Software Communication in Computer Aided Engineering

Tuomas Ruippo

School of Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 12.8.2017

Thesis supervisor:

Prof. Kari Tammi

Thesis advisor:

M.Sc. Janne Ojala

Aalto University
School of Engineering

AALTO UNIVERSITY
SCHOOL OF ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Tuomas Ruippo

Title: Software Communication in Computer Aided Engineering

Date: 12.8.2017 Language: English Number of pages: 7+79

Department of Mechanical Engineering

Professorship: Engineering Design

Supervisor: Prof. Kari Tammi

Advisor: M.Sc. Janne Ojala

The growing number of computer aided design software suites used in the engineering processes of heavy industry. Recently, communicating information between these software suites has gained more interest. Duplication of data for each software separately creates risk of quality issues as well as slows the product development cycles.

The aim of this thesis was to identify the best method for software communication between PTC Creo and Autodesk Revit. Based on the gaps identified in the existing methods, a custom solution prototype was developed to better understand the quality of the existing methods. This custom solution prototype was compared with the existing methods using established engineering design methodology.

This thesis found the custom solution prototype to be the best method for software communication between the software suites in question. The custom solution prototype scored 79.2% of the maximum with the best existing method only achieving a score of 72.9%. The higher evaluation can be attributed especially to the high integrability and low cost of the custom solution compared to existing methods. Owning the solution also enables achieving a competitive edge.

Keywords: CAD, CAE, BIM, Software Communication

AALTO-YLIOPISTO
INSINÖÖRITIETEIDEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Tuomas Ruippo

Työn nimi: Ohjelmistojen välinen tiedonvälitys tietokoneavusteisessa suunnittelussa

Päivämäärä: 12.8.2017 Kieli: Englanti Sivumäärä: 7+79

Konetekniikan laitos

Professuuri: Koneensuunnittelu

Työn valvoja: Prof. Kari Tammi

Työn ohjaaja: DI Janne Ojala

Raskas teollisuus hyödyntää kasvavissa määrin erilaisia suunnitteluohjelmistoja osana suunnitteluprosessejaan. Ohjelmistojen kasvava määrä on lisännyt kiinnostusta tiedon sujuvaan välittämiseen näiden ohjelmistojen välillä. Tiedon kahdentaminen erikseen jokaista ohjelmistoa varten lisää laatuvirheiden riskiä ja hidastaa tuotekehityssyklejä.
Tämän työn tarkoitus oli tunnistaa paras mahdollinen menetelmä tiedon välittämiseen PTC Creon ja Autodesk Revitin välillä. Olemassa olevien menetelmien puutteista johtuen osana työtä tuotettiin oma menetelmäprototyyppi. Tätä menetelmäprototyyppiä verrattiin olemassa oleviin menetelmiin käyttäen vakiintuneita suunnittelumenetelmiä.
Oma menetelmäprototyyppi paljastui vertailussa parhaaksi menetelmäksi tiedon välittämiseen ohjelmistosta toiseen. Oma menetelmäprototyyppi sai pisteitä 79,2 % maksimista lähimmän olemassa olevan menetelmän jäädessä 72,9 % maksimipisteistä. Prototyypin saama korkeampi pistemäärä voidaan yhdistää erityisesti hyvään yhdistettävyyteen muihin järjestelmiin sekä menetelmän matalaan kustannukseen. Tiedonvälitysmenetelmän omistajuus antaa myös mahdollisuuden saavuttaa kilpailuetua.

Avainsanat: CAD, CAE, BIM, Software Communication

Preface

I want to thank supervisor Kari Tammi and the advisors Janne Ojala and Jukka Hämäläinen for their guidance. Thanks also to Ville Lähteinen who provided valuable support for the IPR section.

Otaniemi, 30.6.2017

Tuomas Ruippo

Contents

Abbreviations

Abbreviations

AEC	Architecture, Engineering and Construction
APAC	Asia-Pacific Region
API	Application Programming Interface
BIM	Building Information Model(ing)
CAD	Computer Aided Design
CAE	Computer Aided Engineering
Capex	Capital expenditure
COM	Component Object Model
CVCS	Centralized Version Control System
DCVS	Decentralized Version Control System
FEA	Finite Element Analysis
HG	Mercurial
IP	Intellectual property
IPR	Intellectual property rights
JIT	Just In Time
JVM	Java Virtual Machine
Opex	Operating expenditure
PDM	Product Data Management
PDF	Portable Document Format
PLM	Product Lifecycle Management
R&D	Research and Development
SVN	Subversion
VB	Visual Basic
VCS	Version Control System
XML	EXtensible Markup Language

1 Introduction

1.1 Background

As integrated Computer-Aided Engineering (CAE) solutions have become commonplace in the industry, software communication has become even more important [34]. In this thesis, software communication is defined as an interface for separate software suites to communicate using the same data without human intervention. Software communication reduces the risk of human error and enables a reduction in tedious, repetitive manual copying tasks. Often, the CAE ecosystem contains multiple software suites for specialized engineering tasks, such as one software suite for Computer-Aided Design (CAD), another software suite for Finite Element Analysis (FEA) and a third software suite for Enterprise Resource Planning (ERP).

In an integrated CAE ecosystem, it is mandatory that these systems seamlessly share information. The ability to share information eliminates the need for separately configuring the data for each system. Information sharing leads to benefits in the quality, costs and agility of product development and change management [4, 5].

In the mechanical engineering industry, a core component of CAE is 3D CAD. Although CAD and other components of CAE, such as Computer Aided Manufacturing (CAM), have been extensively used for decades in the mechanical engineering industry[3], the architecture, engineering and construction (AEC) industry has been slow to follow suit. The traditional approach to both architectural design and structural engineering has been to use 2D design, while more developed information management tools, such as Building Information Modeling (BIM), have only recently begun to gain acceptance [18]. In 2007, this use of outdated tools and systems in the AEC industry was estimated to have contributed to 500 billion dollars of waste in the United States alone [29]. BIM reduces waste because most changes are done using a detailed model in the early stage, when changes are cheap. Conventional civil engineering does many changes on-site, when the cost of change is higher. This difference is described in Figure 3 on page 2.

The reluctance of the AEC industry to implement BIM is often explained by four quite simple factors. First, the complexity of a large building with all of its systems is quite high compared to a small-medium mechanical product. Because of this complexity, IT hardware requirements are also quite high, even by today's standards. Most larger buildings are also quite unique in design sharing few obvious features with other buildings designed even by the same architect or structural engineer. A fourth reason for this reluctance is the allocation of work in a building project to multiple subcontractors, which poses challenges in finding other compatible communication media than traditional 2D drawings.

Nevertheless, these factors often presented as obstacles in implementing BIM are actually problems best solved using BIM. For example, the complexity of AEC is better handled using a centralized, consistent information model rather than numerous unconnected files of varying type. Using BIM to consistently document the designs makes it easier for building projects to share design features, as this enables architects and engineers to search for similar designs rather than having to

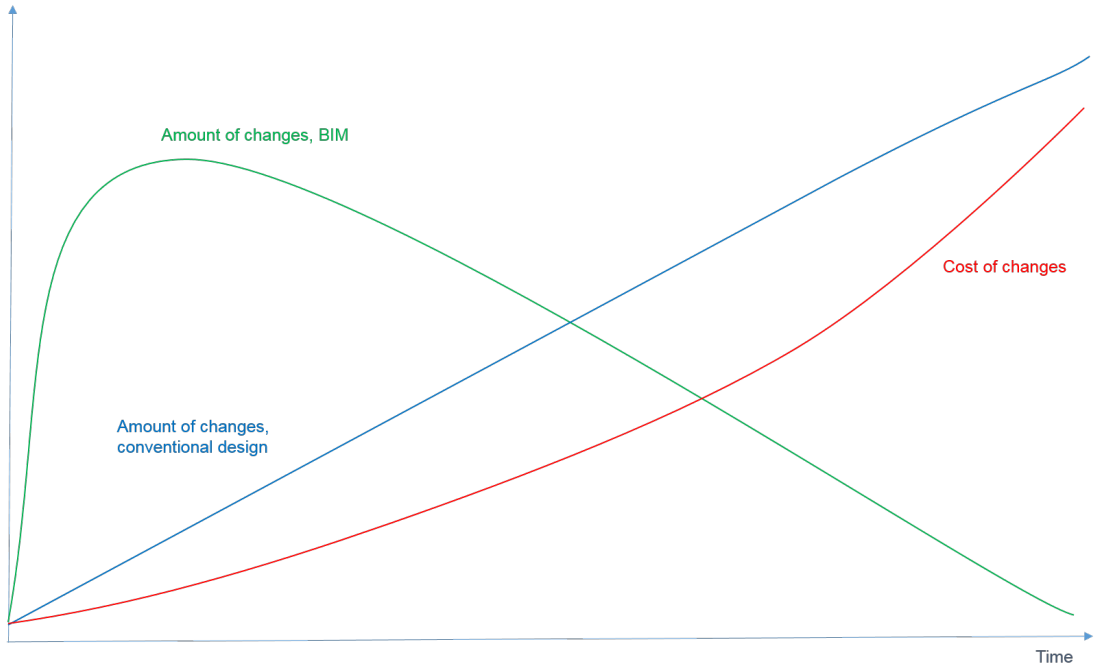

Figure 1: Targeted impact on change cost using BIM

hunt through bookshelves of drawings and project information. Declining IT costs would allow smaller subcontractors to bid for projects requiring BIM collaboration.

BIM has thus slowly evolved into a key tool for managing the building process. Contrary to what is commonly thought, BIM is not just a 3D model, but rather a kind of a visual container for storing as well as relaying metadata along with the actual 3D information [12, 24]. Thus, BIM is more than a counterpart for CAD in the building industry, and that the 3D model would not even be required for a model of building to be BIM.

However, Eastman et al. [12] define BIM as requiring a spatial (3D) representation of the building, even though that is not the full definition. From a mechanical engineering viewpoint, BIM is actually somewhat analogous to Product Data Management (PDM) or Product Lifecycle Management (PLM) systems. Looking at the list of PLM system capabilities provided by Aram and Eastman [2], items such as design data visualization, merging of partial models from various sources and manual notification requests are core functionalities enabled by the use of BIM. Because of these factors, building information modeling is more of a complete philosophy change in the building industry than merely a design tool. The 3D design of a building is rather a subcomponent of the building information model than the actual model. [12]

Despite the rising interest in BIM solutions witnessed by the building industry, the separation between mechanical CAE software and full-bred BIM software creates challenges for companies supplying mechanical equipment to buildings. As customers would also like to have details of the equipment contained in the BIM,

mechanical equipment suppliers face a choice between recreating their product information at some detail level into BIM compatible formats and implementing software communication between the two data collections.

KONE is at an interesting crossroads regarding BIM. The product offering is mechanical in nature, but the products represent integral parts of a building. Thus, although KONE is designing mechanical products, the methods currently in use are compliant with and often derived from methods used in the AEC industry.

In anticipation of the growing impact of BIM on the AEC industry and subsequently on the market, KONE has done preliminary research on how its customers see the demands posed by using the BIM approach on the elevator, escalator and door design. An earlier research paper [22] suggests that BIM will certainly pose new challenges for KONE when communicating different aspects of the elevator, escalator and door designs to the customer. Also discussed in the paper is the fact that as KONE product models are not a part of the construction design as such, some elements of BIM design, including but not limited to cost, should be excluded from the scope of KONE BIM content.

Earlier research on integrating BIM design into the whole product design process of KONE suggested that it would be necessary to examine the possibility of managing more design tasks using a single model. The research showed, however, that no perfect fit exists for the company's needs on the market today. Since this research, the customer needs dictating the properties of engineered BIM content have been analyzed more accurately. As a result, the different approaches to create BIM and their strengths and weaknesses can be more precisely evaluated.

1.2 Research Objective

The aim of this study is to develop a method for software communication between a mechanical CAE software suite, PTC Creo, and a BIM software suite, Autodesk Revit. The master data and design are created in PTC Creo, and BIM is used as an output for communicating product information to the customer. As the product to be engineered is mechanical in nature, using a mechanical CAE software suite provides engineering tools that are rarely included in BIM software.

To ensure maximum compatibility with customer systems, the method for software communication should provide an output that resembles as closely as possible an accepted output generated natively in a BIM software suite. To achieve this goal, three existing options were identified:

1. A commercial application for software communication
2. Native support for BIM output in PTC Creo
3. Conversion of components from PTC Creo to Autodesk Revit

Conversion of only components without full incorporation of the assembly conversion was identified as a minimum requirement for an effective solution. It was decided that a tailor-made application for achieving the objectives of software communication

should be prototyped as a part of this thesis. This decision allows for more accurate estimation of the investment related to creating the final application as well as for benchmarking the solution against the other approaches in order to gather a business case for the possible final development of the solution.

The tailor-made application developed in this thesis will be evaluated by comparing it to the three existing approaches in terms of providing consistent information to the customer at the lowest effort and cost possible, as well as its security, quality and management of information.

1.3 Scope

Several exclusions have been made to provide a well-defined research structure. As this study focuses on the communication between different software architectures, actual BIM design is excluded from this research. Only those BIM topics will be studied that explicitly relate to the conversion of models from one format to another.

As mechanical CAD and CAE software suites and their use is more well-established than their BIM counterparts, mechanical CAD and CAE review are excluded from the scope of this thesis. This is justified given that thee centrality of CAD has already been well accepted and does not need similar assurance as the future centrality of BIM.

Mechanical CAD modeling is not discussed beyond analyzing the requirements that the conversion process imposes on the models to be converted. Existing models are simply used as a reference in validating the functionality of different software communication methods. The goal is to provide a generic solution, well portable to all types of models.

Using software communication to convert a 3D model from a mechanical CAE software suite into a BIM software suite creates the opportunity to use original Research and Development (R&D) design models to produce order specific approval documentation. Although this possibility is briefly discussed in Chapter 7, further analysis is beyond the scope of this study.

For the component conversion approach, the process of creating BIM separately is assumed to be on an optimal level for a separate solution. The improvement of the separate process is not included in the scope of this thesis.

Although the products of KONE are arranged in groups of centrally controlled equipment, this thesis concentrates only on the topic of separate pieces of equipment. Thus, the assembly of groups and the issues related to them are not considered in this thesis.

As found in the literature review, existing research fails to answer the research question of this thesis in an acceptable fashion. The specificity of the problem at hand requires extensive original research in the field related to the research problem.

1.4 Structure of Thesis

This thesis has been structured as follows: First, the relevant literature in fields of BIM, software communication, intellectual property, software development and

solution variant evaluation is studied to establish the relevance of this research as well as the theoretical foundation of the research. Second, the research methodology is defined, mostly related to the solution variant selection process as well as the software engineering process. Third, an analysis is made of the existing methods as well as the development and architecture of the custom solution. Fourth, the different approaches are compared based on the selected evaluation criteria, finally followed by a discussion section of conclusions and future development.

2 Literature Review

The previous chapter established the background and state of the art concerning software communication related to BIM. This chapter analyzes relevant literature concerning BIM, software communication, intellectual property and the engineering design of solutions in general. With respect to BIM, the chapter finds that the benefits of BIM are globally recognized. However, these benefits do not necessarily equate to a wide demand for BIM at KONE.

The chapter also finds that the insubstantial body of knowledge existing in software communication shows an increasing need for such communication, but currently few implementations are available. The intellectual property rights of such implementations are discussed and found to be non-restrictive. However, the studied literature suggests that no intellectual property rights should be sought for a future development either.

Finally, the chapter concludes that software projects are struggling to complete on time and budget. Therefore, it was decided to apply well-proven concepts used by engineering design in general to software development as well.

2.1 BIM

It has been predicted that BIM will revolutionize the AEC industry. Such a revolution would not only introduce cost reductions and gains, thus improving current industry practices, but would also introduce completely new themes, impossible to achieve using conventional methods. Some researchers, such as Young et al. [48], even go as far as stating that without familiarizing oneself with the processes and tools of BIM, even one's career and company are at peril. BIM is often associated with such bold assertions. These statements should be rigorously studied to find the true value behind them.

A fairly recent paper by Li et al. [27] states that BIM, especially in its 4D form, has extensive use in the AEC industry. The authors list the usual information integration, easy sharing and collaboration benefits as the main factors driving BIM adoption in the industry. This view is quite interesting for several reasons. First, the authors are not only from China, but are also representing other Pacific region countries. Additionally, their paper [27] focuses on more significant projects, either in size, as in the case of Wuhan Expo Centre, or significance as in the case of the Shanghai Disaster Control Centre. These projects are mainly located in China, with one project being located in Taiwan [27].

The study by Li et al. [27] is interesting in light of the general experiences of KONE concerning BIM adoption in the Asia-Pacific region (APAC) and especially China. From KONE's perspective, although BIM adoption is quite high in regions such as Singapore and Australia, China seems to be somewhat slower in adopting the new technology. Another interesting factor is the focus of their paper [27] on large-scale, non-residential projects. If the AEC industry in China finds added value using BIM in large-scale projects and moves to more widespread adoption of BIM, the volume of BIM production by KONE will increase exponentially. The volume

and size of residential projects alone are very large in China, in spite of the recent industry cool-down.

Three reasons can be speculated for the contradiction between the experiences of KONE and the situation analysis by Li et al. [27]. First, the current status of BIM adoption in China might be on a level where the AEC industry does use BIM but does not require that all specialty equipment be placed in the model. In such a situation, KONE would not see BIM demand for their products in China, while the AEC industry would still be implementing BIM. The use of BIM would probably then focus on the structural aspect of building design, general architectural planning or other areas where building related specialty equipment can be omitted from the model while still being able to benefit from the use of BIM.

Another reason for the low demand by the Chinese for BIM at KONE is that the AEC industry of China might concentrate on applying BIM to the kind of larger projects described by Li et al. [27]. Thus the large volumes of KONE in the Chinese residential segment and smaller-scale projects would not correlate to a high demand for BIM in the region. This can be interpreted as only part of the solution to the contradiction at hand, since KONE is also a major player in the larger-scale projects.

Another explanation for the lack of correlation between the demand in BIM seen by KONE and the demand projected based on Li et al. [27] is that KONE might be failing to successfully market the BIM solutions it offers and has thus been unsuccessful in turning the customer need for the models into sales of the solutions. This would naturally be a negative scenario for KONE, since the competition might be succeeding on the market because of KONE's failure. A third reason would be that Li et al. [27] are overstating the maturity of the AEC industry to take advantage of developments in BIM. One should maintain a certain skepticism when evaluating new technologies, since there is often a large amount of optimism and marketing hype related to new, exciting concepts. Here, an analogy could be drawn to additive manufacturing (AM), often colloquially referred to as 3D printing. AM technologies are sometimes presented as something that will revolutionize the retail business completely, which might not represent the complete truth [41].

Li et al. [27] have listed the benefits resulting from adopting BIM for each project. They discovered six main items: precision when designing complicated features and the possibility of clash detection, emergency equipment evacuation management, regulation compliance, an integrated facility monitoring system and maintenance database, collaboration facilitation in the pre-design stage, and maintenance scheduling.

The benefits found by Li et al. [27] cannot be described as surprising, but what is quite confusing is the lack of elaboration on the more vague items, such as the upside that BIM provides compared to 2D design in regulation compliance and the rescue of equipment in emergencies. These benefits do not seem obvious to a specialty equipment BIM practitioner, making the paper look exactly like the speculated marketing hype mentioned earlier.

What is intriguing, however, in the study by Li et al. [27] is the benefits section of the very first case, Chong Qing International Circus. The case study concentrates on the challenge posed by the design having a very complicated feature,

requiring the use of an advanced CAE software suite, Dassault CATIA. The authors establish the difficulty of modeling a complicated parametric feature with BIM design tools, approving instead the decision to use a more advanced modeling tool and then exporting the output to Autodesk Revit. More recent developments, such as Revit/Dynamo, partly address the need to communicate complicated parametric features from more advanced modeling tools [9], but the identified need is still interesting. Figure 2 on page 8 shows the software communication, with the CATIA model on the left and the consolidated Revit model on the right.

Figure 2: Chong Qing International Circus City modeled with CATIA and Revit [27]

Even though the models by KONE are far less complex than this case, the philosophy is the same. Although the output needs to be compatible with Autodesk Revit, the selection of the actual modeling/engineering tool should not be dictated by the output. Instead, the optimal tools should be used separately for each stage of work.

When studying the implementation of BIM for a specialty equipment provider, one aspect that cannot be ignored is the challenge of providing BIM training for design engineers not familiar with the AEC methodology. Already in 2006, Woo [47] identified a gap in the way that educational institutes approach the subject of BIM in curricula related to AEC. Although the paper establishes the paradigm change from simple line-based CAD systems to more modern solutions, e.g, BIM as a matter of fact, the paper also postulates that the best practices for educating future professionals in the AEC industry were quite immature at the time. [47]

Woo [47] also notes that a significant pedagogical challenge lies in the way that BIM cannot simply be taught as a fancy method of moving the design from 2D to 3D. The subject is rather a complete paradigm change in the approach to the design and engineering process as a whole. Based on more recent experience, the educational method has probably found a more structured approach to teaching BIM, but this basic problem remains. As Woo [47] phrases the problem, in addition to requiring knowledge in the actual CAD side of the environment, BIM requires high construction expertise as well.

A more recent study by Lee et al. [25] agrees that teaching existing courses as they are and adding BIM to the curriculum as a course or two does not address the issue properly. Instead, BIM should be integrated into all courses in order to fully understand the nature of BIM usage [25]. It is easy to agree. Simple 3D modeling can be taught in a dedicated course, but as mentioned earlier, 3D is not the full nature of BIM, since BIM affects the working practices everywhere in the industry.

If only the minimum objective is met and a method is found to provide software communication to effectively convert the PTC Creo component models into Autodesk Revit component models, engineers are required to use Autodesk Revit to separately create a product model for customer communication purposes. Although Autodesk Revit user interfaces have evolved since the publication of the paper by Woo [47], Autodesk Revit functions might still be difficult to learn at first even after gaining experience in other CAD systems.

Based on the BIM literature review, this thesis seems justified. The adoption of BIM as a widely used design tool seems to be widespread enough to make winning contracts to the AEC industry difficult if BIM cannot be easily provided. Without direct integration of BIM into the engineering process, the design cannot be communicated to the customer quickly enough. Even more importantly, collaboration in the engineering process of the building requires a capability to respond swiftly to change requests, since completely separating engineering from BIM can hinder this capability. The customers will then likely move towards more agile competitors.

2.2 Software Communication

An unsubstantial body of knowledge exists specifically to address the software communication between CAE software suites and BIM software suites. Several reasons for this can be speculated. Not many companies and their products operate deep in the interface between mechanical and civil engineering or architecture, let alone products of a highly complex nature.

Mechanical equipment delivered to be fitted as part of a building often has a relatively simple interface, such as the main unit of the air conditioning system. If the system and its interface is very simple, the system can be communicated to BIM by simply exporting it as a single item into a vendor-neutral CAD format that can be read by a BIM software suite.

If there are more complex items in the system, such as the air conditioning piping, they might be rather well addressed in a BIM software suite. This kind of software generally supports for example piping quite well and may include a good API to model piping automatically based on coordinate and dimension data alone. However, a need for advanced software communication between PTC Creo and Autodesk Revit has been identified by suppliers of equipment with a complicated building interface.

Although the specific kind of software communication sought for by KONE is not well addressed by existing literature, work on other kinds of software communication can be found. For example Ojala [34] studied software communication between PTC Creo and a multi-body simulation (MBS) software suite. The studied software communication is analogous to the solution studied by this thesis.

Two US patent applications also exist for somewhat similar solutions. In the first, Glunz et al. [15] propose a “Method and system for creating 3d models from 2d data for building information modeling (bim)”. The proposed method is actually quite interesting because many traditional companies are dealing with the problem of converting legacy 2D drawings to modern 3D models. A significant demand exists for services and tools that provide this type of functionalities. However, it is not directly

comparable to the research objective at hand, because the input in the scope of this thesis is a 3D model. In the proposed method, 2D data is combined to automatically generate a 3D model in various formats, including Autodesk Revit.

The other patent application by Glunz et al. [14] is of greater interest in the context of this thesis. The patent application [14] proposes a "Method and system for creating composite 3d models for building information modeling (bim)", which by subject is very close to the topic of this thesis. Glunz et al. proposal consists of retrieving several 3D models in different formats and from different vendors and then combining those models together to create a new model that did not previously exist.

The method should be studied very closely in case the component conversion solution is selected. In that scenario, the constraint data for compiling a main assembly 3D model out of individual components could be parsed in a method similar to the one examined in this thesis. The difference could be that the patent claims specifically mention that the 3D models are from multiple vendors and in multiple formats. The solution proposed by this thesis will make use of 3D models from a single vendor only.

2.3 Intellectual Property Considerations

The software communication solution proposed by this thesis would form a backbone for the customer deliverable generation for KONE. As a result, the solution has a high business impact and the intellectual property issues related to such a solution should be carefully considered.

As patents are not awarded globally, these considerations should be applied in the jurisdiction(s) where the solution is sought to be applied. KONE is acting globally, so at least the most important market areas need to be taken into account in the intellectual property considerations. Jurisdictions usually prevent the application of devices or deliverables which infringe a patent even if the actual production had taken place elsewhere [17]. However the patentability study should not be limited to a specific jurisdiction, because the applied requirements for patentable matter vary considerably between different jurisdictions.

First consideration around the intellectual property rights should naturally be the study of the "freedom to operate". If there are no valid patents which would restrict the use of a certain method or device, the freedom to operate exists. The next consideration should be whether there is something to patent in the solution. Halt et al. [17] define patentability by four requirements, focusing on the patenting regulations in the United States. An invention should meet four identifiers:

1. Of patentable subject matter
2. Useful, rarely contested in most technical fields
3. Novel, as in not available in public prior art
4. Non-obvious, at least not to a person with ordinary skill in the subject art

The European Patent Convention (EPC) has a similar definition. Article 52, paragraph 1 of the EPC [33] reads

> European patents shall be granted for any inventions, in all fields of technology, provided that they are new, involve an inventive step and are susceptible to industrial application.

The analogy is quite straight-forward with that of the USC definition. The EPC also takes a position regarding the patenting of software. Article 52, paragraph 2 of the EPC reads

> The following in particular shall not be regarded as inventions within the meaning of paragraph 1: ... (c) schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers;

In the United States, the situation is not as clear. The United States Code Title 35 [36] does not have a mention on the eligibility of a computer program as the subject matter of a patent. The eligibility becomes then very much a matter of interpretation. This is also evident in the extensive case law history in the US courts.

The courts have invalidated numerous software patents which have sought to declare proprietary quite fundamental functionalities of webpages for example. A case example would be the Amazon OneClick patent, which was partly invalidated in 2007, 8 years after issue. [7] The patent basically claimed ownership of storing customer information upon login to enable the user to make purchases with a single click using the existing customer information [17]. To a European observer, this seems quite bizarre, given that the patent is very broadly targeting a common approach to computer programming.

The reason to even consider patents issued in the United States might not be obvious. KONE is not based in the United States and could easily run the applications elsewhere as well, but US patents protect the holder of the patent from the import of an accused invention by others. KONE has previously been reluctant to challenge the validity of this regulation since whether or not the outputs of software are considered import of the solution itself seems unclear.

Another kind of patent that exists in the United States is a design patent. Design patents protect the shape and/or surface ornament of an article, which does not fall under the protection of a utility patent. [17] As the solution proposed in this thesis is not customer facing and has a very limited user interface based on the standard console applications in Microsoft Windows, there is no need to consider a design patent for this solution.

The third patent in existence in the United States is a plant patent. Plant patents protect companies that practice the breeding and cultivation of plants from competition. [17] Naturally, plant patents are not interesting in the context of this thesis.

Existence of patents in the same field should also be studied. Even if it seems that no parts of the solution can be patented in the area of application, in this case the European Union, it is possible that patents still exist in other areas. Particularly

the United States is known for the existence of patent holding companies or non-practicing entities. These are companies that base their business models on ownership of patents and aggressive infringement claims without actually exploiting the patents for manufacturing or service supply themselves.

The combination of numerous non-practicing entities and the lack of clear rules within the patent legislation in the United States, makes it crucial to understand the threat of infringing an existing patent, especially in that market area. As the proposed solution is not to be used as a public facing application, the risk might be somewhat mitigated. If the method used to produce the models is not publicly disclosed, it might be quite difficult for any other patent holders to identify a possible infringement of their patent, let alone prove its usage in a patent feud.

Another possible strategy against non-practicing entities is heavily used by another Finnish technology company, Nokia. Nokia is known for routinely publishing company research papers for inventions and developments that have not been deemed to be of business value to the company itself. Doing so establishes the inventions as so called prior art. A technical solution which can be found in prior art cannot be considered new and as a result prior art is grounds for patent ineligibility or invalidation.

Halt et al. [17] have defined six kinds of prior art:

- The invention is published by another entity than the patent applicant
- The invention is publicly used by another entity than the patent applicant
- The invention is described in a patent by anyone
- The invention is on sale by anyone
- The invention is described in a published in print
- The invention is otherwise made available to the public

All of the above items constitute prior art in case of applicability anywhere in the world before the effective filing date of the patent application [17]. However Halt et al. [17] note that the inventor, joint inventor or another entity that received the subject matter directly or indirectly from the inventor may use, sell or otherwise make the invention publicly available less than one year prior to the effective filing date of the patent application without having the subject matter be considered prior art. This practice is known as the “grace period”. The applicability of the grace period outside of the United States is not clarified by Halt et al. [17]. Upon review with the patents department of KONE, the grace period was confirmed not to apply in Europe.

KONE’s patent department was contacted to estimate the need to mitigate the possibilities of a non-practicing entity to limit the use of the solution through patenting. Their estimation was that the risk is not big enough to warrant measures towards mitigating actions. If it seems that the solution will provide a considerable competitive edge over the competition, measures to protect the source code as other

forms of intellectual property should be taken even if the solution is not patented as such.

It is not only important to consider the intellectual property implications of the software solution itself. As BIM is a method for communicating directly with the customer, CAD software communication is also a matter of intellectual property rights (IPR) management. The IPR point of view cannot be ignored when evaluating the use of CAE component models to generate BIM component models. Sharing BIM component models includes an inherent risk of sensitive or business critical information leaking to competitors or otherwise malevolent parties. This risk needs to be mitigated or eliminated. Because of the risk of information leakage, it is important to value solution variants that display a robust method for managing the IP critical parts of the models and information.

The requirement to protect the detailed data regarding the components is quite well compatible with the requirement to keep the size of the models at a reasonable level. There are then two reasons to simplify the geometry to a level where it does not require as much space and at the same time protect the detail design. The detail limitation can be achieved already in the input CAE models with the use of simplified representations.

Two patent applications in the field of software communication related to BIM were previously identified. The first dealt with 3D output from 2D input and as such is not interesting from the point of view of this thesis. The second by Glunz et al. [14] was very close to the scope of this thesis. Even though this patent application has been rejected, the rejection is non-final. As a result the patent application and its applicability to the solutions studied in this thesis should be subject to research. Whether possible patent infringements should be considered is naturally of great importance.

The solution proposed by Glunz et al. [14] is based on gathering 3D objects in different formats and then creating a composite model based on these 3D objects and a set of parameters and rules. The solution is using a dynamic link library application (DLL or .dll) as the basis for its function. The solution can be applied in the BIM and CAE software suites studied in this thesis. [14]

Halt et al. [17] describe the claims of a patent application elementary in the protection scope definition of the applied patent. Claims are described in literature as “metes and bounds” defining the rights of the patent holder to prevent others from taking advantage of the invention the patent is being applied for. [17] Thus to determine the strength of an existing patent against a solution under development, the analysis should begin from the claims. If the solution under development does not fall within the boundaries defined by the claims, infringement is not likely to happen. Four claims in the patent application can be considered for the possibility of patent infringement:

1. In claim 1 “receiving ... a set of a plurality of 3D object models for a plurality of different manufacturers of 3D objects”

2. In claim 1 “wherein selected ones of the rules and parameters include physical limitations and constraints for combining the set of received plurality of 3D

object models”

3. In claim 1 “into a new composite 3D object model for a new physical 3D object or virtual 3D object that has not previously existed”

4. In claim 7 “the library application includes a Dynamic Link Library (DLL) library application or a Dynamic Library Loading (DLL) application”

Point one seems to have a possible weakness as the patent application targets a method where the 3D objects are explicitly from multiple different parties. If the solution defined by this thesis were to use only objects by a single manufacturer of 3D objects the patent might not necessarily apply. However, the wording in the application requires analysis by legal professionals. Whether or not “plurality” means exclusion of solutions making use of objects by only a single party is unclear.

Point two might also have a weakness depending on the meaning of “physical limitations and constraints”. Location and orientation information is likely needed if anything else than software communication of components only is implemented. An assembly is not easily created without this information. If physical limitations and constraints are defined in a narrow fashion where they would mean a relationship between actual components rather than simply their location in an abstract coordinate system, a weakness might exist.

Point three contains an interesting possibility for a weakness. In the scope of this thesis, the purpose is to replicate an existing assembly instead of combining existing objects into a completely new assembly. It seems that such a solution would not fall inside the boundaries defined as creating “[a] virtual 3D object that has not previously existed”. As long as nothing new is created, infringement should not happen.

Point four is the most technical of the four identified items. The solution proposed by Glunz et al. [14] is based solely on an approach which takes advantage of a Dynamic Link Library or Dynamic Library Loading application. In case the solution proposed by this thesis can be executed without making use of such applications, an infringement can be considered very unlikely.

The fact that the patent application by Glunz et al. [14] exists provokes consideration over the patentability of the solution selected in this thesis. Particularly if the custom software solution is selected and can be implemented without risk for infringement of the patent application by Glunz et al. [14], a patent by KONE would make the field of possible solutions quite difficult for the competition.

Software solutions making use of commercial software suites usually do not contain such an amount of room for circumventing a patent since the APIs of the software suites are usually somewhat restrictive. Since importing of an accused invention could be argued to include using the solution outputs in the United States, a patent could make even a global impact to the competition’s ability to utilize a similar approach.

As the patent application by Glunz et al. [14] has already been rejected, applying for a new patent with similar claims does not seem feasible. Although the rejection is not yet final, the solution seems to be not of patentable subject matter. The patents

department of KONE also commented that the custom communication solution proposed by this thesis may not contain a novel or non-obvious step compared to the patent application by Glunz et al. [14]. Applying for a patent for the custom communication solution is then abandoned. The acceptance of the patent application seems quite unsure and the value of patenting the solution does not seem to be large enough to justify investing the time and money on an unsure success.

2.4 Software Development

To meet the requirement of engineering a prototype of a custom software communication solution, also software development approaches found in literature were reviewed. Given the importance of different software solutions to the society of today, there has naturally been extensive research on the subject.

The concept of software engineering was created already in 1968 in a NATO science committee supported conference [21, 26, 32]. The conference found that a "software crisis" existed [32]. This crisis meant that software production was not able to reach acceptable results in a large scale [21]. As a result, there was a "software gap" [32], a gap between the achievements of software production and what was actually expected [21]. It was then proposed that the solution would consist of applying tested engineering practices to software production or "software engineering" [32, 21].

The importance of approaching software development in a systematic fashion can be seen in the periodically recurring Standish study, which looks into software projects in large, medium and small companies in the United States [21]. For example in 2012 the study reported that 18% of the software projects studied never reached implementation. Additionally 43% of the projects saw challenges in budget, schedule or scope upon implementation. [21] In some years the results have been as high as a total of 67% of failed and challenged projects [42]. These numbers seem very alarming. If 2/3 projects indeed do fail completely or are challenged in any of the three areas, the prognosis for this development is not good either.

Two studies on software project cancellation by El Emam and Koru [13] and Sauer et al. [42] do challenge the numbers in the Standish report. El Emam and Koru [13] found that 15,5% of studied projects were cancelled in 2005, with 11,5% in 2007. This difference was not statistically significant compared to the sample size [13]. The definition of cancellation was binary, a project was considered cancelled if it did not deliver any usable functionality by the first release [13].

El Emam and Koru [13] also found that of the delivered projects, between 16% and 22% were unsuccessful because of performance based issues. El Emam and Koru [13] found the total of between 26% and 35% for cancelled plus unsuccessful projects quite high. The article also postulates that the sample projects are actually likely to perform better than the software industry as a whole. The article finds reasons like selection bias of unusually skilled respondents in implementation of good software engineering practices and focus on smaller projects. [13]

Similarly, Sauer et al. [42] found that 9% of the projects studied were abandoned. The definition of an "abandoned project" is not clearly defined, so it is difficult to compare the results with those of El Emam and Koru [13]. Sauer et al. also found

that 23% of the studied projects were either budget or schedule challenged [42]. Sauer et al. [42] do not define any indicators for an unsuccessful product like El Emam and Koru [13] did, so this study cannot be used for direct comparison.

However, the study by Sauer et al. [42] also indicates quite different numbers for unsuccessful projects than the Standish report. As high as 67% of the studied projects were found to be delivered close to budget, schedule and scope expectations [42]. It would be very interesting to see the quality perceived by the customer also being analyzed when seemingly two thirds of the projects are successful.

Looking at studies like this, the importance of taking their learnings into account cannot be ignored. The lessons taught by the failures in contemporary software projects should be closely studied in order to avoid repeating them. Even though the sample El Emam and Koru [13] studied for the reasons of project cancellation is as small as 18, their findings should still be studied to avoid similar traps. El Emam and Koru [13] found 11 reasons for cancellation (in order of occurrence frequency):

1. Senior Management was not sufficiently involved (33% of cancelled projects)
2. The scope and requirements of the project were changed too often (33%)
3. The project was poorly managed (28%)
4. The project went over budget (28%)
5. Technical skills were lacking (22%)
6. The developed system was obsolete by time of release (22%)
7. The project went over schedule (17%)
8. Technology was too new for implementation (17%)
9. The project was insufficiently staffed (11%)
10. There were critical quality issues (11%)
11. End users were insufficiently involved in development (6%)

It is important to note that there are multiple reasons for cancellation that are not root causes. In root cause analysis, knowing how and what happened is merely a starting point, the actual crux is why it happened [40]. For example items 2, 4, 6, 7 and 10 are not root causes, so they are not of much use when planning a new software project. Planning to not surpass the budget is not exactly revolutionary.

Instead, attention should be turned to studies such as the one conducted by Lehtinen et al. [26]. This study looks to utilize root cause analysis, or identification of causality between project failure causes. By doing so, the study seeks to not only list generic causes for abandoning a project, but also to explain why these generic causes arose.

Lehtinen et al. [26] start by citing a wider range of causes for software project failures from a previous study conducted by McLeod and MacDonell [28]. McLeod

and MacDonell [28] also categorize these causes in four groups: causes related to people, methods, tasks and environment. With the categorization, a closer look at the possible root cause is already achieved. Lehtinen et al., however,take the root cause analysis further. The causes divided into four categories are shown in Figure 3 on page 17.

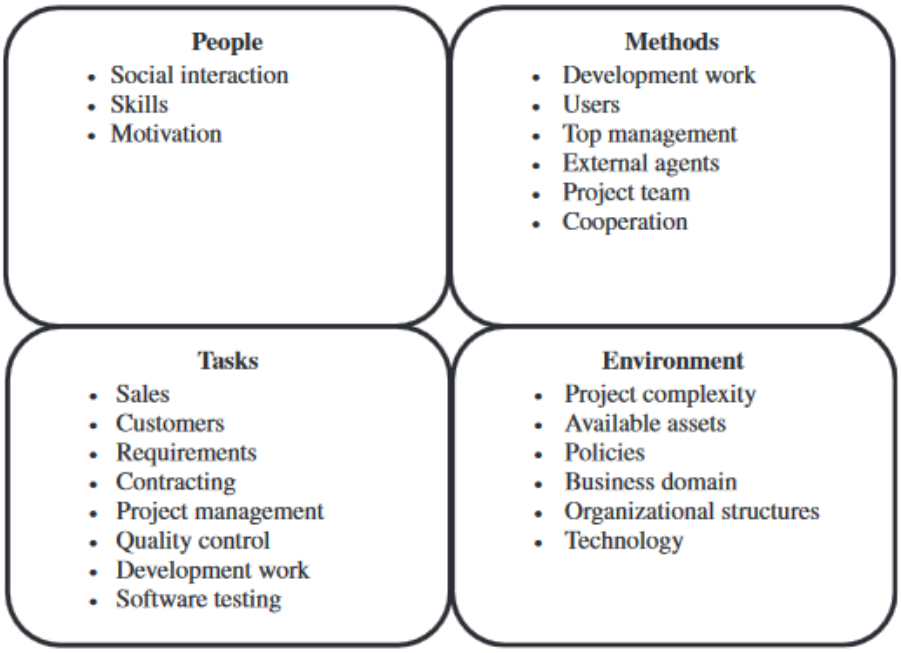

Figure 3: Summary of the common causes of project failures [26]

Lehtinen et al. [26] explain their approach by stating that the previous literature has not paid much attention to the interrelations of the different causes, let alone their causality relations. Lehtinen et al. [26] claim that the software processes cause the engineering problems to be causally related to one another. They continue by claiming that the study of these causalities is beneficial in avoiding these failures in future projects.

Root cause analysis (RCA) is a method commonly implemented in many tools related to process development, such as Six Sigma and CMMI [26]. Rooney and Heuvel [40] describe RCA as a method used to identify the answers to questions what, how and why related to a specific problem. They also define RCA to include recommendations for process development to prevent the issue from recurring in the future [40].

The focus of Lehtinen et al. [26] was specifically on so called bridge causes. These are the causes which connect one process area with others. In addition to the bridge causes, the article attempted to identify the most obvious causes to target in process development. [26] The article concluded, however,that prevention of software project failure requires a case-specific analysis [26]. Even so, a closer look at the results of Lehtinen et al. [26] discloses a number of highly likely causes in case of a software

project failure.

Software engineering is seen in the literature as an extension of engineering sciences in general, but also other opinions exist. Mohaptra [30] notes that the software industry needs to be studied separate from the manufacturing industry because of fundamental differences between these branches of industry. Even if results have been achieved by applying for example the engineering practices from mechanical and manufacturing industry, the comparison cannot be made directly. Productivity of a manufacturing industry is determined by technology, human resource, competence, skill of management and capital, whereas software development processes can be productive and successful without large amounts of capital and high-level technology [30].

Therefore, to ensure a productive and successful software development project, one cannot draw conclusions of successful methods from the way manufacturing industry functions. [30] This is especially important in this type of a study where KONE is an established player in the manufacturing industry and the internal KONE processes may not be applicable in this prototype development.

2.5 Programming Languages

Programming languages evolved to improve the productivity of software developers. Early software implementations were written either in machine code or very low-level programming languages such as Assembly. The resulting software was both restricted to a certain computer architecture and quite difficult to learn because the individual instructions need to be written explicitly. [39, 6] Numerous programming languages were created during the late 1950s and the 1960s, as research sought to provide an answer to this problem [39].

As the software engineering problem of this thesis is focused on APIs that allow the use of Visual Basic .NET as well as C♯ .NET, this thesis will focus on these two programming languages. To be exact, PTC Creo does not contain a C♯ API, but one of its APIs can be leveraged directly using also other languages. Although PTC Creo also provides officially supported APIs for C, C++, JavaScript and Java on top of the languages mentioned above, Autodesk Revit only provides officially supported APIs for Visual Basic .NET as well as C♯ .NET. Shells for Python and Ruby are maintained purely by third parties. The shells are then unfeasible for use in business-critical solutions in a global enterprise.

Purdum [38] notes that some critics propose that "C♯ is the result of Microsoft's stubbornness in refusing to promote a language it did not develop". Purdum then goes on to defend Microsoft that the major reasons for developing this programming language are type-safety of the programs written in C♯ and the managed environment the code is run in. While Purdum's arguments are sound, the criticism is probably not completely off-target either. After all, Microsoft is a publicly traded corporation looking to gain market shares and profits, so it does seem convenient for the company to promote their own developments over others. Haukilehto [19] notes that C♯ was developed as a "perfect language" as Microsoft and Oracle ran into conflict over the extensions to the Java Virtual Machine (JVM).

According to Nagel et al. [31], C♯ is considerably easier to learn than C++ and comparable in difficulty to Java. One could add C as another language that is more difficult in nature to a beginner. While the syntax of C♯ is similar to C, memory management is considerably easier because of the garbage collection functionality that comes with the .NET Framework. C++ offers some optional memory management tools such as constructors and destructors, while .NET Framework handles most of the memory management tasks automatically, whereas C requires the programmer to completely manually implement these functionalities. Arguably, memory management is the root cause for most of the mistakes in a C source code created by a beginner.

It should not be implied that C♯ cannot be a good beginner language because it does give a kick start to easily get simple programs done using the standard libraries of the .NET Framework. Regardless, it is important to understand the concepts of lower-level actions that the standard libraries take care of. This means that understanding the basics of C or C++ does in general prove advantageous to a developer working with source code written in C♯.

When discussing computer programs, the subject of performance and resource requirements is always topical. Several sources [38, 31] state somewhat vaguely that C♯ was designed for performance as well as for other factors. However, the use of the CLR alone means that the performance of software written in C♯ cannot achieve the level of performance seen with similar software written in lower-level languages, such as C++ [31].

The performance restriction is somewhat intuitive. As source code written in C♯ is managed by the .NET Framework and compiled just in time (JIT) for each time the program is run, there is a built-in resource overhead associated with running programs written in C♯. In addition, as working in C♯ is based on the standard libraries, the abstraction level might hide machine instructions that are unnecessary in the task at hand. Working on a lower abstraction level allows the programmer to manage the program closer to the machine instruction level, enabling constantly requesting only the actions that are needed.

However, as Nagel et al. [31] argue, this applies to developments where the time-criticality or performance level is measured in milliseconds and machine cycles. The implementation proposed in this thesis is not anywhere near on the level of these types of performance requirements, so the methodology of this thesis does give a lower weight to the performance of the software solution from a language selection perspective.

2.6 Solution Variant Evaluation

While Mohaptra [30] established that the manufacturing industry methods for product development cannot be directly applied to software engineering, manufacturing industry methods may provide valuable insight to evaluating software solutions as well. As Bosch et al. [5] note, the engineering process has lost popularity in software engineering, leading to a plurality of issues. These issues include for example lack of process discipline, mismatched engineering processes and insufficient pre-iteration cycle work.

The findings by Bosch et al. [5] are aligned with the findings by Sauer et al. [42] as well as El Emam and Koru [13]. The software engineering field does not seem to have matured to the level of older technical fields in the engineering process. Consequently, looking into technical fields where the engineering process has been held to a high value might provide valuable tools for solution variant evaluation.

Pahl et al. [35] provide one of the more extensive studies into systematic engineering design. Their work is mostly focused on engineering design in the field of mechanical engineering. Pahl et al. [35] establish a two phase evaluation paradigm. First, viable solution variants are selected from a large field of possible variants by elimination and preference. After the field of possible solution variants is narrowed, actual evaluation is performed.

Selection is done by eliminating impractical solution variants by assessing compatibility with the overall task at hand, requirements list fulfilment, principle realisability and permissible costs. Preference items, such as safety, ergonomy and company preference for example related to immaterial property conditions are also considered in the selection after the initial elimination. [35]

Related to this study, such a selection of solution variants does not seem important because the field of possible high-level solutions identified is not very extensive. When assessing the details of the custom software solution, even a selection phase might become feasible. As a result, this study will not provide a selection of solution variants on a concept level, but instead the list of solution variants is defined directly.

Pahl et al. [35] present two ways of evaluating the selected solutions, the Cost-Benefit Analysis and Guideline VDI 2225. Regardless of which method of evaluation is used, the process contains somewhat similar steps. The evaluation process begins with identifying evaluation criteria, based on a set of objectives for the solution. Pahl et al. [35] are ambiguous on the differences between the requirements list, objectives and general design guidelines, but the differentiation may be made so that objectives combine the function specific requirements with the general guidelines encompassing the system of which the function is a part.

According to Pahl et al. [35], the Cost-Benefit Analysis requires a hierarchical objective tree at this stage, based on the requirements list. The objective tree is arranged so that different objective areas are on the horizontal axis. Subobjectives of each area run down vertically with the more complex on top. The evaluation criteria are then defined using the simplest subobjectives. Benefit of the objective tree is the facilitation of taking all decision-critical subobjectives into account. VDI Guideline 2225, however,proposes an approach where a non-hierarchical list of criteria is assembled based on the technical details as well as minimum demands and wishes. [35]

To be able to evaluate the different solution variants, one must begin identifying the criteria which are used in this evaluation. Pahl et al. [35] approach the evaluation from a mechanical engineering standpoint, so the headings that they present cannot be directly implemented in this thesis, but must be applied with due consideration. Pahl et al. [35] propose the use of objectives as the basis for establishing the evaluation criteria. The objectives can be distributed into general and task-specific. The general objectives are fulfilment of the technical function, attainment of economic

feasibility and the observance of safety requirements [35]. These general objectives apply partially also to software engineering problems, although the observance of safety requirements is not as relevant as in typical mechanical engineering.

Pahl et al. [35] also propose several headings for the task-specific objectives. Safety as an objective can be applied also in the scope of this thesis if considered in the wider sense of availability and reliability, as these are very relevant aspects for software as well. Ergonomics in the context of human-machine interface is quite similarly a good objective header. Production in the sense of implementation, quality control, operation, maintenance, expenditure and recycling in the sense of usability in other contexts on top of the initially identified task and integrability to other systems can also be accepted. Headers such as assembly and transport should, however,be discarded as irrelevant in the context of this thesis.

Independent of the used method, Pahl et al. [35] propose several conditions that the objectives should satisfy. Naturally the objectives should cover the requirements as completely as possibly. In addition, the objectives should be formulated in such a way that valuation of one objective does not affect the valuation of another. Otherwise one objective might have weight that is not obvious to the observers. Pahl et al. [35] also note that the objectives should be quantitative whenever possible and as concretely qualitative as possible when quantitative objectives cannot be achieved. According to Pahl et al. [35] evaluation criteria derived from the objectives should also be formulated in a way that attributes high value to positive evaluation.

Even though being a study for a software solution, these conditions for the objectives should be followed in this study as well. However, at this stage of evaluation criteria identification, it does not seem possible to identify either the Cost-Benefit Analysis or the Guideline VDI 2225 as the optimal choice for this thesis.

Upon successfully identifying suitable evaluation criteria, weighting of these criteria should follow [35]. Weighting means defining a relative value for each criterion or objective so that the importance of each objective relative to the others can be determined.

In the Cost-Benefit analysis, criteria or objectives are weighted stepwise according to the objective tree. The weights of each branch add up so that the overall weight of the solution is 1 (or 100). [35] The weight of a subobjective then defines the amount of value that subobjective adds to the overall solution.

On the other hand, when following Guideline VDI 2225, weightings are rarely used. Only in cases where some objectives have significantly higher values compared to the others, multipliers such as 2x or 3x can be used. Normally all of the objectives have similar value in the evaluation. [35]

Related to this study, it seems that this part of the Guideline VDI 2225 method might be more feasible. Since the objectives list should not be very long or complex, it seems that a nonhierarchical, mostly unweighted list of criteria might be a viable solution. However, more information about the later stages for both methods is required.

After evaluation criteria has been identified and weighted, the Cost-Benefit Analysis includes one more step [35]. The objectives or criteria are assigned a preferably quantifiable parameter. For example, when the objective for building an

internal combustion engine is light weight construction, the parameter to be followed could be power produced divided by mass of the engine. When following the VDI 2225 Guideline, this parameter definition is ignored and assessing values for the evaluation criteria occurs immediately after the possible weighting of the criteria [35]. The evaluator should also understand how the value function for a certain evaluation criterion behaves. The value function can be linear or exponential, increasing or decreasing. [35] The value function assigns correlation between the parameter and the value scale. In case Guideline VDI 2225 is used, the parameters may not be assigned at all [35]. Particularly qualitative evaluation criteria may make for difficult parameter assignment, but understanding the concept of a value function is still relevant to make proper evaluations.

Value scale

Use-value analysis		Guideline VDI 2225	
Pts.	Meaning	Pts.	Meaning
0	absolutely useless solution	0	unsatisfactory
1	very inadequate solution		
2	weak solution	1	just tolerable
3	tolerable solution		
4	adequate solution	2	adequate
5	satisfactory solution		
6	good solution with few drawbacks	3	good
7	good solution		
8	very good solution	4	very good (ideal)
9	solution exceeding the requirement		
10	ideal solution		

Figure 3.31. Points awarded in use-value analysis and guideline VDI 2225

Figure 4: Scale of values in Cost-Benefit Analysis and guideline VDI 2225 [35]

Value assessment for different objectives or criteria is the next item in the process description by Pahl et al. [35]. In the case of Guideline VDI 2225, the variants are valued on a scale of zero to four. The Cost-Benefit Analysis method implements a scale of zero to ten. Pahl et al. note that while the granularity provided by the

Cost-Benefit Analysis might be beneficial in some cases, the rougher scale in use for the Guideline VDI 2225 is often sufficient. The value scale for both the Cost-Benefit Analysis and Guideline VDI 2225 is presented in Figure 4 on page 22.

Pahl et al. [35] continue that the high resolution valuation in the Cost-Benefit Analysis might often be more difficult because the required thorough analysis might not even be possible at this stage. If one or even multiple solution variants lack a concrete prototype, valuing the properties of an abstract concept might be best done on a simpler scale. It seems that the risk of losing information in such a case is low compared to the added simplicity.

For this study, the method of Guideline VDI 2225 again seems more suitable to follow than the Cost-Benefit Analysis. Seemingly the solutions analyzed by this study force the observer to allow for qualitative objectives and partly vague value definitions. Thus, the shorter scale proposed by the Guideline VDI 2225 seems more feasible.

When the different solution variants have been valued for each objective, Pahl et al. [35] propose that the overall value is determined next. The overall value is often calculated by simple addition, which Pahl et al. [35] point is by definition only acceptable in cases where the independency condition is met. Even when the condition is not completely met, Pahl et al. [35] argue that the addition method is good enough. It seems easy to agree with their conclusion. The logical assumption is that by adding the points valued for a variant in different objectives or criteria, the best solution variant should be obvious.

Guideline VDI 2225 also proposes that at this stage, the economic rating based on manufacturing costs should be determined [35]. In the case of a software solution, capital expenditure (Capex) and operating expenditure (Opex) should be used instead of manufacturing costs. Capital expenditure includes all costs that are related to making the solution ready to use. These can be costs related to physical assets, for example servers or workstations, but also development costs required to prepare the solution as well as legal costs for patents. [1] Capital expenditure is nonrecurring. As such, if perpetual software licenses are bought, they can be considered Capex.

Operating expenditure in contrast includes the costs for operating the solution. Operating expenditure includes for example the non-perpetual license costs of software as well as the costs of running the software such as cloud costs. Also software monitoring and support are included in Opex.

For the comparison of different solution variants, Pahl et al. [35] propose that an economic rating and technical rating should be used separately, if the source data allows for an estimate on the economic rating. Guideline VDI 2225 specifically suggests that this economic rating could then be used on the other axis of an s-diagram, or a strength diagram. For this purpose, the technical and economic ratings should be calculated relative to the optimal solution and not simply as absolute values. The Cost-Benefit Analysis, however, allows for simply looking for the maximum evaluation [35].

Regarding the solution variants to be proposed in this thesis, it seems that the economic rating might be possible to calculate with reasonable effort. Thus it seems that adapting the method outlined in Guideline 2225 might be reasonable, because

the economic rating should be brought in as a variable every time possible.

3 Methods

In the previous chapter, BIM in general, the state-of-the-art of software communication between mechanical CAE and BIM and the software development methods for creating the prototype benchmark as provided by other researchers were analyzed. In the this chapter, the outcome of this analysis is arranged into methods used in this thesis. The engineering design principles are applied to the scope of this thesis by creating the requirements list and evaluation chart to be used in development and comparison of the solution variants.

3.1 Requirements for the Software Communication Solution

As established in the literature review, the design manual by Pahl et al. [35] can be used as a guideline of solution selection in the scope of this thesis. Although Pahl et al. [35] originally based their book on engineering design in the field of mechanical engineering, the general engineering themes are shared also by other fields of technology. The proposed approach was not heavily in conflict with literature in the field of software engineering, such as Mohaptra [30].

Pahl et al. [35] claim that at the core of a solution development and selection should be a requirements list. This list documents the desired functionalities of the solution in question and provides a basis on which to perform the evaluation and verification of the solution. Pahl et al. [35] also propose following criteria that the solution requirements established in a requirements list should satisfy. The requirements should meet three criteria:

- Cover the needs for the solution as completely as possible
- Be defined to maintain their relative independency
- Be quantitative whenever possible

Independency in this context means that the evaluation of one requirement should not directly affect the evaluation of another. If and when quantitative requirements cannot be achieved, qualitative requirements should be defined in as concrete words as possible.

The requirements list setup should be started with very generic demands on the first step, such as “Enable Added Value in the Customer Process”. The demand should then be iteratively reviewed until concrete requirements are found. Pahl et al. [35] refer to the first step as “statement”. The statement includes five items:

- Communicate subcontractor building interface to the customer
- Enable added value in the customer process
- Provide a feasible investment
- Minimize the maintenance cost

- Enable IPR control

Based on the literature review, the most important task for a BIM object is to provide communication between two parties. In the scope of this thesis, the most important communication partner is the customer. Therefore the requirements listing starts with communicating subcontractor building interface to the customer. If the solution fails this requirement, further analysis is futile.

When the first point is revisited for the second step, referred to by Pahl et al. [35] as "development", the building interface can be divided into two parts. The interface consists of geometry and parameter information:

- Communicate subcontractor building interface to the customer
  - Communicate relevant geometry
  - Communicate relevant parameters

The third step is "refinement" [35]. The items can further be divided by refining the term "relevant" as something that is screened. If the information is not properly screened, the customer will likely be overloaded with the amount of detail, which increase model size and reduces performance. Including the refinement step, the first item would look as follows with the numbered items as requirements to be included in the list.

- Communicate subcontractor building interface to the customer
  - Communicate relevant geometry
    1. Communicate geometry in BIM compatible format
    2. Screen amount of geometry detail
  - Communicate relevant parameters
    3. Communicate parameters in BIM compatible format
    4. Screen amount of parameters

The same steps are for the second item in the statement list, "Enable added value in the customer process". Essentially this item means that the structure and content of KONE's BIM must allow the customer to fully utilize the tools enabled by BIM use in general. As per the literature review and KONE sales organization experiences, these tools include but are not limited to clash detection, standardized material management and custom material management.

In the refinement step, clash detection requires that the components are individually identifiable in order to achieve meaningful results. If the components cannot be identified individually, the clash will be detected between the main assembly and the conflicting object. Since the main assembly can be several hundreds of meters tall, such a result is not particularly useful.

Standardized material management allows the model to be viewed in a layered way. For example all of the doors could be hidden from the model simply by hiding

that content category. The categorization can happen in the model classification of Autodesk Revit, standardized classifications and especially the classification of exports to the open IFC format. These categories are manipulated either directly in the model properties or alternatively as parameter-value pairs of the model. The refined requirements for standardized material management would be manipulation of model category and manipulation of standard classification parameter values.

The refinement of custom material management translates to the possibility of defining the main assemblies and components in a certain way. The definition allows the customer to use for example naming based spreadsheet tools for model analysis. This requirement has often been heard from the sales organizations. The customers seem to routinely expect for example subcontractor models to be delivered with a certain naming scheme. The custom material management doesn't necessarily require industry standards to be used, but can be based on any identification of the model. Typically custom material management is not used for model manipulation directly, such as hiding certain component types, but rather as an input for some customized tool. The refined requirement for custom material management is manipulation of model naming convention for both components and the main assembly.

The second item is spread out into different steps:

- Enable added value in the customer process
  - Enable meaningful clash detection
    1. Enable individual identification of components
  - Enable standardized material management
    2. On-demand model category manipulation
    3. On-demand standard classification parameter value manipulation
  - Enable custom material management
    4. On-demand modifiable model naming convention

An implicit requirement for the solution is that the investment needs to be feasible. The feasibility of the investment is often related to how well the existing processes can be enhanced. The different process enhancements are the development for investment feasibility. One way to enhance a process is to simplify it. Process step elimination often leads to significant reductions in operating costs and allows existing resources to be spent better. The solution should preferably perform better than the current method for achieving similar results. To make the solution a feasible investment for KONE, the solution needs to also be easily scalable to the global organization of KONE.

These developments can be refined further. The process step elimination should be refined as eliminating manual process steps and streamlining the automated parts. The performance should be better than with the current methods in terms of time and infrastructure requirements.

Large-scale implementation can be refined to being able to respond to the amount of yearly requests and being able to scale up at a reasonable price. The scaling costs

are quite interesting, because all companies are looking to grow their business. The scaling costs then should not be restrictive, but the costs should relate mostly to infrastructure instead.

The amount of yearly requests based on the orders received by KONE is measured in hundreds of thousands. In 2016, KONE received orders for 158 000 new equipment units [23]. Each unit can be assumed to account for at least 5 tendered units and revisions combined. The refinement then is that the solution needs to be feasibly able to support 800 000 requests per year. The cost of scaling the production up should also be minimized.

The fifth item is spread out into different steps:

- Provide a feasible investment
  - Simplify the process
    1. Eliminate manual process steps
    2. Streamline automated process steps
  - Better performance than current solution
    1. Perform better than the current manual process
    2. Require less infrastructure than the current manual process
  - Support large-scale implementation
    1. Support 800 000 yearly requests
    2. Minimize the cost of scaling up

Minimizing the maintenance cost is a multi-faceted requirement. In general, the objective can be achieved either by reducing maintenance effort, simplifying the scope of maintenance or reducing the unit cost of maintenance. In the development step, these methods can be seen as minimizing the amount of items to be maintained (effort), minimizing the amount of different item types to be maintained (scope) and enabling the maintenance to be performed by different parties (unit cost).

The maintenance effort is an understandable factor in the total maintenance cost. As the solution for software communication does not relate to the development of the content as such, reducing the maintenance effort is based on preferably not requiring duplicates to be kept.

Maintenance scope reduction essentially means that the items to be maintained should be similar in nature. For example changes can be related to the application source code or 3D model content. These two items are probably too different to be handled by a single specialist. Instead, two different resources need to be used, often leading to higher cost.

Enabling different parties to work on the maintenance might seem counter-intuitive to the maintenance scope reduction. However, an important difference exists between the two. Scope reduction means that several resources are needed to implement changes, whereas enabling different parties to work on the maintenance means that the solution is generic and owned by KONE so that maintenance work can be done

in-house or purchased from a variety of vendors. Inviting tenders openly tends to lower the unit cost for maintenance.

The sixth item is spread out into different steps:

- Minimize maintenance cost
  - Minimize maintenance effort
    1. Minimize duplication of data
  - Minimize maintenance scope
    2. Minimize amount of different maintenance tasks
  - Minimize maintenance unit cost
    3. Enable the maintenance to be performed by different parties

BIM might provide an edge in the tightening market situation. As a result, IPR management is a very important topic. The implicit requirement in IPR management is naturally not infringing the patents, copyrights and trademarks held by others. In total, IPR management can be developed into three items, not infringing the patents of others, preventing others from using a similar solution and preventing others from patenting a similar solution.

The refinement for an optimal solution in the sense of infringement is that it either consists of completely original code or is licensed for use from others, making it legal for use by KONE in the first place. Additionally, the solution should avoid infringement of patents and copyrights held by others in both of these cases. Preventing others from using a similar solution can be refined to being of patentable subject matter.

It is also important to restrict the competition from at least preventing KONE from utilizing a similar concept with a patent. Describing a feasible solution in enough detail in this thesis will constitute public prior art, rendering the said solution impossible to patent by anyone. The novelty criterion for patentability described in literature will not be fulfilled after publication of prior art.

The seventh item is spread out into different steps:

- Enable IPR control
  - Avoid infringement of IPRs held by others
    1. Consist of original code or be licensed for use from others
    2. Avoid infringement of a patent or copyright held by another party
  - Prevent others from using a similar solution
    3. Be of patentable subject matter
  - Prevent others from patenting a similar solution
    4. Be possible to describe in enough detail to prevent others from patenting a similar solution

		Requirements list for a software communication solution in Building Information Modeling	Revision -
Changes	D/W	Requirements	Owner
	D	1 Communicate geometry in BIM compatible format	
	D	2 Screen amount of geometry detail	
	D	3 Communicate parameters in BIM compatible format	
	D	4 Screen amount of parameters	
	D	5 Enable individual identification of components	
	W	6 Enable on-demand model category manipulation	
	W	7 Enable on-deman standard classification parameter value manipulation	
	W	8 Enable on-demand manipulation of model naming convention	
	D	9 Eliminate Manual process steps	
	W	10 Streamline automated process steps	
	D	11 Perform better than the current manual process	
	W	12 Require less infrastructure than the current manual process	
	D	13 Support 500 000 yearly requests	
	W	14 Cost less than 10 000 € to scale up 20 000 yearly requests	
	D	15 Minimize duplication of data	
	W	16 Minimize the amount of different maintenance tasks	
	W	17 Enable the maintenance to be performed by different parties	
	W	18 Consist of original code or be licensed for use from others	
	D	19 Avoid infringement of a patent or copyright held by another party	
	W	20 Be of patentable subject matter	
	W	21 Be possible to describe in enough detail to prevent others from patenting a similar solution	

Figure 5: Requirements list for communication solution based on Pahl et al. [35]

Finally, the requirements have been gathered. The enumerated refinements are collected to a requirements list shown in Figure 5 on page 30. The items on the requirements list have also been categorized to wishes and demands. The implicit requirements 1, 3 and 13 are naturally demands. Also requirements 2 and 4 are identified as demands because the customer is likely unable to incorporate KONE's

BIM into their own if the file size is too large because of too much detail.

Requirement 5 is a demand because clash detection is widely recognized in literature as a standard benefit of BIM. Requirements 9, 11 and 15 are defined as demands because they significantly enhance KONE's current approach to implementing BIM. Even if only these items are achieved by implementing a new solution, the investment might still be feasible. Their value compared to the rest of the requirements is then remarkably higher.

Requirement 19 is an absolute demand. Time and money consuming patent infringement court cases could be in order if the selected solution is implemented into production and only then deemed to infringe a patent held by someone else. In the best case scenario, time and money is lost but losing the patent argument could lead to a need to fundamentally change the production processes in a short period of time.

The rest of the requirements are defined as wishes. Issues solved by solutions fulfilling requirements 6 through 8 can also be solved quite easily with separate tools used in the BIM process. However, including them in the software communication solution would naturally simplify BIM related tasks and reduce work that could be automated.

Requirements 10 and 12 provide a solid enhancement to the BIM process and requirement 14 ensures flexibility in scaling. However, these requirements are not critical provided that the solution is otherwise feasible enough. Requirements 16 and 17 relate to making the maintenance of the solution as easy and affordable as possible. Otherwise highly valuable solutions can justify a high maintenance effort. However, low-effort maintenance is usually heavily preferred.

Requirement 20 is considered a wish because software solutions are not usually of patentable subject matter globally by definition. As described in literature, the European Union does not issue patents to inventions consisting only of a software solution. Even though some regions such as the United States issue patents also for software, this is not enough to constitute a demand level requirement. Requirement 21 is a demand, because competitor patents would prevent KONE from utilizing the selected solution in the future, causing fundamental changes to the systems.

3.2 Comparison of Solution Variants

To determine the optimal solution for the research problem of this thesis, a comparison system is required. Pahl et al. [35] propose selecting solution variants for evaluation by means of elimination and preference when faced with a large number of solution variants. In this case, solution variants were already limited to four in the initial study phase. As a result, evaluation of the solution variants can follow directly.

Pahl et al. [35] presented two different methods for evaluating solution variants, the Cost-Benefit Analysis and the Guideline VDI 2225. Based on the literature review, the Guideline VDI 2225 is found to be more suitable for the purposes of this thesis. The decision to use Guideline VDI 2225 is based on multiple factors. Seemingly Guideline VDI 2225 is better equipped to be used in the evaluation of abstractions, whereas the Cost-Benefit Analysis might be more suitable, when the

solution variants are more concretely defined. For example, the vagueness of the solution variants make the 0-4 scale used in Guideline VDI 2225 more feasible than the 0-10 scale of the Cost-Benefit Analysis. The valuation of variants with a greater resolution is not likely to provide any added value in this case, because the objective of achieving adequate software communication is quite vague.

As proposed by Pahl et al [35], five steps are taken in the evaluation process:

1. Identifying evaluation criteria based on objectives
2. Weighting of the evaluation criteria using multipliers
3. Assessing values to the different criteria
4. Determining overall value of the solution variants
5. Comparison of concept variants

According to Pahl et al. [35], the step for compiling and assigning parameters to each evaluation criteria can be omitted when using Guideline VDI 2225.

The literature review identified eight relevant objective headings:

- Fulfilment of technical function
- Attainment of economic feasibility
- Reliability and availability
- Human-machine interface
- Operation
- Maintenance
- Expenditure
- Reusability and integrability

The objectives and the derived evaluation criteria should fall under these headers. In the technical function fulfilment, performance of the solution is quite important. Performance in this case is defined as the time it takes to repeat a PTC Creo assembly as a Autodesk Revit assembly. For a partial conversion solution, the time to convert an individual component is also calculated as an average of the components of a system in order to evaluate the feasibility of keeping the current manual process for creating BIM.

The importance of performance of a solution is obvious, as it naturally is more pleasing to the users not to wait for the models for extended periods of time. KONE also has an automated standard process with high volumes. The solution needs to enable automation and rediuction of the infrastructure requirements. Furthermore, following the throughput volumes of the solution is also important. A high evaluation

will be awarded to a solution that works quickly and can be easily automated. A quick solution in this context is defined as one that is able to achieve the full software communication for a system level model in less than 10 minutes.

Attainment of economic feasibility should be related to the specified amount of yearly requests. A solution that receives a high evaluation in economic feasibility must have a low operating cost compared to the development and implementation cost. Such a solution will quickly bring returns on investment.

Cost is always an interesting factor in any kind of development and the expenditure should be considered next to the feasibility. When planning development based on conceptual design, achieved benefit and added value is weighed against the cost of developing and operating a solution. The solution cost is divided into capital expenditure (Capex) and operating expenditure (Opex). Capex is easy to understand as the price of purchase, or the initial cost of implementing a solution. Opex is the cost related to the rest of the lifecycle of a solution. Everything that has been developed also needs to be kept in operation and maintained until it is finally retired, possibly at some cost as well.

An off the shelf commercial solution typically comes at a higher capital expenditure, but with a smaller and much more predictable operating expenditure. In an in-house application such as the one prototyped in this thesis, capital expenditure is typically low, but operating expenditure is likely to be higher and may include higher unpredictability. As the owner of the solution is inside the company, there is more risk of running into trouble with newer software versions and different requirements than with a commercial product.

Scalability of a solution is also evaluated. In case production volumes need to be scaled up, a higher evaluation is given to a solution which can be scaled at little else than the increased infrastructure costs. Ideally, enhancing application hardware will directly lead to better solution metrics.

Reliability and availability of the different solutions might not be a good evaluation criterion even though it is interesting from an application development point of view. Similar levels of reliability and availability should be possible to implement regardless of the solution. As a result, all of the variants would receive a similar score, rendering the criterion obsolete.

Human-machine interface of the solution should be as simple as possible. BIM is simply an output for KONE so the process phase should not cause extra effort. All three variants can be implemented in a way that does not require more human interference than the click of a single button for the software communication itself. In the component conversion variant, an interface is required to create the main assembly BIM, but this process was excluded of the scope of this thesis.

Genericity of the solution falls under the reusability and integrability header. The selected solution should be applicable to as many different conversion scenarios as possible. To receive a high genericity evaluation, a solution variant must not be dependent on any modeling methodology defined for creating the CAE model. According to Pahl et al. [35] evaluation criteria should be positively formulated, for example “few modeling requirements”.

Having a clearly generic solution available for conversion is essential, since KONE

is only in the process of defining the modeling methodology and approach to properly model the system level of their products. This methodology will certainly see changes and adjustments during the implementation phase.

If these changes can be made without a need to make large adjustments to the BIM conversion solution, the change process remains more agile. If the BIM conversion solution needs major rework for each iteration of the modeling methodology, the increased cost and schedule implications are a major hindrance to the R&D time to market.

Even when a stable modeling methodology is achieved, a generic conversion solution remains vital to the overall performance of KONE. The modeling methodology is not likely to address other than a quite high-level guideline. Great variety exists in the product portfolio of KONE, which will likely result in quite versatile approaches to modeling on the detail level. If a highly generic solution can be achieved, the risk of facing surprises with conversion of different kinds of models is naturally limited.

The integrability of the solution is an evaluation objective in its own right. The research for a conversion solution is part of a bigger ecosystem development ongoing at KONE. An ecosystem is an interconnected web of tools and processes, so the level of integration is another very important objective when evaluating the different solutions. The purpose of the ecosystem development is to integrate the different processes and tools used in the engineering process for a product in order to reduce the throughput times for both R&D and custom engineering processes.

To achieve a high evaluation for integrability, a solution must display a clear method and interface for input and output to the process. These interfaces should preferably be possible to customize as well as implement with an enterprise service bus (ESB) approach. In a best case scenario, the interfaces exist not only for the conversion solution as a whole but also for the individual parts that make up the solution.

Although being a factor in the operating expenditure of a solution, emphasis should be placed on the maintainability of a solution. The maintainability consists of costs in terms of time and cost of maintenance in case of product changes and in case of platform changes. Product changes in this context mean changes to the content that is to be communicated from one piece of software to another. Platform changes in this context mean for example yearly migrations to a newer version of PTC Creo or Autodesk Revit and the implications of those changes.

As there are different factors affecting the cost levels for a solution, care must be placed into cost comparison between different solution variants. Not only are the absolute amounts important, but a believable and well defined development plan and a clear proposal for the required maintenance and support for the solution are required for a high evaluation. As the research is focused on a critical part of the engineering process, the capital expenditure is less important than the operating expenditure.

The intellectual property rights that relate to a solution are also an important factor to consider. If the solution is owned by KONE and and it is of patentable subject matter as described in literature, the solution could provide an edge compared to the competition. To achieve a high evaluation, an optimal solution fulfills both of

these categories. Even if it cannot be patented, IPR ownership is still valuable as changes and customizations can be made at will. A poor solution in this sense is owned by someone else and in the worst-case scenario subject to an existing patent.

Once the evaluation criteria have been identified, weighting of the evaluation criteria is the next step in the systematic approach described by Pahl et al. [35]. Multipliers are used to express pronounced differences in importance of evaluation criteria in Guideline VDI 2225. Of the defined evaluation criteria, low operating expenditure and few modeling methodology requirements are seen to be of critical importance. Therefore multipliers (2×) are added. The value determination and comparison of the solution variants is performed in the Results and Discussion sections of this thesis. The evaluation criteria to be used are presented in Figure 6 on page 36.

3.3 Software Engineering the Custom Solution

As existing off-the-shelf software and features in PTC Creo may not be able to accurately address the specific needs that KONE requires of the software communication solution between PTC Creo and Autodesk Revit. Therefore a prototype for a custom solution was developed. Special emphasis is placed on engineering the prototype of the custom solution in this thesis in order to benchmark the existing solutions and additionally determine the feasibility and required effort of implementing this type of a solution in full. This subsection concentrates on the methods used in the prototyping task.

Two different languages were considered for the prototype development. The Application Programming Interfaces (API) of PTC Creo and Autodesk Revit both directly support C♯ .NET and Visual Basic (VB) .NET. The languages were evaluated and then selected based on evaluation criteria.

C♯ .NET is a high-level, object-oriented, class-based programming language developed by Microsoft in support of the launch of their .NET framework. C♯ .NET is roughly based on C and C++ with special additions to the functionalities.

VB .NET is also a high-level, object-oriented language developed by Microsoft in support of the launch of their .NET framework. VB .NET is an upgrade of classic VB, created to allow support for the new .NET framework. VB .NET is based on the latest “legacy” implementation of the original language, VB 6.0 or VB6. Despite being based on VB6, VB .NET should perhaps be considered as a different language altogether [43, 10].

Because of this, legacy source code written in VB 6.0 does not compile in VB .NET [43, 10]. It should be noted that while conversion of legacy VB6 projects can be done semi-automatically using Visual Studio .NET and its conversion wizard, the conversion is not perfect and should still be checked manually [10].

Davis [10] has established some guidelines to be followed in order to successfully convert projects between VB6 and VB.NET. These guidelines contain for example critically assessing whether a portion of the application makes sense in VB.NET, rewriting the source code so that native features of VB.NET are taken advantage of and finally redesigning the application architecture so that XMLs and other ways

			Evaluation chart for: CAE-BIM Software Communication						Page 1	
P: present variant (P): possible after improvement			A		B		C		D	
No.	Evaluation Criterion	Wt.	P	(P)	P	(P)	P	(P)	P	(P)
1	Good time-wise performance	1								
2	Low Capex	1								
3	Low Opex	2								
4	Low cost of scaling up	1								
5	Amount of information communicated	1								
6	Few modeling methodology requirements	2								
7	Well integrable	1								
8	Low cost for product change implementation	1								
9	Low cost for platform change implementation	1								
10	Patentability and ownership of solution	1								
? Evaluation uncertain ↑ Tendency: better ↓ Tendency: worse		P_{max} = 4 Σ $R_t = P_I / P_{max}$ Ranking								
Justification (J), Weak spot (W), Improvement (I) of variant/criterion										
Decision										
Date: 07.05.2017		Compiled by: Tuomas Ruippo								

Figure 6: Evaluation chart for communication solution based on Pahl et al. [35]

to facilitate interoperability between different pieces of software becomes easier to apply.

It should be noted that KONE has on some occasions chosen to start from scratch rather than attempt to convert an existing application. This has been caused partially by the difficulties in the conversion and partially by the advantages gained by re-engineering the solution completely. Proposed changes which could not have been done with the VB6 application architecture could then be taken into account in the re-engineering.

Although Davis [10] argues that VB .NET is a simpler language to start with and thus might be a suitable choice for this type of a development, VB .NET has not been specifically engineered from ground up to be used with the .NET Framework.

The C♯ .NET programming language was selected based on Autodesk Revit and PTC Creo providing full API support, developer's previous C and C♯ experience and company's related BIM developments being written in the same language. The APIs would have provided support for VB .NET as well and the developer had similar experience in VB but the related BIM developments determined the selection of C♯ .NET.

Despite the impression several sources [10] give, it is possible to do prototype level .NET software engineering without the Integrated Development Environment (IDE) that Microsoft provides for this use, Visual Studio (VS) .NET. Other sources [45, 38] actually point out that Microsoft also provides a free IDE called Visual C♯ Express (VCE), although this IDE contains the restriction of only being able to handle C♯ and not any of the other languages supported by the .NET Framework.

The options are not limited to these two either. The initial development version of the project was programmed with the open source IDE SharpDevelop, which proved to be a fairly positive experience. However, VS .NET proved more advantageous because SharpDevelop only supports .NET Framework up to version 4.5.1. VS also provides support for other languages than C♯, for example VB. The debugging environment could be better defined to work with PTC Creo and Autodesk Revit to achieve a quicker debugging process.

An integral part of a modern software development project is some kind of a version control system (VCS) [8]. The purpose of using version control software is to ensure quality of the development even with a geographically distributed development team, and to allow for merging different versions of the source code [8, 11]. Merging allows combination of the work of each of these developers in a semiautomated fashion. Merging is important especially when two or more developers are working on a different part of the same file. [8]

De Alwis and Sillito [11] postulate that the VCS selected is essential to the organization of the whole development process. They compare the impact that the VCS has on the arrangements of the projects to other auxiliary tools, such as the issue tracking system. The selected software development process should be evaluated in combination with the selected VCS to determine the compatibility between these items.

Basically VCS can be arranged in two different ways. The repository of the VCS can either be centralized (CVCS) or decentralized (DVCS). [8, 11] Centralized VCSs, like Subversion have been the tool of choice before, but the DVCSs have been gaining popularity already a number of years ago [11].

A CVCS would have a single central stable repository, with limited number of people having been granted the rights to commit changes to this master repository. Typically the structure of a CVCS repository will consist of a "trunk", the part where the changes will be committed, "branches", which can be used for stable releases, or developments outside of the trunk and "tags", which are most commonly used for releasing. [11] The branches can also be grown out of the trunk for example to create technological experiments or addressing larger bugs outside of the trunk. Tags, however, are more like status checks of the code and are not usually used for actual developments.

On the contrary, a DVCS does not have a master repository like the CVCSs. Instead, every copy of the repository becomes a master repository that contains the commit history available. [11] The importance of branches grow larger in the DVCS technologies, since all changes are done in the branches and only then pushed to the main development.

Since there are no central master repositories, community often seeks to choose the most successful branches as principal. [11] De Alwis and Sillito [11] mention the Linux kernel and its principal branch, the branch of Linus Torvalds. Other branches may coexist to provide sandbox, testing and validation areas for the changes to be proposed or implemented. As a side product of using a DVCS, de Alwis and Sillito [11] mention the multiplied backup that the decentralized approach provides, for example for disaster recovery situations.

For version control, four different tools were considered. Git is a decentralized source code version control tool developed for the development of the Linux Kernel. [37] Subversion (SVN) is a centralized file version control tool developed to version control complete directory structures containing any kind of file types. A shell extension for simple usage is available. [8] Mercurial (hg) is a decentralized source code version control tool, which also has a shell extension available. [11] Bazaar is a version control tool which can be used in both centralized and decentralized modes. [16]

While version control in a prototype project created by just one developer is not as critical as with a larger development team, it was deemed important to begin the systematic way of working already at the prototyping stage. The project would likely grow in size if and when the prototype would prove successful.

Based on the above, developing a well-defined development process at an early stage would likely lower the costs and quicken the actual development phase. The decentralized approach to a VCS was to be preferred, because the development team would likely be based at locations far from each other. The network connections between these locations might be poor enough to render working on a centralized repository difficult or impractical.

Subversion was left out early in the comparison process, because it seems more logical to version control source code lines themselves rather than the files as a whole as Subversion does. Bazaar was dropped because the support items are dated from 2012, and thus seems like there has not been much development ongoing after that. The final evaluation was then considering only Git and Mercurial as possible VCS tools.

On a quick glance, Git and Mercurial seem fairly similar. Both are decentralized code management VCSs and both offer a Windows Shell Extension to provide a cleaner user experience for a beginner, compared to working directly from the command line. It seems also that the selection of a VCS is quite similar to selecting a programming language to use in a simple project. Both have some marginal upsides, but mostly it seems that developers base their liking on one or the other more or less on personal preference.

The choice was made to use Mercurial as the VCS of this development project. The decision was based on studying the different developer community sites such as

GitHub and Stack Overflow to gain an idea of the experiences of the developers using these tools. This study indicated that while Git provides a powerful tool stacked with features that are useful for experienced developers, Mercurial provides a simpler workflow and a more compact and manageable set of tools.

4 Analysis of Existing Methods

The previous chapter established the methodology and road map to be followed by this thesis for developing and evaluating a software communication method. This chapter presents the analysis of the existing software communication methods. The analysis begins with a look at the different error scenarios that may occur in the conversion, simultaneously providing justification for automating the conversion process. The native BIM support of Creo is found to actually be one of the commercial solutions, thus eliminating this solution variant.

4.1 Need for Software Communication

The minimal requirement for software communication was defined as a means to convert components modeled in PTC Creo into Autodesk Revit in order to reduce the number of component libraries requiring maintenance, since multiple instances of the same data can increase the number of errors in the component instances. The multiple instances may also require specific knowledge of a certain system, in this case a certain modeling software. As a result, the differences in the data systems may require separate "librarians" for each library of instances.

The customer demand for the BIM of system assemblies also dictates that it is not feasible to use large manufacturing CAE component models directly converted into the correct format, since large models slow down the coordination work of the general contractor. The high level of detail in manufacturing models also provides little added value because the manufacturing details are rarely relevant to the customer outside of the building interface. As proposed in Weygant et al. [46], only details distinguishable at a distance of 3 meters should be included in BIM.

Thus, it would be feasible to include component models with a lower level of detail to be included in a BIM assembly. The same component models could be used for general design and BIM because the general design of for the system assemblies only needs only approximately the level of detail used for the BIM of the system assembly, the same component models could be used for general design and BIM. Since the layout design of the system rarely requires that all of the manufacturing details be taken into consideration, using component models with these details poses a performance issue. As a result, layout design could also be accelerated by using a lower level of detail in the CAE component models to be used in the layout design.

Having different levels of detail available for the component model is a standard operational pattern in CAE modeling. Since the general design does not require models containing the exact manufacturing details, the use of manufacturing CAE component models for general design can result in an unnecessary risk of leaking innovative or sensitive design features to competitors or otherwise malevolent parties.

However, creating an effective barrier to deny unnecessary access to the manufacturing CAE component models is not a trivial task. Although using the standard representation based approach to simplify the manufacturing CAE component model for layout design does solve the related performance issues, the PLM solution related to PTC Creo, Windchill, does not support a method for including access management

to the representations. The structure of the model always includes all details even when shown in simplified representation.

Thus, because of IPR management issues, the CAE component models used for layout design and for generating the BIM component model need to be separable from the manufacturing CAE component model. This separation can be created to allow the manufacturing subcomponents to coexist with the layout design subcomponents in the same component assembly with the manufacturing subcomponents being suppressed when not needed. The manufacturing subcomponents can then be stored in a restricted-access location, while the layout design features are available for layout design as well as BIM component model generation.

4.2 Conversion of Components

It is possible to copy the desired amount of geometry from the manufacturing subcomponent to the layout design subcomponent. However, this only applies to the non-configurable geometry, i.e. geometry that does not vary based on parameters. The configurable geometry needs to be modeled separately to the layout design subcomponent. As a result, the theoretical probability for a modeling error in configurable geometry is exactly the same for doing manufacturing and layout design CAE component models separately as it is for CAE and BIM component models separately. However, there are several factors to be accounted for.

First, the need for at least partly separate manufacturing and layout design component models in PTC Creo has been established. A separately modeled BIM component model would then be a third repetitive design task, again adding to the possible modeling error sources as well as hindering the speed of change implementation.

Another factor is if the engineer creating the manufacturing component model also creates the layout design subcomponents, the engineer will likely be less prone to making errors in the layout design subcomponent that do not exist in the manufacturing subcomponent.

It could be argued that the second factor also applies for creating the BIM for the component. However, the third factor of the enhanced quality assurance of having both the manufacturing and the layout design subcomponent in the same component assembly does not replicate into the BIM component model. The enhanced quality assurance means that the engineer can verify model consistency for the manufacturing and layout design subcomponents at the same time. As a fourth factor, modeling in PTC Creo is different from modeling in Autodesk Revit. In addition to requiring different competences, doing the layout design CAE and BIM component models separately also requires licenses to be provided for both software suites to the component engineers. In a large, global corporation such as KONE, the license cost impact cannot be ignored.

In addition to reducing the number of errors in the component models, other factors also need to be taken into account. As the products of KONE consist of a large number of components, the speed and agility of the product development is heavily affected by the speed and agility of the component development. If the component

developments implemented in the manufacturing CAE component model need to be driven separately into the BIM component models, the time it takes to implement a change is increased. An important target identified by KONE is to decrease the time to market of research and development, also favoring the component conversion approach over modeling components separately in different software suites.

The need for converting the layout design CAE component models into BIM component models has been established. As a result, studying the methods for converting the component models is justified. An important factor in the conversion is the need for both constraint configurability on the component assembly level and geometric configurability on the part level. Both kinds of configurability are quite common in the components used by KONE and need to be taken into account when evaluating the different solutions.

Configurability in all forms poses a challenge to component conversion or CAD software communication. Constraint configurability of an assembly requires the subcomponents to be converted separately from the export software suite and then a reassembly in the import software suite. Geometric configurability requires either that the geometry is modeled natively in the import software suite, or that the import software suite is able to open the native file format of the export software suite.

In this case, constraint configurability requires that the subcomponents are exported from PTC Creo into a vendor-neutral CAD file which is then imported into Autodesk Revit using a template specific to the subcomponent type. Using another template specific to the component in question, the subcomponents are then reassembled according to the original constraints.

If only components are converted, the need exists to have a master assembly for the system. In addition, each component type needs to have a master template that is compatible with the system assembly and each subcomponent type for each component type also needs a master template that is compatible with the component assembly. As a result, there is a large number of items to be maintained for BIM in addition to the maintenance items in CAE.

The properties of Autodesk Revit also dictate that translation constraints can be created in only one direction. This results in a very limited selection of methods to create the component assemblies. In some cases, the original CAE component model might also be configurable translation-wise in multiple directions. In these cases creating a similar BIM component model might not be possible at all.

A limitation of this kind needs to be considered as a major downside. If no such cases are identified at the development stage, trusting that there will be no such cases in the future seems naive. A large risk exists that a solution is created that might not be fundamentally able to handle a future development. If such a scenario would materialize, the solution would need to be completely overhauled for a single component. Providing the BIM for the specific customer case would also be unusually slow.

The geometric configurability in this case requires that the geometry is analyzed on the feature level. The file formats that Autodesk Revit is able to import are very limited. For example, Autodesk Revit is not able to import the file format of PTC Creo.

Based on initial studies, feature level analysis of the geometry is only possible in the more advanced APIs of PTC Creo and would require a significant software engineering development. To work on the feature level would also mean that the conversion task would consist of a very large number of different operations. The number of operations in turn would result in a substantially long period of time required for the task.

Essentially, communicating a CAE component assembly into a BIM component assembly is analogous to actually communicating the whole system assembly. In a system assembly, just one more assembly level is included in the conversion. The logic and functionalities for the system level conversion thus already exist, shifting the burden of proof. Instead of reasoning the system level conversion, justification should be provided for still maintaining system level master assemblies in Autodesk Revit.

In addition, BIM essentially comprises means for communications between KONE and the customer instead of a design tool for the internal use of KONE. As a result, if the conversion is done between the PTC Creo assembly and Autodesk Revit assembly, the requirements for converting geometric configurability can be ignored.

When converting individual components, one must also consider the fact that several hundred engineers are designing and modeling components. To be able to repeat the system level design in BIM would equate to defining very strict component level modeling rules. All components of a similar type would have to display exactly similar behavior in order for the system level BIM assembly to function correctly. Even though the restrictions only apply to the references which are used to constrain the component to the system, they still pose a major issue. Particularly in case of differences in the way that a specific component is used in different products, making component conversion truly generic is very challenging.

It is generally not preferable to provide the customer with a model that enables configuring the geometry of components. Even the constraint configurability can be largely ignored based on the same reasons. The system assembly should be provided as KONE has engineered and configured instead of enabling changes to the configuration by the customer. Even though components can still be moved in the converted assembly, breaking the relationships between the components makes it significantly more cumbersome to make such changes.

4.3 Native BIM Support in PTC Creo

Native support for BIM formats in PTC Creo in use by KONE would naturally present a convenient solution for the software communication requirements of KONE. The native support in commercially provided software suites naturally makes the provider of such software accountable for smooth operations using the software. This applies to both third party software and PTC Creo itself.

Using such commercial software might result in cost reductions and rationalizations. Such benefits can be achieved through the fact that the core competences of KONE are not in the software business. However, the limitations of ready-made solutions should be taken into account both with third party software as well as with

PTC Creo itself. In terms of customizability and perfect fit for the exact use case at hand, pre-engineered solutions can rarely prepare for all possible scenarios.

The need for configurability in the software usually arises with the solutions that KONE uses. This is partly due to the fact that KONE operates globally, and has grown to such a position mostly by means of purchasing competitors. As a result, the different divisions have a large variety in operating procedures. Thus, harmonized solutions might be difficult to achieve with an off the shelf product.

In addition, the main products of KONE are immensely configurable in nature. The different product variants have even been estimated to outnumber the amount of stars in the Milky Way, estimated at 10^9 to $4*10^9$. This figure does not even represent the full challenge, since KONE offers also a possibility to customize the products. Thus, variations introduced by all possible customer requests are countless. As a result, the genericity of the solution should be highly valued in solution selection. Any software communication solutions should aim at containing the minimum number of strict constraints in the operation.

To gain knowledge on the future developments planned by PTC, the local vendor of the CAE software in Finland was contacted. The vendor was aware of some upcoming plans by the software provider and helped to contact the software provider to better understand the future road map. The same vendor was also selling one potential solution in Finland and agreed to provide additional information on that as well. The vendor estimated that the possible future software communication solution to be integrated into PTC Creo would probably be the same as was already sold by the local vendor. Seemingly there would be no developments related to BIM formats for PTC Creo itself outside of this conversion tool.

As noted above, the benefits of using a solution directly built into PTC Creo are very similar to those gained by using software by third party providers. In addition it was deemed that the possible solution to be built into PTC Creo would actually be an integration of a third party application. Because of these facts, it did not seem to make sense to analyze a software communication solution integrated into PTC Creo any further in isolation. Instead, this thesis focuses on comparing the commercial software communication solutions with the tailor-made prototype.

4.4 Commercial Software Communication Solutions

The market for software communication solutions between mechanical CAE software and BIM software is not mature, but solutions still exist. Although the majority of solutions concentrate on CAE software and do not provide support for BIM specifically, some suitable solutions do exist. KONE has received multiple marketing visits from a commercial software provider that offers a solution for software communication between PTC Creo and Autodesk Revit. This solution needs to be evaluated.

Initial evaluation of the solution was performed by another employee of KONE. An interview of the employee led to the following notes. The solution is based on extracting the feature data and recreating a custom .bxf file from PTC Creo. This .bxf file is then imported into Autodesk Revit containing several important aspects in the import.

The import process seems to be quite quick compared to the .sat export/import leveraged in the custom software solution proposed by this thesis. This might result from the fact that the .bxf file contains less irrelevant information compared to the .sat file. If the custom software solution is selected, the optimization should be studied to achieve similar results.

From a general BIM point of view, the focus of this solution is somewhat misplaced. Seemingly, the paradigm that the solution is based on concentrates on converting all possible features that might be used in CAE modeling as accurately as possible. The result then is an accurate 3D representation of the original design.

As established in the literature review based on for example by Weygant [46] and Eastman et al. [12], there is little reason in modeling accurately in too much detail. Weygant [46] sets the limit at not modeling features that cannot be distinguished at a distance of 3 meters. Naturally simplified representation approaches could be leveraged also in combination with commercial software solutions to screen the amount of detail.

The analyzed commercial solution utilizes an add-in in PTC Creo to export the .bxf file and then a separate add-in to import it into Autodesk Revit. The working principle of the analyzed commercial software solution is quite interesting. The vendor does not specify which API of PTC Creo they are utilizing. As the add-in is only run on a local system instead of an automated solution, usage of the VBAPI is possible.

The VBAPI utilizes a Component Object Model (COM) interface to connect the application to the instance of PTC Creo and this interface cannot be l on multiple Creo sessions at one time. While this is not an issue when implemented for manual use, it effectively limits the use of the solution on a server. To maximize the performance of a server solution, it should be possible to run multiple instances at the same time. However, the used API was not communicated, so the evaluation should not be tipped either way by this point.

The vendor does not specify how they are using the Autodesk Revit API to run the application, but based on existing knowledge, the solution seems to be running as a Dynamic Link Library application. As shown in the literature review, a patent application by Glunz et al. [14] exists that claims the use of a Dynamic Link Library application for 3D object conversion between a multitude of input software suites, including PTC Creo, and a multitude of output software suites, including Autodesk Revit.

The other claims highlighted by the literature review and the possible weaknesses included in them also apply to the commercial solutions, such as the creation of a 3D object that has not previously existed and the fact that existing 3D objects are simply replicated elsewhere. However, the most concrete and fundamental claim made by Glunz et al. in their patent application [14], the Dynamic Link Library application, seems to be in conflict with the analyzed commercial solution. As a result, the commercial solution would need to be customized to leverage other means of running the application, or risk infringement of the patent application by Glunz et al. [14] if its rejection is lifted.

The amount of customization is the most important deficit of the commercial

software solution that can be seen in the analysis. Most of the points covered lead to the conclusion that while good results can be achieved with the commercial software solution, the necessary coverage requires some customization effort. Even though these points independently rarely require heavy effort, the cumulative scope of work is extensive.

As already shown, the solution would likely need to be rebuilt to leverage the REST API of Autodesk Revit. In addition, the analyzed commercial solution is now only used manually and has not been shown to support assemblies of individually identifiable components in the main assembly. This means that at least the assembly of components would need to be built as a customization in order to effectively run the software communication sought by KONE.

There are several issues with the amount of customization required for commercial software. The solution is a full product being sold, which means that functionality extensions are either customer specific additions to the product or the product itself is extended. If the functionalities are added separately, there is some concern over how the functionalities are supported by the version changes of the product.

Naturally customer specific add-in maintenance can be expected to be invoiced separately in case the customer wants to keep the add-in. This cost should be taken into account in the comparison of solutions. Even if a customer specific add-in is bought, whether the vendor would be interested in selling the complete source code for the add-in remains unproven. As a result, the add-in cannot effectively be opened for bidding by other software vendors.

If the functionalities are added to the product version of the software directly, the support from the changing product versions can be expected to be better and the maintenance cost lower as the software vendor can split the costs between all of their customers. However, this also means that all of the functionalities are available for purchase by the competition. Any expected competitive edge is then potentially lost. The vendor is also not likely to sell the entire source code of the solution, meaning that the product cannot effectively be opened for bidding by other software vendors, since there does not seem to exist other solutions that could fulfill the technical specification at least without similar heavy customization.

Regardless of the chosen route, the solution will also contain features that are not required by KONE. As the custom solution proposed by this thesis shows, the required functionalities of the actual component conversion are quite limited. Simply communicating geometry from PTC Creo to Autodesk Revit does not seem to be complicated at all. However, the analyzed commercial solution contains features such as material and lighting communication that are not included in the current state of the custom communication prototype. The value of these extra features needs to be carefully considered to determine whether having them readily available for future outweighs the induced cost.

Depending on heavily customized commercial software also reduces the possibility of changing vendors in case of problems with the vendor. In a hypothetical scenario of the vendor going bankrupt or facing for example IPR conflicts, the amount of required customization means that there might be a heavy impact on business. If a competing solution cannot be found and implemented in a short period of time, some work

around needs to be found or the integral role that a BIM software communication solution is predicted to have will be left unfulfilled. As the analysis shows, especially the intellectual property rights might become an issue.

5 Custom Solution Prototype

The previous chapter analyzed the identified existing software communication methods. This chapter describes the development and architecture of the custom solution prototype. The prototype, its design and development are described in detail to comply with the scientific principles of reproducibility and falsifiability. The development achieves all set objectives and a functioning prototype is achieved using the described architecture.

5.1 Development of Custom Solution Prototype

The geometry of a given component is at the heart of any CAD software communication problem. Other features of the model, such as parameters can often be quite simply read, stored and written in the destination software suite, but geometry in typical products is complicated enough to cause problems. To avoid these issues, using a vendor-neutral file format is a common method for transferring 3D information between software for different purposes.

This vendor-neutral file contains the geometry information as plaintext structured in publicly disclosed manner. In comparison, the proprietary files may also include encrypted portions. Vendor neutral files allow the communication of geometry without disclosing the details used in the proprietary formats to the public. The drawback in vendor neutral files is that metadata, history information and the distinction of features is usually lost. Instead, the file only contains the geometry of the original model.

Autodesk Revit is quite limited with the file formats it accepts for direct software communication. The software communication between PTC Creo and Autodesk Revit was then initially assumed to require complicated functionalities to recreate the communicated geometry. For example, commonly used transfer file formats .stp and .iges are not supported. In addition using the 3D .dwg format created by the vendor of Autodesk Revit seemed to result in meshed surfaces.

The industry also seems somewhat immature in doing this type of software communication, so the common search engine based software development approach was not applicable here. Searches with seemingly relevant search strings returned quite few results, mainly focusing on irrelevant communication routes as well as the commercial software described in the previous sections.

As a suitable vendor-neutral format seemingly does not exist, the expert interviews then suggested that there would be two options. The first one would be to use APIs for PTC Creo which allow the developer to access individual features of the geometry on a deeper level and then collect information which would allow programmatic remodeling of the geometry in Autodesk Revit.

PTC Creo also provides a view of the model as a program file. This file allows the modeler to do simple programming in the model as well as some details on the geometry or assembly data. The other option would be similar as the first one, but instead of using a different API, the approach would depend on analyzing this program file of the CAE assembly, sub-assemblies and parts.

Regardless of which of these options was picked, the pros and cons would have been quite similar. Both options allow quite robust applications, which lead to only few modeling requirements. The amount of work ahead would still be remarkable, since both alternatives would essentially mean creating a deep-diving feature extractor for PTC Creo in addition to a limited-scope programmatic remodeler for Autodesk Revit. The scope of this thesis would then have to be limited to include only a theoretical analysis of such a solution with possibly some very minor practical studies. The expert interviews suggested that the first option would be preferred, since the vendor of the software is considering the discontinuation of the program file in its current format.

During considerations of which of these approaches to employ, related studies led to the conclusion that the ACIS .sat file format could be used for the communication. The ACIS .sat is not exactly a vendor-neutral file format, as it is the output file of the proprietary 3D ACIS Modeler modeling kernel. However, it is supported by both PTC Creo and Autodesk Revit.

Detailed studies confirmed that the .sat format would indeed be useful for the development. An initial performance study with an actual main assembly resulted in one failed conversion out of 17 components in the main assembly. The error was found to be most likely caused by complicated cast geometry incompatible with Autodesk Revit import from .sat.

Further tests in the scope of this thesis revealed no errors of similar kind. However, the discovery of .sat files as a medium to communicate geometry from PTC Creo to Autodesk Revit led to leveraging this communication method in the current processes for creating BIM representations of the components in the operations of KONE. One instance of creating such a BIM representation resulted in a similar import error, caused by having a sheet metal model cut in the area of a bend.

The geometry communication seems to be reliably solved with the use of ACIS .sat files as communications medium. As a result, the scope of this thesis was kept at the original ambition level of creating a technical prototype. As less attention was required for the communication of the geometry itself, enough resources were available for building a complete communication solution.

Once the feasibility of ACIS .sat as a communications medium was confirmed, the actual development work could be initiated. Development of the solution was begun by setting up a source code repository using the selected version control tool, Mercurial. The immediate deployment of a source code repository ensures that the risk of source code losses and version mix-ups is mitigated.

In a project that requires proper documentation of the progress, such as a thesis, starting version control from the first line of code also provides a convenient way of tracking and reviewing the progress for documentation purposes. As source code was committed to the repository upon successful implementation of each step, the steps can be traced back in the Mercurial log file. Mercurial can also be used to version control the documentation itself, since the document was chosen to be written in LaTeX.

Of the three modules, the BIM part was the first one to be considered. First studies concentrated on establishing the import of the ACIS .sat files to a component

template. To enable flexible testing, the initial studies were done as an external command instead of a full add-in with automatic triggering. The difference is that an external command can be executed at will instead of being executed automatically. The actual automatically triggering add-in would only be created at a later stage. There were also sample commands of a similar nature included in the software development kit (SDK) provided by the vendor of Autodesk Revit.

First tests with a new API are often hindered with the intrinsic traits of application in question. Implementation of the specific API often requires some customization with respect to the structure of how to perform different actions. Once the structure is understood, the development starts picking up pace. In Autodesk Revit, one example of a recognizable trait could be the use of transactions when making any changes to the model, which essentially is a database. For example importing geometry constitutes a change to the database and therefore requires a transaction to be used. The management of these transactions is therefore essential to successful and well-structured API calls in Autodesk Revit.

The import of models requires establishing a proper set of ACIS .sat import instructions. The SDK samples are based on different source formats, so some adaptation for the ACIS .sat import is required. Of the import instructions, the most important is the placement of the imported content. Origin to origin import was selected to enable maximal robustness of the import. The imported content would then always represent a replicate of the CAE source model, including the coordinate system.

After successful implementation of the generic import command, first version of the add-in BIM module and the orchestration executable were created. At this stage, the BIM module only consisted of triggering the external command upon application start. This step was still quite important, since it was implemented using the event handling in Autodesk Revit, on which the rest of the implementation would be based. Initially, only the event triggered when the application was initialized was used, but the logic is quite similar in other events as well. The orchestration executable at this stage took the form of copying the .dll and add-in manifest files for the BIM module to the correct folder, starting the application and deleting the copied files once the BIM module finished running.

Next development item was to leverage also other events, including custom ones and event handlers for those. Instead of triggering an external command when the application reported successful startup, the functionality was split to different parts that are triggered based on the progress of the software communication. The split was done to prepare for the next stage of development, where imports would be done in success for the components and finally for the main assembly.

Separate templates for the components and the main assembly were implemented to enable different properties for these items. Ideally in the production phase global templates would be used by default. The different country units could also customize their own templates according to the specific local needs. For example a European country unit uses a different set of Assembly Code identifications than what Autodesk Revit offers by default. The Assembly Code is a standardized set of identifications to be used for different types of material in a BIM of a building. These identifications

aid in classification of the different types for example for view filtering. Using a local template, this country unit could then acquire their models with the local Assembly Code defined directly in the BIM output without a need for customization for all models separately.

Component imports are a preliminary requirement for recreating the main assembly in BIM. For assembly, the location and orientation of the components in the coordinate system of the assembly need to be defined. In the solution proposed by this thesis, the components are replicated in BIM with the source component coordinate system and then located and oriented around the Z axis in the main assembly coordinate system according to source main assembly.

Autodesk Revit API contains a location coordinate vector property for all components in an assembly that must be set when placing a component. In this coordinate system, the X axis is normal to a "right" plane, Y axis to a "back" plane and Z axis to a "top" plane. The names here refer to the viewing direction from which the positive side of the plane is seen normal to the viewing direction. These directions should be taken into account when comparing to the default coordinate system in a different software suite. If the directions differ, a coordinate transformation needs to be done during import.

When it comes to orientation, the most effective way to account for different orientations is to import the component in a single orientation and then rotate the component instance in the assembly to its proper orientation. Unfortunately, Autodesk Revit prevents a component from being rotated in the assembly around any other axes except Z.

Such a limitation is quite interesting, since any benefits from it are quite difficult to see. Autodesk Revit actually offers a property for any component that prevents an assembly from using the component in any other orientation except vertical. The horizontal-vertical definition is much more rigid overall than in common mechanical CAD applications.

Regardless of the reasoning behind the limitation, it is important to account for by other means. These other means could include orienting the component already in the component file. The drawback is that the same component will need to be imported in multiple instances that each contain a different orientation.

The component import was tested with skeleton geometry as well as regular component geometry. The only geometry limitation to the component export seems to be that the geometry needs to be solidified. Unsolidified surface geometry seems to be prone to meshing as well as losing some of the mesh elements somewhere in the communication process. The solidifying requirement needs to be taken into account in the modeling phase. This does not seem like an overly limiting requirement as there does not seem to be any specific reason to use unsolidified surface geometry. Surface modeling is not widely used by KONE anyway since most of their components consist of quite simple geometry.

In the prototype solution, the assembly is only broken down to the components on the first level of the assembly. For proving the technical feasibility this is enough. In the production solution there is a need to break some of the first level components of the main assembly to the subcomponent level. The use of umbrella assemblies on

the main assembly level is quite common in the existing structures of KONE. At the moment, there does not seem to be any good method to identify an umbrella assembly that should be broken down into subcomponents. An external document which lists the umbrella assemblies used by different platform models looks like the only suitable way to identify these assemblies. The solution is not optimal since the external document naturally needs to be maintained separately, but since the amount of platform models in existence would probably be well under 50, the task does seem manageable. Each platform model is also owned by a dedicated manager, so the maintenance of the external document should be readily available for distribution.

After component imports and recreation of the main assembly were functioning correctly, the 3D aspect of the import was more or less in shape. Some effort was put into the implementation of the orchestration module as a form application instead of a console, in preparation for potentially implementing the solution as a locally run application. However, the added value was quite low, so effort was put into continuing the development of technical functionalities instead. If a user interface is needed at some stage, it would be relatively easy to implement on top of a working solution. Deficiencies in the solution itself could not be similarly covered by a pretty interface.

In addition to 3D geometry, an important aspect of 3D models and especially BIM is the information content. In most applications this information is mainly stored as parameter-value pairs. In a sense, BIM doesn't exist without parameters, since a plain 3D model is incapable of catering for many of the possibilities of BIM.

Some parameters can be set in the local templates mentioned earlier, but many should be created by the communication solution based on what exists in the source CAE model. For example the main dimensions, forces and electrical values should be stored in BIM to be provided to the customer as information. Naturally the identity data of the model, such as manufacturer and manufacturer identification should be stored as well.

Autodesk Revit API provides quite straightforward calls for parameter creation and value assignment although there are three items worth mentioning. Firstly Autodesk Revit provides two different kinds of parameters, family and shared parameters. Family is Autodesk Revit term for a model that represents a smaller entity than a project, which is the other kind of a model produced by Autodesk Revit. In this sense a family parameter is one that can only be accessed on the family level, whereas a shared parameter can be accessed on the project level to which the individual families are loaded. Probably the parameters created in the export should be mostly shared, since the parameters that need not be shared in a project can be excluded in the communication altogether.

Secondly, Autodesk Revit parameters are further differentiated to type and instance parameters. Type parameters of one family type (in practice, one model) can only have a single value in all instances of the type that exist in a project. Instance parameters allow these instances to have a different value for the instance parameter in question.

This classification should not pose a major issue even in the production solution because the individual pieces of equipment provided by KONE have a unique identi-

fication which can be used as the model name. There would then not be instances of the same family type on the project level, since each piece of equipment would make up a unique family type, hence allowing the use of purely type parameters. If individual component instances would need to have different parameter information assigned to them, instance parameters would be needed, but converting a type parameter to an instance parameter is quite effortless.

Thirdly, the technical name for a certain parameter used by KONE might seem like a cipher to the customer. Providing a plaintext parameter whose value is derived from the technical parameter does seem like a good alternative. The technical-plaintext pair is the easiest to provide in the local main assembly template, with the formula of the plaintext parameter equaling the technical name. Translations can be then provided in the same step as well.

To prepare for the 2D view generation, a method for accessing the different 2D views of the model was created. This method was also used as a first step to recreating the datum planes in the CAE model as reference planes in BIM. In the prototype solution, all datum planes in the skeleton of the CAE model were imported. In the production phase, selective import should be implemented, for example using a layer that includes the planes that should be imported.

An important development point concerning the reference planes was also identified at this stage. In PTC Creo, datum planes scale automatically to adapt to the size of the geometry in the model. However, Autodesk Revit lacks this type of a functionality, and reference planes need to be drawn to a specific length.

In the scope of this thesis, the length of the reference planes would not be an important issue since the number of planes is quite low, owing to testing purposes only. As a result, the issue was solved by simply identifying the major dimensions of the model and extending the planes slightly further than these dimensions. In a production solution however, a more sophisticated way of creating reference planes of varying length might be required.

As the BIM module was in quite good shape with the described functionalities, the remaining item would be to develop the CAE module. There the first step was to connect to the software instance, since the calls would be made through a COM interface instead of a .dll extension. Somewhat surprisingly the connection is not as simple as one would expect for an API of a software suite this large. The preparations include defining new environment variables needed by the interface and registration of the API as a COM object.

The initial tests of the API revealed some peculiar traits, similar to the transaction management in Autodesk Revit API. As the interface is originally intended to be used with VB, the corresponding C♯ code is quite verbose and different from typical code written in the language. In addition, managing the COM connection in the development phase where software crashes are quite common is somewhat challenging, since a crash leaves the current connection to the software instance open even if the instance is closed. As a result, the orchestration module was enhanced with an initial call to kill all processes related to the COM connection.

In the production solution, the used PTC Creo API should be switched. The COM interface does not allow for more than one Creo instance to be run at any given

time. Naturally, the scalability of the solution is quite limited as a result. Using another API removes this limitation, allowing a server to run multiple processes in parallel. Even if the solution is distributed to run the CAE module locally, KONE also has an automated standard process that leverages pre-engineered technical platforms. Configurations created by engineering automation would then benefit from the possibility of running multiple PTC Creo instances at a given time.

Once the connection setup was achieved, developing the required code for the export was quite simple. The SDK for PTC Creo contained a sample for exporting a 2D drawing in PTC Creo's native format as a Portable Document Format (PDF). Only small modifications were needed and the code was quite concise. Basically the only items to consider were the structure of the ACIS .sat file and whether the geometry would be exported as shells, quilts or solids.

The structured ACIS .sat was a compelling option, as potentially it could solve replication of main assembly as well. As identification of components as individual objects is a must-have requirement, a cleanly structured import would only require small adjustments such as renaming as well as parameter and material assignments. Unfortunately further study revealed that Autodesk Revit ACIS .sat import would lose the structure of the model. The original plan of recreating the main assembly programmatically would then have to be followed.

An interesting consideration was required when exporting files from PTC Creo. The API throws an exception if the string length of the filename exceeds 40 characters, including the full directory path. Since 40 characters cannot be seen as a loose limitation since the path is also included, the files are initially saved as "0.sat" to minimize the risk of throwing the exception. The name is then changed using the normal Windows API calls for renaming a file.

Geometry can be exported as shells, quilts or solids. The difference between the three is that a quilt is nothing more than individual surfaces at defined location. A shell is a hollow entity bordered by defined surfaces. Finally a solid has a thickness even if it is hollow.

Studies on the different types revealed that Autodesk Revit supports only solid geometry ACIS .sat import properly. Surface geometry was prone to meshing of the surface and losing some of the mesh elements. As there are no requirements on the type of geometry in the finished Autodesk Revit model, solid import can be chosen.

The initial COM connectivity was only implemented for a fresh instance that is created at runtime of the communication solution. If the production solution is distributed so that the CAD module is run locally, starting the import should be possible directly from an open instance of PTC Creo. Waiting for PTC Creo to restart does not make sense for the user. Accordingly, an alternative process was created in the orchestration where the communication solution would connect to an existing PTC Creo instance if one exists and no COM connections are open.

The ACIS .sat exports from PTC Creo and corresponding imports in Autodesk Revit were now available. The XML interface between the two would be considered next. The first item to be created in the structure was the component information. Initially only the location of the component was extracted from the main assembly.

In PTC Creo, the location is noted as an XYZ coordinate vector, quite similarly

to the notation in Autodesk Revit. The notable difference is that PTC Creo uses a different coordinate system. In PTC Creo coordinate system, the X axis is normal to a “right” plane, Y axis to a “top” plane and Z axis to a “front” plane. The names here refer to the viewing direction from which the positive side of the plane is seen normal to the viewing direction.

The coordinates in the interface XML document are defined in the coordinate system of Autodesk Revit. The mapping of PTC Creo coordinates to this coordinate system and the abstract directions in terms of Autodesk Revit views that are taken from the positive coordinate axis direction can be found in table 1 on page 55.

Table 1: Coordinate mapping between PTC Creo and Autodesk Revit

View Direction	**Autodesk Revit**	**PTC Creo**
“Right”	X	X
“Back”	Y	-Z
“Top”	Z	Y

Parameter information must also be included in the interface XML since the ACIS .sat export/import does not include model parameters. The dimension parameters could be sought from either the main assembly of the skeleton model of the assembly. In the production solution all mechanical, electrical and identity parameters might not be stored in the skeleton model and they should probably be sought from the main assembly instead. The effort to change the logic is insignificant as changes are required to only a single line of code.

The proposed solution looks up all parameters that exist in the skeleton. In the production solution it is likely that some sort of screening of the parameters will be required as the model will probably include several hundred parameters all of which will not be interesting in BIM. Another reason to screen the list of parameters is that some of the information included in the parameters may be confidential.

The parameters found from the source model are separated to different groups in the interface XML. This is due to the importance of classifying parameter information in BIM to mechanical, electrical, identity and dimensional data for example. In the production solution it might be beneficial to replace this logic with a single list of parameters that is then classified in BIM according to a reference document. The same reference document could also be used for the screening of parameters mentioned above. If a reference document needs to be created anyway, it would be a feasible solution also for the classification since no logic would be hardcoded, but defined in a light external document instead.

The datum plane features are important for the purpose of referring to certain planes of the produced BIM when it is loaded into another family or a project in Autodesk Revit. Like parameters, datum planes cannot be communicated in the ACIS .sat files. Instead, enough information about them need to be stored in the interface XML in order to replicate them in PTC Creo. The datum planes are looked up from a part skeleton model with the same name as the main assembly.

To create a reference plane in Autodesk Revit, the information that is required includes the name of the plane, the direction of the plane and the position in the coordinate system. The prototype solution includes a logic that first determines the three planes that pass through the origin in different directions. All of the other planes are then traced back to these original three planes. If a plane has a normal in the Z direction for example, its references are traced back to the plane that passes through the origin and has a normal in the Z direction. The offsets to the references in the chain are summed to get the total offset. This total offset is recorded in the interface XML along with the direction of the plane.

However, an inherent weakness exists in this logic. If a plane is defined with another type of a reference than an offset from another plane, this logic will not be able to trace the offset of the plane back to one of the origin planes. This shortcoming should be addressed in the production solution with a complementary approach that supports also planes that are defined with other constraints, such as through a point. Otherwise a modeling requirement is required.

Tens of datum planes are likely to exist in a model that is used in the production solution. The datum planes that should appear in BIM should be screened to avoid a jungle of reference planes appearing in the BIM representation of the model. A complicated set of reference planes is even more troubling in BIM, because individual planes cannot be identified by name from the upper level.

A similar lookup approach that is proposed for the screening of parameters could be used also for the screening of datum planes. However, a more suitable approach also exists. The planes can be easily identified if the planes that should be communicated to BIM are included in a single layer with a predefined name. In this approach, making modifications to the screening can be made directly to the model and explicit datum plane names need not be known.

Once the location of the components can be exported to the interface XML, the orientation needs to be addressed. The already implemented rotation functionality in Autodesk Revit works with Euler angles, which define the orientation around the coordinate axes. The PTC Creo API in turn provides the orientation of a component in an assembly with a 3×3 rotation matrix. The orientation can be calculated using Slabaugh's algorithm [44] presented in Figure 7 on page 57 lets the Euler angles be ψ for rotation around the X axis, θ for rotation around the Y axis and ϕ for rotation around the Z axis. The rotation matrix is of the form

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{1}$$

When implementing this algorithm, the different coordinate systems employed by PTC Creo and Autodesk Revit need to again be taken into account. This is done by assigning switching θ for ϕ. Now the Euler angles can be written for all components in the interface XML. A new node should be created for instances of the same component if ψ or ϕ is different than for the previous instances. This is due to the rotation limitation in Autodesk Revit noted earlier.

The warnings and errors that are meant to be cleared by the user prevent the

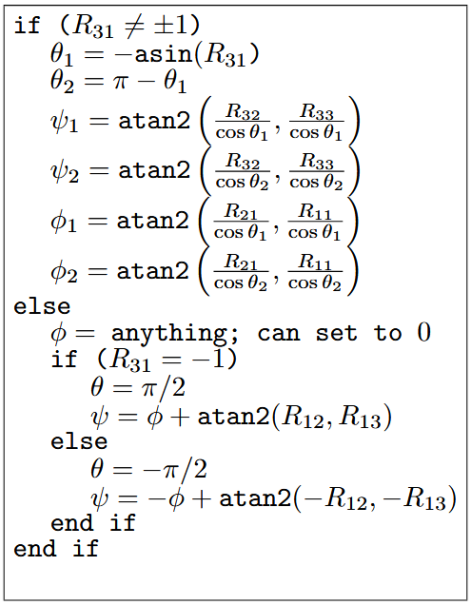

Figure 7: Pseudo-code for computing Euler angles from a rotation matrix [44]

solution from running completely independently. Autodesk Revit API provides a possibility to programmatically "swallow" the popups from appearing, which can be implemented here. Instead of having these errors and warnings appear as popups, they are logged in specific log files for analysis of runtime errors. The orchestration module also shows at the end of the run whether any errors or warnings have appeared during runtime. In PTC Creo popup handling is not as convenient. All exceptions need to be separately handled via normal C♯ exception handling and still some items may pop up unexpectedly.

The only step still remaining in the development was the support for simplified representations in the CAE module. A simplified representation is quite eponymous. Parts are hidden from an assembly and features from parts to reduce the amount of details in a model. This can be done for multiple reasons. First of all, performance is enhanced when working on very large models if the amount of details is tuned down. Particularly in combination with an export solution, the simplified representations allow control of information. The manufacturing details of components will not be exported unnecessarily. For example in a BIM export, the customer is probably happier with a smaller model, where the amount of geometry information is limited.

Simplified representations should be activated prior to the ACIS .sat export in PTC Creo. Unfortunately the use of simplified representations constitutes a modeling requirement. While PTC Creo provides readily available representations such as a light geometry representation, the software communication to BIM is likely to necessitate the use of multiple different kinds of simplified representations. Different levels of detail are inherent in BIM and different components should be possible to portray in different levels of detail. This probably requires different simplified representations, all of which need to then explicitly named for the software communication solution

to recognize them. In the production solution the orchestration module would need to extract the level of detail to be used from the communication request. If the specific simplified representation does not exist for a certain component, the default representation should be used.

5.2 Custom Solution Prototype Architecture

During the initial studies into the research objective, a gap in the existing software market was identified. Even though solutions for software communication between PTC Creo and Autodesk Revit already existed, these solutions did not take into account the specific needs of KONE. To fill this gap, a prototype for such a solution would be created.

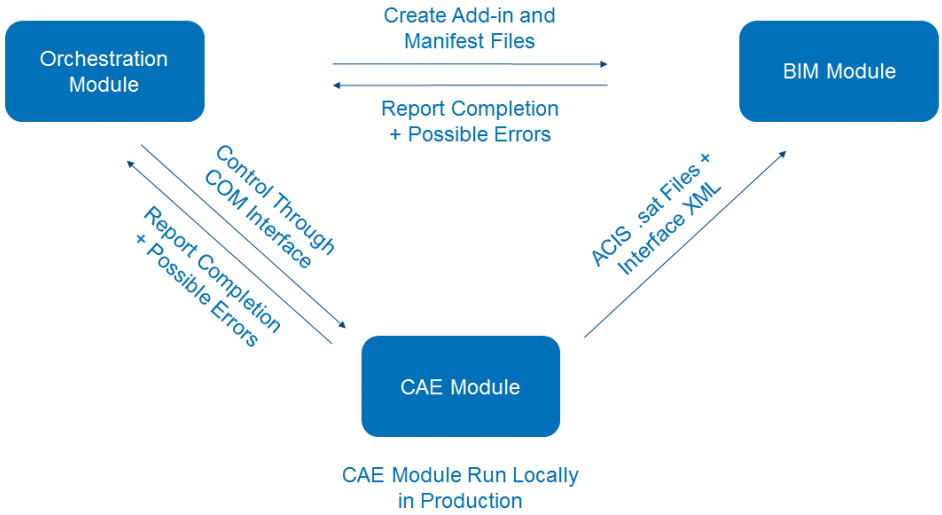

Figure 8: Custom Prototype Architecture

The high-level architecture of the solution is presented in Figure 8 on page 58. The structure of the solution is based on separating the actions done in different software to individual modules. Since one module is dedicated of orchestrating the collaboration between the different software, there are three modules in total. The interface between these modules is a structured XML document combined with the ACIS .sat files. The ACIS .sat files pass the geometry in a simple fashion, while the XML document contains the information that cannot be transferred in the ACIS .sat files. This information includes the metadata for using the ACIS .sat files in the application, transformation matrices to replicate the position of the components in the main assembly, parameter-value pairs as well as data required to recreate datum plane features.

Having clearly separated modules is beneficial, because the different functionalities can then also be used by other tools without having dependencies in the other modules. If the prototype proposed by this thesis is developed to a production solution, the different modules should be distributed between different systems. For example running the CAE module locally enhances performance, since the main assembly does not need to be moved over the network.

The three modules are implemented with slightly different approaches. The orchestration module drives the function of the solution. It is implemented as a console application executable. The console allows for definition of parameters for the communication, such as the main level assembly to be used and whether or not to include the datum features in the conversion. Information about the communication process can also be given to the user via the console. For example the duration of the conversion or error details can be shared conveniently. In a production solution, this executable should be turned into a Windows service that would be run on a server. The functionality would also likely need review if the CAE module is moved to be run at a different location.

The BIM module imports the ACIS .sat files as components and recreates the main assembly, adding parameter and datum information. Autodesk Revit does not allow for running scripting or automation directly in an executable. Instead, the script needs to be packaged in a dynamic-link library (DLL or .dll) that is included as an add-in to Autodesk Revit. This add-in is then defined to run automatically every time an instance of the software is initiated. Alternatively, orchestration could be included in the add-in itself. In a production solution this would enhance performance, since Autodesk Revit would not need to be separately launched for all conversion instances. The BIM module is then implemented as a completely separate .dll that is added to Autodesk Revit using the orchestration module.

The used API of PTC Creo allows direct calls from an executable, taking advantage of the Component Object Model (COM) interface. As a result, the CAE module consists simply of a separate class inside the orchestration executable that contains the required modules for the communication. In a production solution, this module should be run locally to enhance solution performance.

The functional flow of the solution is described in Figure 9. The communication begins by running the orchestration module. A console window is opened in which user defines the communication parameters, such as the path for the main model. After the communication parameters are defined, the automated part of communication is begun and the communication timer started.

The orchestration module flushes all working directories and kills any processes related to the COM interface. Existing processes are killed because the COM interface allows a connection to only a single instance of PTC Creo and connections need to be closed explicitly. In case of errors in the previous run, a connection may be left open, thus preventing all other connections. In case a different API would be used in the production solution, this limitation could be avoided.

Once all existing connections have been closed, a new software instance can be initiated and connected to. If a suitable software instance already exists, a new instance does not need to be opened, but the connection can be made to the existing

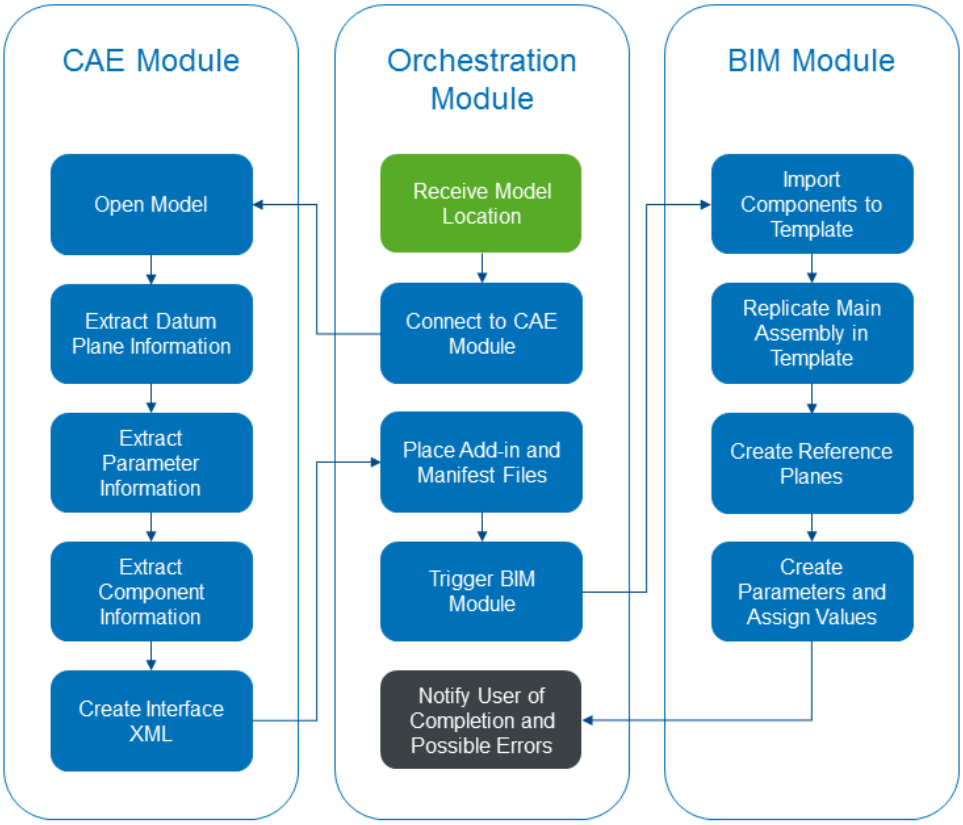

Figure 9: Custom Prototype Functional Flow

instance. Once the connection is complete, the orchestration executable is able to make the calls instructed in the CAE module to the software instance. The main assembly is then opened for analysis.

Datum plane information is the first set of information to be extracted from the model. The datum planes are gathered from the skeleton model of the main assembly. Recorded information includes the direction of the plane and the offset from the reference of the datum plane. Parameter information is similarly extracted from the skeleton model of the main assembly. The parameters are listed in four groups: dimensional, mechanical, electrical and identity parameters.

Component extraction can be divided into two sections. The geometry of the components is exported from PTC Creo as ACIS .sat files and stored in the working directory. The location and orientation of the component instance are extracted from the main assembly to be written in the Interface XML.

Finally the information acquired from PTC Creo model is written into an Interface XML. The XML is structured into the following nodes under the root: Imports, Components, Datums and Parameters. The Imports node contains child nodes for all

unique components that need to be imported, while the Components node contains child nodes for all instances of a component in the main assembly. The location and orientation information are both stored as attributes to support flexible locating and orientation in Autodesk Revit. In addition, the component names and the ACIS .sat filenames are stored as attributes.

Datums node contains child nodes for all datum planes found in the extraction phase. The nodes have the name, reference, offset from the reference and direction information as attributes of the nodes. The Parameters node contains the different parameter groups available as child nodes. The name of the parameter group is written as an attribute and the parameters of the group are in turn child nodes of the ParameterGroup node. For the parameters, name, value and type are recorded.

Once the CAE module is run through, the orchestration module copies the .dll file and the add-in manifest for it to the appropriate locations. The .dll and add-in are copied separately during each run to prevent the communication process from starting when Autodesk Revit is opened for other purposes. This makes the solution convenient for local use, although in a dedicated server environment it could be left out as well. When the files are at their proper location, Autodesk Revit can be started. As the communication solution is set to start running as soon as Autodesk Revit reports successful startup, the BIM module requires no other trigger.

The BIM module begins importing the unique components to a defined component template. The imported components are then saved as native Autodesk Revit files. These component models are then placed and oriented appropriately in an assembly template. The parameters and reference planes are created to complete the main assembly replication. Finally the orchestration module receives completion report from the BIM module and informs the user that the process has finished and also reports possible errors.

6 Results

The previous chapter provided the foundation for evaluation of the different solution variants. In this chapter, the established evaluation criteria is applied based on the analysis according to the engineering design methodology described by Pahl et al. [35]. The evaluation is preceded by confirming that the solution variants to be evaluated meet the previously defined requirements. In the evaluation, the custom solution prototype is given the highest score both in its current state as well as the estimated potential of future improvements.

6.1 Requirements Review

Before proceeding to the actual evaluation, the solution variants should be reviewed against the requirements list (Figure 5) to ensure that they meet all demands. Solutions which do not meet all of the demands should be immediately rejected. As noted previously, support for BIM formats integrated in PTC Creo should not be separately studied, since the researched commercial communication solution is likely to form the integrated communication tool. Thus, only three solution variants will be evaluated:

- Component conversion only
- Commercial communication solution
- Custom communication solution

Pahl et al. [35] note that the variants should be assigned impersonal identification. Accordingly, the component conversion will be identified as solution Variant A, the commercial communication solution as solution Variant B and the custom communication solution as solution Variant C.

Communicate geometry in BIM compatible format

The communication of geometry in the broad sense can be achieved by all three of the proposed solution variants. Further limitations will be studied in the evaluation round.

Screen amount of geometry detail

A similar amount of geometry detail can be achieved in all three by taking advantage of the simplified representations concept in PTC Creo. Alternatively a shrinkwrap can be used. Variant B also has built-in features to exclude components based on volume and quantity. This demand will then be fulfilled already at the input end, making the screening easy and convenient.

Communicate parameters in BIM compatible format

Parameters will need to be added to the replicated model in Autodesk Revit. Parameter communication can be achieved with all three variants, although in Variant A some light customization is required. In Variant A, the main assembly is still built manually or with a separate automation solution from the communicated components. As the parameters will actually be on the main level, the components will not receive any parameters. Instead, a simple tool is required to copy the parameter values for the main assembly from an engineering configurator, such as the PDM system used by KONE.

Screen amount of parameters

Regardless of the variant, parameter screening is quite similar and can be achieved for example using a screening lookup.

Enable individual identification of components

The individual identification of components is a crucial requirement from the clash detection point of view. All of the variants support structuring the model in a way that allows the individual identification. Variant B only supports structuring the model either as a a flat assembly, which does not support the individual identification, or simple logic to break the model down to individual subassemblies or fully down to the part level.

A potential weakness of the solution variants may exist in the structuring, because likely some compromise between subassembly structure and part level structure would be the optimal solution in the production version. Variant A gives full control over the structure, because the main assembly is a completely separate model in Autodesk Revit. Variant C can be customized to any logic which seems feasible from a maintenance standpoint compared to the gained business value.

Eliminate manual process steps

Manual step elimination is an essential requirement in making the investment into the tool feasible for KONE. In Variant A the eliminated manual step is the maintenance of the BIM representation of a given component. With variants B and C the whole process step of creating a BIM representation of the system is eliminated.

Perform better than the current manual process

The enhanced process for creating BIM should perform better than the current one to gain added value. In Variant A the performance enhancement comes in terms of better quality. When component BIM representations are no longer manually created from 2D drawings, risk of errors can be reduced as shown earlier. In variants B and C the automatic generation of BIM both increases the quality and reduces the time required to deliver a BIM of the scope of delivery of KONE to the customer.

Support 800 000 yearly requests

The software communication solution is required to support the full amount of tender and order models of KONE, estimated at 800 000 yearly BIM requests. Technically all variants will be able to support the load by adding infrastructure. In Variant A, manpower is needed, while variants B and C can be mostly handled with software automation.

Minimize duplication of data

Duplication of data is a critical quality risk and the software communication solution needs to minimize such duplication. All solution variants reduce at least one step of data duplication, component modeling in Autodesk Revit. The amount of components is measured in thousands globally, making the component duplication the most important target for elimination. In variants B and C, also main assembly models in Autodesk Revit can be eliminated.

Avoid infringement of a patent or copyright held by another pary

The final demand on the requirements list is to avoid infringement of a patent or copyright held by another party. In other words, the freedom to operate needs to exist. If this is not the case, alternatives such as licensing or purchase of the applying intellectual properties need to be considered. Such actions naturally come at a cost. Based on the literature review, all solution variants seem to have freedom to operate. Variant B is licensed from the intellectual property owner and variants A and C consist of proprietary software exclusively created for KONE.

As all solution variants pass the demand check, these can now be assessed based on the evaluation criteria in the section below.

6.2 Comparison Based on Evaluation Criteria

The three remaining solution variants are evaluated using the nine criteria shown in the evaluation chart based on Pahl et al. [35] presented previously in Figure 6. As proposed in the Methods section, this thesis uses the Guideline VDI 2225 scale of 0–4 for evaluation. The evaluation results are shown in Figure 10.

Good Time-wise Performance

In this thesis, performance is defined the time it takes to communicate information from PTC Creo to Autodesk Revit. The performance evaluation of Variant A must be quite different from a comparable assessment of the other variants. Because assembly of the converted component models is also required, the comparison cannot be made directly by evaluating the time it takes to do the software communication of the main assembly as a whole. While analyzing the solution, it was also found that Autodesk Revit API actually offers limited support in the attempt to create configurable assemblies, because the orientation options are limited. As a result,

			Evaluation chart for: CAE-BIM Software Communication						Page 1	
P: present variant (P): possible after improvement			A		B		C		D	
No.	Evaluation Criterion	Wt.	P	(P)	P	(P)	P	(P)	P	(P)
1	Good time-wise performance	1	1	2	3	4	3	4		
2	Low Capex	1	2	2	3	3	2	2		
3	Low Opex	2	1	1	2	2	4	4		
4	Low cost of scaling up	1	1	1	3	3	3	3		
5	Amount of information communicated	1	2	3	4	4	3	4		
6	Few modeling methodology requirements	2	0	1	3	3	3	3		
7	Well integrable	1	1	1	3	3	4	4		
8	Low cost for product change implementation	1	1	2	3	3	3	3		
9	Low cost for platform change implementation	1	3	3	4	4	3	3		
10	Patentability and ownership of solution	1	2	2	2	2	3	3		
? Evaluation uncertain		P_{max} = 4 Σ	15	20	35	36	38	40		
↑ Tendency: better		$R_t = P_I / P_{max}$	0,3125	0,4167	0,7292	0,75	0,7917	0,8333		
↓ Tendency: worse		Ranking	6	5	4	3	2	1		
Justification (J), Weak spot (W), Improvement (I) of variant/criterion										
Decision	Variant C to be selected and developed further. Possibilities of leveraging variant B for amount of information communicated to be studied									
Date: 07.05.2017		Compiled by: Tuomas Ruippo								

Figure 10: Scored evaluation chart for communication solution based on Pahl et al. [35]

Variant A is evaluated at 1 for performance. Because the mentioned limitations are fundamental to the solution, no improvement can be made.

Variants B and C are quite similar in terms of performance. The communication happens quite quickly, only taking under ten minutes to complete for a typical simplified representation assembly of KONE. The performance can be further enhanced by suppressing the graphical interface. This is not a major enhancement, but potentially helps running the solution on systems that have less advanced display drivers. As a result, both variants are given a score of 3 for the present variant and 4 for what is possible after improvement.

Low Capex

All solutions are assumed to utilize limited period licensing for the required software, which will be included in Opex instead. As a result, Capex evaluations consist of development needs and server infrastructure requirements only.

For variant A, the solution could technically be implemented without large investments. If the components are communicated manually from PTC Creo to Autodesk using the standard functionalities, the only investment required is the training for the people performing the communication. If the solution is sought to be automated, some software development is required. The development of the functionality itself is not too heavy, but as noted, variant A requires a large amount of start parts and templates to run it correctly. The current IT systems require little modification to accept variant A, but the large amount of existing component models requires heavy harmonization work The estimated time required is four person weeks for start parts and templates and two person weeks for the software development. In addition, the harmonization is estimated at 26 person weeks. Despite these drawbacks, the small server infrastructure costs lead to a score of 2 for Capex.

Variant B is quite ready to run as it is a commercial product. However, two items warrant some development. The solution needs to be customized to the use of KONE, because currently it can only be run manually. KONE uses a standard process which needs to run automatically. In addition, the existing systems at KONE need to be modified to accept the new module in the process. The effort estimate for these tasks is 3 person weeks for system modification and 3 person weeks for solution customization. The variant requires production and acceptance/testing environments to be set up from the server infrastructure point of view, leading to a score of 3.

Variant C is only proposed as a prototype in this thesis. It needs to be finalized and the current systems modified to accept the new module. This development is estimated to take 5 person weeks for finalizing the current solution and 3 person weeks to modify the existing systems. The variant requires similar server infrastructure as Variant B and is then awarded 2 points for Capex.

Low Opex

For operating expenditure (Opex), the salary and license costs are considered. Monitoring costs will be quite similar for all solutions so they are excluded from the study. Variant A is the only variant to require salary costs as main assembly BIM will still need to be manually created in the manual process cases. To estimate the cost to support the required load of 800 000 requests, an estimate on the amount of manual process cases is first required. The manual process is responsible of roughly 15% of yearly orders but can be assumed to require more revisions than automatic process cases. At 30% of yearly requests the required cost would likely require a team of 8-10 modelers to support the load. Also the license costs for these modelers need to be accounted for. Variant A is given a score of 1 point for Opex.

Variants B and C require no salary costs and only a few PTC Creo and Autodesk Revit licenses for the servers. As a large benefit, both variants will eliminate the need for a separate BIM automation solution. The main difference between the two

is that Variant B also comes with a license cost. A license is required for both the automation servers as well as the manual process engineers. As the single yearly license cost reported for Variant B is roughly 3000 euros, the cost is not insignificant given that there is a potential pool of several hundred engineers requiring the solution. As a result, Variant B is awarded 2 points for Opex, while Variant C receives 4 points.

Low Cost of Scaling Up

The cost of scaling up the solution is studied based on scaling up 10 000 yearly requests. For variant A with the current manual and automated processes, this would mean that 1500 new requests a year need to be handled manually and 8500 new requests in the automated process. Based on estimating one request to require 4 hours to fulfill, an assumed 40 hour work week would give 150 person work weeks required. The requirement can then be rounded to 3 person work years or three extra resources. The automated solution is able to support roughly 50 000 yearly requests per model production server, so the cost can be assumed to be around 25% of a virtual server allocation when including the required setup and test costs. Variant A is given a score of 1 point for scaling up because of the sizeable workforce requirement.

For variants B and C, only the server costs are required. The elimination of required manual effort is significant. Also the automated solution can be scaled down and only used for certain simple cases where no model in PTC Creo is required. Achieving the desired software communication requires the use of two separate server systems, of which the first is used to extract the data from PTC Creo models in the automated process. This server is not required in the automated process. The other is used in both automated and manual processes to generate the actual BIM. The server costs are then doubled, although the handling times are roughly the same. The cost of scaling up by 10 000 yearly requests is then 50% of a virtual server allocation. Variants B and C are evaluated at 3 points.

Amount of Information Communicated

Variants A and C are based on the .sat conversion, which allows for fairly little detail to be communicated. Only the geometry can be passed for the component models. In addition, Variant C contains the possibility to add parameter and datum information which can also be added to variant A with further improvement. Additional benefits for items such as connectors, materials and lighting, present in Variant B can mostly also be added to Variant C with further improvement.

For the amount of information, variant A is then awarded 2 points for the present variant and 3 points for what is possible with improvement. Variant C is awarded 3 points for the present variant and 4 points for improved variant. Variant B already receives 4 points for the present variant.

Few Modeling Methodology Requirements

Making variant A generic enough not to require modeling methodology requirements is not a trivial task. Each component type to be converted requires its own method of conversion, since the master assemblies in Autodesk Revit have been specifically engineered to meet the requirements of the users so that the placement of components is natural to the engineering process.

In addition to requiring a specific conversion method, only adapting a component conversion requires more discipline at the input end. To automate the conversion, certain datum features would need to be recognized, ergo all of the component types would require a certain start part. Additionally, generic start parts for components hosted in different orientations would be required. By different hosting, two generic types of components can be identified: wall based and floor based components. A start part would define the necessary datum features in a way recognizable for the application.

In the solution variant analysis, a very alarming feature of the variant A was found. Software communication of models between PTC Creo and Autodesk Revit eliminates the geometric configurability of parts. KONE's products consist largely of components that feature geometric configurability. As a result, a library of component models lacking geometric configurability cannot be accepted.

The API is capable of constraint configurable assemblies only with respect to a single coordinate direction. As a result, any assembly that requires constraint configurability in more than one direction would be out of scope for such a solution. Scope limitations are fairly dangerous in a large scale process definition like the one at hand. If the scope of the solution is limited to only cover some of the possible components, at least two parallel processes must coexist. Parallel processes waste many of the proposed benefits and should only be implemented as a transition phase between the old and new processes or very special cases.

Concluding the argumentation of variant A support for few modeling requirements, the criterion for solution variant needs to be valued. According to the chosen evaluation method, Guideline VDI 2225, the valuation should be done on a scale of zero to four. Because the lack of geometric configurability is unacceptable, the modeling methodology requirements evaluation criterion is assigned a value of 0 in the present variant.

With further solution improvement to communicate the components with the correct parameter values required by the case at hand, the modeling methodology requirements are eliminated in terms of geometric configurability. However, the solution is then quite close to Variant C and will still require explicit start parts to be used. Component start parts would naturally require maintenance and ownership. Before the R&D unit responsible for the design of components fully incorporate the BIM component model creation process, maintenance and development of a potentially very large selection of components would be required. Since BIM is a quickly evolving field, a real possibility exists to require changes in the fundamentals of the component modeling methodology. Variant A is then assigned a just tolerable value of 1 for possible after improvement.

Variants B and C are easier in this sense. They do not place heavy requirements on the modeling methodology. Both variants have some recognized deficiencies in communicating more advanced geometric features such as boundary blends and sweeps as well as non-planar cutouts in sheet metal. However, the products of KONE are mostly quite simple, especially when simplified representations are utilized. For producing reasonable BIM content, both variants do require simplified representations however. Both variants will then receive a score of 3 for modeling requirements.

Well Integrable

Integrability is a key factor when looking to enhance the overall engineering and documentation process of KONE. An optimal software communication solution integrates well to the rest of the process. Variant A keeps the BIM process separated from the rest of the engineering process. If BIM is generated separately, integration is not well implemented. However, information can still be moved between different systems. Integrability is then evaluated at 1 point for variant A.

Variants B and C are easier to integrate since they can make full use of the main assembly 3D models. Of the two, Variant C is slightly better since the source code is exclusive property of KONE. As such, even parts of the solution can be reused elsewhere. The evaluation for integrability is 3 points for Variant B and 4 points for Variant C.

Low Cost for Product Change Implementation

Change management of the products is an interesting topic, especially with BIM being in state of speedy development. For variant A the cost for implementing a change in product consists of the effort to create a new BIM representation of a given component. The effort is not huge, especially if changes are made to create the component models on demand, but it cannot be ignored. A new component type altogether requires the creation of suitable start parts and templates for future components of the same kind. In case changes are made to the main assembly, variant A requires corresponding changes to the appropriate master model for creating the main assembly in Autodesk Revit. Variant A is given 1 point for the present variant and 2 points for improvement possibilities.

Variants B and C accept changes very quickly. Since these solutions are not tied to the content being communicated, changes to the content do not constitute a maintenance item if the modeling methodology is not changed. Only changes to the desired structuring would require maintenance work. The variants are then given 3 points for product change implementation.

Low Cost for Platform Change Implementation

In case the platforms for CAE and BIM are updated, the software communication solution needs to comply. All variants use quite fundamental API calls, so major modifications are not expected in case of changes to the software platform. Variants A and C are then assigned 3 points for the platform change management. Variant

B is awarded 4 points because the solution seems to be in the roadmap of being included directly in PTC Creo.

Patentability and Ownership of Solution

For patentability and ownership of the solution, the variants are compared to the similar rejected patent application of Glunz et al. [14]. None of the solutions seem to provide no more novel and innovative steps than the rejected patent application [14]. Patentability does not then seem to exist for any of the variants. Variant A is merely taking advantage of the existing functionalities in PTC Creo and Autodesk Revit to create a solution leaning heavily on manual work and does not then seem to provide any real possibility for a competitive edge. The evaluation for ownership of the solution is then 2.

Variant B is facing a similar situation. The solution is available for purchase by the competition as well and in the roadmap of being included directly in PTC Creo. This inclusion makes the risk of leaning on a single solution vendor smaller as the confidence of solution stability is higher even if the solution is not owned by KONE. Variant B is awarded 2 points as well.

Variant C is completely controlled by KONE. As a result, changes can be easily implemented and the solution developed by the best possible party in all scenarios. Variant C is then awarded 3 points for the last evaluation criterion.

Variant C with improvements has the highest score at 40 points, which is 83.3% of the maximum score. Without improvements, Variant C compiles 38 points or 79.2%, which is still two points better than Variant B with improvements.

7 Discussion

In the previous chapter, results achieved by this thesis were presented. In the this chapter, conclusions are drawn from the results. In addition, a future development plan is proposed. The arrangement of the custom solution to a production service is justified and described as a split between a network service and a local solution. Possible enhancements that can be achieved via using a REST API are also described. The chapter proposes not to pursue intellectual property rights for the solution. Finally, suggestions are made for modeling methodology limitations to comply with the proposed software communication solution.

7.1 Conclusions

This thesis has studied the optimal solution for software communication between PTC Creo and Autodesk Revit. A custom built software solution using the official APIs of the PTC Creo and Autodesk Revit suites gives a high level of customizability and offers flexibility for changes. As a result, this thesis finds that the custom built software solution to solve the research objective in an optimal way.

Furthermore, as part of this thesis, the feasibility and technical concept of the custom software solution were proven by creating a functional prototype. To leverage the results in production use, this thesis proposes a requirements list for a software development project. The functional prototype should be used as a starting point for the software development project. As part of the next project, the potential weaknesses of this study should be considered. These potential weaknesses might include for example the material assignments of BIM component models produced by software communication. In case customers begin to request material assignments as part of the BIM deliverable, the creation of these assignments in the software communication process should be closely studied.

The literature review revealed extensive studies into best practices in software development and analysis of reasons for the failure of software development projects. These best practices and failure causes were analyzed and should be taken into account in the proposed software development project. The reviewed studies show that if the best practices are not followed, the risk of failure in the extensive implementation of such a development project is large.

The evaluation of different solution variants gave expected results, with the minimum objective of communicating component models only receiving the lowest score. The commercial solution received a fairly high score and the detail of information which can be communicated should be duly noted. The integrability and operating expenditure of the custom solution seems advantageous, as few customers seem to be requesting much detail in the component information at the moment.

7.2 Future Development

Although it seems that the prototype performs at an acceptable level and that the cases that it has been used for in testing show substantial potential, several important

developments are proposed. These developments will raise the value of the solution even higher than what it is today. This increase in value should also be achievable at a relatively low cost, because the software is entirely owned by KONE.

The most reasonable way of arranging the solution for scalable production use is to turn the solution into a service. A centralized service running on dedicated servers is beneficial in multiple ways when compared with a scenario where the software communication solution is run locally on the engineers' systems. First of all, it removes the need to provide a license for Autodesk Revit for all system level engineers. The approximate license cost for Autodesk Revit at the time of publication is 2600 euros per year [20] and KONE is employing approximately 200 system level engineers. If the software communication approach were to be implemented individually for each designer, the license costs are obviously unsustainable. The license costs can be significantly reduced by completing the software communication process on a server, reserving a single license for both PTC Creo and Autodesk Revit.

The engineers working in the customer interface naturally still need a complete Autodesk Revit installation. They need to be able to communicate with the customer, which requires extensive work in Autodesk Revit itself. However, the engineers creating the initial general design of the elevator do not require a Autodesk Revit installation, but could be provided with a viewer instead. Thus, the designer can verify that the design still matches the design intent even after converting the design into Autodesk Revit. However, because they do not actually perform any work in Autodesk Revit, a complete installation combined with a license would lead to significant waste. KONE has committed heavily into the principles of Lean engineering and manufacturing processes. Such waste would be in direct conflict with these principles.

A discussion with a software company working extensively on add-ins for PTC Creo produced an interesting variation. Autodesk Revit part of the software communication solution could well be implemented as a service on a centralized server. However, it might be viable to implement the PTC Creo part of the solution rather as a local item.

There are several benefits in this approach. First, the network traffic is significantly alleviated, because complete PTC Creo assemblies are not sent to the central server, but instead only the configuration XML combined with the vendor-neutral CAD file format component models are sent. According to initial tests, the size of the complete system level PTC Creo model is approximately 4,5 times the size of just the vendor-neutral CAE file format component models. Since the scale of the discussion is in the gigabyte range, a 4,5 fold reduction in the amount of data to be transferred is quite significant from a network load point of view. Because of heavy customization, the system level CAE model cannot be loaded permanently on the server, since in some cases the customer requirements necessitate more than mere configuration of a pre-engineered model.

Second, the software communication can be based directly on the model that the engineer is working on. No separate saving and compressing the model files is required. In addition, the visibility settings that a user is currently applying on the model can be taken into account. Automatically detecting the visibility settings

leaves less room for a human error causing quality risks.

Third, both the performance and the capacity of the software communication solution are enhanced, when using a hybrid approach like the one proposed. If the server only needs to take care of half of the process, there will naturally be less queue even in busy hours because half of the process of software communication has already been done. As a result, users will see shorter reaction times to the requests because of the shorter queues but also because the server takes less time to reassemble the vendor-neutral CAD file format according to the instructions in the XML.

However, there are also downsides to splitting the process between local and server-based items. If some components are run locally, it means that there is more of a risk of semi-complete models being created with an outdated engineering structure. It is more difficult to control decentralized solutions compared to centralized solutions. Some workarounds exist, such as forcing the solution to check for updates every time that a run of the solution is attempted.

As mentioned in the IPR section, the patent application by Glunz et al. [14] might prove troublesome. The results section mentioned the use of the REST API of Autodesk Revit as a possible development item to circumvent the patent. As the patent application [14] only claims an application running via the use of DLL, the REST API seems to provide a method to avoid any possible trouble caused by the earlier patent.

However, applying for a patent is always a compromise. Since there are costs involved in applying for and maintaining a patent, in case the patent does not provide value, the investment might not be feasible. Having the working principle published in this thesis means that any attempts to apply for a further patent should be rejected by the patent office. If the patent is not rejected by the patent office, the chances of a favorable outcome from any ensuing law suits are significantly increased with proof of prior art.

The KONE legal department has reviewed the need and benefits of applying for a patent. As a result, the decision has been made to not pursue a patent for the working principle with the use of the REST API. The fact that the working principle is published is seen to be enough.

The logical next step in the development of CAE to BIM conversion is the automatic generation of approval drawings based on BIM. Creating drawings from BIM brings about several key benefits. Having the approval drawings and BIM as mutually connected objects in a single deliverable file is a step forward in itself. It also allows a group of connected systems to be assembled only at this stage, helping reduce the performance requirements of the computers used in the CAE phase. Customers seem to value having elements from the building design incorporated in the same drawings that show the relevant parts of the delivered systems as well.

KONE products are generally arranged in groups. The customers usually also would prefer seeing the groups of equipment as a hierarchical structure in BIM. The next evolution of the solution should be able to not only provide software communication between CAE and BIM, but to also include data from KONE's PDM solution, thus generating the groups and all devices related to its function.

7.3 Modeling Requirements

To allow for successful conversion, several factors would have to be taken into account. These factors affect both the export from PTC Creo to the vendor-neutral CAD file and the import of this vendor-neutral CAD file to Autodesk Revit. It is important to note that unsolidified surface features are not allowed, since they cannot be imported into Autodesk Revit.

Some of the features were built to not appear in the vendor-neutral CAD file at all, but rather be documented as information in the extensible markup language (XML) file. This XML file is then to be read and the features modeled natively into the final model in Autodesk Revit.

Features that do not appear in the vendor-neutral CAD file include the specified datum/reference planes as well as the parameters that were to be created in Autodesk Revit. Thus, the approach taken to build for example the datum planes in PTC Creo does not dictate the way the reference planes are arranged in the final BIM file. Since importing features like datum/reference planes with the vendor-neutral CAD file is often problematic, it is often more feasible to use the definition of the feature to model it natively in Autodesk Revit. In addition, the definition for these features is not as difficult to extract to the XML as it is for the geometry using the basic API.

To capture features like datum/reference planes and parameters easily, they should be arranged in a single skeleton on the main assembly level in order to have all necessary datums appear in the final model. Optimally, this skeleton would not contain any other datums except those to be included in BIM. If the single skeleton contains all datum planes that should be repeated in BIM, no additional lookups are needed to limit the repeated datum planes. As a secondary solution, these planes can also be included in a single group inside the skeleton, if it is necessary to also include datum planes that are not to be repeated in BIM. To create the planes correctly, the naming and relative position in the system of the first datum planes in all directions, XY, XZ and YZ should be standardized. If this standardization is done, all other planes can be placed according to these original planes. Additionally, the converted systems are easy to use in BIM because their relative origin in the physical structure is the same.

As for parameters, parameter types should be limited to those known to both PTC Creo and Autodesk Revit. If other types are to be used, their usage should be limited to PTC Creo integral use. Because the parameter types are mapped in advance between the two software suites, asking for parameter types unknown to Autodesk Revit to be communicated is likely to cause an error. The parameters that are to be included in the final output of the software communication should be provided in the initial configuration XML file.

The vendor-neutral file format import to Autodesk Revit limits the features to be used in the CAE modeling to rather simple ones. However, considering the requirements of BIM, this is an issue that would need to be resolved at some stage anyway. As proposed by Weygant [46], the level of detail should be limited to items that are possible to distinguish from a distance of 3 meters. Any complicated geometry or advanced surface models can be omitted as long as the overall space

reservation and building interface of the component in question can be determined by the customer based on the BIM provided. This seems well aligned with the requirements for component model detail level in the general design phase.

CAD modeling can be roughly divided into surface and solid modeling. Surface modeling develops a geometric shape by first creating the surfaces of the shape. Then, the volume limited by these surfaces is solidified to create physical volumes of material.

Solid modeling seeks to partly semi-automate the process. Instead of defining surfaces, a section of the shape is typically created and then extruded along a specified route. Surface modeling is the earlier innovation of the two, but still is heavily used creation of most complex geometry. In some scenarios, surfaces might be left unsolidified in CAE, for example to define a space reservation.

To determine possible limitations in the utilization of these two different paradigms, the conversion through the vendor-neutral CAD file format was tested using both. The vendor-neutral CAD file transferred the geometry in an acceptable fashion to Autodesk Revit regardless of which method of modeling was used in PTC Creo. However, the results showed that pure surface geometry without solidification did not transfer acceptably from PTC Creo to Autodesk Revit.

Space reservation features often represent simple, box-like geometry. Such features require relatively little data to determine their shape, size and location definition. Software communication between PTC Creo and Autodesk Revit can then be achieved by passing the definition as numbers instead of the geometry as a vendor-neutral CAD file format. The feature can then be remodeled natively in Autodesk Revit with reasonable effort. Accordingly, this thesis proposes that space reservations created with pure surface geometry in the CAE engineering phase should be communicated with native features in Autodesk Revit.

References

[1] Appuhami, B. R. The impact of firms' capital expenditure on working capital management: An empirical study across industries in thailand. *International Management Review 4*, 1 (2008), 8.

[2] Aram, S., and Eastman, C. Integration of plm solutions and bim systems for the aec industry. In *Proceedings of 30th International Symposium of Automation and Robotics in Construction and Mining, Montréal* (2013), pp. 1046–1055.

[3] Bernard, F. The dassault systemes success story, 2010.

[4] Bosch, J. From software product lines to software ecosystems. In *Proceedings of the 13th international software product line conference* (2009), Carnegie Mellon University, pp. 111–119.

[5] Bosch, J., and Bosch-Sijtsema, P. From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software 83*, 1 (2010), 67–76.

[6] Brooks, F. *No silver bullet, essence and accidents of software engineering. IEEE Computer, 20 (4): 10.* April, 1987.

[7] Closa, D., and Falk Giemsa, J. *Patent Law for Computer Scientists.* Springer, 2010.

[8] Collins-Sussman, B., Fitzpatrick, B., and Pilato, M. *Version control with subversion.* " O'Reilly Media, Inc.", 2004.

[9] Danhaive, R. A., and Mueller, C. T. Combining parametric modeling and interactive optimization for highperformance and creative structural design. *Proceedings of the International Association for Shell and Spatial Structures (IASS)* (2015).

[10] Davis, H. *Visual Basic .NET Programming.* John Wiley & Sons, 2006.

[11] De Alwis, B., and Sillito, J. Why are software projects moving from centralized to decentralized version control systems? In *Cooperative and Human Aspects on Software Engineering, 2009. CHASE'09. ICSE Workshop on* (2009), IEEE, pp. 36–39.

[12] Eastman, C., Eastman, C. M., Teicholz, P., Sacks, R., and Liston, K. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors.* John Wiley & Sons, 2011.

[13] El Emam, K., and Koru, A. G. A replicated survey of it software project failures. *IEEE software 25*, 5 (2008).

[14] Glunz, B. F., and Munoz, Alfredo, F. Method and system for creating composite 3d models for building information modeling (bim). https://www.google.com/patents/US20150248504, 2015.

[15] Glunz, B. F., Pearson, Wayne, R., and Munoz, Alfredo, F. Method and system for creating 3d models from 2d data for building information modeling (bim). https://www.google.com/patents/US20150248503, 2015.

[16] Gyerik, J. *Bazaar Version Control*. Packt Publishing Ltd, 2013.

[17] Halt, G. B., Donch, C., Stiles, A., and Robert, F. *Intellectual property in consumer electronics, software and technology startups*. Springer, 2014.

[18] Hardin, B., and McCool, D. *BIM and construction management: proven tools, methods, and workflows*. John Wiley & Sons, 2015.

[19] Haukilehto, A. *Visual C .NET*. Edita Publishing Oy, 2002.

[20] Inc., A. Autodesk revit subscription, 2017.

[21] Janes, A., and Succi, G. *Lean software development in action*. Springer, 2014.

[22] Kallio, T. *Plan for implementing building information modeling: a customer survey*. Tampere University of Technology, 2014.

[23] KONE. Annual review, kone 2016. http://www.kone.com/en/Images/KONE_Annual_Review_2016_tcm17-37391.pdf, 2016.

[24] Lancaster, F. D., and Tobin, J. Integrated project delivery: Next-generation bim for structural engineering. *ASCE, Orlando, Florida 254* (2010).

[25] Lee, N., Dossick, C. S., and Foley, S. P. Guideline for building information modeling in construction engineering and management education. *Journal of Professional Issues in Engineering Education and Practice 139*, 4 (2013), 266–274.

[26] Lehtinen, T. O., Mäntylä, M. V., Vanhanen, J., Itkonen, J., and Lassenius, C. Perceived causes of software project failures–an analysis of their relationships. *Information and Software Technology 56*, 6 (2014), 623–643.

[27] Li, J., Wang, Y., Wang, X., Luo, H., Kang, S.-C., Wang, J., Guo, J., and Jiao, Y. Benefits of building information modelling in the project lifecycle: construction projects in asia. *International Journal of Advanced Robotic Systems 11* (2014).

[28] McLeod, L., and MacDonell, S. G. Factors that affect software systems development project outcomes: A survey of research. *ACM Comput. Surv. 43*, 4 (Oct. 2011), 24:1–24:56.

[29] Miller, R., Strombom, D., Iammarino, M., and Black, B. *The commercial real estate revolution: nine transforming keys to lowering costs, cutting waste, and driving change in a broken industry.* john wiley & Sons, 2009.

[30] Mohapatra, S. *Information Theory and Best Practices in the IT Industry.* Springer Science+Business Media, LLC, 2012.

[31] Nagel, C., Evjen, B., Glynn, J., Skinner, M., and Watson, K. *Professional C# 2008.* John Wiley & Sons, 2011.

[32] Naur, P., and Randell, B. Software engineering: Report on a conference sponsored by the nato science committee, garmisch, germany, 7th to 11th october 1968. In *Software Engineering: Report on a conference sponsored by the NATO SCIENCE COMMITTEE* (1969), Nato.

[33] Office, E. P. The european patent convention. http://www.epo.org/law-practice/legal-texts/epc.html, 2016.

[34] Ojala, J. *Interoperability in Computer Aided Design.* Aalto University, 2013.

[35] Pahl, G., Beitz, W., Feldhusen, J., and Grote, K.-H. *Engineering design: a systematic approach.* Springer Science & Business Media, 2013.

[36] Patent, U. S., and Office, T. Patent laws, united states code title 35 - patents. https://www.uspto.gov/web/offices/dcom/olia/aipa/PatLaws1214.pdf, 2000.

[37] Preisse, R., and Stachmann, B. *Git: Distributed Version Control–Fundamentals and Workflows.* Brainy Software Inc, 2014.

[38] Purdum, J. *Beginning Object-Oriented Programming with C#.* John Wiley & Sons, 2012.

[39] Regan, G. O. *A brief history of computing.* Springer Science & Business Media, 2008.

[40] Rooney, J. J., and Heuvel, L. N. V. Root cause analysis for beginners. *Quality progress 37*, 7 (2004), 45–56.

[41] Ruippo, T. *3D-tulostuksen kuluttajasovellukset.* Aalto University School of Engineering, 2013.

[42] Sauer, C., Gemino, A., and Reich, B. H. The impact of size and volatility on it project performance. *Communications of the ACM 50*, 11 (2007), 79–84.

[43] Sheldon, B., Hollis, B., Windsor, R., McCarter, D., Herman, T., et al. *Professional Visual Basic 2012 and. NET 4.5 Programming.* John Wiley & Sons, 2012.

[44] Slabaugh, G. G. Computing euler angles from a rotation matrix, 2010.

[45] Watson, K., Nagel, C., Pedersen, J. H., Reid, J. D., Skinner, M., and White, E. *Beginning Visual C# 2005*. John Wiley & Sons, 2008.

[46] Weygant, R. S. *BIM content development: standards, strategies, and best practices*. John Wiley & Sons, 2011.

[47] Woo, J. H. Bim (building information modeling) and pedagogical challenges. In *Proceedings of the 43rd ASC National Annual Conference* (2006), pp. 12–14.

[48] Young, N., Jones, S., Bernstein, H., and Gudgel, J. Smartmarket report on building information modeling (bim): Transforming design and construction to achieve greater industry productivity, 2008.