AALBORG UNIVERSITET
STUDENTERRAPPORT

Towards Object-Oriented Lifecycle Costing

Automating BIM objects quantity takeout for lifecycle costing of cleaning operations

10th January 2020

Aalborg University

Adam Piaskowski

Master of Science in Construction Management and Building Informatics

Master Thesis

Aalborg, Denmark

AALBORG UNIVERSITET
STUDENTERRAPPORT

Title:
Towards Object-Oriented Lifecycle Costing.
An investigation towards BIM object-based lifecycle costing of cleaning operations

Project:

Master Thesis Report, final semester.

Project period:

Sept. 1st, 2019 to 10th Jan. 2020

Email:
akptech@outlook.com

Name and signature:

Adam K Piaskowski

Supervisors:

Kjeld Svidt

Report page count:

55

Appendix page count:

47

Number of words:
22,000 (core text)

Number of characters:
191,000 (full document)

Finish date:
January 10th , 2020

Department of civil engineering
Thomas Manns Vej 23,
9220 Aalborg Øst

Version
1.14 (7th Jan. 2020)

Acknowledgments

I wish to thank Kjeld Svidt for his supervision. I wish every graduate to experience such facilitation.

Thanks to my colleagues who had been developing their reports in parallel. It was good to discuss the concepts and formatting with you.

I want to thank Rob Marsh from C.F Møller, for initiating the subject of this research, as well as for his valuable contributions regarding the lifecycle costing audit process. His remarks sparkled important considerations and pushed the project forward.

Thanks to Mathias Rasmussen from C.F Møller for testing the proposal through numerous meetings. His inputs and efforts had significantly impacted the project development.

Special thanks to Maria Saridaki and Kim Haugbølle from AAU Copenhagen, who developed the LCCbyg tool, as well as for an in-depth lifecycle costing research. I consider this work partly a continuation of Marias research, although she may not agree with some of my interpretations.

I want to thank all stakeholders who took their time to actively take part in this research, analyzing ideas and relating them to their circumstances.

Lastly, I would like to thank my father and my partner, who both spent countless hours listening to every presentation I made. They knew how much this meant before it all even started.

Dear reader, thanks for taking the time to read this report. I hope you enjoy it as much as I enjoyed writing it.

Abstract

By 2020, the Danish government will require mandatory evaluation of Lifecycle Costs (LCC) for new buildings, as opposed to only considering procurement costs. The maturing Danish building industry sector looks towards LCC as a source of competitive advantage. LCCbyg is a tool released by the Danish authorities to calculate lifecycle costs. Potential for automation is investigated, as currently, design model quantities need to be manually converted to facilitate the calculations in LCCbyg.

Multiple methods exist for calculating procurement costs at an object level. The same cannot be said about operational costs, where methods still rely simply on building areas. The generic approximations fail to account for object quantity, type, and materials. Therefore, the objective of this study was twofold; firstly, to investigate the potential for basing cleaning costs on objects, and secondly, to develop and test a data transfer method, integrating autonomous software packages for design and costing.

The involved stakeholders debated the potential risks and benefits of the object-oriented approach. Regardless of the procedure, transfer automation was deemed useful. A semi-automated data transfer was tested on an existing, and a design model of two office buildings using Building Information Models (BIMs) and a Visual Programming Language (VPL).

To conclude, the stakeholders saw the benefits of basing operational data on objects, while also pointed at obstacles concerning a lack of model detail in the early design phase. The proposed approach transferred Revit object quantities to the LCCbyg XML schema, and the operational cleaning cost data has been visualized using PowerBI. The method requires further integration with cost databases and other subsets of LCC calculations to be considered a complete solution.

Keywords: Lifecycle Cost, BIM, Data transfer automation, XML, Facilities Management

Table of Contents

Glossary

AEC	Architecture, Engineering and Construction Industry
BIM	Building Information Modelling, or Better Information Management
C.F.M	C.F. Møller - an internationally renowned Danish architecture company.
Cleaning	A task of maintaining a property frequently to appealing standard.
DGNB	The German Green Building Sustainability Rating – German: Deutsche Gesellschaft für Nachhaltiges Bauen (DGNB)
Dynamo	Dynamo or Dynamo Revit or Dynamo VPL is a Visual Programming Language plugged to the Revit API, enabling data manipulation.
UED	User Environment Design - See (Beyer & Holtzblatt, 1998)
Factor-based approach	An approach of calculating cleaning costs based on square meter areas, in accordance with the DGNB proposed approach.
Family parameter	A parameter is unique to a Revit family. It can be accessed from the project level, but may not be deleted without opening the family environment.
Instance parameter	A parameter with a value unique to every object instance.
Interoperability	A flexible data interface, enabling unidirectional or bidirectional communication between two software environments.
KPIs	Key Performance Indicators.
LCA	Lifecycle Assessment, an assessment of lifecycle potential of the building asset and its' components
LCC	Lifecycle Costing, an assessment of the lifecycle cost of the building asset and its' components.
LCCbyg	LCCbyg is a tool released by the Danish Building Research Institute (SBi), at Aalborg University (AAU) on behalf of the Danish Transport, Construction and Housing Agency
Object-orientation	Holding data on an object following a hierarchy, where the object is a parent element, and the information is the child element.
Object-oriented approach	The proposed approach of quantity takeout for calculating cleaning costs based on various units.
OOTB	Out Of The Box – a function contained within the software standard interface.
Operational phase	The period of time that the relevant part of the authorized development is in operation after construction and commissioning is complete
Python 2.7 & 3.7	A coding language compatible with Dynamo VPL.
Revit 2019	A software environment used across the AEC industry for BIM.
Revit Family	A name for an object type used in the Revit software environment
Shared parameter	An exportable parameter stored in a .txt file accessed and shared through multiple Revit projects.
Type parameter	A parameter with a value common to all objects of the same type.
UI	User Interface
XML	eXtensible Markup Language.

List of Tables

List of Figures

List of Figures in the Appendix

1 Introduction

Approaching the end of the second decade of the 21st century, it is no longer valid to solely consider initial construction costs when making design decisions. The availability of poorly designed, low-quality materials and components from all over the world poses a direct threat to sustainable building usability (Shan, Melina, & Yang, 2018).

Western societies seek ways of creating long-lasting value, despite the higher upfront cost, whereby long-term benefits outweigh initial purchase costs. Developed societies such as Denmark take lifecycle costs of components and materials, as long-term benefits often outweigh initial purchase costs. It is no longer just about the price, as early design decisions influence the built environment and its residents often for as long as half a century (Green Building Council, 2013).

Nowadays, the materials used for building construction must undergo a much more rigorous and conscious investigation, a lifecycle assessment (LCA), which will consider the building ecosystem holistically, in terms of procurement, operation, disposal and reuse (Vigovskaya, Aleksandrova, & Bulgakov, 2017). As far as a quantitative point of view is concerned, economic impact can be evaluated using the lifecycle cost (LCC) calculation of the building individual components lifespan, as well as the building as a whole (Maria Saridaki, Psarra, & Haugbølle, 2019).

LCC can be as much about economics as it can be about sustainability, social lifecycle, and environment, combined, these factors can be converted to currency, a commonly agreeable way of expressing human effort in dealing with value created, as well as the cost incurred to maintain a good standard of the buildings' usability.

Uncertainty of results is one of the most significant limitations when attempting to calculate the lifecycle costs accurately. Gluch and Baumann (2005) identify three categories of uncertainties; physical, business-related, and institutional. The physical uncertainties relate to material quality and predicted lifespan, while the business-related to the applicability over the long-term, or a ban of the material from the market. Furthermore, the institutional relates to the detail which would account for each anomaly accurately may be lacking, due to generic models used during the calculation.

The term lifespan is categorized into four distinct categories; the economic, the technical, the physical, and the utility (Kirk & Dell'Isola, 1995). The economic lifespan of a building refers to its profit-generating years, the technical lifespan relates to the same technology used in the building, the physical lifespan relates to the building usability by the users, and finally, the utility lifespan relates to its compliance to the current building standards. Each of those may require a sooner replacement of materials whose physical lifespan is still ongoing. The economic lifespan is the one most used during LCC calculations (Maria; Saridaki & Psarra, 2017).

As a building is a long-term asset, and its lifespan can vary, depending on its design, materials, craftsmanship, and life-long maintenance, it is the combination of those factors that ensure a prolonged economic lifespan. Adequate maintenance during the operations phase is often wholly under-looked, undervalued, and under-costed, resulting in worse material performance and faster building degradation. According to (Guillen et al., 2016), up to 60% of the total lifecycle cost belongs to the post-occupancy phase. This figure may be optimistic, given the detail entailed in the calculations.

An architect can enter a blue ocean of new services when offering additional value, generated from LCC combined with Building Information Modelling (BIM). As more building types become subject to requiring BIM models and IFC models to transfer information to the operational phase (Kiviniemi, Tarandi, Karlshøj, Bell, & Karud, 2008; Rosendahls, 2019), and interoperable data specifications are becoming more standardized in practice (Kiviniemi et al., 2008), it remains a task to find efficient and effective ways of calculating translatable cost incurred during the lifecycle of a building.

Object-oriented parametric modeling used in the BIM models has another benefit; the operational maintenance information can be grouped and explicitly attached to the components, thereby ensuring adjusted quantification of cost and case-specific optimization. Furthermore, it can be used to visualize, simulate, and generate value-adding intelligence, offering significant improvements in the FM operational processes (Olatunji & Sher, 2010).

While there are many BIM environments used by the architects to build 3D BIM models, ultimately, the model information must be accessible to the building owners so that the data can be further re-used during the operational phase. The auditors can use standard file formats to transfer data between independent software environments (Sacks, Eastman, Lee, & Teicholz, 2018).

The UK based Constructions and Operations Building Information Exchange Format (COBie) is a suitable method for transferring project information in a standardized way, accessible by 3rd party Facility Management (FM) software, enabling storage of quantity and relevant information needed for the building component operations and maintenance data (Patacas et al., 2015; thenbs.com, 2016).

The so-called "data drops" can used by external applications to calculate and display data relevant to the building owner. COBie framework could support such information transfer, as COBie attributes can be mapped to specifications (IBM, 2019).

As COBie does not support the transfer of geometric model data, yet stems from the geometry information in the form of an Excel spreadsheet, a method for translating 3D geometry information is to use Industry Foundation Class file format (IFC). The IFC format provides a standardized model data schema, which can be read and used by a vast amount of FM software.

IFC qualifies the model to be stored in a standardized, internationally agreed format, which ensures the data will be safe for use by third-party stakeholders for many years to come. Furthermore, having been agreed by the ISO (ISO 16739-1:2018, 2018), the file format, as well as the accompanying documentation provided by the buildingSMART initiative (BuildingSMART, 2013), serving as a non-profit organization, maintaining the format applicability and cross-platform interoperability.

IFC is a rigid schema that comes out in different versions, and roughly covers 90% of all required exchanges; however the remaining percentage may be vitally breaking the value of the model after it had been exchanged (Wix & Karlshøj, 2010; Yu, Froese, & Grobler, 1998).

What is interesting is that IFC files are interoperable and contain accessible, and human-readable source code. It is, therefore, possible to query the contents of the file. It is also possible to append custom data directly to the source code, to an existing model, and therefore make changes that may be updated over time (Kim et al., 2017; Toth, Janssen, Stouffs, Chaszar, & Boeykens, 2012).

When using the IFC schema, the models are not restricted by the Revit licenses, nor the architects and engineers designing the models can revoke their copyrights. This allows the building owner to use the IFC file throughout the lifespan of the building.

Model View Definitions (MVDs) is a subset of the overall IFC schema, narrowing the scope of Exchange Requirements (ER), which explicitly specifies the information needed for IFC data transfer (BuildingSMART, 2010). The Information Delivery Manual (IDM) provides a standardized method for process documentation. IDM enables third party FM software, to use IFC model data to optimize the building during operations phase (Kang & Choi, 2015; Kim et al., 2017; Lawrence, Pottinger, Staub-French, & Nepal, 2014; Patacas, Dawood, Vukovic, & Kassem, 2015).

1.1 Motivation

Having worked as a BIM manager, hand to hand with architects, and as a member of a maintenance team for building cleaning operations, personal insights are presented on how the industry could benefit from more accurate costing approaches for facility cleaning.

From personal observations, the architects pay limited attention to the operational phase and often hire external consultants to manage waste and janitorial services. On the other side, the companies later facilitating the buildings, often rely on general figures when estimating the costs of building maintenance and planning the cleaning sequencing.

Generic estimations may lead to inadequate contract value approximation, which is burdening either the shoulders of the operational personnel or on the other side of the spectrum, form additional costs to the building owner. It also leads to quicker degradation of components due to inadequate frequency, or reduced service quality due to limitations imposed by the contract value. Thus, having a detailed description of the model and objects contained within the BIM model allows for more accurate cost estimation, offering a possibility of operational cost reduction.

1.2 Research Goals

The investigation aims to show the current approach of estimating the lifecycle cost proposed by the DGNB TEC1.5 (Green Building Council Denmark, 2017, pp. 385–387) is insufficiently detailed to inform the design and further operations effectively. As a proof of concept, this Master Thesis will investigate how seemingly similar building scenarios can show varying lifecycle costing results, depending on the equipment in use, and the calculation approach.

Furthermore, the information gathered during the design phase can be of significant value when reused by Facility Management (FM). Current practices utilized by the FM teams are merely reactive and, therefore, inefficient (Sullivan, Pugh, Melendez, & Hunt, 2010). Creating a reliable database for the operations and maintenance is deemed necessary, and approaches to utilizing BIM models have been tested (Akcamete, Akinci, & Garrett, 2019), using rooms, yet not objects.

1.3 Research Questions

Q1. Is there a possibility the current approach of estimating cleaning lifecycle costs is inaccurate?
Q2. Will the proposed approach increase the complexity of calculations experienced by the auditor?
Q3. Can the lifecycle cost data transfers be automated to prevent extracurricular tasks experienced by the auditor?
Q4. Do the stakeholders agree that object-oriented approach could improve their workflows and generate value?
Q5. Can a prototype include the functionality of a working software product?

2 Methodology

The study is based on a combination of methods. To address the qualitative research questions existing literature on the subject was reviewed, followed by extensive stakeholder interviews. Personal insights into the cleaning trade had been an inspiration for this study and had additionally influenced the prototyping. The Contextual Design (CD) methodology was used to structure the investigation.

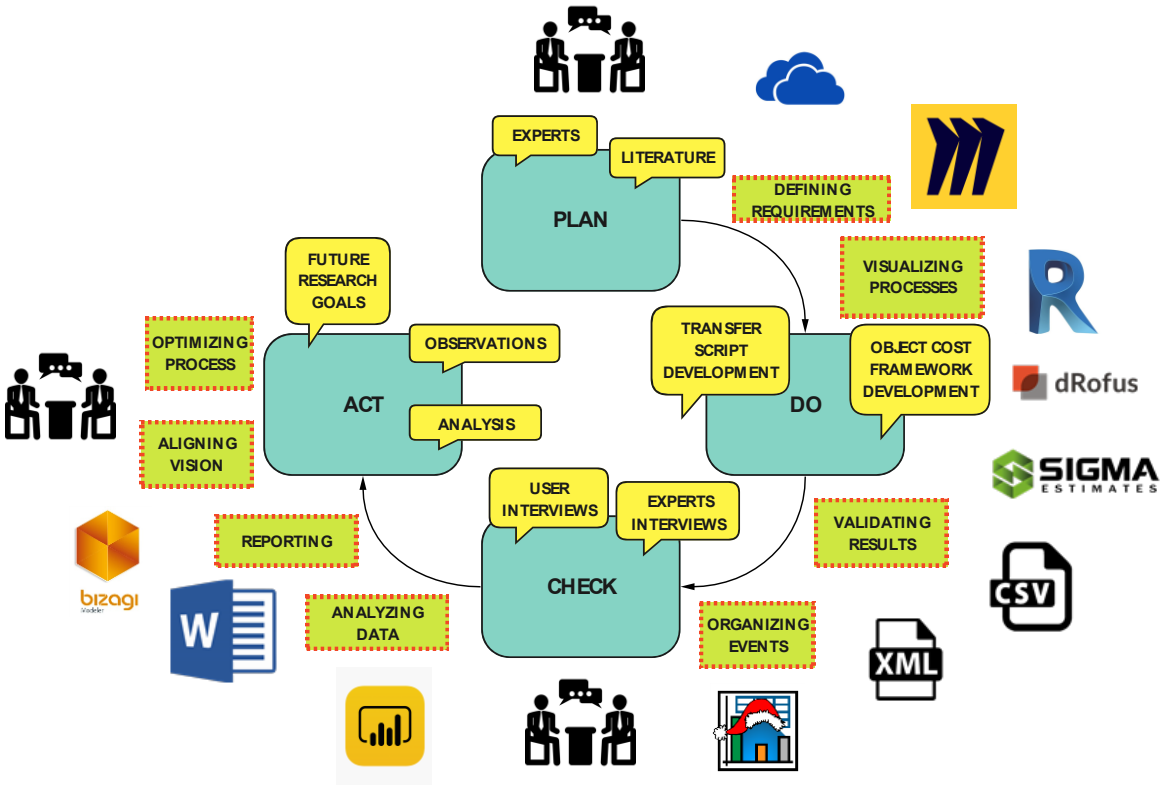


Figure 2.1 The implementation of Plan-Do-Check-Act (PDCA) in the methodology

To implement cost information into the Revit model, two alternatives are presented. One method is to use a V&S price-book, and the second is to use a dRofus proprietary project database. Both ways use classification coding to identify objects and link them with the database counterparts. The subject is investigated in chapter 5.1.

The investigation of BIM models , programming of transfer scripts, and supporting software were carried out in the form of prototyping and case studies. The prototype transfer was tested on two case studies, and the results were presented in two different tools; LCCbyg for LCC analysis and PowerBI for data visualization.

The Autodesk Revit 2019 Environment (Revit) enabled populating the project with parameter placeholders needed to run the lifecycle cost analysis. Furthermore, Dynamo 2.05 for Revit (Dynamo) was used. To link object parametric data with external software, Visual Programming Language (VPL), was used to map quantities with the LCCbyg XML data schema. The XML file is used to transfer generated reports from the Revit model, directly into LCCbyg. More on the software will be described later in chapter 5.3.

2.1 Literature Review

The literature review was carried out to assist the problem development and to outline alternative solutions to the solution discovery. The literature review was performed using the Aalborg University Library called Primo, based on keyword search, using a range of strings and intermediary keywords such as OR XOR, AND, Asterix*, concerning the following keywords: “BIM LCC”, “Lifecycle Cost”, “DGNB”, “Sustainability”, “Operations”, “Maintenance”, “Cleaning”, “FM”, “Facility Management”, “Python”, “Dynamo”, “LCCbyg”, “SBi”, “Classification”, “Design & Build”, “Data automation”, “Object-Oriented”, **“**Building Information Modelling”, “Better Information Management”, “Building Information Model”, “IFC”, “COBie”, “XML”, “XML Schema”, “Future Value”, “Lifecycle Assessment”, “LCA”, “ISO 15686-5:2017”.

The abstracts were chosen and assessed for relevance. Mendeley referencing software was used to gather sources. 92 sources were added to the library, and 65 are directly referenced in this report. The bibliography mainly consists of research articles from trusted journals, as well as books. References to websites and standards are supporting the primary literature. The areas of interest revolved around the following goals:

- To identify existing approaches to cleaning and maintenance.
- To identify classification systems needed to classify components.
- To identify possibilities for automating the data transfer.
- To identify DGNB prerequisites, operational basis, and available databases supporting the lifecycle cost for cleaning estimations.
- To identify existing BIM integration with FM operations.

2.2 Contextual Design

The CD methodology paves the way for a solid grounding of the methods and applications in a real environment, and puts it through a set of testing environments, from the initial contextual inquiry, i.e., figuring what is essentially the problem, through to visioning, i.e., what could potentially solve the problem, to storyboarding, i.e., creating a digestible explanation of the vision to the stakeholders, through to stakeholder engagement.

The stakeholder engagement can be further subdivided into groups of development. Firstly, identifying the stakeholder groups, requesting interviews, to conducting the interviews. The initial stakeholder engagement is thoroughly described in the Contextual Design Methodology (Beyer & Holtzblatt, 1998).

The subgrouping of solution testing is further divided, depending on the development and the rigidity of the solution. The first stakeholder presentations encapsulate presenting the idea and gathering initial feedback, while further development concerns prototyping and presenting live demos of the proposal to the stakeholders. According to the CD methodology, the prototyping should only follow after the UED; however, a deviation from this sequence is made, as the prototype (Chapter 8) is developed right after visioning. Having the prototype early in the process, enabled illustrating the goal of the research more clearly, and encouraged the stakeholders to see the applications of the tool during live demonstrations.

Having developed a prototype, initial testing and feedback can be directed towards a more rigid solution, which in turn can be further tested by the stakeholders – and ideally, be the end-users of the software application. At this stage, the end-users should be able to try the proposed solution and see if they can merge it with their existing workflows. This stage further illustrates the functionality and layout of the tool and is described in the User Environment Design (UED) chapter 7.

3 Background

The following chapter is investigating lifecycle costing (LCC), its significance in the design and operation of the building, and connections to sustainability certification. Moreover, background on cost calculating approaches for operational cleaning is investigated.

3.1 Lifecycle Assessment and Costing

Lifecycle costing was first used in the US in 1933 for purchasing agricultural tractors by the General Accounting Office. Back then, it was understood that the attention was brought to the initial purchase price, not the lifecycle of the tractor. The operational cost over time can be much higher than the initial investment (Thiebat, 2019). In the 1970s, the US Department of Defense drafted directives calculating LCC of expensive military equipment during the development phase. Already then, investments were based not only on the initial, but also on the Operations and Maintenance (O&M) cost of the asset (Lichtenvort et al., 2008).

Lichtenvort et al. (2008) are distinguishing between three types of lifecycle costing – the Conventional, Environmental, and Societal. He also suggests a need for different perspectives as each method is lacking some scope on the others, and the processes need to include geographical differences, exchange rates, discounting, etc. The point being that scoping and defining lifecycle costing can be a difficult challenge, and there are many aspects which this thesis will not investigate. This thesis is concerning the maintenance and operations costs, precisely indoor cleaning operations, which, due to the repetitive nature, can significantly drive costs or offer desirable savings.

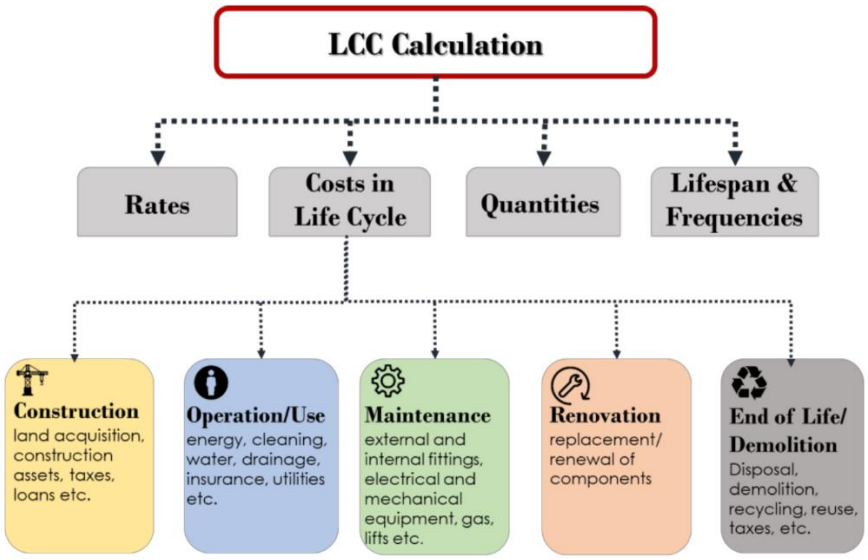


Figure 3.1 Components of lifecycle cost calculation

Source: (Maria Saridaki et al., 2019)

From the administrative standpoint, lifecycle costing will be required by law in 2020 (Rosendahls, 2019), by the Danish Transport, Building and Housing Agency (SBi) ("SBi," 2019). Each new building has not only the initial Design&Build cost considered but also the lifecycle cost of the overall asset's lifespan.

From a practical standpoint, the building owners may require sustainability ratings to ensure that the buildings they invest to be built will be created, following the modern standards. To evaluate if the building is being built sustainably, building certification has become increasingly valuable. Amongst many three certificates stand out: The German Green Building Sustainability Rating – German: Deutsche Gesellschaft für Nachhaltiges Bauen (DGNB) (DGNB.de, 2019), the US made Leadership in Energy and Environmental Design (LEED) and the British Building Research Establishment Environmental Audit Monitoring (BREAM) (Goldstein, Herbøl, & Figueroa, 2013).

According to Goldstein et al. (2013), each tool contains gaps in either embedded energy impacts not being optimized for BREAM and LEED and lacking full Lifecycle Assessments (LCAs). What is a key highlight that proves this thesis point is that although the DGNB satisfies a full-LCAs, it is weakened by the generic data use. Although the DGNB does require building-specific data, precise results are still somewhat ambiguous due to the generic information contained within.

As the DGNB is arguably the most accurate and developed certificate, it is being used by over 100 Architecture Engineering & Construction (AEC) companies in Denmark. A full list of companies affiliated with the DGNB certificate in Denmark can be found here: (Green Building Council, 2013).

3.2 Lifecycle Costing as a part of DGNB

The DGNB is split into five main categories, giving 22.5% to 4 main categories and 10% of the total score going to the process quality category. The four main categories consist of Environmental Quality, Economic Quality, Sociocultural & Functional Quality, and the Technical quality, each equally having a 22.5% say in the total building evaluation score (Green Building Council, 2013).

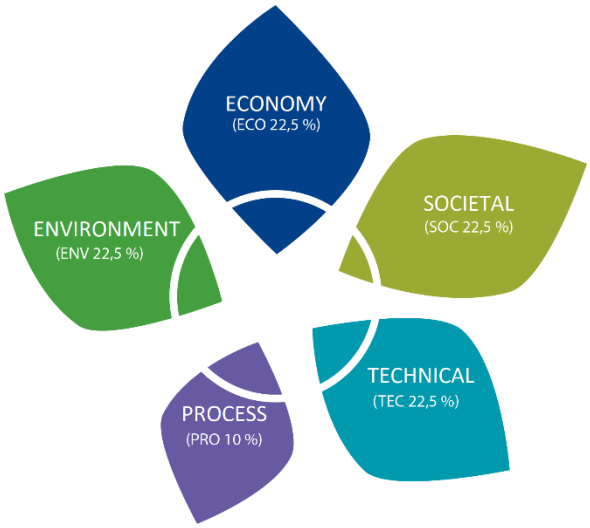


Figure 3.2 Division of categories influencing the total sustainability rating

Source: (Green Building Council, 2013)

Although environmental and socio-cultural qualities are split into more than ten subcategories, while the Technical quality is divided into 5, the economic quality is split to only two subcategories – lifecycle costs and suitability for third-party use (DGNB.de, 2019).

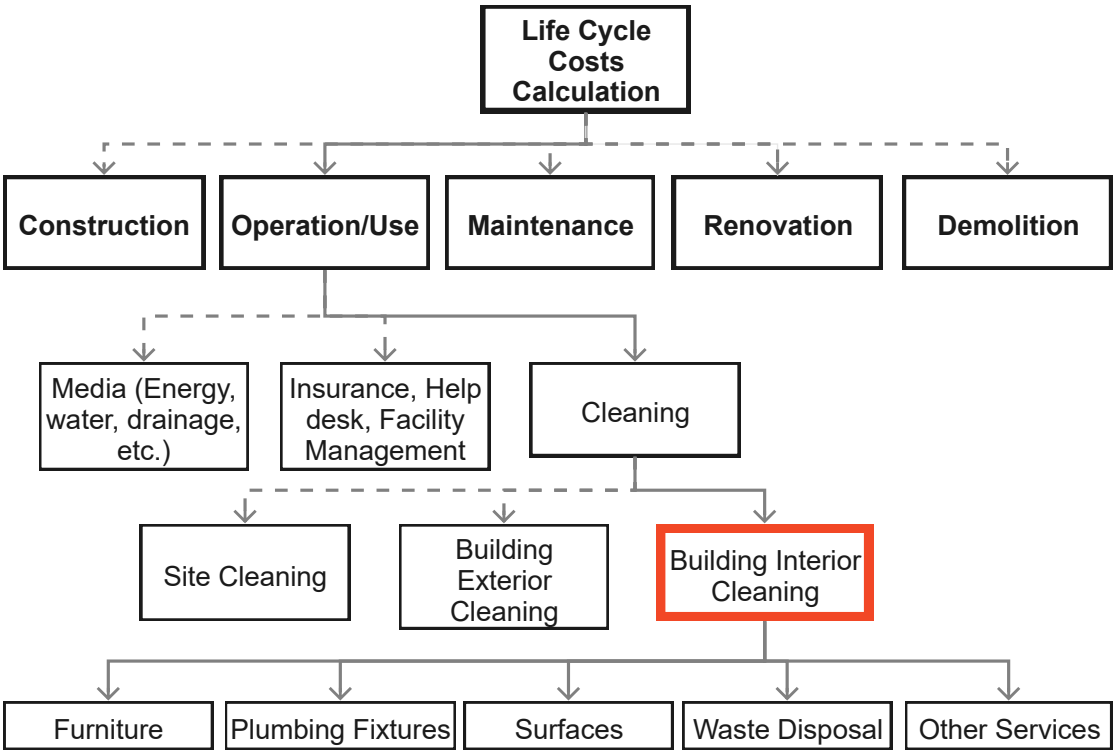


Figure 3.3 The scoping diagram for thesis research

Highlighted in red is the scoped area of research for this report, while the children are some categories amongst many that have the potential to be objectified.

Therefore, the category for lifecycle costs accounts for almost 10% (9.6%) of the overall score (Green Building Council Denmark, 2017). It highlights how significant the lifecycle costs are and justifies why such calculations shall be performed in the first place.

The DGNB focuses on building for people, giving users satisfaction, and ability to reduce costs and optimize processes in the operation phase. Along with the DGNB, the Danish cleaning standard DS/INSTA 800:2010 is responsible for outlining the national level for basing the standard of cleaning (*Dansk standard DS/INSTA 800 Rengøringskvalitet-System til fastlaeggelse og bedømmelse af rengøringskvalitet Cleaning quality-System for establishing and assessing cleaning quality*, 2010). This document is a reference point for quality standards and quality control during operations.

The categories are also interlinked – the lifecycle costs related to cleaning have an influence on the suitability for third-party use (9%) and on the ease of building cleaning and maintenance (5%), as well as other categories where it is compounded, such as optimization and complexity of the planning method, awarding of tenders, conditions for optimal use and management, integrated planning and commissioning, altogether accounting for further 2-6% of the total score.

Therefore, arguably, the cleaning strategy, the object-oriented lifecycle costing, the integrated planning, and suitability for third-party use during operations, all come to form around 15-20% of the overall score, hence find grounding for cleaning operations, an essential aspect of a building, relative to lifecycle costs, integrated planning, and operational building management.

3.3 Lifecycle Costing in Design & Operations Phase

The primary difference between design and operations is that during the design, the project is not yet a physical asset, and can, therefore, be altered with greater ease (DGNB.de, 2019).

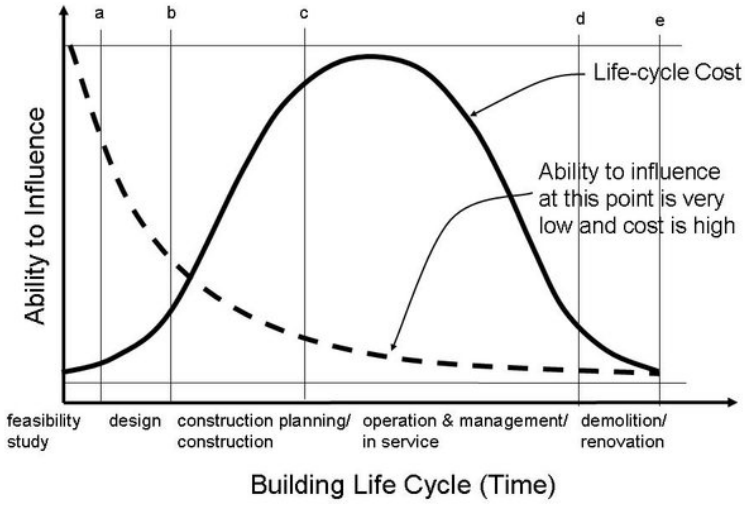


Figure 3.4 The ability to influence design relative to the building development

As seen above, during design and pre-certification, it is much simpler to optimize building performance by benchmarking alternative solutions as early as possible.
Source: (Conradie & Roux, 2008)

During the design phase, calculating lifecycle costs makes the most sense, as the outcomes may influence the decisions taken by the building owners to optimize the design relative to facility maintenance. It is, however, also possible to calculate lifecycle Costs during the operational phase, to inform the layout of the building, by arranging the movable assets, and by optimizing the use of spaces and their maintenance (Huizenga, Hui, Duan, & Arens, 2001).

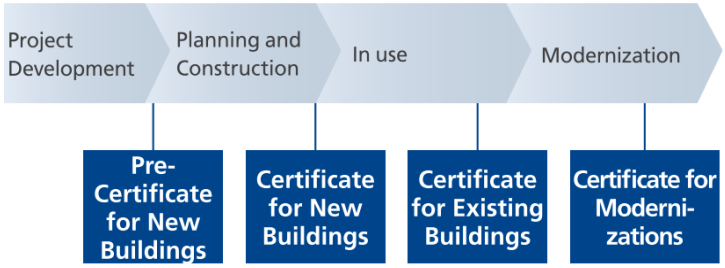


Figure 3.5 Building certification at different stages of the building lifecycle

Source: (DGNB.de, 2019)

It is in the operations phase when the LCCs are being influenced by the way the asset is maintained, hence the information from the design phase is a foundation for future optimization, 4D scheduling, and task management. Data at an object-level may have future benefits during the operations phase, where each object can have its tracked history, operational parameters, and relative location represented in the BIM model. Further discussion about future benefits can be found in the appendix 14.12. To facilitate the information transfer, specific software is required to both derive the objects and calculate the cost.

4 Costing Approaches for Operational Cleaning

There are multiple ways of calculating the lifecycle costs and costs in general, depending on the level of detail, fixed and variable conditions, limitations, and scope. The cost databases may vary depending on the geographic, social, and economic conditions of a particular region. As the Thesis area is in Aalborg, Denmark, prices will be derived using local currency – Danish Krone (DKK), which is worth around 13 cents to a Euro (EUR) as of October 2019. In the following subchapters, the existing approach is firstly presented, followed by the proposed approach, and then the two approaches are compared against each other. Supplementary material to costing can be found in the appendices – chapter 14.2.

4.1 Existing DGNB Cost calculation

The existing costing approach is directly taken from the DGNB LCCbyg template, which stems from the TEC 1.5 (Green Building Council Denmark, 2017) of the DGNB costing manual. The costing approach is based on the areal sum of square meter (m2) units for door areas, including skirtings, window areas, stair areas, and floor areas. Furthermore, the floor areas are subdivided into categories, such as bathrooms, office spaces, and hall spaces. The floor surfaces are then divided into material surfaces, i.e., carpets or hardwood floors, and subsequently, the price variation depends on the intensity of cleaning, i.e., light, average, demanding, with corresponding explanation supplemented in TEC 1.5. The result is therefore adjusted by providing the following variables:

1. Category (doors, windows, stairs, floors)
2. Material type (hard floor, carpet)
3. Quantity (m2)
4. Frequency (days/year)
5. Intensity (Light, average, demanding)

4.2 Object-Oriented Cost calculation proposal

The proposed approach uses information based on objects. Although the variable parameters are deemed to be kept as closely similar to the original DGNB factor-based approach, there is a possibility of achieving a greater accuracy due to calculating the count of individual objects within rooms. It is possible to see differences between seemingly similar rooms in terms of area and function, yet having a different layout, or different equipment within.

The approach uses similar parameters to the ones proposed by the existing DGNB template, with minor changes. Below is a list of variables that will presumably complete the calculation.

1. Category (flexible, i.e., doors, windows, stairs, floors, walls, fixed furniture, furniture, plumbing fixtures, lighting fixtures, skirtings, railings)
2. Material type (flexible, i.e., any material)
3. Quantity (m2, LBM, m3, units)
4. Frequency (days/year)
5. Intensity (Light, average, demanding)

The object categories and materials remain flexible, enabling the application of various components specific to the building type and application, as well as using various materials, which may have its cleaning cost adjusted beyond floor washing and carpet vacuuming. The intensity parameters can also be case-specific and adjusted according to a min-average-max scale for each building or building type.

Keeping the quantity, frequency, and intensity variables fixed to the original template offers the potential to re-apply already existing parts of the LCCbyg software, as well as much of the previously gathered knowledge needed to satisfy the DGNB LCC calculations.

4.3 Factor-based vs. Instance-based vs. Type-based quantity takeout

The calculations provided by the DGNB template account for a moderate level of detail. The quantities are mainly split into a few factors and parameters. Figure 4.1 illustrates the grouping of categories contained within the subset. In the existing DGNB template, the toilet has no detail, beyond defining whether its use will be light, ordinary, or demanding.

Name	Unit	Quantity	Unit price (DKK)	Frequency (per year)
Buildings, internal				
Space				
Doors, window sills, skirting boards, etc.; cleaning - normal	m2	226	2.00	26.00
Floors, hard; cleaning - normal	m2	864	1.30	100.00
Floors, hard; cleaning - demanding	m2	130	2.50	100.00
Toilets/baths; cleaning - normal	m2	49	6.00	252.00

Figure 4.1 Factor-based quantity takeout – LCCbyg 2.2.52, Screenshot

The Instance-based approach undertakes instantiating rooms so that higher occupancy rates can be differentiated from those with lesser occupancy. The automated data transfer can name instances accordingly to Room IDs, function, floor finish material, and cleaning intensity. The greater detail aims for a more accurate quantity takeout.

Name	Unit	Quantity	Unit price (DKK)	Frequency (per year)
586542/Toilet/Tile antiskid/Demanding	m2	3	2.80	252.00
586545/Toilet/Tile antiskid/Demanding	m2	3	3.00	252.00
586548/Toilet/Tile antiskid/Demanding	m2	3	3.00	252.00
586551/Toilet/Tiles/Demanding	m2	5	2.50	252.00
586554/Toilet/Tiles/Demanding	m2	4	2.50	252.00
586560/Toilet/Tiles/Demanding	m2	7	2.50	252.00
586564/Toilet/Tiles/Demanding	m2	4	2.50	252.00
586567/Toilet/Linoleum/Demanding	m2	2	2.10	252.00
586570/Toilet/Linoleum/Demanding	m2	2	2.10	252.00

Figure 4.2 Instance-based quantity takeout – LCCbyg 2.2.52, Screenshot

The type-based approach is an attempt to find a balance between simplicity and accuracy. The combined object types use units to calculate all instances within a building.

Name	Unit	Quantity	Unit price (DKK)	Frequency (per year)
Family Type: Cleaning_Bin_Toilet/ Family: Cleaning_Bin_Standard/Light	units	12	3.00	100.00
Family Type: Test bin123412341234/ Family: Cleaning_Bin_Standard/Light	units	2	2.00	100.00
Family Type: Cleaning_Desk_Chair/ Family: Cleaning_Chair_Standard/Light	units	198	1.00	100.00
Family Type: Cleaning_Lounge_Chair/ Family: Cleaning_Chair_Standard/Light	units	3	4.00	100.00
Family Type: Cleaning_Conference_Tabler/ Family: Cleaning_Conference_Tabler/Light	units	4	6.00	100.00
Family Type: Desk_Meeting/ Family: Desk_Standard/Light	units	1	5.00	100.00
Family Type: Desk_Meeting_800x1400/ Family: Desk_Standard/Light	units	2	2.50	100.00

Figure 4.3 Type-based quantity takeout – LCCbyg 2.2.52, Screenshot

This visualizes cleaning costs on per room basis. This approach is possible, thanks to the structural composition of BIM models. The underlying technology is investigated in the following chapter.

5 Software and Technologies

To facilitate the BIM model quantity takeout for lifecycle costing, the SBi has created a freely available software called the LCCbyg (Haugbølle et al., 2017). More on the software can be found in chapter 5.3. As of the writing of this thesis, there is no direct plugin which can transfer quantities from a Revit model, directly to LCCbyg. Therefore this process is currently transferred manually, or MS Excel spreadsheets are used as opposed to the LCCbyg software (Maria Saridaki et al., 2019).

Manual data entry may take a considerable amount of time to transfer. Additionally, the accuracy of manual data transfer is questionable, as input errors are inevitable (Piaskowski, AK, Petersons, R, Wyke, SCS, Petrova, EA & Svidt, K, 2019).

Thankfully, the LCCbyg calculation report uses a .xml format enabling open access to its template reports. It is, therefore, possible to access the contents and append the structure using Dynamo Revit and Python scripting. This chapter will explain further, the underlying technologies needed for data transfer automation and the technologies used to facilitate data transfer. Firstly, the application of BIM relative to the subject of LCC quantity takeout is explained, followed by cost and object databases, VPLs, and analyzing and visualizing tools. Lastly, open standard technologies are described as alternatives to the proposed workflow.

5.1 Building Information Modelling

Building Information Modelling (BIM), or Better Information Management (BIM), (Eastman, Teicholz, Sacks, & Liston, 2018) both form a backbone for object-oriented building modeling. Building Information Modelling is concerned with geometrical and parametrical, object-oriented modeling of digital twins (Kaewunruen & Lian, 2019).

Better Information Management is about what to do with the parametric models and information attached to objects later on, during the design and operation. Both concepts are what enable a more detailed approach , as opposed to non-parametric CAD.

The hierarchical relationships help establish project breakdown and permit objects such as furniture and plumbing fixtures to be hosted by spaces (rooms in Revit), which can then share common properties of a particular room type, thanks to the concept of inheritance. Inheritance in computational terms is when a child inherits properties of the parent, i.e., the furniture inherits the location of the room it is located in (Ugwu, Kumaraswamy, Kung, & Ng, 2005).

The objects within rooms can further be categorized by the properties the room shares with a building zone, the zone with a building level, and the level with the building envelope. Further, the building relates to the site it is built on, and the site can relate to its geographical placement within the city. (Christian Koch, 2018).

This way, information about assets can be neatly organized, and the level of detail from the city scale to the object scale be distinguishably associated with objects. The data can then be utilized in a range of tests and optimization models.

Autodesk Revit Models

Autodesk Revit is a proprietary BIM environment tool, which encompasses a wide range of external tools that can plug into its API and feed in information to or from the model (Kensek, 2014). Tools like Dynamo use the Revit API and display it in a graphical interface.

Revit rooms can be exported to external software that may further process quantities and specifications. Linking Revit models with external databases facilitate the standardization of workflows and ensure that

the models use accurate and reliable data from a single source of reference. Linking requires classifying objects. What is significant is that Revit supports a certified export using the Industry Foundation Class format (IFC), used by many FM software tools during the operations phase (Autodesk, 2018; ISO, 2013).

Classification systems

A classification type parameter must be provided to connect cost information from an external database with the Revit model. An exception is to use Globally Unique Identifiers (GUIDs); however, those apply to instances at a modeling level. Type properties may also contain GUIDs, just like any other properties; however, GUIDs are not human-readable (Mendes De Farias, Roxin, & Nicolle, 2018).

On the other hand, classifications can be recognizable as their identification is standardized. Three most commonly used classification systems in Denmark are investigated – the SfB classification (Samarbetskomitén för Byggnadsfrågor, 2012), the Cuneco Classification System (CSS) (BIPS, 2015) and the BIM7AA classification system (BIM7AA, 2018).

Because the BIM7AA classification is based on SfB, and both the Sigma Estimates and the LCCbyg software uses SfB to classify objects, SfB had been chosen as a classification for the prototype development. Many tools facilitate automatic object classification. For the sake of project testing, manual classification was entered. Though, using an automated classifier significantly reduces the effort needed to classify objects, as well as ensures no typos break the consistency.

Following the classification, the objects can now be linked to the database placeholders in proprietary databases such as dRofus or Sigma Estimates. The advantage of using proprietary databases is that the links to the Revit model are premade, and further automated connection can be utilized, thus saving effort and increasing transfer accuracy.

Below are two possibilities of cost data placement explained, either directly on objects, using the dRofus template, or in a cost database associated with the objects using the classification code.

dRofus

dRofus is an integrated design validation of the building requirements between the Revit model and the initial clients brief. The purpose is to plan, create, and manage building data from various stakeholders, providing workflow support and building information from multiple phases.

It can save employers' exchange requirements (EIR) and validate the design models against the database. It integrates bi-directional data synchronization, including planning data, geometric data, parametric data, and accessible format documentation links to supplementary images and PDFs.

dRofus is a solution for enhancing architects' capabilities, packaged in a neat and intuitive interface. The architects are using this software as a database storing BIM geometry and information. Its purpose is further extended, to control accessibility and versioning, and to facilitate model checking against building code requirements and the brief.

The significance of dRofus for this project is twofold. One, it is a part of the workflow used by the architects' office with regards to the context of the case study, and two, the software stores parametric information within the template database, and through a bidirectional link with Revit, it can populate the data to multiple models.

The price and frequency data can be populated within the dRofus room and object database, and be linked to the Revit models using the aforementioned classification methods. This way, data transfer has two advantages. One, it can be incorporated into existing processes, and two, a higher degree of automation, reduces manual input while preventing entry errors.

V&S Price books using Sigma Estimates

An alternative to dRofus is to use another database. Sigma Estimates –software capable of a bi-directional link between the Revit model and the V&S Price books, enables a similar workflow. The software has the capability of linking the Revit model with the price books by utilizing its in-built out of the box capability to link project data using classification coding (Maria; Saridaki & Psarra, 2017).

The significance of V&S Price books for this project is the following. The price books are a standard method for costing used by the Danish AEC sector, and thereby its reliability is well validated over the years. Secondly, the price books are maintained and adjusted for inflation and other factors, adjusted yearly to meet current market conditions (Molio, 2019).

To the authors' knowledge, no cost database exists for object-based cleaning. Therefore, linking a price book would require creating a price book in the first place. This is beyond the scope of this thesis. Once the research is conducted on the prices, linking the price book with a Revit model, using the Sigma Autodesk Integrations plugin, is a well-documented process, and therefore it will not be described further. Please refer to (Maria Saridaki et al., 2019) or Sigma Estimates official website for further guidance.

Data visualization

Business Intelligence tools permit an executive view at options by displaying the data in a presentable and easily digestible manner. Key Performance Indicators (KPIs) can be used as metrics for representing how the costs may influence the design. There are various tools to choose from, both open source and paid versions. These tools usually consist of Ad Hoc reports, dashboards for displaying data, KPIs, strategic planning, and visual analytics possibilities. It is generally possible to view the data through the cloud, and the data can be accessed using both mobile and stationary platforms. Due to the integration with the Microsoft Office package, the PowerBI tool was chosen for this project.

5.2 Interoperability using Visual Programming Language

Dynamo is a Visual Programming Language (VPL) which directly plugs to the Revit API. It uses the Dynamo Textual Language (DTL), formerly DesignScript (Aish, 2017). It enables a direct connection with Python 2.7. It has been created to express design intentions and ease the wiring of data and geometry, beyond the out of the box capabilities of the Revit model (Mccrone, 2010). The Dynamo version 2.0.3 is used.

As Dynamo contains a Python scripting interface, the functionality of the software is broad. Python scripts can create loops capable of traversing through data and parsing it to appropriate code snippets, which are appended to the original XML template file from the LCCbyg.

Packages and Prototype dependencies

Packages from external authors are modules increasing the OOTB Dynamo capabilities. The content is open-sourced, and a big thank you goes to all the programmers who spent their effort in creating these excellent packages. Below is a description of how the packages are used explaining the dependencies.

Table 1. Packages and script dependencies

Package and Author	Description
Bakery package (Johnson, 2019)	- It is used to obtain All Family Types of Category – a very useful node that creates a list of family types for each category — used extensively for the LCCbyg data transfer script.
Archilab Net (Sobon, 2019)	FamilyInstance.FamilyType – Takes all family instances within the project and creates a list of family types, which can later be used as keys for further data structuring. Used extensively for the LCCbyg data transfer script. - View.OverrideGraphicsSettings – a custom node used to set color settings for the view filter overrides. It is used for the Calendar filter scripts. - View.SetFilterOverrides - a custom node used to set view filter overrides. It is used for the Calendar filter scripts.
Lunchbox ("LunchBox – PROVING GROUND," 2019)	- RemoveNullValues – a script that removes null values from the data, thereby ensuring the list (column) will always correspond to the correct row. Used extensively for the LCCbyg data transfer script.
Clockwork (Andydandy74, 2019)	- View. Duplicate – duplicate existing views to create new views automatically. It is used for the Calendar filter scripts. - List.JoinbyKey – A node useful for joining two lists of different lengths. Used for the Room cost grand totals script to combine lists of rooms with objects contained within the rooms.
Rhythm (Pierson, 2019)	- Elements.SetParameterByNameorInstance – A node useful for setting parameters to instances of objects. Used for the grand totals room script to populate costing information to every room instance. - Elements.GetParameterByNameorInstance – A node useful for getting parameters from instances of objects. Used for the object cost yearly totals script to populate costing information to every object instance.
Room SVG Exporter, by Adam Bear (Bear, 2019)	- Python Script is enabling the export of Rooms from Views to the .SVG format. Extremely useful for PowerBI applications.

Modularity of Dynamo

Using custom nodes permits scaling of Dynamo scripts, and controlling the granularity of the code. It is possible first to create a draft code transfer, and later replace it by modules of custom nodes or Python scripts.

This way, clarity can be kept, and the detailed source code can be kept away from the primary visual view. Furthermore, there is a possibility to execute subscripts, or custom nodes in a sequence, ensuring there is an order of script execution. The script retrieves, then populates new data, and further retrieves the newly appended data. It is possible to publish self-made packages and custom nodes, in line with the Dynamo coders mentality, that scripts should be openly shared amongst the Dynamo community.

Custom made packages

Creating a custom package requires opening a custom package interface, as well as specifying the inputs and outputs, along with the processing content of the custom script. The custom script is then saved as .dyf file, which can be grouped under appropriate hierarchy within the Dynamo browser interface. The package of custom nodes can then be shared online with the rest of the Dynamo community. For this project, a script package necessary to support the main transfer scripts is created.

Replicating packages to reduce third party reliance is a possibility for applications that are openly compound and designed within Dynamo. For those applications designed in C#, or other external programming languages, they may come as a complete product node, rather than a compound node, and therefore editing and replicating them may require edits on the source code.

The Dynamo Player interface plays scripts without the need to open the Dynamo interface. The player enables data input directly from the interface, given the nodes within the script code are marked as "Input nodes". It is a neat interface for users not familiar with Dynamo, whose job is partly to run scripts that are ready for use.

5.3 Lifecycle Costing tool - LCCByg

LCCbyg is a tool released by the Danish Building Research Institute (SBi), at Aalborg University (AAU) on behalf of the Danish Transport, Construction and Housing Agency (Haugbølle et al., 2017) to allow for an accurate and uniform lifecycle cost calculation for all future public developments in Denmark.

The tools' calculation utility is split into six main categories visible in Figure 5.1. Although all five categories are equally essential to achieve the complete calculation, only cleaning is investigated in detail. The assumption is that if object-oriented quantity takeout can be performed for cleaning, while the other categories relying on objects can be transferred using the same approach.

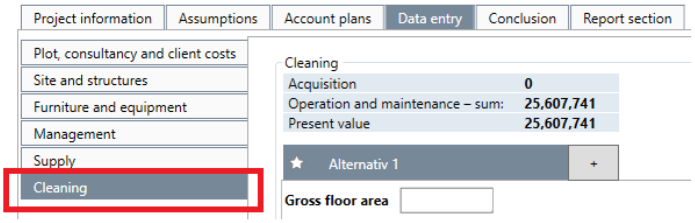


Figure 5.1 The area of interest within the LCCbyg software

To briefly explain the contents of the window panes, the first category accounts for initial development fees, while the site and structures account for factors related to the building envelope. For furniture and equipment, one can expect 2 to 5 changes over the economic lifespan (Maria; Saridaki & Psarra, 2017).The management and supply costs have future value (FV) and price fluctuation adjustments applied.

Cleaning is arguably the most interesting, as it requires maintenance on a daily level. As each inefficiency is multiplied by roughly 200 working days a year, over 50 years, it equates to a factor of 10,000.

LCCbyg and XML

The LCCbyg underlying data schema is parsed to an openly available .xsd schema file for .xml data format, observed from the source materials from the LCCbyg website.

"Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable."

Source: ("XML," 2019)

XML is a standardized and internationally recognized data schema language created by Charles F. Goldfarb, Ed Mosher, and Ray Lorie at IBM in the 1970s (Harold & Means, 2001), enabling a descriptive and hierarchical data transfer. Furthermore, the XML format can be human-readable and uses a set of user-defined tags to tag information it conveys.

"The purpose of an XSD schema is to define the legal building blocks of an XML document."

(“XML Schema Tutorial,” 2019)

The schema is interoperable and fully accessible when openly shared. The main advantage of using an interoperable format is that the data transfer can be mapped to follow a defined schema and be translated from one program to another (Eastman et al., 2018). The attribute values within the XML file can be appended, and new data sets can be merged to an existing document using data parsing modules such as SAX, ElementTree, and DOM (Harold & Means, 2001). Below presented, are lines of code generated by the LCCbyg software in the form of an XML file.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <project generator="LCCbyg 2.2.52" languagecode="en" currency="DKK" locked="false" name="Main Project" projecttypedescription="Project type sui
3   <reportdefinitions>
4     <viewdefinition label="Logo or image" name="metablock" blockname="headerlogo" include="true" comment="Evt. logo" />
5     <viewdefinition label="Title and description" name="metablock" blockname="projectdescription" include="true" comment="Titel på rapport">
6       <body>
7         <div>
8           <h2 id="title">xxx</h2>
9           <h3 id="description">xxx</h3>
10        </div>
11      </body>
12    </viewdefinition>
13    <viewdefinition label="Project" name="metablock" blockname="caseinformation" include="true" comment="Byggesag fra meta" />
14    <viewdefinition label="Client" name="metablock" blockname="builder" include="true" comment="Bygherre" />
15    <viewdefinition label="Consultant 1" name="metablock" blockname="advisor1" include="true" comment="Metadata blokke" />
16    <viewdefinition label="Consultant 2" name="metablock" blockname="advisor2" include="false" comment="Metadata blokke" />
17    <viewdefinition label="Consultant 3" name="metablock" blockname="advisor3" include="false" comment="Metadata blokke" />
18    <viewdefinition label="Alternatives" name="versions" include="true" comment="Alternativer (versioner)">
19      <body>
```

Figure 5.2 XML file generated by LCCbyg 2.2.52

The beginning of the code shows the XML and encoding version, as well as the tool needed for generating the project. Other data is available such as language and currency. *Source: LCCbyg 2.2.52 XML template file*

Line 1 contains the encoding set to *UTF-8*. Line 2 is descriptive of the *project language, currency, accessibility, name, and project description*. Lines 4 and 5 labels the project metadata, such as project *logo, title*, and *description*. View definition labels define views containing information about the project, *the client, the consultants*, and the project alternatives, which are visible between lines 13 and 18. This information is unaffected by the proposed data transfer and can be filled in either manually by the auditor, or through another data automation plugin or module.

```
5443    <accountplan vid="standardcleaning" label="Cleaning" type="management" indexdevelopmentname="inflationgeneral"
5444    npvdevelopmentname="discountrate" lockbasevalues="true" hideamount="false" includeinreport="true">
5445      <structuredefinition calculationmethod="standardizedcleaning" resultgroups="cleaning" indexdevelopments="inflationgeneral">
5446        <field name="indexdevelopment" level="row" group="basevalues" label="Price development" />
5447        <field name="units" level="row" group="basevalues" label="Unit" />
5448        <field name="unitprice" level="row" group="basevalues" label="Unit cost (DKK)" />
5449        <field name="frequency" level="row" group="basevalues" label="Frequency" />
5450        <field name="units" level="row" group="inputvalues" label="Unit" />
5451        <field name="amount" level="row" group="inputvalues" label="Amount" />
5452        <field name="unitprice" level="row" group="inputvalues" label="Unit cost (DKK)" />
5453        <field name="comment" level="row" group="inputvalues" label="Comment" />
5454      </structuredefinition>
5455      <units defaultunit="m2">
5456        <unit name="m2" />
5457        <unit name="rm" />
5458        <unit name="units" />
5459        <unit name="stacked m3" />
5460      </units>
```

Figure 5.3 LCC XML generated Information about cleaning account plan

Source: LCCbyg 2.2.52 XML template file

The data about project cleaning is represented in line 5443. The line count can vary depending on the data input from other categories, i.e., *construction* or *demolition*. Therefore any code relying on character or line count will not work here.

The account plan from line 5443 is created to contain information for standard cleaning. It has a general inflation rate and is categorized under *management* costs. Sub children of the *cleaning account plan* include *price development*, “initial” *units, frequency* and *unit price*, and after that, *input values* for instantiated units, *unit price*, *frequency*, and *count* (amount). The available units, along with the default unit, can be seen in lines 5455-5460.

The account plan enables the data to be hierarchically intertwined with the rest of the software. Its children indented further, will store the input values once those are transferred, and it is where the Python script will have to input new information, directly based on the Revit model and its parameter values.

```
5654      <maingroup name="Buildings, internal" sharedvaluegroups="indoor">
5655        <subgroup templatename="Rum" istemplate="true" name="Space">
5656          <row name="empty" include="false" indexdevelopmentname="inflationgeneral" units="m2">
5657            <basevalues>
5658              <values vid="template" />
5659            </basevalues>
5660            <inputvalues>
5661              <values vid="1" />
5662            </inputvalues>
5663          </row>
5664        </subgroup>
5665      </maingroup>
5666    </accountplan>
5667  </costdatabreakdown>
5668</project>
```

Figure 5.4 Empty placeholders for XML data needed for this project

Above, the predefined element tree is visible for the maingroup = "Buildings, internal." *Source: LCCbyg 2.2.52 XML template file*

An empty space called "*Rum*" contains no base and no input values besides a version ID (*vid*) set to a template for *base values*, and set to "1" for *input values*. The data is missing regarding *units*, *unit price*, *frequency*, and *amount*. This is the data needed to be filled in from Revit using a coding technique that will convert the data appropriately.

Before this can take place, the Revit model must be populated with unit price data, as well as cleaning frequency, units, and quantities. To do so, besides linking the database with the Revit parameters, the attribute values within must be assessed. Costing methods can be taken from external databases, as explained earlier. The frequency of cleaning is case-specific, and no specific calculation approach is considered in this report.

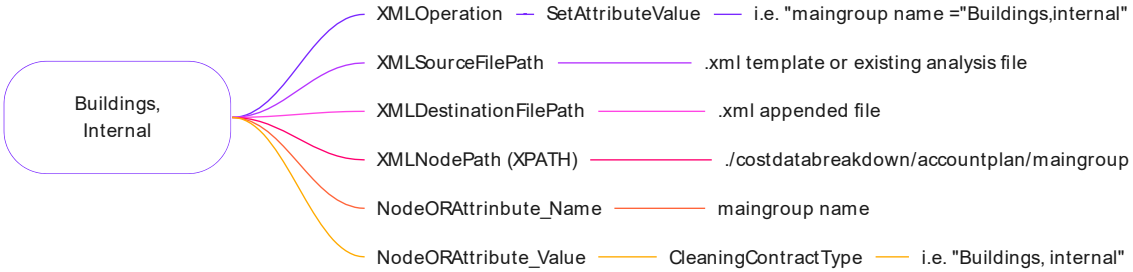


Figure 5.5 The structure of the subgroup called Buildings Internal

To create a Revit counterpart of the frequency, cost, count, and unit type parameters, Revit counterpart parameters were added in the form of shared parameters. The data mapping diagram can be observed in the appendix 14.3. Shared parameters can be shared across models and be stored separately.

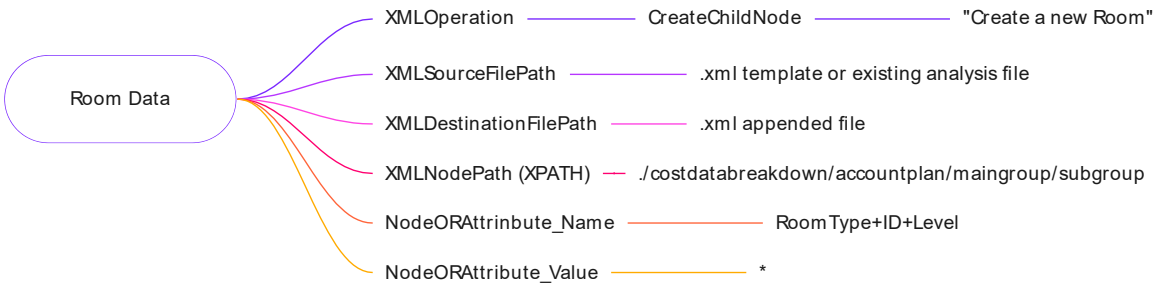


Figure 5.6 Room data level of granularity

Accessing the XML structure requires reaching a certain sub-level of the element tree. In the above illustration, XPATH is used to access the *Buildings Internal* subcategory – a child of the *maintenance* category. After that, rooms are the children of the *maingroup* (Buildings Internal subcategory), holding room name information.

A level below, "*Room Data*" is a container of room instances within the building. For simplicity zones are omitted in the schema; however, it is possible to use zones, instead of rooms, by reorganizing the structure.It is also possible to place all objects directly under a single room – which is the approach used by the prototype to simplify the data transfer.

Object structure and object attributes

There are at least three possible approaches to structuring the data:

1. Grouping objects by combined aerial categories: i.e., *stairs, floor surfaces, doors, etc. using areas (m2)*
2. Grouping object by combined unit type counts: i.e., *stairs Type A, single leaf wooden door, etc. using units (pieces)*
3. Grouping objects by Room IDs, *i.e., Room 123456/stairs Type A, Room12345/wooden floor, etc. using both units and areas.*

Alternatively, a combination of metrics can be used to simplify the data transfer while maintaining the level of detail. This approach uses a mix of the three approaches above.

1. The objects' categories are used for the main classification (categories).
2. The object types are forming a child of the categories superstructure (types) through a division of types depending on the metric, i.e., units, running meters, areas, and volumes,
3. The floor surfaces are individually assessed using Room instances, sorted by Revit element IDs (Room IDs).

This approach permits testing of all three approaches with a relatively lesser amount of scripts needed. All three approaches can be combined and coded to form a basis for future updates, once a preferred approach is chosen. As for the initial prototype, the room container will hold areas of floor surfaces, doors, curtain walls, and windows, while furniture, fittings, and plumbing fixtures will be grouped per object type. A diagram depicting units of measure per object type can be found in Appendix 14.3 - Figure 14.13 A diagram depicting the different units of measure for different cost objects.

The proposed approach quantifies objects by object type. This approach has a drawback that the objects within rooms are not independently costed. Instead, they are costed together, as a sum of all objects of a specified category and type – say, a laminated plywood desk 1500x750mm will be counted for all instances within the building, rather than a single room. The location of a new object in the XML schema can be seen in Figure 5.7 A diagram illustrating adding a new object to the LCCbyg Schema.

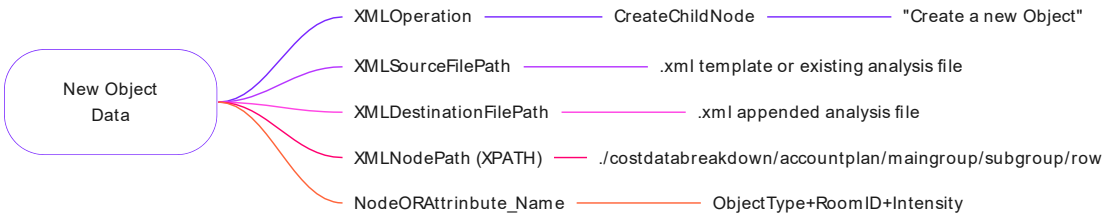


Figure 5.7 A diagram illustrating adding a new object to the LCCbyg Schema

The advantage is that the data structure will be much simpler. It is also unlikely there should be discrepancies between rooms with similar object types. It is possible to account for anomalies by creating a new object type.

Adding Attributes to objects

The database must store attributes. The placeholders must follow a schema that will be uniform, from documentation to the database, to the model, the scripts, all the way to the LCCbyg costing tool. The structure is taken from the LCCbyg tool, given this is the part that cannot be altered. As the tool requires costing, frequency, and quantity inputs, grouped by cleaning intensity, the below diagram in Figure 5.8 depicts the parameter structure.

Figure 5.8 Parameter placeholders, relationships, and data types

The light, average, demanding intensity parameter can be tailored to the specific use case of the building, and not necessarily follow the guidelines of the TEC 1.5 DGNB standard.

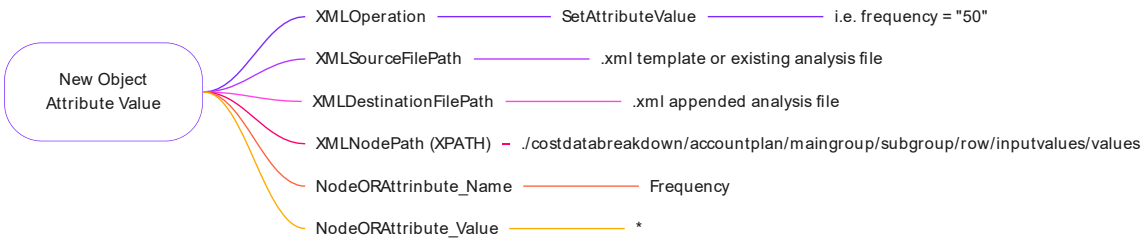


Figure 5.9 A diagram is depicting adding a new attribute to an object

Attributes such as frequency or cost are placed as values under objects in the "inputvalues" hierarchy level. Objects are called rows in the XML schema, where subgroups equate to room instances in the Revit model.

Specifying the intensity levels lets the auditor to explicitly state what is meant by light cleaning, as opposed to average, for a particular object. The list of object types and intensity specifications should be accounted for in the database, or linked to an external document file.

6 Case Studies

Two case studies are investigated for this project. The initial case study comprises an office building created to develop the prototype for the data transfer. as well as to establish modeling procedures required to create a BIM model for a facility that is already in operation. The second case study is an office building during its design development phase, supplied by C.F Møller. This case study is used as a testing model to figure any transfer errors or modeling techniques for which the transfer scripts cannot account for.

To enable object-oriented lifecycle costing, a BIM model is a prerequisite. This is a lesser problem for new project procurements, as BIM has become very popular, and most of the new public projects do have BIM models available for this calculation as required by the Danish law (Smith, 2014). It may not be the same for existing buildings built decades or centuries ago, where BIM models are not so readily available. Even when the BIM model is available, the model might not be modeled accurately, or may not contain adequate objects needed for the data transfer.

The case study aims to investigate the minimum data required to satisfy the LCC data transfer to LCCbyg. To do that, an object library is developed, along with the parameters, placeholders, and data transfer scripts. The data is developed directly within Autodesk Revit, as this BIM environment is familiar to the author. No external database is used to support the model setup, and the parameter values are keyed in manually using the Revit schedules. The schedules can be found in appendix 14.2. The second case study uses dRofus to transfer parameters between the database and the Revit model.

6.1 Test Revit Model

The model has been created in Revit, along with families needed to represent the furniture, plumbing fixtures, and lighting fixtures. The geometry level of detail was generic LOD200 (DiKon, 2017), as the only purpose of object geometry was to retrieve item location relative to its placement in the building and having an object placeholder for parameter storage.

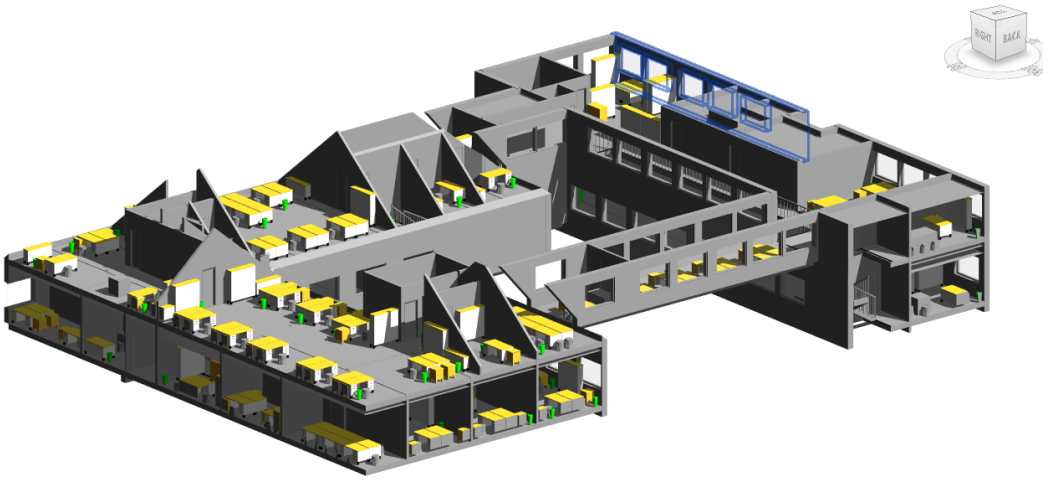


Figure 6.1 Case Study Revit model in 3D - Screenshot

Note that the roof and windows have been hidden, and sections were cut on the two front-facing walls. The illustration presents 3D objects taken into calculations.

Families have been created to support object-based library creation for information exchange and to support visual and spatial planning. The list of family types can be seen in Table 2. The families must have a 3D geometry for bounding boxes to exist, allowing for intersections with room bounding boxes.

1	Carpet	☑	RGB 191-235	<Solid fill>
2	Linoleum	☑	RGB 128-128	<Solid fill>
3	Perforated Metal	☑	RGB 192-192	Steel
4	Polished Concre	☑	RGB 218-219	<Solid fill>
5	Tile antiskid	☑	RGB 147-147	<Solid fill>
6	Tiles	☑	RGB 064-192	<Solid fill>
7	Wood	☑	RGB 221-198	<Solid fill>

Figure 6.2 Schema color legend representation for Revit rooms (Screenshot)

Figure 6.2 and Figure 6.3 present a color schema legend depicting floor surfaces and a sample floor plan of the model. Full plans can be found in chapter 14.1 - Appendix A – Floor plans. The drawing represents an existing office building of roughly 1700m2, built in the 1980s.

Figure 6.3 Case study office building – Ground-floor plan (Screenshot)

Orange-colored objects are made of laminated wood.

The model setup took just a few hours, proving its early design phase applicability. It is hard to quantify the effort needed to create models of existing buildings, yet with the advancement of photogrammetry and machine learning, it may soon be automated and down to a surveyor capturing the structure on a camera (Barazzetti, Banfi, Brumana, & Previtali, 2015). Until then, a standardized object database and workflow will reduce the model constructing process.

Some of the families were imported from other Danish-made Revit models, to see how such items will behave and to check for any bugs related to premade families, with pre-existing parameters. One bug came out for a family with no geometry, which only contained 2D annotations. As it could not intersect with a room, it had given a null value to the group by key code, thereby creating an error.

The test model did not contain all of the elements due to time constraints; however, scripts were made, capable of transferring various units of measure, which work with multiple categories. The data transfer is further explained in chapter 8 - Data transfer Prototyping.

Table 2. Components and component materials included in the Case study model

Furniture Families	**Plumbing Fixtures**	**Casework**
Desks, Tables, Chairs, Bins	Toilets, Washbasins, Hand dryers, Towel dispensers, Soap dispensers, Sanitary bag mounts, Mirrors	-
Doors	**Windows**	**Curtain Walls**
Single & double leaf	Openable, Fixed	Openable, Fixed
Stair runs	**Lighting Fixtures**	**Railings/Skirtings**
Steel staircase runs	-	Stair railings, Balusters
Ceilings	**Object Materials**	**Floor Materials**
-	Smooth laminated wood, Gripped rubber plastic, Metal, Porcelain/Ceramic, Glass, Paint, Steel, Glass	Mosaic Wood, Tile antiskid, Tile smooth, Carpet, Perforated Steel, Linoleum, Entry Mats

The model was created based on an existing building, personally investigated, and measured over a period of two years, employed as a janitor. The case study has a practical grounding concerning cleaning and anomalies derived from this space. One oddity was the occupancy rate of individual bathrooms in the building. The two bathrooms on the first floor to the left (West side of the building) were used by 44 people (desk count), and 23 people only used the four toilets to the right. The high occupancy rates made the two toilets to the left extremely time-consuming, whereby the four bathrooms to the right relatively quick to clean.

Such analysis could be derived from a BIM model given it has shortest path algorithms calculating occupancy rates. A Dynamo script is capable of calculating the shortest paths is made available by Dieter Vermuelen (2017). It is possible to adapt the script to uses other than a fire escape.

Although bins are not objects included by the architects, they are placed afterward by the cleaning companies can often be cumbersome on the cleaning personnel, as explained in this model, 93 bins were placed in the building, which made emptying a 30-45 minute task, each day. If the containers were instead located at strategic locations, this effort could be significantly reduced.

6.2 Implementation case study

The second case study is a model provided by CF Møller is a 700m2 office building. It can be observed from the floor plans, that surface material optimization and furniture layout, as well as craftsmanship, will likely influence the LCC calculations. The plans are not made available due to copyrights. The following section describes the testing of prototypes on the case study model. Some readers may prefer to read chapters 7 - User Environment and chapter 8 - Data transfer Prototyping, prior to reading the case study implementation.

The dRofus implementation for costing and using a standardized template is the desired workflow; however, the initial test was performed without the dRofus database, as it required permissions to alter its setup. The model was to be populated with sample data that may not provide 100% accuracy due to a lack of reliable object costing information. The primary purpose was to highlight all deficiencies of data transfer and potential risks when implementing the workflow on alternative buildings.

While checking the model, it became apparent that additional scripts were to be required to facilitate curtain panel windows and the transfer for lighting fixtures. The scripts have proven to support both curtain wall windows and doors, as well as lighting fixtures, after minor modifications. The scripts are explained in chapter 8.

A shared parameter file is imported to commence the transfer. The simplest way to do this is to insert a view from another project. In this case, – the views consist of schedules necessary for data transfer. The

schedules automatically populate parameters needed for each component, without the need of running any Dynamo scripts.

Floor Finish parameter values were missing, and the floor finishes were annotated as surface materials on floor build-ups. This may require an additional script that will match the material type with the room type. Unfortunately, there were double floor buildups made in the model – one generic, marking the space build-up of the structural floor, and the other finish floors. The attempt to transfer materials to room data has failed due to the duplications of floors intersecting with the rooms; however, an alternative approach of if-then based logic gateways could have been used instead. The room floor finish parameters were instead filled in manually due to a small complexity of the model, taking approximately 10 minutes.

Some objects are created as generic models, where instead, they should belong to the furniture type. This is a small issue, as it only requires the objects to be classified as furniture families. A bigger problem was presented, when stairs were modeled as a generic family. The stairs are a built-in Revit family and must be modeled appropriately, or otherwise, the family does not contain nested components such as baluster, railings, or stair runs, and therefore, will not show in the transfer. Lastly, some objects were created as 2D plan representations instead of 3D geometry; therefore, those objects required adding a 3D component, or otherwise, were not considered during the data transfer.

The auditor needs to pick up those errors and fix them before the data transfer, or otherwise, the LCC estimations will lack in accuracy. According to the BIM manager, it is challenging to convince seasoned employees of big organizations to stick with modeling conventions, and therefore a proposed approach is to; one, audit the model, two, switch the families to temporary ones, keeping in mind that the detailed geometry is not required for the transfer, three, make the LCC transfer, and four, return to the original model objects afterward. This is not the best practice, but it does not disturb the workflow and keeps the job of change advocates much easier. A long-term solution will require the correct object database, accessible by all modelers.

6.3 Implementation of database setup

To set up and link a dRofus database, specific actions are needed to be carried out. Firstly, a Dynamic GUI editor must be appended with new LCC window pane, and parameter placeholders, and secondly, the parameter values must be filled in for each object in the database. Lastly, a link must be created between the database and the Revit model.

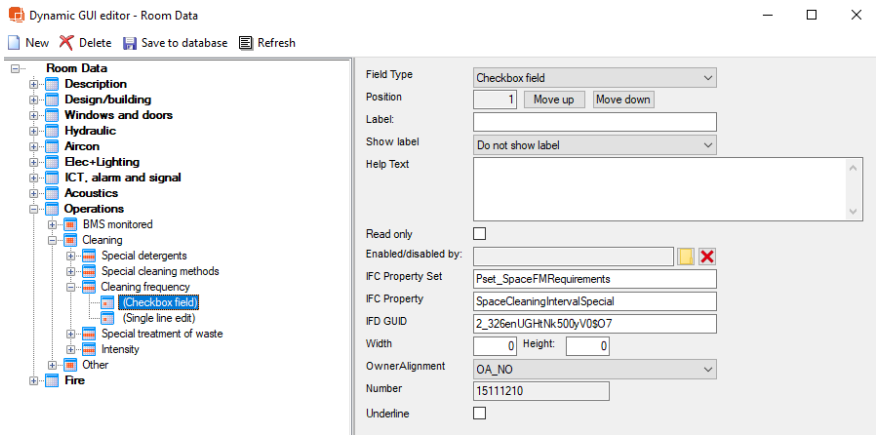


Figure 6.4 Placing LCC placeholders at a Room Level – dRofus GUI editor

Source: dRofus (screenshot)

To alter the template administration rights are required. To change the administration rights, the head of BIM must be informed. In the settings, it is possible to set up the necessary parameters for the LCC data transfer both at room and object level (see Figure 6.4 and Figure 6.5 for reference).

Setting up the parameter placeholders at room and item level is required to be done only once. The harder part is updating parameter values when the object or floor cleaning prices change.

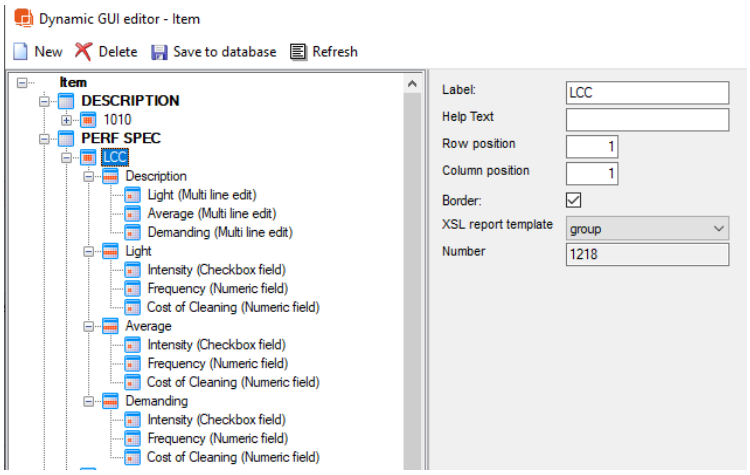


Figure 6.5 Placing LCC placeholders at an object-level - dRofus GUI editor

Source: dRofus (screenshot)

It is also possible to change the instance parameters, which is unique for dRofus, as most other databases only work on type parameters. Seeing all instances of type is extremely useful when making a quantity takeout for the model. Furthermore, some instances may be highlighted with instance parameters such as areas and quantities. It also shows the location within the model, such as level or zone, the base, and top offset if applicable, and other instance parameters.

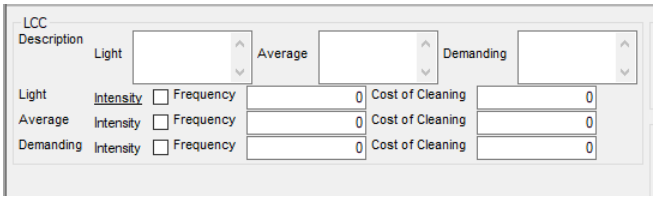


Figure 6.6 Adding parameter boxes at an object level

In dRofus referred to as "Item-level," instead of "Object-level."

Source: dRofus (screenshot)

The dRofus database has predefined filters included in the template. Those are classifications based on different countries, measurements, and specifications of work. It is also possible to remove parts of the template that are not being used on a specific project to make it more user-friendly.

For this project, the building owner requested a CCS classification. It is possible to use merged parameters to merge the type name and the type code to name objects fully. The combined parameter is then pushed into Revit, naming family types accordingly.

BIM7AA can be used as a structure classification that can be translated to other classifications. The dRofus can automatically transfer it to other classification systems. Specifications are used as headlines for the building material tender. This plugs into the building component journal. Building component descriptions code is used for specifications connection with the building journal.

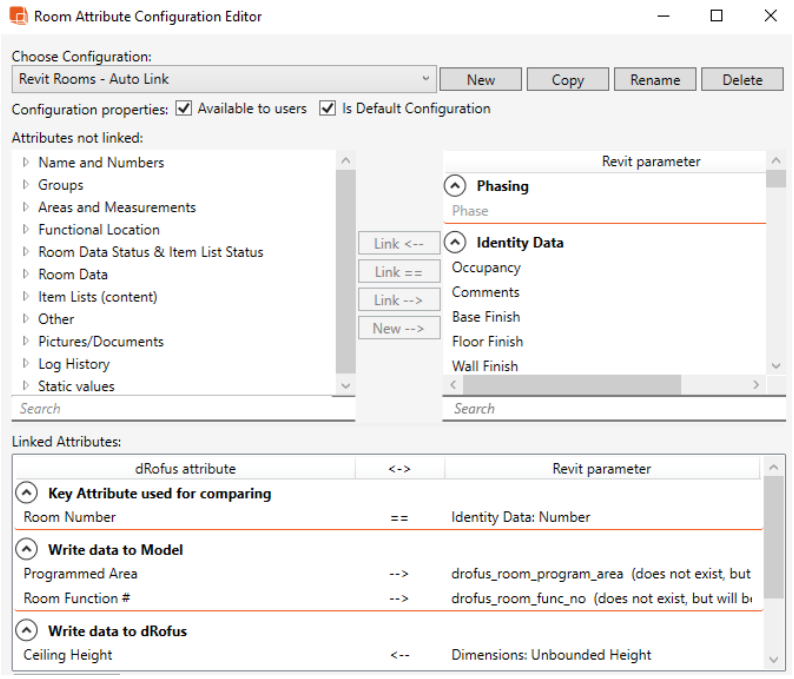


Figure 6.7 dRofus: Bi-directional link interface between a database and the model

Source: dRofus (screenshot)

If the family is both in the model and in dRofus, the two can be linked. A human-readable classification parameter links the two models. Once the correct element classification is found, the component, along with the geometry, is uploaded to dRofus, automatically adding the family geometry to the dRofus documents. Once the object is in dRofus, it can be renamed to an appropriate name.

The Revit-dRofus plugin permits bi-directional information exchange. *Attribute Configuration Settings* is a pane where the parameters are mapped between the dRofus and Revit. The database holds all presets when the project is started. It is possible to edit the mapping in the *item Attribute Configurator Editor*, as the list shows the parameters which can be linked from dRofus to Revit, all in one interface. There is a window of dRofus parameters and a window for Revit parameters. Below are two mapping windows, one from dRofus to Revit and the other from Revit to dRofus.

"Some parameters are defined in Revit, such as the wall face, i.e., Exterior, which goes from Revit to dRofus, but most of the parameters go the other way around, from the database to Revit. Inside the dRofus, we can change the item naming. If we have a building component in the model that we want to load into the dRofus, we simply select the building component on the selection window pane, and the mapping arrows are used to create it in dRofus or Revit, depending on the arrow direction. If a component is in another project, the object geometry and information can be transferred directly from the other dRofus project. The files of the family relate to the dRofus item by the Documents tab. This way, every modeler is using precisely the same family. This also ensures there are the same families and types across two models of the same project."

"*Normally, we would have to remember to do this in the same way, but with dRofus, it is possible to synchronize the two files. Often we have to change the descriptions of say the wall buildup, by specifications, and it will synchronize to all the models, not just one of them."* -BIM Manager

The database was not fully integrated with the model due to a lack of adequate administration rights. It remains a task to integrate the processes to a greater extent. The full potential will only be utilized; once appropriate costing mechanisms will be in place to correctly estimate the cost of object cleaning. Meanwhile, the area based calculations transfer will have to suffice.

7 User Environment

A summary of the user environment and the resulting workflow design will be presented following the User Environment Design Contextual Methodology. The software sequence model below best describes an overview process necessary to store, model, map, transfer, and view data concerning lifecycle costing.

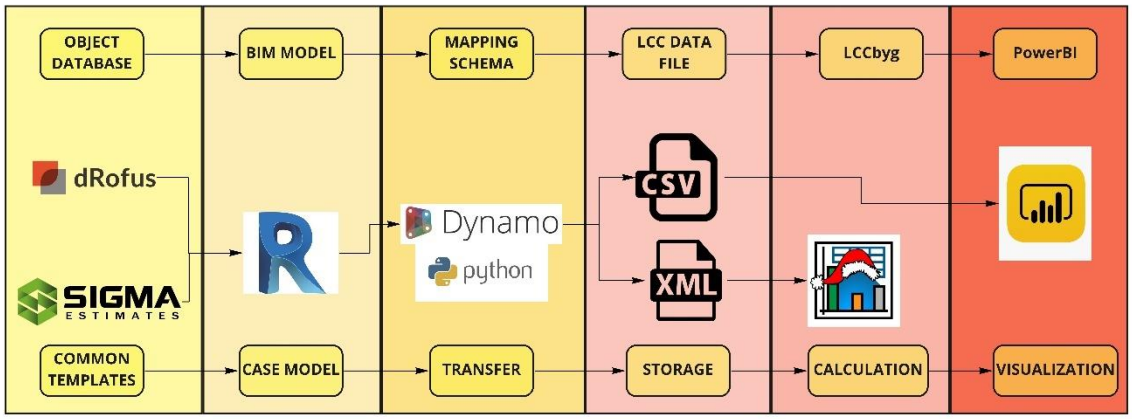


Figure 7.1 Software sequence model process representation

The object database contains costing information at an object level, which is linked with the BIM model using a classification code. Alternatively, entire objects (Revit families) can be stored in a database containing necessary LCC information within the template file. All it requires is to connect a database with a BIM model.

The Revit template file either already holds parameter placeholders for cleaning operations, or if those are missing, a view schedule can be inserted from a parallel project. This enables the automatic parameter placeholder transfer, ensuring correct naming, data types, and grouping. Furthermore, this approach creates schedules named in an orderly fashion, creating a standardized appearance across all projects.

From there on, the Dynamo data transfer script is run, and the Python code sends the data to the LCCbyg software. The chosen design is then saved as a final option within Revit, and the reports from LCCbyg can be generated.

If the reports generated by LCCbyg are hard to interpret, or the difference between options is difficult to observe, the information can be carried over to PowerBI. The results are then made presentable and easily digestible for clients and other stakeholders.

PowerBI can then be used to visualize optimization efforts and resulting savings. The design choices can also generate updates to the templates from the dRofus database if new optimized discoveries had proven more cost-effective than previous assumptions.

The resulting workflow can inform best practices, which can then be applied to a variety of projects. Flexibility can be achieved by enabling updating of the results once new discoveries regarding materials, costs, or cleaning techniques are brought to light.

Figure 7.2 Proposed workflow BPMN diagram shows the process using swim lanes – following the conventions of a Business Process Modelling and Notation (BPMN). The description can be found in Appendix K – BPMN diagram description.

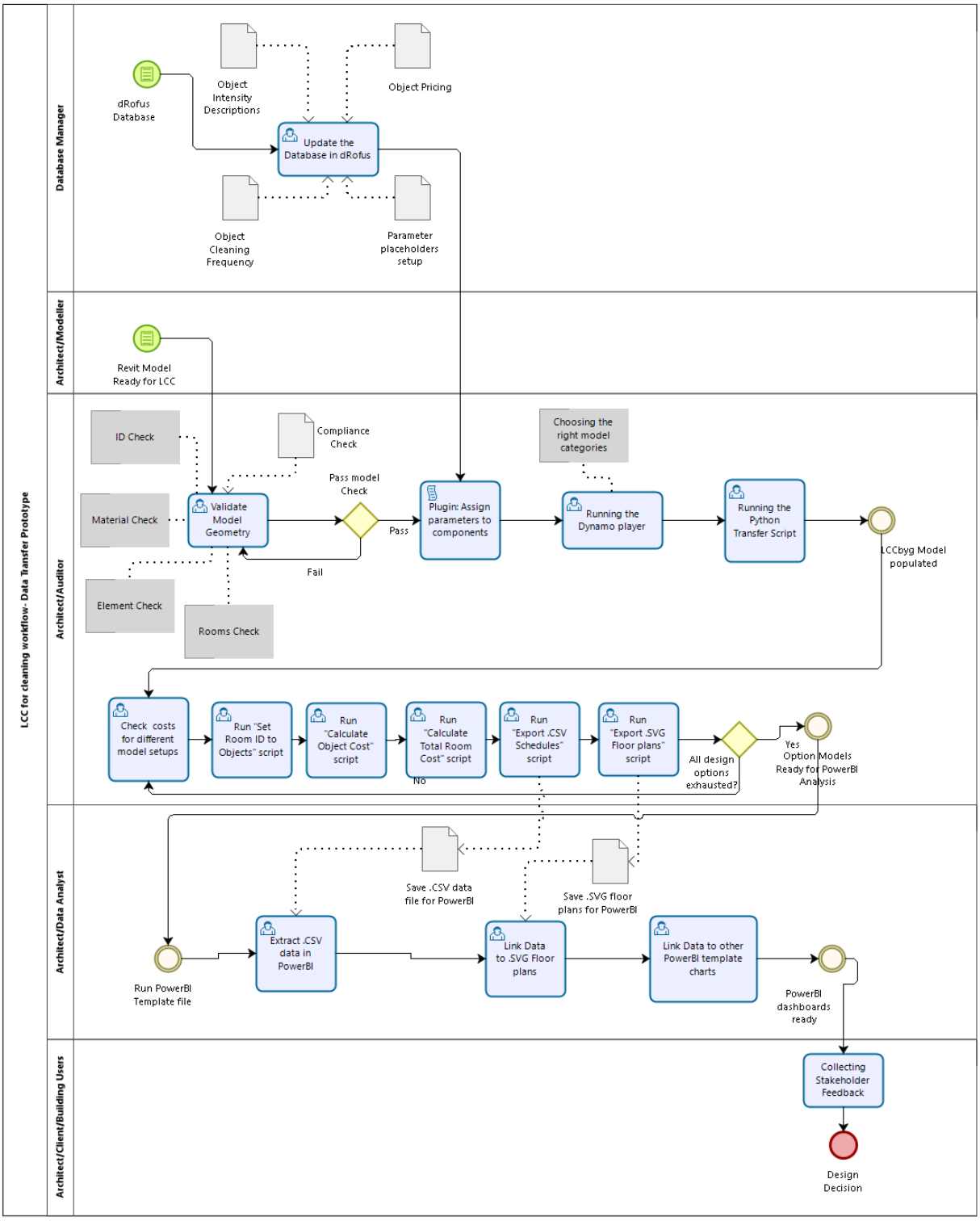


Figure 7.2 Proposed workflow BPMN diagram

See appendix 14.11 for node descriptions.

Updating the database is often the job of a different person from the designer, and there exist various milestones for new design trends, cleaning techniques, and products. The diagram above depicts what roles are expected of the team to succeed when performing the transfer.

The job of the auditor is to validate the model, prior to transfer. This gateway event ensures the model is compliant with the analysis model view definition. Once the data had content had been approved, the updated parameters from the database can be assigned.

A set of scripts are then turned on following the sequence illustrated in Figure 7.2. It is likely that the PowerBI analyst will merge various workflows, based on data and data view GUI templates, including combining data with the .SVG floor plan infographics.

Lastly, the clients and the architects, given all the required data, and thanks to their project overview, will be able to make design decisions and choose the right solution.

For this to work smoothly with the auditor and the database manager, the workflow must be easily replicable, ensuring correct deployment, adoption, and standardization. Below, Figure 7.4 presents the process overview sequence model, followed by more detailed explanations of the process, found in the following subchapters.

To accommodate a model which will be of any complexity, and of a customized workflow, specific to the company needs and standards, flexible mapping procedures will be required. Furthermore, the end solution user must be able to understand the prerequisites of the workflow, as well as be able to customize the setup, and in some cases, create a custom mapping system. The purpose of the diagrams in this chapter is to show the workflow and functions, as well as to highlight potential risks.

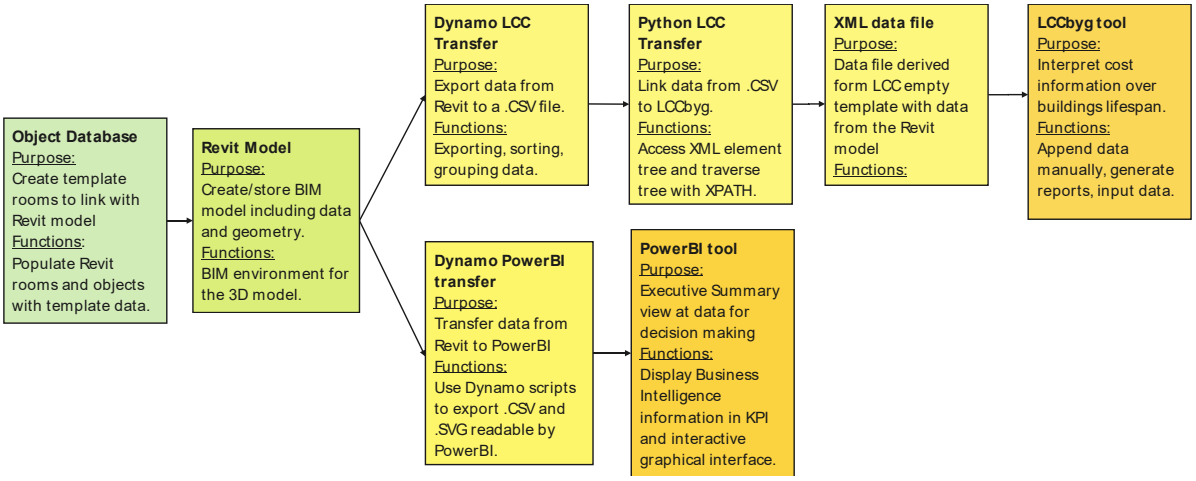


Figure 7.3 Overview diagram depicting LCC for cleaning process flow

Full illustration on the next page.

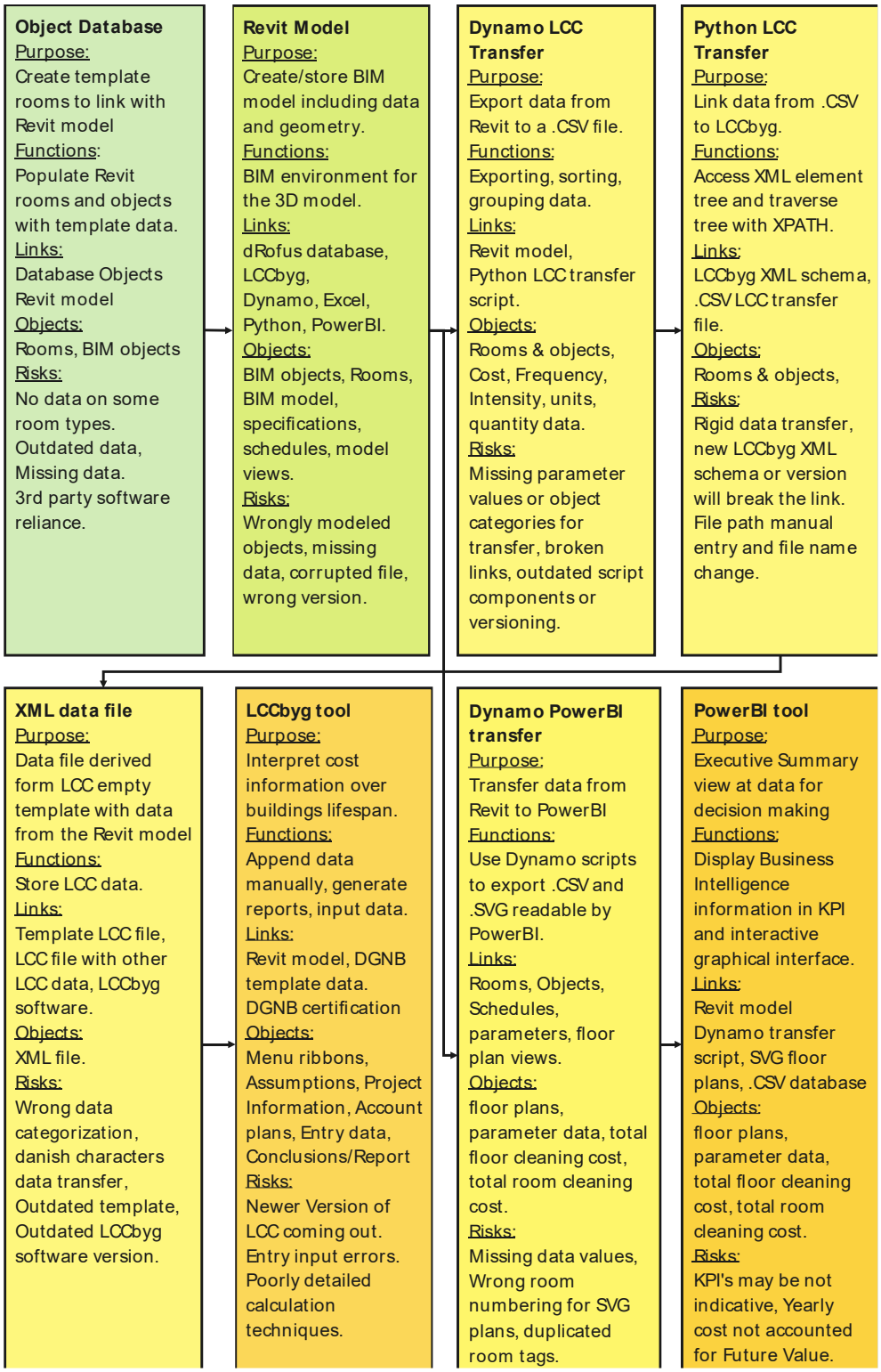


Figure 7.4 User Environment(UED) – Overview of functions

7.1 Using the dRofus database

The proposed workflow suitable for the use of C.F Møller architects, and mainly their auditor is based on using existing work processes as much as possible. In the dRofus software, the architects store standardized information suitable for use in future projects. Room templates for standard offices,

bathrooms, hospital rooms, and other room types alike are stored, enabling knowledge gathering from previous projects. The eight steps below investigate adding the required parameter information from the database to the Revit model.

Figure 7.5 dRofus EUD sequence: Adding data to the database diagram

It is also possible to create custom room instances – those slightly different than the template. Within those templates, information regarding the cost and frequency of cleaning for both rooms and objects contained within can be stored. A link is established, creating a bi-directional connection between the Revit model and the dRofus database.

Once the Revit model is linked, and data is populated, discrepancies between desired object/room count are checked against, and missing objects are populated. dRofus displays object counts within the Revit model relative to the desired object count from the template. The only nonautomatic task left is to place the object in the desired location. Ideally, to populate the data automatically, the data should be directly linked from the database, as explained in chapter 5.1.

7.2 Using the Revit model

The eight steps below will ensure that the user will check if all needed categories are matching the script input expectations. Generic models are models that are not grouped into any category, and category is a primary data filter used to access information hosted in the children of the category data tree. Therefore, all generic objects must be assigned an object type, albeit furniture, plumbing fixtures, a door, or be it stair runs. Furthermore, 2D family annotations require a 3D component to account for object intersection with the room bounding box, which is needed for the cost calculations script.

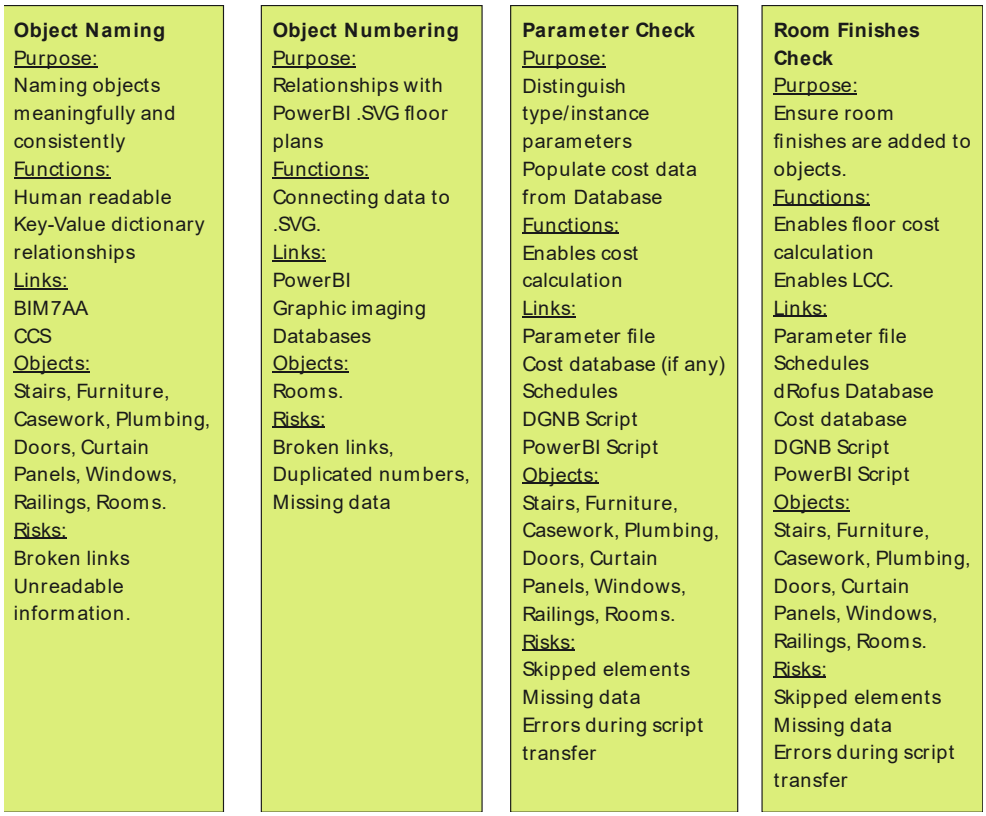


Figure 7.6 Revit model UED sequence: Consistency Check procedures (Part 1/2)

Unplaced rooms must be deleted, and grouped objects must be exploded, as group object data cannot be edited otherwise, in a simple way. Object naming must adhere to type coding so that the subset of LCC cleaning can be connected with other LCC calculations and databases, which will rely on the code type as the type ID, a mapping value which will be shared between platforms, thus enabling seamless interoperability. Object numbering for Rooms permits a direct link with Simple Vector Graphics (SVG) hosted in the PowerBI application. This will, in turn, permit interaction with the .SVG graphics, as the .SVG objects will be linked to the Revit room numbers.

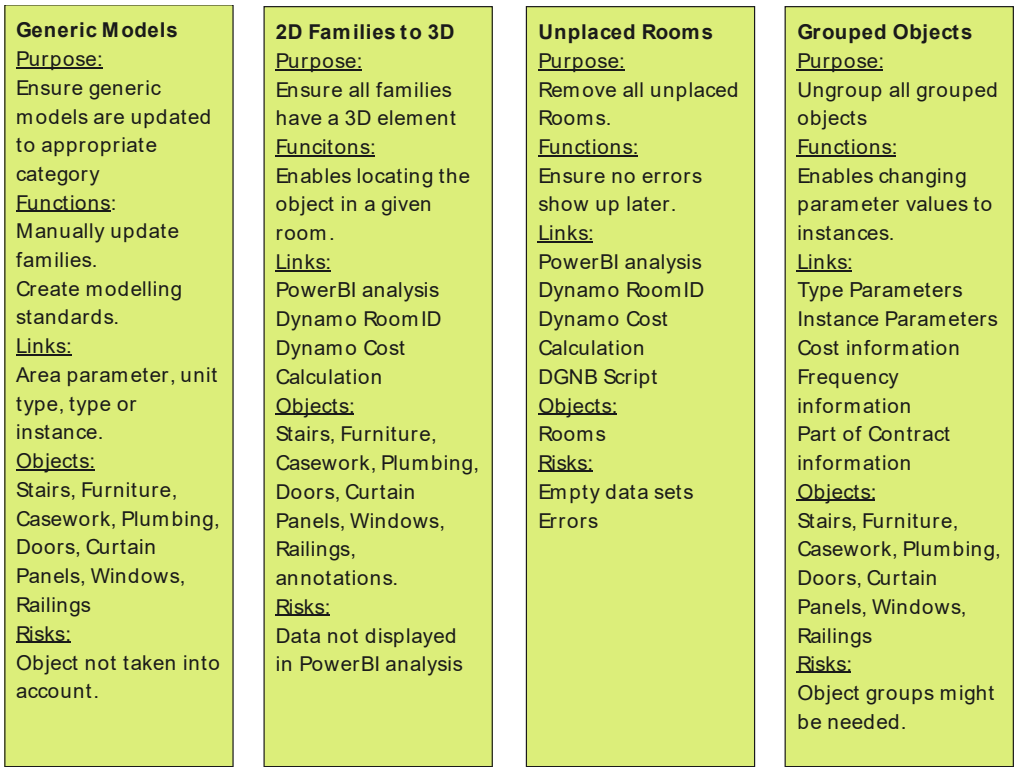


Figure 7.7 Revit model UED sequence: Consistency Check procedures (Part 2/2)

Lastly, the parameter values must be checked to ensure that the data is filled in as required. This concerns the cost, frequency, intensity (instance or type) parameters such as light intensity, average intensity, and demanding intensity, as well as room floor finishes. Room floor finishes must be populated with material finishes, as it is often the case, that the floor material is hosted within the floor composition model. From there, it must be referenced to the room the floor belongs to. Rooms cannot contain more than one material. A separate room container must be made if the floor contains two materials, or a cost parameter with a value which will account for the proportional mix of two-floor materials must be considered.

7.3 Using LCCbyg

Here the process of using LCCbyg to evaluate how each of the approaches assumed costs and how does the level of detail influence the price fluctuation and variance. Here the information retrieved from the lifecycle costs can be further used during the Operations phase, and the description of how such processes can grant further DGNB certification points be investigated. Currently, to retrieve alternative results, a user must make a change in the Revit model, then run the dynamo script, followed by the Python script. Then the changes need to be saved to a new file, and the file can be compared. Although not an entirely automated process, each version change takes less than a minute.

The downside is that the versions cannot be compared within the software, as in the object level approach, altering existing template elements would require accessing each sub child. Hopefully, there is an alternative approach to the data transfer, which can account for such changes. An alternative is to publish to parallel transfers and disable one of them manually inside the LCCbyg software. This way, one solution will have “template x” greyed out, while solution two will have “template y” greyed out. This is a workaround solution to comparing versions within the software. However, since the Lifecycle costs are displayed as a lump sum, PowerBI is a more suitable tool to analyze cost savings and display optimization potential.

Figure 7.8 LCCbyg Software Environment – window panes clarification

The LCCbyg 2.2.52 software environment is best described in the software documentation (Haugbølle et al., 2017). However, for a quick overview and relation to the project, the above figure depicts the five primary tabs. The main project information tab holds information about the client, the building itself, the company carrying out the LCC, information about consultants, etc. The data here is filled in manually unless automation is developed internally by the auditors' office.

The purpose of the assumptions tab is to calculate the present value of future costs. It consists of features such as price development, discount rate, calculation principles, calculations periods, and calculation methods for media, maintenance, replacement, operations, and demolition.

Account plans and data entry fields account for template and data entry, respectively. Those are independent for each category, i.e., plot, consultancy, structure, furniture, supply, and cleaning costs. This report is only concerned with the latter cleaning costs, and the purpose is to showcase cleaning placement located in the greater context of the LCCbyg. Automatically filling data using the approach proposed here will populate both the account plan template and the data entry field. This is because the parameter “isTemplate” is set to “true” when parsing data from the .CSV file to the LCCbyg XML schema.

To fully understand the data gathered, as well as optimize the cost and foresee where the cost is assigned to, it is proposed to use the PowerBI platform. This platform will enable a detailed outlook on the frequency and cost of cleaning, individually for each room. The data will also form a transfer backbone so that it can be later utilized as a subset for Facility Management (FM) programs such as MdocFM or Dalux FM.

Correct sub-scripts mapping	**Data Flexibility**	**Versioning**	**LCCbyg Version Update**	**Using Python 3.7**
Purpose: Ensure object categories use appropriate data transfer mechanism. Features: Object Type Area-based script, Room Instance Area-based script, Object Type Unit-based script, Stairs Area script (runs only) Object Running-meter script Links: Object Categories Export to .CSV file Objects: Stairs, Furniture, Casework, Plumbing, Doors, Curtain Panels, Windows, Railings, Rooms. Risks: Incorrect mapping. User error. Script error. Modelling error. Too much model data.	Purpose: Ensure minor errors are supported and don't break the program. Features: Change Danish Characters to standard English set. Remove Empty datasets. Remove null values. Preserve List indices. Editable Scripts. Links: Sub-scripts Objects: Dynamo nodes. Data objects. Version ID Change Dictionary Risks: Inexperienced users may break the script without knowing. Hard to reverse errors if not saved. Dynamo crashing.	Purpose: Enabling comparing versions inside LCCbyg Features: Version control from within the Dynamo Script Links: Model elements, .CSV export Objects: Version ID, Version template ID, IsTemplate (yes/no) Risks: LCCbyg version update. LCCbyg crashing.	Purpose: Improve the program Features: Extended Features. Links: DGNB Script Revit Model Cost data Frequency data Objects: LCCbyg interface LCCbyg XML LCCbyg documentation Risks: Needs script update to facilitate the data transfer	Purpose: Transfer .CSV data to LCCbyg XML file. Features: Hard-coded data transfer to a single data tree location : internal cleaning within Operations and Maintenance Category. Links: .CSV file generated by the DGNB script LCCbyg software LCCbyg XML file LCCbyg XML schema structure Objects: Transfer script written in Python 3.7, accessed through Spyder or pip execute. Risks: Hard-coded (inflexible), Subject to LCCbyg updates, Insufficiently accurate for future needs. Non-extensible

Figure 7.9 LCCbyg Dynamo transfer UED sequence: Functions and risks explanation

Specific measures must be taken by the auditor to familiarize himself/herself with the transfer scripts. Firstly, the auditor must ensure correct sub-script mapping. As mentioned before, the units of data transfer are category dependent. Rooms will take room floor finish information, whereas windows will take area information in m2, regardless of its location within the room.

Subsequently, object families who rely on object counts, such as furniture, plumbing fixtures, or casework, will use an object unit-based script. On the other hand, system families such as railings or stairs will require a separate script operation to transfer the data to the LCCbyg software. This will also be true for other system families that may be added in the future. Two sub-scripts were created, enabling the transfers. One for stair runs calculation, which takes measurements from parameters unique to stair runs, such as run width, or thread height, and for railings, which is based on the length parameter measured in running meters. This script can also be used for other length-based measurements (RMT), such as skirtings or ceiling architraves.

The script is made in a way that will be forgiving of some prevalent errors. It has an inbuilt list cleaner, meaning that null values and 0's will be removed from the list, to ensure the transfer works. However, this means that the auditor must be aware that there will be no error visible, if the data inside the parameters are incorrect and must visually check if the subscripts are sending the data through, by looking at the watch nodes. The data can be revised before export either by checking the schedules, or by applying Revit filters to objects, to hide all elements that have been mapped, and expose those with missing information. A rigid solution should ideally use error prompts instead of concealing errors. It is a bad practice to hide transfer errors as those lead to calculation errors.

As of now, the approach used to transfer data between the .CSV file and the LCCbyg software is creating a template file, which means that automatic versioning inside of the LCCbyg software is currently not supported. A workaround to versioning can be achieved by creating lateral files and assessing the data using PowerBI. This would be most likely addressed in the future if it was considered a significant letdown. However, this does not mean that versioning is impossible; it just means that it might need some degree of manual rework at this point. An alternative is to publish a second child subset and then disable the previous subset, which will permit versioning.

Lastly, the current script is using Python 3.7, and therefore it requires an extra step, as opposed to direct export through Dynamo, which uses Python 2.7. This is most likely a simple fix, but during the short development time of the script, this was not achieved.

7.4 Using PowerBI

The PowerBI project dashboard enables users to have an executive view of the data gathered throughout the optimization process. The stakeholders can then make informed decisions on design considerations and are able to compare solutions with other data sets. The project dashboard can consist of multiple components. The representation model envisions using SVG floor plans to link data with visual location representation.

Graphs, tables, and lists are other mechanisms of depicting data and information regarding the room cleaning cost, the floor cleaning cost, and the objects contained within the rooms. Furthermore, graphs present the visual representation of anomalies not so easily visible in the datasets, or in the LCC graphs, which summarize total data values.

Project Dashboard	**Floor plans**	**Graph charts**	**Tables**	**Lists**
Purpose: Executive information on the project status.	Purpose: Visualize data per specific room.	Purpose: Visualize data per specific room.	Purpose: Visualize data per specific room.	Purpose: Visualize data per specific room.
Features: Interactive data display on various graphs.	Features: Direct link to room number. Data adjustment based on Room location	Features: Direct link to room number. Data adjustment based on Room location	Features: Direct link to room number. Data adjustment based on Room location	Features: Direct link to room number. Data adjustment based on Room location
Links: Revit model data LCC data Revit model plan views	Links: Total Room cost Total Floor cost Floor Finish Populate CSV Script	Links: Total Room cost Total Floor cost Floor Finish Populate CSV Script	Links: Total Room cost Total Floor cost Floor Finish Populate CSV Script	Links: Total Room cost Total Floor cost Floor Finish Populate CSV Script
Objects: Visualizations Fields	Objects: Entry fields.	Objects: Entry fields.	Objects: Entry fields.	Objects: Entry fields.
Risks: Wrong data source Manual editing Inexperienced users	Risks: Wrong data source Manual editing Inexperienced users	Risks: Wrong data source Manual editing Inexperienced users	Risks: Wrong data source Manual editing Inexperienced users	Risks: Wrong data source Manual editing Inexperienced users

Figure 7.10 PowerBI project dashboard components summary

Graphs show properties such as cost variance relative to the room size or cost variance between rooms of the same type, function, occupancy, or size. The data can be further used to calculate contract values for specific building users, albeit if the building was divided between various tenants so that the costs can be assigned accurately and without a dispute.

A set of scripts is created in Dynamo, to showcase how such data can be grouped, filtered, and organized to support the transfer. This list is not explicit; it thus serves as a method representation, and parts of the Dynamo scripts can be utilized to derive other metrics and performance indicators.

A Belonging Room ID parameter is used to associate a component with the room it is residing in. For this to work, the object must have a 3D geometry, so that the geometry bounding box can intersect the room bounding box.

Belonging Room ID	**Room Floor cleaning cost**	**Total Room cleaning cost**	**Simple Vector Graphics Script**	**Populate CSV Script**
Purpose: Ensure objects are located within rooms to calculate total room cost. Features: Script appending Room ID to every object, relative to its location. Links: Objects and Rooms. Revit model. Objects: Stairs, Furniture, Casework, Plumbing, Doors, Curtain Panels, Windows, Railings, Rooms. Risks: Incorrect model preparation. Insufficient data for transfer (especially for area or running meters scripts).	Purpose: Calculate cost of cleaning floors relative to rooms for PowerBI analysis. Features: Script calculating cost based on floor surface and area. Links: Sub-scripts Objects: Dynamo nodes. Data objects. Version ID Change Dictionary Risks: Floor finishes are not added to Room data in the schedule.	Purpose: Enabling comparing versions inside LCCbyg Features: Script searching for all objects in room. Adding each object cost Links: Model elements, .CSV export Objects: Version ID, Version template ID, IsTemplate (yes/no) Risks: LCCbyg version update. LCCbyg crashing.	Purpose: Export SVG plans of the building to PowerBI. Features: Exports plans with one button press. Exports corresponding Numbers. Links: Revit Model Room numbering PowerBI Objects: SVG script Risks: -	Purpose: Transfer .CSV data to PowerBI database. Features: Collect all schedule parameter data relevant to PowerBI Links: SVG floor plans Data from Revit: Belonging Room ID Room Numbers Floor cleaning cost Room cleaning cost Objects: Script Risks: Dynamo knowledge needed to extend script functionality.

Figure 7.11 Scripts which are supporting the PowerBI data transfer

One of the possible performance indicators may be the room floor cleaning cost. This approach calculates the cost of cleaning floors relative to the room location within the building. Depending on the material, the frequency, and the intensity, the costs are calculated per year. Note that LCC costs are not calculated in PowerBI, and the costs do not account for Future Values (FV), nor do they consider a building lifespan of x years. All data is calculated yearly in the Present Value (PV).

Another metric is the Total Room Cleaning Cost, which encapsulates all objects with the Belonging Room ID parameter inside a room, and then uses a calculation mechanism to multiply the cost, again, depending on the frequency, the intensity, and the cost of cleaning the object. The two metrics of floor cleaning and room cleaning are then added, and they can be visualized and compared inside of the PowerBI tool.

Another useful script is the SVG generator script. This step can be omitted, as many websites provide such service free of charge, based on the images given; however, this approach allows for a rapid transfer of the room plan model, directly to the SVG. This can be made even better by adapting the script to work with object instances within the room, thereby showing the data per object, as initially intended.

Lastly, *populate .CSV* script has been created, to automatically extract data from Revit schedules and create a consistent .CSV file, ready for PowerBI input. This script collects parameter values and converts them to a tabular form. The list of parameters collected by the script is presented in the figure below. The list is not exhaustive, and can be easily altered, or appended, depending on the performance measurement use case. The script can be seen in the Appendix F PowerBI scripts.

8 Data transfer Prototyping

To enable a connection between the model environment in Revit and lifecycle costing environment in LCCbyg, an interfacing VPL in Dynamo is used to extract information directly from Revit, by plugging to its API, and performing a series of operations, including categorization, grouping, and mapping. More on Dynamo functionality can be found at "What is Dynamo? | The Dynamo Primer" (2018). The resulting.CSV file is parsed using Python.

The prototype is capable of grouping the parameters and exporting them to a structured list, for multiple categories, accounting for the anomalies of their calculation approaches. The prototype does not entail any other LCC calculations beyond indoor cleaning. It is, however, possible to apply the principles presented below and make them available for other LCC categories such as object-based construction, maintenance, or demolition cost. The following methods are used to create shared parameter placeholders:

- Parameter insert Script creates shared parameters by reading an Excel shared parameter file.
- Schedules are manually adjusted, and the .rvt or .rte template file is saved for use in other projects. Contact the author for the Revit schedules template file.

The parameters can be loaded directly to the families or added to the project. Adding them to the project by using "Insert views from file" command is the preferred option, as adding parameters directly to elements makes version control quite cumbersome, unless when using a database. As Dynamo can only be run on a single project at a time, it means that changing the information contained within families, even using Dynamo scripting, will require opening each family at a time, thereby taking more extended time, as opposed to changing the parameters at the Revit project level.

The model is now ready for export having the above parameters populated with data, ideally, through a bi-directional link with the database, see Chapter 7.1 - Using the dRofus database, as an example. The following Dynamo VPL scripts were created for the LCCbyg.CSV data mapping:

- **Main Script**: LCCbyg cleaning data transfer to .CSV (see Appendix D - LCCbyg Transfer Dynamo Scripts for reference to the Dynamo Script canvases):
  - Custom node #1: Grouping objects by Areas (Curtain Walls, Windows, Doors, etc.),
  - Custom node #2: Grouping objects by Running Meters (Railings, skirtings, etc.),
  - Custom node #3: Grouping objects by Unit Count (Furniture, Plumbing, etc.),
  - Custom node #4: Grouping floor surfaces by Room (Room instances),
  - Custom node #5: Grouping stair areas by Stairs run type (Stairs types).
- **Python Script**: .CSV to LCCbyg XML mapping script (see Appendix E – Python Transfer Script Appendix D - LCCbyg Transfer Dynamo Scripts for reference to the Dynamo Script canvases)

The following Dynamo VPL scripts were created for the PowerBI.CSV data mapping(see Appendix F PowerBI scripts Appendix D - LCCbyg Transfer Dynamo Scripts for reference to the Dynamo Script canvases):

- Script #1: Assign *belonging_RoomID* to every object instance including rooms,
- Script #2: Calculate the cost for floor cleaning of each room instance,
- Script #3: Calculate the total cost including floor cleaning and object cleaning of each room instance,
- Script #4: Export structured schedule information .CSV for PowerBI application.

Furthermore, a Dynamo, Python coded script by Adam Bear (Bear, 2019) was borrowed to convert Revit floor plans to.SVG floor plans.

The following Dynamo VPL scripts were created for the cleaning calendar work schedule (see Appendix G – Future Research Calendar Potential & Filter setting Scripts Appendix D - LCCbyg Transfer Dynamo Scripts for reference to the Dynamo Script canvases):

- Script #1: Set cleaning filters to views
- Script #2: Set cleaning parameters to objects

Below are subchapters dwelling into a further explanation of each script structure supporting the data transfer and presentation.

8.1 Dynamo Scripting between Revit and LCC

The purpose of this transfer is explained in the User Environment Design Chapter 7.3 - Using LCCbyg. The script modules can accommodate various architectural components.

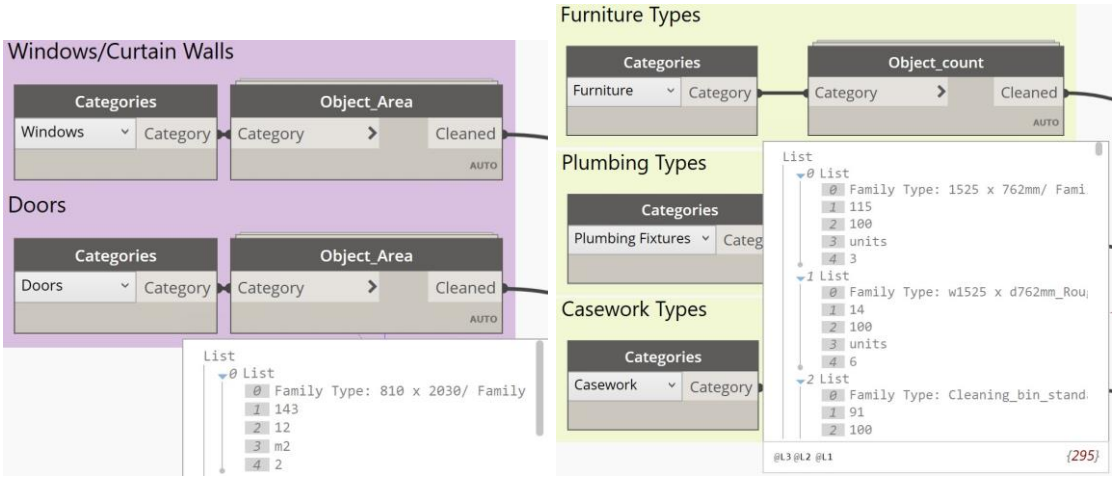


Figure 8.1 Dynamo components transferring quantities (1/2)

The primary categories used can be seen in Figure 8.1, and those support the transfer needed for the initial case study, as well as the implementation case study. The modular design of scripts embodies complex transfers to be interlinked, or nested within a larger software schema.

The "*Object Area*" custom node groups data for families/objects with area parameters, such as doors and windows, where quantity does not well represent the cost of maintenance. The "*RoomFloorsbyID*" custom node groups data for room instances with the area and finishes parameters.

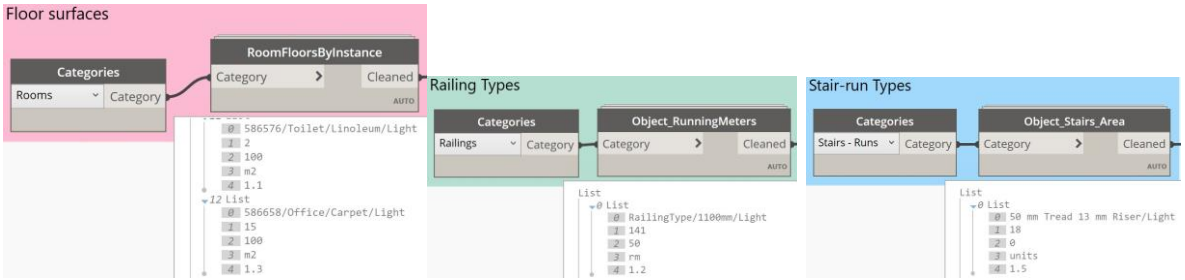


Figure 8.2 Dynamo components transferring quantities (2/2)

The "*Object_count*" custom node is grouping data for object types, based on the repeated count of the objects within the project. The custom node best suits furniture and fixtures. The data is then converted to a single list, appended by the version ID, and the Danish characters are converted to the standard alphabet set.

8.2 Python Scripting between .CSV and LCCbyg XML

Finally, the data is sent to a .CSV file, ready to be converted to the LCCbyg XML format. The Python version used by Revit and Dynamo is IronPython – version 2.7. Therefore, there is no possibility to use some libraries that come as standard with Python 3.7 in the Anaconda environment. There are also some syntax differences that can be corrected using a commonly available Python validator[1].

The basic principle of converting a .CSV file to an LCCbyg XML file is to access the Element Tree by using the *ElementTree* Python module. The module has a function called XPATH, which allows accessing the right level in the file structure and altering the information at that level. The code creates new child nodes containing pieces of information on objects and objects' attributes, following the LCCbyg XML schema. The prototype script is hardcoded and does not use LCCbyg XSD; therefore, it is only a proof of concept and not a complete solution.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct  7 13:45:51 2019 by Adam Piaskowski
akptech@outlook.com
"""
import csv
import xml.etree.ElementTree as ET
tree = ET.parse('./Test samples/EmptyProject_ENG.xml')
root = tree.getroot()
subgroup_element = root.find("./accountplan/maingroup[@  name='Buildings, internal']")
room_name = 'Version2'
yesno = 'true'
values =''
```

Figure 8.3 Importing modules to Python

Modules imported are .CSV – a module needed to read comma-separated values files, *XML.etree.ElementTree* – a module needed to traverse through a tree structure. Parsing the Element tree (ET) lets accessing information within the XML file. Navigating to the correct subgroup element is made possible with the root find function, which takes the input of the *XPATH*. For an explanation on *XPATH* refer to ("XML Schema Tutorial," 2019).

```
subgroup = ET.SubElement(subgroup_element, 'subgroup', {'name':room_name,
                                                        'istemplate':yesno})
with open('DGNBdatapass.csv', 'rt') as f:
    row = None
    reader = csv.reader(f)
    for row_att in reader:
        object_name, amount, frequency, units, unitprice, inputvid, vid, include = row_att
        if row is None or object_name != row:
            # Start a new group
            row = ET.SubElement(subgroup, 'row', {'name':object_name,
                                                  'include':include,
                                                  'units':units})
        basevalues = ET.SubElement(row, 'basevalues')
        if values is None or values != basevalues:
            values = ET.SubElement(basevalues, 'values', {'frequency':frequency,
                                                          'unitprice':unitprice,
                                                          'vid':vid})
        inputvalues = ET.SubElement(row, 'inputvalues')
        if values is None or values != inputvalues:
            values = ET.SubElement(inputvalues, 'values', {'amount':amount,
                                                           'vid':inputvid})
new_xml_tree_string = ET.tostring(root)
with open('./Test samples/OfficeModel_test.xml', 'wb') as f:
    f.write(new_xml_tree_string)
```

LCC datafile exported from Revit

Name of the Revit object

basevalues = frequency + price

inputvalues = quantities

Figure 8.4 Passing comma-separated values to the XML *SubElement*

The *subelement* takes values on a row-level, which has two children: base values and input values. At each level, the data is parsed. The data cells include, amongst others, object name, frequency of cleaning,

[1] http://infoheap.com/Python-lint-online/

and the number of objects, as well as a unit used and unit price. The ending step is to convert the Element Tree to string and write it to a new file at a specified location.

8.3 Dynamo Scripting between Revit and PowerBI

Dynamo scripts can automate a big part of the job when exporting data from the Revit model to the PowerBI model. The model can then be used to derive KPIs based on the data from the Revit model. Two examples of KPIs were derived to support the process description.

The first KPI is the yearly cost of floor cleaning per room. This KPI calculates the cost of a square meter of a floor surface, which is multiplied per room size. Furthermore, the cost per single cleaning is then multiplied by the yearly frequency. The cost depends on the intensity; hence, three intensities (light, average, demanding) are calculated individually and then added together.

The same calculation principle concerns the objects situated in rooms. Those objects are grouped in rooms using their geometry as a location indicator. A single script calculates both floor cleaning costs and object costs, and then, sets cost parameter values within the room instances, and object instances. The script can be found in Figure 14.24 PowerBI Total costs of items per year script.

The yearly costs of objects cleaning, again, depending on intensity, are then added together with the floor cleaning cost, giving a cost of room cleaning all-inclusive. This script involves matching keys with lists, and the script can be found in Figure 14.25 PowerBI Grand totals per room script. The .SVG files are generated based on the floor plans. These floor plans are later connected with the data inside PowerBI. The rooms must have unique numbering for this script to work.

Figure 8.5 .CSV data and model .SVG maps visualized in PowerBI.

Following the SVG exports, the data can be exported to a .CSV file using the script showcased in Figure 14.26 Script automating .CSV export of room data from parameter placeholders. This script exports room schedule information needed for the PowerBI to work with the .SVG plans and the costing parameters set up earlier on. The last thing left to do is to map the .CSV data frames and the .SVG floor plans with the graphical interfaces available in the PowerBI application.

9 Analysis

The purpose of this workflow proposal is to check if the proposed approach can give more accurate results with lesser effort. The goal of the analysis chapter is to analyze the tool, rather than the results that can be obtained from the individual projects. It is as if a hammer was being created to hammer nails, and the quality and usability of the hammer was tested, rather than the applicability of the project the hammer was used on.

9.1 Transfer accuracy

The proposed approach allows for adding costs based on objects contained within the Revit model, meaning that costs can be flexibly added and removed depending on the quantity, intensity, frequency, and object cost per unit of measure. The accuracy relies on correct BIM modeling and correct costing databases for cost estimation. However, the accuracy of the tool is somewhat concerned with how flexibly the tool will be able to transfer this information.

Furthermore, dividing intensity into three categories, “Light, Average, Demanding,” can pose inaccuracies due to various assumptions companies may take on what is considered “Light” and what is considered “Demanding.” It may relate to high-quality standard cleaning, or “Demanding” may mean, say in the case of carpet cleaning, that the carpet needs a heavy-duty carpet washing machine to freshen the carpet.

A building-specific definition of what is meant by Light, Average, and Demanding for specific building parts and furniture objects should be provided. This will further inform the object price. Such standards already exist (Green Building Council Denmark, 2017) for the DGNB surface material cleaning; however, the considerations specified there do not relate to the intensity of cleaning, but rather the ease of accessibility, say the height of the window – accessible with or without a ladder, as an example (Green Building Council Denmark, 2017, pp. 385–387).

The transfer of data sends information that is beforehand linked from the database directly to the Revit parameter placeholders. Therefore, if the data is accurate, there is hardly anything that can go wrong, given the modeling techniques are in accordance with the assumptions presented earlier.

9.2 Workflow time savings

LCCbyg

The proposed approach requires a certain Level of Detail (LoD) of object geometry and information. Though creating Revit models should not be considered as time spent on making the LCC transfer, some adjustments mentioned in the User Environment Design in chapter 7, may take a considerable amount of time if the workflow is not standardized and the techniques are not followed by the modelers. Given the process is standardized, altering model information will depend on the scale of the task.

The actual process of data transfer is pretty much fully automated. All the auditor has to do, once the model preparation phase is over and validated, is to click a button in the Dynamo player to transfer the data. The processing time takes between 5 and 15 seconds. After that, the Python script must be appended with a new file name and ran to transfer data. This takes further 10-15 seconds. Lastly, LCCbyg has to be re-opened, and the results can be observed, taking additional 10-20 seconds. This means that the time taken for the tool to run a full calculation cycle (excluding the time taken to alter the model environment) takes less than one minute.

PowerBI

The process of transferring information to PowerBI takes a little longer. Given the SVG map is already there, and the PowerBI dashboard is setup from the template, four scripts are executed, one following the other. The first script must reassign object IDs from Rooms to objects, and the second calculates floor cleaning cost, the third populates room costs and the final script transfers data from Revit schedules to a .CSV file.

The PowerBI dashboard template must be linked to new datasets, which may take a couple of minutes. The resulting time for running scripts and linking datasets should take no longer than 5 minutes. It takes a considerable amount of time to create a new Dynamo script for a new KPI dataset; however, this is a one-time-only operation.

9.3 Applicability

The benefit of object-oriented cost calculation is that any objects can be calculated. Internal cleaning can be just as well converted to gardening or washing facades. All it will take is to apply the same structure – by using quantity, intensity, frequency, and object cost per unit of measure, while changing the objects, the cost databases and the descriptions for what "Light, Average, and Demanding" intensities mean.

The author is convinced that any objects cleaning cost can be calculated using the proposed process workflow. It may involve changing some custom Dynamo nodes, too, for example, change the area or volume calculation principles; however, the overall approach of data transfer will vastly remain the same.

Say, instead of calculating the LCC of cleaning, gardening was calculated, and something as replacing soil was to be measured. Soil is a volume, and the object category for soil may not exist. However, a script can be created, which will take generic objects of type "soil" and will calculate the metric volume of soil factored with soil compression multiplier.

Following the same logic, calculations for specific window awnings or brick and grout layout may be calculated, with only change needed being the modification of the Dynamo script. If the auditor has certain Dynamo VPL skills, it should not be challenging to alter the script, as the granularity of Dynamo scripting is high, while inputs and outputs clearly visible.

Additionally, the Python-based Element Tree XPATH may require alteration if other LCCbyg software categories were altered, beyond internal cleaning. This can again, follow the same approach and overall structure of the script, only changing names of placeholder values of children and sub-children.

To further test the applicability of the proposed approach for data transfer, as well as data displaying using PowerBI and further benefits the approach can have during the operations and maintenance phase of the building, various stakeholders were questioned to analyze how they could benefit from the object-oriented approach.

9.4 Stakeholder interviews and testing

For three months, various stakeholder meetings were arranged to find out how would object-oriented costing could affect their workflows. The stakeholders were presented with a PowerPoint presentation, relevant for their trade and interests, followed by a prototype presentation ended by an open discussion. The results from those meetings are summarized in Table 3 and Table 4.

The full transcription summary can be found in chapter 14.12 - Appendix I – Summarized results of stakeholder meetings. Chapter 14.13 - Appendix J – Key benefits and challenges is a chapter describing

the authors' take on the benefits and threats of the approach. It has been removed from the main section to keep the page count reasonable.

Table 3. Stakeholder opinions summary – object-oriented cleaning

STAKEHOLDER	MAIN CONCERN	MAIN BENEFIT	INTEREST
BUILD TO SELL OWNERS	Extra costs of design analysis	DGNB certificate.	Not interested
BUILD TO USE OWNERS	Extra costs of design analysis	Long-term savings, FM cost tracking, detailed Service Agreement, Material choice help, DGNB certificate.	Interested
ARCHITECTS	Complexity, Lack of early design info, extra workload	Long-lasting design, DGNB certificate, Extra fees.	Partially interested
FM SOFTWARE DEVS.	Relying on geometry, updating models, proprietary software	sensor integration, relatively easy implementation, value creation.	Interested
LCCBYG SOFTWARE DEVS.	Lack of early design info, too specific, rigid transfer, deviation from DGNB, Lack of cost databases.	KPIs visualization, FM integration, increased reliability	Partially interested
HOSPITAL FM TEAM	Must be integrated into the same software already used. Older hospitals lack BIM models.	Integrated issue reporting, design option support, location-based object ID, IoT potential, Quality Control.	Interested
CLEANING MANAGEMENT	Expensive software and hardware.	work scheduling, route optimization, pay per object, tacit knowledge transfer, performance tracking. Accurate tendering. Quality control.	Interested
OPERATIONAL CLEANERS	Undercosting of objects, reliance on technology, IT competences.	work schedule, more accurate pay, relaxed working time, reduced surveillance.	Interested

Architects and BIM managers were also queried about the Revit-LCCbyg automation workflow. A summary of the interview and testing meetings is visible in Table 4 below.

Table 4 Stakeholder opinions summary – data transfer proposal

STAKEHOLDER	MAIN CONCERN	MAIN BENEFIT	INTEREST
ARCHITECTS	Not every project requires LCC, troubleshooting.	Increased efficiency, reduced auditing costs. Increased utility, dRofus integration.	Interested
BIM MANAGERS	Dynamo interaction, Coordinating modeling techniques, updating the database, challenging existing methods, Issues with grouped objects.	Simplified auditing process, standardized workflow, dRofus integration.	Interested

10 Results

In this chapter, cost approaches, applications, and complexity will be analyzed in detail to present the reader with convincing reasons as to why a more detailed analysis can find application not only in the correct cost estimation but preliminarily for decision making.

10.1 Comparison of cost

The two quantity takeout approaches were compared. The existing, DGNB approach, which uses an m2 basis, against the proposed approach using a combination of metrics explained in chapter 4.2 on page 10.

Cost deviation between DGNB approach and the proposed approach

What is apparent is that the existing cost approach proposed by the DGNB template has a high potential for undercosting the actual cost of cleaning, mainly because of excluding objects. It may, at the same time, overvalue the cost, as a result of taking generic values for cost estimation. That said, to appropriately compare the two approaches, a trustworthy object costing database would be required. It is possible to compare costs using two price points to visualize how small price changes multiply due to the frequency of tasks.

To compare "apples to apples," the frequencies on all components are set to be the same for both the existing template and the proposed work template. The pricing based on areas is taken directly from the existing template. Prices for cleaning furniture and other unit-based objects are derived from cleaning time calculations presented in the appendix 14.2.

It can be observed that the cost deviation on the same frequencies and quantities is quite considerable, as depicted in the figure below. This was, however, a single test, and the prices were estimated with an unvalidated method. The thesis aims to create the tool, and further analysis is subject to separate research.

Conditions:

Frequency from DGNB template = Constant
Frequency for object cleaning = based on case study furniture 2x a week, Bathrooms 5x a week
Cost priced from DGNB template = Constant
Cost for object cleaning = average cleaning time
LCC Period: 50 years. Constant.
LCC Price development = Constant

Total LCC Costs

	Light	*Average*	*Demanding*
Method: DGNB template cleaning parameters	7 861 964.00 DKK	13 054 053.00 DKK	15 543 383.00 DKK
Method: Object-based cleaning	12 570 532.00 DKK	17 888 978.00 DKK	25 999 257.00 DKK
Increase	4 708 568.00 DKK	4 834 925.00 DKK	10 455 874.00 DKK
% increase	59.89%	37.04%	67.27%

Figure 10.1 Cost comparison variables and constants

The sample results are here to showcase the potential applications and the data that can be generated from the models, once the cost databases are accurate and available.

Cost deviations using the proposed approach

A case study is examined to illustrate the potential application better. The case study model has 106 laminated and 23 rough-surfaced desks. By changing the current desk mix to either 129 laminated or

129 rough-surfaced desks, an experiment can be run checking the difference in cost resulting from the change in the cleaning effort needed. Multiplying the tasks by 50 years will definitely generate significant results. This is taken as a demonstrative example. Of course, having so many levers to press, the optimization possibilities are quite diverse, and desks are treated as an illustrative example. A more realistic example will look at a decade period, rather than 50 years, as desks' lifespan is rarely this long. However, as the general calculations are performed on a 50 years period, an assumption is made that the desks are replaced without affecting the desk replacement cost.

A	B	C	D	E	F	G	H	I
Family and Type	Count	Family	Cost_Av.	Cost_D	Cost_L	Average_Freq	Demanding_Freq	Light_Freq
Cleaning_Desk_Standard: 1525 x 762mm	106	Cleaning_Desk_Sta	5	8	3	100	50	100
Cleaning_Desk_Standard: w1525 x d762mm_Rough_Surface	23	Cleaning_Desk_Sta	10	14	6	100	50	100

Figure 10.2 A table is showing two exemplary desks of two varied finish materials

The rough-surfaced desks are more expensive to clean at an average cost of 10dkk per object. The smooth-surfaced desks are less expensive to clean at an average cost of 5dkk per object.

In this case, the above Figure 10.2 showcases the pre-optimization setup, where some of the desks had been replaced from the standard smoothly laminated desks to desks covered with a rough rubber anti-skid surface, which is perhaps convenient for a mouse tracker, but not so much when it comes to cleaning.

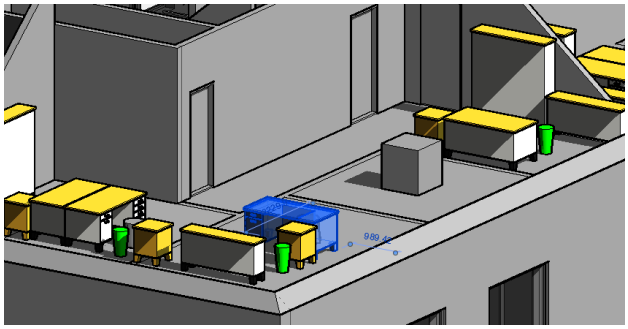


Figure 10.3 Model representation of smooth surface desks

White and yellow desk surfaces will be changed to black antiskid surfaces, resulting in need of an increased effort to maintain its cleanliness.

Due to its surface properties, the cleaning process of such surface requires the iterative application of cloth, as the human skin cells struck between the porous surface, form stripes of skin which, with every motion, only move a few centimeters. For the laminated desks, a cloth has a stickier surface, so all the human skin and dust sticks to it. This is not the case for the rough desks; hence, they tend to take 2-3 times longer to clean than the standard desks.

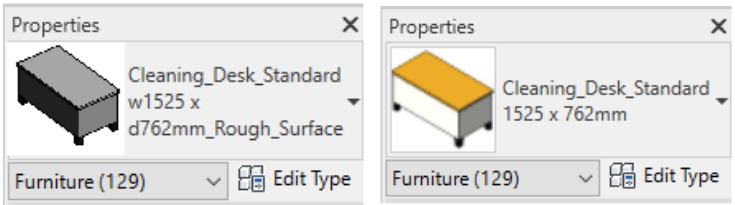


Figure 10.4 Changing the type of desks from rough to smooth-surfaced

Note that the image for the rough-surfaced desk has been altered for demonstrative purposes. Revit keeps the same image for all instances of the family by default.

As a result, the costs associated per item are almost doubled. In the scenarios below, all desks are either smooth or rough. For 129 desks of smooth surface, the cost of maintenance cleaning is reduced by 3,250,000DKK over 50 years (see Figure 10.6). The cleaning intensity and frequency are the same, and the only difference is the time it takes to clean a single desk, therefore the cost.

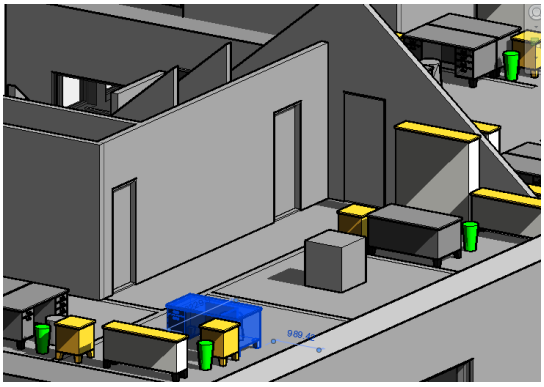


Figure 10.5 129 Rough surface desks (notice the desk color change in the BIM model)

It can be argued that the desks will not remain the same for 50 years' time, which is most likely correct. The same calculation can be assessed in 10 years, and the results show a difference of 1.1m DKK. The real difference may be less significant; however, the vital point here is twofold. One, the difference clearly exists, and two, the scale factor of frequency and price make every single second count.

Conditions:
Cost for object cleaning = Variable
Frequency for object cleaning = Constant
LCC Period: 50 years.

Cost per object

	Light	*Average*	*Demanding*
Design Option: Laminated desks	3.00 DKK	5.00 DKK	8.00 DKK
Design Option: Anti-skid desks	6.00 DKK	10.00 DKK	14.00 DKK
Increase	3.00 DKK	5.00 DKK	6.00 DKK
% increase	**100.00%**	**100.00%**	**75.00%**

Frequency per object

	Light	*Average*	*Demanding*
Design Option: Anti-skid desks	100 times/year	100 times/year	50 times/year
Design Option: Laminated desks	100 times/year	100 times/year	50 times/year
Increase	0 times/year	0 times/year	0 times/year
% increase	**0.00%**	**0.00%**	**0.00%**

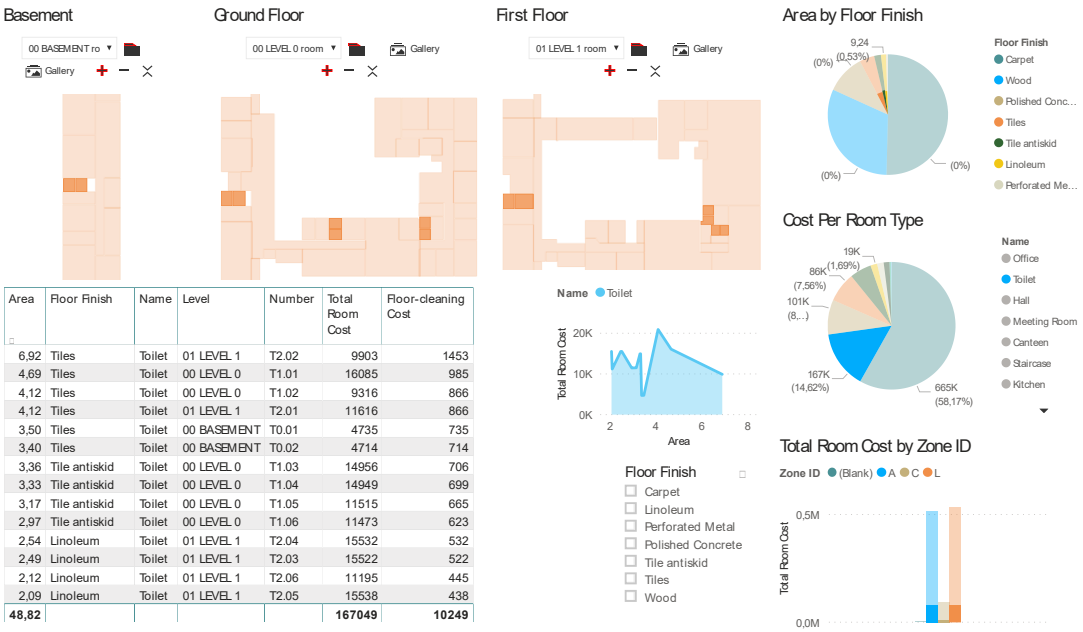
LCC Cost Difference

	Total LCC cost
Laminated desks	25 607 741.00 DKK
Anti-skid desks	28 858 267.00 DKK
Increase	3 250 526.00 DKK
% increase	**12.69%**

Figure 10.6 LCC cost difference resulting from the material finish for 129 desks

10.2 Beyond LCC – Operational Use Cases

The proposed software applications vary – from lifecycle costing, which formed the central theme of this thesis, to future applications concerning Facility Management and cleaning operations. The primary purpose of this report was to automate the data transfer between a BIM model and an LCC calculation tool. Furthermore, a proposal for the costing approach was envisioned. Future applications were then stemming from the voices of stakeholders, as well as the data gathered from the case studies.



Area	Floor Finish	Name	Level	Number	Total Room Cost	Floor-cleaning Cost
6,92	Tiles	Toilet	01 LEVEL 1	T2.02	9903	1453
4,69	Tiles	Toilet	00 LEVEL 0	T1.01	16085	985
4,12	Tiles	Toilet	00 LEVEL 0	T1.02	9316	866
4,12	Tiles	Toilet	01 LEVEL 1	T2.01	11616	866
3,50	Tiles	Toilet	00 BASEMENT	T0.01	4735	735
3,40	Tiles	Toilet	00 BASEMENT	T0.02	4714	714
3,36	Tile antiskid	Toilet	00 LEVEL 0	T1.03	14956	706
3,33	Tile antiskid	Toilet	00 LEVEL 0	T1.04	14949	699
3,17	Tile antiskid	Toilet	00 LEVEL 0	T1.05	11515	665
2,97	Tile antiskid	Toilet	00 LEVEL 0	T1.06	11473	623
2,54	Linoleum	Toilet	01 LEVEL 1	T2.04	15532	532
2,49	Linoleum	Toilet	01 LEVEL 1	T2.03	15522	522
2,12	Linoleum	Toilet	01 LEVEL 1	T2.06	11195	445
2,09	Linoleum	Toilet	01 LEVEL 1	T2.05	15538	438
48,82					**167049**	**10249**

Figure 10.7 PowerBI User Interface

An interactive user interface showcases an executive view at the costing data. Bathrooms are highlighted and all diagrams adjust accordingly to showcase the bathroom data subset, on the background of the overall dataset.

The applications from there on are manifold. Costing, tendering, calendars, training, feedback, change management, user engagement, is certainly worth investigating. Chapter 14.13 discusses the potential risks and benefits of object-oriented cleaning, while a summary of stakeholder opinions can be found in Table 3 on page 44.

Tools such as PowerBI best showcase how easily the data gathered from the models can be digestible by the end-users and how it can potentially inform FM operations. The two simple KPIs investigated in this report are one; the floor cleaning cost per room, and two; the total cleaning cost per room.

The sample dashboard from Figure 10.7 PowerBI User Interface permits some assumptions to be drawn from the graphical interface. The bottom right graph calculates the Total Room cost by Zone ID, enabling splitting the bills of cleaning amongst different building users. In the case study, two companies occupied one building and shared the canteen.

For one, Figure 10.8 Cost of cleaning relative to the room area shows how room size does not necessarily scale linearly with the cost, and therefore poses an interesting problem with the way the current calculations approach estimate cleaning tenders. Having area data is better than no data at all, but that said, having object data is better than area data, and the figure below demonstrates it.

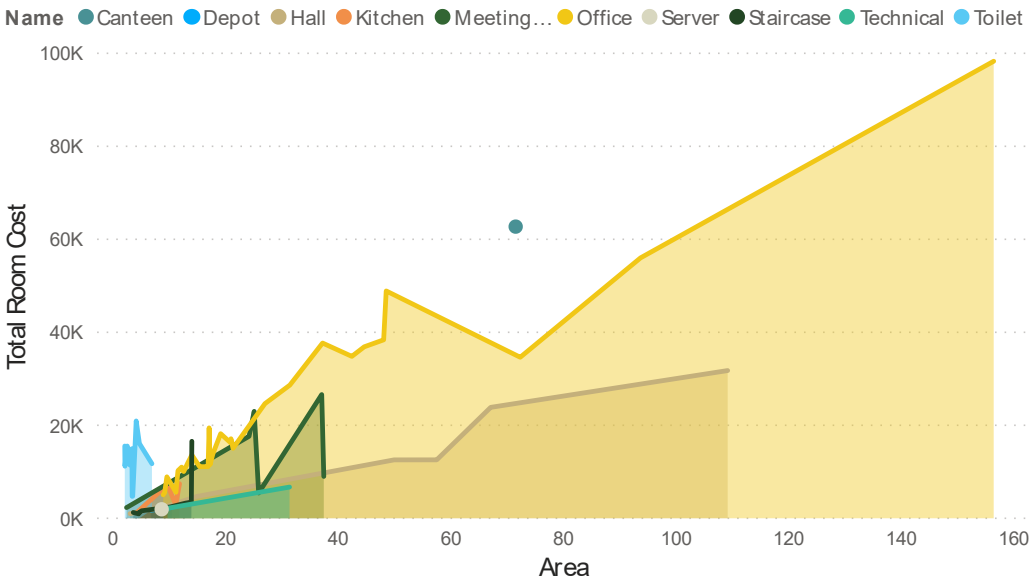


Figure 10.8 Cost of cleaning relative to the room area

The room area does not ideally scale with cost, therefore assuming only area metric presents inaccuracies.

The cost of cleaning offices increases linearly in some cases, yet for some other cases, bigger does not mean more expensive. This is in the case of the attic rooms, as well as executive offices, where the floor area is far higher per desk than in the other spaces. Were the costs corresponding to areas, the functions' lines would be perfectly straight.

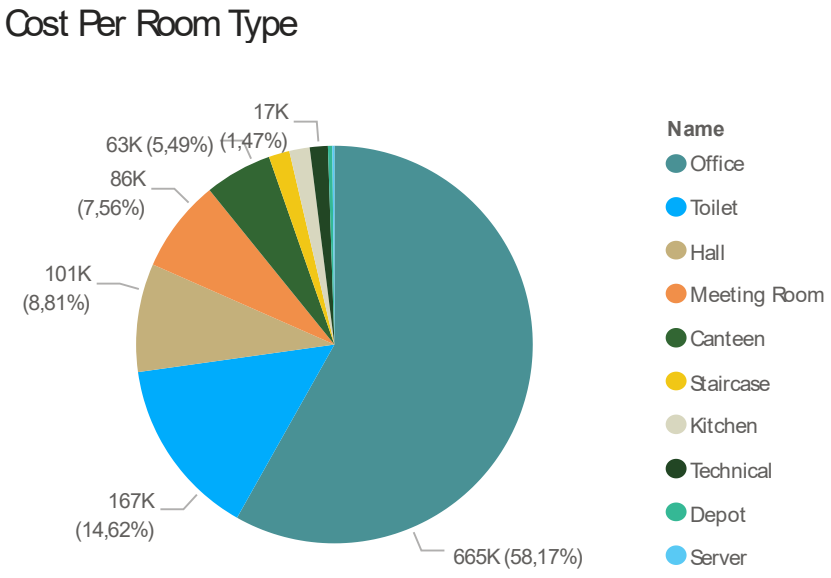


Figure 10.9 PowerBI dashboard - an overview of cleaning cost allocation

The hallways, kitchens, and bathrooms also show varying costs. It can be observed that the bathrooms, despite its small areas, are significantly more expensive to clean than any other spaces – which is valid from personal experience. The results of different versions can be compared against each other, showing the distribution of the cost. This would be even better showcased if the .SVG maps transferred object geometry. KPIs at object geometry would be ultimately desirable.

11 Discussion

The current approach of estimating the lifecycle Cost proposed by the DGNB is insufficiently detailed to inform the design and further operations adequately. The assumption is that object data-driven cleaning information takeout from a BIM model will increase the accuracy of cost prediction and will enable more detailed optimization of the design.

Q1. Is there a possibility the current approach of estimating cleaning lifecycle costs is inaccurate?

All models can be flawed due to the boundary critique condition (Ulrich, 2005) but some can be more accurate than others. It can be seen from the resulting test runs, that a seemingly insignificant price change of 5DKK a day, per cleaning of a single desk, can result in over 3,250,000DKK in savings over 50 years period, considering a three times a week cleaning frequency and 130 desks in the office. This change would not be picked up by the traditional, aera based approach.

Furthermore, not only changing costs impacts the results, including the objects at all seems to have the most impact. The DGNB status quo approach does not consider the cleaning costs of furniture or other interior fitting equipment, thereby, in the opinion of the author, greatly under-costs its LCC assumptions. Most stakeholders agree that there is a big potential in object-oriented costing.

Practical observations lead to conclusions that certain material finishes and object designs lead to a more tedious or demanding cleaning. Furthermore, the increase of frequency greatly affects the price, yet it is vital to keep objects in the usable state so that the assumptions about the economic lifespan can be sustained.

Q2. Will the proposed approach increase the complexity of calculations experienced by the DGNB auditor?

An honest answer to this question is yes, and no. The complexity will stem from rigid model checks consisting of checking if objects are modeled, whether they contain the parameters, and whether the parameters are populated correctly. There will be no extra complexity in calculating per se, as the LCCbyg software will remain the same tool to calculate the costs; however, the auditor will likely need to be able to understand at least the basics of Dynamo, to troubleshoot any issues that may arise during the quantity transfer. One of the frequent issues is that the console can get stuck after one calculation and may require restarting for a quick refresh. However, given the auditor must perform the transfer regardless of the approach, and that there will be multiple transfers following each design iteration, the complexity encountered is solely initial. The repeated optimization iterations will be straight forward and take under a minute to run, as presented in the analysis chapter on accuracy and time taken. Once a rigid workflow process is established, the complexity has the potential to be significantly reduced, as the models will consist of rigid and standardized information and geometry.

Q3. Can the lifecycle cost data transfers be automated to prevent extracurricular tasks experienced by the DGNB auditor?

Given there is no direct link between Revit and LCCbyg, automation proposed here should reduce the effort needed to calculate lifecycle costs. In short, the transfer can be automated to a high degree. The transfer of information from Revit parameters can be fully grouped and organized using Dynamo-Python workflow. However, not without issues, to the least encountered using the proposed prototype. The more important question is what will be transferred. The proposed solution offers to transfer objects by type, not instance. Creating a Dynamo mapping to list instances is possible, but by the author, and backed by the auditor, deemed not worth the effort. The issue of grouping by instances is that every single case will require its data transfer. This can be not only quite tedious to replicate but may also be hard on the hardware. The solution may be split into parts to solve this problem. For example, when dealing with

surface finishes of floors within rooms, instantiating rooms is generally a good idea. In Revit, the project is built in a way that rooms are considered unique and may only use instance parameters. However, when using objects, such as desks or toilets, those are families of a specific type. Therefore, it is possible to use type parameters when quantifying them. This means that their base unit will relate to unit count, not area, or running meters. Furthermore, instead of instantiating every single object of precisely the same attributes, those can be grouped by type. If there is a need to create a unique case, the object type may be duplicated, and data may be added to facilitate quantity takeout correctly.

There are not many additional tasks that the auditor must perform to facilitate the transfer. Once the database is populated, and template costing and object data is available, filling in Revit models with already data-rich objects is simple. The auditors' tasks are to solely press a couple of “run” buttons and analyze the results, given the data is populated in the model, and the model is correctly made. All it comes down to is creating a reliable database of objects, which include LCC information and updating the costs of object cleaning if necessary.

Q4. Do the stakeholders agree that object-oriented approach generates value?

The critical observation is that each stakeholder values something different. The beneficiaries of the results generated by the workflow are the building owners, the architects, the auditors, the FM teams, the cleaning companies, the cleaning operators, and finally, the building users.

The building owners value having a DGNB certificate as the entire reason for it, is to ensure the building will be a profitably running asset over time. It is unclear whether the building owners would value knowing the cleaning costs, as no building owners were questioned, however, according to the architects, the building developers, those who build to sell, do not trouble themselves with finding out operational costs; however, the building owners who later own the property, such as the state - pay great attention to it. The author finds good examples to be hospitals and daycare institutions, hotels, airports and other buildings that have high operational costs and are often publicly owned.

The architects seem to value operational estimations, as it can be a tool for them to explain to clients why some fittings or materials may be a better option, despite the high initial purchase cost. Furthermore, having an automated LCC data transfer will allow checking for multiple configurations and offer a true data-driven design process. This, in turn, will likely create a longer-lasting design, and indirectly, the careful analysis of cleaning may reap in the prolonged economic lifespan of the building and its components.

The auditors, people directly responsible for the calculation process, will find automating the transfer improving their workflows. They should find using Dynamo relatively easy, and in the long run, use a direct plug-in with a user-friendly interface. Perhaps, in the future, the task will be automatic and incorporated in the software, or so easy to make that a general BIM modeler, or the architect, will be running the process along as the design progresses.

The key value for FM teams is information. Having the right information and the right tools to view the information effectively can make or break the FM personnel to use it with delight. As we see more benefits to the use of BIM models and the use of standards such as IFC becomes more widespread, there must be a standardized way of passing the costing design information forward to the FM teams. However, further research must be carried out to check to what extent could the initial early design cost calculation, benefit future use during operations. The author foresees the development of object-oriented cleaning software, which will benefit the cleaning operations and users. Having interviewed Dalux software consultant, integrating cleaning costs to the existing Dalux FM platform should be relatively straight forward, and indeed be valuable to the customers. This was confirmed by the FM BIM consultants at the state hospital in Aalborg. They wish that cleaning operations would be integrated with Dalux, as they do not want to use a separate system for cleaning personnel.

The author finds the hospital as a perfect future research facility to test the object-oriented cleaning during the operations phase, as the Aalborg hospital is owned publicly, the cleaning personnel is insourced, they are implementing Dalux FM software, and their BIM adoption is at a high level.

The potential benefits of placing costs on objects for cleaning companies, cleaners, and users may be thoroughly investigated in future research. A vision seen by the author sees a benefit in pay-per-object pricing and salary calculation. Such a technique will likely soften the frictions encountered by the cleaning operators by being inadequately paid to the effort required or the cleaning companies overpaying for jobs that require much less time. The same is true for overpaying for work where the employees are spending idle time, or clean objects too frequently.

Lastly, the users will also benefit from detailed information about the building they are situated in and a possibility to report issues as they arise directly to the cleaning personnel, skipping unnecessary attention of the help desk, administration, and operational managers in the process.

Q5. Can a prototype include the functionality of a working software product?

When it comes to functionality, the Dynamo prototype can well transfer any information that can be derived from the parametric model. The question is how much of the code will be needed to be re-developed in the case of an anomaly. The other question is whether there is a capable person nearby that can do it. Therefore, the prototype will always be somewhat inaccessible to the crowd of users, reliant on an intuitive user interface and a bug-free software solution.

Presentations and user testing had shown that the tool is versatile and flexible. It carries over data transfers for multiple types of families, including Revit built-in families. Furthermore, the Dynamo script distinguishes differences between area-type objects, unit type objects, and running meter objects. The author did not create a volume-based transfer, as none were deemed necessary for cleaning. However, it is possible, and applications are present, such as in gardening when exchanging plant soil.

The prototype lacks integration with IFC and other software. The prototype is also contained within the Autodesk environment; therefore, it cannot be considered an interoperable solution. The transfers still require partially manual operations, for example, due to the two-step data transfer process.

Furthermore, the databases needed to inform the model information are not there, and the links were not fully integrated with the scripts. More work will be needed to create a standard for calculating costs based on objects, and more testing will be needed to validate the anticipated prices.

11.1 Evaluation

Comparing an existing factor-based approach with the proposed object-oriented approach is difficult to delimit. According to W. Ulrich, (2005), “both the meaning and the validity of professional propositions always depend on boundary judgments as to what 'facts' (observation) and 'norms' are to be considered relevant" or not. It is referred to as systems thinking and it concerns an underlying issue of accuracy of data and approximations. Due to the lack of cleaning object cost data, the detail of the cost calculations method is difficult to properly assess and clearly state, whether it is better or worse than the factor-based approach.

Furthermore, as developing an interoperable, end-user software is often a considerable feat, a glitch-free, rigid, and holistic solution was deemed impracticable due to restricted time frames. Developing a rigid framework for object-oriented costing would require longitudinal research analyzing the time it takes to clean specific objects over many months, measuring various people with varying ages and abilities. The methods would then need to be cross-validated by multiple industry professionals before they can be deemed reliable.

As a result, a proposal was made for an approach for calculating the costs; however, the statistical data for actual figures do not exceed measurements during limited sample trial tests, thereby the costs shall be deemed as sample values. Missing the inclusion of entire categories in the existing approach will account for changes to price, regardless of the actual cost data. Accurate cost data will be down to the users to develop.

Due to the spread of analysis and research goals, sometimes the material may seem inconsistent, as some parts are detailed and some quite generic. The generic parts will require further research. Reasons for this approach are justified, as two approaches are merged, one IT-based, and the other discussion and function-based.

As a result, the LCC data transfer automation is paired with methods for creating cleaning calendars, calculating wages for future operational employees, and user feedback reporting systems. The latter considerations had been moved to the appendices, as to ensure a consistent flow of the main thesis content.

The limitation should strictly specify that the main application of this thesis is to create a prototype capable of transferring and analyzing cost data, but not necessarily providing a rigid, and bug-free solution. The prototype aims to prove its feasibility and maintain a stable reference point for further usability discussions with various stakeholders that may potentially find use in the tool.

The end-user application should be developed using a structured database such as MySQL or a Labelled Property Graph database (LPG) utilizing IFC data or other non-proprietary solutions, enabling direct mapping to a standard property set transfer. Furthermore, the software should be coded using an lccXML schema, flexible enough to accommodate schema updates and new software versions in the future. The transfer should also account for remaining lifecycle costs, including procurement, maintenance, and demolition.

The User Interface application must be developed, likely using a .NET or .net core framework. Furthermore, the application should have full documentation provided on a dedicated website and be connected with cost databases commonly used in Denmark, such as V&S Price books.

11.2 Future Development

The short term goal will be to figure a Python code capable of directly appending the LCCbyg XML schema, without using a rigid code solution. Ideally, such code would transfer data from all LCC stages, as opposed to cleaning only. This could be easily expanded to other LCC categories, following the principles of coding established in this report.

A connection with IFC transfer is deemed an essential subject for future research, as it is hard to expect FM teams to have access to the Revit models and update them accordingly. IFC will facilitate interoperability with the common FM platforms such as Dalux FM or Mdoc FM. For this to happen, an ontology must be developed.

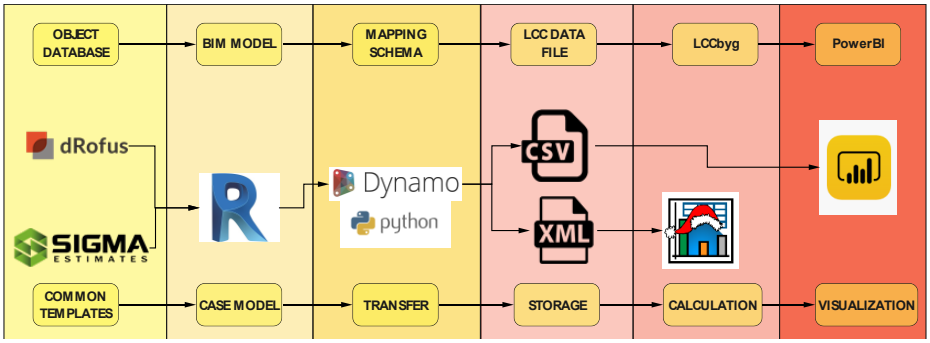
There is a range of potential benefits of object orientation described in A vision of key benefits of object-oriented LCC, appendix 14.13, and each benefit could be a research subject on its own. It would be a good idea to use the tool envisioned in this report to conduct data-driven researching and see if object-oriented costing optimization is worth the initial effort. Utilizing the application in a rigid form will require IT software engineers and further investigation.

During the writing of this thesis, a beta version of LCCbyg 3.0 came out; however, the data transfer is made to work for LCCbyg 2.2.52 version. The reason the transfer was not updated was that the documentation of the new lccXML schema was not available before the submission of this thesis. It will be a future task to update the schema to version 3.0 accordingly.

12 Conclusion

Object-oriented Lifecycle Costing has been researched before, for example, for bridge structures (Ugwu et al., 2005). Costing at object level is a popular method for calculating construction costs, widely used by software such as Vico Office. Due to the high cost of operational costs, there seem to be good reasons to calculate cleaning costs at an object level. There are tools available to quantify objects, and there are tools available to calculate the lifecycle cost. Furthermore, the stakeholders agree that having cost databases on upkeep costs, along with usable lifespans, would bring value when making design decisions.

What is missing, besides the operational object cost databases, is a method to quickly assess the overall building cost of operational maintenance during its lifespan. This gap is bridged in this report by providing an approach to quantifying, sorting, and mapping data to software capable of such calculations. Moreover, to ease the design decision making, a supplementary solution is proposed and prototyped to extract information to visualization software. The contribution of this thesis is the workflow process necessary to optimize operational costs using a Revit model, Dynamo and Python scripting, and a Lifecycle Costing tool LCCbyg. Supplementary processes of database attachment and data visualization, along with future benefits, are investigated. The diagram summarizes the process.



The prototype emerged as a result of the available software and user environment. The chosen software solutions were either commonly used, or being implemented in the Danish AEC industry. The resulting workflow has emerged from the industry needs and personal insights in both BIM and cleaning operations.

The workflow, along with the prototype proposed, is capable of transferring LCC data to the LCCbyg from the Revit model. The process of data transfer can be obtained in under one minute per design option, so long the data is correctly assigned from the template database. The coding of data transfer is flexible and can carry information about a variety of object types, grouping them accordingly by types or instances, and by measurement units.

The transfer is a two-step process requiring a Python script to run outside of the Dynamo environment, which can be hopefully fixed in the future. The Python script is a working proof of a concept, and a long term solution would be to use the full LCCbyg XML schema. Therefore, it is worth noting the next step in the development may take a different shape, yet schematically, be derived from the proposed approach. A possible further development may well encapsulate other areas of lifecycle costing, following the same object-oriented principles as a basis for calculation and data transfer applications.

The function-based approach proposed by the existing transfer is lacking in detail and poses a threat that the real costs of operational maintenance are much more significant than initially considered. Having in mind an object-based approach, design decisions that consider operations costs can be made before construction. The results from such analyses can showcase how changes as small as 5DKK can have

significant impacts on the lifecycle costs. The economies of scale, resulting from dense frequency and vast object areas and counts, add up to astronomical sums.

The case study shows that a surface material change can alter the LCC cost by DKK3.25 million, given 129 desks require more attention over 50 years. The attention as a result of this is calculated by a 5DKK cost increase in the cleaning cost of a single desk, a single time. The stakeholder analysis and user testing verify the general interest expressed by the stakeholders from the informative results generated by the proposed tool and the case study used.

The concluding results can be analyzed further using data analytics tools such as PowerBI. Supplementary scripts were developed, which present cost data for future stakeholders, on a yearly basis. The graph from Figure 10.8 clearly indicates that the relationship between a room area and its cleaning cost is not linear. Visible discrepancies occur at an object level, and the detail is clearly necessary to evaluate the viability of one design option over another properly.

The architects may have found an excellent way to justify the higher upfront cost associated with high-quality materials, thus prolonging the cycle of use and contributing to the sustainability of building use. Having in mind finding an easy way to calculate such detailed costs, the architect should be encouraged to use the tool and be able to share its benefits contagiously with related stakeholders. However, until object costing database is made commonly available, the architects have no choice but to stick to the standard approach proposed by the DGNB, to calculate LCC based on aerial costs.

The hospital FM team had found the subject of cleaning very relevant. The interviewees concerned themselves with a cleaning strategy for the hospital building. Having an object-oriented approach for cleaning costing would optimize design choices, cleaning strategies, and staff training to a great extent. Due to the sheer size of a hospital and the governance of its cleaning operations, due to insourcing, the environment would be ideal for further testing of this solution on a real case example.

The adoption of Dalux FM as a Facility Management Software, along with testable time allocations for object cleaning, will allow for establishing fully-functional workflows. Evaluating cleaning time is necessary for object costing while appending models with cleaning data will facilitate quantity takeout and cost optimization.

Furthermore, the information attached can inform FM systems necessary to run the operations scheduling and prompt user feedback smoothly. Further longitudinal research and practical implementation are deemed a desirable method to validate the thesis proposal.

The future approaches were investigated to pave the way for a set of user-friendly applications for lifecycle costing, tendering, training, and communication. The future development will require significant programming efforts, and the stakeholder feedback indicates that those who will have access to this information will reap benefits outstanding the costs of the initial setup.

One potential application admired by the cleaning companies and the operational staff themselves was to develop object-oriented colored calendars, where the cleaning operations team would know who is cleaning what, and when. Furthermore, estimating the time could well inform the calculation proposals for the tender bids and inform wages.

It will be down to functional integration of the proposed workflow with an existing FM tool to fully reap the benefits of object-orientation for cleaning and having interviewed FM software companies, and it seems like a reasonably straight forward task to append existing models with the proposed parameters for object-oriented costing. The hospitals may well be the first potential integrated users, both benefiting from informed design decisions, and operational benefits the object orientation brings.

13 Bibliography

Aish, R. (2017). DesignScript User Manual, (October 2016). Retrieved from https://www.researchgate.net/publication/320346998

Akcamete, A., Akinci, B., & Garrett, J. H. (2019). Potential utilization of building information models for planning maintenance activities. *EG-ICE 2010 - 17th International Workshop on Intelligent Computing in Engineering*, (January).

Andydandy74. (2019). andydandy74/ClockworkForDynamo: A collection of 400+ custom nodes for the Dynamo visual programming environment. Retrieved December 4, 2019, from https://github.com/andydandy74/ClockworkForDynamo

Autodesk. (2018). Revit IFC manual. Detailed instructions for handling IFC files, 1–52. Retrieved from www.buildingsmart.org/compliance/certified-

Barazzetti, L., Banfi, F., Brumana, R., & Previtali, M. (2015). Creation of Parametric BIM Objects from Point Clouds Using Nurbs. *The Photogrammetric Record*, *30*(152), 339–362. https://doi.org/10.1111/phor.12122

Bear, A. (2019). Amoursol/dynamoPython: Python Modules for Dynamo. Retrieved December 4, 2019, from https://github.com/Amoursol/dynamoPython

Beyer, H., & Holtzblatt, K. (1998). *Contextual design : defining customer-centered systems*. Morgan Kaufmann.

BIM7AA. (2018). *Bim7Aa Typekodning*. Europaplads 2, 11. sal, 8000 Aarhus C. Retrieved from BIM7AA.dk

BIPS. (2015). CCS. Retrieved from https://bips.dk/værktøjsemne/ccs#0

Building smart. (2010). Information Delivery Manual Guide to Components and Development Methods. *BuildingSMART*, 1–84. Retrieved from http://idm.buildingsmart.comhttp//idm.buildingsmart.comashttp://idm.buildingsmart.com

Christian Koch. (2018). *BIM Handbook*. (J. B. E. König Markus, André Borrmann, Christian Koch, Ed.). Springer International Publishing AG.

Conradie, D. C. U., & Roux, E. (2008). QUALITY MANAGEMENT IN CONSTRUCTION PROJECT DESIGN AND MANAGEMENT. *Gazette*, 16–18. Retrieved from https://www.researchgate.net/publication/30511316_Quality_management_in_construction_project_design_and_management/figures

Dansk standard DS/INSTA 800 Rengøringskvalitet-System til fastlaeggelse og bedømmelse af rengøringskvalitet Cleaning quality-System for establishing and assessing cleaning quality. (2010). Retrieved from https://webshop.ds.dk/da-dk/standard/ds-insta-8002011

DGNB.de. (2019). DGNB Lifecycle cost. Retrieved September 29, 2019, from https://www.dgnb-system.de/en/system/version2018/criteria/life-cycle-cost/

Dieter Vermuelen. (2017). (63) Fire Exit Risk Assessment - Automated creation of evacuation - YouTube. Retrieved from https://www.youtube.com/watch?v=nHKrxw-FsRE&t=143s

DiKon. (2017). DiKon Building part specification- for selected building parts in building models. Retrieved from https://www.dikon.info/en/publications/

Eastman, C., Teicholz, P., Sacks, R., & Liston, K. (2018). *BIM Handbook*. *BIM Handbook*. Hoboken, NJ, USA: John Wiley & Sons, Inc. https://doi.org/10.1002/9780470261309

Goldstein, B. P., Herbøl, M., & Figueroa, M. J. (2013). Gaps in tools assessing the energy implications

of renovation versus rebuilding decisions. *Current Opinion in Environmental Sustainability*, *5*(2), 244–250. https://doi.org/10.1016/j.cosust.2013.03.005

Green Building Council. (2013). *An introduction to DGNB : Ensure the quality of your sustainable buildings in planning, construction, and operation. The DGNB system helps you get there*. Retrieved from https://www.dk-gbc.dk/media/2292/dgnb_dk-gbc_oct_2012.pdf

Green Building Council Denmark. (2017). *DGNB System Denmark manual for kontorbygninger 2016* (Vol. 1.1). Retrieved from http://www.dk-gbc.dk

Guillen, A. J., Crespo, A., Gómez, J., González-Prida, V., Kobbacy, K., & Shariff, S. (2016). Building Information Modeling as Assest Management Tool. *IFAC-PapersOnLine*, *49*(28), 191–196. https://doi.org/10.1016/j.ifacol.2016.11.033

Harold, E. R., & Means, W. S. (2001). *XML in a Nutshell : A Desktop Quick Reference (Nutshell Handbook)*. Retrieved from https://kbdk-aub.primo.exlibrisgroup.com/discovery/fulldisplay?docid=alma9920672192905762&context=L&vid=45KBDK_AUB:AUB&lang=da&search_scope=MyInst_and_CI&adaptor=Local Search Engine&tab=Everything&query=any,contains,XML in a Nutshell&offset=0

Haugbølle, K., Scheutz, P., Saridaki, M., & Sørensen, N. L. (2017). *Installation Guide and User's Guide LCCbyg version 2.2*. Retrieved from https://sbi.dk/Pages/Installation-Guide-and-User-s-Guide-LCCbyg-version-2-2.aspx

Huizenga, C., Hui, Z., Duan, T., & Arens, E. (2001). An improved multinode model of human physiology and thermal comfort. *Building and Environment*, *36*(July), 691–699. https://doi.org/10.1111/j.1600-0668.2011.00745.x

IBM. (2019). Creating maps from COBie attributes. Retrieved November 27, 2019, from https://www.ibm.com/support/knowledgecenter/en/SSLKT6_7.6.1/com.ibm.mbs.doc/bim/t_map_cobie_attrib.html

ISO. (2013). ISO 16739. (2013). Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries.

ISO 16739-1:2018. (2018). ISO 16739-1:2018 Preview Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries -- Part 1: Data schema. *International Organization for Standardization*, *2018*, 70303. Retrieved from https://www.iso.org/standard/70303.html

Johnson, L. (2019). Bakery Dynamo Package. Retrieved December 4, 2019, from https://dynamopackages.com/#

Kaewunruen, S., & Lian, Q. (2019). Digital twin aided sustainability-based lifecycle management for railway turnout systems. *Journal of Cleaner Production*, *228*, 1537–1551. https://doi.org/10.1016/j.jclepro.2019.04.156

Kang, T.-W., & Choi, H.-S. (2015). BIM perspective definition metadata for interworking facility management data. https://doi.org/10.1016/j.aei.2015.09.004

Kensek, K. M. (2014). Integration of Environmental Sensors with BIM: Case studies using Arduino, Dynamo, and the Revit API. *Informes de La Construccion*, *66*(536). https://doi.org/10.3989/ic.13.151

Kim, K., Kim, H., Kim, W., Kim, C., Kim, J., & Yu, J. (2017). Integration of ifc objects and facility management work information using Semantic Web. https://doi.org/10.1016/j.autcon.2017.12.019

Kirk, S. J., & Dell'Isola, A. J. (1995). *Lifecycle Costing for Design Professionals*. *Proceedings of Joint International Symposium of CIB Working Commissions W55/W65/W107 in Knowledge*

Construction. Retrieved from http://www.sciepub.com/reference/174753

Kiviniemi, A., Tarandi, V., Karlshøj, J., Bell, H., & Karud, O. J. (2008). Review of the Development and Implementation of IFC compatible BIM Executive Summary. *ERA Build*, 1–2.

Lawrence, M., Pottinger, R., Staub-French, S., & Nepal, M. P. (2014). Creating flexible mappings between Building Information Models and cost information. https://doi.org/10.1016/j.autcon.2014.05.006

Lichtenvort, K., Rebitzer, G., Huppes, G., Ciroth, A., Seuring, S., Schmidt, W. P., … Hunkeler, D. (2008). Introduction history of lifecycle costing, its categorization, and its basic framework. In *Environmental Lifecycle Costing* (pp. 1–16). CRC Press. https://doi.org/10.1201/9781420054736

LunchBox – PROVING GROUND. (2019). Retrieved December 4, 2019, from https://provingground.io/tools/lunchbox/

Mccrone, C. (2010). Dynamo Language Manual, 1–57. Retrieved from https://dynamobim.org/wp-content/uploads/forum-assets/colin-mccroneautodesk-com/07/10/Dynamo_language_guide_version_1.pdf

Mendes De Farias, T., Roxin, A., & Nicolle, C. (2018). A rule-based methodology to extract building model views. https://doi.org/10.1016/j.autcon.2018.03.035

Molio. (2019). Molio prisdata : Renovering & drift. Retrieved November 29, 2019, from https://kbdk-aub.primo.exlibrisgroup.com/discovery/fulldisplay?docid=alma9920661403005762&context=L&vid=45KBDK_AUB:AUB&lang=da&search_scope=MyInst_and_CI&adaptor=Local Search Engine&tab=Everything&query=any,contains,molio&offset=0

Olatunji, O. A., & Sher, W. D. (2010). The Applications of Building Information Modelling in Facilities Management (pp. 239–253). https://doi.org/10.4018/978-1-60566-928-1.ch011

Patacas, J., Dawood, N., Vukovic, V., & Kassem, M. (2015). *BIM for facilities management: evaluating BIM standards in asset register creation and service life*. *Journal of Information Technology in Construction (ITcon)* (Vol. 20). Retrieved from http://www.itcon.org/2015/20

Piaskowski, AK, Petersons, R, Wyke, SCS, Petrova, EA & Svidt, K. (2019). Automation of data transfer between a BIM model and an environmental quality assessment application. *Cib Proceedings*. Retrieved from https://www.forskningsdatabasen.dk/en/catalog/2447076400

Pierson, J. (2019). johnpierson/RhythmForDynamo: A collection of nodes for use in Dynamo with Revit. Retrieved December 4, 2019, from https://github.com/johnpierson/RhythmForDynamo

Rosendahls. Strategi for digitalt byggeri (2019). Retrieved from https://www.trm.dk/da/publikationer/2019/strategi-for-digitalt-byggeri

Sacks, R., Eastman, C., Lee, G., & Teicholz, P. (2018). *BIM Handbook Rafael Sacks* (Vol. 25). https://doi.org/10.1016/S0926-5805(02)00090-0

Samarbetskomitén för Byggnadsfrågor. (2012). SfB-systemet, 316–320. Retrieved from https://byg-erfa.dk/en/node/5278

Saridaki, Maria;, & Psarra, M. (2017). Integration of LCC into BIM concept, (DTU Byg, shared internally.), 137.

Saridaki, Maria, Psarra, M., & Haugbølle, K. (2019). *IMPLEMENTING LIFE-CYCLE COSTING: DATA INTEGRATION BETWEEN DESIGN MODELS AND COST CALCULATIONS*. *Journal of Information Technology in Construction (ITcon)* (Vol. 24).

SBi. (2019). Retrieved September 29, 2019, from https://www.sbi.aau.dk/om/

Shan, X., Melina, A. N., & Yang, E. H. (2018). Impact of indoor environmental quality on students'

wellbeing and performance in educational building through lifecycle costing perspective. *Journal of Cleaner Production*, *204*, 298–309. https://doi.org/10.1016/j.jclepro.2018.09.002

Smith, P. (2014). BIM implementation - Global strategies. In *Procedia Engineering* (Vol. 85, pp. 482–492). https://doi.org/10.1016/j.proeng.2014.10.575

Sobon, K. (2019). Dynamo | archi-lab. Retrieved December 4, 2019, from https://archi-lab.net/category/dynamo/

Sullivan, G. P., Pugh, R., Melendez, A. P., & Hunt, W. D. (2010). Operations & Maintenance Best Practices: A Guide to Achieving Operational Efficiency. *Federal Energy Management Program*, (August 2010), 321. https://doi.org/10.2172/1034595

thenbs.com. (2016). What is COBie? | NBS. Retrieved November 27, 2019, from https://www.thenbs.com/knowledge/what-is-cobie

Thiebat, F. (2019). *Lifecycle Design, An Experimental Tool for Designers*. Retrieved from http://www.springer.com/series/13890

Toth, B., Janssen, P., Stouffs, R., Chaszar, A., & Boeykens, S. (2012). Custom Digital Workflows: A New Framework for Design Analysis Integration. *International Journal of Architectural Computing*, *10*(4), 481–499. https://doi.org/10.1260/1478-0771.10.4.481

Ugwu, O. O., Kumaraswamy, M. M., Kung, F., & Ng, S. T. (2005). Object-oriented framework for durability assessment and lifecycle costing of highway bridges. *Automation in Construction*, *14*(5), 611–632. https://doi.org/10.1016/j.autcon.2005.01.002

Ulrich, W. (2005). A Mini-Primer of Boundary Critique. In *The Informed Student Guide to Management Science* (p. 41f.).

Vigovskaya, A., Aleksandrova, O., & Bulgakov, B. (2017). Lifecycle Assessment (LCA) in building materials industry. In *MATEC Web of Conferences* (Vol. 106). https://doi.org/10.1051/matecconf/201710608059

What is Dynamo? | The Dynamo Primer. (2018). Retrieved May 8, 2019, from https://primer.dynamobim.org/01_Introduction/1-2_what_is_dynamo.html

Wix, J., & Karlshøj, J. (2010). Information Delivery Manual Guide to Components and Development Methods. *BuildingSMART*, 84. Retrieved from http://idm.buildingsmart.comhttp//idm.buildingsmart.comashttp://idm.buildingsmart.com

XML. (2019). Retrieved September 29, 2019, from https://en.wikipedia.org/wiki/XML

XML Schema Tutorial. (2019). Retrieved November 29, 2019, from https://www.w3schools.com/xml/schema_intro.asp

Yu, K., Froese, T., & Grobler, F. (1998). International Alliance for Interoperability: IFCs. In *Congress on Computing in Civil Engineering, Proceedings* (pp. 395–406). Retrieved from https://www.google.com/search?ei=Ll-_XNufJeflrgSn_pnIAw&q=international+alliance+for+interoperability&oq=International+alliance+for+interop&gs_l=psy-ab.1.0.0j0i22i30.3520.8687..10023...1.0..0.100.2882.34j1.....0....1..gws-wiz.....6..0i71j35i39j0i67.wJ4_M