HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr

Carl Philipp Fürstenberg

Systematisierung von Bauteilinformationen zur planungsbegleitenden Betriebskostenoptimierung von Immobilien nach der BIM-Arbeitsweise

Chemnitz, 2023

Fakultät Wirtschaftsingenieurswesen

BACHELORARBEIT

Systematisierung von Bauteilinformationen zur planungsbegleitenden Betriebskostenoptimierung von Immobilien nach der BIM-Arbeitsweise

Autor:
Herr

Carl Philipp Fürstenberg

Studiengang:
Immobilien- und Facilities-Management

Seminargruppe:
FM18w2-B

Erstprüfer:
Prof. Dr.-Ing Jörg Mehlis

Zweitprüfer:
Dipl.-Ing. Ronny Windisch

Einreichung:
Mittweida, 02. März. 2023

Verteidigung/Bewertung:
Mittweida, 2023

Faculty of Industrial Engineering

BACHELORTHESIS

Systematization of building element information for optimized estimation of operational cost for real estate projects according to the BIM approach

author:

Mr.

Carl Philipp Fürstenberg

course of studies:

Real Estate- & Facilities Management

seminar group:

FM18w2-B

first examiner:

Prof. Dr.-Ing Jörg Mehlis

second examiner:

Dipl.-Ing. Ronny Windisch

submission:

Mittweida, 02. March. 2023

defence/ evaluation:

Mittweida, 2023

Bibliografische Beschreibung:

Fürstenberg, Carl Philipp:

Systematisierung von Bauteilinformationen zur planungsbegleitenden Betriebskostenoptimierung von Immobilien nach der BIM-Arbeitsweise - 2023. - 11, 55, 4 N S.

Mittweida, Hochschule Mittweida, Fakultät Wirtschaftsingenieurswesen, Bachelorarbeit, 2023

Referat:

Die vorliegende Arbeit befasst sich mit dem Aufbau eines Arbeitsprozesses, zur modellbasierten Kostenschätzung der Nutzungskosten einer Immobilie während der Planung. Mit der Zuordnung der Kostenstellen zu deren im Modell vorhandenen Verursachern, sollen die Kosten teilautomatisiert ermittelt werden.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BIM	**B**uilding **I**nformation **M**odelling
CAFM	**C**omputer-**A**ided **F**acility **M**anagement
DIN	**D**eutsches **I**nstitut für **N**ormung
EN	**E**uropäische **N**orm
HOAI	**H**onorar**o**rdnung für **A**rchitekten und **I**ngenieure
IFC	**I**ndustry **F**ormation **C**lasses
ISO	**I**nternational **O**rganization for **S**tandardization
KG	**K**osten**g**ruppe
LPH	**L**eistungs**ph**ase
LOD	**L**evel **o**f **D**evelopement
LOI	**L**evel **o**f **I**nformation
LOIN	**L**evel **o**f **I**nformation **N**eed
NRF	**N**etto-**R**aumfläche
NUF/NF	**N**utzungs**f**läche
TF	**T**echnik**f**läche
TGA	**T**echnische **G**ebäude**a**usstattung
VF	**V**erkehrs**f**läche

Vorwort

Mit der vorliegenden Abschlussarbeit schließe ich meine vergleichweise lange Studienphase zum Erlangen des Bachelor-Titels, welche durch verschiedene Berufsfelder führte, ab. In diesem Zuge möchte ich mich insbesondere bei meinen Kolleginnen, Kollegen und Vorgesetzten innerhalb der **iproplan®** Planungsgesellschaft mbH bedanken, welche mir stets Vertrauen und Unterstützung in meiner Schaffensarbeit schenkten und mir die Möglichkeit bieten, das während der Studienzeit erlangte Wissen unabhängig des Branchenfeldes anzuwenden. Ich freue mich sehr auf die weitere Zusammenarbeit.

Besonderer Dank gilt außerdem meiner Familie und meinen Freunden, welche mich durch die Höhen und Tiefen meiner Ausbildung begleitet haben und mir stets unterstützend zu Seite standen.

Chemnitz, den 28.02.2023 Carl Philipp Fürstenberg

1. Einleitung

1.1 Motivation

Durch die zunehmende Verbreitung der Anwendung von BIM (Building Information Modelling) werden die FM-bezogenen BIM-Anwendungsfälle in der Bauausführungsplanung von Immobilien immer bedeutender. Insbesondere dem Anwendungsfall der modellbasierten Kosten- & Mengenermittlung kommt besondere Aufmerksamkeit zu.

Aktuell werden Mengenansätze im Verlaufe der Entwurfsplanung und vor der Vergabe aus dem digitalen Gebäudemodell erhoben und entsprechenden Kenngrößen zugeordnet. Dabei handelt es sich in der Regel ausschließlich um die Investitions- bzw. Baukosten. Die Kosten des Gebäudebetriebes finden zu diesem Zeitpunkt nur ungenügend Berücksichtigung. Dabei beläuft sich der Kostenanteil der Gebäudenutzungsphase Erfahrungswerten nach auf ein Vielfaches der Investitionskosten[1], wodurch sich ein erhebliches Entwicklungspotenzial abbildet.

1.2 Zielstellung

Mit BIM-Technologien besteht die Möglichkeit, eine verbesserte Bewertung der Betriebskosten bereits während der Planungsphase zu erreichen. Diese können so bereits optimiert werden, während kostenrelevante Stellen noch flexibel und kostengünstig anpassbar sind.

Dazu stellen insbesondere die sinnvolle Hinterlegung von relevanten Daten an den digitalen „Bauteil-Zwillingen“, deren eindeutige Zuordnung zu den realen Objekten und zu den Kostenstellen des Gebäudebetriebes eine Herausforderung dar. Eine präzise Aussage über die Nutzungskosten eines Gebäudes ist, aufgrund dessen langer Lebenszeit, eine Herausforderung und es kann größtenteils nur mit Richt- und Erfahrungswerten gearbeitet werden.

[1] (Prof. Dipl.-Ing. Uwe Rotermund M. Eng. TM, 2016), S.7

Es bestehen bereits diverse Ansätze zur Kostenermittlung anhand von digitalen Planungsdaten. Für die modellbasierte Ermittlung der Baukosten und die Erstellung erforderlicher Leistungsverzeichnisse bestehen bereits Lösungen wie die Software Bechmann BIM/AVA[2,3] oder Thinkproject DESITE BIM[4], bedürfen jedoch einem hohen Anpassungsgrad je Projekt.

Mit der Software SPARTACUS FM[5] wird neben umfangreichen CAFM-Funktionen bereits eine Lösung zur Kostenermittlung & -kontrolle geboten. Hierbei können Planungsdaten eingelesen und mit kostenrelevanten Daten gespeist und ausgewertet werden. Dazu stehen unter anderem verschiedenste Module wie „Instandhaltungsmanagement“, „Flächenmanagement“, „Budgetmanagement und Kostenverfolgung“ zur Verfügung. Diese Software findet jedoch erst während der Nutzungsphase Anwendung und bietet bezüglich der Modellauswertung wenig individuelle Anpassungsmöglichkeiten, da lediglich 2D Pläne zur Verortung von Bauteilverknüpfungen herangezogen werden und verarbeitete Daten bis auf Raumflächen nicht automatisiert bezogen werden.

Während der Planungsphase kann die Software Bechmann BIM zur Ermittlung von Betriebskosten dienen. Bechmann BIM dient zur objektorientierten Ermittlung von Baukosten und deren Überführung in Positionen des Leistungsverzeichnisses. Mit der Software besteht die Möglichkeit, auch betriebsrelevante Kosten anhand von hinterlegten Attributwerten zu ermitteln. Somit können sogenannte Kostenelemente mit sämtlichen im Modell vorhandenen Bauteilen verknüpft und mit den hinterlegten Parameterwerten diverse Kostenstellen individuell angelegt und berechnet werden. Da die in Bechmann BIM angewandte Arbeitsweise zum Zweck der Mengenermittlung und folgenden Erstellung von Leistungsverzeichnissen per Überführung zu Bechmann AVA gedacht ist, ist unklar, ob eine Berechnung der Nutzungskosten ohne weiteres möglich ist. Die Möglichkeiten der Software müssen für eine entsprechende Anwendung erforscht werden. Aufgrund der einhergehenden Lizenzkosten

[2] (BECHMANN, 2022)

[3] (BECHMANN, 2022)

[4] (Thinkproject, 2022)

[5] (N+P Informationssysteme GmbH, 2022)

wird jedoch davon abgesehen, da der Zielstellung dieser Arbeit eine möglichst kostengünstige Lösung, ganz im Sinne des „Open BIM“-Gedanken, innewohnt.

Der Prozess zur Ermittlung der Betriebskosten bietet mit der Anwendung von BIM – und somit einer modellbasierten Ermittlung – ein hohes Optimierungspotenzial; so kann dieser Prozess, wie es mit anderen Schwerpunkten der Anwendung von BIM bereits der Fall ist, parallel zur Planung eingesetzt werden, da kostenrelevante Änderungen während dieser Phase noch flexibel und kostengünstig anzupassen sind.

Festzustellen und zu strukturieren ist dazu, welche Mengenansätze entsprechend der Leistungsphase im Planungsmodell vorhanden sind und verwendet werden können, welche anzulegen oder abzuleiten sind und welche nicht aus dem Modell bezogen werden können.

Auf dieser Grundlage soll ein Prozess zur teil-automatisierten Abschätzung der Betriebskosten anhand eines Gebäudeplanungsmodells erarbeitet werden, welcher die voraussichtlichen Kosten bei aktuellem Planungsstand prognostiziert.

Mit der Entwicklung dieses Prozesses kann die Relevanz und Ausbaufähigkeit der Anwendung von BIM im Zuge der Digitalisierung und weiteren Automatisierung wichtiger Planungsprozesse unterstrichen und ausgebaut werden.

1.3 Gang der Untersuchung

Die Untersuchung ist in **vier Hauptteile** untergliedert.

Der **erste** Teil befasst sich mit den grundlegenden Informationen, welche zum Verständnis und zur Durchführung der Untersuchung nötig sind. Es wird auf die Grundbegriffe des Building Information Modelling und die für den Prozess hinzugezogenen normativen Dokumente und Richtlinien zur Betriebsphase von Immobilien eingegangen.

Der **zweite** Teil systematisiert gewonnene Informationen und dient zur Aufbereitung dieser, um den Voraussetzungen und Anforderungen der Untersuchung zu entsprechen und die Möglichkeit der Weiterverarbeitung zu gewährleisten. Dies geschieht durch die Entwicklung und Identifizierung konkreter Arbeitsprozessschritte.

Im **dritten** Teil wird die beispielhafte Durchführung eines Berechnungsprozesses behandelt. Die im zweiten Teil erarbeiteten Systematisierungen und Informationen werden zur Lösungsfindung herangezogen und an einem beispielhaften Planungsprojekt in Form des entwickelten Arbeitsprozesses umgesetzt.

Der **vierte** Teil schließt die Untersuchung in Form einer Auswertung und einer Potenzialabschätzung ab.

2. Grundlagen

2.1 Building Information Modelling

„Building Information Modeling ist die digitale Darstellung physischer und funktioneller Merkmale einer Anlage und schafft durch einen gemeinsam nutzbaren Pool relevanter Daten eine zuverlässige Entscheidungsgrundlage während des gesamten Lebenszyklus des Bauwerks, von der frühesten Idee bis hin zum Rückbau.“[6]

Das Building Information Modelling (zu Deutsch: Bauwerksinformationsmodellierung) – kurz BIM – gewinnt mit dem Informationszeitalter fortwährend an Bedeutung. Grund dafür ist die Zielstellung, eine effizientere Arbeitsweise der Beteiligten und Handhabung von Informationen über den gesamten Lebenszyklus einer Immobilie zu verwirklichen. Grundsätzlich wird von folgenden übergeordneten Zielen gesprochen:

- Verbesserung der Kommunikation und Schnittstellenkoordination
- "Erhöhung der Planungssicherheit, insbesondere in Form gesteigerter Termin- und Kostensicherheit"
- Erhöhung der Transparenz (Nachverfolgbarkeit von Entscheidungen und Konsequenzen sowie von entstandenen Kosten)
- Minimierung von Risiken
- Effizienzgewinn durch Verwendung des „As-Built“-Modells für den Betrieb und nachgelagerte Arbeiten[7]

Hervorzuheben ist, dass die BIM-Arbeitsweise nicht als alleinstehender, optionaler Prozess zu behandeln ist, sondern standardmäßig in allen immobilienbezogenen Prozessen der Planung und des Betriebes Anwendung findet.

Namensgebend steht das Erzeugen und Zentralisieren von Informationen im Mittelpunkt des Prozesses. Stichworte sind hierbei Informationsbeschaffung, -erzeugung, -auswertung und Dokumentation.

---

[6] (Handwerkskammer Münster, 2023)

[7] Vgl. (ARGE BIM4INFRA, 2017) S. 4

2.1.1 BIM-Anwendungsfälle

Einen ersten Anhaltspunkt, für welche Zwecke BIM angewendet werden kann, bieten die sogenannten BIM-Anwendungsfälle, welche von der Arbeitsgemeinschaft bim4infra erarbeitet wurden:

Tabelle 1, Übersicht der BIM-Anwendungsfälle

Bestandserfassung	Planungsvariantenuntersuchung
Visualisierungen	Bemessung und Nachweisführung
Koordination der Fachgewerke	Fortschrittskontrolle der Planung
Erstellung von Entwurfs- und Genehmigungs-plänen	Arbeits- und Gesundheitsschutz: Planung und Prüfung
Planungsfreigabe	Kostenschätzung und Kostenberechnung
Leistungsverzeichnis, Ausschreibung, Vergabe	Terminplanung der Ausführung
Logistikplanung	Erstellung von Ausführungsplänen
Baufortschrittskontrolle	Änderungsmanagement bei Planungsänderungen
Abrechnung von Bauleistungen	Mängelmanagement
Bauwerksdokumentation	Nutzung für Betrieb und Erhaltung

Zu erwähnen ist, dass die Möglichkeit der Umsetzung aller Anwendungsfälle generell für sämtliche Projekte besteht, jedoch projektspezifisch, entsprechend dem Verhältnis von Aufwand und Gewinn abzuwägen ist, ob die spezifische Anwendung gemäß den Anforderungen tatsächlich sinnvoll ist. Der Umsetzungsumfang von Anwendungsfällen wird in den Auftraggeber-Informationsanforderungen (AIA) und dem BIM-Abwicklungsplan (BAP) näher definiert.[8]

Für die Durchführung dieser Untersuchung ist der Anwendungsfall Kostenschätzung und Kostenberechnung relevant. Während BIM4infra[9] diesen Anwendungsfall zur Leistungsphase 2-3 einordnet, wird in dieser Ausarbeitung untersucht werden, inwiefern sich ein Anwendungspotential während der weiteren Planungsphase abbildet, um eine potenziell höhere Präzision der Kostenaussage zu den Nutzungskosten zu erreichen.

[8]Vgl. (ARGE BIM4INFRA2020, 2019)

[9] (ARGE BIM4INFRA2020, 2019)

	Nr	Anwendungsfälle	1	2	3	4	5	6	7	8	9	Betrieb
Bestandserfassung												
	AWF 1	Bestandserfassung										
Planung												
	AWF 2	Planungsvariantenuntersuchung										
	AWF 3	Visualisierungen										
	AWF 4	Bemessung und Nachweisführung										
	AWF 5	Koordination der Fachgewerke										
	AWF 6	Fortschrittkontrolle der Planung										
	AWF 7	Erstellung von Entwurfs- und Genehmigungsplänen										
	AWF 8	Arbeits- und Gesundheitsschutz: Planung und Prüfung										
	AWF 10	Kostenschätzung und Kostenberechnung										

Abbildung 1, Einordnung der Kostenberechnung in die Leistungsphasen der HOAI [10]

Anwendungsfall: Kostenschätzung und Kostenberechnung

Für gewöhnlich werden unter Anwendung dieses Falles Mengen- und Kostenansätze bezüglich der Gebäudeerstellung erhoben, jedoch keine Aussagen über die Kosten des Gebäudebetriebs getroffen. Grund dafür ist unter anderen, dass Investoren und Bauherren in seltenen Fällen in der Rolle des Betreibers zu finden sind, sondern Investitionsobjekte zumeist zeitnah und gewinnbringend veräußern. Im Hinblick auf einen tendenziell steigenden Anspruch an Nachhaltigkeit stellt sich jedoch die Frage der Umsetzbarkeit einer Kostenschätzung – oder gar -ermittlung – über den Betrieb eines Gebäudes. Die betrieblichen Kosten einer Immobilie sollten sich als äußerst relevant herausstellen, da sich deren Umfang, wie eingangs erwähnt, statistisch gesehen bereits nach wenigen Jahren des Betriebes über die Herstellungskosten hinaus entwickeln. Des Weiteren sollte die Inbezugnahme potenzieller Mieteinnahmen auf Grundlage der zu erwartenden Kosten für Teilhaber interessant sein, um die Gesamtwirtschaftlichkeit des Objektes bereits während der Planung optimieren zu können. Welche Grundlagen und Informationen dazu notwendig sind, wird im Kapitel 4 – Voraussetzungen zur Kostenanalyse behandelt.

[10] (ARGE BIM4INFRA2020, 2019) S. 8

2.1.2 Begrifflichkeiten

2.1.2.1 AIA – Auftraggeber-Informationsanforderungen

Das Dokument der Auftraggeber-Informationsanforderungen definiert „[…] vornehmlich informationsbezogene, inhaltliche Anforderungen des Auftraggebers an die digitale Abwicklung von Bauprojekten.“.[11] Und gilt somit als Grundlage für die Bestimmungen der Zusammenarbeit zwischen Auftraggeber und Auftragnehmern unter Anwendung von BIM. Als Teil der Ausschreibung liegt der Fokus hier insbesondere auf den BIM-Anwendungsfällen, welche einen Überblick über den Umfang der geforderten BIM-Leistungen darstellen.

Das AIA ist im Allgemeinen an die Auftragnehmer gerichtet und wird vom Auftraggeber erstellt, kann und wird jedoch im Regelfall – wie auch der BIM-Abwicklungsplan – in Kooperation fortgeschrieben und angepasst.

2.1.2.2 BAP – BIM-Abwicklungsplan

Aufbauend auf den AIA, welche die Anforderungen seitens des Auftraggebers deklarieren, werden im BIM-Abwicklungsplan konkrete Umsetzungsschritte festgehalten, um den Vorgaben des Auftraggebers zu entsprechen.[12]

Vereinfacht können die Dokumente mit den Fragen „**Was** ist gefordert?“ (AIA) und „**Wie** werden die Forderungen umgesetzt?“ (BAP) differenziert werden. Sowohl für die Projektphasen der Planung als auch der Bauausführung werden entsprechende BAP aufgrund der unterschiedlichen Anforderungen an die Liefergegenstände benötigt. Ein BAP sollte im Verlauf der Planung fortgeschrieben werden, um Änderungen im Projekt zu entsprechen.

2.1.2.3 LoD – Level of Development

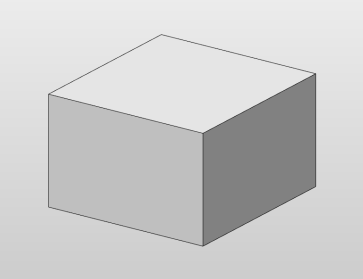
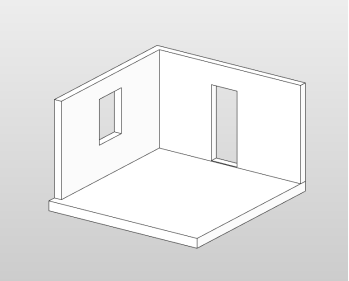
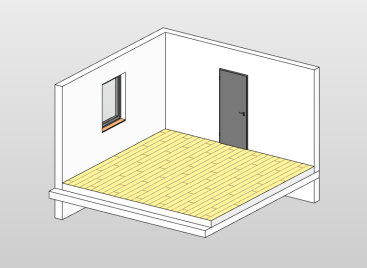
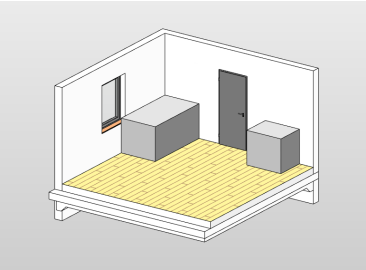
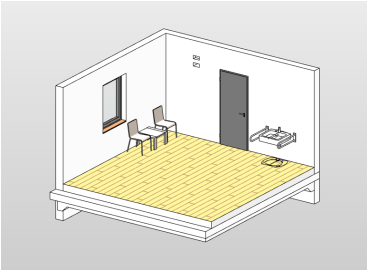
Der Level of Development (zu Deutsch „Entwicklungsgrad“) beschreibt die Informationstiefe von digitalen Modellelementen. Ferner kann der LoD in die Teile LOI (Level of Information) und LOG (Level of Geometry) unterteilt werden. Im amerikanischen Raum wird hier wie folgt zwischen den LOD 100 – 500 mit entsprechend steigendem Informationsgrad unterschieden. Weitere LOD der Stufen 600 und 700

[11]Vgl. (ARGE BIM4INFRA2020, 2019) S.6

[12]Vgl. (ARGE BIM4INFRA2020, 2019) S.6

sind möglich und würden die stetige Aktualisierung aller Daten, wie Umbauten oder Schäden bezeichnen. Hier wird das As-Built-Modell im LOD 500 als finaler und zu erreichender Datenstand betrachtet, da eine Betrachtung der Nutzungsphase den Umfang dieser Untersuchung übersteigen würde. Dennoch sollten entwickelte Arbeitsprozesse auf den Betrieb des Objektes bei steter Pflege des Koordinationsmodells anwendbar sein. Betrachtet wird speziell die Leistungsphase 5, da diese im Zuge der Planung den größten Anpassungsspielraum bietet und zeitgleich einen hohen Informationsgrad birgt.

Tabelle 2, Unterscheidung der LOD[13]

LOD 100 – Konzept	**LOD 200 – Schema**	**LOD 300 – Detail**
Lph. 1-2	**Lph. 2-3**	**Lph. 3-4**
		
Das 3D-Modell beinhaltet grundlegendste und somit konzeptionelle Informationen. Eigenschaften wie Fläche, Höhe, Volumen, Position und Ausrichtung sind definiert.	Annähernde Informationen wie Mengen, Größen, Formen, Position und Ausrichtung sind hinterlegt. Weiterhin können nicht-geometrische Informationen hinterlegt sein.	Elemente sind als spezifisches System, Objekt oder Gruppe in Hinsicht ihrer geometrischen Attribute modelliert. Auch semantische Informationen sind hinterlegt.
LOD 350 – Konstruktionsdokumentation	**LOD 400 – Herstellung und Zusammenbau**	**LOD 500 – „Wie gebaut“**
Lph. 4-5	**Lph. 5-7**	**Lph. 7-9**
		
Details wie die Referenzierung von Bauteilen zueinander und gegenüber verschiedenen Systemen sind grafisch und schriftlich definiert.	Elemente sind als spezifische Bauteilgruppen mitsamt präzisen geometrischen Daten, nicht-geometrischen Informationen und Bauausführungsinformationen angelegt.	Die Elemente und Informationen sind so modelliert, wie sie gebaut wurden, sodass sie zur Anwendung herkömmlicher CAFM-Prozesse verwendet werden können. Informationen sind entsprechend der tatsächlichen Werte angelegt.

[13] Vgl. (BIM Level of Development, 2020)

2.1.2.4 IFC – Industry Formation Classes

IFC stellt ein international standardisiertes digitales Schema zur Bauteilklassifizierung in Form eines XML-, JSON- oder STEP-basierten Datenaustauschformates dar. Mithilfe dieses Standards können Planungsdaten und -modelle unabhängig von der verwendeten Software ausgetauscht und gesichtet werden. Der Standard wurde 2013 ISO-zertifiziert und wird mit der DIN EN ISO 17739 beschrieben.

Zusammengefasst ist das IFC-Schema ein Datenmodell, welches die Identität, Semantik, Charakteristiken, Attribute und Beziehungen von Objekten, abstrakten Konzepten, Prozessen und Personen logisch kodiert.[14]

Charakteristisch ist hierbei die angewandte Ordnungsstruktur. Bauteile werden entsprechend ihrer „Entity“ kategorisiert und mit dem zutreffenden „Predefined Type“ spezifiziert. Ein gutes Beispiel ist die Entity „IfcSensor“ für Sensoren mit den Predefined Types „CONTACTSENSOR“, „WINDSENSOR“, „CO2SENSOR“ und anderen. Deutlich wird hier die hohe Detailierung, welche das Schema ermöglicht. Bereits mit der Zuordnung der Entities sind Bauteile klar voneinander differenzierbar, die Bauart kann weiterhin mit dem Predefined Type festgelegt werden.

2.1.2.5 Information Delivery Manual

In diesem Dokument werden ergänzend zum Inhalt des BAP spezifische Anforderungen an den Datenaustausch gestellt und systematisiert. Es werden Verantwortlichkeiten und Schnittstellen zum Datenaustausch über den gesamten Gebäudelebenszyklus festgelegt.

[14] Vgl. (buildingSMART International, 2022)

Grundlegend ist das IDM durch folgende Dokumente definiert:

- Geschäftsfall
- Prozesslandkarte
- Austauschanforderungen (Tabellen)
- Austausch-Anforderungsmodell
    - Beschreibung
    - Diagramme
    - Definitionen von Austauschkonzepten[15]

Neben Flussdiagrammen zur Übersicht der Arbeitsprozessschritte, spielt unter anderen eine so genannte Attributmatrix eine große Rolle. Diese dient zur Übersicht aller Parameter mit deren Erhebungszeitpunkt nach Leistungsphase, deren Wertefestlegung, als auch zur Zuordnung zu den zugehörigen Bauteilen. Eine Attributmatrix mit Bezug auf die Betriebskostenermittlung soll während der Untersuchung entwickelt werden. Diese wird im Kapitel 4.3 – Nachvollziehbare Bauteilverknüpfungen entwickelt und dient als Grundlage zur Umsetzung der Kostenermittlung.

2.2 Verwendete Software

2.2.1 Autodesk Revit

Mit Revit bietet das Unternehmen Autodesk eine komplexe Softwarelösung für Aufgaben in der Gebäudeplanung und „[…] wird für die Planung, Dokumentation, Visualisierung und Bereitstellung von Architektur-, Ingenieur- und Bauprojekten verwendet.“.[16] Das Programm basiert auf der Programmarchitektur von AutoCAD und wird aufgrund der zentralen Möglichkeit zur Informationsmodellierung als BIM-Authoring-Software bezeichnet. Als eine der populärsten Lösungen für Planungs- und BIM-spezifischen Aufgaben liegt die Verwendung dieser Software für die Erarbeitung neuer BIM-Anwendungsmethoden nahe.

Charakteristisch für die Handhabung von Revit ist das objektorientierte Arbeiten. Innerhalb der Struktur von Bauteilkategorien wie Wänden, Türen, Fenstern,

[15] Vgl. (Buildingsmart International, 2012) S. 16

[16] (Autodesk Inc., 2023)

Geschossdecken und weiteren, sind dazugehörige Elementfamilien wie beispielsweise eine zweiflügelige Tür vorhanden. Deren genauen Maße werden in einer Instanz der Familie als Typenparameter festgelegt, woraus sich unterschiedliche, namensgebende Familientypen ergeben. Diese Eigenschaften sind also stets identisch, bei Verwendung desselben Elementtyps. Eigenschaften, die sich nach Vorgaben der Verortung ergeben, wie zum Beispiel der Fußbodenaufbau, werden in Exemplarparametern hinterlegt, welche sich für die Exemplare, also die individuellen Elemente, unterscheiden können. Neben der möglichen Nutzung vorhandener Elementfamilien können diese von Grund auf konstruiert, verschachtelt und parametrisiert werden, um den Planungsanforderungen gerecht zu werden. Zumeist unterliegen den Parametern abhängige Bemaßungen verschiedener geometrischer Operationen und können mit Berechnungsformeln hinterlegt werden. Auch nichtgeometrische Parameter können angelegt werden. Diese dienen zur Hinterlegung von Informationen und haben keinen Einfluss auf die Geometrie.

Wie in Abbildung 2 veranschaulicht, entstammen drei der vier Exemplare der Familie „TU DF 1 – Stahlzarge Falztüre“. Zwei Exemplare dieser Familie teilen sich die Eigenschaften des Typen „ML – 885 x 2135“, unterscheiden sich jedoch in den Werten ihrer Exemplarparameter „Fußbodenaufbau“.

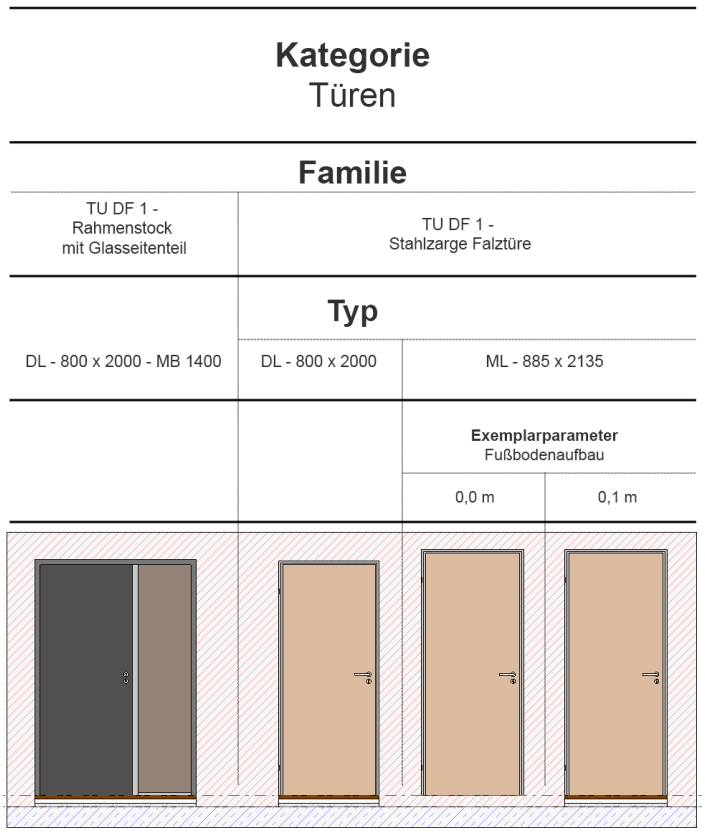


Abbildung 2, Eigene Darstellung zu Bauteilbeziehungen

Dieses Vorgehen kann allgemeinhin als Informationsmodellierung bezeichnet werden und bildet die Grundlage der BIM-Arbeitsweise.

2.2.2 Dynamo

Mit Dynamo bietet Autodesk Revit eine mächtige Schnittstelle zur internen Revit API in Form einer visuellen Programmierungsumgebung oder „Visual Scripting Language“. Die Handhabung basiert auf der Programmiersprache Python, setzt jedoch nicht zwingend Programmierkenntnisse voraus. Hauptsächlich wird mit der Verknüpfung von Ein- und Ausgängen vorhandener Funktionsblöcke gearbeitet, welche Listen aus Daten bestimmter Datentypen einlesen, verarbeiten, beziehungsweise umwandeln und weitergeben. In Abbildung 3 ist die Verknüpfung von eingehenden Elementen zum Funktionsblock „GetLocation“ dargestellt, welcher die Positionen der Elemente entsprechend ihrer Erstellungsbasis auflistet. Im Fall von Wänden, welche linienbasiert erzeugt werden, werden diese Linien ausgegeben. Werden beispielsweise Fenster zu diesem Block überführt, werden die Einfügepunkte mit ihren X-, Y- und Z-Koordinaten ausgegeben.

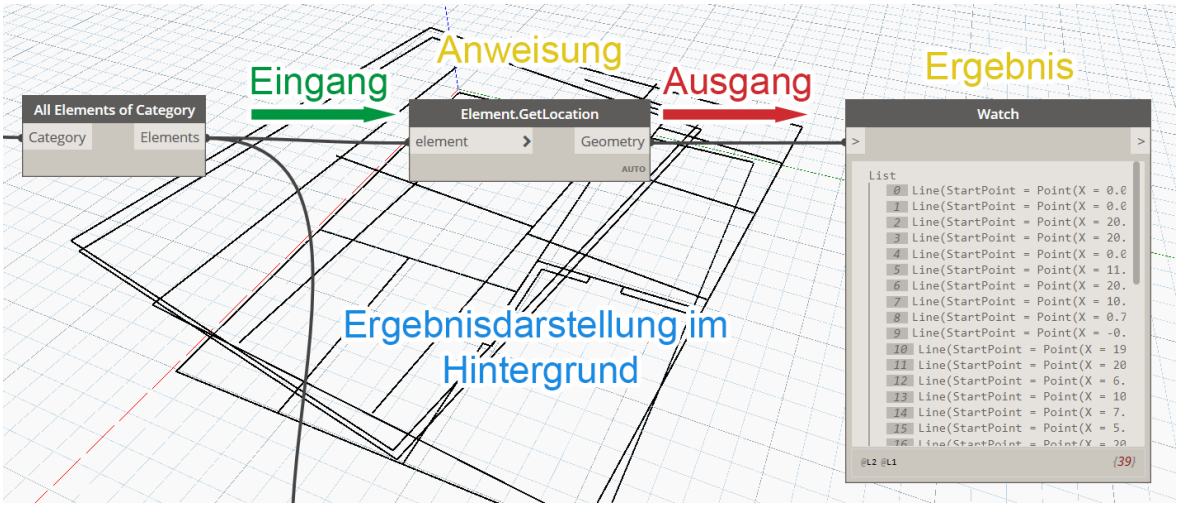


Abbildung 3, Grundlegender Aufbau von Dynamo-Funktionsblöcken

2.2.3 Python

Bei Python handelt es sich um eine effiziente und leicht erlernbare Programmiersprache. Die Differenzierung von Code-Blöcken findet hier über Einrückung, statt über Sonderzeichen, wie geschweifte Klammern oder Semikolons statt, was die Sprache unter Anderem leicht lesbar erscheinen lässt.

Diese Eigenschaften und die Möglichkeit der Entwicklung von Programmen mit, im Vergleich zu anderen Sprachen, relativ geringer Anzahl Code-Zeilen, macht Python für diese Untersuchung zu einem hervorragenden Werkzeug. Als Entwicklungsumgebung wurde sich für die Software „PyCharm“ der Firma JetBrains entschieden.[17]

Mit Python können größere Datensätze effizient ausgewertet werden. Python-Programme können auf mehreren Prozessorkernen ausgeführt werden und bieten somit eine erhöhte Skalierbarkeit. Ein weiterer Vorteil ist die Flexibilität, die mit einer Programmierung mit Python einhergeht. Beispielsweise kann eine Zusammenführung mehrerer Datenquellen mit Excel Schwierigkeiten bereiten, nicht zuletzt mit schnell unübersichtlich werdenden Formeln, welche auf S-Verweise angewiesen sind. Mit Python können außerdem Datenquellen wie JSON oder XML Daten mithilfe zusätzlicher Bibliotheken einfach ausgelesen werden. Insbesondere bei komplexeren Berechnungen ist Python oft schneller in der Verarbeitung großer Datenmengen. Aufgrund vieler leistungsfähiger Code-Bibliotheken in Python kann die Funktionalität des Codes erweitert werden. Auch die Verfügbarkeit spezieller Bibliotheken, welche speziell zum Zweck der Datenauswertung entwickelt wurden stellt einen großen Vorteil dar. Python-Code kann außerdem um einiges einfacher gewartet werden. Tabellenblätter in Excel können gerade durch komplexe Formeln und Zell-Abhängigkeiten oft unübersichtlich werden.

Zusätzlich ist zu erwähnen, dass die Verwendung von Python kostenlos möglich ist und keinen Lizenzerwerb voraussetzt.

2.3 Relevante Anwendungsstandards

Zur ordnungsgemäßen Zuordnung von Planungsinformationen zu den resultierenden Kosten ist zunächst eine Auflistung der betriebsrelevanten Kostenarten und deren Kostenstellen von Nöten. Nach Auflistung der zu berücksichtigenden Kosten, können diese mit den vorhandenen Informationen der Fachmodelle in Verbindung gebracht werden.

Für die Untersuchung sind diverse Standards, welche im Prozess der Objektplanung mit BIM unabdingbar sind, anzuwenden. Diese behandeln unter anderem die Handhabung der Informationen, welche für die teilhabenden Parteien während der

[17] (JetBrains s.r.o., 2023)

Planung und des Betriebes nötig sind, um einen möglichst wirtschaftlichen und zeiteffektiven Umgang des Projektes zu gewährleisten. Deren Inhalt wird folgend ohne direkten Bezug auf die Untersuchung erläutert, um einen groben Überblick der vorgeschriebenen Grundlagen zu schaffen.

Folgende Informationsquellen bieten die rechtlichen und strukturellen Grundlagen für eine Berechnung der Betriebskosten.

2.3.1 DIN EN ISO 276

„Die Berechnung von Kosten im Bauwesen wird in der DIN 276 „Kosten im Bauwesen“ geregelt. Diese Norm gilt für die Ermittlung und die Gliederung von Kosten im Hochbau. Sie erfasst die Kosten für Maßnahmen zur Herstellung, zum Umbau und zur Modernisierung der Bauwerke sowie die damit zusammenhängenden Aufwendungen (Investitionskosten).

In der DIN 276 werden nicht nur die Kosten von Hochbauten, sondern darüber hinaus auch weitere Leistungen mit einbezogen, die in unmittelbarem Zusammenhang mit der Erstellung von Hochbauten stehen, wie beispielsweise Kosten für Grundstück, Erschließung und Außenanlagen.“[18]

2.3.2 DIN EN ISO 19650 – Informationsmanagement mit BIM

Die DIN EN ISO 19650 stellt eine grundlegende Empfehlung für das Informationsmanagement im Kontext des Building Information Modeling dar. Diese Empfehlung betrifft die Verwaltung, den Austausch, die Aufzeichnung, die Versionierung und die Organisation von Informationen für alle am BIM-Prozess beteiligten Akteure.[19]

[18] (Gutachten.net, 2023)

[19] (f:data GmbH, 2023)

2.3.3 DIN 18960 – Nutzungskosten im Hochbau

Während die DIN ISO 276 die Kostengruppen der Bauwerkskosten zusammenfasst, liefert die DIN ISO 18960 eine Gliederung der nutzungsspezifischen Kostengruppen. Hierbei werden die Kosten in vier Hauptgruppen unterteilt:

- 100 - Kapitalkosten
- 200 - Objektmanagementkosten
- 300 - Betriebskosten
- 400 - Instandhaltungskosten

Ergänzt werden diese Gruppen um zwei weitere Ebenen, welche die Kostenstellen weiter spezifizieren. Aufgrund der modelbasierten Berechnungsfähigkeit der Kostengruppen, welche in Kapitel 4 – Voraussetzungen zur Kostenanalyse näher erläutert ist, wird sich allein auf die Kostengruppen 300 und 400 bezogen. In den Kostengruppen 310 und 320 werden die Ver- & Entsorgung von Wasser und Energieträgern und Abfall behandelt. Die Gruppen 330 und 340 behandeln Kostenstellen der Reinigung und Pflege des Gebäudes und Außenanlagen. Die Kosten für die Bedienung, Wartung und Inspektion werden in der KG 350 verortet, während Kosten für Sicherheits- und Überwachungsdienste in der KG 360 hinterlegt sind. Die KG 370 schließt die KG 300 mit Kosten von Abgaben und Beiträgen, wie Steuern und Versicherungsbeiträgen ab. Die Kostengruppe 400 befasst sich mit den Kosten für die Instandsetzung von Bauteilen, dabei wird in der 2. Ebene zwischen Baukonstruktionen, technischen Geräten, der Außenanlage und Ausstattungen und Kunstwerken unterschieden.[20]

2.3.5 AMEV TGA Kosten 2013

Die AMEV hat im November 2013 eine „Empfehlung zur Ermittlung der Kosten für das Betreiben von technischen Anlagen in öffentlichen Gebäuden“[21] (TGA-Kosten Betreiben 2013 herausgebracht. In diesem Dokument sind Berechnungsverfahren für den Betrieb von technischen Anlagen beschrieben. Wichtig zu erwähnen ist hierbei der Geltungsbereich der Berechnungsverfahren:

[20] Vgl. (f:data GmbH, 2020)

[21] (Arbeitskreis Maschinen- und Elektrotechnik BMWSB (AMEV), 2013)

„*Das Berechnungsverfahren gilt*

▪ *für die technische Ausrüstung von Liegenschaften oder Gebäudeportfolien die in ihrer Gesamtheit eine Größenordnung von ca. 1.000 m2 Nettogrundfläche (NGF) nicht unterschreiten.*

▪ *für die technischen Anlagen innerhalb ihrer technischen Lebensdauer (vgl. VDI 2067); übersteigt das Alter einer Anlage diesen Betrachtungszeitraum, erhöhen sich die Instandhaltungskosten überproportional.*

▪ *bei einer sachgerechten Aufbau- und Ablauforganisation unter Einsatz der Sachmittel, die im Einzelfall für eine wirtschaftliche Durchführung der Aufgaben erforderlich sind; das Anwachsen eines Instandhaltungsstaus, der zu erhöhten Instandsetzungsmaßnahmen führt, ist dabei nicht berücksichtigt.*

▪ *Für größere Liegenschaften bzw. Immobilienportfolios, bei welchen sich Abweichungen im Mittel ausgleichen; bei Einzelgebäuden können größere Streuungen auftreten, so dass eine Anwendung problematisch ist. Bei Gebäuden mit einem sehr hohen Wiederbeschaffungswert (WN (KG 400) > 10 Mio. €) kann das Verfahren zu einem überhöhten Mittel- bzw. Personalbedarf führen. Die Ergebnisse der Berechnung für solche Gebäude sind daher kritisch zu prüfen und ggf. zu korrigieren, da ansonsten das gesamte Liegenschaftsergebnis verfälscht werden kann*.“[22]

In dieser Untersuchung werden die Formeln zur Ermittlung von den Kosten für Wartung, Inspektion und Bedienung und der Instandsetzungskosten des Dokuments genutzt. Mehr dazu im Kapitel 4.2 Kostenermittlung und Bezugsgrößen.

[22] (Arbeitskreis Maschinen- und Elektrotechnik BMWSB (AMEV), 2013) S.7

3 Grundlegende Methodik

3.1 Organisatorischer Prozessablauf

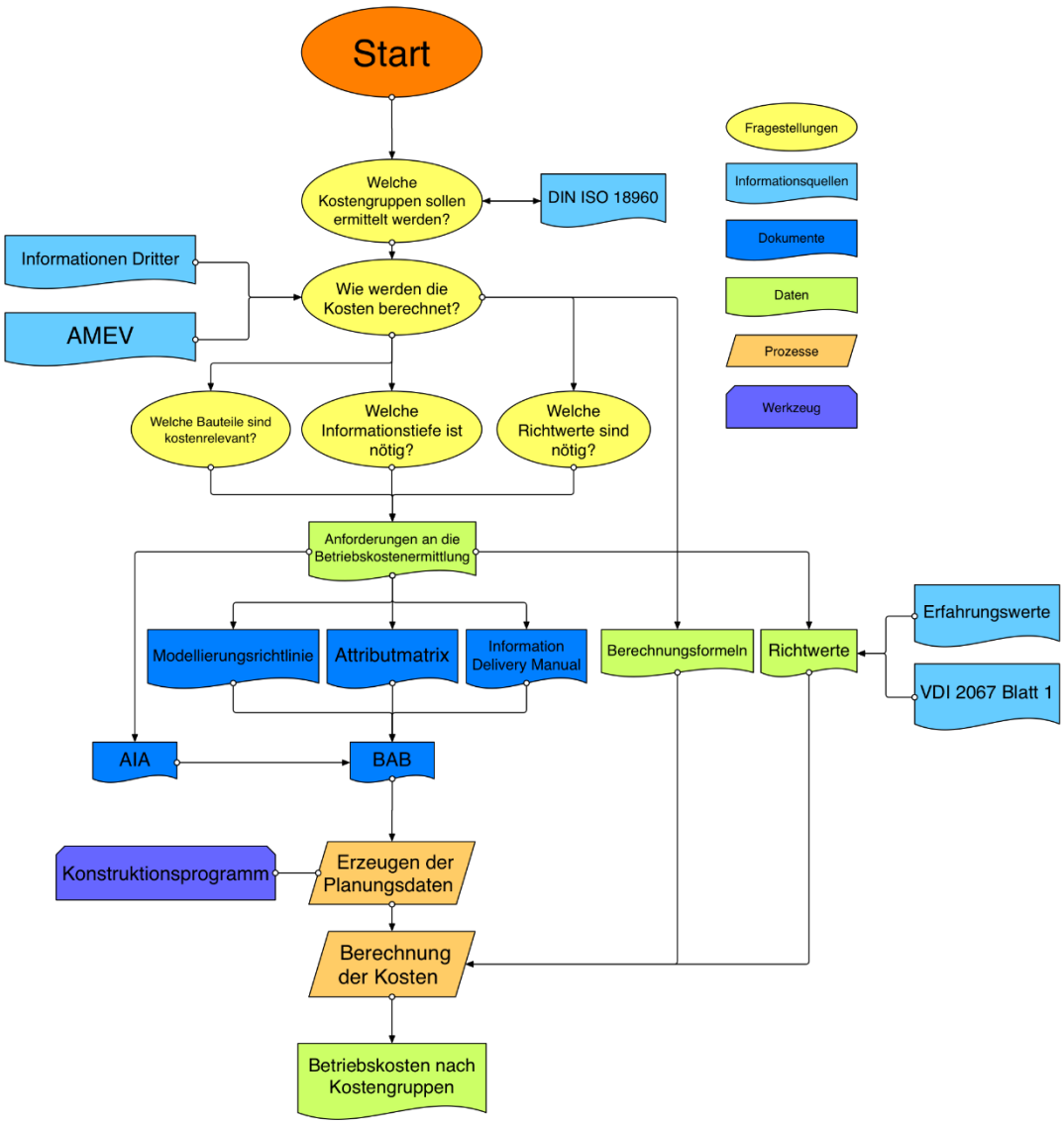


Abbildung 4, Organisatorischer Ablauf zur Berechnung der Kostenermittlung im BIM-Prozess, eigene Darstellung

Grundsätzlich müssen für die modellbasierte Kostenermittlung grundlegende Anforderungen an die Planung anhand der in Abbildung 5 dargestellten Fragestellungen geklärt werden. Es ist festzulegen, welche Kostengruppen tatsächlich aus dem Modell erhoben werden sollen, wie diese errechnet werden und welche Richtwerte

dazu angesetzt werden. Außerdem sollten die betroffenen Elemente und die nötige Informationstiefe der Daten festgestellt werden. Ergänzend sollten die Erhebungszeitpunkte und die für die Erhebung verantwortlichen Akteure definiert werden. Weitere nötige Festlegungen sind die zu verwendenden Werkzeuge und Schnittstellen, also wo welche Informationen wie verarbeitet und übermittelt werden. Die Ergebnisse dieser Fragestellungen werden in den AIA und dem BAP in Form der Modellierungsrichtlinie, einer Attributmatrix und dem Information Delivery Manual niedergeschrieben. Auf Grundlage dieser Festlegungen können die Planungsdaten erstellt und Berechnungen durchgeführt werden.

3.2 Iterativer Prozessablauf

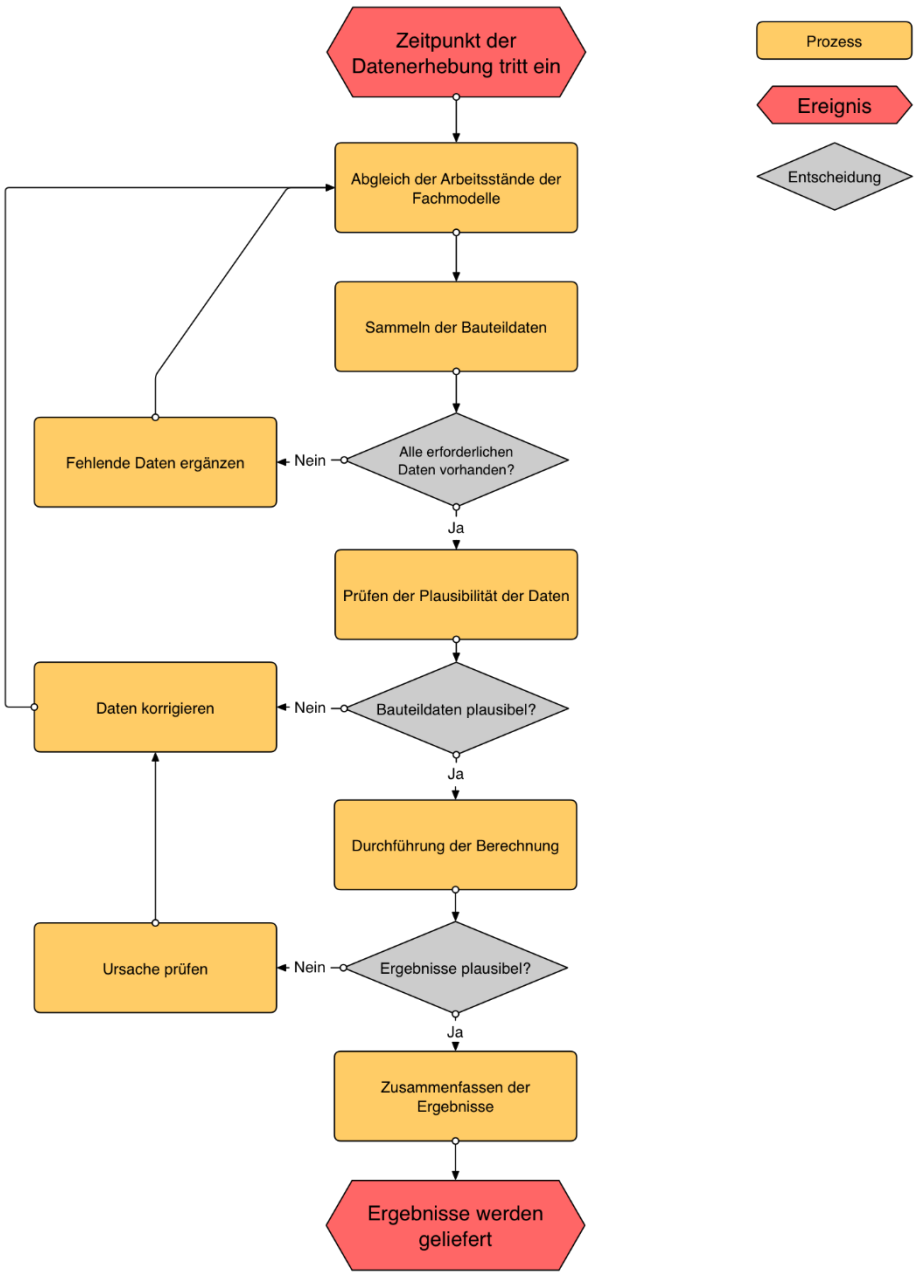


Abbildung 5, Iterativer Prozessablauf der Datenerhebung, eigene Darstellung

Entsprechend der festgelegten Zeitpunkte einer Datenerhebung sollte dessen Ablauf klar definiert sein, sodass alle nötigen Prozessschritte eingehalten werden, mit jeder Erhebung wiederholbar sind und über den Planungsprozess konsistent bleiben.

3.3 Zeitpunkt der Informationserhebung

Wie eingangs erwähnt wird der Anwendungsfall der Kostenschätzung und -ermittlung in die Leistungsphasen 2-3 eingeordnet. Zweck der planungsbegleitenden Erhebung der prognostizierten Betriebskosten, soll die Möglichkeit der Anpassung der Planung auf Grundlage der Kosten sein, woraus folgt, dass die Erhebung auch in der Lph 5 angewandt werden soll. Entsprechend des Aufwands des Erhebungsprozesses sollten Festlegungen bezüglich des Erhebungszyklus getroffen werden. Als zukunftsorientierte Zielstellung gilt eine Erhebungssystematik, welche so nah am Planungsprozess wie möglich ist. Denkbar wäre eine Ausgabe der Kosten in Echtzeit, also eine Aktualisierung der Berechnung, welche im Zuge einer Änderung in der Planung erfolgt, sodass Konstrukteure und Planer die Auswirkungen ihrer Überlegungen auf die Betriebskosten direkt dargelegt bekommen. Aktuell ist für die Implementierung der Systematik in dieser Form viel Vorarbeit nötig, weshalb eine Erhebung zum Zeitpunkt der Lieferung von Arbeitsständen, oder ein wöchentlicher Turnus als realistisch einzustufen ist. Die zwei-wöchentliche Erhebung dürfte hierbei sinnvoll sein, da sich Planungsanpassungen so flexibler gestalten dürften und dennoch Zeit für eventuelle Fehlerbeseitigung vorhanden ist. Bei der alleinigen Erhebung zum Lieferzeitpunkt der Arbeitsstände werden die Informationen verzögert erfasst, was eine zeitnahe Reaktion und Optimierung erschwert. Infolgedessen könnte dem Auftraggeber ein unerwünschter Eindruck bezüglich der Prognose der Betriebskosten vermittelt werden.

3.4 Verantwortlichkeiten

Bis zur Entwicklung einer vollautomatisierten Datenerhebung von errechneten Betriebskosten wird es nötig sein, Personal zur Begleitung und Durchführung des Prozesses abzustellen. Naheliegend wäre die Verantwortungsübertragung an den BIM-Koordinator eines Projektes oder an den Modellverantwortlichen eines Fachbereiches. Ausschlaggebend für die Aufgabenverteilung ist die Schnittstellenfestlegung. Entsprechend der beteiligten Gewerke im Planungsprozess und dem Datenursprung der Informationen zur Kostenermittlung sollten die Verantwortlichkeiten zur Datenpflege festgelegt werden.

Eine Übersicht über die Verantwortlichkeiten lässt sich gut über eine sogenannte RACI-Matrix darstellen. Hierbei werden Prozessschritte und teilhabende Akteure gelistet und miteinander in Verbindung gebracht. Mit den Buchstaben R, A, C und I werden die jeweiligen Verantwortlichkeiten wie folgt gekennzeichnet: R = responsible/durchführend, A = accountable/entscheidend, C = consulted/beratend und I =

informed/informiert. Eine denkbare Verteilung der Verantwortlichkeiten ist in Tabelle 3, RACI-Matrix zur Darstellung der Verantwortlichkeiten in Form einer einfachen RACI-Matrix dargestellt, berücksichtigt sind hierbei die vorgenannten organisatorischen Prozessschritte, welche generell angewandt werden können, weiterhin werden mit den Subprozessen des Prozessschritt 8 – Berechnung der Kosten spezifische Schritte behandelt, die im Zuge der beispielhaften Umsetzung in Kapitel 5: Prototypische Implementierung erarbeitet wurden. Im Prozessschritt 7: Erzeugen der Planungsdaten wird davon ausgegangen, dass eine Familienbibliothek mitsamt der nötigen Bauteilinformationen nicht vorhanden ist und diese Daten per Hand eingetragen werden.

Tabelle 3, RACI-Matrix zur Darstellung der Verantwortlichkeiten

Prozessschritt		Involvierte Rollen					
	R - responsible; **A** - accountable; **C** - consulted; **I** - informed	Auftraggeber	BIM-Manager	BIM-Koordinator	Projektleiter	Modellverantwortlicher	Konstrukteur/Projektingenieur
1.0	**Ermittlung der zu berechnenden Kostengruppen**	A/I	R	C	C		
2.0	**Ermittlung der Anforderungen an die Berechnung**		A	R			I
3.0	**Ermittlung relevanter Bauteile**			A	I	I	R/C
4.0	**Festlegen der nötigen Informationstiefe**		A	R	I	I	C
5.0	**Ermittlung der benötigten Richtwerte**		I	R	I		C
6.0	**Formulierung der AIA und des BAP**	I	A	R	C/I	I	
7.0	**Erzeugen der Planungsdaten (berechnungsbezogen)**			C		A	R
7.1	Anlegen der Parameter in den Fachmodellen			C	A	R	I
7.2	Platzierung der Bauteile			I	A	I	R
7.3	Eintragen der Bauteildaten in die Parameter		I	C/I	I	A	R
7.4	Qualitätssicherung der Daten		I	A	I	R	C
8.0	**Berechnung der Kosten (iterativ)**			R	A	I	C
8.1	Elementweise Berechnung der Kosten		I	C	I	R/A/C	C
8.2	Zusammenführen der Ergebnisse in Sammelparameter		I	C	I	R/A/C	I
8.3	Ausgabe der Sammelparameter		I	C	I	R/A/C	
8.4	Berechnung mit weiteren, externen Richtwerten		I	R/A/C	I	I	
8.5	Zusammenfassung der Ergebnisse nach Kostengruppen	I	I	C	I	A	R

4 Voraussetzungen zur Kostenanalyse

4.1 Erforderliche Parameter

Aus Kapitel 4.2 werden zur Kostenermittlung nötige Daten ersichtlich. Diese Daten sollen in Parametern hinterlegt werden, welche mit den Bauteilelementen im Planungsmodell verknüpft werden. Diese werden in Werte- und Sammelparameter unterteilt, wobei Sammelparameter die zugehörige Kostengruppe nach DIN 19860 im Namen tragen.

Die Parameter folgen einer festgelegten Syntax:

BK_[KG nach DIN 19860]_Name

BK_Name

Es ergeben sich folgende Parameter zur Hinterlegung im Planungsmodell:

Tabelle 4, Übersicht resultierender, erforderlicher Parameter

Bauteilparameter	**Raumparameter**	**Flächenparameter**
BK_UseAmount	BK_Occupancy	BK_321_WasteAmount
BK_UsePerDay	BK_311_UseAmount	BK_333_CleaningExp
BK_UseTime	BK_312_UseAmount	BK_343_CleaningExp
BK_CleaningExp	BK_313_UseAmount	BK_345_CleaningExp
BK_CleaningCycle	BK_314_UseAmount	
BK_CleaningPerf	BK_315_UseAmount	
BK_ApprAcqCost	BK_316_UseAmount	
BK_YCRoof	BK_317_UseAmount	
DIN276	BK_321_WasteAmount	
	BK_322_WasteAmount	

	BK_331_CleaningExp	
	BK_332_CleaningExp	
	BK_334_CleaningExp	
	BK_350_ServiceCost	
	BK_420_ServiceCost	

Eine Übersicht der Parameter und deren Zugehörigkeit zu spezifischen Bauteilen werden in einer sogenannten Attributmatrix zusammengeführt. Mehr dazu folgt im Kapitel 4.3.1 – Verknüpfung der Parameter und Bauteile.

4.2 Kostenermittlung und Bezugsgrößen

Zur modellbasierten Auswertung können und sollten lediglich Kostenstellen hinzugezogen werden, deren Datenursprung im Modell liegen kann. Kapitalkosten ergeben sich beispielsweise aus Werten, welche nicht dem Modell, sondern vorgehenden Verträgen zu entnehmen sind. Diese Daten können nach der Datenerhebung aus dem Modell addiert werden. Aus diesem Grund wird in dieser Untersuchung auf die Inbezugnahme der Kostengruppen 100 und 200 der DIN 18960 verzichtet.

Ausschlusskriterium für die Kostengruppen ist die Möglichkeit der Hinterlegung relevanter Informationen im Modell. Weiterhin unterscheidet sich die Zusammensetzung von Kosten mancher Kostengruppen von Projekt zu Projekt so weit, dass eine Standardisierung nicht möglich ist.

Zur Berechnung der relevanten Daten wird, bis auf wenige Ausnahmen, von einem Elementbezug ausgegangen. Das heißt, dass die Ergebnisse durch die Rückrechnung der jeweiligen Formeln auf die Ausgangswerte einzelner Bauteile zurückgeführt werden können. Ein veranschaulichendes Beispiel ist hier die Berechnung des Wasserverbrauchs von Sanitäreinrichtungen. Da die Anzahl von Sanitärobjekten zumeist aus der Belegung errechnet wird, würden sich die Kosten auf ein Bad auf Grundlage der Anzahl der Nutzer beziehen, jedoch kann nur über eine Durchschnittsbildung Rückschluss über den Verbrauch eines einzelnen Waschbeckens getroffen werden. Da in der Konstruktionssoftware grundsätzlich elementbezogen gearbeitet wird, wird auch in der Auswertung im Sinne der konsistenten Arbeitsweise elementbezogen gerechnet. Daraus ergibt sich außerdem die Möglichkeit, die

Kostenbeeinflussung durch Optimierung der Wahl der Bauteile analysieren zu können (bspw. mit dem Einsatz von wassersparenden Ausführungen der Spülung oder Wasserentnahme). Die Untersuchung umfasst die Entwicklung einer Berechnungsmethodik, jedoch keine Variantenuntersuchungen, weshalb zum Umsetzungsbeispiel mit angenommenen- oder richtwertbezogenen Werten gerechnet werden wird.

Kosten, welche einer vergleichbaren Berechnungssystematik folgen, werden zusammengefasst.

Für die nachfolgenden Berechnungsformeln sind bauteilspezifische Variablen zu vergeben, welche sich aus den erforderlichen Werten ergeben und in Tabelle 4 zusammengefasst sind.

Tabelle 5, Übersicht nötiger Variablen für die Berechnung

Variablen		
Bedeutung	**Parameter**	**Funktion**
Raumbelegung	BK_Occupancy	m² Raumnutzfläche pro Person
Nutzungszyklus	BK_UsePerDay	Anzahl Nutzungen pro Person pro Tag
Nutzungsdauer	BK_UseTime	Nutzungsdauer pro Nutzung in s (max. 86400 s)
Reinigungsturnus	BK_CleaningCycle	Anzahl Reinigungen pro Woche
Leistungszahl	BK_CleaningPerf	Reinigungsleistung in m²/h oder s/NS
Wiederbeschaffungswert	BK_ApprAcqCost	Näherungsweiser Neubeschaffungswert in Euro
Ertragsbeiwert der Dachfläche	BK_YCRoof	Faktor des Ertragsbeiwertes einer Dachfläche

Es folgt eine Auflistung der Berechnungsweisen der Kostengruppen nach DIN 18960. Sofern anwendbar finden sich im jeweiligen Tabellenkopf die zu erreichenden Ergebniswerte.

4.2.1 KG 310 - Versorgung

4.2.1.1 KG 311 – Wasser

Regen- und Abwasser, welches zur Subvention des Trinkwassers genutzt wird, kann vom Wasserverbrauch abgezogen werden, erfolgt jedoch im Umsetzungsbeispiel in Kapitel 5 nicht.

Tabelle 6, Berechnung der KG 311

Kubikmeter pro Jahr * Kosten pro Kubikmeter		
Ergebnis	**Parameter**	**Formeln**
Kubikmeter pro Jahr	**KG 311**	$\sum_{n=1}^{Räume} (\text{Liter pro Raum}) * 210\ Arbeitstage\ im\ Jahr$ **(1)**
Kubikmeter pro Raum	**BK_311_UseAmount**	$\sum_{n=1}^{Sanitärinstallationen} (\text{Nutzmenge})$ **(2)**
Nutzmenge pro Tag	**BK_UseAmount**	$((\frac{Raumnutzfläche}{Raumbelegung}) * Nutzungszyklus * Nutzungsdauer * Durchflussgeschwindigkeit\ (\frac{m^3}{s}))$ **(3)**

4.2.1.2 KG 312 bis 314 - Gas, Öl & feste Brennstoffe

Tabelle 7, Berechnung der KG 312 bis 314

Kubikmeter pro Jahr * Kosten pro Kubikmeter		
Ergebnis	**Parameter**	**Formeln**
Kubikmeter pro Jahr	**KG 312** **KG 313** **KG 314**	$\sum_{n=1}^{Räume} (BK\_xxx\_UseAmount) * 210\ Arbeitstage$ **(4)**
Kubikmeter pro Raum	**BK_312_UseAmount** **BK_313_UseAmount** **BK_314_UseAmount**	$\sum_{n=1}^{Heizkörper} (\text{BK\_UseAmount})$ **(5)**
Nutzmenge pro Tag	**BK_UseAmount**	$BK\_UsePerDay * Verbrauch\ (\frac{m^3}{s})$ **(6)**

4.2.1.3 KG 315 & 316 – Fernwärme & Strom

Tabelle 8, Berechnung der KG 315 und 316

Kilowattstunden pro Jahr * Kosten pro Kilowattstunde		
Ergebnis	**Parameter**	**Formeln**
Kilowattstunden pro Jahr	**KG 315** **KG 316**	$\sum_{n=1}^{Räume} (\text{kWh pro Raum}) * 210$ **(7)**
Kilowattstunden pro Raum	**BK_315_UseAmount** **BK_316_UseAmount**	$\sum_{n=1}^{Installationen} (\text{Nutzmenge})$ **(8)**
Kilowattstunden pro Tag	**BK_UseAmount**	$(\left(\frac{NF}{BK_{Occupancy}}\right) * BK_{UsePerDay} * \left(\frac{\frac{BK_{UseTime}}{60}}{60}\right) * Verbrauch\ (W\ oder\ VA)/1000)$ **(9)**

4.2.1.4 KG 317 - Technische Medien

Die Anforderungen für die Versorgung technischer Medien einer Immobilie, wie der Anschluss an das Internet, das Telekommunikationsnetz oder die Bereitstellung der Fernsehübertragung über anliegende Netzanbieter, sind nicht effektiv im Planungsmodell abzubilden. Sie sind stark von der Nutzungsart abhängig und unterscheiden sich mit jedem Gebäudenutzer. So haben Planungsbüros andere Ansprüche an die Versorgung als Werbeagenturen, Produktionsstätte oder Verwaltungsbüros, um allein unter der Betrachtung von Büro- und Verwaltungsgebäuden wenige vieler Beispiele zu nennen. Nicht zuletzt ist ein Kostenanschlag schwer durchzuführen, da sich die Preise stark mit individuellen Vertragskonditionen und der Entwicklung technischer Standards verändern können.

4.2.2 KG 320 - Entsorgung

4.2.2.1 KG 321 - Abwasser

Die Menge des Abwassers entspricht per se dem Wasserverbrauch, welchem die Menge des abgeleiteten Niederschlagswassers hinzuaddiert wird. Als abgeleitet zählt Niederschlagswasser, welches über Dachrinnen und Abflüsse dem Abwassersystems zugeführt wird

Vor dem Addieren der abgeleiteten Mengen wird ein Korrekturfaktor von 0,5 angesetzt, da Regenwasser einen geringen Verschmutzungsgrad aufweist und somit einen geringeren Reinigungsaufwand seitens der Wasserwerke bedeutet.[23]

Zur Berechnung der Menge des abgeleiteten Regenwassers werden die Fläche des Daches und dessen Ertragsbeiwert benötigt, welche mit der standortspezifischen Niederschlagsmenge multipliziert werden. Diese Werte können aus dem Planungsmodell erhoben werden.

[23] (Bundesamt für Bauwesen und Raumordnung, 2011), S. A4, Formel (7)

Kubikmeter pro Jahr * Kosten pro Kubikmeter		
Ergebnis	**Parameter**	**Formeln**
Kubikmeter Abwasser pro Jahr	**KG 321**	$BK_{321_{WasteAmount}} + BK_{311_{UseAmout}}$ **(10)**
Abgeleitetes Niederschlagswasser pro Jahr	**BK_321_WasteAmount**	$\sum_{n=1}^{Dachflächen} (\mathrm{A_D} * e_D * S_{RW})$ **(11)**

S_{RW} standortspezifische jährliche Niederschlagsmenge in m³/m² Dachfläche p.a.
A_D Dachfläche in m²
e_D Ertragsbeiwert der Dachfläche

Angaben zur standortspezifischen, jährlichen Niederschlagsmenge sind der Statistik-Plattform Statista zu entnehmen.[24] Kennzahlen zu Ertragsbeiwerten von Dachflächen können der DIN 1989-1 entnommen werden und sind in Abbildung 6 einzusehen.[25] Die Berechnungsformel ergibt sich in Anlehnung an den Steckbrief des Bundesministeriums für Verkehr, Bau und Stadtentwicklung zur Ressourceninanspruchnahme zum Thema Trinkwasserbedarf und Abwasseraufkommen.[26]

Beschaffenheit	Ertragsbeiwert % e
geneigtes Hartdach [a]	0,8
Flachdach unbekiest	0,8
Flachdach bekiest	0,6
Gründach intensiv	0,3
Gründach extensiv	0,5
Pflasterfläche/Verbundpflasterfläche	0,5
Asphaltbelag	0,8
[a] Abweichungen je nach Saugfähigkeit und Rauheit	

Abbildung 6, Ertragsbeiwerte von Dächern nach DIN 1989-1: 2002-04

[24] (Statista GmbH, 2022)

[25] (Deutsches Institut für Normung e.V., 2002), S. 27, Tabelle 3 - Ertragsbeiwerte

[26] (Bundesamt für Bauwesen und Raumordnung, 2011)

4.2.2.2 KG 322 - Abfall

Tabelle 9, Berechnung der KG 322

Abfallmenge R,B,A,L pro Jahr * Kosten pro Kubikmeter Abfall		
Ergebnis	**Parameter**	**Formeln**
Abfallmenge Restmüll **Abfallmenge Biomüll** **Abfallmenge Altpapier** **Abfallmenge Leichtverpackung**	**KG 322_R** **KG 322_B** **KG 322_A** **KG 322_L**	**KG 322 * 0,6** **KG 322 * 0,1** **KG 322 * 0,15** **KG 322 * 0,15**
Abfallmenge pro Jahr	**KG 322**	$\sum_{n=1}^{Räume} \left(\frac{\mathbf{BK322_{WasteAmount}}}{\mathbf{1000}}\right)$ **(12)**
Abfallmenge pro Raum pro Monat	**BK_322_WasteAmount**	$NF * 2,5\ l * 4,33$ **(13)**

Ausgehend von einer Infobroschüre der Stadtreinigung Hamburg wird der Richtwert von 2,5 Litern Abfallmenge pro m² Nutzungsfläche und Woche für Bürogebäude angesetzt. Die Aufteilung der Abfallarten wird ebenfalls auf dieser Grundlage angenommen.[27]

[27] (Stadtreinigung Hamburg, 2023), S. 6

4.2.3 KG 330 - Reinigung und Pflege

4.2.3.1 KG 331 - Unterhaltsreinigung

Tabelle 10, Berechnung der KG 331

Reinigungsaufwand pro Jahr in h * Kosten pro Stunde		
Ergebnis	**Parameter**	**Formeln**
Reinigungsaufwand pro Jahr	**KG 331**	$\sum_{n=1}^{Räume} (BK\_331\_UseAmount) * 4,33 * 12$ **(14)**
Reinigungsaufwand pro Raum	**BK_331_CleaningExp**	$\sum_{n=1}^{Flächenarten} (Oberflächenabhängiger\ Aufwand) * Reinigungsturnus$ **(15)**
Oberflächenabhängiger Aufwand pro Reinigung	**BK_CleaningExp**	$\frac{Fläche\ der\ Materialität\ X}{Leistungszahl}$ **(16)**

Die angesetzten Leistungszahlen nach Oberflächen werden auf Grundlage des Merkblattes der Gareba GmbH angenommen und gemittelt.[28] Diese Werte gelten als variabel und können nach Bedarf und entsprechend den abweichenden Anforderungen an die Räume angepasst werden. Für eine Berechnung nach Nutzungsart können Richtwerte nach dem Merkblatt der Gütegemeinschaft Gebäudereinigung e.V. angenommen werden.[29]

[28] (Gareba - GmbH, 2012)

[29] (Gütegemeinschaft Gebäudereinigung e.V., 2020)

4.2.3.2 KG 332 - Glasreinigung

Tabelle 11, Berechnung der KG 332

Reinigungsaufwand pro Jahr in h * Kosten pro Stunde		
Ergebnis	**Parameter**	**Formeln**
Reinigungsaufwand pro Jahr	**KG 332**	$\sum_{n=1}^{Räume} (BK\_332\_CleaningExp) * 4,33 * 12$ **(17)**
Reinigungsaufwand pro Raum	**BK_332_CleaningExp**	$\sum_{n=1}^{Fenster} (BK\_CleaningExp) * BK\_CleaningCycle)$ **(18)**
Oberflächenabhängiger Aufwand pro Reinigung	**BK_CleaningExp**	$\frac{(Glasfläche\ Innen + Glasfläche\ Außen)}{40}$ **(19)**

Nach dem Datenblatt der Gareba GmbH liegt die Leistungszahl für die zweiseitige Reinigung von Fenstern ohne Rahmenwäsche je nach Zugänglichkeit bei 20-60 m²/h. Ausgehend davon wird ein Mittelwert von 40 m²/h angenommen. Für die Inbezugnahme der Rahmenwäsche ist anzuraten, einen weiteren Raumparameter für den abweichenden Reinigungszyklus der Rahmen anzulegen.

4.2.3.3 KG 333 - Fassadenreinigung

Tabelle 12, Berechnung der KG 333

Reinigungsaufwand pro Jahr in h * Kosten pro Stunde		
Ergebnis	**Parameter**	**Formeln**
Reinigungsaufwand pro Jahr	**KG 333**	$\sum_{n=1}^{Geschossflächen} (BK\_333\_CleaningExp) * 4,33 * 12$ **(20)**
Reinigungsaufwand pro Raum	**BK_333_CleaningExp**	$\sum_{n=1}^{Flächenarten} (Oberflächenabhängiger\ Aufwand) * Reinigungsturnus)$ **(21)**
Oberflächenabhängiger Aufwand pro Reinigung	**BK_CleaningExp**	$\frac{Fläche\ der\ Materialität\ X}{Leistungszahl}$ **(22)**

4.2.3.4 KG 334 - Reinigung Technischer Anlagen

Tabelle 13, Berechnung der KG 334

Reinigungsaufwand pro Jahr in h * Kosten pro Stunde		
Ergebnis	**Parameter**	**Formeln**
Reinigungsaufwand pro Jahr	**KG 334**	$BK\_334\_CleaningExp * 4,33 * 12$ **(23)**
Reinigungsaufwand pro Raum	**BK_334_CleaningExp**	$\sum_{n=1}^{Räume} (Aufwand\ pro\ Reinigung * Reinigungsturnus)$ **(24)**
Aufwand pro Reinigung	**BK_CleaningExp**	$\frac{(Aufwand\ pro\ Reinigung\ in\ s)}{60/60}$ **(25)**

4.2.4 KG 340 - Reinigung und Pflege von Außenanlagen

4.2.4.1 KG 341 bis 343 - Befestigte Anlagen, Pflanz-, Grün- & Wasserflächen

Tabelle 14, Berechnung der KG 341 bis 343

Reinigungs- /Pflegeaufwand pro Jahr in h * Kosten pro Stunde		
Ergebnis	**Parameter**	**Formeln**
Aufwand pro Jahr	**KG 341** **KG 342** **KG 343**	$\sum_{n=1}^{Flächen} (BK\_34X\_UseAmount) * 4,33 * 12$ **(26)**
Aufwand pro Fläche	**BK_331_CleaningExp**	$\sum_{n=1}^{Flächenarten} (Oberflächenabhängiger\ Aufwand * Reinigungsturnus)$ **(27)**
Oberflächenabhängiger Aufwand pro Reinigung	**BK_CleaningExp**	$\frac{Fläche\ der\ Materialität\ X}{Leistungszahl}$ **(28)**

4.2.4.2 KG 344 bis 346 – Baukonstruktionen, technische Anlagen und Einbauten in Außenanlagen

Die Berechnung der Kostengruppen 344, 345 und 346 können mit der Berechnungsformel der KG 334 umgesetzt werden. Der Unterschied liegt allein in der Zuordnung der Elemente zu ihrer Verortung, welche hier in Sammelparametern der beinhaltenden Flächenelemente erfolgt.

4.2.4 KG 350 & 400 - Betreiben

Zur Ermittlung der Kosten für den Betrieb eines Gebäudes wird das Berechnungsverfahren der AMEV – TGA-Kosten Betreiben 2013 angewandt. Da dieses Verfahren sowohl die Wartungs-, Inspektions- und Bedienungskosten, als auch die Instandsetzungskosten inkludiert, werden die Kostengruppen unter diesem Punkt zusammengefasst behandelt. Folgendes Schema findet hierbei Anwendung[30]:

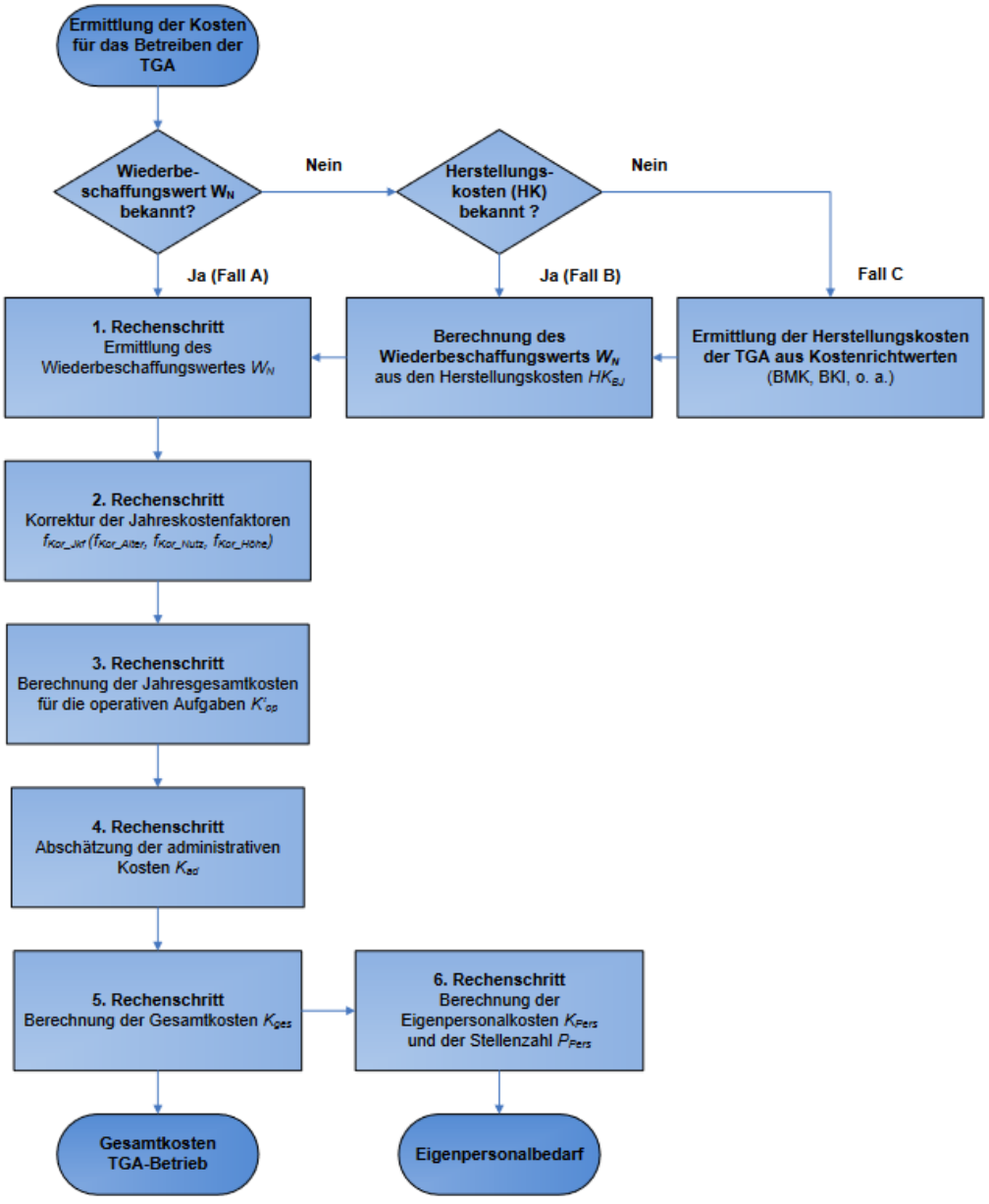


Abbildung 7, Schema des AMEV Berechnungsverfahrens

[30] (Arbeitskreis Maschinen- und Elektrotechnik BMWSB (AMEV), 2013), S.16

4.2.4.1 Ermittlung der Herstellungskosten

Mit steigender Informationstiefe der Planung kann mit bekannten Herstellungskosten von Bauteilen, welche im Parameter BK_ApprAcqCost hinterlegt werden, gerechnet werden. Innerhalb des digitalen Planungsmodelles sollte mit Werten zum Zeitpunkt der Planung gearbeitet werden. Trendanalysen und Prognosen zum Abschätzen zukünftiger Kosten sollten außerhalb des Modells erfolgen, da entsprechende Berechnungen ohne integrierte Zusatzprogramme der Planungssoftware schwer- bis unmöglich umzusetzen sind.

Zum Zeitpunkt der frühen Planungsphase sind die genauen Herstellungskosten von Bauteilen oft nicht bekannt, weshalb folglich mit Kostenrichtwerten gerechnet, und entsprechend nach Fall C des Berechnungsverfahrens gearbeitet wird. Eine elementbezogene Rechnung ist hier noch nicht möglich. Der zu diesem Zeitpunkt aus dem Modell heranziehbare Wert ist die Nutzfläche des Gebäudes, um mit den nutzflächenbezogenen Richtwerten arbeiten zu können. In der Leistungsphase 5 sollte dann mit der Verfügbarkeit von Herstellungskosten der Bauteile mit diesen gerechnet werden können und eine höhere Präzision der Berechnung mit Bezug auf die individuelle Planung erreicht werden.

Mit Bezug auf die Leistungsphase 2 sollte für die Herstellungskosten mit Orientierungswerten gerechnet werden. Die AMEV-Empfehlung bezieht sich hier auf Werte für Hochschulgebäude, welche seit 2018 jährlich vom Ausschuss für staatlichen Hochbau der ARGEBAU veröffentlicht werden. Durch die Bildung des Durchschnittswertes aus den Orientierungswerten nach Gebäudenutzung für weitere Hochschulgebäude[31] ergeben sich Bauwerkskosten von 4354,67 €/m² NUF und ein Technikkostenanteil von 28,83 %, also 1255,45 €/m² Nutzfläche. Dieser Wert wird als Überschlag der Herstellungskosten H_K angenommen.

Zur Ermittlung des Wiederbeschaffungswertes wird der Baupreisindex (BPI) hinzugezogen und wie folgt verrechnet:

$$W_N = HK_{BJ} * \frac{BPI_{aktuell}}{BPI_{BJ}} \quad \textbf{(29)}$$

[31] (Geschäftsstelle der Bauministerkonferenz Land Baden-Württemberg, 2022), S.2

Im Sinne der Kostenprognose ist für den $BPI_{aktuell}$ der BPI des Betrachtungsjahres einzusetzen, welcher insofern aus vorhandenen Daten[32] zu extrapolieren/anzunehmen ist. Für den BPI_{BJ} ist der BPI zum Zeitpunkt der Berechnung einzutragen.

Jahr, Quartal		Wohngebäude	Büro-gebäude	Gewerbliche Betriebs-gebäude
2022	IV	154,7	157,2	157,7
	III	151,0	153,4	154,2
	II	147,2	149,2	150,4
	I	138,1	139,7	140,0
2021	IV	132,3	133,4	134,1
	III	129,6	130,0	131,0
	II	125,2	125,4	126,0
	I	120,8	121,2	121,4
2020	IV	115,6	116,0	116,0
	III	115,1	115,5	115,6
	II	117,7	118,1	118,2
	I	117,2	117,6	117,7
2019	IV	115,7	116,1	116,2
	III	115,1	115,4	115,6
	II	114,3	114,6	114,7
	I	113,4	113,7	113,9
2018	IV	111,5	111,7	111,9
	III	110,6	110,9	111,0
	II	109,2	109,5	109,5
	I	108,2	108,5	108,5
2017	IV	106,4	106,7	106,8
	III	105,7	105,9	105,9
	II	104,9	105,1	105,1
	I	104,0	104,2	104,1

Abbildung 8, Reihe des Baupreisindex von 2017 bis 2022

[32] (Statistisches Bundesamt (Destatis), 2023)

4.2.4.2 Korrektur der Jahreskostenfaktoren

Im Sinne der AMEV werden Jahreskorrekturfaktoren einbezogen, dabei handelt es sich um Faktoren in Bezug auf das Anlagenalter, die Nutzungsart des Gebäudes und die Gebäudehöhe.[33]

Altersspanne	f_{Korr_Alter}
Alter 0 – 5 Jahre (Instandsetzung)	0,5
Alter 6 – 20 Jahre (Instandsetzung)	1,0
Alter ab 21 Jahre (Instandsetzung)	1,8
Alle Altersklassen (Wartung/Inspektion/Bedienung)	1,0

Abbildung 9, Jahreskorrekturfaktoren des Anlagenalters nach AMEV

Je nach Betrachtungszeitraum der Kostenermittlung werden zur Berechnung der KG 400 jeweilige Korrekturfaktoren der Instandsetzung entsprechend Abbildung 8 eingesetzt.

Gebäudenutzungsart	f_{Korr_Nutz}
Instituts-/Lehrgebäude	0,8
Forschungs-/Laborgebäude	0,9
Kindertagesstätten	0,9
Sportbauten	1,1
Schulgebäude	1,3
Büro-/Verwaltungsbauten I* ** = (Kleinere Bauten ohne Repräsentationsanspruch)*	1,3
Feuerwehrgebäude	1,4
Büro-/Verwaltungsbauten II** *** = (Großbauten mit Repräsentationsanspruch)*	1,6
Sonstige Gebäudetypen	1,0

Abbildung 10, Jahreskorrekturfaktoren der Gebäudenutzungsart nach AMEV

[33] (Arbeitskreis Maschinen- und Elektrotechnik BMWSB (AMEV), 2013), S. 22 & 23

Gebäudehöhe	$f_{Kor\_Höhe}$
Kein Hochhaus* * ... in der Regel 1-8 Vollgeschosse	1,0
Hochhaus** ** ... in der Regel mehr als 8 Vollgeschosse	1,7

Abbildung 11, Jahreskorrekturfaktoren der Gebäudehöhe nach AMEV

Aus diesen Einzelfaktoren ergibt sich der Korrekturfaktor:

$$f_{Kor_{Jkf}} = f_{Kor_{Alter}} * f_{Kor_{Nutz}} * f_{Kor_{Höhe}} \quad \textbf{(30)}$$

4.2.4.3 Berechnung der Jahresgesamtkosten für die operativen Aufgaben

Mit der Inbezugnahme von Jahreskostenfaktoren werden zuletzt die jährlichen Kosten der operativen Aufgaben erhoben. Anwendbare Kostenfaktoren sind der Tabelle 4 der AMEV-Empfehlung zu entnehmen.[34] Bis zum Zeitpunkt verfügbarer Herstellungskosten von einzelnen Bauteilelementen sollten hier die Kostenfaktoren nach Nutzung angewandt werden.

Es gilt

$$K'_{op_{WIB}} = \sum W_N * f_1 * f_{Kor_{Jkf}} \quad \textbf{(31)}$$

für die Berechnung der operativen Kosten für Wartung, Inspektion und Bedienung und somit der Kostengruppe 351 und 353. Für die operativen Kosten der Instandsetzung und somit der Kostengruppe 420 gilt:

$$K'_{op_{In}} = \sum W_N * f_2 * f_{Kor_{Jkf}} \quad \textbf{(32)}$$

[34] (Arbeitskreis Maschinen- und Elektrotechnik BMWSB (AMEV), 2013), S.27 & 28, Tabelle 4 & 5

Da angenommen wird, dass die Berechnung der Nutzungskosten auf einzelne Gebäude und nicht auf eine gesamte Liegenschaft angewandt wird, wird von der Verrechnung der Wegekosten abgesehen.

4.2.4.4 Abschätzung der administrativen Kosten

Abhängig vom Fremdvergabeanteil werden die Kosten für administrative Aufgaben mithilfe eines weiteren Faktors ermittelt. Dieser liegt zwischen 10 % bei 0% Fremdvergabeanteil und 20 % bei 100% Fremdvergabe. Zwischenwerte werden interpoliert.[35]

$$K_{ad} = f_{ad} * K'_{op} \quad \textbf{(33)}$$

4.2.4.5 Berechnung der Gesamtkosten

Zuletzt werden die errechneten operativen mit den administrativen Kosten addiert.

$$K_{ges} = K_{op} + K_{ad} \quad \textbf{(34)}$$

In Anlage 4 findet sich ein entsprechender Python-Code, mit welchem ein Überschlag der Kosten für die Kostengruppen 350 und 400 unter Angabe von händischen Eingaben liefert.

4.2.4.6 Berechnung in der Ausführungsplanung

Zum Zeitpunkt der Ausführungsplanung sollten die Herstellungskosten der Bauteile hinterlegt sein. Bestenfalls werden entsprechende Daten direkt von den Bauteilherstellern bezogen. Sind die spezifischen Produkte noch nicht festgelegt, ist die Eintragung von Richtwerten zu empfehlen. Das Nutzen einer Datenbank aus Platzhalterelementen, welche besagte Richtwerte bereits mit sich bringen, stellt den Optimalfall dar. Da, wie vorangehend erwähnt, für die Kostengruppen der DIN 276 von Bauteilen unterschiedliche Jahreskostenfaktoren angesetzt werden und die Berechnung der Kosten für die Kostengruppen 350 und 400 nach DIN 18960 je nach Möglichkeit außerhalb des Authoring-Werkzeuges durchgeführt wird, ist eine Gliederung der Bauteilherstellungskosten nach Kostengruppen der DIN 276 zu empfehlen. So

[35] (Arbeitskreis Maschinen- und Elektrotechnik BMWSB (AMEV), 2013), S. 24

können die ausgegebenen Kosten entsprechend der Jahreskostenfaktoren getrennt berechnet werden. Sollen die Kosten bis zur 3. Kostenebene ausgegeben werden, ist eine weitere Trennung der ausgegebenen Herstellungskosten nach diesen erforderlich. Gemeinhin können die Kostengruppen 352, 354, 355, 410, 430 und 440 nur mit einbezogen werden, wenn Jahreskostenfaktoren für diese bekannt sind. Eine Erhebung dieser hätte den zeitlichen Rahmen der Abschlussarbeit gesprengt, weshalb auf die Inbezugnahme dieser Kostengruppen verzichtet wurde.

4.2.5 KG 360 - Sicherheits- und Überwachungsdienste

Die Planung für Sicherheits- und Überwachungsdienste erfolgt über eine Vielzahl von Faktoren und ist im Zuge der Ausführungsplanung noch nicht möglich. Erst nach Festlegung der Sicherheitsanforderungen, welche einen hohen Grad der Individualität eines Planungsprojektes und einer darauf aufbauenden Planung von Kontrollgängen zugrunde liegen, ist eine Ermittlung des personellen Aufwandes möglich und sind in einem digitalen Planungsmodell nur schwer abzubilden. Kosten für die Unterhaltung von Sicherheitssystemen wie Überwachungskameras dagegen können über die Kostengruppen 316, 353, 425 und 434 im Sinne der Stromversorgung, der Wartung und Instandsetzung abgebildet werden.

4.2.6 KG 370 - Abgaben und Beiträge

Ebenso wie im Fall der KG 360, sollten die Abgaben und Beiträge nicht auf Grundlage von Modelldaten der Planung bezogen werden. Neben der Tatsache, dass sich die Planungsdaten stark vom Zustand der Bauausführung unterscheiden können, dieser erst in das Modell nachgearbeitet werden muss und präzise Daten so erst nach dem Gutachten der Versicherung im Modell vorhanden sind, läge des Weiteren die Verantwortung für Fehleinschätzungen der Rahmenbedingungen für Versicherungsgutachten bei dem planenden Unternehmen, was so definitiv nicht umzusetzen ist, da das Planungsunternehmen keinen direkten Einfluss auf den tatsächliche Qualität der Bauausführung hat. Generell sollte man den Planungsdaten ihren Nutzen zur Umfangsermittlung der KG 370 jedoch nicht abstreiten. So sollten weniger variable Daten der Planung, wie zum Beispiel Flächenwerte, deren Ursprung im digitalen Modell liegen, zum Gutachten hinzuziehbar und unter ständiger Kommunikation nachzupflegen sein. Voraussetzung ist dafür selbstredet ein verifizierbarer Abgleich der Soll- und Ist-Daten. Bei wem die Verantwortung zur Pflege der Planungsdaten liegt, ist ebenso festzulegen. Aktuell findet sich dazu keine direkte Zuständigkeit – Planungsdaten werden den ausführenden Auftragnehmern und dem

Bauherrn übergeben, jedoch nicht immer über die Ausführungsphase hinaus genutzt. Hieraus ergibt sich ein mögliches Geschäftsfeld von Verwaltungsunternehmen, worin bislang jedoch keine Notwendigkeit, aufgrund der unabsehbaren Wirtschaftlichkeit, ersichtlich scheint. Eine Überlegung wäre, eine Art Wartungsvertrag zu etablieren, welcher die Pflege und Weiterführung der Planungsmodelle im Rahmen der Nutzungsphase inkludiert. Im Idealbild der BIM-Arbeitsweise sollte das BIM-Modell/Koordinationsmodell als zentrales Kommunikationsmittel rund um die Unterhaltung eines Gebäudes dienen. Dafür müssen Schnittstellen für Verantwortlichkeiten definiert werden, um die Daten bspw. im Falle einer Sanierung aktuell zu halten. Dieser Ansatz spiegelt Möglichkeiten zur Fortentwicklung der Untersuchung wider.

4.3 Nachvollziehbare Bauteilverknüpfungen

4.3.1 Verknüpfung der Parameter und Bauteile

Wie bereits erwähnt, wird zur Übersicht der im Modell geforderten Parameter und deren Anforderungen eine Attributmatrix entwickelt. Im Fall von Autodesk Revit werden die Bauteile entsprechend der erwähnten internen Bauteilkategorien sortiert und Parameter nach diesen zugeordnet. Eine Parameterzuordnung kann auch über die Elementfamilien geschehen, stellt aber einen erheblichen Mehraufwand dar, da sämtliche Familien angepasst werden müssten.

Für die weitergehende Kategorisierung kann der Predefined Type des IFC-Schemas herangezogen werden. Die Einarbeitung von Predefined Types ist insbesondere für die Bauteilkategorien HLS-Bauteile, Elektroinstallationen und Elektrische Ausstattung zu empfehlen, da diese Revit-Kategorien sehr verallgemeinert bezeichnet sind und Bauteile dieser Kategorien unterschiedliche Anforderungen an die Kostenermittlung bringen können. Eine weitere Möglichkeit ist die Entwicklung und Zuweisung von Bauteilkennzeichen/-schlüsseln, mithilfe derer eine Filterung der Bauteile nach den Anforderungen der Berechnung der betriebsrelevanten Daten möglich ist. Auf diesen Sachverhalt wird in Kapitel 5.2 – Umsetzung der Kostenermittlung näher eingegangen.

Es ist zu erwarten, dass Parameter bei der Zuordnung nach Revit-Kategorie nicht auf alle Bauteile der Kategorie anzuwenden sind. Im Falle dessen sind null-Werte zu definieren, welche in der Auswertung nicht zu berücksichtigen sind, da die

Parameter abhängig von der Entität und nicht vom Predefined Type beim IFC-Export mitgegeben werden.

Abhängig vom Datentyp der Parameter werden folgende null-Werte definiert:

- Datentyp Zahl: *„999999“*
- Datentyp Text: „*N/A“*
- Datentyp Boole: nicht definiert *(null)* oder *“False“*

Findet die Auswertung nicht über die IFC-Datei, sondern über eine individuelle Ausgabe statt, ist dies nicht zwangsweise zu berücksichtigen, da entsprechende Parameterwerte von vorhinein ausgeschlossen werden können.

Die für die Untersuchung definierte Attributmatrix ist in Anlage 1 beigefügt.

4.3.2 Verknüpfung der Bauteile mit deren realen Gegenstücken

Für eine transparente, nachvollziehbare Mengenermittlung aus einem Planungsmodell, ist eine eindeutige Zuordnung der digitalen Bauteile zu deren realen Gegenstücken unabdingbar. Dies dient nicht zuletzt zur Datenpflege des BIM-Modells während der Nutzungsphase.

Revit als Beispiel vergibt intern sogenannte Element-IDs. Diese Identifikationszeichenfolgen werden in Laufzeit erzeugt und dienen zur Identifikation von Bauteilen, Parametern und anderen Datensätzen des Projektes. Die Element-ID ist eine für den Menschen leserliche, 6-stellige Nummer, welche in der Theorie auch mit einem Export per IFC mitgegeben wird und zur Bauteilidentifikation dienen kann. Praktisch ist es jedoch nicht zu empfehlen, diese Nummer zur Identifikation für weitere Anwendungen zu nutzen, da sich die Nummern beispielsweise bei einem Löschen und Neuerstellen von Bauteilen verändert und nicht ohne Weiteres manipuliert werden kann.

Aus diesem Grund sollte eine eigene Semantik erdacht werden, um reale Bauteile mit ihren digitalen Gegenstücken zu verknüpfen. Ein eigener Bauteilschlüssel kann dann in einem eigenen Parameter hinterlegt werden und bei einem Austausch von Objekten erneut vergeben werden. Eigene Bauteilkennzeichen können des Weiteren beispielsweise per QR-Code auf eine Plakette des realen Bauteils gebracht werden, um eine schnell zu prüfende, direkte Verknüpfung zum Planungsmodell herzustellen.

4.4 Modellbasierte Auswertung von Bezugsgröße

4.4.1 Anforderungen an Vollständigkeit und Nachvollziehbarkeit

Um ein möglichst exaktes Berechnungsergebnis auf Basis der festgelegten Anforderungen zu gewährleisten, ist es unabdingbar, sämtliche zur Berechnung relevanten Bauteile identifizieren zu können und alle Werte zur Verfügung stehen zu haben.

Entsprechende Anforderungen sind im BAP festzulegen und diesem auch während der Planung zu entnehmen. Allgemein müssen die Ein- und Ausgangswerte der Auswertung und Mengenermittlung absolut definiert sein, sodass die Ergebnisse derer nicht falsch interpretiert werden. Eine Dokumentation der tatsächlich ermittelten Kosten und der grundlegenden Datenannahme muss in jedem Fall vorhanden sein, sodass etwaige Anpassungen an abweichende Anforderungen und das Erweitern des Ermittlungsprozesses möglich sind.

Die Vorgehensweise einer modellbasierten Auswertung ist von der dafür verwendeten Software abhängig, da sich die Funktionsweisen und die generelle Methodik von Software stark unterscheiden.

Unter Verwendung von Autodesk Revit gibt es einige Rahmenbedingungen, die während des Prozesses beachtet werden müssen. So erfolgt die Zuordnung von Bauteilen, wie bereits erläutert, in Bauteilkategorien und können bei der Erstellung neuer Bauteile vom Konstrukteur definiert werden – dies gilt als erster Schritt der Klassifizierung innerhalb der Konstruktionssoftware und bedarf einer ebenso konsequenten Durchführung wie die Definition der Parameter. Herkömmlich werden zur Auswertung der Bauteildaten Bauteillisten erstellt, welche sämtliche Exemplare einer Bauteilkategorie mitsamt der gewünschten Parameterdaten auflistet. Hier sind bereits Methoden verfügbar, die beispielsweise das Aufsummieren von Werten möglich machen. Eine direkte Zuordnung der Bauteile zu den Räumen, in denen sie sich befinden ist jedoch nicht ohne weiteres möglich.

Aufgrund der erhöhten Flexibilität und möglicher Korrekturen während der Datenerhebung wird sich dafür entschieden, das Revit-interne Visual-Scripting Add-On „Dynamo“ zu verwenden. Mithilfe dieses Add-Ons ist es möglich, Bauteile auf Basis ihrer geometrischen Lage zu Räumen zuzuordnen und entsprechende Raumparameter mit entsprechenden Bauteildaten zu befüllen. Weiterhin besteht die Möglichkeit, ohne zusätzliche Software die in Kapitel 4.1 – Kostenermittlung und Bezugsgrößen genannten Formeln umzusetzen – also die Werte der Parameter zu

ermitteln. Außerdem ist es mit dem Dynamo-Player möglich, Eingabemasken zur Verfügung zu stellen, um dem für die Auswertung Verantwortlichen Eingabemöglichkeiten für Werte zu bieten, welche auf externen Informationen beruhen. Ein Beispiel dafür wäre der Fremdvergabeanteil oder aktuelle Versorgerpreise von Medien. Die per Dynamo ermittelten Daten und resultierende Ergebnisse können daraufhin als *.csv- oder *.xlsx-Datei für weitere Auswertungsschritte ausgegeben werden.

Der Aufbau entsprechender Skripte und der darauffolgenden Auswertung per Excel und Python wird in Kapitel 5.2 erläutert.

4.4.1.1 Nicht-geometrische Anforderungen

Im Rahmen einer Qualitätsprüfung sollte sichergestellt werden, dass alle Parameter ausgefüllt sind und die eingetragenen Werte im definierten Bereich gesetzt werden.

Je nach Datentyp eines Parameters müssen also Festlegungen über dessen Inhalt getroffen werden. Parameter, deren Werte einer bestimmten Einheit entsprechen müssen konsequent mit diesen gefüllt werden. Am Beispiel des Parameters zum Wasserverbrauch pro Raum muss also sichergestellt werden, dass die eingetragenen Werte nicht in Liter pro Stunde, sondern in Kubikmeter pro Stunde angegeben werden, sodass bei der weiteren Auswertung Fehleintragungen das Ergebnis nicht verfälschen. Auch Eigenschaften wie die Funktion von Fenstern und Türen sollten korrekt hinterlegt sein, da dies beispielsweise für durchzuführende Skripte bezüglich der zu errechnenden Glasflächen Auswirkungen haben kann. Selbstredend steigt im Allgemeinen die Aussagekraft der Auswertung mit der Informationstiefe und Exaktheit der Modelldaten.

Neben den Bauteilen besitzen auch Parameter eine eindeutige interne ID. Um zu garantieren, dass in sämtlichen Modellen und Arbeitsschritten mit denselben Parametern gearbeitet wird und keine Dopplungen auftreten werden so genannte gemeinsam genutzte Parameter verwendet. Dabei handelt es sich um eine Textdatei, welche eine Sammlung von Parametern mit deren Eigenschaften und eindeutigen Identifikationsnummern beinhaltet. Eine solche Datei der in Kapitel 4.1 genannten Parameter wird per Revit generiert.

4.4.1.2 Anforderungen an die Geometrie

Eine korrekte Geometrie eines Planungsmodells ist für die Auswertung nicht weniger wichtig als deren verknüpfte Daten. Um eine Zuordnung der Bauteile zu deren umschließenden Räume zu ermöglichen, ist es wichtig, dass entsprechende Bauteil korrekt positioniert sind und sich nicht mit bspw. Wänden oder Geschossdecken schneiden.

Die Raumgeometrie muss ebenfalls in einem definierten Rahmen erstellt werden: Wird die Höhe der Raumgeometrie mit der lichten Höhe gleichgesetzt, beziehungsweise ist die Oberkante der Raumgeometrie abschließend mit einer eventuellen Abhangdecke oder mit der Rohdecke / ist die Raumunterkante bis unter den Fußbodenaufbau definiert oder liegt sie auf diesem? Wie wird mit Verkehrs- und Konstruktionsflächen umgegangen? Wo liegt die Berechnungshöhe des Raumes? All diese Fragen sind ausschlaggebend für die Auswertung der Rauminformationen und sollten im BAP beantwortet werden.

Da in Revit ein Unterschied zwischen Räumen der Architektur und Räumen der technischen Ausstattung (sog. MEP-Räume) gemacht wird, muss garantiert sein, dass diese jeweiligen Räume deckungsgleich angelegt werden und deren Einfügepunkt relativ mittig zum Raum liegt. Durch auftretende Änderungen der Planung und so auch der Geometrie von Räumen ist ein regelmäßiger Abgleich der Modelle über Modellverknüpfungen durchzuführen, sodass die MEP-Räume identisch mit den Architekturräumen sind, welche sich natürlich bei Änderungen der Wandaufbauten und Raumteilungen ebenfalls ändern. Auch Daten wie Raumnummern und -namen sollten stets abgeglichen werden. Da für Außenanlagen für keine Räume angelegt werden können, ist ein Flächenplan für weitere oder abweichende Raumeinteilung nötig. Über diesen können Zuordnungen und Verortungen von Bauteilen ebenfalls durchgeführt werden. Grundlegend relevant für das Zusammenführen der Fachmodelle sind übereinstimmend angelegte Weltkoordinaten, sodass die Modelle exakt übereinander liegen und so ein plausibles Koordinationsmodell ergeben.

4.4.2 Arbeitsablauf und Prozessschritte der softwarespezifischen Anwendung

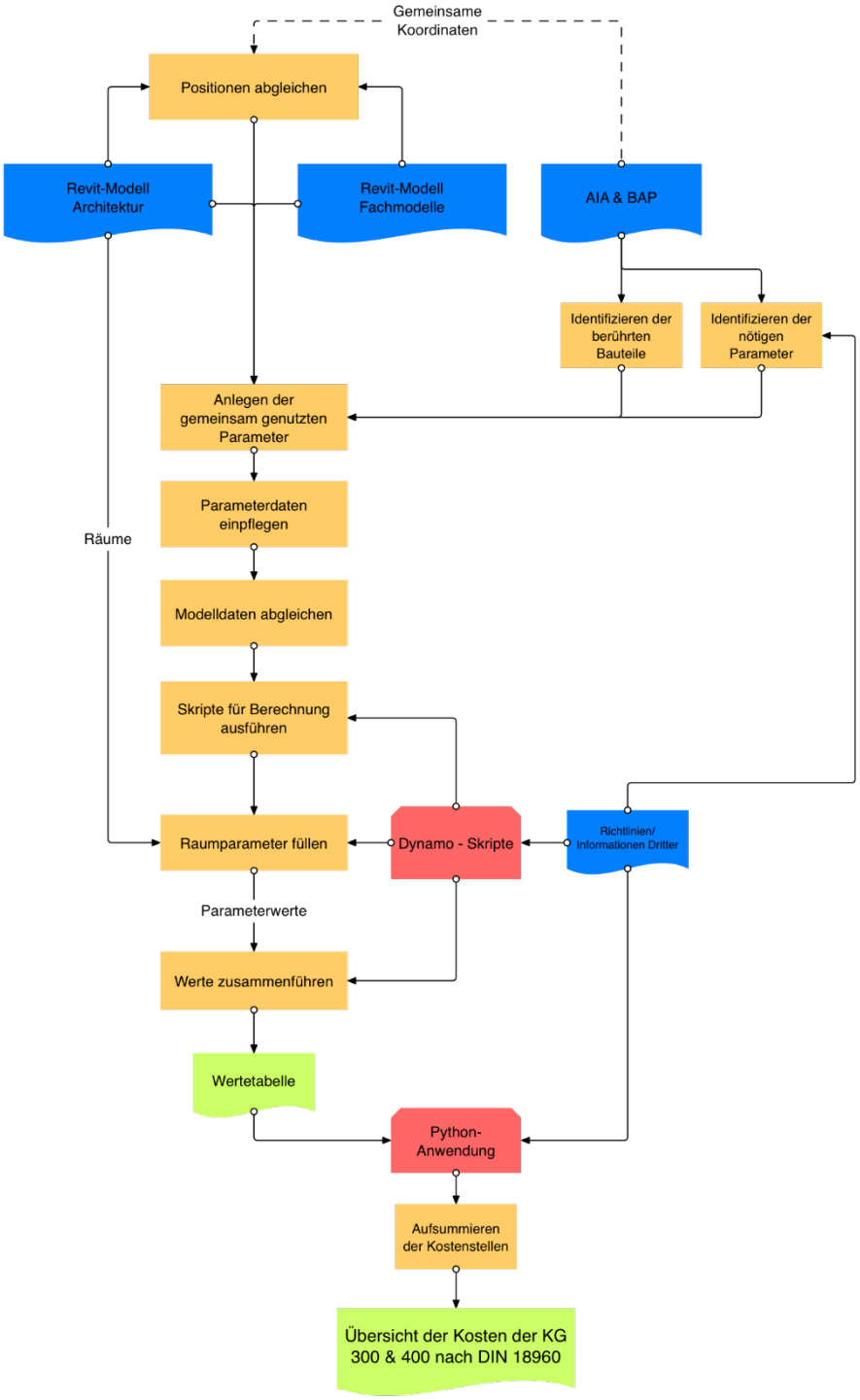


Abbildung 12, Untersuchungsspezifischer Arbeitsablauf, eigene Darstellung

Der in Abbildung 11 dargestellte Prozessablauf ist auf Grundlage der in der Abschlussarbeit festgelegten verwendeten Softwarelösungen modelliert. Im ersten Schritt sollten die grundlegenden Anforderungen an die modellbasierte Nutzungskostenermittlung eingesehen werden. Wichtig ist hier, welche Kostengruppen abgedeckt werden sollen. Im Vorfeld sollten Kostengruppen, deren Erhebung nicht im Planungsmodell erfolgen kann, ausgeschlossen werden.

Sind die Kostengruppen bekannt, muss festgestellt werden, welche Bauteile und zugehörige Parameter anzulegen sind. Ausschlaggebend sind die in Kapitel 4.2 gelisteten Formeln. Auf diesen Formeln aufbauend, ist eine Attributmatrix zu erstellen, aus welcher die Bauteile und deren Parameter abgelesen werden. In dieser Matrix können auch Zeitpunkte festgelegt werden, zu welchen bestimmte Parameter ausgefüllt sein müssen. Davon ausgehend, dass sämtliche Planungsmodelle angelegt worden, müssen nun sämtliche Parameter in diesen angelegt werden. Innerhalb Autodesk Revit werden diese Parameter über sogenannte "gemeinsame Parameter“ angelegt. Der Vorteil dieser besteht darin, dass diese mit dem Anlegen in einer Text-Datei abgespeichert werden und daraufhin in Folgeprojekten identisch eingebunden werden können.

Bei Verwendung mehrerer Fachmodelle ist ein Positionsabgleich unabdinglich. Nicht nur im Rahmen einer ordnungsgemäßen Planung ist dies notwendig, sondern insbesondere unter der Anwendung der in Kapitel 5.2 erläuterten Dynamo-Skripte. Da die Zuordnung der Bauteile zu deren Räumen und Flächen über die Positionskoordinaten erfolgt, ist es unbedingt erforderlich, dass sämtliche Fachmodelle deckungsgleich positioniert sind. Um das zu garantieren, sind im BAP gemeinsame Koordinaten festgelegt, welche über eine Festlegung eines Vermessungs- und eines Basispunktes auf dem tatsächlichen Grundstück definiert sind. Beide dieser Punkte werden nach deren Weltkoordinaten verortet und deren Bezug zueinander ebenso definiert. In Relation zu den gemeinsamen Koordinaten werden sämtliche Fachmodelle beplant.

Ist der Positionsabgleich durchgeführt, werden nun die angelegten Element-Parameter befüllt, wenn nicht bereits durch eine vorhandene Datenbank geschehen. Während der Datenpflege, welche zumeist durch viele Mitarbeiter verschiedener Fachbereiche durchgeführt wird, ist ein Abgleich der Arbeitsstände in regelmäßigen Abständen durchzuführen, sodass stets mit aktuellen Daten gearbeitet wird. Wird die Berechnung der Nutzungskosten angestoßen, ist dieser Abgleich im Vorfeld durchzuführen. Dies geschieht im Fachmodell der Architektur, da sich in diesem die Raumelemente befinden, in welchen die verknüpften Sammelparameter mit den

Ergebnissen der Berechnungen befüllt werden. In diesem Fall erfolgt dies über erarbeitete Dynamo-Skripte. Über ein weiteres Skript werden die Sammelparameter zusammengeführt und in Form von Tabellen exportiert.

Zur Verrechnung mit weiteren Faktoren und externen Daten, welche nicht im Planungsmodell hinterlegt sind, werden die Tabellendaten an eine Python-Anwendung überführt und resultierende Ergebnisse zusammengefasst ausgegeben. Mit jeder neuen Ergebnislieferung, welche den aktuellen Planungsstand widerspiegeln sollen, sind entsprechend des in Kapitel 3.2 aufgeführten, iterativen Prozessablaufs Arbeitsschritte durchzuführen. Demzufolge muss, bezugnehmend auf die hier erdachte Abfolge, erneut der Datenabgleich der Fachmodelle, sämtliche Dynamo-Skripte ausgeführt und die Überführung in – und Ausführung der Python-Anwendung erfolgen.

5. Prototypische Implementierung

5.1 Praktisches Beispiel

Um die grundlegende Vorgehensweise bei der softwarespezifischen Auswertungsmethodik zu veranschaulichen ist es ratsam, ein Planungsbeispiel anzuwenden, welches unter Laborbedingungen allein für den Zweck der Betriebskostenermittlung entwickelt wird. So kann garantiert werden, dass sämtliche Voraussetzungen für eine fehlerfreie Auswertung erfüllt sind. Aktuelle reale Planungsdaten können zwar testweise zur Durchführung verwendet werden, jedoch sind Fehler in diesem Fall nicht auszuschließen, da nicht nur die angewandte Modellierungsrichtlinie nicht an den Zweck der Betriebskostenermittlung angepasst sein wird, sondern auch die Planungsumsetzung im Hinblick auf die schiere Anzahl der am Projekt beteiligten Planer nicht die nötige Konsistenz aufweist. Die Voraussetzungen müssen entsprechend im Vorfeld der Planung definiert und umgesetzt werden und sind im Sinne der nachträglichen Anpassung nicht wirtschaftlich umsetzbar. Ein veranschaulichendes Beispiel hierfür ist beispielsweise die Verwendung von Bauteilgruppen. Diese sind in Revit als eigene Kategorie definiert und stellen eine Art Container-Objekt verschiedener, zusammenhängender Bauteile dar. Bauteilgruppen können Bauteile sämtlicher Art beinhalten und werden häufig zur Erleichterung der Planungsaufgabe verwendet. Diese in der Gruppe befindlichen Bauteile sind über Dynamo nicht direkt abrufbar, was im Falle der differenzierten Auswertungsmethoden verschiedener Bauteile äußerst ungünstig ist.

Zu erwähnen ist hierbei, dass benannte Bauteilgruppen beim Export in das IFC-Austauschformat aufgelöst werden und die Bauteile einzeln abrufbar werden. Folglich stellt sich die Frage, ob eine Kostenberechnung mit der IFC-Datei als Quelldatei im Vergleich zur in dieser Untersuchung gewählten Herangehensweise nicht als sinnvoller einzustufen ist. Dazu sei gesagt, dass das Lesen von IFC-Daten einen programmiertechnisch sehr hohen Aufwand bedeutet. Zwar bestehen bereits diverse Softwarelösungen zur Auswertung und Betrachtung von IFC-Daten, jedoch sind diese größtenteils auf die Qualitätssicherung der Daten oder Mengenermittlungen ausgelegt und bieten keine Möglichkeit zur Manipulation oder weiterführenden Berechnung unter Verwendung zusätzlicher, externer Daten. Der hier erarbeitete Prozess, stellt dementsprechend eine Grundlage für die Entwicklung einer Softwarelösung zur Erhebung der Betriebskosten auf Grundlage von IFC-Daten dar.

Die ordnungsgemäße Umsetzung der Arbeitsschritte zur Kostenermittlung für den Betrieb der Immobilie ist folglich nur mit entsprechender Vorplanung und Einhaltung der Richtlinien zur Modellierung und Planung möglich. Um diese Anforderungen zu eruieren, wird die Umsetzung an einem konzeptionellen Beispielprojekt erfolgen. Dabei handelt es sich um ein kleines Bürogebäude mit einer Bruttogeschossfläche von ~1530 m², dessen Detailierung bezüglich der Planung der baulichen Ausführung der Leistungsphase 2+ entsprechen dürfte. Die eingetragenen Werte der angelegten Bauteile entsprechen in Kapitel 4.2 genannten, extern bezogenen Richtwerten aus Normativen und von Dritten veröffentlichten Vergleichswerten.

5.2 Umsetzung der Kostenermittlung

5.2.1 Dynamo-Skripte zur Berechnung der jährlichen Mengendaten

Unter der Voraussetzung, dass die Eintragungen aller betriebskostenrelevanten Parameter erfolgt ist, kann begonnen werden, Dynamo-Skripte zur Berechnung der Sammelparameter zu entwickeln oder zu implementieren, sollte die Entwicklung bereits erfolgt sein. Entsprechend der nötigen Element-Abfragen und dem Berechnungsablauf, werden mehrere Skripte angelegt. Dies ist nötig, da die Verwendung von nur einem Skript, welches sämtliche Berechnungen abdeckt, die Berechnungsdauer stark erhöht, sollten nur kleine Änderungen an den Planungsdaten erfolgt sein, würden mit jeder Ausführung sämtliche Werte neu berechnet werden und so einen unnötigen Mehraufwand bedeuten. Durch die Aufteilung der Berechnung in mehrere Skripte kann so Rechenzeit gespart werden. Weiterhin wurde die Unterteilung anhand der abzurufenden Elemente festgelegt, sodass Elementabfragen so sparsam wie möglich durchgeführt werden und wenn möglich ohne Dopplungen vorkommen.

Grundlage der Skripte bildet die Zuordnung von Bauteilen zu deren Verortung, um berechnete Daten den korrekten Sammelparametern zuzuordnen. Unterschieden wird dabei zwischen der Raum- und der Flächenzuordnung. Folgend wird beispielhaft anhand der Berechnung des Wasserverbrauchs für Sanitäreinrichtungen die Vorgehensweise der Skripte beschrieben. Grundsätzlich bleibt die Vorgehensweise für andere Kostengruppen identisch, es unterscheiden sich lediglich die hinzugezogenen Elemente und die in Kapitel 4.2 beschriebenen Berechnungsformeln. Es handelt sich hierbei um Arbeitsstände und es wird kein Anspruch auf Vollständigkeit erhoben.

Schritt 1: **Sammeln aller Elemente der jeweiligen Kategorien (2) und Sammeln aller im Projekt angelegten Räume (1).** Wichtig hierbei ist die Feststellung des Ursprungs – welche Bauteile finden sich in welchen verknüpften Fachmodellen (3) und welche liegen in dem Modell, in welchem das Skript ausgeführt wird, beziehungsweise die Sammelparameter angelegt wurden. Durch das Vorhandensein der Raum-Elemente im Architekturmodell, bietet sich dieses als „Sammelstelle“ an.

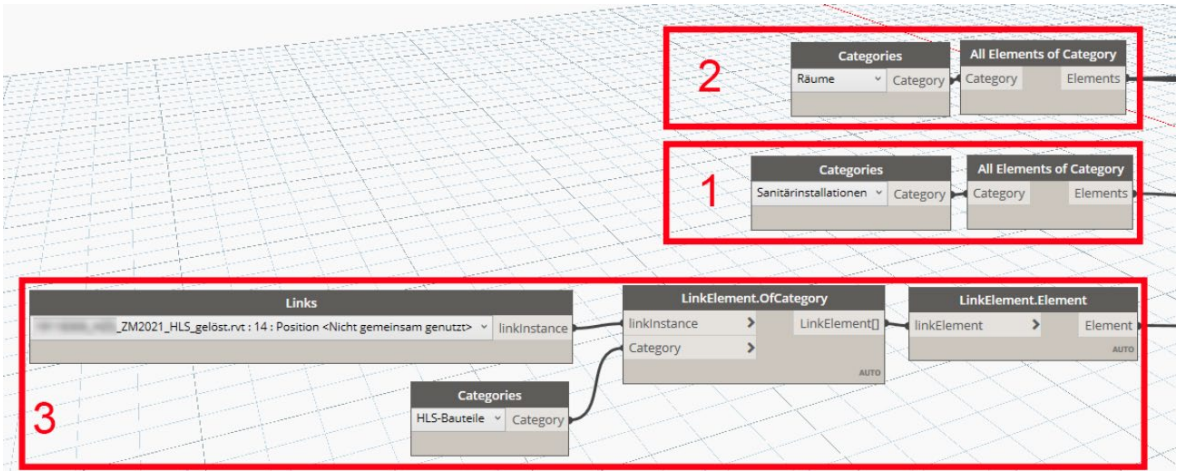


Abbildung 13, Dynamo - Sammeln von Elementen

Schritt 2: Sortieren/Filtern der Bauteile nach Berechnungsgrundlage. Für die Berechnung des Wasserverbrauchs sind beispielsweise Bauteile der Kategorie Sanitärobjekte relevant. Deren Verbrauch wird unterschiedlich errechnet, so ergibt sich der Wasserverbrauch für WCs nach dem Verbrauch pro Nutzung (Spülung) und der von Waschbecken nach Nutzungsdauer mal dem Durchfluss pro Sekunde. Diese Sanitärobjekte müssen also unterschieden werden.

Zu dieser Sortierung kann im Idealfall der Familienname hinzugezogen werden. Die Benennung muss dazu eindeutig sein. Unterschiedliche Ausführungen von Bauteilen können verschiedenen Familien entstammen, weshalb sich die Familiennamen eine eindeutige Kennung teilen sollten. Sämtliche WCs sollten also bspw. die Zeichenfolge „WC“ beinhalten. Bauteile, die zu einem WC gehören, jedoch kein WC als solches darstellen, sollten die Bezeichnung WC somit nicht im Namen stehen haben. Dementsprechend muss in diesem Fall eine Familienbibliothek vorhanden sein oder angelegt werden, in welcher die Familiennamen dieser Maßgabe folgen. Zur Erstellung einer solchen Bibliothek ist nicht allein aufgrund der immensen Vielfalt von Bauteilherstellern und -ausführungen projektspezifische Entwicklungsarbeit vonnöten. Für das Anlegen einer solchen Bibliothek werden die meisten

Planungsunternehmen nicht über die nötigen Kapazitäten verfügen. Im Allgemeinen ist die Wirtschaftlichkeit der Erstellung einer firmeneigenen Datenbank als fragwürdig zu betrachten. Im Regelfall werden für Familien oft Geometriedaten direkt von den Herstellern verwendet und importiert, welche selbstredend keiner firmeninternen Namenskonvention folgen. Daraus folgt, dass für eine präzise und relativ einfache Markierung bestimmter Bauteilgruppen ein bezeichnender Parameter angelegt werden muss.

Wie bereits erwähnt bietet das IFC-Schema ebenfalls eine Möglichkeit der Zuordnung von Bauteilen über die IFC-Entität und einen weiter untergliedernden Predefined Type. Zur Weiterverarbeitung und dem softwareunabhängigen Austausch stellt dies eine sinnvolle Alternative zu spezifischen Familiennamen dar. Während es Vorgaben nach IFC-Schema für die vordefinierten Typen gibt, können diese um eigene Typen erweitert werden, sollten diese nicht im Schema vorhanden sein. Als unvorteilhaft ergibt sich hier jedoch die Tatsache, dass eigens angelegte Predefined Types in einem anderen Parameter zu hinterlegen sind als Typen, die dem Schema entspringen. Zur Definition der Typen wird der vordefinierte Parameter „IfcExportType“ befüllt. Sollen individuelle Typen vergeben werden, wird dieser Parameter mit dem Wert „USERDEFINED“ belegt und der entsprechende Typ in den Parametern „IfcObjectType“ oder „IfcElementType“ festgelegt. Für die Revit-interne Sortierung der Bauteile per Dynamo-Skript ist es vorteilhaft, die Zuordnungsinformation aus einem gleichbleibenden Parameter beziehen zu können, da sonst eine zusätzliche Abfrage prüfen muss, ob der Typ individuell definiert wurde. Infolgedessen wird auch diese Variante vorerst ausgeschlossen.

Eine vielversprechende Möglichkeit der eindeutigen Zuordnung von Bauteilen bietet die Verwendung von Baugruppenkennzeichen. Diese sind einmalig festzulegen und daraufhin an die Familientypen der Bauteile zu vergeben. Die Kennzeichen unterliegen ebenfalls einer Ebenenstruktur, wodurch je nach Gliederungsanforderungen gefiltert werden kann. Um bei dem vorangehenden Beispiel zu bleiben: Sanitärinstallationen werden im Fall der firmeninternen Anlage mit der Zeichenfolge „SA“ gekennzeichnet, gefolgt von weiteren zweistelligen Zeichenfolgen, getrennt durch einen Unterstrich. Die Kennzeichen für die Unterscheidung von WCs, Waschbecken und Duschen sind dementsprechend wie folgt festgelegt:

Duschen: **SA_DU**;
Vorwandinstallation Urinal: **SA_VW_UR**;
Vorwandinstallation WC: **SA_VW_WC**;
Waschbecken: **SA_WB**;

Waschbecken barrierefrei: **SA_WB_BF**;

…

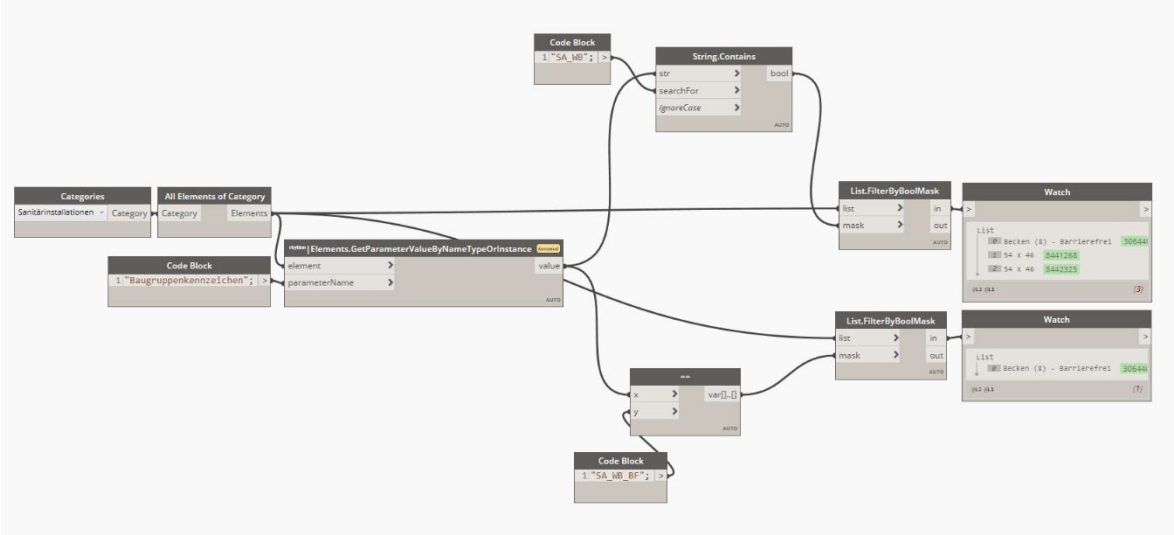


Abbildung 14, Dynamo - Sortieren und Filtern von Elementen nach Baugruppenkennzeichen

Auf diese Weise sind sehr präzise Filterungen möglich, da sämtliche Waschbecken mit der Suche nach dem Vorkommen der Zeichenfolge „SA_WB“ abgedeckt werden. Sind jedoch nur barrierefreie Waschbecken gesucht, wird der Filter um den Wert „_BF“ erweitert. Werden nur nicht-barrierefreie Waschbecken gesucht, kann der Filter von „Beinhalten“ des Wertes auf „Ist gleich“ der Wert gesetzt werden.

Schritt 3: Zuordnung der Elemente zu deren Verortung. Zu diesem Zweck werden zuerst die Einfügepositionen der Elemente als Punkt-Koordinaten abgerufen (1), weiterhin wird die Geometrie, also der Volumenkörper des Raumes oder von Flächen abgerufen. Flächen werden in diesem Fall in Höhe einer Etage extrudiert, um ebenfalls einen Volumenkörper zu erzeugen. Es entstehen damit zwei Listen, einmal eine Liste aller Einfügepunkte und eine Liste aller Volumenkörper. Es wird nun geprüft, ob sich ein Einfügepunkt, innerhalb der Geometrie des Raumes X befindet, ist dies der Fall, wird der boolesche Wert „true“ zurückgegeben. Es wird jeder Punkt mit jedem Raum abgeglichen. Es entsteht eine verschachtelte Liste – also eine Liste aus Listen. Jeder Eintrag stellt nun also einen Raum dar, welcher wiederum die Liste aller Bauteile beinhaltet, jedoch nicht mit den Bauteilen als Wert, sondern dem Wert des im Raum Vorhandenseins dessen (2). Es können nun die Indexwerte der „Wahren“ Werte ausgelesen werden (3), die entsprechenden Indices werden daraufhin aus der Liste der Bauteile gezogen. Das Resultat ist eine Liste aus Listen, welche eine Liste der Räume, mitsamt der in Ihnen befindlichen

Elemente umfasst (4). Die Abbildung zeigt: In Raum des Index 13 ist das Element mit dem Listenindex 6 mit „true“ gekennzeichnet, ist also in diesem Raum verortet. Im letzten Schritt wird also das Element mit dem Index 6 der Bauteilliste ausgelesen und dem Raum des Listenindex 13 zugeordnet.

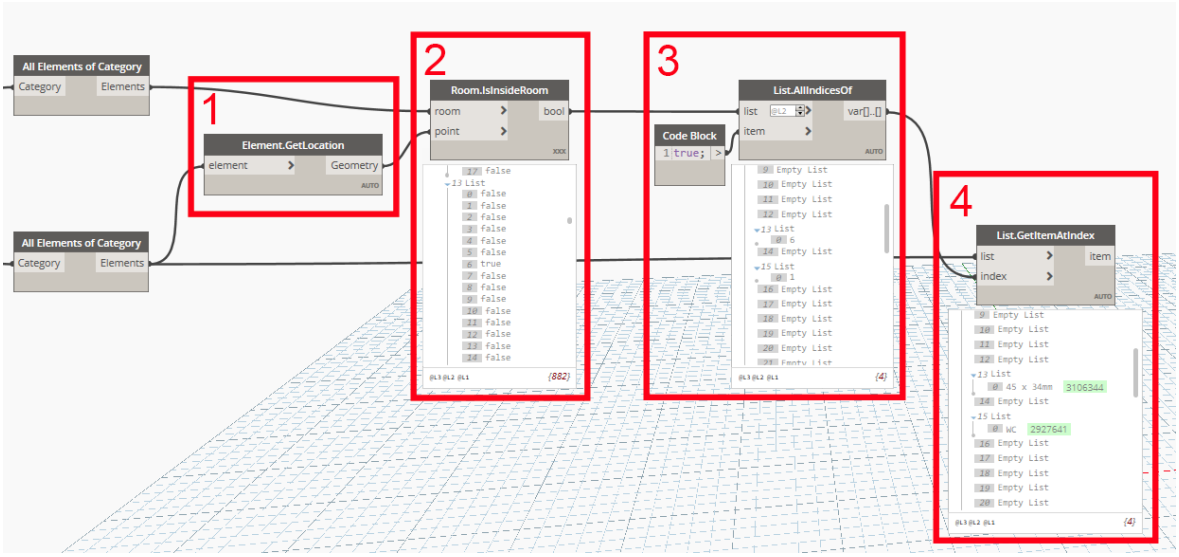


Abbildung 15, Dynamo - Zuordnung von Elementen zu Räumen und Flächen

Schritt 4: Sammeln der Parameterwerte und Berechnung der Sammelparameter. Nachdem alle relevanten Elemente ihren Räumen zugeordnet wurden, können nun die Parameterwerte herangezogen werden, die für die Berechnung benötigt werden. In der Abbildung werden (1). Einmal die Wasserverbrauchswerte von WCs und einmal von Duschen und Waschbecken berechnet. Der Parameter BK_Use-Amount, welcher die Verbrauchswerte pro Nutzung für WCs und pro Sekunde für Waschbecken und Duschen innehält ist als Typparameter angelegt. Zum Auslesen von Typenparametern wird das Dynamo-Paket „Rhythm“ hinzugezogen, welches diesen Vorgang vereinfacht. Nachfolgend werden die gesammelten Werte entsprechend der vordefinierten Formel auf den Zielwert, in diesem Fall dem Verbrauch in Kubikmeter pro Jahr, gebracht (2). Zur Fehlervermeidung folgt der Austausch unter Umständen entstehender null-Werte zum Zahlenwert „0“ (3). Da sich in einem Raum mehrere Sanitärobjekte befinden können, werden die Werte daraufhin aufsummiert und ergeben den Verbrauchswert pro Raum (4). Da WCs und Waschbecken/Duschen getrennt voneinander berechnet werden müssen, werden diese nun addiert (5) und in den entsprechenden Sammelparameter der Räume eingetragen (6).

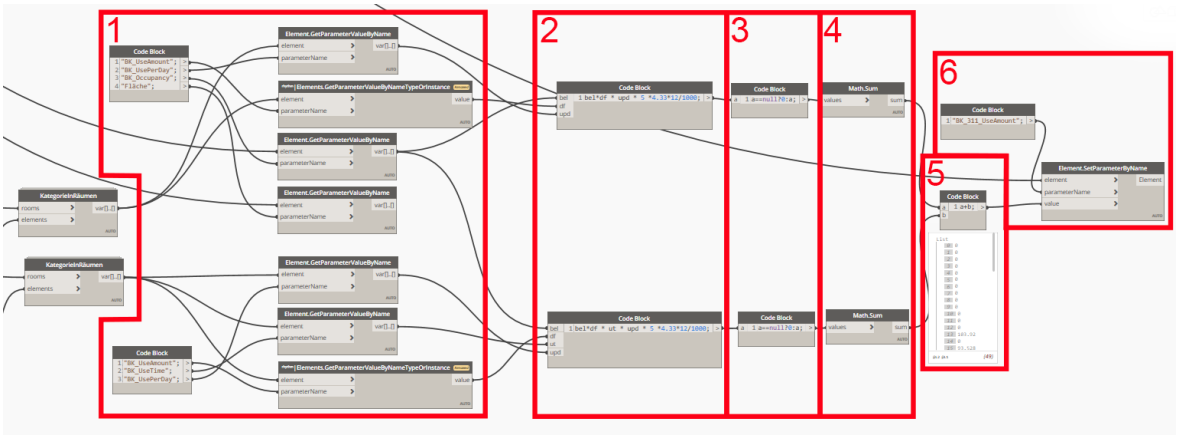


Abbildung 16, Dynamo - Berechnung der jahresbezogenen Werte und Zuweisung zu Sammelparametern

5.2.2 Zusammenführen der Daten

Trotz der vorhandenen Möglichkeit über die Exportfunktionen von Revit Bauteillisten in Tabellenform als *.csv-Datei auszugeben, wird sich für einen Export als *.xlsx-Datei per Dynamo Skript entschieden. Diese Variante bietet die Möglichkeit eine automatisierte Formatierung und Manipulation der resultierenden Tabelle zu realisieren. So wird sich der Zwischenschritt gespart, die Tabellendaten per Excel nacharbeiten zu müssen und für das auslesende Python-Programm vorzubereiten. Weiterhin können zusätzliche Informationen zu den Daten mitgegeben werden, wie in diesem Fall die Anzahl der mitgelieferten Parameter und Räume, beziehungsweise Flächen in der ersten Zeile der Tabelle. Zur Kontrolle der Daten empfiehlt sich dennoch ein Blick in die Tabellendaten.

Schritt 1: Sammeln aller gesetzten Räume und zu exportierender Parameter. Im ersten Schritt werden die nötigen Parameter als Liste aus den Räumen extrahiert.

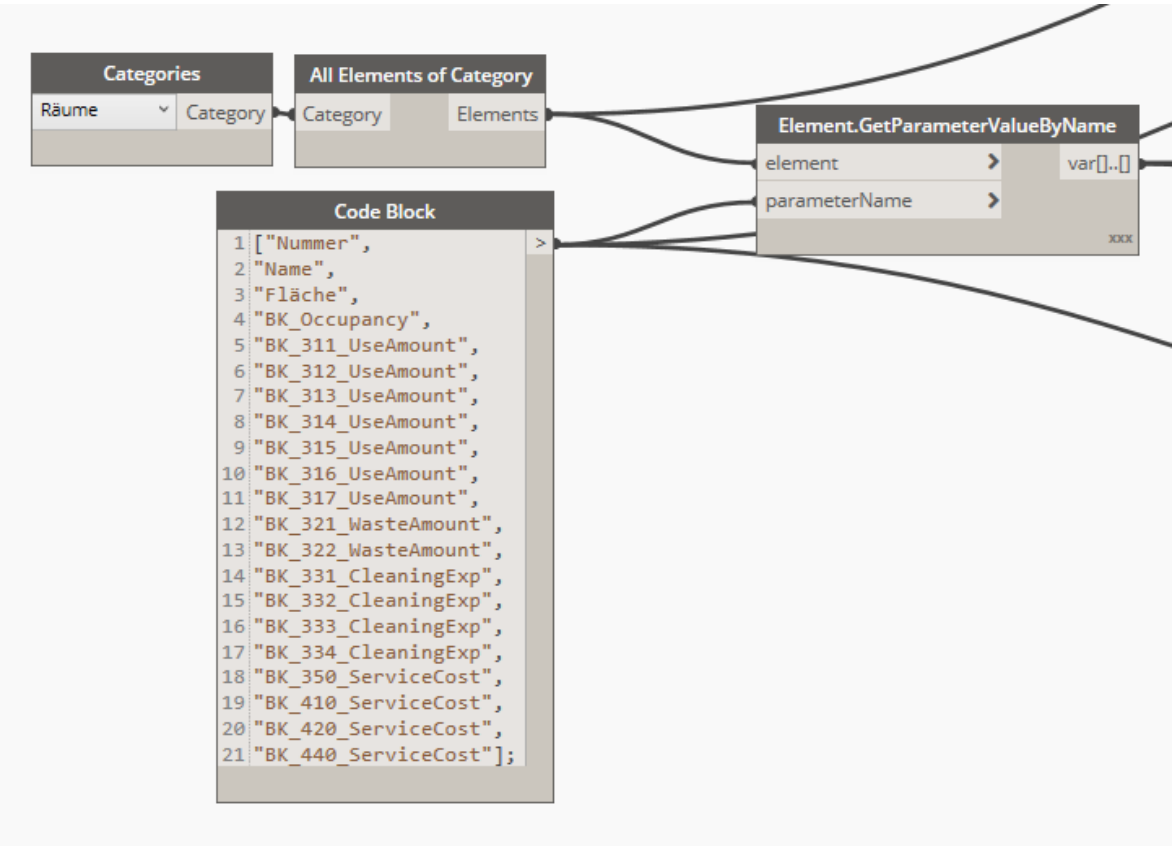


Abbildung 17, Dynamo - Sammeln der zu exportierenden Parameter

Schritt 2: Anforderungen an die Formatierung umsetzen. Um die resultierende Tabelle im für die Weiterverarbeitung nötigen Format ausgeben zu können, müssen diverse Parameterdaten angepasst werden. Raumnummern wird ein Apostroph vorangeführt, um die Konvertierung des Zellenwertes zu einem Zahlenwert zu verhindern und bei einem Zeichenfolgenwert zu belassen (1). Daraufhin werden sämtliche Räume entsprechend ihrer Raumnummer sortiert (2). Ausschlaggebend für die Zuordnung der Zellenwerte zu deren Parametern ist die Kopfzeile, welche die Spalten als Parameterwerte definiert. Genutzt werden hier die eingangs angegebenen Parameternamen. Da die Benennung dieser Parameter zur Datenverarbeitung unvorteilhaft ist, werden diese mit einem in Dynamo integrierten Python-Skript in für die Programmierung vorteilhafte Variablennamen umgewandelt und daraufhin an die erste Position der Liste aus Raumdaten gesetzt (3) und dienen als Kopfzeile der Tabelle. Es handelt sich hierbei um ein äußerst simples Skript, welches sämtliche Einträge der Eingangsliste durchläuft und allein Großbuchstaben und Nummern bis

zu fünf Zeichen weitergibt. Die Parameter Name, Nummer, Fläche und BK_Occupancy werden in die Werte „name“, „nr“, „area“ und „bel“ umgewandelt. Zuletzt wird eine weitere Zeile für sonstige Informationen an den Anfang der Liste gesetzt, welche in diesem Fall die Anzahl der Parameter und die Anzahl der Räume beinhaltet (4).

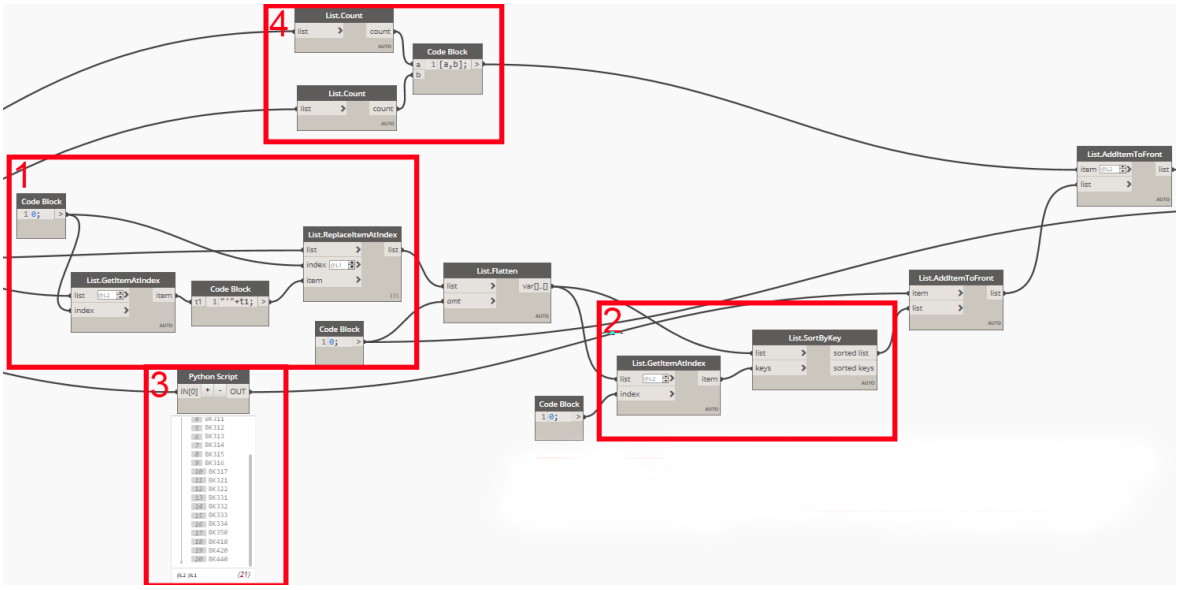


Abbildung 18, Vorbereiten der Parameterdaten zum Listenexport

Schritt 3: Export als *.xlsx-Tabelle. Im letzten Schritt wird die manipulierte Liste aus Raumdaten in eine neue Excel-Datei mit vordefiniertem Namen geschrieben. Es werden insgesamt zwei Tabellenblätter ausgegeben, welche in Hinblick auf die Formatierung diesem Ablauf folgen – eine Tabelle der Raumdaten und eine Tabelle der Flächendaten. Der Ausgabepfad kann festgelegt werden und sollte für einen nahtlosen Übergang in einem festgelegten Ordner liegen, welcher im auslesenden Python-Programm ebenfalls definiert ist. Auch eine Ausgabe als CSV-Datei ist hier möglich. Diese Variante erscheint eingangs sinnvoller als ein Export zu Excel, jedoch kann sich so eine Konvertierung in die lesbarere XLSX, zum Zweck einer schnellen Sichtkontrolle, gespart werden.

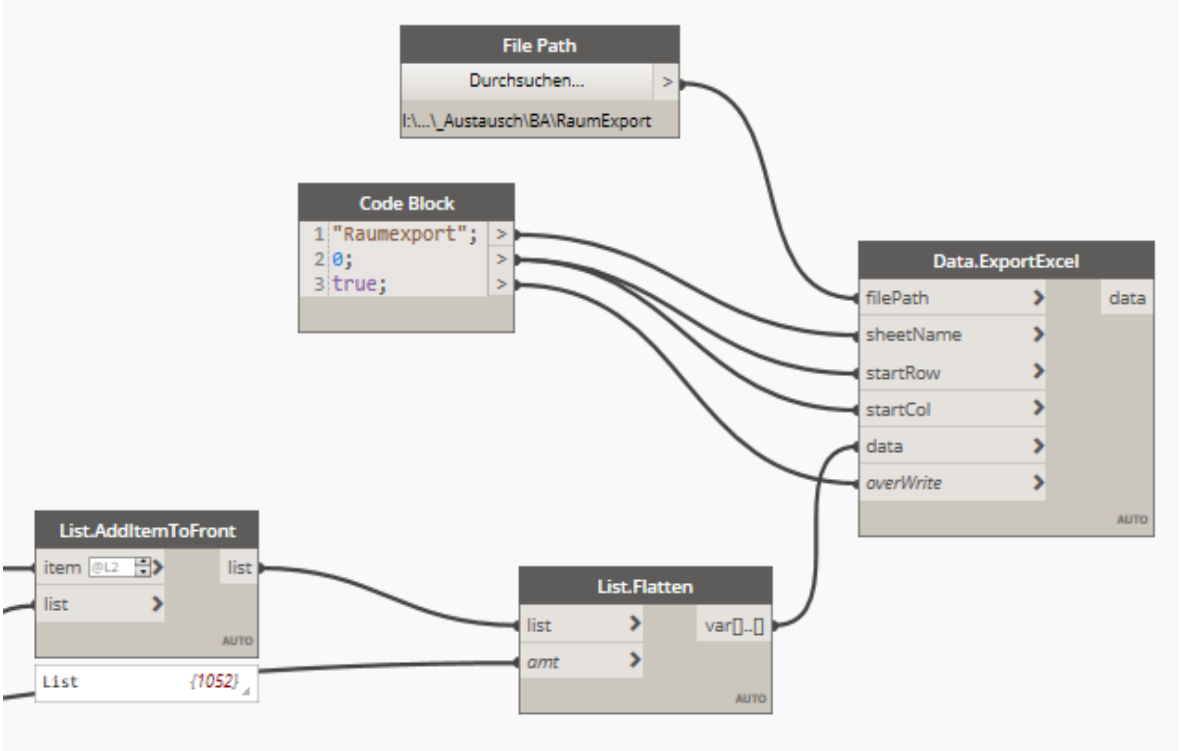


Abbildung 19, Dynamo - Export einer Liste als xlsx-Tabellendatei

5.2.3 Weiterverarbeitung außerhalb der Revit-Umgebung

Zur Weiterverarbeitung der ausgegebenen Datentabellen wurde sich gegen eine Verarbeitung per Excel und für die Entwicklung eines simplen Python-Codes entschieden. Dieser Entscheidung unterliegen den in Kapitel 2.2.3 genannten Vorteilen.

Für die Grundfunktionalität der Anwendung, welche darin besteht, xlsx-Dateien auszulesen, werden vorhandene Python-Pakete genutzt. Dabei wurde sich für „pandas“ entschieden. Unter Verwendung des „tkinter“-Pakets wird ein Benutzerinterface realisiert.

„tkinter“ wird verwendet, um berechnete Daten für den Nutzer übersichtlich darzustellen. Die Interfaceerstellung nimmt einige Code-Zeilen in Anspruch ist jedoch für die Untersuchung irrelevant. Von einer näheren Erläuterung des Interface-spezifischen Codes wird aufgrund dessen abgesehen.

Mithilfe von „pandas“ können xlsx-Dateien ausgelesen und manipuliert werden. Gemeinhin stellt das Modul ein mächtiges Datenanalyse-Tool dar. Verwendet wird die Funktion „pandas.read_excel“, um Zellenwerte einer xlsx-Datei auszulesen.

Der gesamte Code ist in Anlage 3 zu finden.

Folgend werden die grundlegenden Funktionen der Python-Anwendung erläutert:

Grundlage bildet die Funktion „**readXLvalue**“, welcher die Tabellendatei, der Spaltenname und die Zeilennummer übergeben wird und diese den entsprechenden Wert der Zelle ausgibt.

Die Funktion „**import_spaces**“ erhält als Eingangsvariable die aus Revit exportierte Liste und übergibt diese der in ihr verschachtelten „readXLvalue“ Funktion. Die ausgelesenen Daten werden daraufhin als eine verschachtelte Liste zwischengespeichert. Für eine optimale Abrufbarkeit werden sämtliche Räume als Instanzen einer Klasse namens „Space“ angelegt. Diese Klassen führen sämtliche gelesene Daten als Attribute mit sich und können diese daraufhin unter Angabe der Attributbezeichnung präsentieren. Selbes geschieht über die Funktion „**import_areas**“ mit den ausgelesenen Flächen, welche als Instanzen der Klasse „Area“ angelegt werden.

Die Funktionen „**get_sum**“, „**get_list**“ und „**get_avg**“ geben unter Angabe der Raum- oder Flächeninstanzen und der auszuwertenden Spalte einerseits die Summe des Parameters aller Räume, gibt sämtliche Parameter der Spalte als Liste aus oder bildet den Durchschnitt aus diesen.

Die Funktion „**get_value**“, führt nach Eingabe der Raum- oder Flächeninstanzen die weiterführende Berechnung der Parameter in Abhängigkeit der spezifischen Kostengruppe durch. Hier werden Werte, wie die Stundenlöhne von Reinigungskräften oder Kosten für die Entsorgung verschiedener Abfallarten mit den Ergebnissen der Funktionen „get_sum“, „get_list“ oder „get_avg“ verrechnet und daraufhin als Ergebniszurückgegeben.

Mit der Funktion „**get_data**“ werden sämtliche vorgenannten Funktionen angestoßen, ruft für jedes Ausgabefeld des User-Interfaces die Funktion „get_value“ auf und befüllt diese mit den zurückgegebenen Ergebnissen.

6. Schlussbetrachtung

6.1 Fazit

Im Rahmen der vorliegenden Bachelorthesis werden erfolgreich Ansätze für Nutzungskosten auf Basis eines digitalen Planungsmodells erhoben. Auf Grundlage gängiger Normative und Veröffentlichungen können Rechenansätze für die Kostengruppen der DIN 19860 entwickelt werden. Im Sinne der BIM-Arbeitsweise werden besagte Rechenansätze für eine elementbezogene Auswertung abgewandelt. Die Inbezugnahme vorhandener Richtwerte ist an unterschiedlichen Stellen des Ermittlungsprozesses möglich. Dabei sollte abgeschätzt werden, wo genau dies geschieht und welche Möglichkeiten die verwendeten Softwarelösungen diesbezüglich bieten.

Unter Anwendung der BIM-Arbeitsweise konnten mit Hilfe der Authoring-Software Revit, der inkludierten Visual-Scripting-Language Dynamo und einer eigens entwickelten Python-Anwendung im Modell hinterlegte Richtwerte ausgewertet, in kostengruppenspezifischen Sammelparametern zusammengeführt und anschließend mit weiteren Daten, welche nicht dem Planungsmodell entspringen, verrechnet werden. Diverse Kostengruppen konnten naturgegeben nicht auf Grundlage der Planungsdaten berechnet werden und müssen der Vollständigkeit halber nachgeführt werden. Aufgrund einer mangelnden Datenlage bezüglich vereinzelter Richtwerte, wie zum Beispiel den Jahreskostenfaktoren für Elemente der Baukonstruktion und Außenanlagen können auch die Kosten der Bedienung, Wartung, Inspektion und Instandsetzung nicht in Gänze abgedeckt werden.

Die Anforderungen an die Kostenerhebung, sowie auch die Verantwortlichkeiten der teilnehmenden Akteure, sind im Vorfeld detailliert zu definieren, sodass die Ergebnisse der Berechnung zu jedem Zeitpunkt nachvollziehbar sind und eine Fehlinterpretation dieser nicht möglich ist. Begleitend zum Planungsprozess sollte immer eine Qualitätskontrolle der digitalen Fachmodelle unter Bezugnahme der Modellierungsrichtlinie und der Ergebnisse der Berechnung durchgeführt werden.

Generell ist ausschlaggebend, mit welcher Datengrundlage gearbeitet wird. Die Entwicklung einer allgemein anwendbaren Bauteilbibliothek kann zur Reduzierung des Aufwands bezüglich der Datenpflege während der Planung beitragen und kann durch die stetige Weiterentwicklung über mehrere Projekte hinweg optimiert und

angepasst werden. Im Idealfall sind in einer solchen Bibliothek spezifische Herstellerdaten hinterlegt. Dennoch ist bereits eine Datenbank aus Platzhalterelementen, welche Richtwerte mit sich bringen, ausreichend, um eine erhöhte Mitarbeiterauslastung zu umgehen. Entsprechend dem aktuellen Entwicklungsstand der Berechnungssystematik, müssen für die Durchführung Mitarbeiter abgestellt werden, welche den Prozess begleiten und potenzielle Fehler beseitigen.

Mithilfe der erarbeiteten Abläufe und Systematiken dieser Ausarbeitung sollte es Dritten möglich sein, diese mit Bezug auf die eigenen Anforderungen zu ergänzen, eine spezifische Berechnungsmethodik zu entwickeln und diese auf eigene Projekte anzuwenden. Die unter Kapitel 5 angewandte Methode ist einer vieler Lösungsansätze, die vorangegangene Systematik anzuwenden und stellt eine prototypische Umsetzung dar, um die grundlegende Systematik zu veranschaulichen. Aufgrund dessen erhebt diese Methode keinen Anspruch auf Vollständigkeit.

6.2 Ausblick

Unter Betrachtung der hier erarbeiteten Systematik bilden sich diverse Potenziale zur Weiterentwicklung und ergänzenden Forschungsthemen ab.

Zum einen kann es sinnvoll sein, weitere Richtwerte, wie beispielsweise die Jahreskostenfaktoren von Baukonstruktionen, zu erheben und zu erforschen, um weitere Kostengruppen im Zuge der Berechnung abdecken zu können. Diese könnten unter anderem in einer Bibliothek von Platzhalter-Bauteilen hinterlegt werden. Generell ist die Entwicklung eines offenen Standards denkbar, auf Grundlage dessen Bauteilhersteller ihre Produkte in digitaler Form mit zusammenhängenden Informationen bestücken können. Ist diese Informationshinterlegung und eine Datenbankstruktur standardisiert, können Softwarelösungen, die mit diesen Informationen arbeiten, effektiv entwickelt werden.

Mit Blick auf die stetige Entwicklung und der aktuellen Popularität von künstlicher Intelligenz, liegt die Überlegung nahe, die Entwicklung der Baupreisindizes oder auch die Erhebung von Richtwerten auf Grundlage veröffentlichter Daten mit Hilfe einer KI zu prognostizieren. So könnten potenziell akkuratere Ergebnisse erzielt werden als über erfahrungsbasierte Annahmen oder die Extrapolierung einzelner Statistiken.

Über eine Verknüpfung der im Anhang befindlichen Python-Anwendungen könnten die Überschläge über die Kosten der Kostengruppen 350 und 400 in Abhängigkeit

der verfügbaren Daten erhoben werden. Über eine Eingabemaske könnte so entschieden werden, ob die Daten auf Grundlage der Flächendaten oder der exportierten, bauteilspezifischen Werte berechnet werden sollen. Prinzipiell ist der Code bezüglich der Datenauswertung der aus Revit gezogenen Daten auszubauen und zu vervollständigen. Dazu sollten vorerst alle erforderlichen Nutzereingaben implementiert werden und alle Kostengruppen abgedeckt sein. Die Inbezugsetzung einer Datenbank aus Richtwerten ist außerdem möglich, welche jedoch angelegt, und in für das Programm lesbare Form gebracht werden muss. Im weiteren Schritt gilt es, Möglichkeiten zur Auswertung von IFC-Daten zu erforschen. Eine vorerst eigenständige Software könnte somit die Verortung und Sortierung der Bauteile, das Auslesen der Bauteildaten und die Berechnung der Kostengruppen übernehmen. Eine Integration in Authoren-Werkzeuge wie Autodesk Revit stellt das übergeordnete Ziel dar. So wäre es denkbar, eine Berechnung per Knopfdruck auf Grundlage des aktuellen Planstands auszulösen. Dies könnte in diesem Zuge durch den Planer geschehen, ohne auf die Mitarbeit weiterer Akteure angewiesen zu sein, womit die Arbeitsressourcen, welche mit Beginn der Entwicklung der Berechnungsmethodik steigen, zu reduzieren.

Die angehangenen Dynamo-Skripte befinden sich in einem prototypischen Zustand und sind aktuell nicht für den Einsatz an realen Projekten zu empfehlen. Eine Ausarbeitung und Standardisierung dieser Skripte ist erforderlich, um diese projektübergreifend anwenden zu können und nachvollziehbare Ergebnisse zu erzielen.

Literatur

Arbeitskreis Maschinen- und Elektrotechnik BMWSB (AMEV). (2013). *TGA-Kosten Betreiben 2013.* Von www.amev-online.de: https://www.amev-online.de/AMEVInhalt/Organisation/TGA-Kosten%20Betreiben/tga2013.pdf abgerufen

ARGE BIM4INFRA. (28. 09 2017). *Umsetzung des Stufenplans Digitales Bauen.* Von www.bim4infra.de: https://bim4infra.de/wp-content/uploads/2018/04/AP1_Workshop_Handout.pdf abgerufen

ARGE BIM4INFRA2020. (04 2019). *Leitfaden und Muster für Auftraggeber-Informationsanforderungen (AIA).* Von www.bim4infra.de: https://bim4infra.de/wp-content/uploads/2019/07/BIM4INFRA2020_AP4_Teil2.pdf abgerufen

ARGE BIM4INFRA2020. (04 2019). *Leitfaden und Muster für den BIM-Abwicklungsplan (BAP).* Von www.bim4infra.de: https://bim4infra.de/wp-content/uploads/2019/09/BIM4INFRA_AP4_Teil3.pdf abgerufen

ARGE BIM4INFRA2020. (04 2019). *Steckbriefe der wichtigsten BIM-Anwendungsfälle*. Von www.bim4infra.de: https://bim4infra.de/wp-content/uploads/2019/07/BIM4INFRA2020_AP4_Teil6.pdf abgerufen

Autodesk Inc. (21. 02 2023). *Revit: BIM-Software für Planer, Bauunternehmer und ausführende Unternehmen*. Von www.autodesk.de: https://www.autodesk.de/products/revit/overview?term=1-YEAR&tab=subscription&plc=RVT abgerufen

Autodesk Inc. (kein Datum). *Bauplanung und -ausführung mit BIM.* Von www.autodesk.com: https://www.autodesk.de/solutions/bim abgerufen

Baukosteninformationszentrum Deutscher Architektenkammern GmbH. (2022). *Baupreisindex 2022 für den* . Von www.bki.de: https://bki.de/baupreisindex.html abgerufen

BECHMANN. (2022). *bechmann-software.de*. Von BECHMANN BIM - Kernmodule: https://bechmann-software.de/modules/bechmann-bim-kernmodule/ abgerufen

BECHMANN. (2022). *bechmann-software.de*. Von BECHMANN AVA - Kernmodule: https://bechmann-software.de/modules/kernmodule/ abgerufen

BIM Level of Development. (2020). Von www.srinsofttech.com: https://www.srinsofttech.com/bim-level-of-development-lod-300-400-500.html abgerufen

Buildingsmart International. (09 2012). *IDM Guide - Integrated Process.* Von www.standards.buildingsmart.org: https://standards.buildingsmart.org/documents/IDM/IDM_guide-IntegratedProcess-2012_09.pdf abgerufen

buildingSMART International. (2022). *Industry Foundation Classes (IFC) - An Introduction*. Von www.technical.buildingsmart.org: https://github.com/buildingSMART/technical.buildingsmart.org/blob/main/Industry-Foundation-Classes-(IFC).md abgerufen

Bundesamt für Bauwesen und Raumordnung. (04 2009). *Bewertungssystem Nachhaltiges Bauen (BNB) - Trinkwasserbedarf und Abwasseraufkommen.* Von www.nachhaltigesbauen.de: https://www.nachhaltigesbauen.de/fileadmin/RunderTisch/steckbriefe-2010/123.pdf abgerufen

Bundesamt für Bauwesen und Raumordnung. (01 2011). *Bewertungssystem Nachhaltiges Bauen (BNB) - Trinkwasserbedarf und Abwasseraufkommen.* Von www.bnb-nachhaltigesbauen.de: https://www.bnb-nachhaltigesbauen.de/fileadmin/steckbriefe/verwaltungsgebaeude/neubau/v_2011_1/BNB_BN2011-1_123.pdf abgerufen

Deutsches Institut für Normung e.V. (04 2002). DIN 1989-1:2002-04. Von https://manualzz.com/doc/4509458/gem.-din-1989-1 abgerufen

DGNB GmbH. (2018). *static.dgnb.de.* Von DGNB System - Kriterienkatalog Gebäude Neubau: https://static.dgnb.de/fileadmin/dgnb-system/de/gebaeude/neubau/kriterien/03_ECO1.1_Gebaeudebezogene-Kosten-im-Lebenszyklus.pdf abgerufen

f:data GmbH. (28. 11 2020). *Nutzungskosten im Hochbau*. Von www.bauprofessor.de: https://www.bauprofessor.de/nutzungskosten-hochbau/ abgerufen

f:data GmbH. (2023). *DIN EN ISO 19650-1 | 2019-08*. Von www.baunormenlexikon.de: https://www.baunormenlexikon.de/norm/din-en-iso-19650-1/1c11c2ce-c22a-485d-b11b-c9456f27027b abgerufen

Gareba - GmbH. (2012). *Leistungszahlen Gareba GmbH plus.* Von www.qminus.ch: https://www.qminus.ch/dl/daten/rt/Leistungszahlen_Gareba-GmbH_plus.pdf abgerufen

Geschäftsstelle der Bauministerkonferenz Land Baden-Württemberg. (2022). *Orientierungswerte für Hochschulgebäude 2022*. Von www.is-argebau.de: https://www.is-argebau.de/IndexSearch.aspx?method=get&File=b8a892y3y8b984808abb92b8y9ya8ayyb9y884b992a2a0a1a0a1aya4a34b80b8y0wojducjnfs5o1a51x1qppps5 abgerufen

Gutachten.net. (2023). *DIN Norm 276-1 2008-12 Kosten im Hochbau / Kostengruppen*. Von www.gutachten.net: https://gutachten.net/baugutachten/din-normen/din-276-2008.html abgerufen

Gütegemeinschaft Gebäudereinigung e.V. (07 2020). *RAL Merkblatt Unterhaltsreinigung.* Von www.gggr.de: https://www.gggr.de/dateien/pdfs/merkblaetter/ral_merkblatt_unterhaltsreinigung.pdf abgerufen

Handwerkskammer Münster. (2023). *Was ist Building Information Modeling (BIM)*. Von www.handwerkdigital.org: https://www.handwerkdigital.org/index.php/themen/digitales-bauen/10-was-ist-bim-building-information-modeling abgerufen

JetBrains s.r.o. (21. 02 2023). *PyCharm*. Von www.jetbrains.com: https://www.jetbrains.com/de-de/pycharm/ abgerufen

N+P Informationssysteme GmbH. (2022). *spartacus-fm.de*. Von FM-Prozesse: https://www.spartacus-fm.de/fm-prozesse abgerufen

Prof. Dipl.-Ing. Uwe Rotermund M. Eng. TM, F. M. (2016). *Aktueller Entwicklungsstatus Lebenszyklusberechnung.* Von www.fh-muenster.de: https://www.fh-muenster.de/fb5/downloads/departments/rotermund/2016_lebenszyklusko sten_rotermund.pdf abgerufen

Stadtreinigung Hamburg. (2023). *Informationen für Architekten und Bauherren.* Von files.stadtreinigung.hamburg: https://files.stadtreinigung.hamburg/srh-typo3/website/download/PDF/ArchitektenBauherrenBeratung_Infobroschu ere.pdf abgerufen

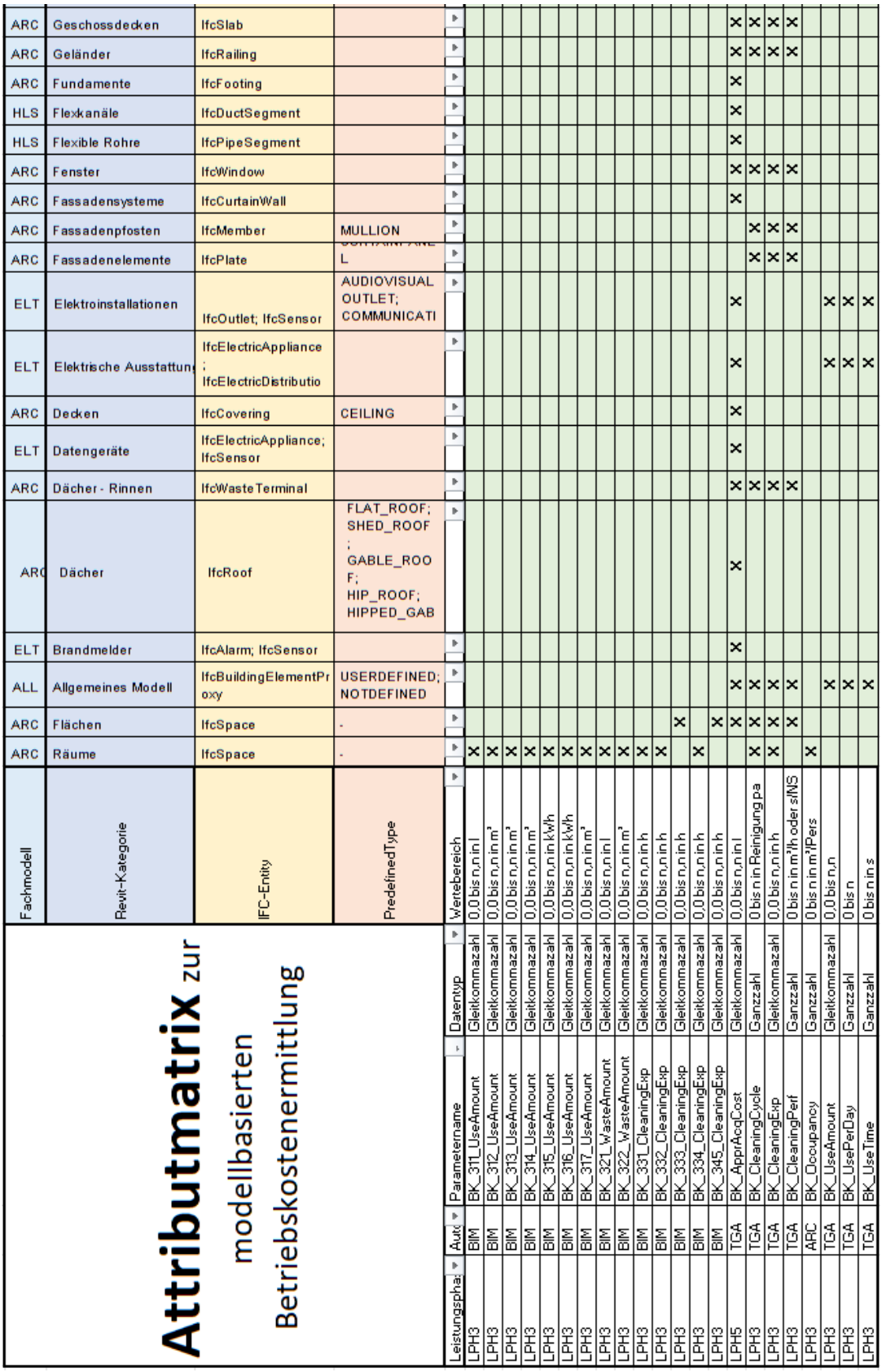
Statista GmbH. (12 2022). *Niederschlagsmenge im Jahr 2022 nach Bundesländern*. Von de.statista.com: https://de.statista.com/statistik/daten/studie/249926/umfrage/niederschlag-im-jahr-nach-bundeslaendern/ abgerufen

Statistisches Bundesamt (Destatis). (2023). *Preisindizes für Bauwerke, Wohngebäude und Nichtwohngebäude*. Von www.destatis.de: https://www.destatis.de/DE/Themen/Wirtschaft/Konjunkturindikatoren/Preis e/bpr110.html#241648 abgerufen

Thinkproject. (2022). *thinkproject.com*. Von Desite BIM: https://thinkproject.com/de/produkte/desite-bim/ abgerufen

Anlagen

Anlage 1, Attributmatrix



Attributmatrix zur modellbasierten Betriebskostenermittlung

Leistungspha	Aut	Parametername	Datentyp	Wertebereich	PredefinedType	IFC-Entity	Revit-Kategorie	Fachmodell
LPH3	BIM	BK_311_UseAmount	Gleitkommazahl	0,0 bis n,n in l	-	IfcSpace	Räume	ARC
LPH3	BIM	BK_312_UseAmount	Gleitkommazahl	0,0 bis n,n in m³	-	IfcSpace	Flächen	ARC
LPH3	BIM	BK_313_UseAmount	Gleitkommazahl	0,0 bis n,n in m³	USERDEFINED; NOTDEFINED	IfcBuildingElementProxy	Allgemeines Modell	ALL
LPH3	BIM	BK_314_UseAmount	Gleitkommazahl	0,0 bis n,n in m³	-	IfcAlarm; IfcSensor	Brandmelder	ELT
LPH3	BIM	BK_315_UseAmount	Gleitkommazahl	0,0 bis n,n in kWh	FLAT_ROOF; SHED_ROOF; GABLE_ROOF; HIP_ROOF; HIPPED_GAB	IfcRoof	Dächer	ARC
LPH3	BIM	BK_316_UseAmount	Gleitkommazahl	0,0 bis n,n in kWh	-	IfcWasteTerminal	Dächer - Rinnen	ARC
LPH3	BIM	BK_317_UseAmount	Gleitkommazahl	0,0 bis n,n in m³	-	IfcElectricAppliance; IfcSensor	Datengeräte	ELT
LPH3	BIM	BK_321_WasteAmount	Gleitkommazahl	0,0 bis n,n in l	CEILING	IfcCovering	Decken	ARC
LPH3	BIM	BK_322_WasteAmount	Gleitkommazahl	0,0 bis n,n in m³	-	IfcElectricAppliance; IfcElectricDistributio	Elektrische Ausstattun	ELT
LPH3	BIM	BK_331_CleaningExp	Gleitkommazahl	0,0 bis n,n in h	AUDIOVISUAL OUTLET; COMMUNICATI	IfcOutlet; IfcSensor	Elektroinstallationen	ELT
LPH3	BIM	BK_332_CleaningExp	Gleitkommazahl	0,0 bis n,n in h	L	IfcPlate	Fassadenelemente	ARC
LPH3	BIM	BK_333_CleaningExp	Gleitkommazahl	0,0 bis n,n in h	MULLION	IfcMember	Fassadenpfosten	ARC
LPH3	BIM	BK_334_CleaningExp	Gleitkommazahl	0,0 bis n,n in h	-	IfcCurtainWall	Fassadensysteme	ARC
LPH3	BIM	BK_345_CleaningExp	Gleitkommazahl	0,0 bis n,n in h	-	IfcWindow	Fenster	ARC
LPH5	TGA	BK_ApprAcqCost	Gleitkommazahl	0,0 bis n,n in l	-	IfcPipeSegment	Flexible Rohre	HLS
LPH3	TGA	BK_CleaningCycle	Ganzzahl	0 bis n in Reinigung pa	-	IfcDuctSegment	Flexkanäle	HLS
LPH3	TGA	BK_CleaningExp	Gleitkommazahl	0,0 bis n,n in h	-	IfcFooting	Fundamente	ARC
LPH3	TGA	BK_CleaningPerf	Ganzzahl	0 bis n in m²/h oder sIMS	-	IfcRailing	Geländer	ARC
LPH3	ARC	BK_Occupancy	Ganzzahl	0 bis n in m²/Pers	-	IfcSlab	Geschossdecken	ARC
LPH3	TGA	BK_UseAmount	Gleitkommazahl	0,0 bis n,n				
LPH3	TGA	BK_UsePerDay	Ganzzahl	0 bis n				
LPH3	TGA	BK_UseTime	Ganzzahl	0 bis n in s				

ARC	Wände	IfcWall																X			X				
ARC	Türen	IfcDoor																X	X	X	X				
ARC	Treppen	IfcStair																	X	X	X				
ARC	Tragwerksstützen	IfcColumn																X							
ARC	Trägersysteme	IfcElementAssembly																X							
ARC	Stützen	IfcColumn																X							
HLS	Sprinkler	IfcFireSuppressionTerminal																X							
ARC	Skelettbau	IfcBeam																X							
ELT	Sicherheitsgeräte	IfcAlarm																X							
ARC	Schreinerarbeiten	IfcBuildingElement																	X	X	X				
HLS	Sanitärinstallationen	IfcSanitaryTerminal; IfcFlowTerminal																X	X	X	X		X	X	X
HLS	Rohrzubehör	IfcValve																X							
HLS	Rohrformteile	IfcPipeFitting																X							
HLS	Rohre	IfcPipeSegment																X							
HLS	Rohrdämmung	IfcCovering	INSULATION															X							
ARC	Rampen	IfcRamp																	X	X	X				
ELT	Notrufgeräte	IfcAlarm																X							
ARC	Möbelsysteme	IfcFurniture																	X	X	X				
ARC	Möbel	IfcBuildingElement																	X	X	X				
TGA	MEP-Räume	IfcSpace		X	X	X	X	X	X	X	X					X			X			X			
HLS	Luftkanalzubehör	IfcDuctSilencer; IfcDamper																X							
HLS	Luftkanalformteile	IfcDuctFitting																X							
HLS	Luftkanäle	IfcDuctSegment																X							
HLS	Luftdurchlässe	IfcAirTerminal																X							
ELT	Lichtschalter	IfcSwitchingDevice																X							
ELT	Leuchten	IfcLightFixture																X					X	X	X
ELT	Kommunikationsgeräte	IfcCommunicationsAppliance																X					X	X	X
ELT	Kabeltrassenformteile	IfcCableCarrierFitting																X							
ELT	Kabeltrassen	IfcCableCarrierSegment																X							
HLS	HLS-Bauteile	IfcValve; IfcTank; IfcPump; IfcUnitaryEquipment; IfcSpaceHeater; IfcCoil; IfcChiller																X	X	X	X		X	X	X

Anlage 2, Dynamo-Skripte

Aufgrund schlechter Darstellungsmöglichkeiten der Skripte, sind diese im angehangenen Datenträger einzusehen.

Anlage 3, Python-Code Datenverarbeitung

```
# Imports ____________________________________________
import json
import tkinter as tk
import string
import pandas as pd


# Global Variables ___________________________________

alp = list(string.ascii_uppercase)
concatfunc = lambda x, y : x+""+y
for i in alp:
    alph = []
    for i in alp:
        alph.append(list(map(concatfunc, "A", i)))
palpha = [item for sublist in alph for item in sublist]
alpha = alp + palpha
dataWidth = [x for x in alpha]

i1 = 5
i2 = 2
# Functions __________________________________________


def readXLvalue(file, column, row):
    data = pd.read_excel(file, skiprows=row - 1, nrows=1, usecols=co-
lumn, header=None, names=["Value"]).iloc[0][
        "Value"]
    return data


def importSpaces(dataFile):
    roomCount = readXLvalue(dataFile, "A", 1)
    paramCount = readXLvalue(dataFile, "B", 1)
    columns = dataWidth[0:paramCount]
    roomList = []
    for i in range(roomCount):
        roomList.append([])
        for j in columns:
            roomList[i].append(readXLvalue(dataFile, j, i+3))
    data = []
    x = 1
    for k in roomList:
        nr = k[0]
        name = k[1]
        area = k[2]
        bel = k[3]
        BK311 = k[4]
        BK312 = k[5]
        BK313 = k[6]
        BK314 = k[7]
        BK315 = k[8]
        BK316 = k[9]
        BK317 = k[10]
        BK321 = k[11]
```

```
        BK322 = k[12]
        BK331 = k[13]
        BK332 = k[14]
        BK333 = k[15]
        BK334 = k[16]
        BK350 = k[17]
        BK410 = k[18]
        BK420 = k[19]
        BK440 = k[20]
        spce = Space(nr, name, area, bel, BK311, BK312, BK313, BK314,
BK315, BK316, BK317, BK321, BK322, BK331, BK332, BK333, BK334, BK350,
BK410, BK420, BK440)
        data.append(spce)
        x += 1
    return data


def import_areas(dataFile):
    roomCount = readXLvalue(dataFile, "A", 1)
    paramCount = readXLvalue(dataFile, "B", 1)
    columns = dataWidth[0:paramCount]
    roomList = []

    for i in range(roomCount):
        roomList.append([])
        for j in columns:
            roomList[i].append(readXLvalue(dataFile, j, i+3))
    data = []
    for k in roomList:
        nr = k[0]
        name = k[1]
        area = k[2]
        BK341 = k[3]
        BK342 = k[4]
        BK343 = k[5]
        BK344 = k[6]
        BK345 = k[7]
        BK346 = k[8]
        BK431 = k[9]
        BK432 = k[10]
        BK433 = k[11]
        BK434 = k[12]
        BK435 = k[13]
        BKAAC = k[14]
        BKCC = k[15]
        BKCE = k[16]
        BKCP = k[17]
        BKSC = k[18]
        BKSL = k[19]
        are = Area(nr, name, area, BK341, BK342, BK343, BK344, BK345,
BK346, BK431, BK432, BK433, BK434, BK435, BKAAC, BKCC, BKCE, BKCP, BKSC,
BKSL)
        data.append(are)
    return data


def get_value(spaces, column):
    if column == 'nr':
        return get_list(spaces, column)
```

```
    elif column == 'name':
        return get_list(spaces, column)
    elif column == 'area':
        return get_sum(spaces, column)
    elif column == 'bel':
        return get_avg(spaces, column)
    elif column == 'BK311':
        return get_sum(spaces, column)
    elif column == 'BK312':
        return get_sum(spaces, column)
    elif column == 'BK313':
        return get_sum(spaces, column)
    elif column == 'BK314':
        return get_sum(spaces, column)
    elif column == 'BK315':
        return get_sum(spaces, column)
    elif column == 'BK316':
        return get_sum(spaces, column)
    elif column == 'BK317':
        return get_sum(spaces, column)
    elif column == 'BK321':
        e = (get_sum(spaces, 'BK311')+get_sum(spaces, column))*in-
put_wastewatercost.get()
        return e
    elif column == 'BK322':
        data = get_sum(spaces, column)
        res = round(data * 0.6 * input_wastecost_a.get() + data * 0.1 *
input_wastecost_b.get() + data * 0.15 * input_wastecost_c.get() + data *
0.15 * input_wastecost_d.get(), 2)
        e = round(res, 2)
        return e
    elif column == 'BK331':
        return get_sum(spaces, column)*input_cpcost.get()*4.33*12
    elif column == 'BK332':
        return get_sum(spaces, column)*input_cpcost.get()*4.33*12
    elif column == 'BK333':
        return get_sum(spaces, column)*input_cpcost.get()
    elif column == 'BK334':
        return get_sum(spaces, column)*input_cpcost.get()
    elif column == 'BK341':
        return get_sum(spaces, column)
    elif column == 'BK342':
        return get_sum(spaces, column)
    elif column == 'BK343':
        return get_sum(spaces, column)
    elif column == 'BK344':
        return get_sum(spaces, column)
    elif column == 'BK345':
        return get_sum(spaces, column)
    elif column == 'BK346':
        return get_sum(spaces, column)
    elif column == 'BK350':
        return get_sum(spaces, column)
    elif column == 'BK410':
        return get_sum(spaces, column)
    elif column == 'BK420':
        return get_sum(spaces, column)
    elif column == 'BK431':
        return get_sum(spaces, column)
    elif column == 'BK432':
```

```
        return get_sum(spaces, column)
    elif column == 'BK433':
        return get_sum(spaces, column)
    elif column == 'BK434':
        return get_sum(spaces, column)
    elif column == 'BK435':
        return get_sum(spaces, column)
    elif column == 'BK440':
        return get_sum(spaces, column)
    elif column == 'BKAAC':
        return get_sum(spaces, column)
    elif column == 'BKCC':
        return get_sum(spaces, column)
    elif column == 'BKCE':
        return get_sum(spaces, column)
    elif column == 'BKCP':
        return get_sum(spaces, column)
    elif column == 'BKSC':
        return get_sum(spaces, column)
    elif column == 'BKSL':
        return get_sum(spaces, column)
    else:
        return "Error. Ansicht wählen."


def get_sum(spaces, c):
    vals = []
    for i in spaces:
        vals.append(getattr(i, c))
    sumcol = round(sum(vals), 2)
    return sumcol


def get_list(spaces, c):
    vals = []
    for i in spaces:
        vals.append(getattr(i, c))
    return vals


def get_avg(spaces, c):
    vals = []
    for i in spaces:
        vals.append(getattr(i, c))
    sumcol = round(sum(vals), 2)
    val = sumcol/len(vals)
    return val


def get_data(spaces, areas):
    output_ngf.set(get_value(spaces, 'area'))
    output_bgf.set(get_value(areas, 'area'))
    output_310.set(get_value(spaces, 'BK311')+get_value(spaces,
'BK312')+get_value(spaces, 'BK313')+get_value(spaces, 'BK314')+get_va-
lue(spaces, 'BK315')+get_value(spaces, 'BK316')+get_value(spaces,
'BK317'))
    output_311.set(get_value(spaces, 'BK311'))
    output_312.set(get_value(spaces, 'BK312'))
    output_313.set(get_value(spaces, 'BK313'))
```

```
    output_314.set(get_value(spaces, 'BK314'))
    output_315.set(get_value(spaces, 'BK315'))
    output_316.set(get_value(spaces, 'BK316'))
    output_317.set(get_value(spaces, 'BK317'))
    output_320.set(get_value(spaces, 'BK321') + get_value(spaces,
'BK322'))
    output_321.set(get_value(spaces, 'BK321'))
    output_322.set(get_value(spaces, 'BK322'))
    output_330.set(get_value(spaces, 'BK331') + get_value(spaces,
'BK332') + get_value(spaces, 'BK333') + get_value(spaces, 'BK334'))
    output_331.set(get_value(spaces, 'BK331'))
    output_332.set(get_value(spaces, 'BK332'))
    output_333.set(get_value(spaces, 'BK333'))
    output_334.set(get_value(spaces, 'BK334'))
    output_340.set(get_value(areas, 'BK341') + get_value(areas, 'BK342')
+ get_value(areas, 'BK343') + get_value(areas, 'BK344') + get_va-
lue(areas, 'BK345') + get_value(areas, 'BK346'))
    output_341.set(get_value(areas, 'BK341'))
    output_342.set(get_value(areas, 'BK342'))
    output_343.set(get_value(areas, 'BK343'))
    output_344.set(get_value(areas, 'BK344'))
    output_345.set(get_value(areas, 'BK345'))
    output_346.set(get_value(areas, 'BK346'))
    output_350.set(get_value(spaces, 'BK350'))
    output_410.set(get_value(spaces, 'BK410'))
    output_420.set(get_value(spaces, 'BK420'))
    output_430.set(get_value(areas, 'BK431') + get_value(areas, 'BK432')
+ get_value(areas, 'BK433') + get_value(areas, 'BK434') + get_va-
lue(areas, 'BK435'))
    output_431.set(get_value(areas, 'BK431'))
    output_432.set(get_value(areas, 'BK432'))
    output_433.set(get_value(areas, 'BK433'))
    output_434.set(get_value(areas, 'BK434'))
    output_435.set(get_value(areas, 'BK435'))
    output_440.set(get_value(spaces, 'BK440'))
    output_400.set(output_410.get() + output_420.get() + out-
put_430.get() + output_440.get())
    output_300.set(output_310.get() + output_320.get() + out-
put_330.get() + output_340.get() + output_350.get())
    output_ges.set(output_300.get() + output_400.get())

def close():
    window.destroy()

class Area(object):
    def __init__(self, nr, name, area, BK341, BK342, BK343, BK344,
BK345, BK346, BK431, BK432, BK433, BK434, BK435, BKAAC, BKCC, BKCE,
BKCP, BKSC, BKSL):
        self.nr = nr
        self.name = name
        self.area = area
        self.BK341 = BK341
        self.BK342 = BK342
        self.BK343 = BK343
        self.BK344 = BK344
        self.BK345 = BK345
        self.BK346 = BK346
        self.BK431 = BK431
        self.BK432 = BK432
        self.BK433 = BK433
```

```
        self.BK434 = BK434
        self.BK435 = BK435
        self.BKAAC = BKAAC
        self.BKCC = BKCC
        self.BKCE = BKCE
        self.BKCP = BKCP
        self.BKSC = BKSC
        self.BKSL = BKSL

    def __str__(self):
        return
f"[{self.nr},{self.name},{self.area},{self.BK341},{self.BK342},{self.BK3
43},{self.BK344},{self.BK345},{self.BK346},{self.BK431},{self.BK432},{se
lf.BK433},{self.BK434},{self.BK435},{self.BKAAC},{self.BKCC},{self.BKCE}
,{self.BKCP},{self.BKSC},{self.BKSL}]"

    def __repr__(self):
        return
f"[{self.nr},{self.name},{self.area},{self.BK341},{self.BK342},{self.BK3
43},{self.BK344},{self.BK345},{self.BK346},{self.BK431},{self.BK432},{se
lf.BK433},{self.BK434},{self.BK435},{self.BKAAC},{self.BKCC},{self.BKCE}
,{self.BKCP},{self.BKSC},{self.BKSL}]"


class Space(object):
    def __init__(self, nr, name, area, bel, BK311, BK312, BK313, BK314,
BK315, BK316, BK317, BK321, BK322, BK331, BK332, BK333, BK334, BK350,
BK410, BK420, BK440):
        self.nr = nr
        self.name = name
        self.area = area
        self.bel = bel
        self.BK311 = BK311
        self.BK312 = BK312
        self.BK313 = BK313
        self.BK314 = BK314
        self.BK315 = BK315
        self.BK316 = BK316
        self.BK317 = BK317
        self.BK321 = BK321
        self.BK322 = BK322
        self.BK331 = BK331
        self.BK332 = BK332
        self.BK333 = BK333
        self.BK334 = BK334
        self.BK350 = BK350
        self.BK410 = BK410
        self.BK420 = BK420
        self.BK440 = BK440

    def __str__(self):
        return
f"[{self.nr},{self.name},{self.area},{self.bel},{self.BK311},{self.BK312
},{self.BK313},{self.BK314},{self.BK315},{self.BK316},{self.BK317},{self
.BK321},{self.BK322},{self.BK331},{self.BK332},{self.BK333},{self.BK334}
,{self.BK350},{self.BK410},{self.BK420},{self.BK440}]"

    def __repr__(self):
        return
```

```
f"[{self.nr},{self.name},{self.area},{self.bel},{self.BK311},{self.BK312
},{self.BK313},{self.BK314},{self.BK315},{self.BK316},{self.BK317},{self
.BK321},{self.BK322},{self.BK331},{self.BK332},{self.BK333},{self.BK334}
,{self.BK350},{self.BK410},{self.BK420},{self.BK440}]"


# GUI

# INPUTS _____________________

areaFile = 'AreaExport.xlsx'
spaceFile = 'RaumExport.xlsx'


# VARS ________________________

areas = import_areas(areaFile)
spaces = importSpaces(spaceFile)

paramCount_s = readXLvalue(spaceFile, "B", 1)
paramCount_a = readXLvalue(areaFile, "B", 1)
columns_s = dataWidth[0:paramCount_s]
columns_a = dataWidth[0:paramCount_a]

spaceParams = []
for i in columns_s:
    spaceParams.append(readXLvalue(spaceFile, i, 2))
areaParams = []
for i in columns_a:
    areaParams.append(readXLvalue(areaFile, i, 2))

ngf = get_value(spaces, 'area')
bgf = get_value(areas, 'area')


#  TKINTER INTERFACE ____________________________________________
# ELEMENTS _____________
window = tk.Tk()
window.title('BK-Berechnung')
window.geometry('1280x1000')
frame_main = tk.Frame(master=window, width=100)
frame_in = tk.Frame(master=frame_main, width=25,bd=2,relief="sunken")
frame_out = tk.Frame(master=frame_main, width=75,bd=2,relief="sunken")
frame_out_1 = tk.Frame(master=frame_out, width=25,bd=1,relief="sunken")
frame_out_2 = tk.Frame(master=frame_out, width=25,bd=1,relief="sunken")
frame_out_3 = tk.Frame(master=frame_out, width=25,bd=1,relief="sunken")
frame_set = tk.Frame(master=window, width=50, height=5, bg='#666666')

btn_quit = tk.Button(
    master=frame_set,
    command=close,
    text='X',
    width=5,
    height=2,
    bg='#666666',
    fg='white',
    bd=0
).pack(side=tk.RIGHT, anchor='n', ipady=2, ipadx=2)

btn_go = tk.Button(
```

```
    master=frame_set,
    command=lambda: getdata(spaces, areas),
    text='START',
    width=15,
    height=2,
    bg='#22aa22',
    fg='white',
    bd=0
).pack(fill=tk.Y ,side=tk.LEFT, anchor='n', ipady=2, ipadx=2)
frame_set.pack(fill=tk.BOTH, side=tk.TOP, expand=True)
frame_main.pack(fill=tk.BOTH, side=tk.BOTTOM, expand=True)
frame_in.pack(fill=tk.BOTH, side=tk.LEFT, expand=True, ipadx=20, i-
pady=20)
frame_out.pack(fill=tk.BOTH, side=tk.RIGHT, expand=True, ipadx=20, i-
pady=20)
frame_out_1.pack(fill=tk.BOTH, side=tk.LEFT, expand=True, ipadx=20, i-
pady=20)
frame_out_2.pack(fill=tk.BOTH, side=tk.LEFT, expand=True, ipadx=20, i-
pady=20)
frame_out_3.pack(fill=tk.BOTH, side=tk.LEFT, expand=True, ipadx=20, i-
pady=20)

# inputs

input_cpcost = tk.DoubleVar()
input_wastewatercost = tk.DoubleVar()
input_wastecost_a = tk.DoubleVar()
input_wastecost_b = tk.DoubleVar()
input_wastecost_c = tk.DoubleVar()
input_wastecost_d = tk.DoubleVar()

label_cpcost = tk.Label(
    master=frame_in,
    text='Stundenlohn Reinigungskräfte:'
).pack(side=tk.TOP, padx=(20, 0), anchor='w')
entry_cpcost = tk.Entry(
    master=frame_in,
    textvariable=input_cpcost
).pack(side=tk.TOP, padx=(20, 0), pady=(5, 10), anchor='w')

label_wastewatercost = tk.Label(
    master=frame_in,
    text='Kosten pro m³ Abwasser:'
).pack(side=tk.TOP, padx=(20, 0), anchor='w')
entry_wastewatercost = tk.Entry(
    master=frame_in,
    textvariable=input_wastewatercost
).pack(side=tk.TOP, padx=(20, 0), pady=(5, 10), anchor='w')

label_wastecost_a = tk.Label(
    master=frame_in,
    text='Abfallkosten_A pro m³:'
).pack(side=tk.TOP, padx=(20, 0), anchor='w')
entry_wastecost_a = tk.Entry(
    master=frame_in,
    textvariable=input_wastecost_a
).pack(side=tk.TOP, padx=(20, 0), pady=(5, 10), anchor='w')

label_wastecost_b = tk.Label(
```

```
    master=frame_in,
    text='Abfallkosten_B pro m³:'
).pack(side=tk.TOP, padx=(20, 0), anchor='w')
entry_wastecost_b = tk.Entry(
    master=frame_in,
    textvariable=input_wastecost_b
).pack(side=tk.TOP, padx=(20, 0), pady=(5, 10), anchor='w')

label_wastecost_c = tk.Label(
    master=frame_in,
    text='Abfallkosten_C pro m³:'
).pack(side=tk.TOP, padx=(20, 0), anchor='w')
entry_wastecost_c = tk.Entry(
    master=frame_in,
    textvariable=input_wastecost_c
).pack(side=tk.TOP, padx=(20, 0), pady=(5, 10), anchor='w')

label_wastecost_d = tk.Label(
    master=frame_in,
    text='Abfallkosten_D pro m³:'
).pack(side=tk.TOP, padx=(20, 0), anchor='w')
entry_wastecost_d = tk.Entry(
    master=frame_in,
    textvariable=input_wastecost_d
).pack(side=tk.TOP, padx=(20, 0), pady=(5, 10), anchor='w')

# outputs

output_ngf = tk.DoubleVar()
output_bgf = tk.DoubleVar()
output_310 = tk.DoubleVar()
output_311 = tk.DoubleVar()
output_312 = tk.DoubleVar()
output_313 = tk.DoubleVar()
output_314 = tk.DoubleVar()
output_315 = tk.DoubleVar()
output_316 = tk.DoubleVar()
output_317 = tk.DoubleVar()
output_320 = tk.DoubleVar()
output_321 = tk.DoubleVar()
output_322 = tk.DoubleVar()
output_330 = tk.DoubleVar()
output_331 = tk.DoubleVar()
output_332 = tk.DoubleVar()
output_333 = tk.DoubleVar()
output_334 = tk.DoubleVar()
output_340 = tk.DoubleVar()
output_341 = tk.DoubleVar()
output_342 = tk.DoubleVar()
output_343 = tk.DoubleVar()
output_344 = tk.DoubleVar()
output_345 = tk.DoubleVar()
output_346 = tk.DoubleVar()
output_350 = tk.DoubleVar()
output_410 = tk.DoubleVar()
output_420 = tk.DoubleVar()
output_430 = tk.DoubleVar()
output_431 = tk.DoubleVar()
output_432 = tk.DoubleVar()
output_433 = tk.DoubleVar()
```

```
output_434 = tk.DoubleVar()
output_435 = tk.DoubleVar()
output_440 = tk.DoubleVar()

output_400 = tk.DoubleVar()
output_300 = tk.DoubleVar()
output_ges = tk.DoubleVar()


label_f = tk.Label(
    master=frame_out_1,
    text='Grundflächen',
    font=('Arial', 10, 'bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')

label_ngf = tk.Label(
    master=frame_out_1,
    text='rechn. NGF:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_ngf = tk.Label(
    master=frame_out_1,
    textvariable=output_ngf
).pack(side=tk.TOP, anchor='w')

label_bgf = tk.Label(
    master=frame_out_1,
    text='rechn. BGF:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_bgf = tk.Label(
    master=frame_out_1,
    textvariable=output_bgf
).pack(side=tk.TOP, anchor='w')

label_ges = tk.Label(
    master=frame_out_1,
    text='Ergebnisse',
    font=('Arial', 10, 'bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')

label_310 = tk.Label(
    master=frame_out_1,
    text='KG 310',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_310 = tk.Label(
    master=frame_out_1,
    textvariable=output_310
).pack(side=tk.TOP, anchor='w')

label_320 = tk.Label(
    master=frame_out_1,
    text='KG 320',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_320 = tk.Label(
    master=frame_out_1,
    textvariable=output_320
).pack(side=tk.TOP, anchor='w')
```

```
label_330 = tk.Label(
    master=frame_out_1,
    text='KG 330',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_330 = tk.Label(
    master=frame_out_1,
    textvariable=output_330
).pack(side=tk.TOP, anchor='w')

label_340 = tk.Label(
    master=frame_out_1,
    text='KG 340',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_340 = tk.Label(
    master=frame_out_1,
    textvariable=output_340
).pack(side=tk.TOP, anchor='w')

label_350_ges = tk.Label(
    master=frame_out_1,
    text='KG 350',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_350_ges = tk.Label(
    master=frame_out_1,
    textvariable=output_350
).pack(side=tk.TOP, anchor='w')

label_410_ges = tk.Label(
    master=frame_out_1,
    text='KG 410',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_410_ges = tk.Label(
    master=frame_out_1,
    textvariable=output_410
).pack(side=tk.TOP, anchor='w')

label_420_ges = tk.Label(
    master=frame_out_1,
    text='KG 420',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_420_ges = tk.Label(
    master=frame_out_1,
    textvariable=output_420
).pack(side=tk.TOP, anchor='w')

label_430 = tk.Label(
    master=frame_out_1,
    text='KG 430',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_430 = tk.Label(
    master=frame_out_1,
    textvariable=output_430
).pack(side=tk.TOP, anchor='w')
```

```
label_440_ges = tk.Label(
    master=frame_out_1,
    text='KG 440',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_440_ges = tk.Label(
    master=frame_out_1,
    textvariable=output_440
).pack(side=tk.TOP, anchor='w')

seperator = tk.Canvas(frame_out_1, height= 1,
bg="black").pack(fill="x",pady=(5,10),padx=15)

label_300_ges = tk.Label(
    master=frame_out_1,
    text='KG 300',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_300_ges = tk.Label(
    master=frame_out_1,
    textvariable=output_300
).pack(side=tk.TOP, anchor='w')

label_400_ges = tk.Label(
    master=frame_out_1,
    text='KG 400',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_400_ges = tk.Label(
    master=frame_out_1,
    textvariable=output_400
).pack(side=tk.TOP, anchor='w')

seperator = tk.Canvas(frame_out_1, height= 2,
bg="black").pack(fill="x",pady=(5,10),padx=10)

label_ges = tk.Label(
    master=frame_out_1,
    text='Gesamt',
    font=('bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_ges = tk.Label(
    master=frame_out_1,
    textvariable=output_ges
).pack(side=tk.TOP, anchor='w')

label_300 = tk.Label(
    master=frame_out_2,
    text='KG 300',
    font=('Arial', 10, 'bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')

label_311 = tk.Label(
    master=frame_out_2,
    text='KG 311:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_311 = tk.Label(
    master=frame_out_2,
    textvariable=output_311
```

```
).pack(side=tk.TOP, anchor='w')

label_312 = tk.Label(
    master=frame_out_2,
    text='KG 312:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_312 = tk.Label(
    master=frame_out_2,
    textvariable=output_312
).pack(side=tk.TOP, anchor='w')

label_313 = tk.Label(
    master=frame_out_2,
    text='KG 313:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_313 = tk.Label(
    master=frame_out_2,
    textvariable=output_313
).pack(side=tk.TOP, anchor='w')

label_314 = tk.Label(
    master=frame_out_2,
    text='KG 314:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_314 = tk.Label(
    master=frame_out_2,
    textvariable=output_314
).pack(side=tk.TOP, anchor='w')

label_315 = tk.Label(
    master=frame_out_2,
    text='KG 315:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_315 = tk.Label(
    master=frame_out_2,
    textvariable=output_315
).pack(side=tk.TOP, anchor='w')

label_316 = tk.Label(
    master=frame_out_2,
    text='KG 316:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_316 = tk.Label(
    master=frame_out_2,
    textvariable=output_316
).pack(side=tk.TOP, anchor='w')

label_317 = tk.Label(
    master=frame_out_2,
    text='KG 317:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_317 = tk.Label(
    master=frame_out_2,
    textvariable=output_317
).pack(side=tk.TOP, anchor='w')

label_321 = tk.Label(
    master=frame_out_2,
    text='KG 321:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
```

```
label_r_321 = tk.Label(
    master=frame_out_2,
    textvariable=output_321
).pack(side=tk.TOP, anchor='w')

label_322 = tk.Label(
    master=frame_out_2,
    text='KG 322:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_322 = tk.Label(
    master=frame_out_2,
    textvariable=output_322
).pack(side=tk.TOP, anchor='w')

label_331 = tk.Label(
    master=frame_out_2,
    text='KG 331:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_331 = tk.Label(
    master=frame_out_2,
    textvariable=output_331
).pack(side=tk.TOP, anchor='w')

label_332 = tk.Label(
    master=frame_out_2,
    text='KG 332:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_332 = tk.Label(
    master=frame_out_2,
    textvariable=output_332
).pack(side=tk.TOP, anchor='w')

label_333 = tk.Label(
    master=frame_out_2,
    text='KG 333:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_333 = tk.Label(
    master=frame_out_2,
    textvariable=output_333
).pack(side=tk.TOP, anchor='w')

label_334 = tk.Label(
    master=frame_out_2,
    text='KG 334:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_334 = tk.Label(
    master=frame_out_2,
    textvariable=output_334
).pack(side=tk.TOP, anchor='w')

label_341 = tk.Label(
    master=frame_out_2,
    text='KG 341:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_341 = tk.Label(
    master=frame_out_2,
    textvariable=output_341
).pack(side=tk.TOP, anchor='w')
```

```
label_342 = tk.Label(
    master=frame_out_2,
    text='KG 342:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_342 = tk.Label(
    master=frame_out_2,
    textvariable=output_342
).pack(side=tk.TOP, anchor='w')

label_343 = tk.Label(
    master=frame_out_2,
    text='KG 343:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_343 = tk.Label(
    master=frame_out_2,
    textvariable=output_343
).pack(side=tk.TOP, anchor='w')

label_344 = tk.Label(
    master=frame_out_2,
    text='KG 344:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_344 = tk.Label(
    master=frame_out_2,
    textvariable=output_344
).pack(side=tk.TOP, anchor='w')

label_345 = tk.Label(
    master=frame_out_2,
    text='KG 345:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_345 = tk.Label(
    master=frame_out_2,
    textvariable=output_345
).pack(side=tk.TOP, anchor='w')

label_346 = tk.Label(
    master=frame_out_2,
    text='KG 346:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_346 = tk.Label(
    master=frame_out_2,
    textvariable=output_346
).pack(side=tk.TOP, anchor='w')

label_350 = tk.Label(
    master=frame_out_2,
    text='KG 350:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_350 = tk.Label(
    master=frame_out_2,
    textvariable=output_350
).pack(side=tk.TOP, anchor='w')

label_400 = tk.Label(
    master=frame_out_3,
    text='KG 400',
    font=('Arial', 10, 'bold')
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
```

```
label_410 = tk.Label(
    master=frame_out_3,
    text='KG 410:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_410 = tk.Label(
    master=frame_out_3,
    textvariable=output_410
).pack(side=tk.TOP, anchor='w')

label_420 = tk.Label(
    master=frame_out_3,
    text='KG 420:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_420 = tk.Label(
    master=frame_out_3,
    textvariable=output_420
).pack(side=tk.TOP, anchor='w')

label_431 = tk.Label(
    master=frame_out_3,
    text='KG 431:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_431 = tk.Label(
    master=frame_out_3,
    textvariable=output_431
).pack(side=tk.TOP, anchor='w')

label_432 = tk.Label(
    master=frame_out_3,
    text='KG 432:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_432 = tk.Label(
    master=frame_out_3,
    textvariable=output_432
).pack(side=tk.TOP, anchor='w')

label_433 = tk.Label(
    master=frame_out_3,
    text='KG 433:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_433 = tk.Label(
    master=frame_out_3,
    textvariable=output_433
).pack(side=tk.TOP, anchor='w')

label_434 = tk.Label(
    master=frame_out_3,
    text='KG 434:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_434 = tk.Label(
    master=frame_out_3,
    textvariable=output_434
).pack(side=tk.TOP, anchor='w')

label_435 = tk.Label(
    master=frame_out_3,
    text='KG 435:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_435 = tk.Label(
```

```
    master=frame_out_3,
    textvariable=output_435
).pack(side=tk.TOP, anchor='w')

label_440 = tk.Label(
    master=frame_out_3,
    text='KG 440:'
).pack(side=tk.TOP, pady=(i1, i2), anchor='w')
label_r_440 = tk.Label(
    master=frame_out_3,
    textvariable=output_440
).pack(side=tk.TOP, anchor='w')


# GUI inputs
cpcost = 13.5
wastewatercost = 0.6
wastecost_a = 2
wastecost_b = 1
wastecost_c = 3
wastecost_d = 0.5

input_wastewatercost.set(wastewatercost)
input_wastecost_a.set(wastecost_a)
input_wastecost_b.set(wastecost_b)
input_wastecost_c.set(wastecost_c)
input_wastecost_d.set(wastecost_d)
input_cpcost.set(cpcost)

window.mainloop()
```

Anlage 4, Python-Code KG 350 & 400

```
# Imports
```

```
import json
import tkinter as tk
import void as void
import pandas as pd

# Funktionen
```

```
def readXLvalue(file,column,row):
    data = pd.read_excel(file, skiprows=row - 1, nrows=1, usecols=co-
lumn, header=None, names=["Value"]).iloc[0][
        "Value"]
    return data

def interp(x1, x2, y1, y2, val):
    return y1 + (((val-x1)/(x2-x1))*(y2-y1))


def WBWert(HKbj, BPIa, BPIbj):
    return HKbj * BPIa/BPIbj


def anteilP(perc, val):
    return val * perc/100


def Kges(wb, jkf, jkw):
    return wb * jkf * jkw


def mult(a,b):
    return a*b


def getFkorN(id):
    for i in JKFData['NF']:
        if i['NFID'] == id:
            return i['FkorN']
        else:
            void


def getFkorH(l):
    if l < 9:
        return 1.0
    else:
        return 1.7


def getFkorA(age,wib):
    if wib == True:
```

```
        if age < 6:
            return 0.5
        elif age < 21:
            return 1.0
        else:
            return 1.8
    else:
        return 1.0


# Jahreskostenfaktor ____________________________________________
def getJKF(id):
    for i in JKFData['NF']:
        if i['NFID'] == id:
            return [i['F1'], i['F2'], i['FBE']]
        else:
            void


# Jahreskorrekturfaktor _________________________________________
def getJKW(FkorN,FkorA,FkorH):
    return FkorN * FkorA * FkorH


# Wegekosten ____________________________________________________
def getfWeg(i):
    if i == 1:
        return 0.0
    elif i == 2:
        return 0.05
    else:
        return 0.1


# Eigenmaterialkosten ___________________________________________
def getmatK(kopIn, kopWIB, eaIn, eaWI):
    return round(kopIn * eaIn * 0.25 + kopWIB * eaWI * 0.1, 2)


# GUI Funktionen ________________________________________________
def Close():
    window.destroy()


def verify():
    # Berechnung
_____________________________________________________________________
_____________________
    # Jahreskorrekturfaktoren_____________________________________
    # Alter
    BJ = input_BJ.get()
    FkorAin = getFkorA(2023 - BJ, False)
    FkorAwib = getFkorA(2023 - BJ, True)
    # Höhe
    FkorH = getFkorH(input_Lvl.get())
    # Nutzung
    FkorN = getFkorN(input_GebN.get())
    # Von Baukosten anteilige Technikkosten pro m² _______________
    TK = anteilP(input_TKperc.get(), input_BK.get())
    # TGA Herstellungskosten gesamtes Gebäude ____________________
```

```
    HK = round(mult(input_NUF.get(), TK), 2)
    # TGA Wiederbeschaffungswert ____________________________________
    WBW = round(WBWert(HK, input_BKIa.get(), input_BKIbj.get()), 2)
    # operative Jahresgesamtkosten ____________________________________
    # Wartung Inspektion Bedienung
    JGKopWib = round(WBW * getJKF(input_GebN.get())[0] * FkorAwib, 2)
    # Instandhaltung
    JGKopIn = round(WBW * getJKF(input_GebN.get())[1] * FkorAin, 2)
    # Betrieb
    JGKopBe = round(WBW * getJKF(input_GebN.get())[2] * FkorAwib, 2)
    # Wegekosten ____________________________________
    wegK = round(JGKopBe * getfWeg(input_Weg.get()), 2)
    # Fremdvergabeanteil ____________________________________
    fva = (input_FvWI.get() / 100 + input_FvIn.get() / 100) / 2
    # administrative Kosten ____________________________________
    Kad = round(JGKopBe * interp(0, 100, 0.1, 0.2, fva), 2)
    # Gesamtkosten ____________________________________
    GK = round(JGKopBe + Kad, 2)
    # Eigenanteile ____________________________________
    # Eigenanteil Instandsetzung
    eaIn = interp(0, 100, 1.0, 0.0, input_FvIn.get())
    # Eigenanteil Wartung/Inspektion
    eaWI = interp(0, 100, 1.0, 0.0, input_FvWI.get())
    # Eigenmaterialkosten ____________________________________
    matK = getmatK(JGKopIn, JGKopWib, eaIn, eaWI)
    # Personal-Sollkosten ____________________________________
    persK = round(GK - matK - input_FK.get(), 2)
    # Personal Soll-Stellenzahl ____________________________________
    persS = round(persK / input_PKa.get(), 1)
    # Aus-
gabe____________________________________________________________
_____________________________

    output_TK.set(TK)
    output_HK.set(HK)
    output_WBW.set(WBW)
    output_JkwWIB.set(getJKW(FkorN, FkorAwib, FkorH))
    output_JkwIn.set(getJKW(FkorN, FkorAin, FkorH))
    output_oJGKWIB.set(JGKopWib)
    output_oJGKIn.set(JGKopIn)
    output_oJGKBe.set(JGKopBe)
    output_aK.set(Kad)
    output_WK.set(wegK)
    output_emK.set(matK)
    output_PK.set(persK)
    output_PSS.set(persS)
    output_GKoW.set(GK)
    output_GKmW.set(GK + wegK)

# JSON-Import ____________________________________________________
f = open('JahreskostenfaktorenDIN.json')
JKFData = json.load(f)

# GUI
_____________________________________________________________
_____________________________

window = tk.Tk()
window.title('BK-Berechnung')
window.geometry('1280x1000')
```

```
frame_main = tk.Frame(master=window, width=100)
frame_in = tk.Frame(master=frame_main, width=50)
frame_out = tk.Frame(master=frame_main, width=50)
frame_set = tk.Frame(master=window, width=50, height=5, bg='#666666')
btn_quit = tk.Button(
    master=frame_set,
    command=Close,
    text='X',
    width=5,
    height=2,
    bg='#666666',
    fg='white',
    bd=0
).pack(side=tk.RIGHT, anchor='n',ipady=2,ipadx=2)
btn_ctrl = tk.Button(
    master=frame_set,
#    command=,
    text='SETTINGS',
    width=15,
    height=2,
    bg='#666666',
    fg='white',
    bd=0
).pack(side=tk.RIGHT, anchor='n',ipady=2,ipadx=2)
btn_chk = tk.Button(
    master=frame_set,
    command=verify,
    text='CHECK',
    width=15,
    height=2,
    bg='#22aa22',
    fg='white',
    bd=0,
    font=('Arial', 9, 'bold')
).pack(side=tk.LEFT, anchor='n',ipady=2,ipadx=2)
frame_set.pack(fill=tk.X,side=tk.TOP, expand=True)
frame_main.pack(fill=tk.BOTH, side=tk.BOTTOM,expand=True)
frame_in.pack(fill=tk.BOTH, side=tk.LEFT,expand=True, ipadx=20, i-
pady=20)
frame_out.pack(fill=tk.BOTH, side=tk.RIGHT,expand=True, ipadx=20, i-
pady=20)


input_NUF = tk.DoubleVar()
input_BK = tk.DoubleVar()
input_TKperc = tk.DoubleVar()
input_GebN = tk.IntVar()
input_Lvl = tk.IntVar()
input_BJ = tk.IntVar()
input_FvIn = tk.DoubleVar()
input_FvWI = tk.DoubleVar()
input_BKIa = tk.DoubleVar()
input_BKIbj = tk.DoubleVar()
input_Weg = tk.IntVar()
input_FK = tk.DoubleVar()
input_PKa = tk.DoubleVar()

label_Main = tk.Label(
    master=frame_in,
    text='Betriebskostenberechnung',
```

```
    font=('Arial',14,'bold')
).pack(side=tk.TOP,padx=(20,0),pady=(20,40),anchor='w')
label_NUF = tk.Label(
    master=frame_in,
    text='Nutzfläche in m² :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_NUF = tk.Entry(
    master=frame_in,
    textvariable=input_NUF
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_BK = tk.Label(
    master=frame_in,
    text='Bauwerkskosten pro m² :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_BK = tk.Entry(
    master=frame_in,
    textvariable=input_BK
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_TK = tk.Label(
    master=frame_in,
    text='davon Technikkosten in % :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_TK = tk.Entry(
    master=frame_in,
    textvariable=input_TKperc
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_GebN = tk.Label(
    master=frame_in,
    text='Gebäudenutzungs-ID :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_GebN = tk.Entry(
    master=frame_in,
    textvariable=input_GebN
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_Lvl = tk.Label(
    master=frame_in,
    text='Anzahl der Geschosse :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_Lvl = tk.Entry(
    master=frame_in,
    textvariable=input_Lvl
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_BJ = tk.Label(
    master=frame_in,
    text='Baujahr :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_BJ = tk.Entry(
    master=frame_in,
    textvariable=input_BJ
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_FvIn = tk.Label(
    master=frame_in,
    text='Fremdvergabeanteil für Instandhaltung in % :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_FvIn = tk.Entry(
    master=frame_in,
    textvariable=input_FvIn
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_FvWI = tk.Label(
    master=frame_in,
```

```
    text='Fremdvergabeanteil für Wartung und Inspektion in % :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_FvWI = tk.Entry(
    master=frame_in,
    textvariable=input_FvWI
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_BKIa = tk.Label(
    master=frame_in,
    text='BKI aktuell :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_BKIa = tk.Entry(
    master=frame_in,
    textvariable=input_BKIa
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_BKIbj = tk.Label(
    master=frame_in,
    text='BKI Bezugsjahr :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_BKIbj = tk.Entry(
    master=frame_in,
    textvariable=input_BKIbj
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_Weg = tk.Label(
    master=frame_in,
    text='Wegentfernungsgrad (1-3) :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_Weg = tk.Entry(
    master=frame_in,
    textvariable=input_Weg
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_FK = tk.Label(
    master=frame_in,
    text='Getätigte Fremdkosten :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_FK = tk.Entry(
    master=frame_in,
    textvariable=input_FK
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')
label_PKa = tk.Label(
    master=frame_in,
    text='Personalkostenäquivalent :'
).pack(side=tk.TOP,padx=(20,0),anchor='w')
entry_PKa = tk.Entry(
    master=frame_in,
    textvariable=input_PKa
).pack(side=tk.TOP,padx=(20,0),pady=(5,10),anchor='w')

output_TK = tk.DoubleVar()
output_HK = tk.DoubleVar()
output_WBW = tk.DoubleVar()
output_JkwWIB = tk.DoubleVar()
output_JkwIn = tk.DoubleVar()
output_oJGKWIB = tk.DoubleVar()
output_oJGKIn = tk.DoubleVar()
output_oJGKBe = tk.DoubleVar()
output_aK = tk.DoubleVar()
output_WK = tk.DoubleVar()
output_emK = tk.DoubleVar()
output_PK = tk.DoubleVar()
output_PSS = tk.IntVar()
```

```
output_GKoW = tk.DoubleVar()
output_GKmW = tk.DoubleVar()

label_out_Main = tk.Label(
    master=frame_out,
    text='Berechnungsergebnisse',
    font=('Arial',14,'bold')
).pack(side=tk.TOP,pady=(20,40),anchor='w')
label_ro_TK = tk.Label(
    master=frame_out,
    text='Technikkosten in €/m²'
).pack(side=tk.TOP, pady=(10, 5),anchor='w')
label_r_TK = tk.Label(
    master=frame_out,
    textvariable=output_TK
).pack(side=tk.TOP,anchor='w')
label_ro_HK = tk.Label(
    master=frame_out,
    text='Herstellungskosten der TGA in €'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_HK = tk.Label(
    master=frame_out,
    textvariable=output_HK
).pack(side=tk.TOP,anchor='w')
label_ro_WBW = tk.Label(
    master=frame_out,
    text='Wiederbeschaffungswert der TGA in €'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_WBW = tk.Label(
    master=frame_out,
    textvariable=output_WBW
).pack(side=tk.TOP,anchor='w')
label_ro_JkWIB = tk.Label(
    master=frame_out,
    text='Jahreskorrekturwert Wartung/Inspektion/Bedienung'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_JkwWIB = tk.Label(
    master=frame_out,
    textvariable=output_JkwWIB
).pack(side=tk.TOP,anchor='w')
label_ro_JkIn = tk.Label(
    master=frame_out,
    text='Jahreskorrekturwert Instandsetzung'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_JkwIn = tk.Label(
    master=frame_out,
    textvariable=output_JkwIn
).pack(side=tk.TOP,anchor='w')
label_ro_JGKWIB = tk.Label(
    master=frame_out,
    text='Operative Jahresgesamtkosten für Wartung, Inspektion und Be-
dienung in € p.a.'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_oJGKWIB = tk.Label(
    master=frame_out,
    textvariable=output_oJGKWIB
).pack(side=tk.TOP,anchor='w')
label_ro_JGKIn = tk.Label(
    master=frame_out,
    text='Operative Jahresgesamtkosten für Instandsetzung in € p.a.'
```

```
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_oJGKIn = tk.Label(
    master=frame_out,
    textvariable=output_oJGKIn
).pack(side=tk.TOP,anchor='w')
label_ro_JGKBe = tk.Label(
    master=frame_out,
    text='Operative Jahresgesamtkosten für Betrieb in € p.a.'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_oJGKBe = tk.Label(
    master=frame_out,
    textvariable=output_oJGKBe
).pack(side=tk.TOP,anchor='w')
label_ro_aK = tk.Label(
    master=frame_out,
    text='Administrative Kosten in € p.a.'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_aK = tk.Label(
    master=frame_out,
    textvariable=output_aK
).pack(side=tk.TOP,anchor='w')
label_ro_WK = tk.Label(
    master=frame_out,
    text='Wegekosten in € p.a.'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_WK = tk.Label(
    master=frame_out,
    textvariable=output_WK
).pack(side=tk.TOP,anchor='w')
label_ro_emK = tk.Label(
    master=frame_out,
    text='Eigenmaterialkosten in € p.a.'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_emK = tk.Label(
    master=frame_out,
    textvariable=output_emK
).pack(side=tk.TOP,anchor='w')
label_ro_PK = tk.Label(
    master=frame_out,
    text='Personalkosten in € p.a.'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_PK = tk.Label(
    master=frame_out,
    textvariable=output_PK
).pack(side=tk.TOP,anchor='w')
label_ro_PSS = tk.Label(
    master=frame_out,
    text='Soll-Stellen Personal'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_PSS = tk.Label(
    master=frame_out,
    textvariable=output_PSS
).pack(side=tk.TOP,anchor='w')
label_ro_GKoW = tk.Label(
    master=frame_out,
    text='Gesamtkosten ohne Wegekosten in € p.a.'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_GKoW = tk.Label(
    master=frame_out,
    textvariable=output_GKoW
```

```
).pack(side=tk.TOP,anchor='w')
label_ro_GKmW = tk.Label(
    master=frame_out,
    text='Gesamtkosten mit Wegekosten in € p.a.'
).pack(side=tk.TOP, pady=(10,5),anchor='w')
label_r_GKmW = tk.Label(
    master=frame_out,
    textvariable=output_GKmW
).pack(side=tk.TOP,anchor='w')

# Eingabe
```

```
nuf = 6000
age = 2023
BWK = 3801
TKperc = 30
levels = 3
NFID = 3
BKIa = 100
BKIbj = 100
entfGrad = 1
fvaIn = 50
fvaWI = 50
fremdK = 0
persKaq = 35000

input_NUF.set(nuf)
input_BK.set(BWK)
input_TKperc.set(TKperc)
input_GebN.set(NFID)
input_Lvl.set(levels)
input_BJ.set(age)
input_FvIn.set(fvaIn)
input_FvWI.set(fvaWI)
input_BKIa.set(BKIa)
input_BKIbj.set(BKIbj)
input_Weg.set(entfGrad)
input_FK.set(fremdK)
input_PKa.set(persKaq)


window.mainloop()
```

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Carl Philipp Fürstenberg