agile-software-development-benefits/ (accessed 16 May 2018)

Webster, L. (2017). *Visual Studio IDE, Code Editor, Team Services, & Mobile Center.* Visual Studio. Available at: https://www.visualstudio.com (accessed 31 July 2017)

de Weck, O. (2012). Design Structure Matrix. Lecture, Massachusetts Institute of Technology. Available at: https://ocw.mit.edu/courses/engineering-systems-division/esd-36-system- project-management-fall-2012/lecture-notes/MITESD_36F12_Lec04.pdf (accessed 9 April 2018)

Wheelwright, S. C. (1985). *Product Development and Manufacturing Start-up*. Manufacturing Issue.

Yazdani, B. (1999). *Four Models of Design Definition: Sequential, Design Centered, Concurrent and Dynamic.* Journal Of Engineering Design, 10(1), 25-37. http://dx.doi.org/10.1080/095448299261407

Appendix A: An example of constructing an extrusion body with C3D Modeler

As described in Section 5 of Chapter 3, it is proposed to use C3D Modeler as a tool for geometrical representation. In this Appendix an example of an extrusion operation used in the software prototype is provided for general understanding of the way the API of C3D Modeler works.

In the example bellow, construction of an extrusion body is performed by a C++ function **ExtrusionSolid**.

```
ExtrusionSolid (const MbSweptData & sweptData,
                const MbVector3D & direction,
                const MbSolid * solid1,
                const MbSolid * solid2,
                bool checkIntersection,
                ExtrusionValues & params,
                const MbSNameMaker & names,
                PArray<MbSNameMaker> & cnames,
                MbSolid *& result)
```

The function takes the following input parameters:

- **sweptData** is the data for a curve generator,
- **direction** is the extrusion direction,
- **solid1** is used when option «To next object in forward direction is selected,
- **solid2** is used when option «To next object in forward direction is selected,
- *checkIntersection* is a flag indicating that it is necessary to merge **solid1** and **solid2** bodies subject to checking the intersection,
- *params* are construction parameters,
- names are face names
- cnames are names of curve generator segments

Method output parameter is a constructed body **result**. If successful, the method returns rt_Success, otherwise it returns an error code from MbResultType listing.

It could be noticed that the core objects used in C3D Modeler and represented in Figure 3.4 are basically written to be new types of variables. Figure A.1 represents the data used for construction, as well as the scheme for inheriting the parameters of constructed extrusion body to better visualize

the linkages between different types of variable in this example. This greatly improves the software development process using C3D Modeler as it brings simplicity in working with it.

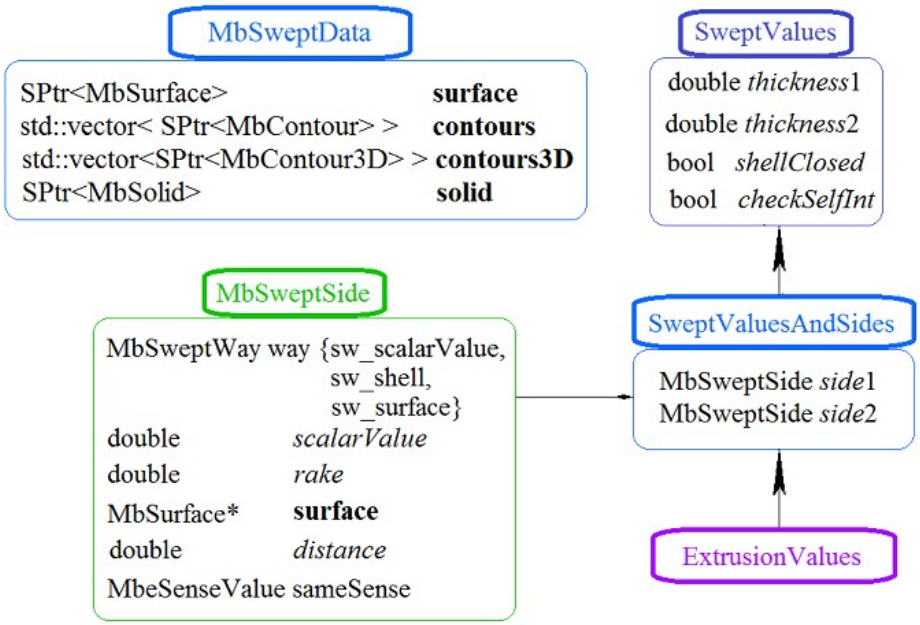

Figure A.1
Data used for construction of an extrusion body and the scheme of inheriting the parameters of constructed extrusion body, adapted from C3D Labs (2017)

A two-dimensional contour and flat surface (`MbPlane`) that can be used for an extrusion are shown in Figure A.2. Figure A.3 represents a thin-walled closed body that was constructed by extrusion based on specified contour parameters. Each contour segment has a corresponding face of the body, its name is taken from the corresponding element of `cnames[0]` name generator embedded into C3D Modeler.

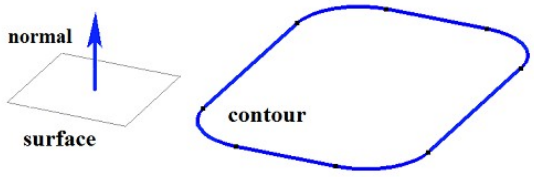

Figure A.2
A two-dimensional contour and flat surface that can be used for an extrusion, adapted from C3D Labs (2017)

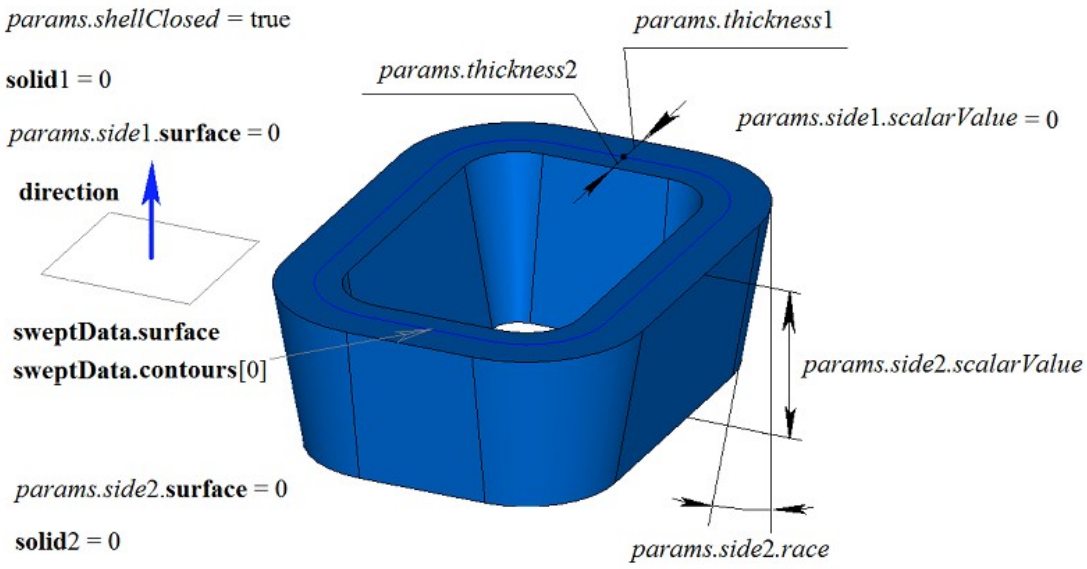

Figure A.3
A thin-walled closed body that was constructed by extrusion based on specified contour parameters, adapted from C3D Labs (2017)

Appendix B: The Python script used for establishing a secure connection between an application working on C++ and MySQL databases

As described in Section 2.4 of Chapter 4, it is proposed to use the Python Driver for MySQL for establishing a secure connection between the software prototype working on C++ and MySQL databases operated by CEDESK.

Below is the code of the Python script with the MySQLConnector/C++ library included. Relevant comments are included.

```
# !/usr/bin/python
# version.py - Fetch and display the MySQL database server version.

# File name: MySQL_connection.py
# Author: Nikita Letov
# Date created: 03/12/2018
# Date last modified: 04/04/2018
# Python Version: 2.7.14


# Import MySQL connector modules.
from mysql.connector import MySQLConnection, Error
# Import Pandas library.
import pandas as pd
# Import the CSV hanling library.
import csv
from collections import defaultdict
# Import the library for addressing Windows Command Line.
import optparse

# Importing user credentials for accessing a MySQL database.
parser = optparse.OptionParser('usage%prog' +
                               '-username <username>' +
                               '-password <pass> ' +
                               '-host <host> ' +
                               '-database <database> ')

parser.add_option('-u', dest = 'username', help = 'The Username for
authentication.')
parser.add_option('-p', dest = 'password', help = 'The password for
authentication.')
parser.add_option('-a', dest = 'host',     help = 'The host to interact
with.')
parser.add_option('-d', dest = 'database', help = 'The database to run')

(options,args) = parser.parse_args()


def query_with_fetchone():
    try:
        # Open a database connection.
        ccnx = MySQLConnection()
```

```
        ccnx.connect(user     = options.username,
                     password = options.password,
                     host     = options.host,
                     database = options.database)
        # Prepare a cursor object using cursor() method.
        cursor = ccnx.cursor()

        # Execute the SQL query using execute() method.
        cursor.execute ("CREATE OR REPLACE VIEW parameter_dependencies AS "
                        "SELECT "
                        "y.id AS system_id, "
                        "y.`name` AS system_name, "
                        "s1.id AS target_subsystem_id, "
                        "s1.`name` AS target_subsystem_name, "
                        "p1.id AS target_parameter_id, "
                        "p1.`name` AS target_parameter_name, "
                        "p1.`value` AS target_parameter_value, "
                        "u.`name` AS target_parameter_unit, "
                        "p2.id AS source_parameter_id, "
                        "p2.`name` AS source_parameter_name, "
                        "s2.id AS source_subsystem_id, "
                        "s2.`name` AS source_subsystem_name "
                        "FROM "
                        "parametermodel p1 "
                        "JOIN "
                        "unit u ON p1.unit_id = u.id "
                        "JOIN "
                        "subsystemmodel s1 ON p1.parent_id = s1.id "
                        "JOIN "
                        "systemmodel y ON s1.parent_id = y.id "
                        "JOIN "
                        "parametermodel p2 ON p2.id = p1.valueLink_id "
                        "JOIN "
                        "subsystemmodel s2 ON s2.id = p2.parent_id "
                        "WHERE "
                        "p1.valueLink_id IS NOT NULL; ")
        cursor.execute ("SELECT "
                        "`target_subsystem_id`, "
                        "`target_subsystem_name`, "
                        "`source_subsystem_id`, "
                        "`source_subsystem_name`, "
                        "COUNT(source_parameter_id) AS linked_parameters "
                        "FROM "
                        "parameter_dependencies "
                        "WHERE "
                        "system_name = \"LaserNaut\" "
                        "GROUP BY `target_subsystem_name` ,
`source_subsystem_name`; ")

        # Allocating lists with imported values.
        target_subsystem_id = []
        target_subsystem_name = []
```

```
        source_subsystem_id = []
        source_subsystem_name = []
        linked_parameters = []

        # Adding imported values in the lists.
        row = cursor.fetchone()
        while row is not None:
            target_subsystem_id += [row[0]]
            target_subsystem_name += [str([row[1]])[13:-3]]
            source_subsystem_id += [row[2]]
            source_subsystem_name += [str([row[3]])[13:-3]]
            linked_parameters += [row[4]]
            row = cursor.fetchone()

        # Creating a dataframe with the data.
        d = {'Target subsystem id'     :   target_subsystem_id,
             'Target subsystem name'   :   target_subsystem_name,
             'Source subsystem id'     :   source_subsystem_id,
             'Source subsystem name'   :   source_subsystem_name,
             'Linked parameters'       :   linked_parameters}

        df = pd.DataFrame(data = d)
        df = df[['Target subsystem id',
                 'Target subsystem name',
                 'Source subsystem id',
                 'Source subsystem name',
                 'Linked parameters']]

        filename = 'dsm_data.csv'
        df.to_csv(filename)
        print(df)


    except Error as e:
        print(e)

    finally:
        # Closing the connection.
        cursor.close()
        ccnx.close()


if __name__ == '__main__':
    query_with_fetchone()
```

As a result of running the Python above from the C++ application, the software prototype gets data represented in Table B.1 to operate with.

Table B.1
Data imported from an example MySQL database by running the Python script from the C++ software prototype

Subsystem ID	Subsystem name	Parameter ID	Parameter name	Parameter value	Parameter unit
17838	Mission + Programmatics	17839	mission lifetime	0.5	year
17877	Orbit	17913	earth radius	6378	Kilometer
17877	Orbit	17914	orbit altitude	600	Kilometer
17877	Orbit	17915	inclination	97.7924	Degree
17877	Orbit	17916	SMA	6878.14	Kilometer
17843	AOCS	17984	Pointing Accuracy	0.1	Degree
17843	AOCS	17985	Peak power consumption	5.46	watt
17843	AOCS	17986	Mass	0.865	kilogram
17846	Optical Comms	17987	Pointing accuracy	0.1	Degree
17846	Optical Comms	17988	Power consumption	5.064	watt
17846	Optical Comms	17989	Mass	0.3	kilogram
17846	Optical Comms	18000	Orbit	600	Kilometer
17846	Optical Comms	18008	earth radius	6378	Kilometer
17838	Mission + Programmatics	18052	Deployment velocity	2	metre per second
17838	Mission + Programmatics	18053	Min delay between 2 launches	1	minute
17841	Structure	18171	Max Payload Size X	0.093	Meter
17841	Structure	18172	Max Payload Size Y	0.093	Meter
17841	Structure	18173	Max Payload Size Z	0.2	Meter
17841	Structure	18174	Max Payload Mass	2.5	Kilogram
17841	Structure	18175	Momentum of Inertia X	0.02733417	Kilogram Meter Squared
17841	Structure	18176	Momentum of Inertia Y	0.01500083	Kilogram Meter Squared
17841	Structure	18177	Momentum of Inertia Z	0.03666667	Kilogram Meter Squared
17841	Structure	18192	Launch Vibration Amplitude	7.84	Meter Per Second Squared
17841	Structure	18193	Launch Vibration Frequency	100	Hertz

Table B.1 continuation
Data imported from an example MySQL database by running the Python script from the C++ software prototype

Subsystem ID	Subsystem name	Parameter ID	Parameter name	Parameter value	Parameter unit
17841	Structure	18194	Launch Acceleration	42.14	Meter Per Second Squared
17840	Bio Payload	18216	Capsule size X	0.093	Millimeter
17840	Bio Payload	18217	Capsule size Y	0.093	Millimeter
17840	Bio Payload	18218	Capsule size Z	0.093	Millimeter
17840	Bio Payload	18219	Capsule Mass	0.9	Kilogram
17840	Bio Payload	18220	Average Power	12	watt
17840	Bio Payload	18221	Bio Experiment Data Rate	0.05	kilobit per second
17840	Bio Payload	18222	Bio Experiment Duration	28.105	day
17846	Optical Comms	18247	Experiment Telemetry generation	0.1	kilobit per second
17846	Optical Comms	18248	Required radio satellite interlink datarate	0	kilobit per second
17842	Power + Thermal	18263	Bus Voltage	11.1	Volt
17842	Power + Thermal	18278	EPS Dimensions X	75	Millimeter
17842	Power + Thermal	18279	EPS Dimensions Y	70	Millimeter
17842	Power + Thermal	18280	EPS DImensions Z	45	Millimeter
17841	Structure	18311	Aerodynamic Disturbances	1.81E-05	Newton
17841	Structure	18384	Inclination	97.7924	Degree
17841	Structure	18385	Orbit Altitude	600	Kilometer
17841	Structure	18386	Satellite Velocity	7612.608	Meter Per Second
17841	Structure	18387	Mission Lifetime	0.5	year
17841	Structure	18388	Bio Payload Size X	0.093	Millimeter
17841	Structure	18389	Bio Payload Size Y	0.093	Millimeter
17841	Structure	18390	Bio Payload Size Z	0.093	Millimeter
17841	Structure	18391	Laser Payload Size X	95	millimetre

Table B.1 continuation
Data imported from an example MySQL database by running the Python script from the C++ software prototype

Subsystem ID	Subsystem ID	Subsystem ID	Subsystem ID	Subsystem ID	Subsystem ID
17841	Structure	18392	Laser Payload Size Y	95	millimetre
17841	Structure	18393	Laser Payload Size Z	85	millimetre
17841	Structure	18433	Bio Payload Mass	0.9	Kilogram
17841	Structure	18441	Laser Payload Mass	0.3	kilogram
17844	RF Comms	18468	Power consumption (RX mode)	0.13	watt
17844	RF Comms	18469	Average access time for Moscow per day	2055.15755	second
17844	RF Comms	18470	Power consumption (TX mode)	3.75	watt
17844	RF Comms	18471	Transceiver mass	100	Gram
17844	RF Comms	18476	UHF antenna mass	50	Gram
17844	RF Comms	18483	Time in TX mode (per day)	10080	Second
17845	OBDH	18491	Mass	80	Gram
17845	OBDH	18492	power consumption	1	watt
17844	RF Comms	18502	orbit altitude	600	Kilometer
17877	Orbit	18509	orbit velocity	7612.608	Meter Per Second
17877	Orbit	18514	average access time to ground station	2055.15755	second
17842	Power + Thermal	18551	Orbit altitude	600	Kilometer
17842	Power + Thermal	18552	Daylight duration	3673.658	second
17842	Power + Thermal	18553	Orbit period	5801.23	second
17877	Orbit	18569	orbit period	5801.23	second
17877	Orbit	18570	sunlight time	3673.658	second
17842	Power + Thermal	18578	Orbit inclination	97.7924	Degree
17846	Optical Comms	18642	Portion of the orbit dedicated to experiment	10	percent
17846	Optical Comms	18643	Dimensions Z	85	millimetre
17846	Optical Comms	18671	Dimensions X	95	millimetre

Table B.1 continuation
Data imported from an example MySQL database by running the Python script from the C++ software prototype

Subsystem ID	Subsystem ID	Subsystem ID	Subsystem ID	Subsystem ID	Subsystem ID
17846	Optical Comms	18672	Dimensions Y	95	millimetre
17844	RF Comms	18725	Portion of time dedicated to laser experiment	10	percent
17844	RF Comms	18762	Maximum distance between satellites	841.519556	kilometre
17840	Bio Payload	18775	Max Payload size X	0.093	Meter
17840	Bio Payload	18776	Max Payload size Y	0.093	Meter
17840	Bio Payload	18777	Max Payload size Z	0.2	Meter
17840	Bio Payload	18778	Total Payload mass	2.5	Kilogram
17840	Bio Payload	18798	Inner Satellite T Sunside	0.15009647	Degree Celsius
17840	Bio Payload	18799	Inner satellite T darside	-11.22916	Degree Celsius
17840	Bio Payload	18800	Launch Vibrations Amplitude	7.84	Meter Per Second Squared
17840	Bio Payload	18801	Launch Vibrations Frequency	100	Hertz
17840	Bio Payload	18802	Launch acceleration	42.14	Meter Per Second Squared
17840	Bio Payload	18805	Mission lifetime	0.5	year
17877	Orbit	18918	computed lifetime	6.4	year
17838	Mission + Programmatics	19073	Starting distance between satellites	50	kilometre
17842	Power + Thermal	19084	Power System Mass	0.63783	Kilogram
17841	Structure	19127	Integrated Bus Mass	1.5	Kilogram
17842	Power + Thermal	19176	Solar Cells Mass	0.18783	Kilogram
17841	Structure	19186	Solar Arrays Mass	0.18783	Kilogram
17842	Power + Thermal	19257	Temperature Zenith Wall	0.15009647	Degree Celsius
17842	Power + Thermal	19262	Temperature Nadir Wall	-11.22916	Degree Celsius
17843	AOCS	19298	Average power consumption	1	Watt

Table B.1 continuation
Data imported from an example MySQL database by running the Python script from the C++ software prototype

Subsystem ID	Subsystem ID	Subsystem ID	Subsystem ID	Subsystem ID	Subsystem ID
17846	Optical Comms	19515	LaserTransmitter_diameter	10	Millimeter
17846	Optical Comms	19516	LaserTrasmitter_powerDissipation	2.1252	Watt
17842	Power + Thermal	19552	Laser Base diameter	10	Millimeter
17842	Power + Thermal	19567	Laser Power Disipation	2.1252	Watt
17845	OBDH	19623	Amount of data to transmit per day	1.64794922	Megabyte
17844	RF Comms	19632	Amount of data to transmit per day	1.64794922	Megabyte
17845	OBDH	19633	Bio payload data rate	0.05	kilobit per second
17845	OBDH	19634	Laser payload data rate	0.1	kilobit per second
17842	Power + Thermal	19680	Disipator cilinder height	90.4010843	Millimeter

Appendix C: Code of the main CMake file used in the study

As described in Section 2.2 of Chapter 4, it is proposed to CMake for building of the software prototype.

In this project, CMake files were adapted and expanded with new functionality relevant for the study.

Below is the code for the main CMake file used to build the whole application. Relevant comments are included.

```
# Minimum CMake version required for building
CMAKE_MINIMUM_REQUIRED(VERSION 3.2.2)
PROJECT(Test)
SET(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/..)

# Libraries
SET(Math_MATH_LIBRARY
    optimized ${CMAKE_BINARY_DIR}/../Release/c3d.lib
    debug ${CMAKE_BINARY_DIR}/../Debug/c3d.lib)

SET(Math_LIBRARIES ${Math_MATH_LIBRARY} CACHE FILEPATH "" FORCE)

# Checking if PC runs 32-bit version
IF(CMAKE_HOST_WIN32)
  SET(Test_OUTPUT "Test")
ENDIF()

# Unicode build
OPTION(MathTest_USING_UNICODE "Enable Unicode support" ON)

# Math names are not included by default
OPTION(MathTest_QT "Use Qt Library" OFF)
IF(MathTest_QT)
  FIND_PACKAGE( Qt5Core )
  FIND_PACKAGE( Qt5Gui )
  FIND_PACKAGE( Qt5OpenGL )
ENDIF()

# Source C++ files for creation of geometrical objects.
SET(Create_SRC
  ./Create/test_constraint.cpp
  ./Create/test_arc.cpp
  ./Create/test_assembly.cpp
  ./Create/test_contour.cpp
  ./Create/test_curve.cpp
  ./Create/test_plane.h
  ./Create/test_line.cpp
  ./Create/test_multiline.cpp
  ./Create/test_multithreading.cpp
  ./Create/test_point.cpp
  ./Create/test_curve3d.cpp
  ./Create/test_sheet.cpp
  ./Create/test_shell.cpp
  ./Create/test_map.cpp
```

```
  ./Create/test_solid.cpp
  ./Create/test_space.h
  ./Create/test_surface_.cpp
  ./Create/test_point3d.cpp
  ./Create/test_surface.cpp
  ./Create/test_user.cpp
  )
SOURCE_GROUP(Create FILES ${Create_SRC})

# Source C++ files for editing geometrical objects.
SET(Edit_SRC
  ./Edit/test_edit_contour.cpp
  ./Edit/test_edit_curve.cpp
  ./Edit/test_edit_plane.h
  ./Edit/test_edit_multiline.cpp
  ./Edit/test_edit_curve3d.cpp
  ./Edit/test_edit_map.cpp
  ./Edit/test_edit_solid.cpp
  ./Edit/test_edit_space.cpp
  ./Edit/test_edit_space.h
  ./Edit/test_edit_surface.cpp
  )
SOURCE_GROUP(Edit FILES ${Edit_SRC})

# Source C++ files of the GUI.
SET(Main_SRC
  ./Main/test_draw.cpp
  ./Main/test_draw.h
  ./Main/test_frame.h
  ./Main/test_frame1.cpp
  ./Main/test_frame2.cpp
  ./Main/test_frame3.cpp
  ./Main/test_service.cpp
  ./Main/test_service.h
  ./Main/test_variables.cpp
  ./Main/test_variables.h
  )
SOURCE_GROUP(Main FILES ${Main_SRC})

# Source C++ files for performing geometrical calculations.
SET(Make_SRC
  ./Make/test_computation.cpp
  ./Make/test_computation.h
  ./Make/test_converter.cpp
  ./Make/test_converter.h
  ./Make/test_mates.cpp
  ./Make/test_mates.h
  ./Make/test_rendering.cpp
  ./Make/test_rendering.h
  ./Make/test_rendering_.cpp
  ./Make/test_rendering_context.cpp
  ./Make/test_rendering_context.h
  )
SOURCE_GROUP(Make FILES ${Make_SRC})

# Source C++ files for managing processes in the application.
SET(Manager_SRC
```

```
  ./Manager/test_comanager.h
  ./Manager/test_comanager.cpp
  ./Manager/test_gr_draw.cpp
  ./Manager/test_gr_draw.h
  ./Manager/test_manager.cpp
  ./Manager/test_manager.h
  ./Manager/test_property.cpp
  ./Manager/test_property.h
  ./Manager/test_property_title.h
  ./Manager/test_style.h
  ./Manager/test_temporal_plane.cpp
  ./Manager/test_temporal_plane.h
  ./Manager/test_tree.cpp
  ./Manager/test_tree.h
  ./Manager/test_tree_filter.cpp
  ./Manager/test_tree_filter.h
  ./Manager/test_temporal.cpp
  ./Manager/test_temporal.h
  ./Manager/test_window.cpp
  ./Manager/test_window.h
  ./Manager/test_window_add.cpp
  ./Manager/test_window_find.cpp
  ./Manager/test_window_move.cpp
  )
SOURCE_GROUP(Manager FILES ${Manager_SRC})

# Source C++ files for the sample control.
SET(Samples_SRC
  ./Samples/test_samples.h
  ./Samples/test_sample_attributes.cpp
  ./Samples/test_sample_user_attributes.h
  ./Samples/test_sample_user_attributes.cpp
  ./Samples/test_sample_solid_elementary.cpp
  ./Samples/test_sample_solid_splitting.cpp
  ./Samples/test_sample_parametric_sketch.cpp
  ./Samples/test_sample_read_write_constraints.cpp
  ./Samples/test_sample_spinning_block.cpp
  ./Samples/test_sample_wireframe.cpp
  )
SOURCE_GROUP(Samples FILES ${Samples_SRC})

# Optional source C++ files for building using the Qt libraries.

IF(MathTest_QT)
  SET(QtTest_SRC
    ./Qt/test_main.cpp
    ./Qt/test_main_window.h
    ./Qt/test_main_window.cpp
    ./Qt/test_child_window.h
    ./Qt/test_child_window.cpp
    ./Qt/test_property_dialogs.h
    ./Qt/test_property_dialogs.cpp
    ./Qt/test_main_dialogs.h
    ./Qt/test_main_dialogs.cpp
  )
  SOURCE_GROUP(QtTest FILES ${QtTest_SRC})
```

```
# MOC files.

  SET(MOC_Files
    ./Qt/test_main_window.h
    ./Qt/test_child_window.h
    ./Qt/test_property_dialogs.h
    ./Qt/test_main_dialogs.h
  )

  QT5_WRAP_CPP(MOC_Gui
    ${MOC_Files}
  )

# RCC files.

  SET(RCC_Files
    ./Qt/test.qrc
  )

 QT5_ADD_RESOURCES(RCC_Gui ${RCC_Files})
 INCLUDE_DIRECTORIES(${QT_USE_FILE})

# Optional source C++ files for building without using the Qt libraries.

ELSE()

  SET(WinTest_SRC
    ./Win/test.cpp
    ./Win/test.h
    ./Win/test.rc
    ./Win/test_dialogs.cpp
    ./Win/test_frame.cpp
    ./Win/test_info.h
    ./Win/test_prompt.h
    ./Win/test_set_filter.cpp
    ./Win/test_set_property.cpp
    ./Win/test_set_tree.cpp
    ./Win/test_std_afx.cpp
    ./Win/test_std_afx.h
    ./Win/test_window_graphic.cpp
  )
  SOURCE_GROUP(WinTest FILES ${WinTest_SRC})

ENDIF()

# Source sub-directories.
INCLUDE_DIRECTORIES(${CMAKE_BINARY_DIR}/../Include
      ${Test_SOURCE_DIR}/Create
      ${Test_SOURCE_DIR}/Edit
      ${Test_SOURCE_DIR}/Main
      ${Test_SOURCE_DIR}/Make
      ${Test_SOURCE_DIR}/Manager
      ${Test_SOURCE_DIR}/Samples
      ${Test_SOURCE_DIR}/Win
      ${Test_SOURCE_DIR}/Qt
)
```

```
ADD_DEFINITIONS(
  -D__TEST_ONLY__
  )

IF(MSVC)
IF(MathTest_USING_UNICODE)
  ADD_DEFINITIONS(
    -DUNICODE
    -D_UNICODE
    )
ENDIF(MathTest_USING_UNICODE)
ELSE()
  ADD_DEFINITIONS(
    -std=c++0x
    )
ENDIF()

IF(MathTest_QT)
  ADD_DEFINITIONS(
    -D__USE_QT__
    )
ENDIF()

IF(MathTest_WITH_VLD)
  ADD_DEFINITIONS(
    -DENABLE_VLD
    )
ENDIF()

IF(MSVC)
  SET(CMAKE_CXX_FLAGS_DEBUG   "${CMAKE_CXX_FLAGS_DEBUG_INIT} -D_DEBUG -D_DRAWGI /Zi
/W4")
  SET(CMAKE_CXX_FLAGS_RELEASE "${CMAKE_CXX_FLAGS_RELEASE_INIT} -D_SECURE_SCL=0 /W4")
ELSE()
  SET(CMAKE_CXX_FLAGS_DEBUG  "${CMAKE_CXX_FLAGS_DEBUG_INIT} -D_DEBUG -D_DRAWGI -Wno-
deprecated-declarations")
  SET(CMAKE_CXX_FLAGS_RELEASE "${CMAKE_CXX_FLAGS_RELEASE_INIT} -D_SECURE_SCL=0 -Wno-
deprecated-declarations")
ENDIF()

IF(MathTest_QT)
  IF(MSVC)
    SET(CMAKE_EXE_LINKER_FLAGS "${CMAKE_EXE_LINKER_FLAGS_INIT} /ENTRY:mainCRTStartup")
  ENDIF()

  ADD_EXECUTABLE(${Test_OUTPUT} WIN32
    ${Create_SRC}
    ${Edit_SRC}
    ${Main_SRC}
    ${Make_SRC}
    ${Manager_SRC}
    ${Samples_SRC}
    ${QtTest_SRC}
    ${MOC_Gui}
    ${RCC_Gui}
    )
```

```
ELSE()
  ADD_EXECUTABLE(${Test_OUTPUT} WIN32
    ${Create_SRC}
    ${Edit_SRC}
    ${Main_SRC}
    ${Make_SRC}
    ${Manager_SRC}
    ${Samples_SRC}
    ${Win_SRC}
    ${WinTest_SRC}
    )
ENDIF()

IF(MathTest_QT)
  QT5_USE_MODULES( ${Test_OUTPUT} Core Gui OpenGL )
ENDIF()

IF(MSVC)
  TARGET_LINK_LIBRARIES(${Test_OUTPUT}
    ${Math_LIBRARIES}
    ${VLD_LIBRARIES}
    ${QT_LIBRARIES}
    opengl32
    glu32
    comctl32
    )
ELSE()
  TARGET_LINK_LIBRARIES(${Test_OUTPUT}
    ${Math_LIBRARIES}
    ${VLD_LIBRARIES}
    ${QT_LIBRARIES}
    GL
    GLU
    )
ENDIF()
```

Appendix D: Excerpts of the code used for 3D DSM plotting

In this appendix some of the functions participating in building 3D DSM diagrams are presented.

Not all aspects and functions of the code could be presented here because of its complexity and size. However, functions for establishing a MySQL connection through the Python script, for building a simple cuboid that can be used for building a 3D histogram, and for the actual 3D DSM representation have been excerpted as ones that allow users to see the result of their work.

```
//-----------------------------------------------------------------------------
// Establish MySQL connection
// ---
void SQLconnection()
{
	TCHAR username[256];
	TCHAR password[256];
	TCHAR host[256];
	TCHAR database[256];

	// Accessing the Python script
	std::string filepath = "C:/Users/nikita.letov/Documents/MySQL_test_app/Build/";
	std::string filename = "MySQL_connection.py";
	std::string command = "cd " + filepath + " && python " + filename;

	if (GetString(IDS_ENTER_SQL_USERNAME, _T(""), username, STRINGLENGTH)) {}
	if (GetString(IDS_ENTER_SQL_PASSWORD, _T(""), password, STRINGLENGTH)) {}
	if (GetString(IDS_ENTER_SQL_HOST,     _T(""), host,     STRINGLENGTH)) {}
	if (GetString(IDS_ENTER_SQL_DATABASE, _T(""), database, STRINGLENGTH)) {}

	SetWaitCursor(true);

	command += " -u ";
	command += TcharToString(username);
	command += " -p ";
	command += TcharToString(password);
	command += " -a ";
	command += TcharToString(host);
	command += " -d ";
	command += TcharToString(database);

	FILE* in = _popen(command.c_str(), "r");
	_pclose(in);

	// Building 3D DSM
	BuildDSM3D();

	// Refreshing the screen
	TestVariables::viewManager->RefreshModel();

	filename = "dsm_data.csv";
	command = "cd " + filepath + " && start " + filename;

	SetWaitCursor(false);
}
```

```
//-----------------------------------------------------------------------------
// Asses risks
// ---
void RiskAssessment()
{
	SetWaitCursor(true);

	BuildDSMrisk();

	// Refreshing the screen
	TestVariables::viewManager->RefreshModel();

	SetWaitCursor(false);
}

//-----------------------------------------------------------------------------
// Build a cuboid by 3 points base and height.
// ---
void buildCubeBy3Point(MbCartPoint3D p1, MbCartPoint3D p2, MbCartPoint3D p3, double h,
uint32 color)
{

	// Allocating an array of points.
	MbCartPoint3D p[4];
	p[0] = p1;
	p[1] = p2;
	p[2] = p3;

	// Type of the solid: block (cuboid).
	ElementaryShellType type = et_Block;


	SArray<MbCartPoint3D> points(4, 1);
	points.Add(p[0]);
	points.Add(p[1]);
	points.Add(p[2]);

	MbSNameMaker names(503, MbSNameMaker::i_SideNone, 0);


	// Allocating memory for a solid.
	MbSolid * solid = NULL;

	// Avoiding the heght to be less than the metric precision.
	if (h < METRIC_PRECISION)
		h = METRIC_PRECISION;

	MbVector3D to;
	MbCartPoint3D p0;

	// Building the vector needed for building the cuboid.
	to = MbVector3D(p[0], p[1]) | MbVector3D(p[0], p[2]);

	// Normalizing the vector to the height.
	to.Normalize();
	to *= h;
```

```
	p0.Move(to);
	points.Add(p0);

	// Creating the cuboid.
	::ElementarySolid(points, type, names, solid);

	// Checking, adding to the screen and coloring.
	if (solid != NULL)
	{
		TestVariables::viewManager->AddObject(TestVariables::ELEMENTARY_Style,
solid);
		solid->SetColor(color);
	}
}



//-----------------------------------------------------------------------------
// Build DSM3D
// ---
void BuildDSM3D()
{
	std::string filename = "dsm_data.csv";
	std::ifstream file(filename); // declare file stream:
http://www.cplusplus.com/reference/iostream/ifstream/

	int CSVrows = countLinesInCSV(filename);

	// Allocate arrays.
	int*         dsm_id = new int[CSVrows];
	std::string* target_subsystem_name = new std::string[CSVrows];
	int*         target_subsystem_id = new int[CSVrows];
	std::string* source_subsystem_name = new std::string[CSVrows];
	int*         source_subsystem_id = new int[CSVrows];
	int*         linked_parameters = new int[CSVrows];

	// Access the data retrieved by the Python script and allocating it in arrays.
	for (int row = 0; row <= CSVrows; ++row)
	{
		std::string line;
		std::getline(file, line);
		if (!file.good())
			break;

		std::stringstream iss(line);
		for (int col = 0; col < 6; ++col)
		{
			std::string val;
			std::getline(iss, val, ',');

			if (row != 0)
			{
				switch (col)
				{
				case 0:
					dsm_id[row - 1] = StringToInt(val);
```

```
                                    break;
                              case 1:
                                    target_subsystem_id[row - 1] = StringToInt(val);
                                    break;
                              case 2:
                                    target_subsystem_name[row - 1] = val;
                                    break;
                              case 3:
                                    source_subsystem_id[row - 1] = StringToInt(val);
                                    break;
                              case 4:
                                    source_subsystem_name[row - 1] = val;
                                    break;
                              case 5:
                                    linked_parameters[row - 1] = StringToInt(val);
                                    break;
                              }
                        }
                  }
            }

            // IDs of all the subsystems
            int* all_subsystem_id = new int [2*CSVrows];
            std::copy(target_subsystem_id, target_subsystem_id + CSVrows,
all_subsystem_id);
            std::copy(source_subsystem_id, source_subsystem_id + CSVrows, all_subsystem_id
+ CSVrows);
            int subsystems_number = 0;
            std::set<int> sa(all_subsystem_id, all_subsystem_id + 2*CSVrows);
            subsystems_number = sa.size() - 1;
            delete [] all_subsystem_id;

            // Names of all the subsystems
            std::string* all_subsystem_name = new std::string[2 * CSVrows];
            std::copy(target_subsystem_name, target_subsystem_name + CSVrows,
all_subsystem_name);
            std::copy(source_subsystem_name, source_subsystem_name + CSVrows,
all_subsystem_name + CSVrows);
            std::set<std::string> sn(all_subsystem_name, all_subsystem_name + 2 * CSVrows);
            delete[] all_subsystem_name;

            // Redefining the arrays with all susbsystem IDs and names
            all_subsystem_id = new int[subsystems_number];
            all_subsystem_name = new std::string[subsystems_number];

            for (int i = 0; i < subsystems_number; i++)
            {
                  std::set<int>::iterator iter_id = sa.begin();
                  std::set<std::string>::iterator iter_name = sn.begin();

                  std::advance(iter_id, i + 1);
                  std::advance(iter_name, i + 1);

                  all_subsystem_id[i] = *iter_id;
                  all_subsystem_name[i] = *iter_name;

                  myfile << *iter_id;
```

```
			myfile << ",";
			myfile << *iter_name;
			myfile << "\n";
		}
		myfile.close();

		// DSM array
		int** dsm = new int*[subsystems_number];
		for (int i = 0; i < subsystems_number; ++i)
			dsm[i] = new int[subsystems_number];
		for (int i = 0; i < subsystems_number; ++i) // for each row
		{
			for (int j = 0; j < subsystems_number; ++j) // for each column
			{
				dsm[i][j] = 0;
			}
		}

		for (int i = 0; i < CSVrows; ++i) // for each row
		{
			int dsm_x = FindIndex(all_subsystem_id, subsystems_number,
target_subsystem_id[i]);
			int dsm_y = FindIndex(all_subsystem_id, subsystems_number,
source_subsystem_id[i]);
			if (dsm_x >= 0 && dsm_y >= 0)
				dsm[dsm_x][dsm_y] = linked_parameters[i];
		}

		// Finding the highest risk
		int max_risk = 0;
		for (int i = 0; i < CSVrows; ++i)
		{
			if (linked_parameters[i] > max_risk)
				max_risk = linked_parameters[i];
		}
		float medium_risk = (float)max_risk / 2;

		int cube_side = 100;
		int risk_increment = 100;

		// Coloring the histogram
		for (int i = 0; i < subsystems_number; ++i) // for each row
		{
			for (int j = 0; j < subsystems_number; ++j) // for each column
			{
				if (dsm[i][j] > -1)
				{
					float red_color = 0;
					float green_color = 0;
					float blue_color = 255;
					if (dsm[i][j] <= medium_risk)
					{
						red_color = 255* (float)dsm[i][j] / max_risk;
						green_color = red_color;
						blue_color = 255 - red_color;
					}
					else
```

```
                        {
                                red_color = 255;
                                green_color = 255 * (1 - (float)dsm[i][j] /
max_risk);
                                blue_color = 0;
                        }
                        buildCubeBy3Point(MbCartPoint3D(i * cube_side, 0, j *
cube_side),
                                MbCartPoint3D(i * cube_side, 0, (j + 1) * cube_side),
                                MbCartPoint3D((i + 1) * cube_side, 0, j * cube_side),
                                risk_increment * dsm[i][j],
                                RGB(red_color, green_color, blue_color));
                }
        }
    }
}
```