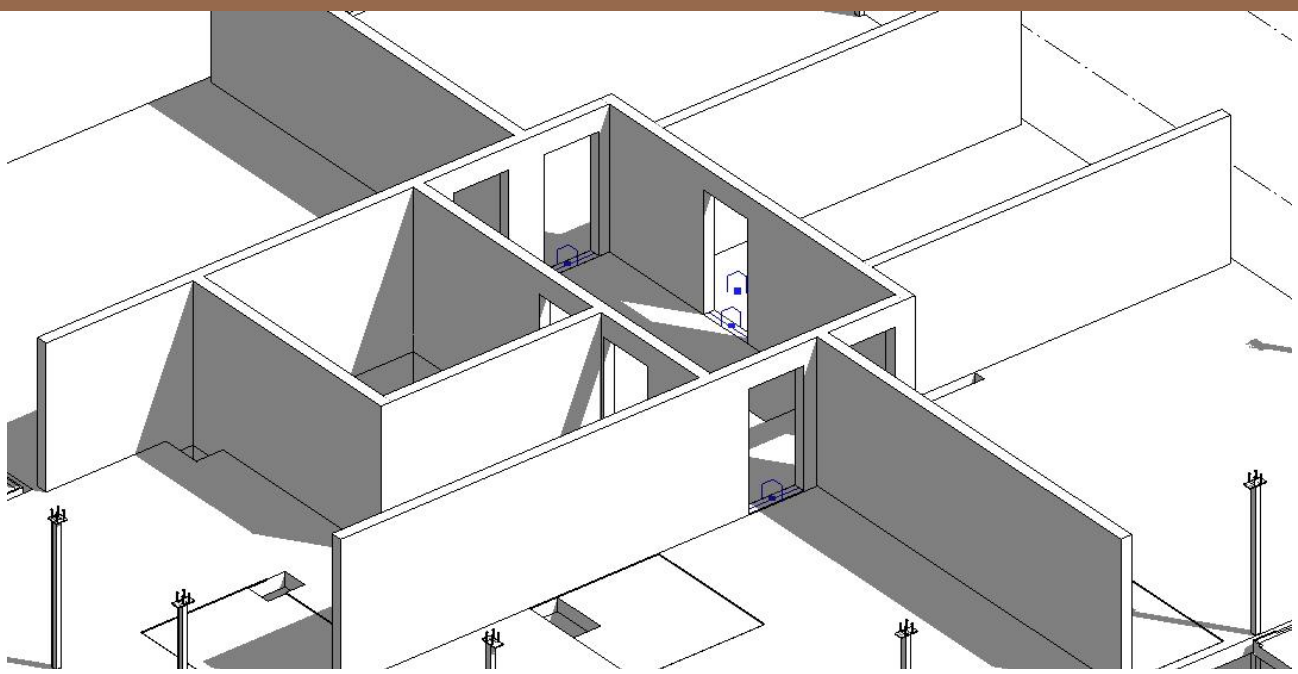
CHALMERS



Optimering av flödesprocesser i projektering med Dynamo

En utredning kring möjligheterna och begränsningarna att optimera datautbytet mellan arkitektur och konstruktion med Dynamo

Examensarbete inom högskoleingenjörsprogrammet
Samhällsbyggnadsteknik

RAMEZ SBINATI
OSCAR SÖDERBERG

INSTITUTIONEN FÖR ARKITEKTUR OCH SAMHÄLLSBYGGNADSTEKNIK

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2024
www.chalmers.se

EXAMENSARBETE ACEX20

Optimering av flödesprocesser i projektering med Dynamo

En utredning kring möjligheterna att optimera datautbytet mellan arkitektur och konstruktion med Dynamo

Examensarbete inom högskoleingenjörsprogrammet

Samhällsbyggnadsteknik

RAMEZ SBINATI

OSCAR SÖDERBERG

Institutionen för arkitektur och samhällsbyggnadsteknik
Avdelningen för Construction Management
Forskarguppsnamn
CHALMERS TEKNISKA HÖGSKOLA

Göteborg, 2024

Optimering av flödesprocesser i projektering med Dynamo

En utredning kring möjligheterna att optimera datautbytet mellan arkitektur och konstruktion med Dynamo

Examensarbete inom högskoleingenjörsprogrammet

Samhällsbyggnadsteknik

RAMEZ SBINATI

OSCAR SÖDERBERG

Examensarbete ACEX20
Institutionen för arkitektur och samhällsbyggnadsteknik
Chalmers tekniska högskola 2024

Institutionen för arkitektur och samhällsbyggnadsteknik
Avdelningen för Construction Management
Chalmers tekniska högskola
412 96 Göteborg
Telefon: 031-772 10 00

Omslag:
Resultat av skript programmerat i fallstudie 1.
Institutionen för arkitektur och samhällsbyggnadsteknik
Göteborg 2024

Optimering av flödesprocesser i projektering med Dynamo

En utredning kring möjligheterna att optimera datautbytet mellan arkitektur och konstruktion med Dynamo

Examensarbete inom högskoleingenjörsprogrammet
Samhällsbyggnadsteknik

RAMEZ SBINATI

OSCAR SÖDERBERG

Institutionen för arkitektur och samhällsbyggnadsteknik
Construction Management
Chalmers tekniska högskola

SAMMANFATTNING

Byggbranschen har digitaliserats och med detta medförs en projekteringsprocess präglat av digitala modeller och ett informationsutbyte av projekts olika parter. Med hjälp av modelleringsprogram, däribland Revit, skapas nya processflöden i projekteringsprocessen. Med visuella programmeringsverktyg, såsom Dynamo, har modelleringsprogrammens möjligheter vidgats och utveckling av effektivare processflöden har blivit mer tillgänglig för den enskilde ingenjören. Detta arbete har utforskat Dynamos möjligheter till att skapa automatiserade processflöden mellan konstruktörer och arkitekter.

Arbetet har gjorts på uppdrag av VBK genom en litteraturstudie av den teoretiska projekteringsprocessen, en intervjustudie av VBK:s nuvarande projekteringsprocess samt en fallstudie där vi laborativt programmerat skript. Fallstudien uppgick till fem olika fall i datautbytet mellan konstruktörer och arkitekter, där tre olika K-modeller har varit utgångspunkt.

Fallen behandlar olika moment där metadata, skapandet av objekt, geometrier och abstrakta villkor hanteras. Det laborativa arbetet kulminerade i sju olika skript som belyste Dynamos begränsningar och möjligheter för dessa moment genom den problematik som uppstod vid programmeringen.

Resultaten visar att Dynamo kan eliminera manuellt arbete vilket kan bidra till tidsbesparing, mindre risk för mänskliga fel, eliminera mindre givande och repetitiva uppgifter. Arbetets slutsats resulterade i att det aktuella projekteringsskede för skriptet, projektets storlek, detaljeringsgrad och skriptets omfång, är styrande för möjligheten av automatisering och optimering av respektive flödesprocess. Skripten har olika känslighet för variationer i modeller och varierande metadatastruktur.

Nyckelord: Automatisering, BIM, Dynamo, Flödesprocesser, Metadata, Projekteringsprocessen, Revit

Optimization of Flow Processes in Design with Dynamo
An investigation into the possibilities of optimizing data exchange between architecture and construction with Dynamo.
Degree Project in the Engineering Programme
Civil and Environmental Engineering

RAMEZ SBINATI

OSCAR SÖDERBERG

Department of Architecture and Civil Engineering
Division of Construction Management
Chalmers University of Technology

ABSTRACT

The construction industry has been digitalized, bringing with it a new design process characterized by digital models and information exchanges between the various disciplines. Using modelling programs, such as Revit, new process flows in the design process has been created. With visual programming tools such as Dynamo, the capabilities of modelling programs have been expanded and the development of more optimized workflows has been more accessible to engineers. This study has explored the potential of Dynamo to create automated process flows between structural engineers and architects.

The study was conducted on behalf of VBK through a literature study of the theoretical design process, an interview study of VBK's current design process and a case study where scripts were programmed. The case study treated 5 different scenarios of the data exchange between structural engineers and architects with three different structural models as canvases.

The cases treat different scenarios where metadata, the creation of objects, geometries and abstract conditions are managed. The experimental work culminated in seven different scripts that highlighted Dynamo's limitations and capabilities for the program's management of these scenarios.

The results show that Dynamo can eliminate manual work, which can contribute to time savings, reduce the risk of human errors and eliminate less rewarding and repetitive tasks. The conclusion of this study resulted that the current design stage, the project's size, the level of detail, and the scripts scope were decisive for the possibility of automation and optimization of the process flows. The scripts have different sensitivity to variations in the structural models and varying metadata structures.

Key words: Automation, BIM, Dynamo, Process flows, Metadata, Design Process, Revit

Innehåll

Förord

Detta examensarbete är skrivet på samhällsbyggnadsprogrammet, högskoleingenjör vid Chalmers Tekniska Högskola. Arbetet omfattar 15 högskolepoäng, inleddes januari 2024 och avslutades juni 2024.

Vi vill tacka Mikael Johansson, vår handledare och examinator för stödet, inspirationen och rådgivningen under arbetets gång. Vi vill även tacka VBK och i synnerhet vår handledare Rasmus Öberg som gediget handlett oss, agerat bollplank och givit oss förmånen att använda VBK:s kontors som vår främsta arbetsplats.

Göteborg juni 2024
Ramez Sbinati
Oscar Söderberg

Beteckningar

3D	Tredimensionell
(A)	Referering till arkitekt som disciplin
A-modell	Modell av arkitekt
API	Applikationsprogrammeringsgränssnitt, ett gränssnitt mellan applikationsprogrammet och mjukvarufunktioner som lagras i ett kodbibliotek
BIM	Building Information Model
CAD	Computer Aided Design
Dynamo	Instickspogram för visuell programmering av skript i några av Autodesks produkter
IFC	Filformat som används gemensamt i olika modelleringsprogram
Input	Alternativ för inmatning
(K)	Referering till Konstruktör som disciplin
K-modell	Modell av konstruktör
Metadata	Data som beskriver annan information, till exempel ett objekt
Output	Alternativ för utmatning
Python	Ett programmeringspråk som Dynamo är baserat på
Robusthet	Förmåga att motstå förändringar i underlag och mjukvaruuppdateringar
Skript	Program som utför uppgifter i ett annat program

1 Inledning

Kapitel 1 innehåller de grunder och ramverk arbetet bygger på. Dessa grunder är bakgrund, syfte, frågeställningar, avgränsningar och metod.

1.1 Bakgrund

Byggbranschen undergår en digitalisering där traditionella arbetssätt i byggprocessen alltmer byts ut mot digitala, däribland inom byggprojektering. Under projekteringskedet börjar projekt gestaltas och utformas med ritningar och modeller från olika samverkande discipliner (Boverket, 2021a). Dagens byggprojekt medför en oerhört stor mängd information som hanteras mellan disciplinerna. Det interna och externa informationsutbyten görs i långa led samt kan ofta kräva handpåläggning av mottagande part. Beroende på hur informationen är strukturerad, tolkas och projekttyp, varierar även hur informationen lagras i modellen.

Det komplexa informationsflödet utgör en utmaning för dagens ingenjörer. Ofta återkommer upprepande uppgifter som kan vara både tidskrävande och repetitiva i projekteringsprocessen. Behovet av effektivare informationsutbyten och bättre processflöden ökar ständigt som underlättar för projekteringsprocessen.

Branschen arbetar med modelleringsprogram, exempelvis *Revit* för att framställa modeller och handlingar. Autodesks produkter, däribland Revit, har en inbyggd visuell programmeringsapplikation som heter Dynamo. Trots att Dynamo har medföljt Revit sedan 2020 används det inte till sin fulla potential under projektering.

1.2 Syfte och frågeställningar

Syftet med denna rapport var att undersöka hur användningen av visuell programmering i Dynamo kan effektivisera och förbättra processflödena mellan konstruktörer och arkitekt. Med hjälp av automatisering och förbättring av processflöden kan projekts datahantering effektiviseras samt dataset förädlas med mer information. Bättre processflöden skulle kunna bidra till ett enklare samarbete mellan olika discipliner vilket i sin tur hade sparat tid och gett utrymme för alternativa lösningar, däribland hållbarare lösningar. Arbetets syfte var även att belysa de begränsningar Dynamo kan ha under konstruktörers projektering.

Frågeställningar:

- Vad för möjligheter har Dynamo att effektivisera/automatisera processflöden genom optimering eller förädling av data i datautbytet mellan (K) och (A) under projektering?
- Vilka begränsningar och logiska problem har Dynamo vid en effektivisering/automatisering av processflöden mellan (K) och (A)?

1.3 Avgränsning

Arbetet kommer endast utgå från externt processflöde mellan arkitekten och konstruktören, alltså inte hantering av alla flöden i projekteringsprocessen. Under fallstudien kommer arbetet utgå utifrån VBK:s egna framtagna modeller. Programvarorna som används är Revit och Dynamo. Fallstudien utgår endast ifrån Dynamos inbyggda funktioner och noder samt externa paket och inte egenkodade block. Studien utgår från en svensk kontext det vill säga, svenska byggstandarder och regler.

1.4 Metod

För att få en heltäckande bild samt bygga förkunskaper inom ämnet genomfördes en förstudie med inläsning inom Dynamo och projektering samt laborering med Revit och Dynamo. Laboreringen bestod av följandet av introducerande videobeskrivningar på Dynamos webbplats samt utförande av moment i Revit med hjälp av Dynamo.

Genom en litteraturstudie fördjupades arbetet i projekteringsprocessens teoretiska grunder och teori för de programvaror som användes. Under litteraturstudien sonderades även tidigare publikationer om Dynamo i projekteringsprocessen.

Genom intervjustudie och kontinuerlig dialog med Rasmus Öberg, redovisningsingenjör på VBK, undersöktes hur projekteringsprocessen genomförs i praktiken. Under dialogen presenterades även fallen samt dess grundläggande krav, för att sätta arbetet i ett verkligt sammanhang.

Till sist utfördes en fallstudie som bestod av fem typiska fall för projektering inom konstruktion. Studien genomfördes i Revit och Dynamo där skript programmerades för att automatisera processflödet i respektive fall. Fallen utgick utifrån VBK:s framtagna modeller samt dess korresponderande A-modeller som underlag. Momenten behandlar överföring av data, objektskapande, hantering av geometriska data, hantering av metadata samt förädling av dataset. För de hinder och problematik som uppstod i programmeringen av skripten, undersöktes lösningar från nätforum, diverse webbplatser och genom handledning med Rasmus.

2 Teoretisk bakgrund

Under detta kapitel ges grundläggande teori för att ge kontext och teoretisk förståelse för arbetet.

2.1 Tidigare arbeten

För att undersöka vad för arbeten som redan utförts granskade vi databaser för tidigare examensarbeten. Vid sökning av ”Dynamo” på Chalmers, KTH:s och LTH:s databaser för examensarbeten hittades 27 examensarbeten. Arbetena karaktäriseras främst av användandet av det visuella programmeringsverktyget för att utveckla metoder inom specifika problemområden, exempelvis energianalys, strukturell analys eller parametrisk design. Utav dessa 27 arbeten hade fyra arbeten som syfte att undersöka möjligheter för visuell programmering inom byggprojektering.

Dessa fyra är:

- ”*Automatisering av informationshantering inom BIM*” (Wikland & Karlgren, 2022).
- ”*En analys av arbetsflödet för 3D - armering mellan Revit och Robot*” (Hujdur & Karlsson, 2020)
- ”*Armering i pålfundament - Effektivare byggprojektering med grafisk programmering*” (Schmied & Strömberg, 2019)
- ”*Effektivisering av byggprojektering med hjälp av grafisk programmering*” (Ahlner & Dahl, 2018).

De tre första arbetena har som huvudfokus informationshantering för klimatkalkyler respektive skapandet av armering i modellen. Det är endast det sista arbetet, *Effektivisering av byggprojektering med hjälp av grafisk programmering*, som har som huvudfokus att undersöka hur visuell programmering kan användas i byggprojektering generellt. Arbetet hanterar moment i byggprojektering såsom vyinställningar, metadata, skapandet av ritningar och modellering. Arbetets slutsats var att de skapade skripten möjliggjorde en effektivisering i form av tidsbesparing i projekteringen.

2.2 BIM

Building Information Model (BIM) är en process som innebär skapandet av en digital modell av ett byggprojekt (Trimble, 2022). BIM skiljer sig från traditionell 3D-CAD genom intelligensnivån hos objekten i modellen. Intelligensen hos objekten är information såsom geometri, materialdata och relationer till andra objekt. Genom delning av modellen mellan discipliner möjliggör BIM informationsutbyten där discipliner kan ta del av varandras arbeten och således öka samverkan inom byggprojektering. BIM är även användbart i produktion och förvaltning där modellen redogör information som kan underlätta beslutstagande.

BIM-processen delas ofta in i nivåer av mognadsgrad som beskriver i vilken utsträckning arbetssättet har tillämpats (Trimble, 2022).

- **Nivå 0**: Arbetssättet används inte i vidare utsträckning, till exempel genom minimal samverkan och användning av 2D-CAD-ritningar.

- **Nivå 1**: Användning av främst 2D-CAD ritningar och minimal samverkan mellan aktörer men 3D-modeller används för koncept och visualisering av byggnaden.

- **Nivå 2**: Discipliner arbetar i sina egna 3D-modeller samt delar information i form av modeller som exporteras i ett gemensamt filformat, exempelvis IFC.

- **Nivå 3**: Discipliner samverkar i samma centrala 3D-modell vilket underlättar samarbetet och upptäckt av kollisioner av varandras objekt.

- **Nivå 4:** Introduktion av tidsaspekten i modellen och redogörelse för tidsåtgången av exempelvis montering av byggelement i produktionen.

- **Nivå 5:** Kostnader adderas till modellen, exempelvis kostnadsuppskattningar eller budgetering.

- **Nivå 6:** Energyanalys tilläggs till modellen för att redogöra byggnadens energieffektivitet i exempelvis förvaltningsskedet.

2.3 Programvaror

I det här avsnittet presenteras de programvaror arbetet avgränsats till.

2.3.1 Autodesk Revit (Version 2023.1)

Autodesk Revit är ett modelleringsprogram som används i byggbranschen och är anpassat för olika discipliner, däribland arkitekter och konstruktörer. (Microsol Resources, 2021). Programmet möjliggör 3D-CAD-modellering av konstruktioner och har möjligheten att lagra information i modellens objekt. Den lagrade informationen kan exempelvis vara olika standardiserade klassificeringar men också andra parametrar såsom brandklass, material, dimensioner, priser eller leverantörer. Revits funktion att lagra information ger förutsättningar för att ha uppdaterad bygginformation i byggprojektet.

2.3.2 Dynamo

Dynamo är ett visuellt programmeringsverktyg som medföljer vissa Autodesk produkter såsom Civil 3D, Robot och Revit (DynamoBIM, 2024.). Programmet är open-source och ger användaren möjlighet att använda Revits API för att grafiskt skapa anpassade funktioner och skript som inte är fördefinierade i Autodesks produkter. Dynamo använder sig av förskapade noder som kopplas samman för att definiera instruktioner och relationer som i sin tur bildar flödesscheman av kod. Det finns också möjlighet att programmera egna noder genom ”code blocks”, Python-skript eller att installera paket programmerade av andra användare.

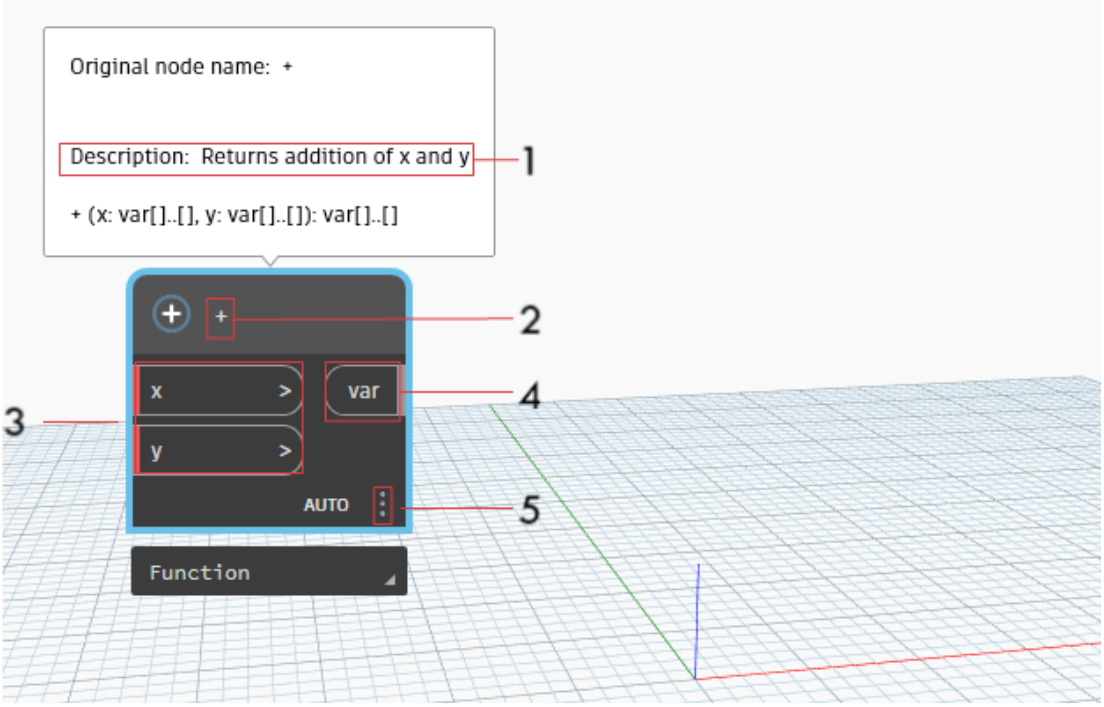
Möjligheten till att programmera med ett visuellt gränssnitt bryggar gapet för individer utan erfarenhet av programmering samt gör det enklare att se relationer hos kodens funktioner (DynamoBIM, 2024.). Programmet är utrustat med Dynamo Player vilket är ett gränssnitt som kör koden utan att användaren behöver någon förkunskap om skriptet (Autodesk, 2024a).

2.4 Dynamo: funktioner, noder och lacing

Syftet med avsnittet är att ge en grundläggande förståelse för Dynamos grundfunktioner vilket arbetet bygger på.

2.4.1 Noder

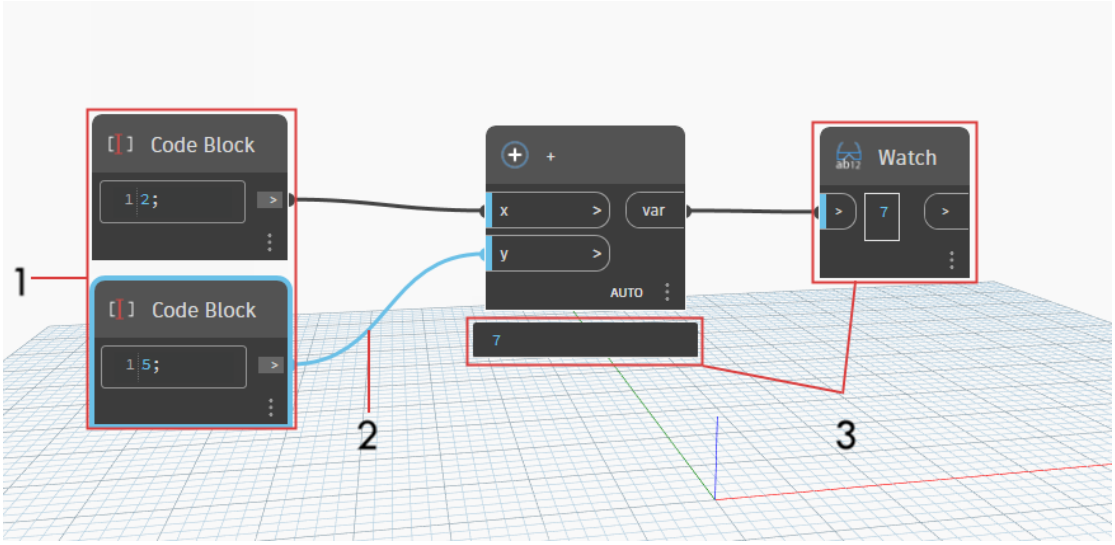
Noder i Dynamo är funktioner som vanligtvis består av inputs, outputs och en bakomliggande operation som utförs. Nodens har olika egenskaper (se Figur 1.) och man har möjlighet att sammankoppla noder till ett flöde (se Figur 2.) vilket är det som utgör skriptet



Figur 1: Exempel på en nod i Dynamo

Beskrivning av nodens egenskaper:

1. Beskrivning av noden
2. Namn: Namnet på noden, detta är en additions nod som heter +
3. Input: Indata till noden
4. Output: Nodens resultat av vad för input man matar in
5. Nod inställning: Kan ändra nodinställningar, exempelvis lacing för att specifikt matcha listornas input på ett specifikt sätt



Figur 2: Exempel på ett enkelt flöde i Dynamo

Beskrivning av ett enkelt flödes utformning:

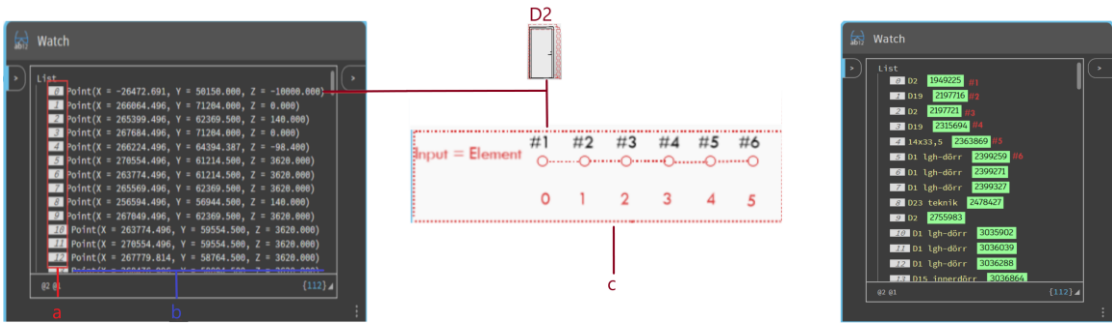
1. Input: 2 och 5
2. Ledning: Genom ledningar kopplar man ihop olika noder för att skapa relationer mellan noderna.
3. Output: Finns olika sätt att visualisera ett resultat. Man kan antingen hålla över nodinställningarna eller placera ut en ”Watch”-nod. Resultatet av detta flöde är “2+5=7”.

2.4.2 Geometrier i Dynamo

I Dynamo finns det olika geometriska noder som förhåller sig till varandra på olika sätt. Det är en form av hierarki där punkter ansluts och skapar linjer och linjer skapar ytor. Geometrier i Dynamo kan beskrivas utifrån position, storlek, form och förhållningen i ett dimensionellt rum. En punkt definieras endast av en koordinat i form av en siffra och har ingen dimension. En linje är definierad av en startpunkt och en slutpunkt och har en dimension. Ett plan är definierad av två linjer och har två dimensioner. En box är definierad av två plan och har tre dimensioner. Det finns både beroende och oberoende geometrier. Geometrier i dynamo begränsas inte till bara punkter, linjer, plan eller boxar för när en geometri är kurvig, svänger och vrider passar inte geometrin in i någon av kategorierna. Termen ”Curve” är ett samlingsbegrepp för alla punkt-geometrier som kan definieras genom en funktion för att skapa en krök. Geometrin behöver nödvändigtvis inte vara böjd för att defineras som en ”curve”. Alla kurvor har ett startläge och ett slutläge i form av punkter. Exempel på kurvor är cirklar och linjer. Beroende geometri är den som direkt är kopplad till Revit-modellen och ändras när man gör ändringar i projektet. Oberoende geometri är den som inte är direkt kopplad till Revit-projektet. Detta ger möjligheten att skapa geometrier i Dynamo utan att det omedelbart blir en del av Revit-modellen.

2.4.3 Listor i Dynamo

En lista är en samling data som kan vara exempelvis ett element, eller objekt. Listor används för att organisera och hantera data. Listor kan vara både enkla, komplexa samt innehålla andra listor vilket skapar hierarkier så kallat listor i listor. För varje individuellt element eller objekt sker en gruppering efter parametriska relationer i en datastruktur. I dynamo är listorna ordnade och det första objektet har index ”0” och det andra objektet har index ”1” och så vidare.



Figur 3: En lista som innehåller placering av dörrar i form av punkter

Beskrivning av en listas uppbyggnad (se figur 3.):

a. Index
b. Punktkoordinaten
c. Objekt

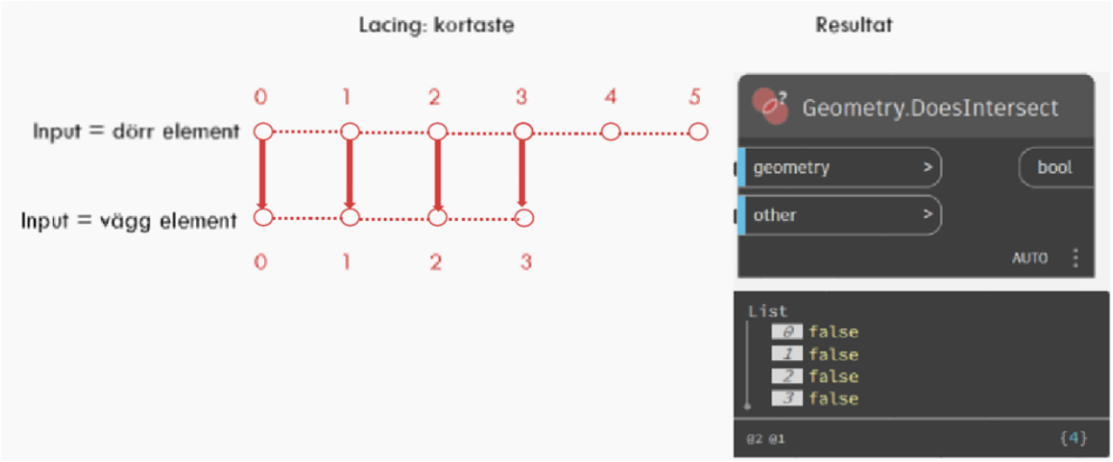
Utifrån listan får vi information om var alla dörrar är placerade i ett Revit projekt. Enligt listan så har första dörr elementet i listan, D2 som motsvarar #1, det vill säga index 0 motsvarande placering (-26472, 50150, -10000) som ses i figur 3.

2.4.4 Lacing

Lacing är en funktion som styr hur listor av olika längder interagerar med varandra. Att matcha ihop listor med data kan vara problematiskt då listorna innehåller indata av olika storlek. Beroende på hur dataalgoritmen matchar ihop data kan det leda till att man får olika resultat. Det finns fyra lägen man kan ställa in noden på: automatiskt, kortast, längst och kryssprodukt. För att visualisera hur Dynamo matchar ihop listor används noden ”Geometry.DoesIntersect” som ett exempel, där indata är geometrier som input från exempelvis dörrar och väggar. Noden gör en kollisionskontroll mellan de geometrier man matat in och resultatet blir antigen ”true” eller ”false” beroende på om geometrierna kolliderar eller inte.

2.4.4.1 Lacing: Kortast

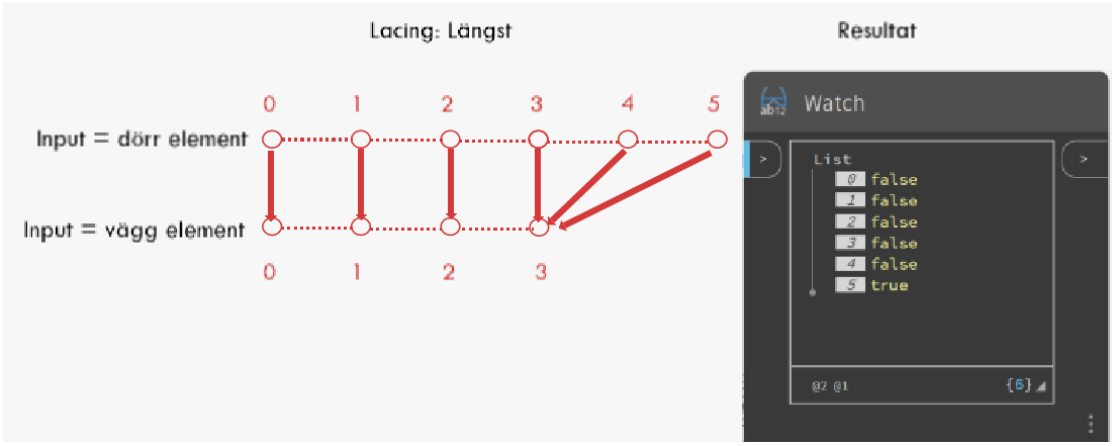
Detta är standardinställningen i Dynamo. Kollisionskontrollen associerar varje dörrelement i listan med motsvarande väggelement i andra listan. Om listorna har olika längder ignoreras elementen i den längre listan, som ses nedan i Figur 4. Kollisionskontrollen utförs mellan dörr- och väggelement med samma motsvarande index fram till ”index 3” därefter ignoreras dörrelementen med index ”4” och ”5”. Resultatet i Figur 4. visar att ingen kollision upptäcktes.



Figur 4: Lacing, kortast

2.4.4.2 Lacing: Längst

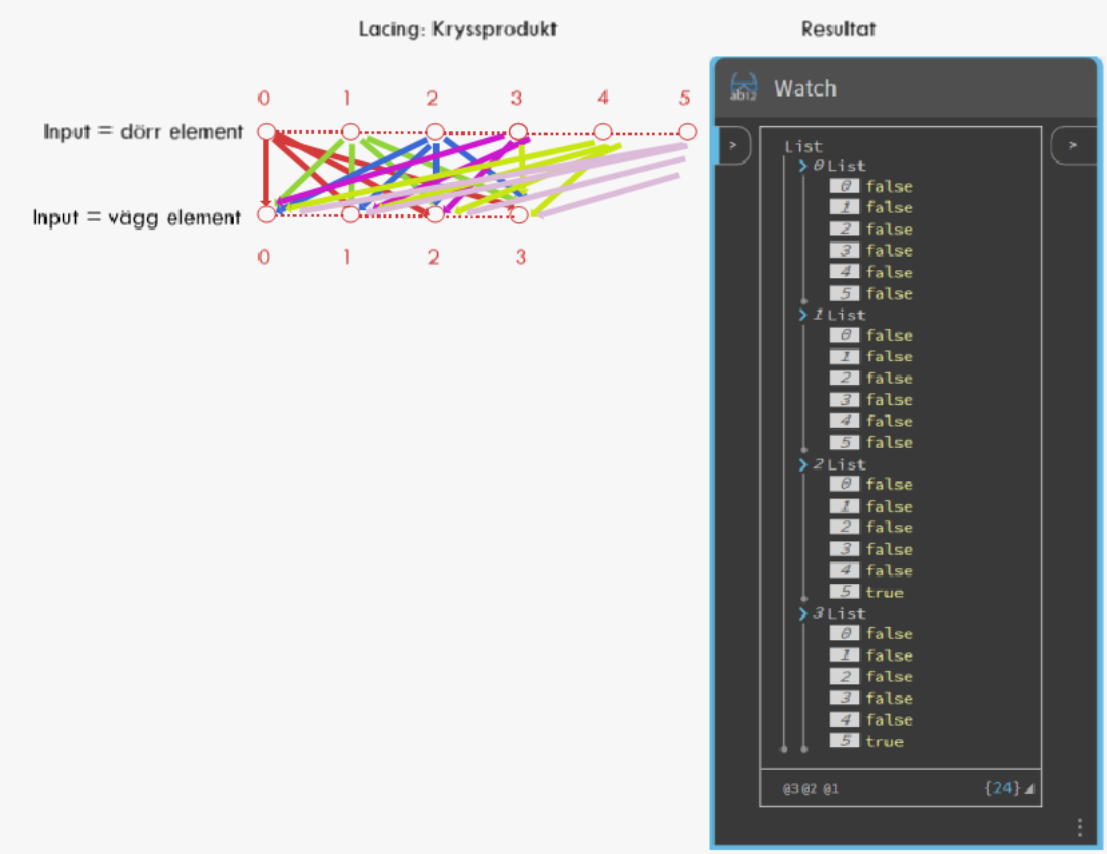
Längst lacing associerar varje element i listan med motsvarande element i den andra listan och återanvänder det sista elementet till det inte finns fler element kvar i den längsta listan. Resultatet i Figur 5. nedan visar att dörrelement med index ”5” kolliderar med väggelement med index ”3”.



Figur 5: Lacing, längst

2.4.4.3 Lacing: Kryssprodukt

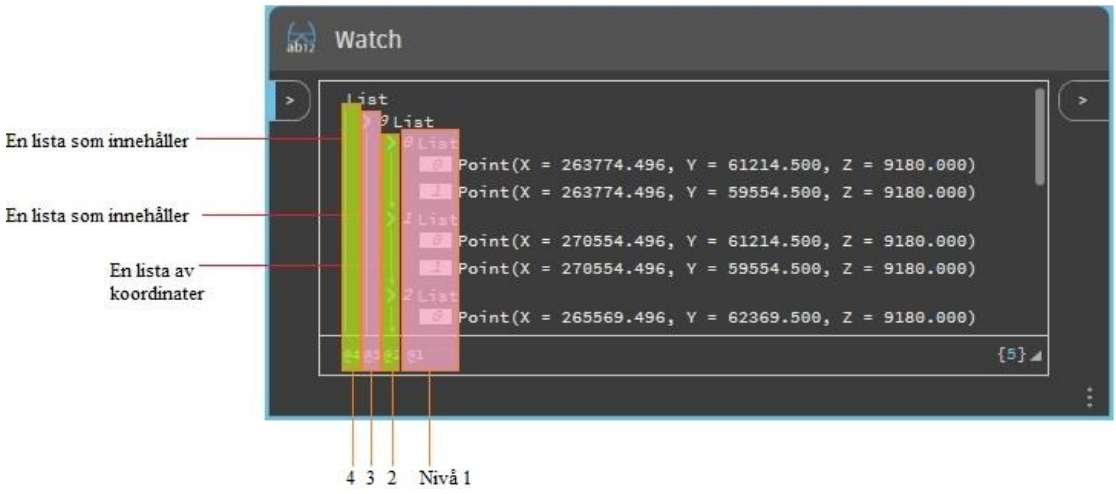
Kopplar varje element i en lista till alla element i den andra listan. Detta generar en större mängd resultat eftersom varje möjlig matchning görs mellan listorna. I Figur 6. observeras två kollisioner i resultatet när läget på lacing är kryssprodukt. Medan i Figur 4. hittades inga kollisioner och i Figur 5. hittades en.



Figur 6: Lacing, kryssprodukt

2.4.5 Nivåer

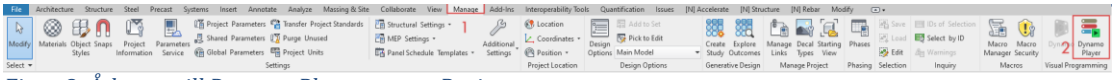
Listor kan förekomma i olika strukturer där en huvudlista kan innehålla allt från en till flertals listor. Då listor som innehåller andra listor kan nivåer skapas i Dynamo som refererar till strukturen och hierarkin av listor inom en lista. Nivåer ger möjligheten att hantera och manipulera data i komplexa liststrukturer samt hjälper till att specificera på vilken nivå en operation ska tillämpas.



Figur 7. Exempel på nivåer av listor i Dynamo.

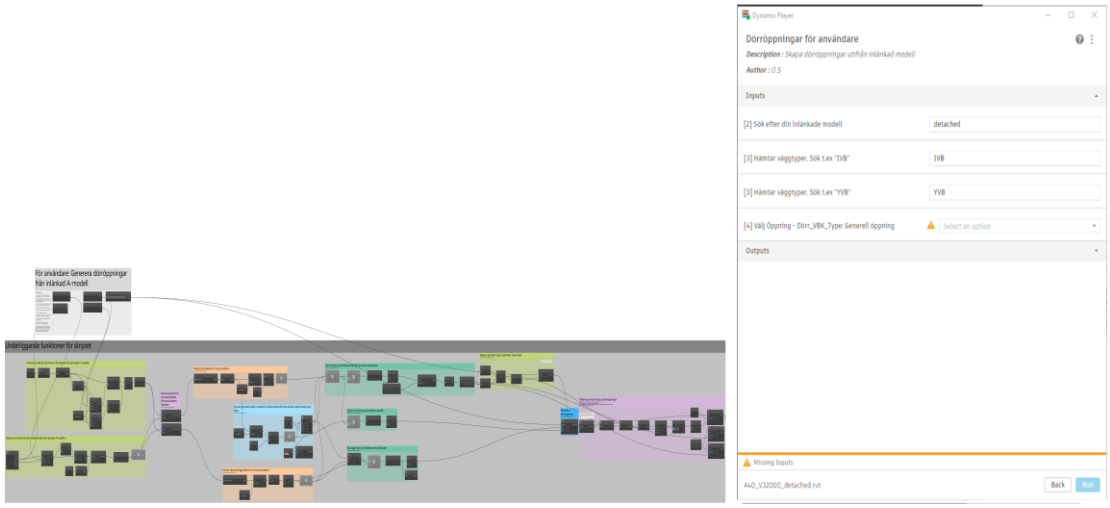
2.4.6 Dynamo Player

För att få tillgång till Dynamo Player klickar man först på manage-fliken och därefter Dynamo Player ikonen ikonen som illustreras i Figur 8.



Figur 8: Åtkomst till Dynamo Player genom Revit

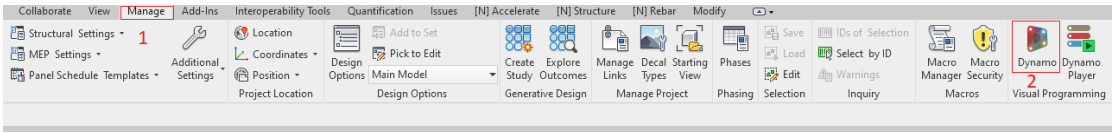
Dynamo Player är ett användarvänligt gränssnitt som gör det enkelt för användare utan tidigare erfarenhet av Dynamo att enkelt kunna använda sig av scriptet. Detta görs genom att högerklicka på noder i Dynamo och välja alternativet input. Användaren i Dynamo Player gränssnittet kan då enkelt ändra fördefinierade noder som har blivit satta som input. Figur 9 nedan, visar hur ett script i Dynamo kan förenklas med Dynamo Player.



Figur 9: Ett dynamoskript i Dynamo Player

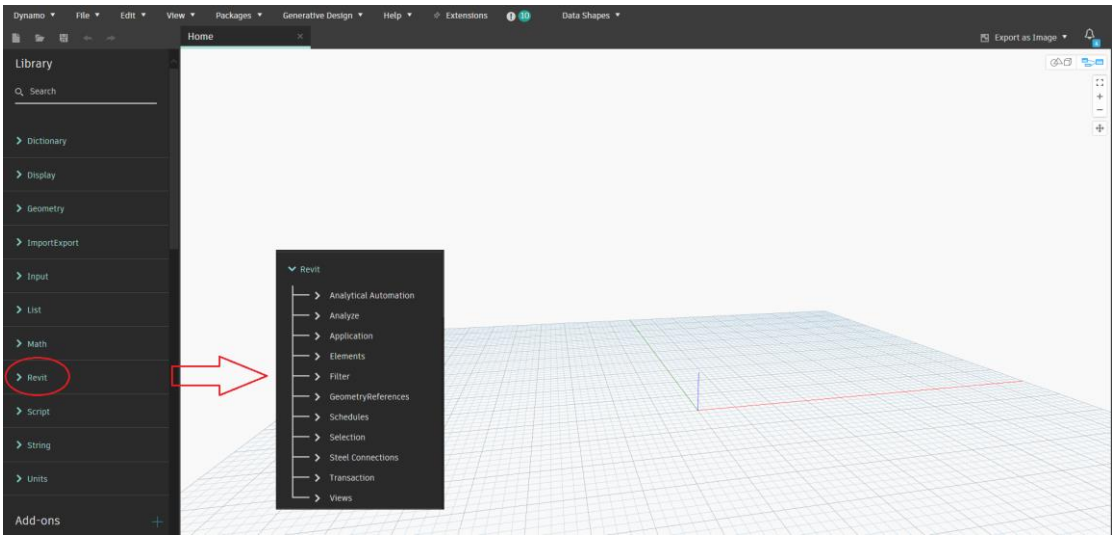
2.5 Dynamos koppling till Revit

För att få tillgång till Dynamo genom Revit klickar man först på manage fliken och därefter Dynamo ikonen som illustreras i Figur 10. Dynamo kan ändvändas som ett fristående program eller kopplar ihop med Revit. När Dynamo används fristående ger det användaren att skapa och manipulera geometrier, utföra matematiska och logiska operationer samt hantera data från olika källor. Detta ger användaren möjlighet att skapa komplexa modeller och analyser direkt i Dynamo.



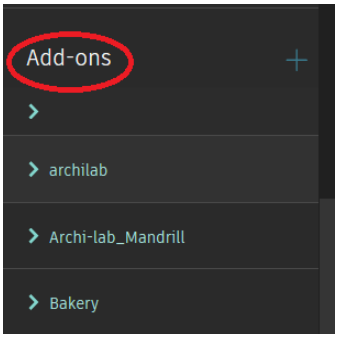
Figur 10: Åtkomst till Dynamo genom Revit

I Dynamo finns en kategori som kallas ”Revit” som innehåller förskapade noder. Det är ett omfattande tillägg till användargränssnittet som erbjuder noder som är specifikt utformade för att stödja Revit-flöden, se Figur 11.



Figur 11: Dynamos förskapade noder kopplat till Revit

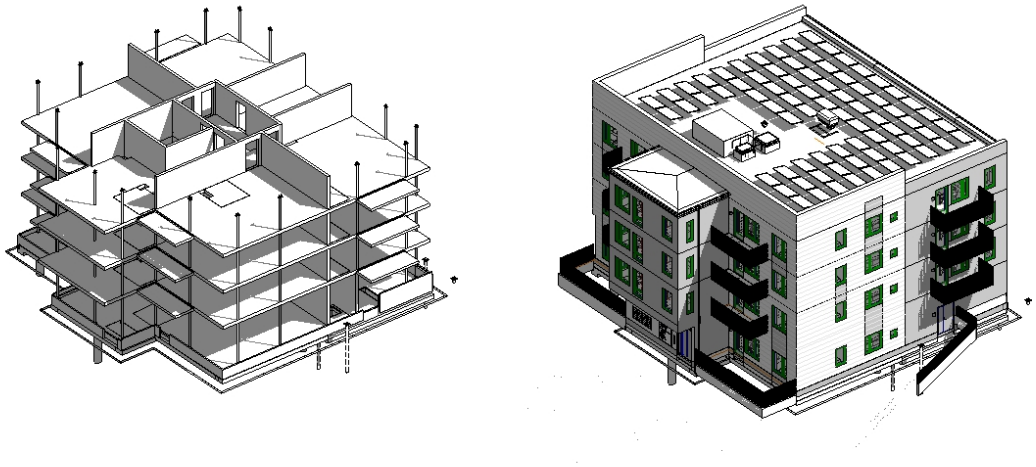
I en omfattande samling av förkonstruerade noder kopplat till Revit-flöden kan det emellertid vara sällsynt att stöta på en nod som motsvarar det specifika funktionsbehovet som du är ute efter. I sådana situationer kan det vara fördelaktigt att installera externa paket som potentiellt inkluderar en nod som adresserar den aktuella funktionsuppgiften. Externa paket utökar Dynamos grundläggande funktionalitet. Att installera färdigställda paket är effektivt då det avlägsnar kravet på kodningskunnande hos användaren där kodningen redan har utförts av andra. De installerade paketen hamnar under Add-ons, se Figur 12.



Figur 12. Exempel på färdig installerade paket

2.5.1 Länkade modeller

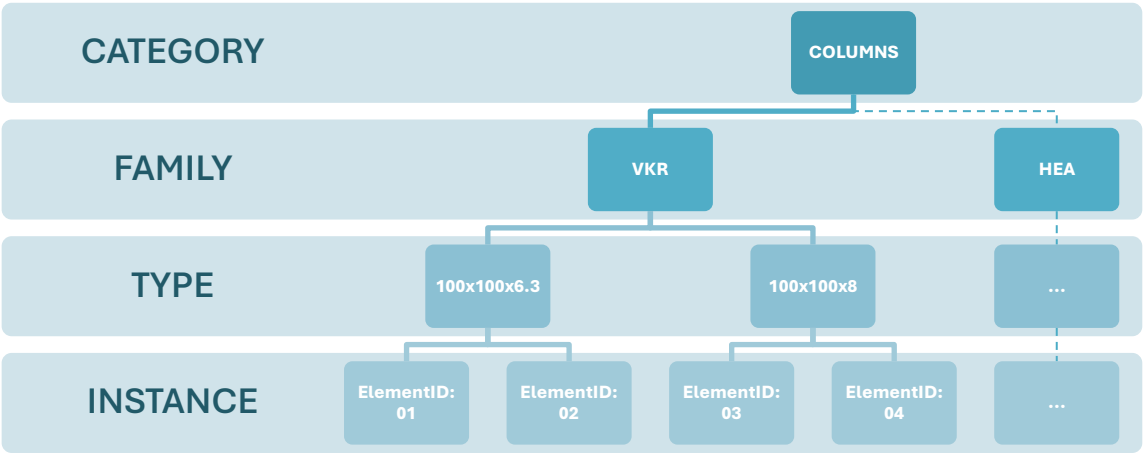
För att kunna samarbeta med andra discipliner, exempelvis arkitektur, arbetar konstruktören med arkitektens modell som underlag. Inlänkning av modeller innebär att konstruktören har möjlighet att se arkitektens modell och dess information i samma arbetsyta som K-modellen (se Figur 13.). Genom att länka in andra discipliners modeller kan konstruktören ta hänsyn till de övriga disciplinernas projektering med förutsättningen att arkitektens modell är uppdaterad och relevant i projektets skede.



Figur 13. Sisjödals K-modell (vänster) och K-modell med inlänkad A-modell i arbetsytan (höger).

2.5.2 Revits elementhierarki

Klassificeringssystemet av element i Revit kan betraktas som en hierarki och är uppdelad i olika nivåer. Klassificeringssystemet består av fyra huvudelement visas i Figur 14.



Figur 14. Informationshierarki för ett pelarelement i Revit-modellen.

2.5.2.1 Categories

Den första nivån högst upp i hierarkin är kategori. Detta är den huvudsakliga klassificeringen av typen av element. Kategorier möjliggör gruppering och organisation av olika typer av objekt i projektet. Exempel på kategorier är: dörrar, pelare och fönster. Kategorierna i Revit är fördefinierade och kan inte ändras, läggas till eller tas bort (Terrol, 2020).

2.5.2.2 Families

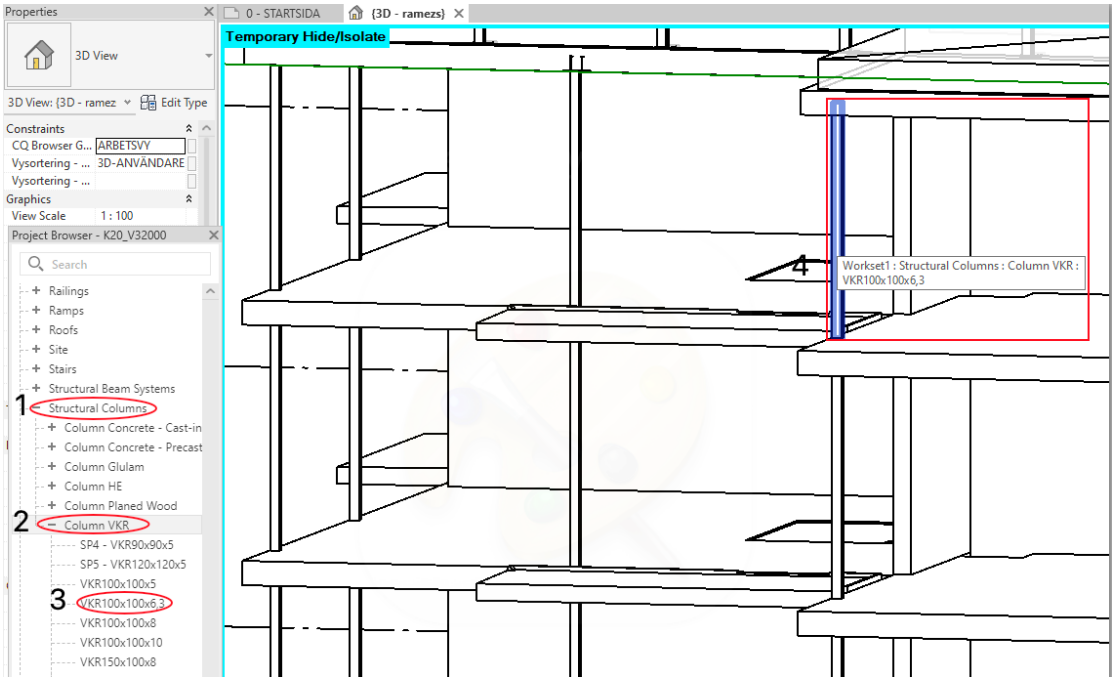
Grupperingen av familjer utgörs av kategorierna. Klassificeringen av familjer utgörs av element som delar samma egenskaper, beteende, och fysikaliska egenskaper (Terrol, 2020). Det finns tre olika typer av familjer, *system familes, loadable families*, och *in-place families* där varje element tillhör en av dessa tre familjer. *System families* är förinställda inom Revit projektet eller projektmallar. *Loadable families* är externa inladdade familjer som är skapade, definierade och modifierade i Revit Family Editor (RFA). *Inplace families* är unika komponenter och är projektunika (Autodesk, u.å.).

2.5.2.3 Types

Typer är klassificeringar inom familjen och utgörs av en specifik typ av element inom familjen där varje element med samma egenskaper. Typens egenskap är unik inom familjen (Terrol, 2020). Ett exempel på ett typbaserat element inom familjen dörrhåltagning är dimensioner. Det kan finnas olika dimensioner under familjen hålltagning, såsom 1110x2100 och 3110x2100.

2.5.2.4 Instances

Instances eller instanser är unika element inom en typ (Terrol, 2020). Klassificeringssystemet illustreras i ett praktiskt projekt i Figur 15 nedan.



Figur 15. Illustration av element klassificeringssystemet i Revit.. Exempel av elementhierarki för pelarelement i en K-modell.

2.5.3 Alternativa program

Andra program som är relevanta och delar likheter med Revit eller Dynamo:

Autodesk AutoCAD – Ursprungligen ett linjebaserat ritningsprogram inom samhällsbyggnad likt Revit. Finns viss möjlighet till att arbeta objektbaserat men har inte samma möjligheter för informationshantering som Revit (Autodesk, 2024.c).

Trimble Tekla Structures – Objektorienterat 3D-modelleringsprogram för samhällsbyggnad, skapat med fokus på strukturell design av byggprojektets BIM-modell (Tekla, 2024).

Rhinoceros 3D – Rhinoceros 3D eller Rhino används för 3D-modellering inom olika branscher däribland samhällsbyggnad. Programmet är inte anpassat för informationshantering av byggdelar såsom Revit (Rhino3D, 2024.a).

Grasshopper – Visuellt programmeringsverktyg utvecklat av Rhino som ursprungligen inte är objektbaserat som Revit men som börjar få möjlighet till att föra information på objekt. (Rhino3D, 2024.b)

2.6 Projektering inom byggbranschen

Avsnittet beskriver översiktligt de olika skeden i byggprojekt samt en mer ingående beskrivning av projektering inom byggbranschen för att ge en förståelse för processen.

2.6.1 Övergripande om byggprocessen

Byggprocessen kan delas upp i fem olika skeden (se Figur 16.) för att ta ett byggprojekt från en idé till ett avslutat projekt.

Figur 16. Illustration för byggprocessens skeden.

Byggprocessen inleds med en förstudie vars huvudsakliga syfte är att formulera övergripande mål och kartlägga förutsättningar för projektet (Boverket, 2021b). För att formulera mål och förutsättningarna krävs det att befintliga planer och underlag beaktas. Planer och underlag som är relevanta att beaktas är detaljplan, översiktsplan, miljökonsekvensbeskrivning, grön- eller naturvårdsplan, handlingsplan för grön infrastruktur och inventeringar. När underlagen är undermåliga kan egna utredningar komplettera och sakkunniga kan knytas an till projektet. Målen som ofta formuleras är exempelvis ramar för ekonomi, tekniska egenskaper, funktion, miljömål och juridiska förutsättningar. Efter förstudien är gjord bestämmer byggherren om projektet ska fortskrida.

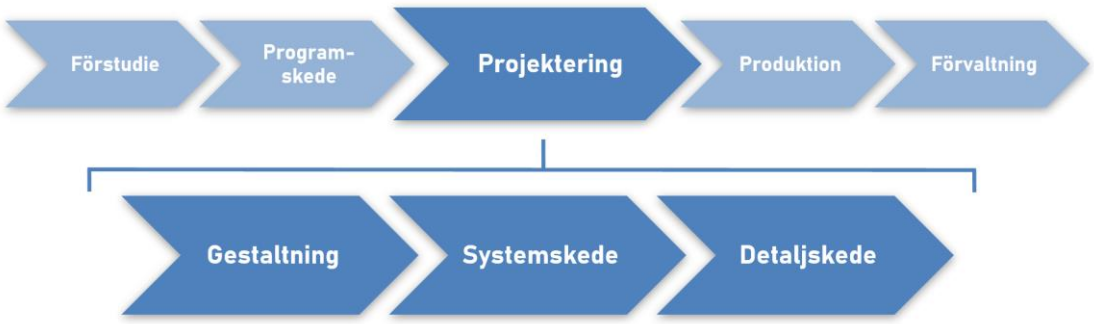
När förstudien är avslutad och byggherren valt att fortsätta projektet, kan programskedet påbörjas. I programskedet preciseras entreprenadform, krav och mål som formulerats av byggherren och samhället (Boverket, 2021a). Sammanställningen av kraven och målen utgör programhandlingar. Programhandlingar innehåller beslut om exempelvis byggnadens tekniska standard, dimensioner och grundläggande utformning (Akademiska hus, 2024.). Programhandlingarna utgör då en sammanställning av förutsättningarna vilket beslut om fortsatt projektering baseras på.

Under projekteringen gestaltas och konkretiseras byggnaden genom framtagandet av preliminära skisser och lösningar vilket sedan väljs ut av byggherren (Boverket, 2021a). Under detta skede utförs sedan en systemprojektering vars syfte är att utreda tekniska system och material som är mest optimala för byggnadens krav och mål samt sammanställa en kostnadsuppskattning (Akademiska hus, u.å.). Efter systemprojekteringen utförs detaljprojekteringen där bygghandlingar tas fram och är underlaget entreprenaden bygger utifrån. Under projekteringsskedet söks bygglov som beviljar produktionen av projektet (Boverket, 2021a)

Efter bygglovet beviljas och kommunen angett startbesked kan produktionen påbörjas och byggnaden går från handlingar till en fysisk byggnad (Boverket, 2021a.). Projektet måste byggas efter de ritningar och handlingar som angivits vid det beviljade bygglovet. Efter projektet genomgått en slutbesiktning och kommunen angett slutbesked anses produktionsskedet klart. Det sista skedet i byggprocessen är förvaltning. Under skedet överlämnas byggnaden till kunden och uppföljningar kan genomföras för att utreda utvecklingsmöjligheter (Akademiska hus, 2024.).

2.6.2 Projekteringsskedet

Projekteringen kan också delas upp i olika skeden och kan generaliseras i ett ramverk (se Figur 17). I nedanstående avsnitt beskrivs gestalningen, systemskedet och detaljskedet inom projektering



Figur 17. Projekteringens skeden i relation med byggprocessen

2.6.2.1 Gestaltning

Under gestaltningen framställs olika alternativ av byggnadens utformning som uppfyller kraven ställda i programskedet (Nordstrand, 2008). Gestaltningens syfte är att fastställa ett huvudalternativ som man vidareutvecklar. Ansvaret för gestaltningen ligger främst hos arkitekten men övriga discipliner måste också medverka. Till exempel måste konstruktörer säkerställa att bärande konstruktioner överensstämmer med förslagen från arkitekten. Då processen kräver bearbetning och omarbetning krävs det ett nära samarbete med byggherren samt mellan arkitekten och konstruktören. För att ge byggherren en mer holistisk vy av förslaget kan även 3D-modeller upprättas. Ritningar och modeller som framställs i skedet kallas förslagshandlingar och är underlaget som fortsatt projektering bygger på. Enligt Hansson m.fl. (2015) kan förslagshandlingarna summeras som en översiktlig redovisning av:

- Situationsplan i skala 1:500 eller 1:1000
- Sammanställningsritningar, det vill säga plan, snitt och vy av exempelvis fasader, skala 1:200
- Bärande stomme i snitt och planer, skala 1:200
- Installation i form av snitt, planer och scheman i skala 1:200
- Digitala visualiseringar och perspektivskisser
- Fysiska modeller
- Översiktlig beskrivning och motivering av utformning
- Uppskattning av kostnad

2.6.2.2 Systemskede

När ett huvudalternativ valts påbörjas systemskedet. Systemskedets, även kallat huvudhandlingsskedet, uppgift är att fastställa och precisera byggnadens utformning och konstruktionssystem utifrån kraven i programhandlingarna (Hansson m.fl., 2015). Systemskedet är framställningen av systemhandlingarna och dessa ligger vanligtvis till grund för hyreskontrakt, produktionshandlingar och bygglov. För att uppfylla beställarens behov brukar arkitektens rumsliga utformning vara en central roll (Hansson m.fl., 2015). Utifrån arkitektens plan och fasadritningar kan konstruktören framställa det bärande systemet för byggnaden. Hansson m.fl., (2015) beskriver hur systemhandlingarna kan bestå utav:

- Situationsplaner: redovisning av vatten och avlopp, mark, el-installationer och fiberanslutning, skala 1:500
- Sammanställningsritningar i form av: snitt, planer, vyer och fasader av hus, mark, installation i skala 1:100
- Redovisning av schakt, teknikrum och kanalisation i form av planer och snitt
- Scheman för VVS och el
- Systembeskrivning av anläggning, byggdelar och installationssystem men även typrumsbeskrivning
- Kostnadskalkyl

2.6.2.3 Detaljskede

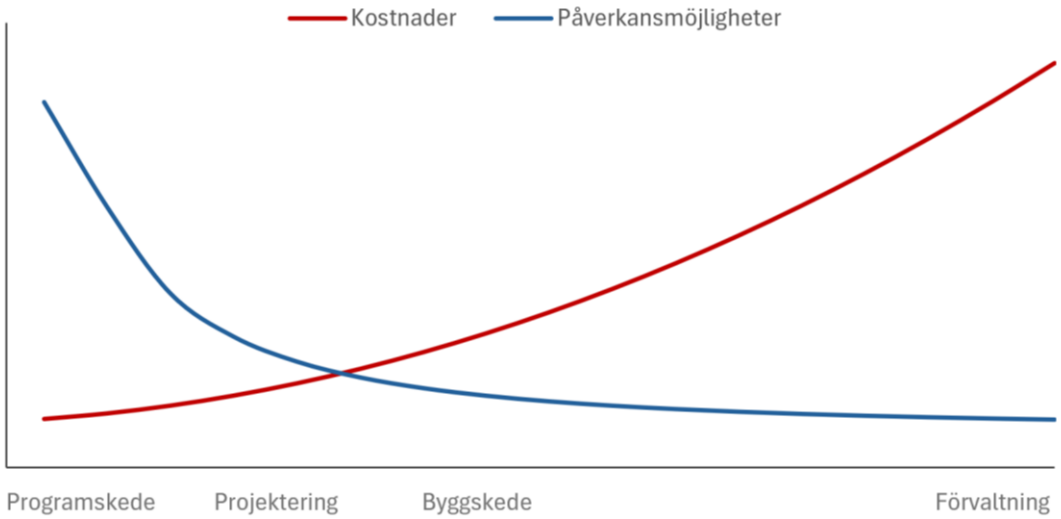
Det sista skedet i projekteringen är bygghandlingsskedet, också kallat detaljsskedet. Bygghandlingarna beskriver en detaljerad och definitiv lösning av byggprojektet (Hansson m.fl., 2015). Handlingarna består utav texthandlingar, i form av beskrivningar och textförteckningar, samt ritningar. Dessa dokument utgör en produktbestämning som byggnaden kommer produceras efter. Ritningarna i bygghandlingarna kan omfatta:

- Sammanställningsritningar: Planer, vyer, snitt av byggnaden som anger mått, läge och referenser.
- Uppställningsritningar: Uppgifter om tillverkning och montering av sammansatta byggdelar, till exempel trappor.
- Förteckningsritningar: Utförandedata och mått för typer av komponentgrupper, till exempel fönster.
- Detaljritningar: Redovisning av byggdelar med detaljerade uppgifter om till exempel utformning, montering, konstruktion.
- Samordningsritningar: samordning av arbetet från projektörer.

Beskrivningarna och förteckningarna i bygghandlingar kan innehålla:

- Teknisk beskrivning: krav och föreskrifter på kvalitet av arbete samt utförande.
- Mängder
- Tabeller som utgör rumsbeskrivningar, till exempel målning, ytskikt eller golv.
- Förteckning av lager- och standardvaror
- Förteckning av produkter som inte behöver figurredovisning
- Övrigt komplement till beskrivningar och ritningar

Eftersom projekteringen utgör en kontinuerlig specificering av byggprojektet med definitiva beslutstaganden, ändras även påverkansmöjligheterna för ändring av byggnadens utformning (Hansson m.fl., 2015). Ändringar i projekteringen blir därav mer kostsamma vid senare skeden och fel bör åtgärdas tidigt. Minskningen av påverkansmöjligheter och ökningen av kostnader för ändringar kan även appliceras på hela byggprocessen (se Figur 18.). Då projekteringen är ett av de tidigare skedena är vikten av en god projektering med genomtänkta beslut stor.



Figur 18. Generell beskrivning av hur påverkansmöjligheter och kostnader varierar över byggprojektets gång. Vågrät axel visar skeden under byggprojektets tid, lodrät axel visar magnitud av kostnad eller påverkansmöjligheter.

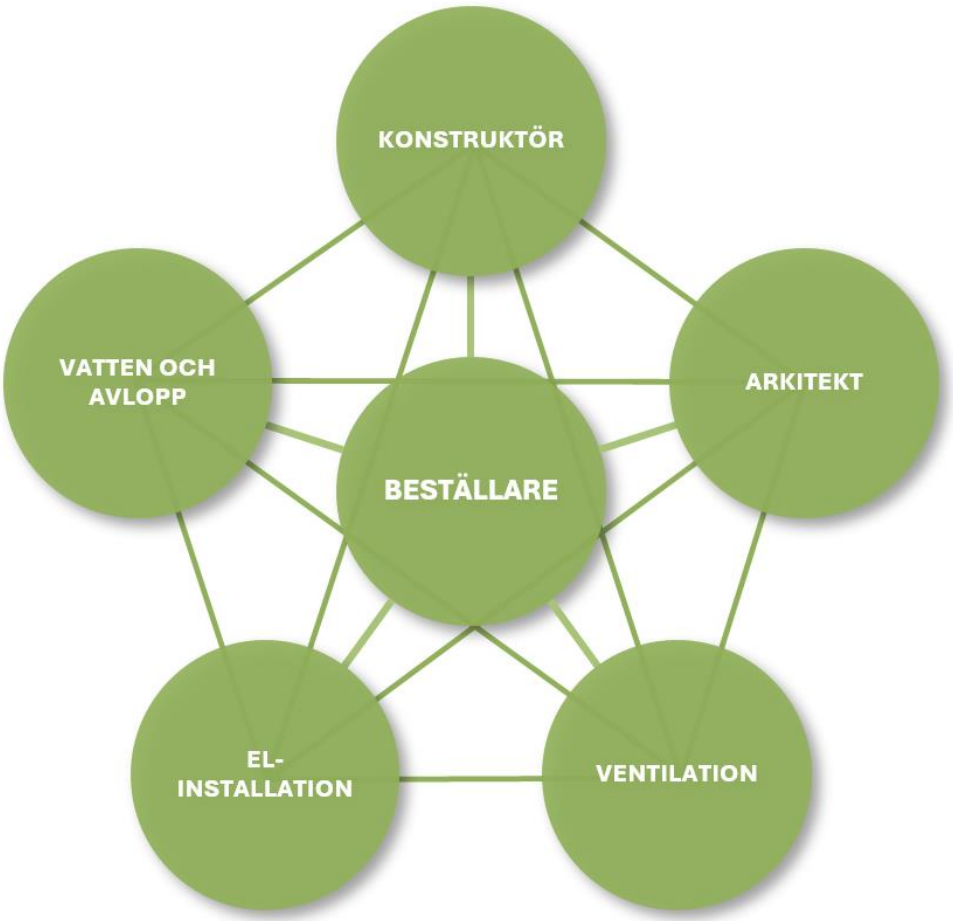
2.6.3 Projekteringsprocessen och processflöden hos VBK

För att undersöka VBK:s nuvarande projekteringsprocess samt förekommande processflöden, genomfördes en intervju med Rasmus Öberg. Rasmus arbetar som redovisningsingenjör sedan 2015 med en senior roll inom redovisning. Hans uppgift är att framta konstruktionsritningar och BIM-modeller, vanligtvis i Revit, samt driva intern utveckling inom redovisning och BIM på VBK.

Företaget verkar främst i systemhandlings- och detaljskedet av projekt men seniora ingenjörer kan även medverka som sakkunniga i tidigare skeden, vilket kan vara en ingång för nya uppdrag (Öberg, 2024). I systemhandlingsskedet framställer företaget byggnaders bärande system det vill säga dess material, utformning, dimensioner, stabilisering, grundläggning och principiell utformning av klimatskalet. I detaljsskedet utformas detaljlösningar för de stomsystem och klimatskal som valts under systemhandlingsskedet. Exempel på detaljlösningar är armering i platsgjuten betong eller infästningsdetaljer för stål och trä. Företaget arbetar mycket med både betong, stål och trä.

Arbetsflödena och processerna i projekteringen styrs mycket av projektets typ beskriver Öberg (2024). Under husbyggnation måste ett flertal olika discipliner samverka och det finns många beroenden konstruktören måste ta hänsyn till (se Figur 19) medan projektering av betongfundament för ett vindkraftverk beror i princip enbart på geoteknik och vindkraftverket i sig. Projekteringen påverkas även av entreprenadformen där detaljeringsnivån på handlingar styrs beroende på konstruktörens uppdrag. I en totalentreprenad projekterar VBK ofta fram ett förfrågningsunderlag vars syfte är att specificera funktioner som beställare önskar. Därefter handlas en entreprenör upp som ska lösa dessa funktioner vilket ibland leder till att VBK anlitas för detaljprojekteringen. I utförandeentreprenader projekterar företaget i de tidigare skedena ända fram till full detaljeringsgrad i

detaljskedet. Dessa handlingar upphandlas sedan av entreprenören som utför arbetet.



Figur 19. Figur som principiellt beskriver beroenden mellan olika discipliner under projektering.

Beroendena ger upphov till processflöden där informationsutbyte sker mellan disciplinerna och utbytet sker på olika sätt beroende på flödesprocess (Öberg, 2024). För att samverka med de andra disciplinerna skapas vanligtvis en portal av beställaren för utbyte av filer. Filerna kan vara modeller från revit eller i IFC-format, DWG:er eller PDF:er. En annan typ av processflöden är möten, vars medium är mejl-korrespondens, telefonsamtal eller Teams. För att strukturera upp flödesprocesser kan rutiner tillämpas för att säkerställa att informationsutbytet sker frekvent och att parterna arbetar med uppdaterad information. Graden av satta rutiner och uppdateringar, ökar med storleken på projektet. Öberg (2024) beskriver även hur molnbaserade tjänster som direkt samverkar med modelleringsprogrammen börjar tillämpas för att strukturera modellhanteringen mellan parter genom exempelvis tillämpning av frågor-och-svarfunktion. Exempel på sådana portaler är Dalux, Byggnet och Autodesk Construction Cloud.

Öberg (2024) beskriver hur informationsflödet externt i modeller med övriga discipliner, exempelvis arkitektur, är ett förbättringsområde. Då ändringar i modellen sker löpande och versionshistorik saknas, skapas en problematik för konstruktören då denne behöver okulärt granska vad som ändrats i modellen.

Ofta tänds modellen upp tillsammans med samlänkade modeller för att identifiera avvikelser. Tillvägagångssättet är dock riskabelt då konstruktören riskerar att missa ändringar. Öberg (2024) poängterar att denna typ av processflöde förekommer även internt men kommunikationskanalen konstruktörer emellan på företaget har inte samma barriär som kommunikationen mellan projektörer från olika företag. Det finns verktyg för att bemöta problemet, bland annat genom granskning av objekt i kontrollvyer, användning av olika program, exempelvis BIMvision (en IFC-läsare med versionshistorik), molnbaserade tjänster eller genom egen utveckling av skript i Dynamo.

Processflöden varierar även beroende på underlag i projekteringen (Öberg, 2024). Underlag från arkitekt, konstruktör eller från installation förekommer ofta i modellbaserad form med varierad grad av information lagrad i modellen.
Andra discipliner såsom landskap och mark jobbar i 2D-ritningar med limiterade metadata. Det finns även discipliner som levererar enbart textbaserade underlag såsom hiss-leverantörer.

De interna flödena på företaget omfattas generellt av tre parter, uppdragsledning, beräkning och redovisning. Utbytet mellan redovisning och beräkning består av att kommunicera beräkningsresultat, såsom geometrier, material eller dimensioner till redovisning som i sin tur kommunicerar frågeställningar tillbaka till beräkningsansvarig. Flödets medium kan bestå av handskisser, PDF:er, modeller eller ett till och med ett integrerat flöde genom ett skript mellan parterna. Det interna flödet med uppdragsledning består främst av att styra uppdraget mot budget, hålla etablerad tidsplan samt fatta beslut utifrån extern information.

2.6.4 Nuvarande användning av Dynamo i projekteringen

Det existerar flera automatiserade flöden inom projekteringen som har skapats och används i varierande grad, vilket beror på vilka individer som är engagerade i projektet. Utvecklingen av Dynamo på VBK är främst en individuell process där en person kan upptäcka en lösning och sedan delar den med andra kollegor genom olika forum. Det är vanligt att skapa skript för att lösa specifika problem som uppstår under projektets gång. I framtida projekt kan man använda sig av det befintliga skriptet som en grund och sedan vidareutveckla det eller göra anpassningar enligt behov.

Ett praktiskt exempel på implementering av Dynamo för att förbättra och automatisera flöden på VBK är betongfundament för vindkraftverk där VBK har investerat i att utveckla ett flöde där betonggeometri och armering skapas utifrån utdata från beräkning. Anledningen till att VBK skapade skriptet är på grund av att liknande projekt är återkommande och det blir då oerhört effektivt.

I andra projekt kan underlagen ha större variation vilket medför att man behöver anpassa skript. Detta skapar ett mindre incitament för att göra ett bättre flöde med Dynamo. Däremot i mindre moment som är återkommande, är ett robust skript av värde då det ökar effektiviteten och ger möjlighet till att utvärdera fler alternativ. Det skapar ett värde för kunden då bättre lösningar kan hittas och ge ett bättre

beslutsunderlag. Att använda automatisering genom skript i projekteringen skapar även ett värde för projektören då mindre tid behöver läggas på repetitiva arbetsuppgifter och mer tid ägnas åt mer givande uppgifter.

3 Fallstudie

För att undersöka huruvida Dynamo kan användas för att optimera processflöden mellan konstruktörer och arkitekter under projekteringen utfördes en fallstudie där vanligt förekommande typer av uppgifter behandlas med hjälp av dynamoskript. Scenarion fallen behandlar togs fram av VBK genom kontinuerlig dialog.

3.1 Fall 1: Geometrisk överföring från A till K

Fall 1 omfattar överföringen av geometrier från A till K. Det konkreta momentet som utförs i fallet är identifiering av dörrar (A) som sammanfaller med bärande väggar (K). Därefter genomförs håltagningar i bärande väggar (K). För nuvarande utförs processen manuellt där konstruktören tänder upp den länkande A-modellen och navigerar i modellen för att identifiera dörrar som överlappar med bärande väggar (K). Genom positioneringen av dörrar (A) skapar konstruktören håltagningar i bärande väggar manuellt för varje dörr. Håltagningarna är familjetyper med typbaserade parametrar för mått. Därav måste konstruktören skapa en ny familjetyp för varje typ av dörrkarmsmått i A-modellen.

3.1.1 Modellen och dess information

Vid analys av fall 1 var modellvalet Sisjödal som är en lämplig modell baserat på innehållet av modellen då K-modellen har bärande väggar som kräver dörrhåltagningar och A-modellen har olika dörrobjekt. Den framtagna modellen är VBK:s egen modell. Sisjödal-projektet utfördes av Tuve Bygg och utformades av Werner Arkitekter med hjälp av VBK som konstruktörer. Beställare av projektet var Framtiden och hyresvärden är Familjebostäder. Handlingar och ritningar för projektet framställdes år 2022 och omfattade en nybyggnation av bostäder i området.

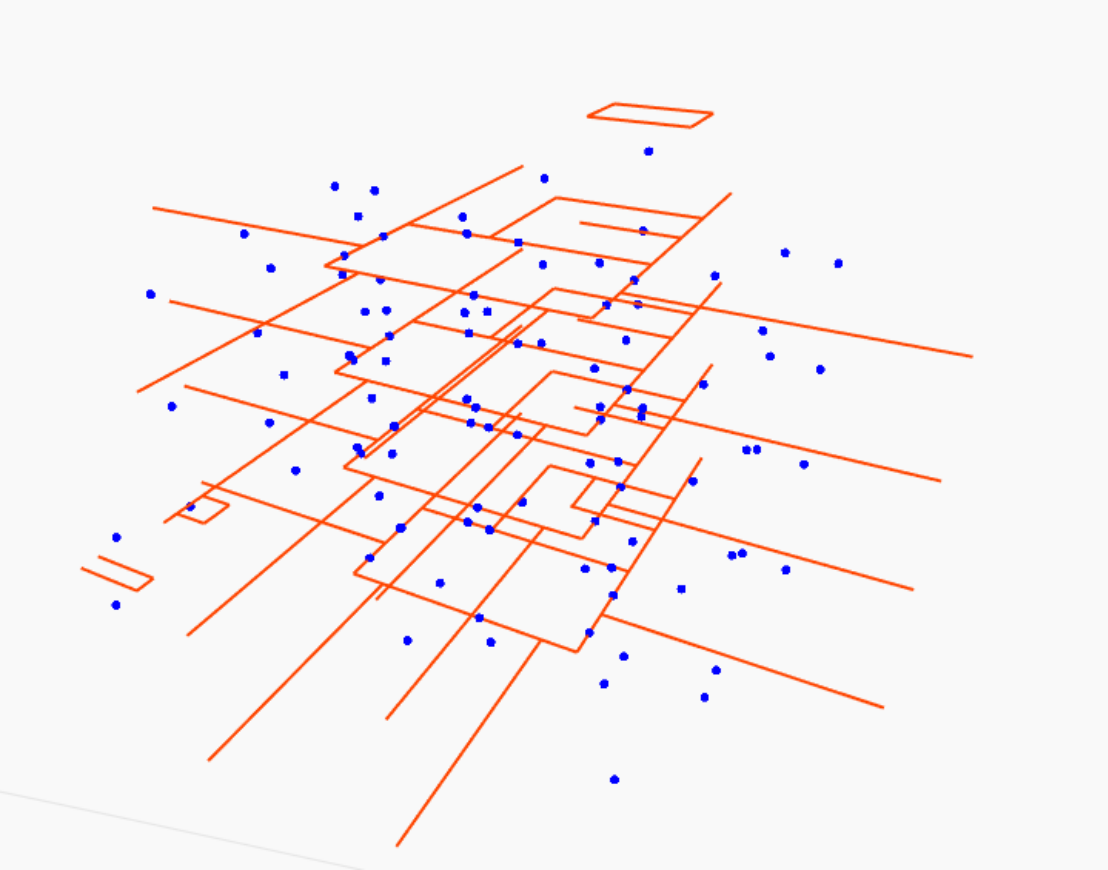
3.1.2 Krav på skript

Dörröppningarna som skapas tilldelas dimensioner i familjetypen i stället för instansbaserade mått då det underlättar hanteringen av objekteten vid exempelvis mängdning och kan återanvändas i projektet.

3.1.3 Beskrivning av Dynamoskriptets flöde

Filtrering av väggelement

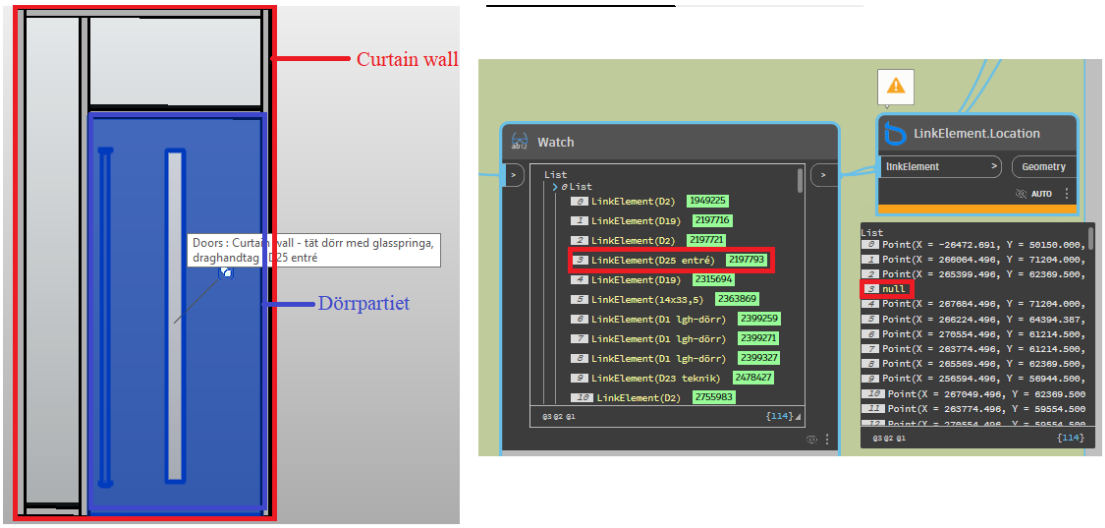
Scriptet inleds med att filtrera alla relevanta element. Först filteraras alla ytterväggar bärande (YVB) och innerväggar bärande (IVB) i den lokala K-modellen. Fortsättningsvis inhämtas alla dörrar från den inlänkade A-modellen. Detta illusteraras i Figur 18 nedan där de röda linjerna symboliserar alla väggar från K modellen och de blå punkterna symboliserar alla dörrar från A-modellen.



Figur 20. Illustration av vägg och dörrelement i Dynamo

3.1.3.1 Curtain walls

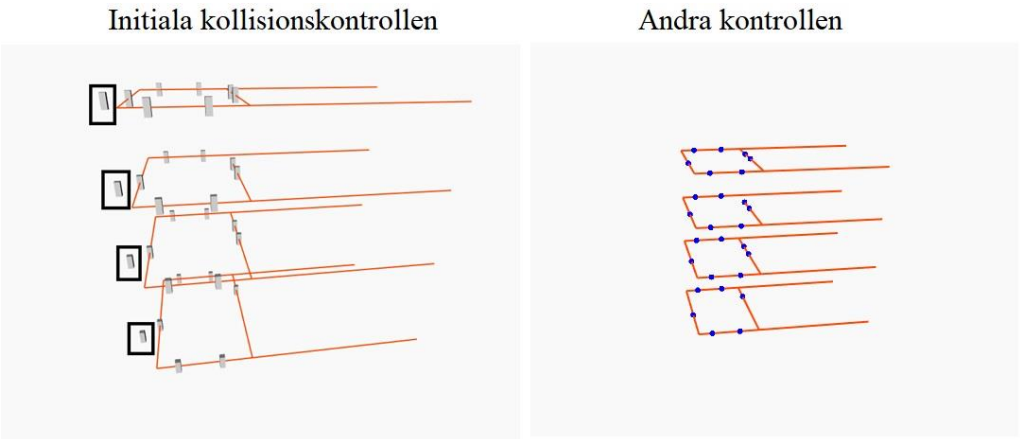
”Curtain walls” är väggelement i Revit som består av glas eller andra material. Det finns möjlighet att placera dörrar i glaspartiet. Dock kan dessa dörrelement i ”curtain walls” inte upptäckas med Dynamos positionsnoder som exempelvis ”LinkElement.Location”. I Figur 21. illusteras en dörr som sitter i ett glasparti samt att dörrens position i Dynamo är ”null”.



Figur 21. Dörr i Curtain Wall samt samma dörr i Dynamolista.

3.1.3.2 Filtrering av dörrar

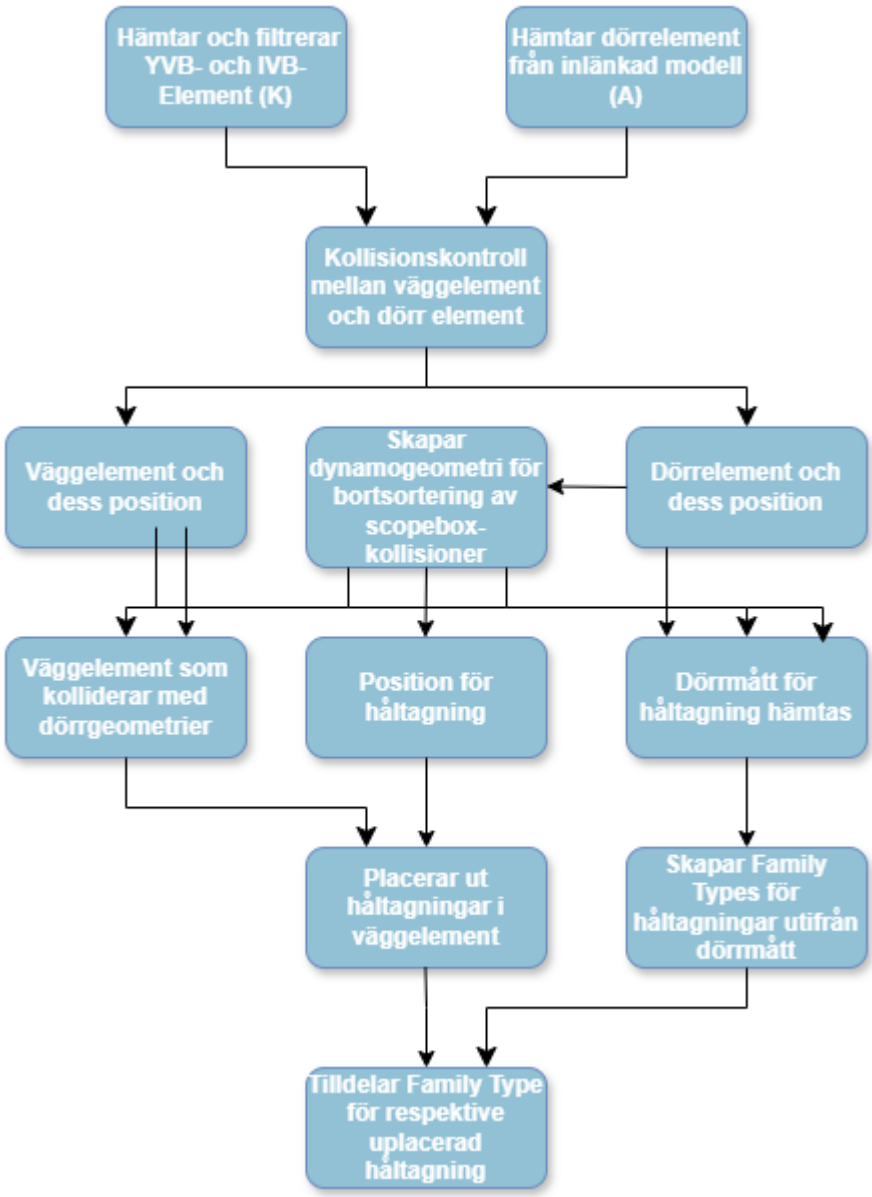
Nästa steg är att identifiera alla relevanta dörrar (A) som överlappar med alla bärande väggar (K) och filtrera de i listor. Genom två kollisionskontroll-noder kan relevanta element identifieras. Anledningen till att två kollisionskontroller utförs är på grund av att den initiala kollisionskontrollen är mellan elementen det vill säga dörrar (A) och väggar (K). Vid kollisionskontrollen upptäcktes kollisioner mellan dörr (A) och vägg (K) som inte kolliderade. Orsaken var att synligheten av dörr gränsen det vill säga ”boundingboxen” ingick i kollisionskontrollen. För att exkludera ”boundingboxen” gjordes dörrgeometrierna om till dynamo kubgeometrier med styrbara dimensioner. Figur 22. belyser de inringade dörrar som upptäcks vid den första kontrollen som inte överlappade med bärande väggar (K). Den andra kontrollen som utförs är mellan dynamogeometrierna och väggarna. Då fångas de relevanta element som kolliderar och position för håltagningen blir känt.



Figur 22. Visar de två kollisionskontrollerna som utförs mellan dörr (A) och vägg(K). De inringade kuberna i den initiala kontrollen är de extra dörrar som fångas. De orangea linjerna symboliserar väggar (K). De blå punkterna i den andra kontrollen är de relevanta dörrpositonern.

3.1.3.3 Position och placering av typbaserade håltagningar

När läge och relevant dörrelement är känt blir följande steg att filterara dörrkarmsmått från relevant dörr (A). Slutligen skapas typbaserade håltagningar utifrån dörrdimensioner och placeras ut på korresponderande väggelement (K).



Figur 23. Övergripande flödesschema för dynamoskriptet.

Paketen som användes i skriptet är är Archi-lab, BimorphNodes, Clockwork och Rhythm eftersom nödvändiga noder inte finns tillgängliga i Dynamos bibliotek.

3.2 Fall 2: Överföring av metadata från A till K

Att överföra parametrar från A-modellen kan berika K-modellen med relationer mellan objekten. Exempel på sådana relationer kan vara rumsrelation, brand samt pelar och vägg relationer. Det kan vara till intresse för konstruktören att få en uppfattning om i vilket rum en pelare står i. Pelar-vägg relationer kan ge en uppfattning om en pelare får plats i en vägg. I fall 2 undersöktes hur väggars BSAB-kod från A-modellen kan överföras till K-modellens överlappande pelare. Informationen som överförs är av texttyp och tilldelas på pelares nyskapade eller befintliga parametrar. Nuvarande tillvägagångsätt är att manuellt skriva in relationer mellan väggelement (A) och pelarelement (K) som textparameter.

3.2.1 BSAB-koder

BSAB-systemet är ett tabulerande klassificeringssystem inom byggbranschen för att kunna tala gemensamt språk att uttrycka information i modeller (Svensk (Byggtjänst, u.å). Några praktiska exempel på framtagna tabeller är byggdelar, byggdelstyper, produktionsresultat, resurser. I Figur 24 nedan visas ett utdrag av BSAB 96 koder som är en av de framtagna koderna för byggdelar och byggdelstyper.

4. **42.D** - Öppningskompletteringar i ytterväggg
    1. **42.DB** - Fönster
    2. **42.DC** - Fönsterpartier, klimatskiljande
    3. **42.DD** - Väggpartier, klimatskiljande
    4. **42.DE** - Ytterdörrar
    5. **42.DF** - Portar
5. **42.E** - Ytterväggskompletteringar
6. **42.Z** - Övriga klimatskiljande delar och kompletteringar i ytterväggg

4. **43** - INRE RUMSBILDANDE BYGGDELAR
    1. **43.A** - Sammansatta inre rumsbildande byggdelar
    2. **43.B** - Kompletterande väggkonstruktioner
    3. **43.C** - Innerväggar (ej stominnerväggar) och öppningskompletteringar
        1. **43.CA** - Innerväggar, sammansatta
        2. **43.CB** - Innerväggar (ej stominnerväggar)
            1. **43.CB/20** - Innerväggar (ej stominnerväggar) - murverk, puts
            2. **43.CB/35** - Innerväggar (ej stominnerväggar) - element av trä eller träbaserat material
            3. **43.CB/37** - Innerväggar (ej stominnerväggar) - element av skivor och träregelverk eller träbaserade regelverk
            4. **43.CB/40** - Innerväggar (ej stominnerväggar) - skivor och regelverk
            5. **43.CB/41** - Innerväggar (ej stominnerväggar) - skivor och stålregelverk
            6. **43.CB/42** - Innerväggar (ej stominnerväggar) - skivor och träregelverk eller träbaserade regelverk
        3. **43.CC** - Öppningskompletteringar i innervägg
            1. **43.CCD** - Väggpartier
            2. **43.CCE** - Innerdörrar
            3. **43.CCF** - Portar
    4. **43.D** - Bjälklagsöverbyggnader och öppningskompletteringar
    5. **43.E** - Innertak
        1. **43.E/22** - Innertak - puts
        2. **43.E/40** - Innertak - skivor och regelverk
    6. **43.Z** - Övriga inre rumsbildande byggdelar
5. **44** - INVÄNDIGA YTSKIKT
6. **45** - HUSKOMPLETTERINGAR
7. **46** - RUMSKOMPLETTERINGAR
8. **49** - ÖVRIGA RUMSBILDANDE BYGGDELAR, HUSKOMPLETTERINGAR, YTSKIKT OCH RUMSKOMPLETTERINGAR

6. **5** - VA-, VVS-, KYL- OCH PROCESSMEDIESYSTEM
7. **6** - EL- OCH TELESYSTEM
8. **7** - TRANSPORTSYSTEM M M
9. **8** - STYR- OCH ÖVERVAKNINGSSYSTEM
10. **9** - ÖVRIGA BYGGDELAR OCH INSTALLATIONSSYSTEM

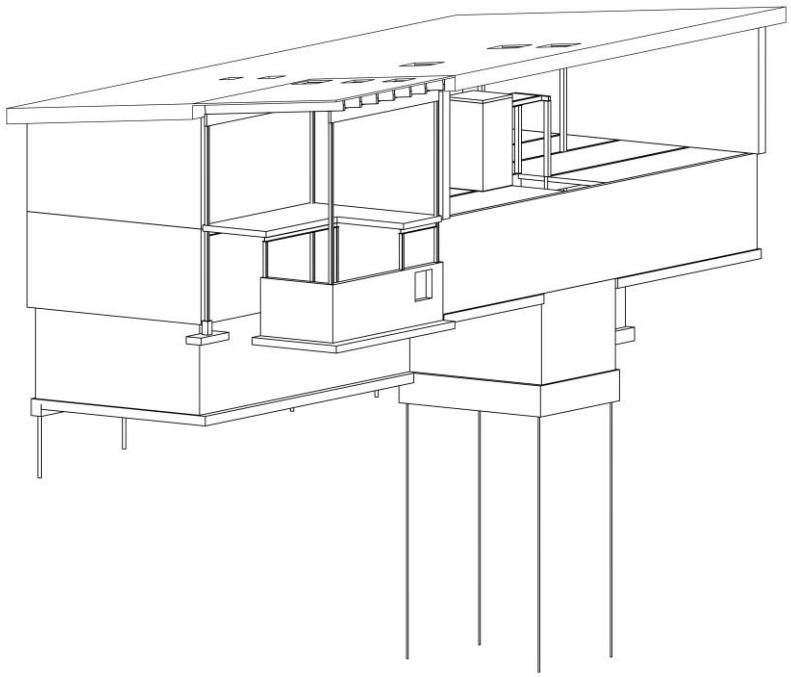
Figur 24. Ett utdrag på BSAB 96 koder

3.2.2 Parametertyper i Revit

Ett revitelement kan ha fyra olika typer av ändringsbara parametrar; Project-, Family-, Shared-, och Global Parameter (Autodesk, 2024.b). Project Parameter är projektspecifika parametrar som tilldelas till kategorier. Global Parameters är också projektspecifika parametrar men tilldelas inte till någon kategori. Family Parameter är parametrar som är kopplade till familjedokumentet och används för till exempel mått. Shared Parameters är parametrar som förvaras i ett separat dokument istället för projektfilen och kan därav inte ändras. Shared parametrar har då samma villkor oavsett projekt och kan därför användas till tags och schedules. För fall 2 var det Shared Parameters som var lämpligast att använda då man kan använda den överförda informationen i tags och schedules. Tags i Revit är etiketter som används för att identifiera och visa information om byggnadselement direkt i ritningar och modeller. Schedules ger en tabellbaserad översikt av element och deras egenskaper.

3.2.3 Modellen och dess information

Vid analys av fall 2 var modellvalet en nybyggnation av en förskola belägen i östra Göteborg. VBK:s uppdrag var att delta i programhandlingen och ta fram en systemhandling som skulle gå ut som förfrågningsunderlag för totalentreprenad. K-modellen har pelarobjekt utplacerade medan A-modellen har väggar med BSAB-kod.



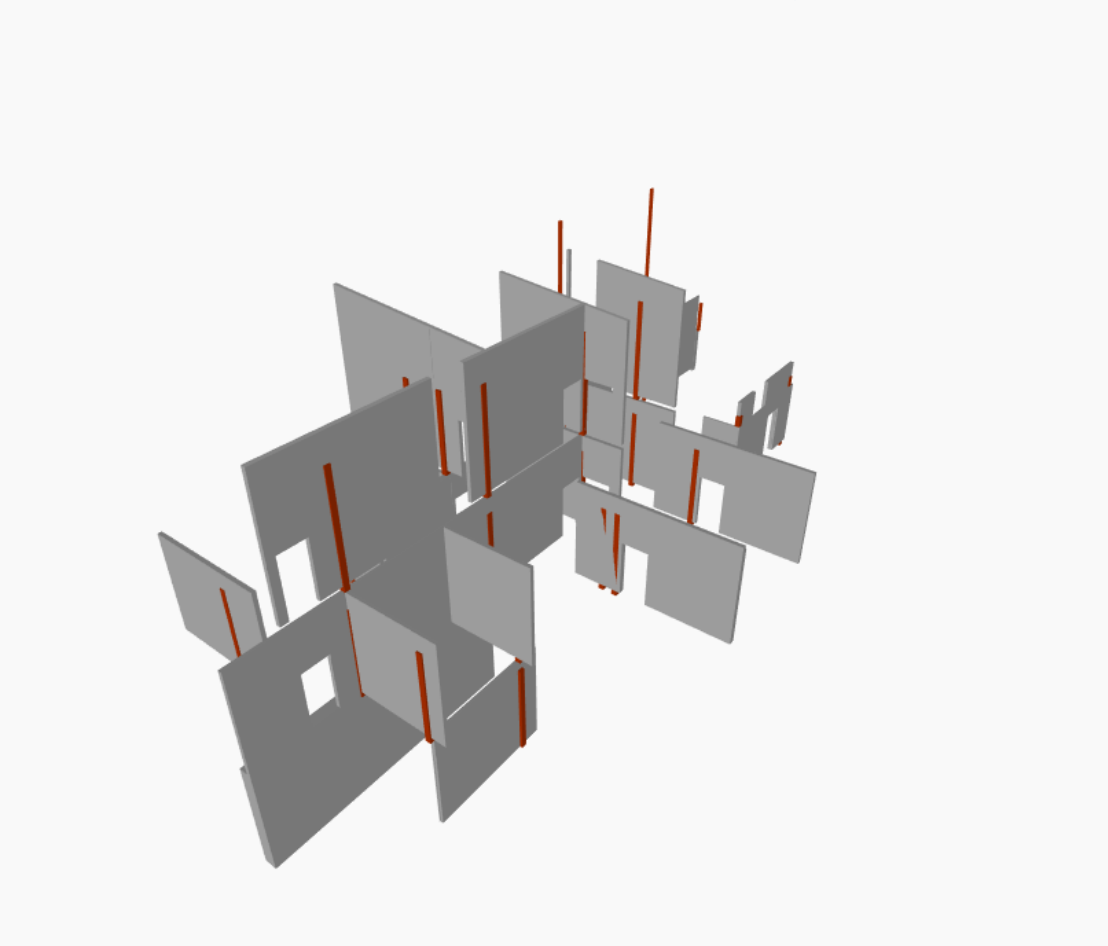
Figur 25. K-modellen av förskolan.

3.2.4 Krav på skript

Praxis är att överförd metadata bör vara under en shared parameter för att kunna användas i schedules och tags.

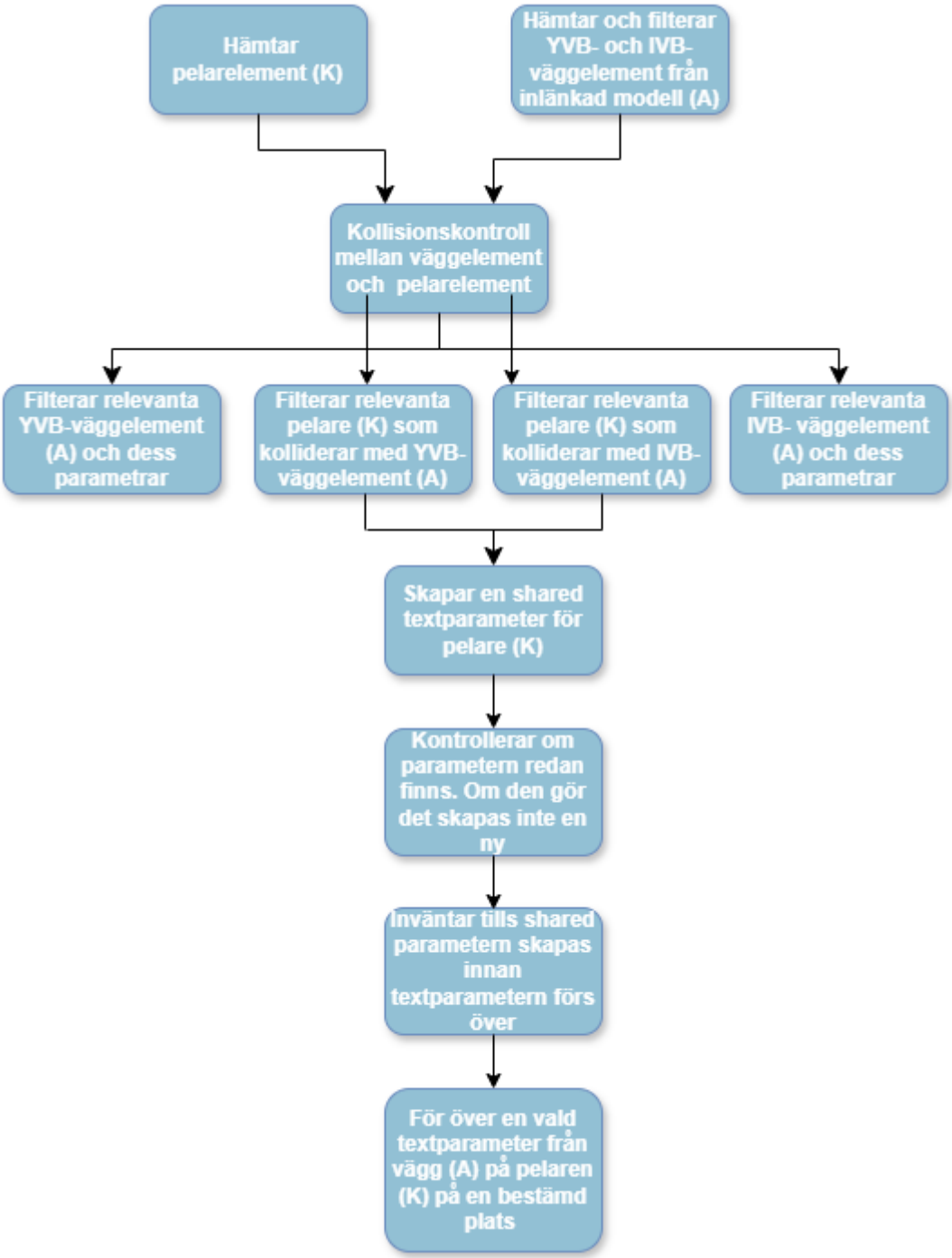
3.2.5 Beskrivning av Dynamoskriptets flöde

Skripten inleds med att hämta alla vägg-element från den inlänkade A-modellen och sedan filtrera innerväggar och ytterväggar i separata listor. I samma skede hämtas alla pelare från K-modellen. Därefter utförs en kollisionskontroll mellan pelare och väggar för att upptäcka om pelare (K) sitter i vägg-element (A). Kontrollen skapar en lista med pelare och en lista med väggar som korresponderar alla pelare som sitter i väggar. Hur kollisionen ser ut visuellt i Dynamo visas nedan i Figur 26.



Figur 26. Illustration i Dynamo av pelare (K) i väggar (A)

Fortsättningsvis ur vägg-listan (A) med innehållande pelare (K) filtreras önskad metadata i detta fall BSAB-koden.
En kontroll utförs innan överföringen sker för att säkerställa att den nya parametern inte redan finns på pelaren. Om parametern redan finns, överskrivs den med nya metadata. Om parametern inte finns, skapas en ny shared parameter med önskad typ och gruppering. Resulterande skript visualiseras i ett konceptuellt flödesdiagram (se Figur 27)



Figur 27. Flödesdiagram för konceptuellt flöde av dynamoskriptet för överföring av BSAB-koder.

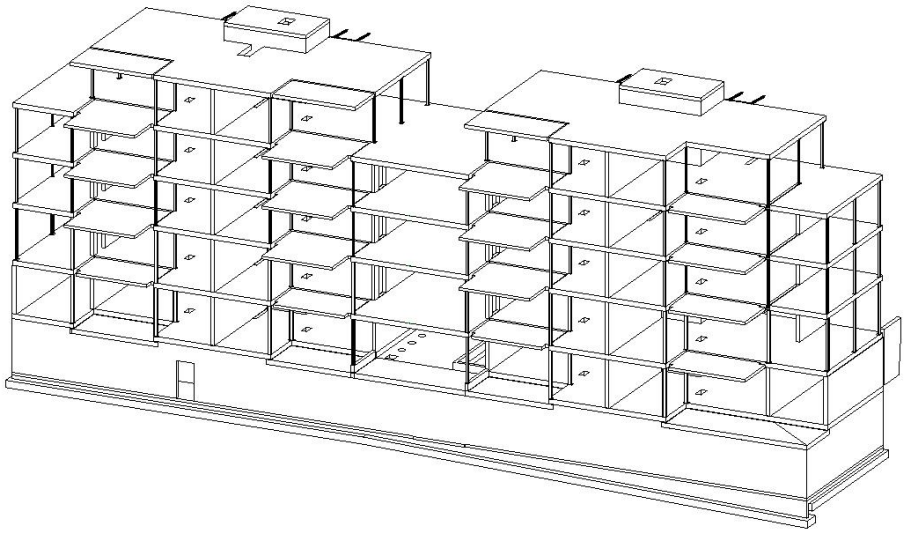
Paketen som användes i skriptet är Archi-lab, BimorphNodes, Orchid och Rhythm eftersom nödvändiga noder inte finns tillgängliga i Dynamos bibliotek hittades i paketen.

3.3 Fall 3: Skapa K-Objekt utifrån A-geometrins villkor och metadata

Fall 3 behandlade skapandet av K-objekt utifrån A-modellens geometriska villkor och metadata. Det konkreta exemplet skriptet skapades för är skapandet av ursparningar (K) i bjälklag, utifrån badrums-rumsobjekt (A) samt metadataöverföring av rumsnummer (A) för respektive ursparning. Nuvarande tillvägagångsätt är att manuellt skriva in rumsrelation som textparameter samt manuellt placera ursparningar för respektive rum och skapa olika familjetyper för de olika mått ursparningarna består av. Målet med skriptet är att skapa ursparningar utifrån rumsobjektens placering, bredd och längd samt överföra rumsnumret eller rumstypen som en inlagd textparameter på ursparningsobjektet.

3.3.1 Modellen och dess information

Modellen som fall 3 utgår ett flerbostadshus med sex våningsplan beläget i östra Göteborg (se Figur 28.). I K-modellen finns en ursparningsfamilj redan inladdad i modellen som laddades ned och anpassades för att kunna bifogas med skriptet. I den inlänkade A-modellen finns rumsobjekt modellerade. Rumsobjekten har namn efter rumstyp, exempelvis ”BADRUM” satta i ”Name”-fältet och numreringar i form av textparametrar satta under ”Number”-fältet. Att A-modellen har rumsobjekt och metadata att överföra är en förutsättning för att skriptet ska vara relevant.



Figur 28. K-modell av flerbostadshuset fall 3 utgår från.

3.3.2 Krav på skript

Skriptet skapades utifrån ett flertal krav. Ett av kraven är att överförd metadata bör vara placeras under en shared parameter för att kunna användas i schedules och tags. Dessutom bör objekten som skapas tilldelas mått i familjetypen i stället för på instansnivå, då det underlättar mängdning.

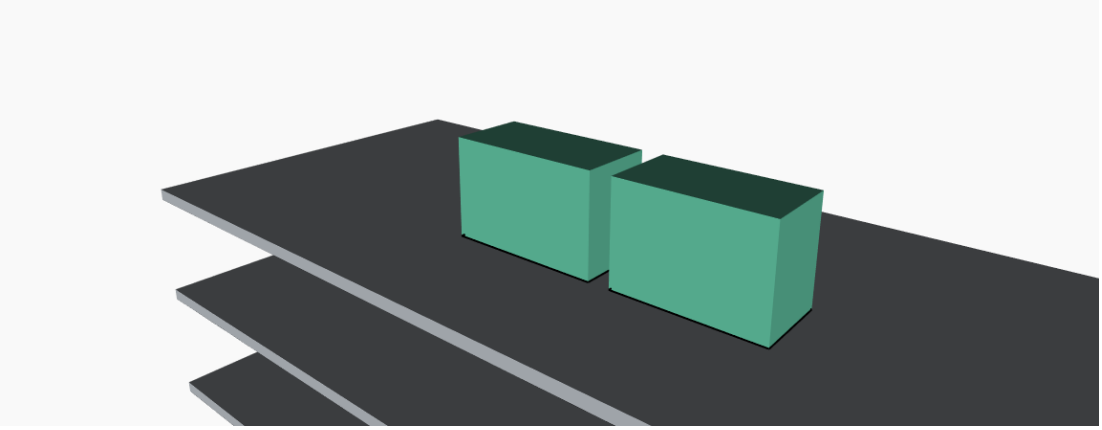
3.3.3 Utplacering av objekt

I fallet undersöktes två tillvägagångssätt för att placera ut ursparningsobjektet, yt-baserad (Face based) och golv-baserad (Floor based) objektplacering (Autodesk, 2015). Golvbaserad utplacering innebär att objektet använder bjälklaget som värd. Yt-baserad utplacering av ett objekt innebär att objektet använder ett elements ytor som referensplan och tillhör ytan av elementet. I VBK:s egna generella golvbaserade ursparningsobjekt, placeras alltid ursparningen på toppen utav bjälklaget.

I fallet var det inget krav att använda ett specifikt tillvägagångssätt för utplacering. Därav programmerades två olika skript för att sedan kunna undersöka för- och nackdelar med yt- och golvbaserad utplacering.

3.3.4 Beskrivning av Dynamoskriptets flöde

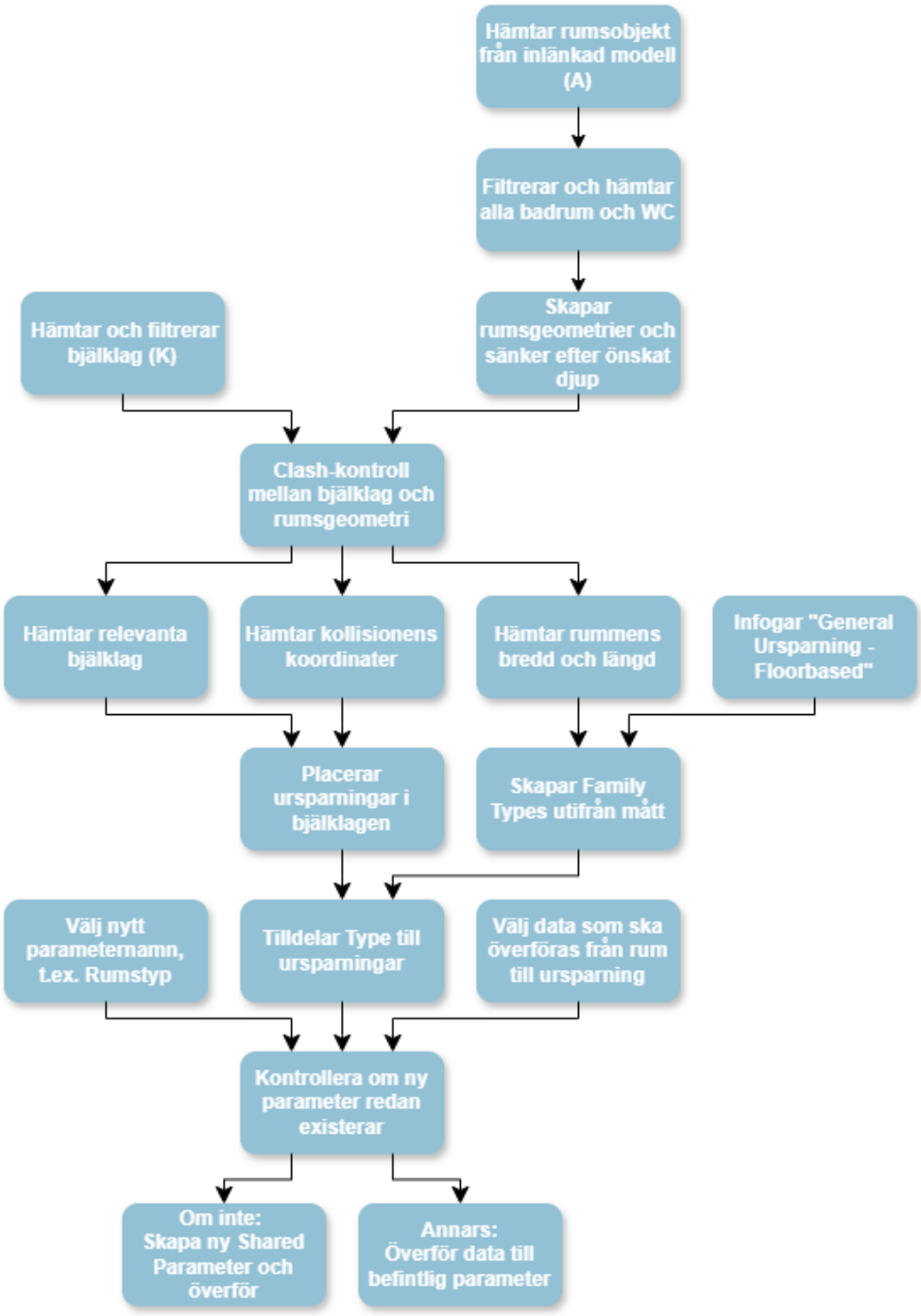
De två skript som skapades har samma struktur bortsett från hur ursparingsobjekten placeras. Skripten börjar med att hämta alla rumsobjekt från den inlänkade A-modellen och sedan filtrerar ut de rum som eftersöks, i detta fall 29 badrum/WC. I samma skede hämtas alla bjälklag från K-modellen. Därefter skapas dynamogeometrier för varje rum och sänks i Z-led med önskat djup för att säkerställa att geometrin kolliderar med relevant bjälklag (se Figur 29). Efter att rumsgeometrin sänks, utförs en kollisionskontroll mellan rumsgeometrin och bjälklag. Kontrollen skapar en lista med rumsgeometrier och en lista med bjälklag som korresponderar med listan av rumsgeometrier. Genom kontrollen skapas en volym utifrån differens mellan bjälklag- och rumsgeometrin. Ur denna volym kan max- och min-koordinater samt rummens bredd och längd hämtas.



Figur 29. Rumsobjekt som omvandlats till dynamogeometrier (Cyan) för ett bjälklag.

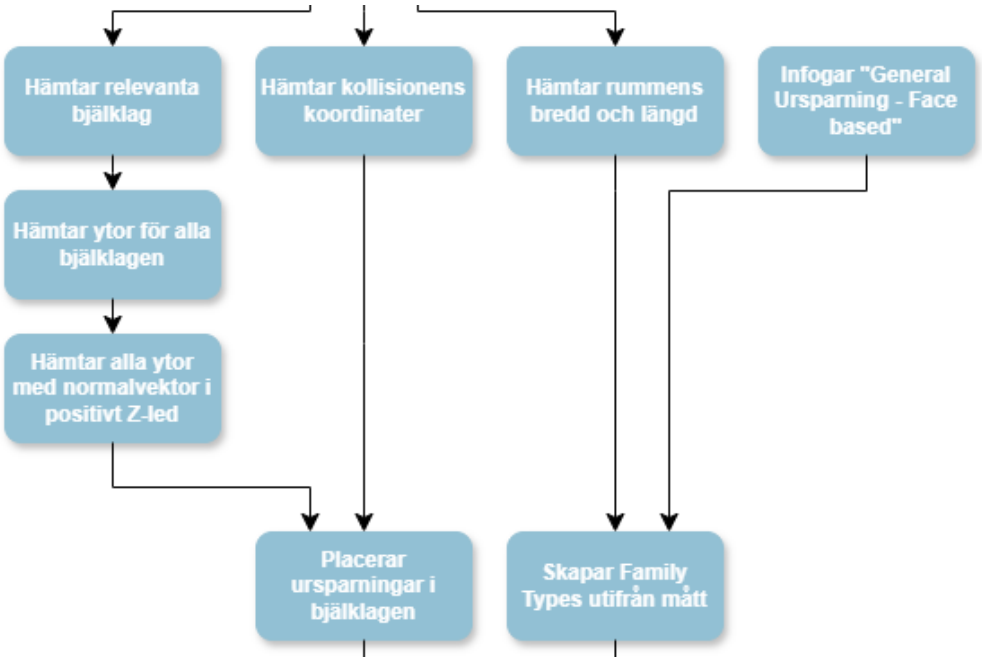
Efter att respektive bjälklag, rumsdimensioner och koordinater för ursparningen hämtats, placeras ursparningsobjekt ut på bjälklaget med den generella ursparningsfamiljen. Familjetypen dupliceras utifrån dynamogeometrins mått och de duplicerade familjetyperna tilldelas bredd, längd, önskat djup och namn

med mått. Till sist hämtas önskad textparameter från rumsobjektet i A-modellen samt vad den nya parametern ska heta på ursparningsobjektet. En kontroll utförs innan överföringen sker för att säkerställa att den nya parametern inte redan finns på ursparningen. Om parametern redan finns, överskrivs den med nya metadata. Om parametern inte finns, skapas en ny shared parameter med önskad typ och gruppering. Resulterande skript visualiseras i ett konceptuellt flödesdiagram (se Figur 30.).



Figur 30. Flödesdiagram för konceptuellt flöde av dynamoskriptet för floor-based utplacering.

Ett till skript skapades för face-based ursparningar. Flödet är identiskt med floor-basedskriptet förutom ett annat tillvägagångssätt i utplaceringsmomentet. I stället för att enbart hämta och använda relevant bjälklag som värd för objektet krävdes specifika elementytor. Således hämtas alla ytor för de relevanta bjälklagen. Därefter hämtas normalvektorer för ytorna och de ytor som inte har normalvektor positiv i Z-led bortsorteras (se Figur 31.).



Figur 31. Konceptuellt flödesdiagram av tillvägagångssättet för face-based utplacering av ursparningsobjektet. Notera att det endast är två ytterligare processer i flödesdiagrammet (Ytor för bjälklag och normalvektor).

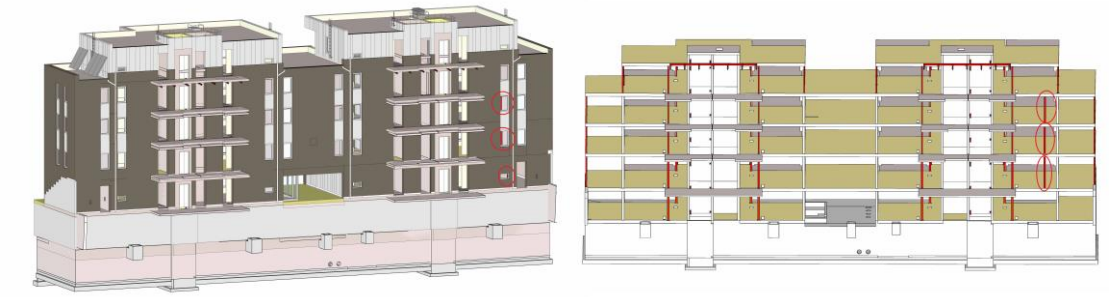
Skriptet använde sig av olika paket för funktioner som inte fanns i Dynamos egna bibliotek. Dessa paket var Archi-labs, BimorphNodes, Clockwork, Rhythm, SpringNodes.

3.4 Fall 4: Identifiering av pelare som syns i A-modellens fönster

Fall 4 behandlade identifiering av pelare (K) som syns i A-modellens fönster. Nuvarande tillvägagångsätt är genom okulära besiktningar. Målet med skriptet är att identifiera om en pelare sitter i eller precis intill ett fönster som resulterar att pelaren syns genom fönstret.

3.4.1 Modellen och dess information

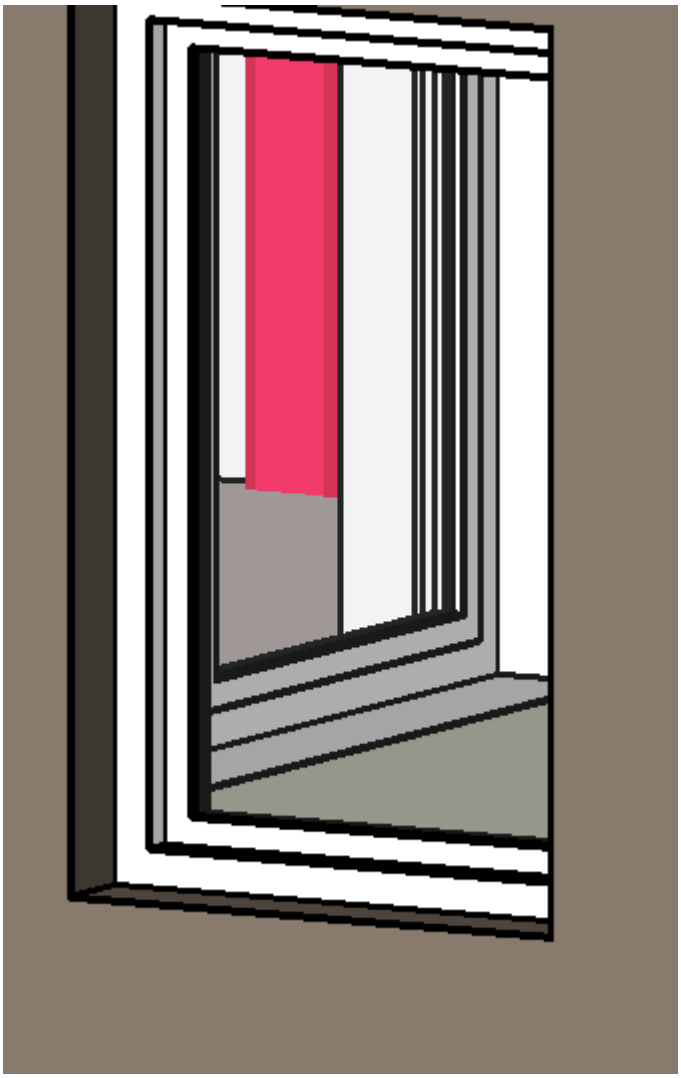
I fall 4 används återigen A- och K-modellen från fall 3. I modellen placerades pelare ut i ett experimentellt syfte för att tillhandhålla att ha något att laborera med för att lösa fallet med att identifiera pelare som syns genom fönster (se Figur 32.).



Figur 32. Modellen till vänster är K-modellen och modellen till höger är A-modellen. De inringade pelarna är de som placerades i experimentellt syfte.

3.4.2 Krav på script

Skriptet skapades utifrån ett krav. Kravet är att pelare sitter i eller precis intill ett fönster ska identifieras. Insynsvinkel 90 grader ska endast beaktas det vill säga att insyn genom fönster-fönster, insyn genom fönster-dörrhåltagningar ska inte undersökas (se figur 33.).



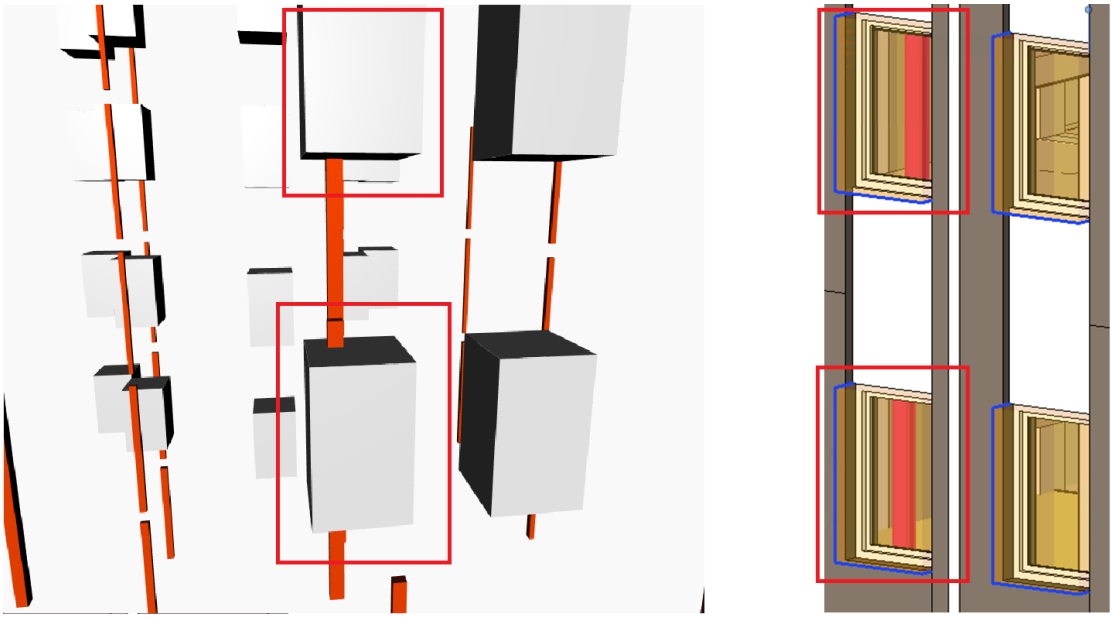
Figur 33. Skärmklipp i Revit som visar insyn av pelare (röd) från fönster-fönster.

3.4.3 Beskrivning av Dynamoskriptets flöde

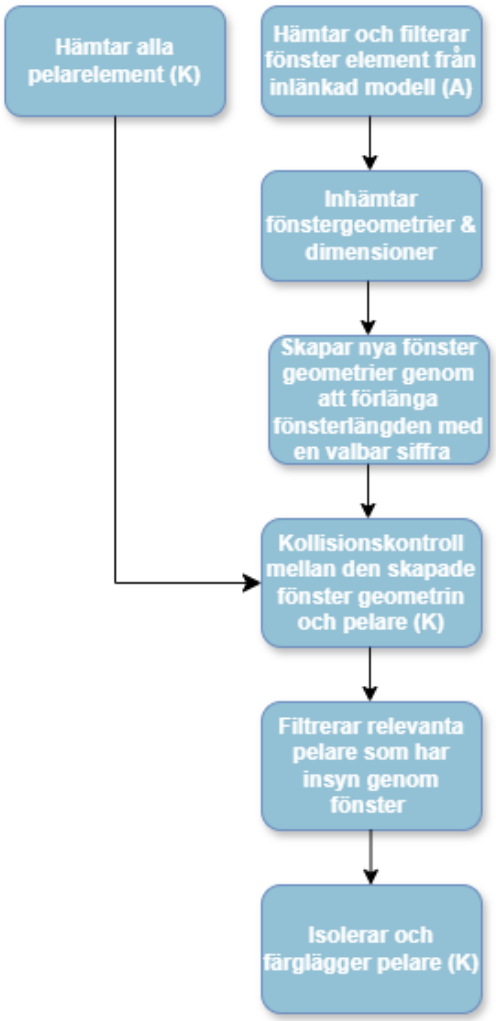
Två skript skapades som hade olika tillvägagångssätt. Skript 1 bygger på att inhämta fönster geometrier från A-modellen och förlänga de. Medan skript 2 bygger på att hitta insida fönster och korresponderande vektor och förflytta fönstergeometrin i riktning mot fönstrets insida.

3.4.3.1 Skript 1

Skript 1 inleds med att hämta alla pelarelement från K-modellen. I samma skede hämtas alla fönsterelement från den inlänkade A-modellen. Därefter skapas dynamogeometrier utifrån fönstrets dimensioner. Dynamogeometrin som skapas plockar fönstrets dimensioner och förlänger fönsterlängden på vardera sida och skapar dynamokuber. Därefter utförs en kollisionskontroll mellan pelare och den skapade geometrin för att upptäcka om pelare (K) kolliderar med den skapade geometrin (se Figur 34.). Kontrollen skapar en lista med pelare som har insyn genom fönster. Slutligen isoleras pelarna i Revit för att lättare kunna upptäckas och färgläggs med valfri färg. Resulterande skript visualiser i ett flödesdiagram (se Figur 34.)



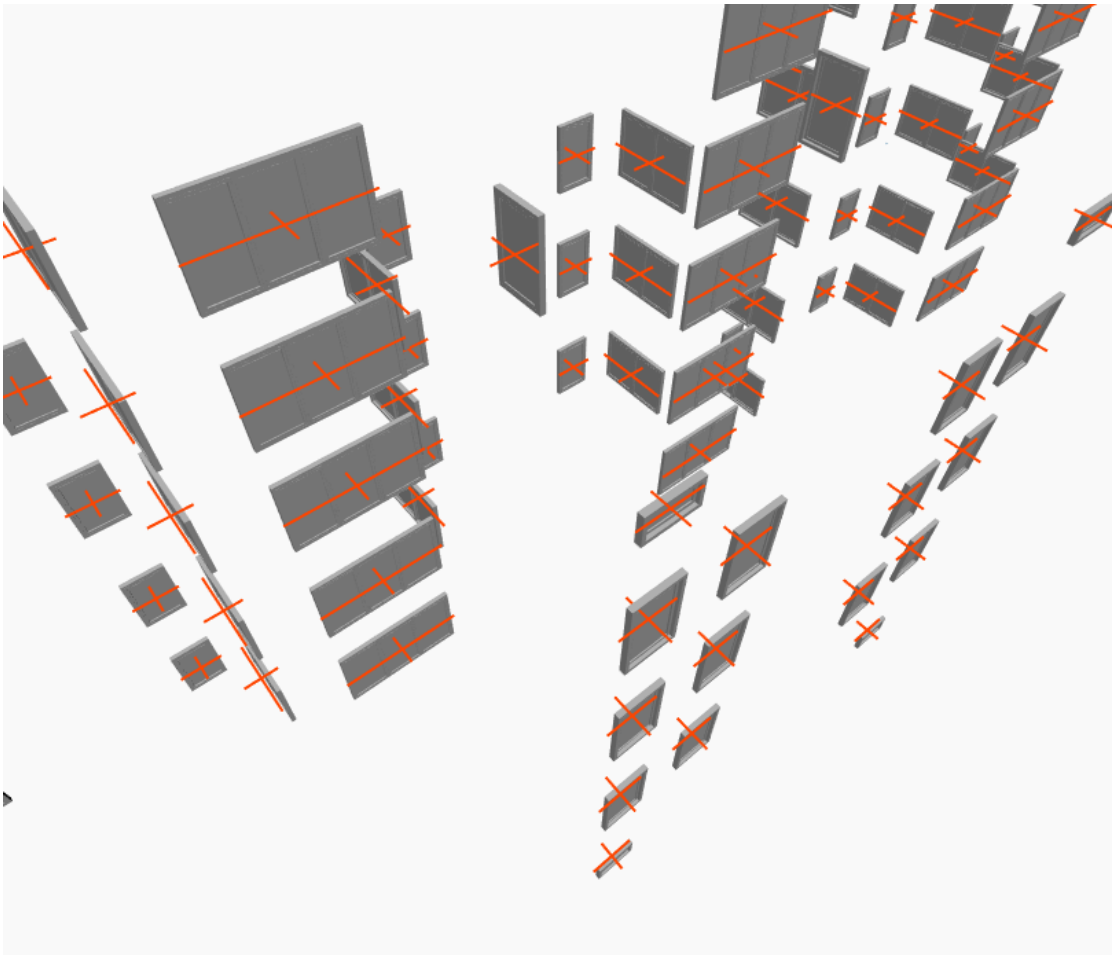
Figur 34. Den vänstra bilden visar den skapade dynamogeometrin som kolliderar med pelare och den högra bilden illustrerar det i Revit.



Figur 35. Konceptuellt flödesdiagram över Skript 1.

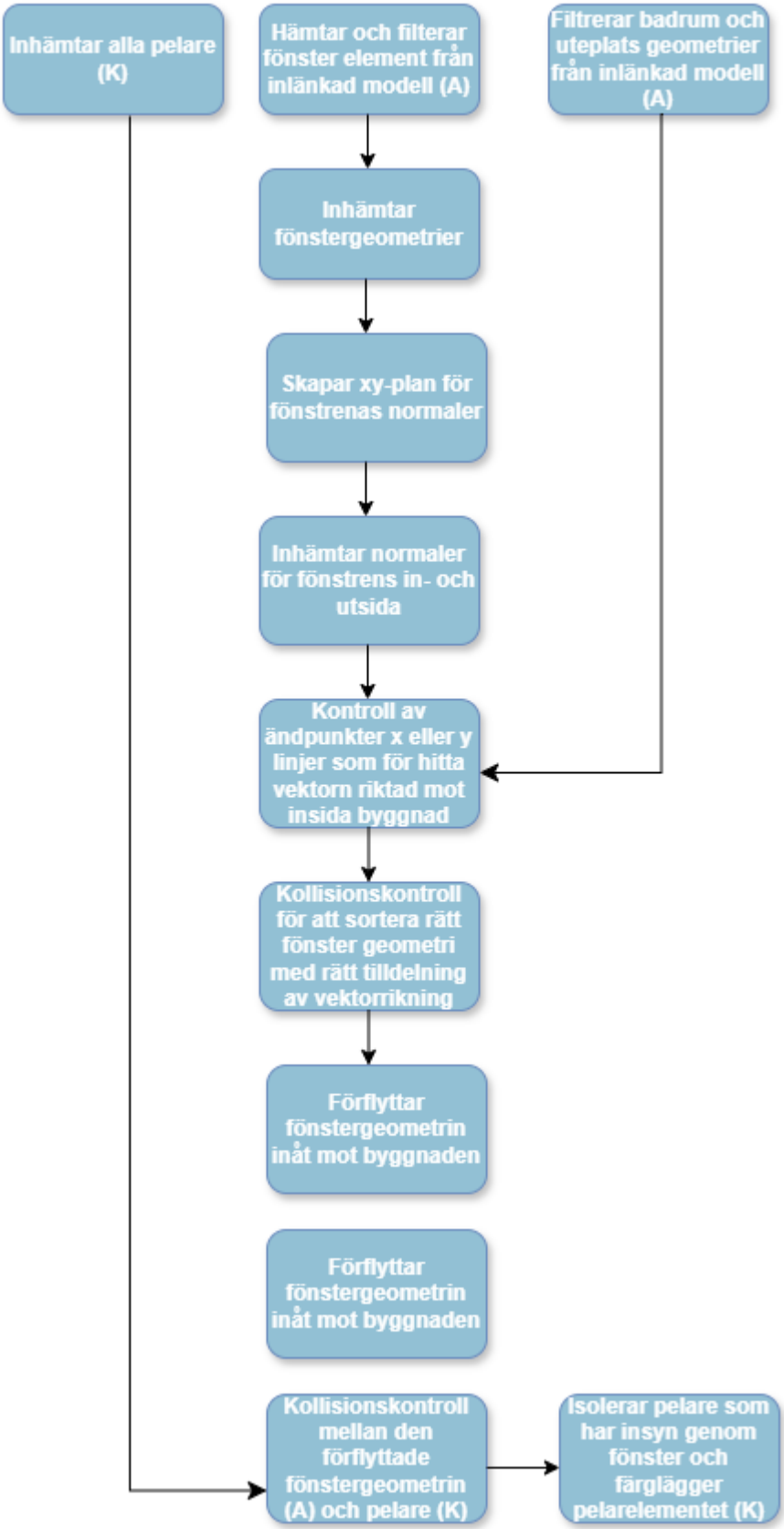
3.4.3.2 Skript 2

Inledningsvis i skript 2 inhämtas alla pelarelement, rums- och fönstergeometrier. Genom fönsterposition - och geometrier skapas xy-plan för fönsternas normaler det vill säga xy-plan vinkelrät mot fönstret (se Figur 36.).



Figur 36 Illustration i Dynamo av xy-planen som symboliseras av de röda linjerna. De rektangulära geometrierna är fönstergeometrier. Xy-planen skapar normaler för fönstret.

För att identifiera normalen in mot byggnaden kontrolleras ändpunkter av x och y linjerna. En kollisionskontroll utförs mellan ändpunkterna och rumsgeometrier för att se om ändpunkten ligger i ett rum. Kontrollen skapar en lista med alla punkter riktade in mot byggnaden samt vektorer med korresponderande riktning. Därefter förflyttas fönstergeometrin i riktningen in mot byggnaden med en valbar siffra och en kollisionskontroll utförs mellan den förflyttade fönstergeometrin och pelare från K-modellen. Skript 2 avslutas på samma sätt som skript 1. Kontrollen skapar en lista med pelare som har insyn genom fönster. Slutligen isoleras pelarna i Revit för att lättare kunna upptäckas och färgläggs med valfri färg.



Figur 37. Konceptuellt flödesdiagram av skript 2.

3.5 Fall 5: Fördjupning av att skapa K-objekt utifrån A-modellens geometri

I fall 5 fördjupas komplexiteten av skapandet av K-objekt utifrån A-modellens geometri. Processen som undersöks i fallet är skapandet och förskjutning av bjälklag (K) utifrån insida yttervägg (A) samt placera ut pelare (K) efter önskad förskjutning från insida yttervägg (A).

3.5.1 Modellen och dess information

I fall 5 återanvänds A- och K-modellen från fall 3. Flerbostadshuset har en genomgående portal centrerad på entréplan samt delade bjälklag. K-modellen har ett flertal pelarfamiljer med olika tvärsnitt och bjälklagstyper inladdade. Skriptet utgår från att placera pelare av typen VKR 100x100x10 (kvadratiskt tvärsnitt), VKR 150x100x10 (rektangulärt tvärsnitt) samt betongbjälklag 260 mm.

3.5.2 Krav på skript

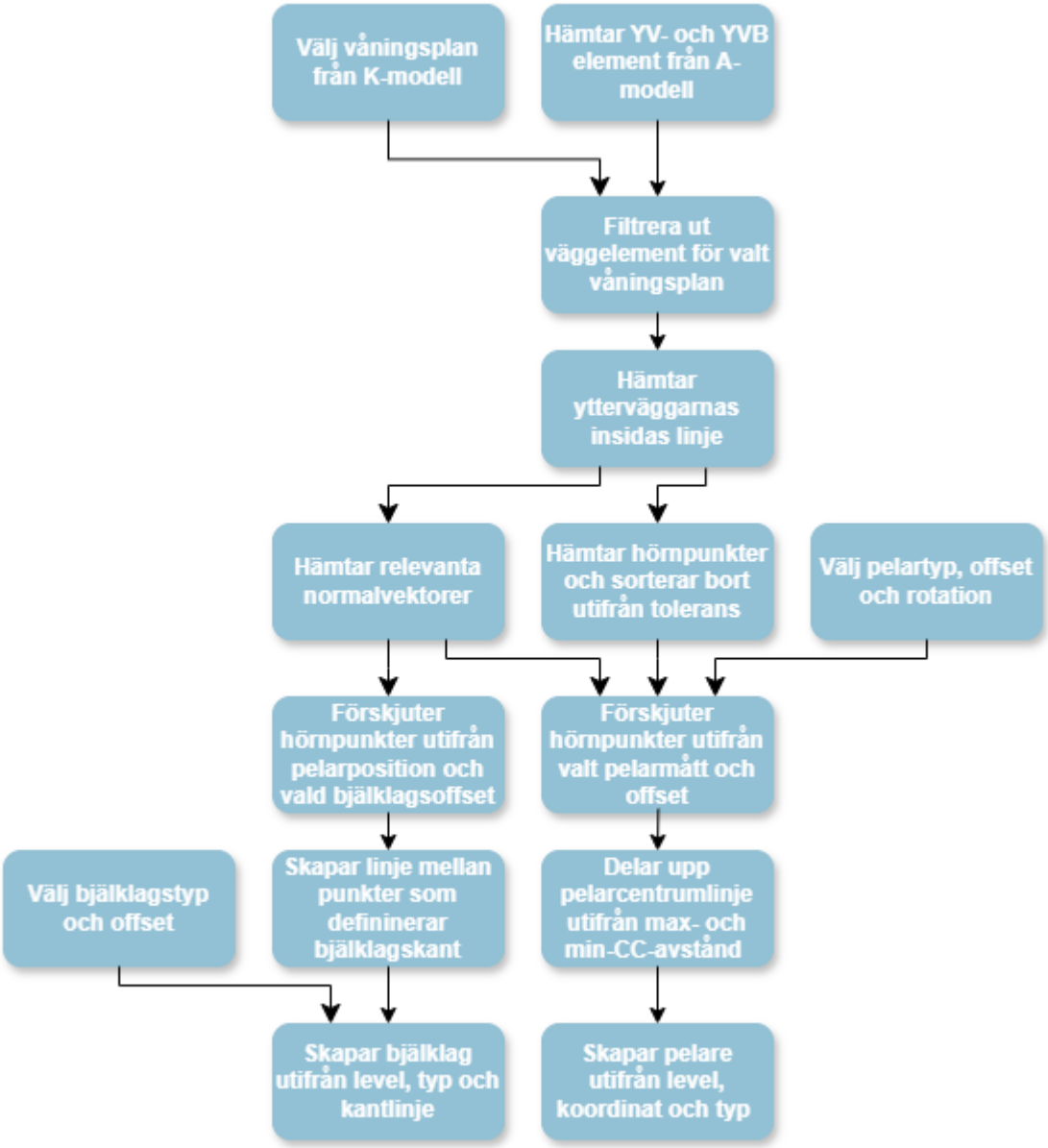
Krav som ställdes på skriptet var att ha möjligheten som användare att välja förskjutning av bjälklagets ytterkant samt förskjutning och rotation av pelare. Utöver förskjutningen krävdes det att kunna ställa in maximalt och minimalt CC-avstånd mellan pelare. Ingen lastnedräkning eller optimering av CC-avstånd krävdes och skriptet behöver endast hantera ett våningsplan i taget.

3.5.3 Beskrivning av Dynamoskriptets flöde

En övergripande beskrivning av dynamoskriptets flöde samt avsnitt som tydliggör vissa steg i flödet.

3.5.3.1 Övergripande beskrivning Dynamoskriptet

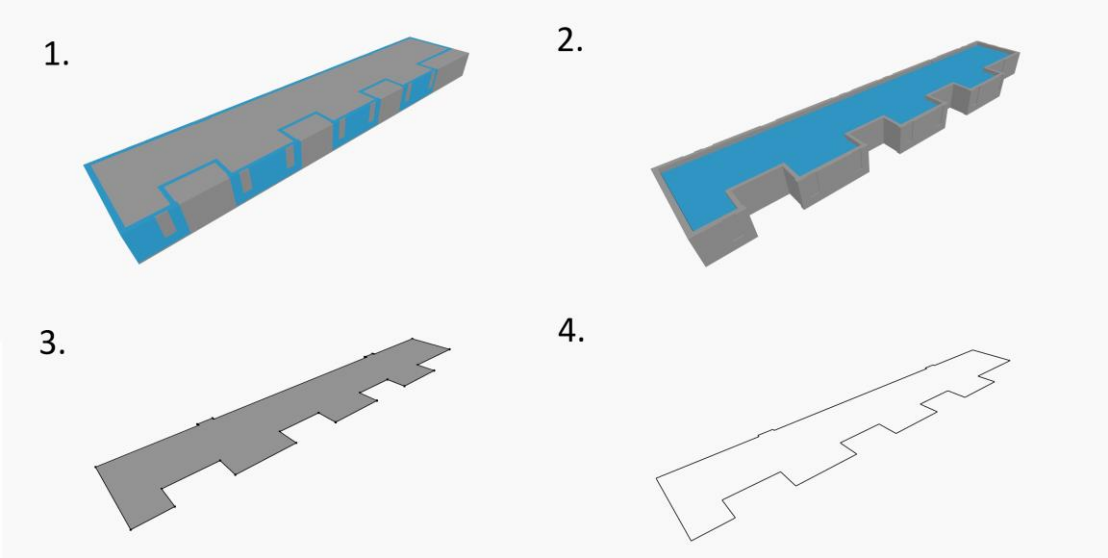
Skriptet inleds med att inhämta A-modellens ytterväggar från önskat våningsplan och omvandlar dessa till dynamogeometrier. Därefter skapas en geometri i mittpunkten för en volym skapad av väggeometrier och kontrollerar att ingen vägg kolliderar med geometrin för att undvika att vägglinjen bryts upp till två delar. Från resterande väggeometrier kan relevant kantlinje hämtas. Från kantlinjen hämtas hörnpunkter åt hörnpelare som förskjuts ut från linjen med en normalvektor. Mellan hörnpelare mäts avstånden där villkoren för utsättning är det önskade minimala och maximala CC-avståndet. Till sist förskjuts hörnpunkter ytterligare för att skapa bjälklagets ytterlinje. Det övergripande flödet redovisas i ett flödesdiagram (se Figur 38.).



Figur 38. Flödesdiagram över övergripande flöde för fall 5.

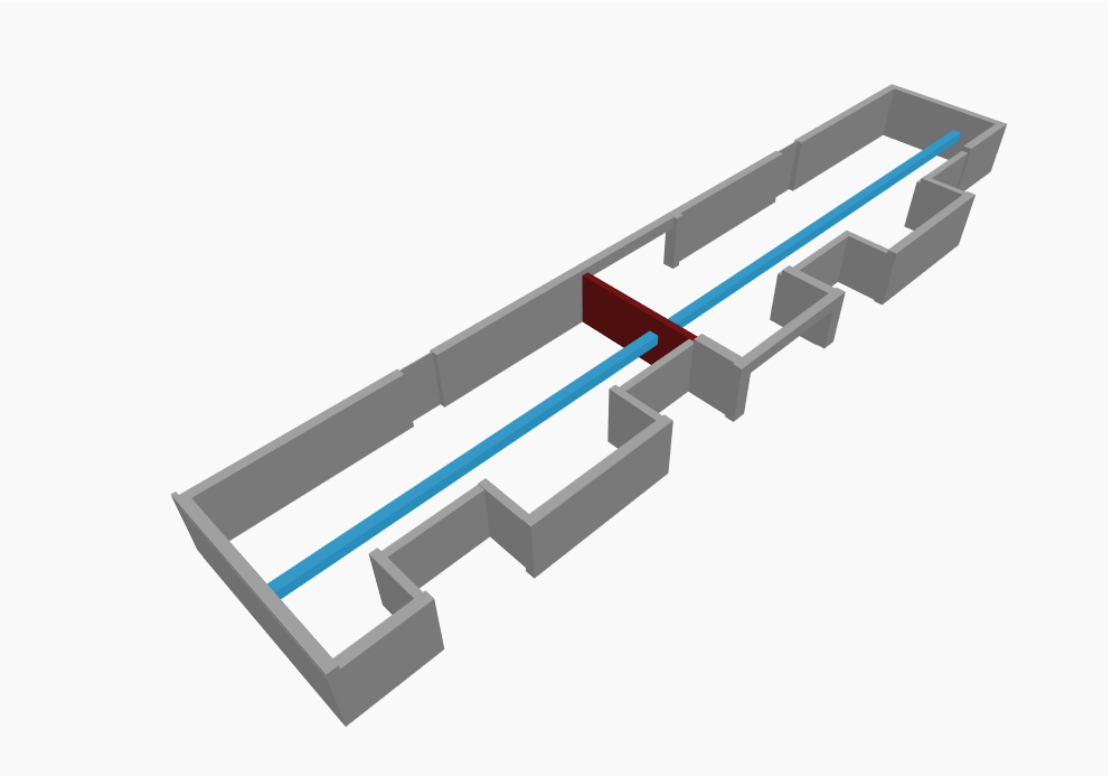
3.5.3.2 Definition av insida yttervägg

För att hitta ytterväggens insidas linje skapas ytterväggsgeometrier och en större volym skapade utifrån max- och minpunkt för samlingen av väggeometrierna. Därefter hämtas differensgeometrin och den största ovansidan filtreras ut. Till sist skapas en linje på utkanten av denna geometri (se Figur 39.).



Figur 39. Stegen för att hämta ytterväggens insidas kantlinje visualiserat med Dynamo.

För att undvika väggar som bryter ytterväggslinjen i två delar skapas en geometri mellan de yttersta väggarna. Efter geometrin krockar med en väggeometri (se Figur 40.) sorteras väggen bort. Denna funktion implementerades eftersom entréplanet har en portal med ytterväggar som bryter vägglinjen.

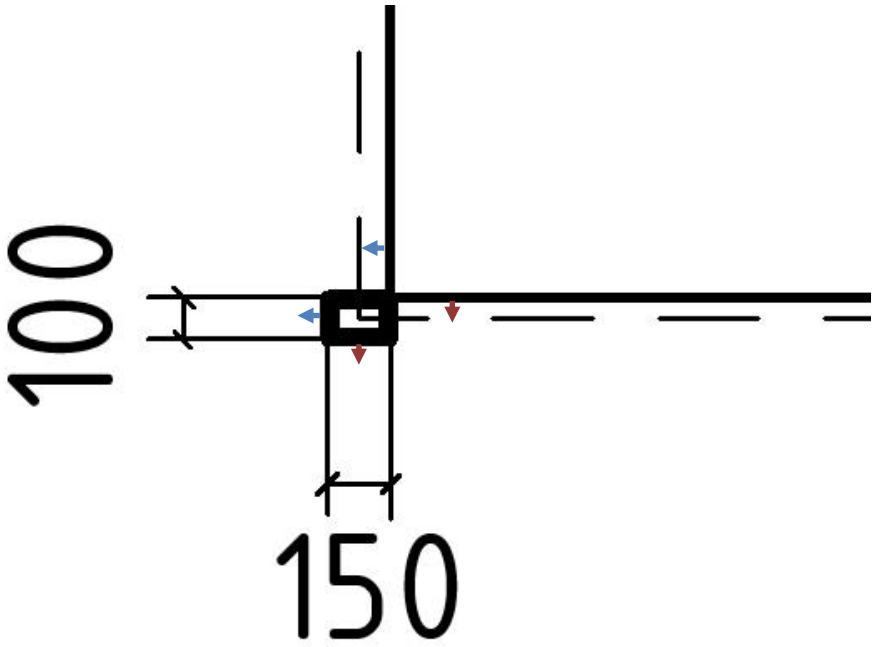


Figur 40. Bortsortering av yttervägg (röd) som delar insida väggeomtrier (grå) med hjälp av kollision av inplacerad geometri (blå).

3.5.3.3 Placera hörnpelare

Vägglinjen som skapas är en ”polycurve”, en linje sammansatt av flera raka linjer. För att få hörnpunkterna hämtas punkterna för polycurve:ns linjeuppdelning. Eftersom dörrhåltagningar skapar ojämnheter sorteras dessa punkter bort genom att mäta distanser mellan alla punkter och sedan bortsortera punkterna med en

distans under 1 m. Då skriptet dels arbetar med pelare av rektangulärt tvärsnitt, krävs olika stor förskjutning i olika riktningar. För att hantera pelarens rotation skriver användare in önskad rotation av pelare innan vidare beräkning av förskjutning sker. Förskjutningen löses med hjälp av att återanvända ytan som skapar vägglinjen. Ytan ges en tjocklek och på så sätt sidoytor. En dynamogeometri som representerar pelarna skapas för att para ihop rätt normalvektor med respektive sidoyta av pelare. Sidoytornas normalvektorer ger riktningen och hörnpunkterna för linjen förskjuts med pelarens halva h- och b-mått samt önskad distans från vägglinje (se Figur 41.).



Figur 41. Tillvägagångssätt för förskjutning av pelare (fet linje) från insida yttervägg (tunn linje), pelarens centrumlinje (streckad) och förskjutning (pilar) inkluderad. I exemplet används ingen ytterligare förskjutning.

3.5.3.4 Placera pelare mellan hörnpunkter

För att placera pelare mellan hörnpunkter sattes punkter genom att dela upp centrumlinje mellan hörnpelare i jämna mellanrum. Villkoret för maximalt CC-avstånd uppfylls genom att dividera CC-avståndet mellan hörnpelare med maximalt CC-avstånd och sedan avrunda resultatet uppåt. Linjen delas upp i jämna mellanrum som inte överstiger maximalt CC. För att uppfylla minimalt CC-avstånd sorteras pelarcentrumlinjer bort som är ≤ dubbla minimala CC-avståndet. När avstånden är beräknade placeras pelarobjekten för respektive punkt med önskad top- och bottomoffset och kontrolleras att de ligger i en yttervägg, om inte skapas inte pelaren. Den sista kontrollen görs för att inte få pelare i den genomgående portalen på entréplan.

3.5.3.5 Skapa bjälklag

För att skapa bjälklaget, förskjuts hörnpunkter ytterligare med hjälp av samma vektorer som hörnpelare använder. Punkterna förskjuts utifrån pelardimensioner

samt önskad offset. Nya linjer skapas mellan dessa punkter och bjälklag placeras ut efter valt familjetyp, kantlinje och våningsplan.

4 Resultat

Under detta kapitel presenteras resultat från fallstudien.

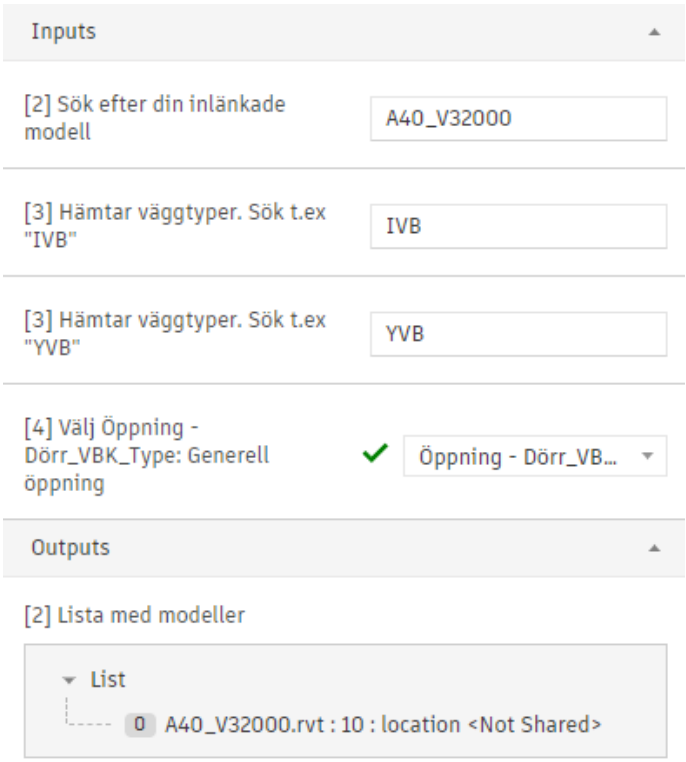
4.1 Fall 1

Resultatet för fall 1 är ett skript som placerar ut håltagningar i bärande väggelement (K) och skapar familjetyper för varje variation av håltagningarna. Körtid för skriptet med Intel(R) Core(TM) i5-9500 processor resulterar i en körtid på 4,5 sekunder.

4.1.1 Dynamo Player

De noder som har specificerat som inputs i Dynamoskriptet kräver interaktion och manipulering från användaren för att skriptet ska fungera. Val av vilka noder som sätts som inputs baseras ofta på oförutsägbara variabler. Man vill även kunna ge användaren möjligheten att manipulera vissa funktioner efter behov. Slutligen kan vissa noder kräva att användaren behöver genomföra nödvändiga manuella åtgärder. Val av inputs redovisas nedan i Figur 42.

Namn på inlänkad A-modell [2], är en förekommande specificerad input i alla analyserade fall. Logiken till att det sätts som en input är på grund av att det kan finnas flera inlänkade modeller i ett Revit projekt samt att det är oförutsägbart att veta vad arkitekten döper A-modellen till. Logiken bakom [3] är att ge användaren möjligheten att avgöra vilka väggelement användaren tycker är relevanta. En manuell inmatning av VBK förkonstruerade familj för håltagningar är nödvändig för att skriptet ska fungera [4].

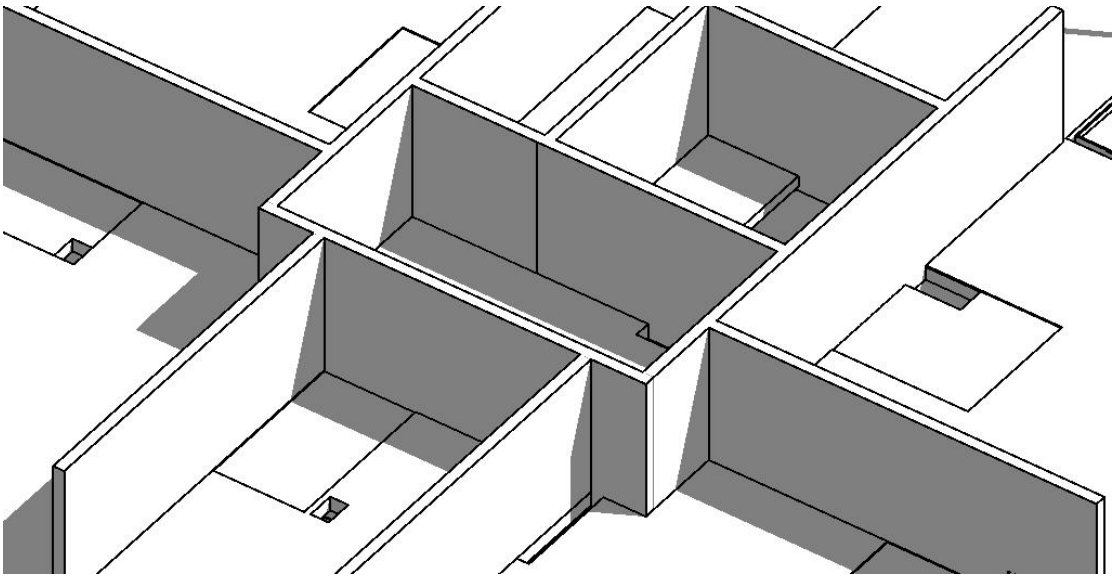


Figur 42. Dynamo player för fall 1

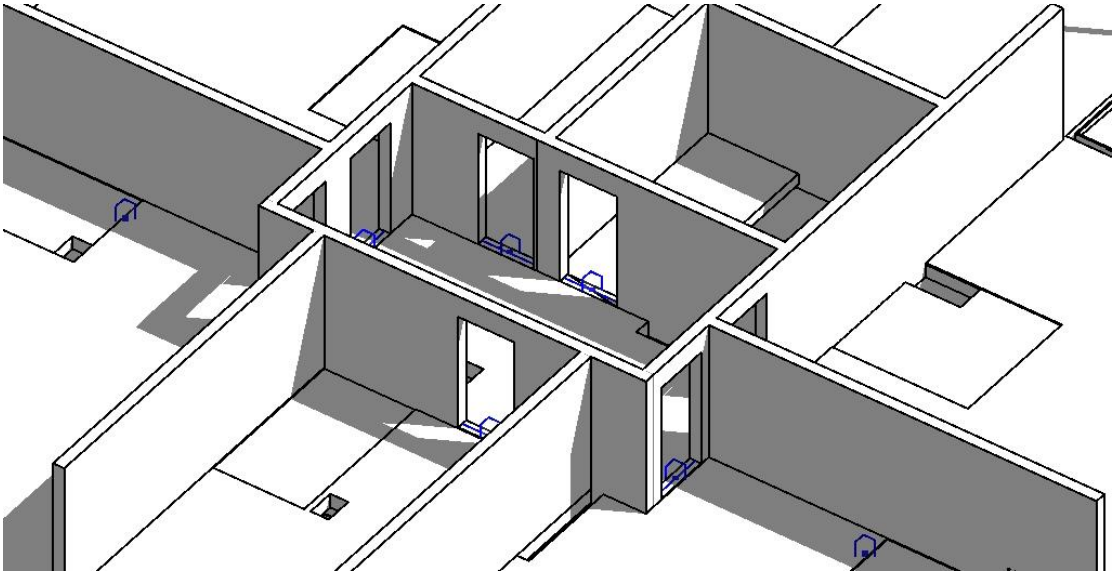
4.1.2 Placering

Vid placering av håltagningarna krävs att berörande dörrdimensioner inhämtas och omvandlas till dynamogeometrier. Kollisionskontroll mellan antalet dynamogeometrier och bärande väggar i K-modellen visar att samtliga 27 kollisioner upptäcks. Kontrollen sorterar och visar endast de dörrar i A-modellen som överlappar med bärande väggar i K-modellen och irrelevanta dörrar bortsorteras.

K-modellen visar att 27 håltagningsselement placeras i korresponderande väggelement utifrån dörrgeometri. Håltagningens bredd, längd och läge stämmer överens med dörrarnas karmmått (A). (se Figur 43. och 44.).



Figur 43. Sisjödals översta våningsplan, K-modell, före kört skript.



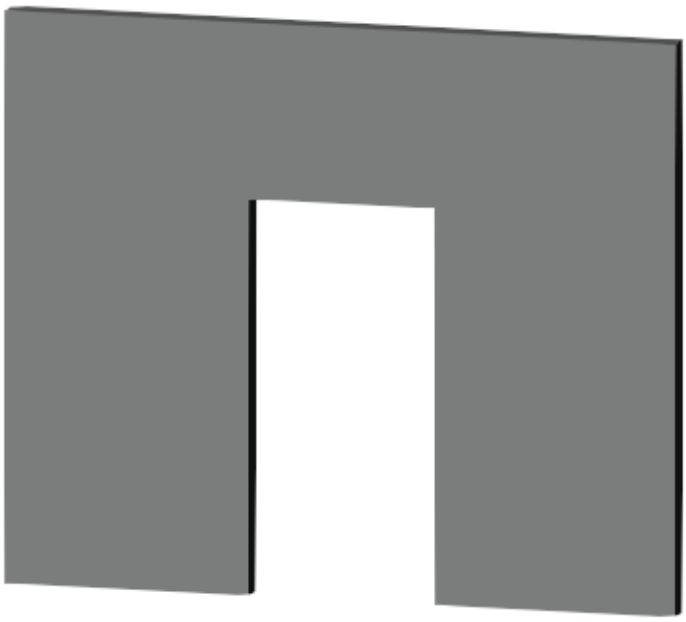
Figur 44. Sisjödals översta våningsplan, K-modell, efter kört skript.

4.1.3 Familjetyper

Skriptet skapar unika familjetyper utifrån ”Öppning – Dörr_ VBK_ Type”, beroende av dörrdimensioner (se Figur 45.) Skapandet av familjetyper sker utifrån dimensionerna 1010x2110 och 1020x2110.

Figur 45. Dörrdimensioner från A-modellen är synkade med de skapade dörröppningarna i K-modellen

Vid skapande av typbaserade håltagningar användes VBK:s förgenererade dörröppning, som kräver en manuell inladdning av familjen (se Figur 46). Familjens dimensioner är styrbara och reglerades för att skapa de typbaserade hålltagningarna.



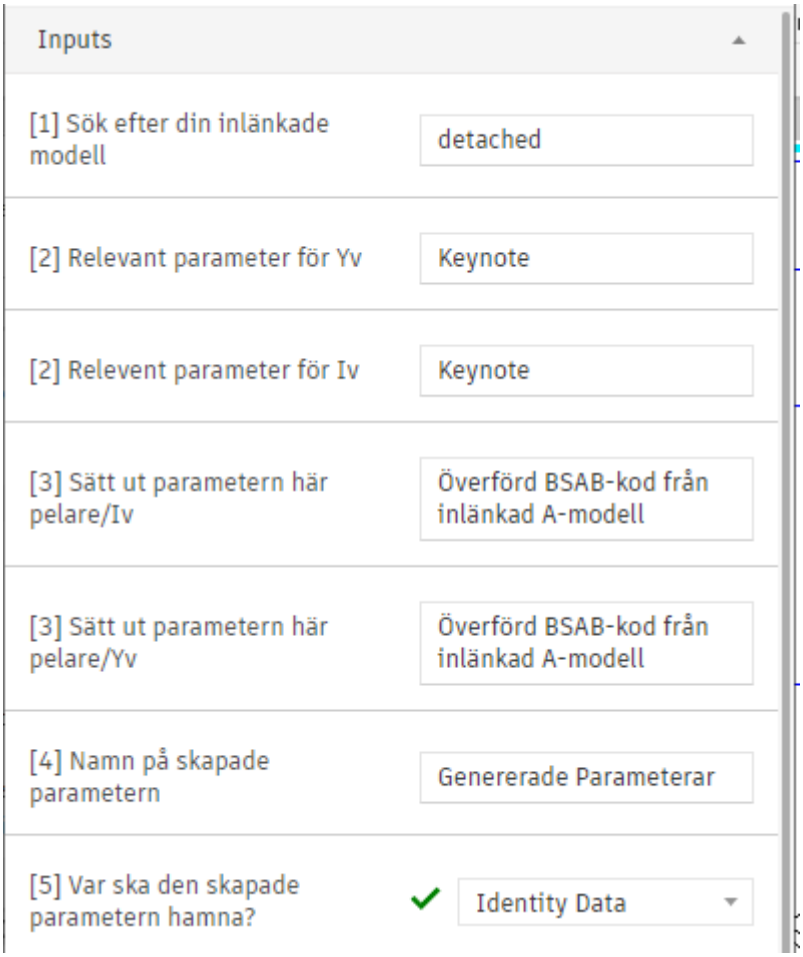
Figur 46: ”Öppning – Dörr_ VBK_ Type

4.2 Fall 2

Skriptet överför BSAB-koder från väggelement (A) till pelarelement (K) genom en skapad shared textparameter eller en befintlig. Det totala antalet BSAB-koder som skrivs över (A) till pelare (K) är 40 koder och är synkat med antalet överlappande samt relationer. Textparametern hamnar på valbar placering. Körtid för skriptet med Intel(R) Core (TM) i5-9500 processor resulterar i en körtid på 3,3 sekunder.

4.2.1 Dynamo player

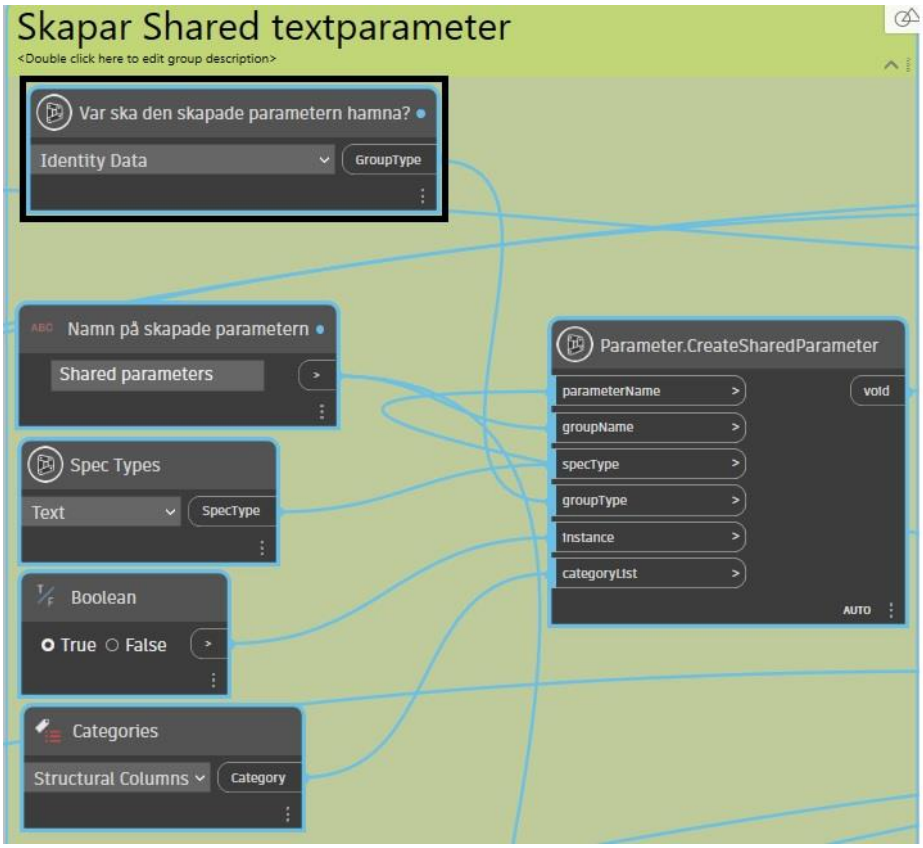
Namn på inlänkad A-modell [1]. Man vill kunna ge användaren möjlighet att kunna överföra valfri metadata mellan A och K därmed har man möjlighet att manuellt skriva in vilken parameter man vill göra över [2]. Det kan vara till intresse att dela upp inner- och ytterväggar därmed är [2] och [3] uppdelade. Vid
[4] ger möjligheten för användaren att avgöra benämning på den skapade ”Shared parametern”. Slutligen får även användare att avgöra under vilken grupparametern ska sättas ut under [5]:



Figur 47. Dynamoplayer för fall 2

4.2.2 Överförd metadata

De överförda parametrarna hamnar under önskad gruppering Identity Data. Skriptet kan köras flera gånger för att överföra fler parametrar på samma pelare. Var den skapade textparametern ska placeras är valbart genom noden som visas nedan i Figur 48.

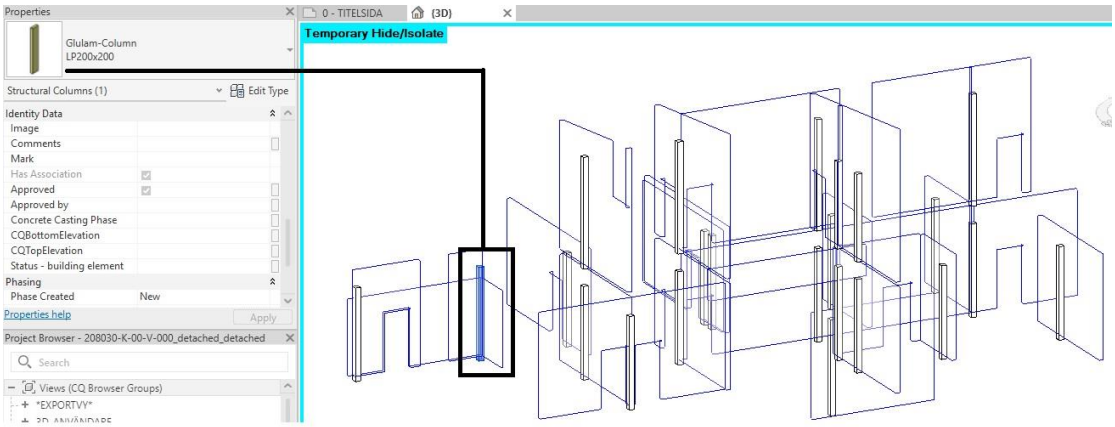


Figur 48. Sharedparameter-noden i Dynamo

Den skapade ”shared parameter” hamnar utanför Revit projektet som en egen textfil vid en vald delad parameterfil.

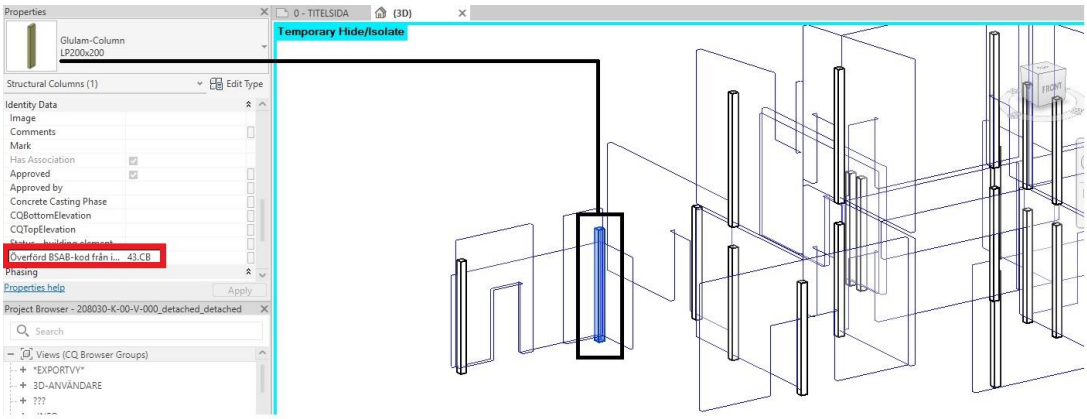
Figur 49. Sharedparameter filen där metadata skrivs över

Innan skriptet körs finns ingen tilldelad metadata under Identity Data på pelare som sitter i väggar.



Figur 50. Innan metadata från A till K överförs

När skriptet körs överförs valbar metadata från A till K. Figur 51. nedan visar att överförd BSAB kod från A till K är 43.CB vilket innebär att pelare (K) sitter i en innervägg (A) (ej stominnervägg) som kan läsas av genom Figur 24. ovan.



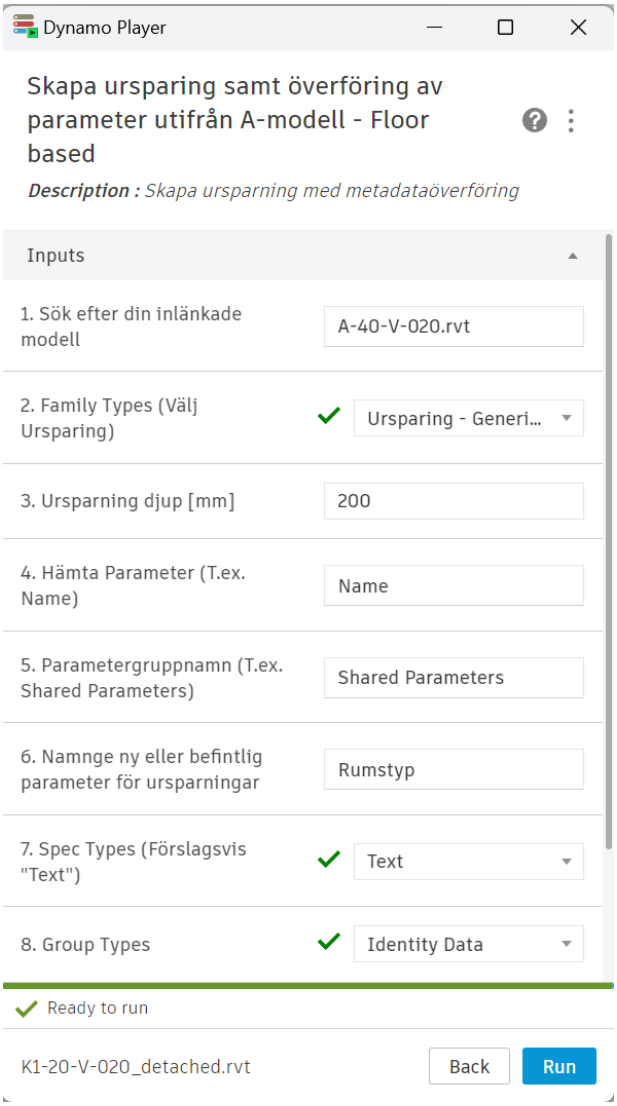
Figur 51. Skärmklipp från Revit, överförd metadata (röd markering) samt väggelement (A) som symboliseras av de blå linjerna.

4.3 Fall 3

Resultatet för fall 3 är två skript för två olika tillvägagångssätt. Skripten placerar ut ursparningselement i bjälklagen, skapar familjetyper för varje variation av ursparning samt överför önskad textparametrar till ursparningen. Körtid för floor-basedskriptet med en Ryzen 5 5500U processor resulterar i en körtid på 3,8 sekunder. Körtid för face-basedskriptet med samma processor resulterar i 10,9 sekunder.

4.3.1 Dynamo player

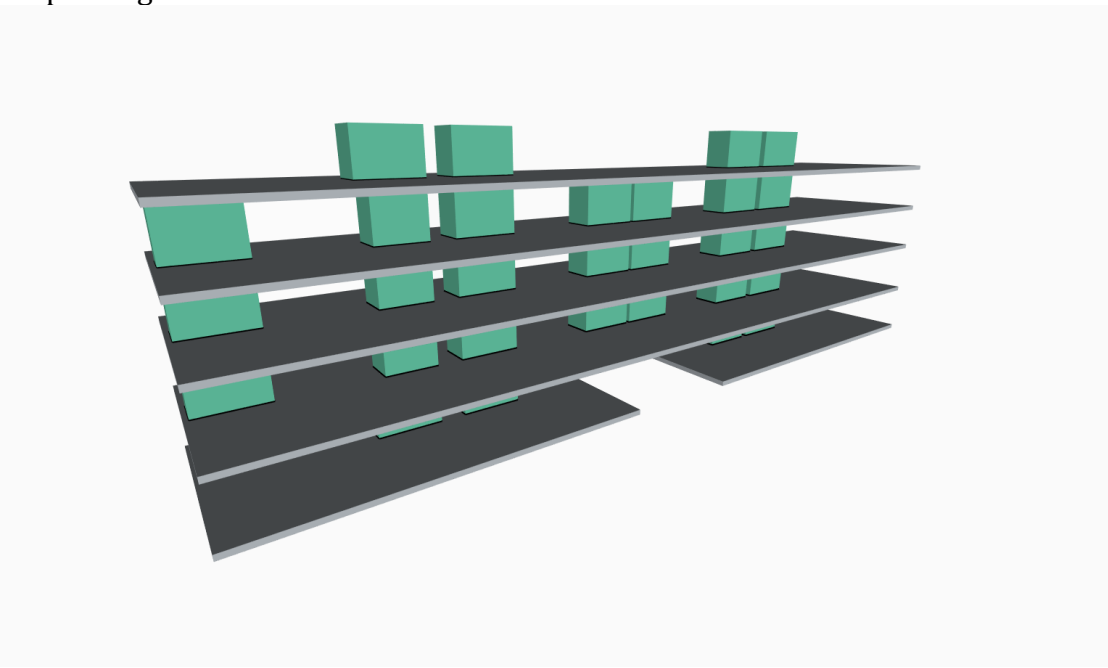
Inputs för användaren redovisas i Figur 52.



Figur 52. Dynamo player för floor-based utplacering och metadataöverföring.

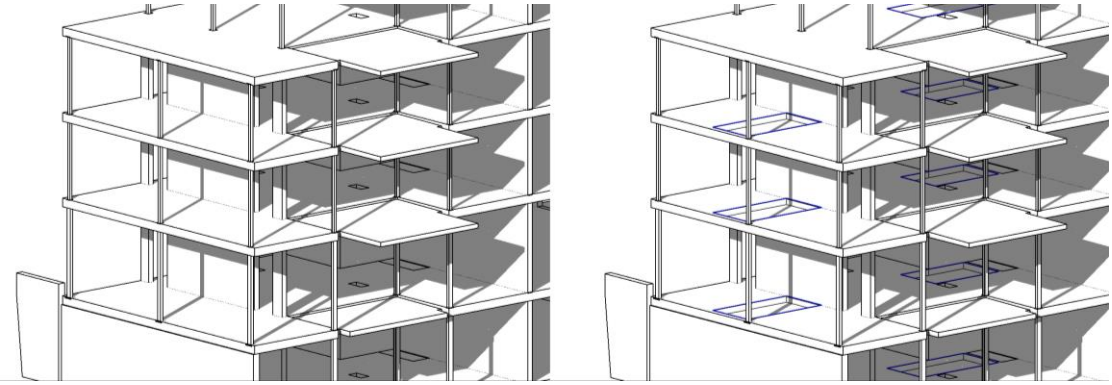
4.3.2 Placering

Vid placering av ursparningarna krävs att berörande geometri inhämtas och sorteras korrekt. Kontroll av antal dynamogeometrier visar att samtliga 29 badrum- och WC-rumsobjekt inhämtas och konverteras till Dynamogeometri (se Figur 53.). Kontrollen visar även att irrelevanta bjälklag som inte kräver ursparningar bortsorteras.



Figur 53. 3D-Arbetsyta i Dynamo, rumsgeometrier och bjälklagsgeometrier för skript 1.

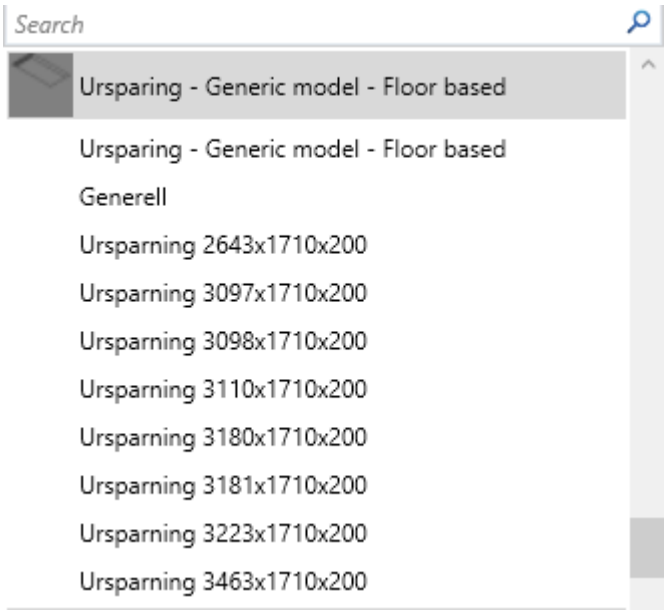
K-modellen visar att 29 ursparningselement placeras i korresponderande bjälklag utifrån rumsobjektens geometri. Ursparningarnas bredd, längd och läge stämmer överens med rumsgeometrierna (se Figur 54.).



Figur 54. K-modell, före (vänster) och efter körning av dynamoskript (höger) där blå linje är rumsgeometrins kanter. (Blåa kantlinjer höjs upp 10 mm för att särskilja linjerna från bjälklagets kant, annars syns inte linjerna då det ligger i exakt samma position som bjälklagets/ursparningens kantlinje i Revit).

4.3.3 Familjetyper

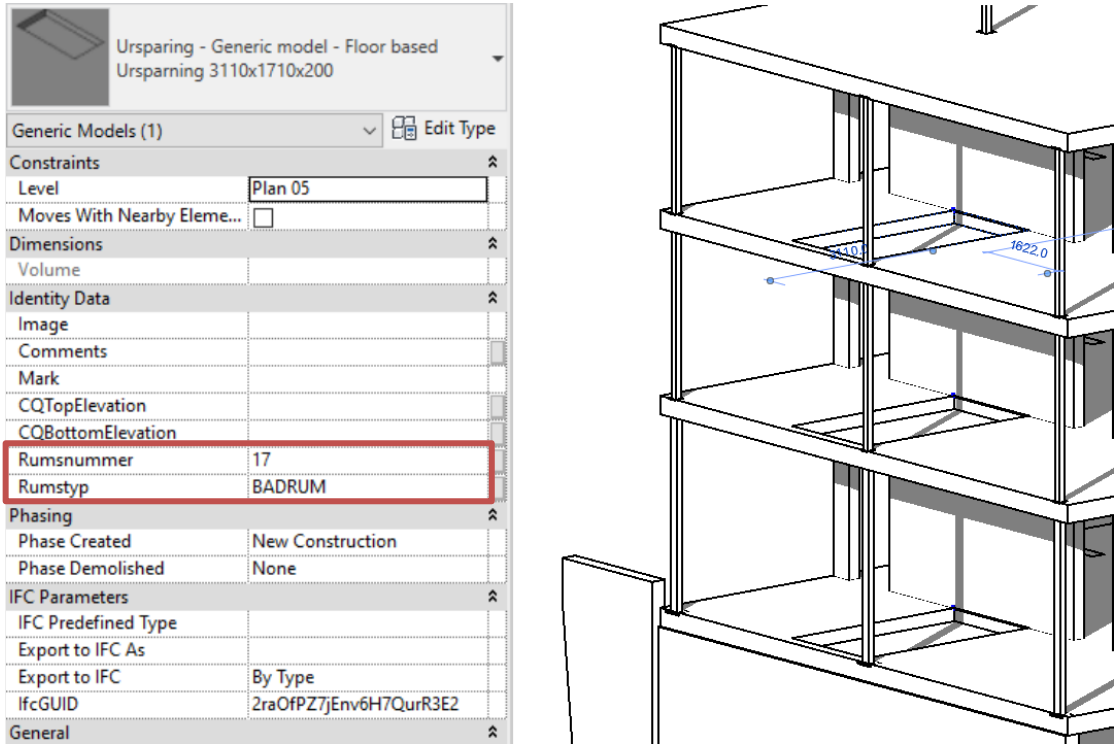
Skriptet skapar 8 unika familjetyper utifrån ”Ursparing – Generic model – Floor based Generell, beroende av rumsgeometrin (se Figur 55.). Rummen mäts exakt men i skapandet av familjetyper sker en avrundning på ± 0,5 mm och således skapas ursparningar med 1 mm skillnad, exempelvis 3097x1710x200 och 3098x1710x200.



Figur 55. Skapade familjetyper efter körning av skriptet.

4.3.4 Överförda parametrar

De överförda parametrarna hamnar under önskad gruppering, Identity Data. Efter kontroll med A-modellen stämmer rumsnummer och rumstyp överens med metadata överförd till ursparningarna (se Figur 56.). För floor-based-skriptet kan koden köras flera gånger för att överföra fler parametrar på samma ursparning. Denna möjlighet återfinns inte i face-based-skriptet utan kan endast överföra en parameter per utplacerad ursparning.



Figur 56. Skärmklipp från Revit, överförd metadata (röd markering) samt ursparningelementet (översta ursparningen av tre).

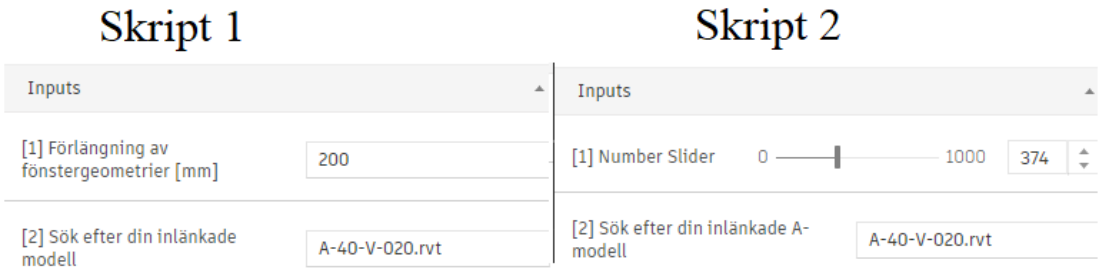
4.4 Fall 4

Resultatet av fall 4 blev två skript. Båda skripten åstakommer samma resultat men genom olika metoder. Skripten upptäcker de tre pelare som har insyn genom ett fönster och isolerar de i den aktiva vyn i Revit. Körtid för skript 1 med Intel(R) Core(TM) i5-9500 processor resulterar i en körtid på 4,2 sekunder. Körtid för skript 2 med Intel(R) Core(TM) i5-9500 processor resulterar i en körtid på 9,5 sekunder.

Figur 57: Skärmklipp från Revit som visar vilka pelare som syns i fönster som resulteras i att de isoleras i den aktiva 3D-vyn

4.4.1 Dynamo player

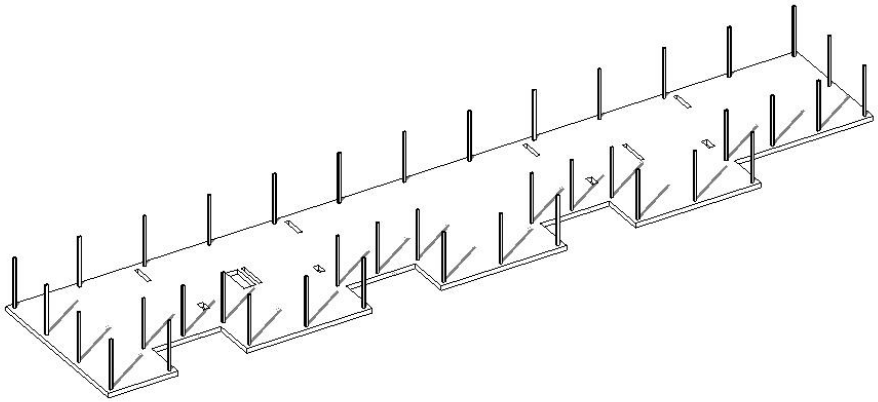
Val av inputs i skript 1 är hur många millimeter man vill förlänga fönstergeometrin på längden i y-led vid skapande av dynamogeometrier [1]. Den andra inputen [2] är namnet på inlänkad A-modell. Val av inputs i skript 2 är snarlika [1] är hur många mm man vill förflytta fönstergeometrier in mot bygnaden och [2] är likadan som i skript 1.



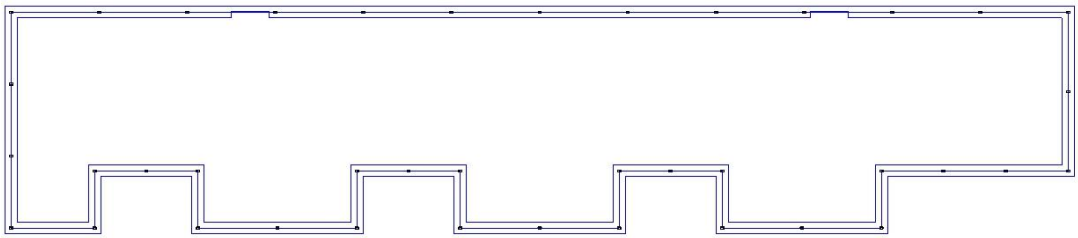
Figur 58: Dynamo player, fall 4

4.5 Fall 5

Resultatet av fall 5 blev två skript. Ett skript som hanterar rektangulära tvärsnitt och ett som hanterar kvadratiska tvärsnitt. Skillnaden mellan skripten är egentligen två olika listsorteringar för en av noderna men resterande uppbyggnad skriptens flöde är identiska. Skripten resulterar i att bjälklaget placeras ut efter förskjuten kantlinje samt förskjutna pelare med önskat min- och max CC-avstånd (se Figur 59. och 60.).



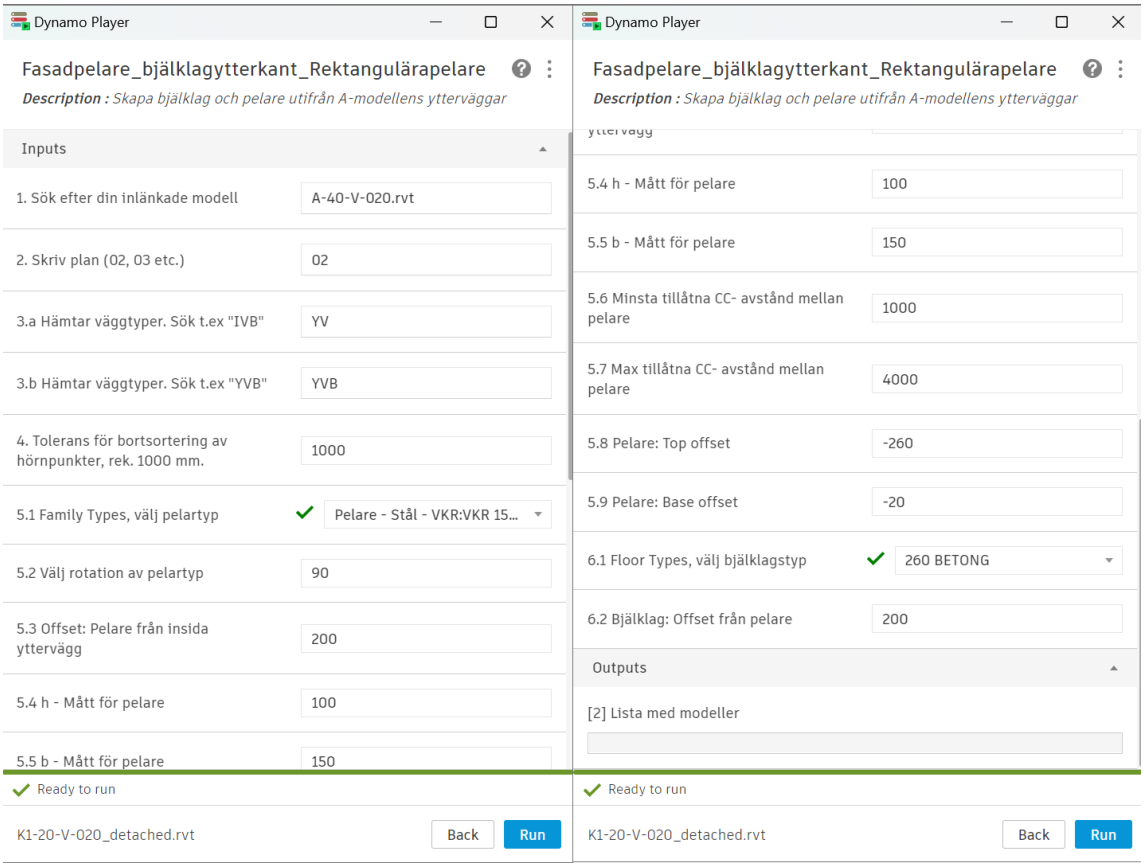
Figur 59. Resulterande pelare och bjälklag efter skriptet körs. Schakt existerade i K-modellen sedan innan.



Figur 60. Planvy av resulterande bjälklag (yttersta linjen), pelare och centrumlinje (mittersta linjen och punkter) och ytterväggens insidas kant (innersta linjen).

4.5.1 Dynamo player

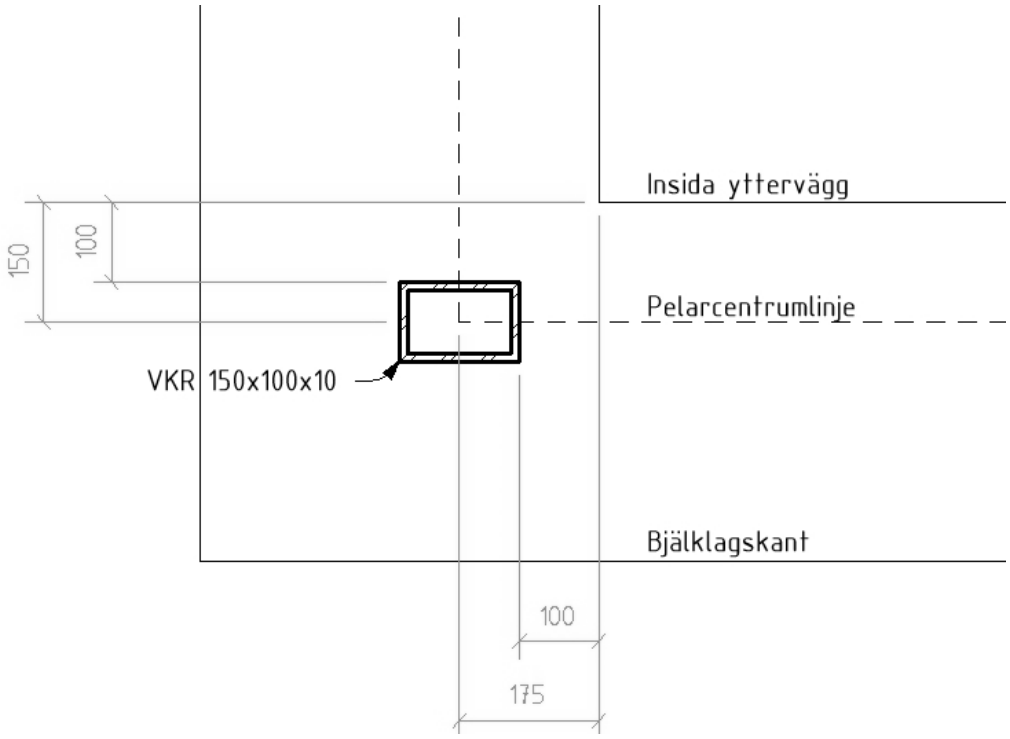
Inputsen för användaren av skriptet redovisas i Figur. 61.



Figur 61. Dynamo player för skapandet av bjälklag och pelare.

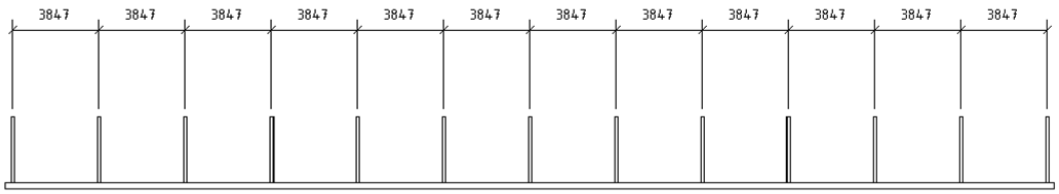
4.5.2 Skapandet av pelare

Vid inmatning av 100 mm önskad förskjutning placerar skriptet pelare med korrekt avstånd från insida yttervägg (se Figur 62.). Ingen pelare placeras i portalen på entréplan. Pelare förskjuts med önskad offset och halva h- och b-mått för pelare. Resultatet visar även att skriptet parar ihop rätt storlek på förskjutning med rätt vektor.

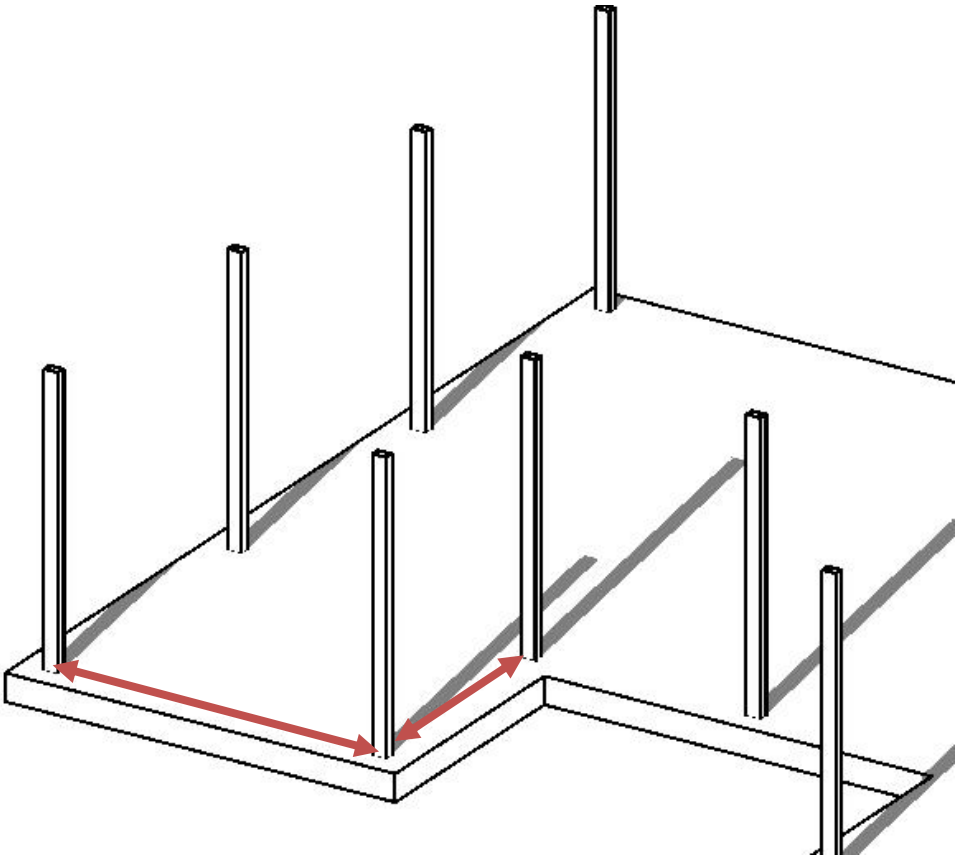


Figur 62. Skapad pelare och dess förskjutning i relation med insida yttervägg.

Inmatning av 4000 mm maximalt tillåtet CC-avstånd ger ett resultat där uppdelningen anpassas så CC-avstånden mellan pelare ej överstiger värdet (se Figur 63.). Vid inmatning av minimalt tillåtet CC-avstånd på 1000 mm skapas inte pelare på mellanrum som hade skapat ett CC-avstånd som understiger det inmatade värdet (se Figur 64.). Pelarna justeras även med top- och botoffset.



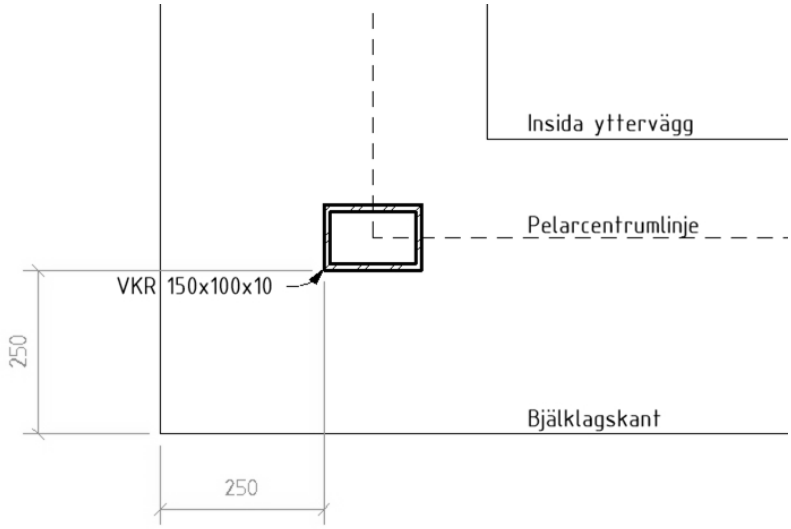
Figur 63. Fasadvy av bjälklag och skapade pelare på norrsida med CC-avstånd.



Figur 64. Utplacerade pelare på bjälklag. Röd markering indikerar mellanrum där pelare inte har skapats på grund av villkoret för minimalt CC-avstånd mellan hörnpelare.

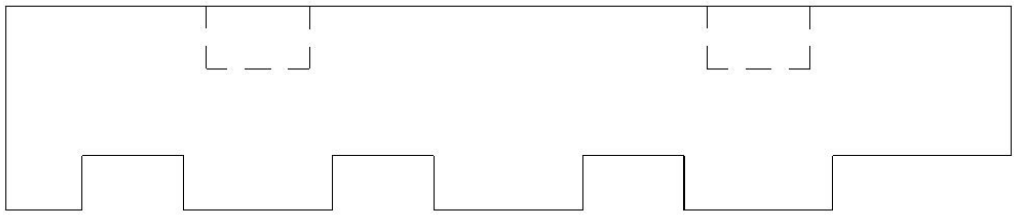
4.5.3 Skapandet av bjälklag

Bjälklaget placeras ut på rätt våningsplan och vid inmatning av en förskjutning på 250 mm, förskjuts kantlinjen med rätt riktning och storlek (se Figur 65.).

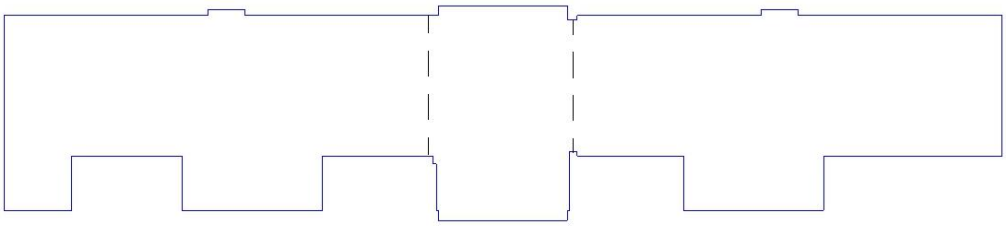


Figur 65. Bjälklagskantens förskjutning i relation med pelare.

För att se hur mycket kantlinjen skiljer sig från det befintliga modellens bjälklag utfördes en kontroll där offset från insida yttervägg sattes till 110 mm. Vid kontrollen upptäcktes att den nya kantlinjen är indentisk med befintligt bjälklag förutom vid hissarna (se Figur 66.). Hissarna täcks av ytterväggselement och därav går bjälklagets kantlinje förbi hissarna och gör inga indrag. För entréplan skapas ett sammansatt bjälklag i stället för två separata som det befintliga entréplanet har. För entréplanets delade bjälklag och portal blev resultatet ett enat bjälklag på grund av portalens ytterväggar (se Figur 67.).



Figur 66. Skillnad mellan skapat bjälklag (hela linjer) och befintligt modellerat bjälklag (streckad linje), Torpaterrassen våningsplan 3.



Figur 67. Ytterväggarnas insidas kantlinje (hela linjer) vid entréplan. De streckade linjer är skillnaden för kantlinje mellan de befintliga och nyskapade bjälklagen.

5 Diskussion

Under diskussion framförs resonemang kring möjligheter, utmaningar, begräsningar och andra tillvägagångssätt för dels specifikt för varje fall och dels Dynamo i helhet.

5.1 Utvärdering av fall

Under detta avsnitt utvärderas resultaten för de specifika fallen. Avsnittet omfattar begränsningar, för- och nackdelar med skriptens logik samt alternativa tillvägagångssätt.

5.1.1 Fall 1

Fall 1 hade som syfte att skapa håltagningar i bärande väggar (K) utifrån dörrar i A-modellen. Skriptet uppfyller sitt syfte men har begränsningar. En begränsning är dörrar som tillhör Curtain Wall. Dessa dörrars position returneras null vid inhämtning av koordinater. Orsaken undersöktes men ingen slutsats kring varför detta sker hittades. I en situation där dörren sitter i en Curtain Wall är det dock mer relevant att skapa en håltagning för hela Curtain Wall:en som kolliderar med en bärande vägg. En annan problematik som upptäcktes var att skriptets första körning placerar ibland ut osynliga håltagningar. Objekten finns i modellen på rätt position med rätt mått men är ej synliga. Vy-inställningar undersöktes i Revit men orsaken hittades inte.

En annan begränsning är om dörrar är under en annan kategori än ”Doors”, till exempel ”Generic Models”. Då kommer skriptet att misslyckas eftersom noden som inhämtar alla dörrar från den inlänkade A-modellen är inställd på att endast söka efter dörrar under kategorin ”Doors”. För att undkomma detta hade skriptet kunnat justeras så att det hanterar dörrar i båda kategorierna. Dock när skriptet utformades gjordes en avvägning att endast hantera dörrar under kategorin ”Doors” för att inte komplicera skriptet då det förutsätts att arkitekten sätter dörrarna under kategorin ”Doors”.

En ytterligare begränsning med skriptet är hantering av förflyttningar av dörrar (A) efter att man kört skriptet och skapat håltagningar i K-modellen. Om en håltagning redan existerar i vägg (K) krävs det att alla befintliga håltagningar tas bort och ersätts med nya genom att köra skriptet igen. Detta kan medföra förluster av metadata satta på objekten. Därav är skriptet mer användbart i tidigare skeden av projekteringsprocessen exempelvis gestaltning eller tidigt i systemskedet då discipliner inte lagrar detaljerad information i objekten i samma utsträckning som detaljskedet.

Skriptets utformning och logik har vissa för- och nackdelar. Skriptet använder sig av BimorphNodes kollisionskontroll som utför kontrollen snabbt jämfört med Dynamos geometriska kollisionskontroll. Bimorph-noden användes i detta fall för att först hitta relevanta element som sedan görs till Dynamogeometrier. Sedan kontrolleras varje vägg med varje dörr. Denna logik medför en nackdel då det resulterar i att skriptet är beroende av tredjepartspaket. Fördelen med logiken är att skriptet undgår en långsam geometriskt kontroll mellan varje dörr och väggeometri samt logiken möjliggör även bortsortering av kollisioner med

tillgänglighetsutrymmen för dörren. För att undgå en längre körtid hade även ett alternativt tillvägagångssätt med kontroller per våningsplan, per väggelement resulterat eller implementering av accelerationsstruktur resulterat i kortare körtider.

5.1.2 Fall 2

Fall 2 automatiserade överföring av parametrar från väggar (A) till pelare (K). Resulterande skript förutsätter att metadata är satt på A-objekten vilket inte alltid är fallet beroende på vilket skede i projekteringen modellen är i. Under förutsättningar att det finns någon metadata i A-modellen går det att föra över det på överlappande element. Logiken för skriptet kan anses som robust då det endast krävs två typer av kolliderande element för att skriptet ska fungera vilket är en fördel då skriptet kan anpassas till mer än pelar-väggkollisioner, till exempel pelar-rumskollisioner. En nackdel med skriptet är dock att om användaren ångrar sin inmatning krävs det att den skapade Shared Parametern måste raderas för att kunna överföra data till samma parametergruppnamn.

5.1.3 Fall 3

I detta fall utvecklades två skript för att placera ursparningar (K) för badrum- och WC-rumsobjekt med överföring av metadata. Skriptet förutsätter att rumsobjekt är grupperade under samma namn samt att relevant metadata finns att överföra. En nackdel med logiken i skriptet är att det förutsätter ett rektangulärt format rumsobjekt vilket inte alltid är fallet. En aspekt av skriptet som både är en för- och nackdel är hur måtten inhämtas. I och med att mått inhämtas och sedan avrundas till närmsta 0,5 mm skapas fler familjetyper än vid en större avrundning. Fördelen med detta är en mer exakt representation av vad Arkitekten faktiskt har modellerat.

Tillvägagångssättet att hämta koordinater och dimensioner från dynamogeometrin i stället för att inhämta dessa från rumsobjektet direkt, bidrar till att skripten kan anses robusta för varierande modeller. Bredd och längd som parameterdata kan ha olika benämningar samt beror på objektets rotation och läge. Att i stället hämta dessa från dynamogeometrier säkerställer att data struktureras och hanteras på samma sätt oberoende av ingående metadatastruktur.

Två olika metodiker programmerades för utplacering av ursparningarna, floor-based och face-based. Fördelen med floor-based är att körtiden blev kortare samt upprepad körning av skriptet resulterade i att ny metadata kunde överföras på samma ursparningar. Nackdelen är att ursparningar endast kan göras på bjälklagets ovansida. Fördelen med Face-based var att ursparningar kan göras på bjälklagets undersida men nackdelen blev en längre körtid och skriptet kunde inte repetitivt köras för att sätta nya parametrar på samma ursparning. Detta visar att de olika utplaceringsmetoderna spelar roll för skriptets effektivitet och bör ta hänsyn till i framtida utformningar av dynamoskript.

5.1.4 Fall 4

I fall 4 kontrollerades pelares (K) position i relation med fönster (A). Resultatet blev två skript, ett som använder sig av en statisk geometri utifrån fönstrets position och ett som använde sig av vektorer för att endast hitta pelare på insidan av fönstret. Skript 1 skulle kunna anses robust då det förutsätter endast att det finns fönstergeometrier placerade i A-modellen. Dock när fönstergeometrierna expanderas i skript 1 expanderas långsidorna båda i riktning in och ut mot byggnaden. Detta gör att den skapade fönstergeometrin kommer att sticka ut genom fönstret beroende på hur vilken längd man väljer att expandera geometrin med. Detta kan anses som en nackdel men i och med att pelare utanför fönstret även fångas medför det upptäckt av pelare som förstör vyn från insida fönster också vilket är en fördel.

Skript 2 fångar endast pelare i och insida fönster. Däremot baseras vektorerna, som hittar insidan, på rumsobjekts namn i A-modellen vilket medför ett högre krav på detaljeringsgrad hos A-modellen.

5.1.5 Fall 5

I fall 5 skapades bjälklag (K) och pelare (K) utifrån A-modellens ytterväggar. En begränsning eller förutsättning för detta skript att våningsplan är etablerade och att ytterväggar är modellerade. Då skriptet använder sig av befintliga pelar- och bjälklagstyper kan detta också anses som en begränsning.

Nackdelar med skriptets logik är ingående vägglayouter och ytterväggens insidas definition. När vägglayouten skapar delade bjälklag, till exempel en genomgående portal blir insida yttervägg mycket svår att definiera. Skriptet programmerades därför för att skapa ett enat bjälklag som kräver handpåläggning, det vill säga manuellt arbete, för att delas in i två. En annan nackdel är att insida yttervägg hämtas ur den största ytan från en differensvolym (se Figur 39.). Detta betyder att vid ett L-format hus med slank utformning kan fel yta användas som insida yttervägg. En fördel med tillvägagångssättet är att det skapas en kontinuerlig kantlinje oberoende av indrag i väggstrukturen medan användningen av väggens centrumlinje medför detta.

Ett problem som stöttes på var tillvägagångssättet för maximala och minimala CC-avståndet för pelare. Vid nuvarande metodik produceras mer komplexa CC-avstånd, till exempel 3847 mm. Ett alternativt tillvägagångssätt hade varit att sätta ut pelare med önskat CC-avstånd och låta en pelare göra avsteg för att det ska gå jämnt ut. Detta hade medfört mer lätthanterliga CC-avstånd.

5.1.6 Övergripande faktorer som påverkar Dynamoskript

Avsnittet diskuterar övergripande faktorer som påverkar Dynamoskript som upptäcktes under fallstudien.

5.1.6.1 Inputs

Vid definition av inputs i Dynamoskript är avgörande att man har ett grepp kring Revits API. Vid inmatning av inputs har hantering av indata i Dynamo stor

betydelse för att skriptet ska fungera. En vanlig källa till att skriptet fallerar är namnge indata felaktigt. Exempelvis om en input missar en versal då kan det leda till att skriptet faller. Ett praktiskt exempel är i Revit är parametergruppen definierad som ”Identity Data”, matar man in ”Identity data” som input kommer Dynamo inte att känna igen den som samma inputvariabel.

5.1.6.2 Listhantering, nivåer & längder på data

Listhanteringen i Dynamo kan vara utmanande då det kan ske i långa komplexa kedjor av operation. I Dynamo kan listor vara hierarkiska och innehålla andra listor på flera nivåer. Hantering av dessa komplexa strukturer kräver en noggrann förståelse för hur man navigerar och manipulerar data inom dessa hierarkier. På grund av dess komplexa struktur kan det vara svårt att korrekt indexera, hantera listor och komma åt rätt element. Felaktig indexering kan medföra att operationer utförs på fel element eller nivå, vilket kan skapa oönskade och felaktiga resultat. Det är viktigt att säkerhetsställa att rätt matchningen av olika listlängder utförs korrekt. För att säkerställa att matchningen utförs korrekt bör rätt läge av lacing vara vald för att styra hur listorna av olika längder interagerar med varandra.

5.1.6.3 Prestanda

När listorna blir långa och flera operationer utförs kan prestandan bli en utmaning. Ju fler operationer som utförs desto längre blir körtiden av skripten. Större modeller innebär fler element och data som måste hanteras och bearbetas. Prestandan varierar över lag beroende på storleken av modellen. För att få en bättre prestanda är det väsentligt att skripten man skapar är optimala. Genom att minska antalet onödiga operationer och använda effektiva metoder för att uppnå samma resultat med färre operationer. Dessutom bör man använda specifika noder som är optimerade för att hantera större listor effektivt. Det är även betydelsefullt att vara medveten om hur man hanterar listor och manipulerar data effektivt på olika nivåer för att undvika onödig komplexitet.

5.2 Övergripande diskussion

I detta avsnitt diskuteras arbetets resultat i syfte att besvara frågeställningarna.

5.2.1 Möjligheter

Frågeställning: *Vad för möjligheter har Dynamo att effektivisera/automatisera processflöden genom optimering eller förädling av data i datautbytet mellan (K) och (A) under projektering?*

Genom fallstudien visar dynamo möjligheter att automatisera processflödet mellan arkitektur och konstruktion. Automatiseringen kan leda till tidsbesparingar då manuellt arbete elimineras vilket bidrar till att konstruktören kan använda arbetstiden för att utveckla alternativa lösningar för beställaren. Automatiseringen kan även bidra till en högre social hållbarhet då den enskilde konstruktören kan använda skript istället för att manuellt projektera monotona och repetitiva arbetsmoment. Tidsbesparingen hade även kunnat bidra till att mer

tid läggs på att hitta lösningar med mindre klimatpåverkan vilket hade bidragit till det ekologiska perspektivet i hållbar utveckling.

Att automatisera processflödet med dynamo bidrar även till konsekventa resultat om skripten rutinmässigt tillämpas. Detta skapar ett värde för discipliner som använder K-modellen som underlag eftersom informationen är konsekvent strukturerad, exempelvis parameteröverföringen i fall 2 eller att alla ursparningar har mått på typnivå i fall 3.

5.2.2 Begränsningar

Frågeställning: *Vilka begränsningar och logiska problem har Dynamo vid en effektivisering/automatisering av processflöden mellan (K) och (A)?*

Projekteringsprocessen i teorin visar hur påverkansmöjligheter minskar under projekteringens gång vilket påverkar användningsområdena för skripten. Under tidigare skeden, exempelvis gestaltning eller systemskedet, innehåller modellen mindre lagrad information vilket gör modellen mindre känslig för informationsförluster vid förändringar. Till exempel vid fall 1, om befintliga håltagningar i K-modellen hade haft detaljerad, lagrad information såsom brandklass eller BSAB-koder, skulle det krävas en avvägning ifall borttagande och nyskapande av håltagningar är en tidsbesparing kontra manuell förflyttning av de befintliga objekten. Alternativt skulle konstruktören investera tid i att skapa ett anpassat skript för förflyttning av befintliga håltagningar. En annan begränsning är den initiala tidsåtgången för att framställa skript.

Gemensamt för alla skript är att det skapas nya objekt och inga befintliga modifieras samt att skripten endast körs en gång under fallstudien. Det vill säga att skapandet av objekt såsom håltagningar, ursparning och bjälklag samt överföring av textinformation utförs endast en gång. Under förutsättningarna att det tillkommer en ny modell från arkitekten måste skripten köras återigen.

Ytterligare en utmaning med att utveckla Dynamoskript är också att hantera abstrakta definitioner geometriskt. Ett exempel på abstrakt definition är begreppet ”insida” vilket hanterades under både fall 4 och fall 5. För att skapa ett skript som kan hantera alla typer av varierande underlag krävs ett mer generellt tillvägagångssätt än vad som användes under fall 4 och 5.

En av de största utmaningarna med att försöka optimera processflöden med hjälp av Dynamo är att få en tillräckligt robust kod. Robustheten av Dynamoskripten är väldigt beroende av vad för underlag konstruktören får. Det beror även på hur komplex hantering av objekt man har. Vid komplexare hantering kan anpassning behöva göras och handpåläggning kan krävas. Desto större och komplexare underlag konstruktören får desto mer komplex blir automatisering då det är fler faktorer att beakta. En annan faktor att beakta är hur många tredjepartspaket ett skript använder. Vid ökat antal är skriptet mer känsligt för uppdateringar för ett av paketen och kan behöva uppdateras och underhållas. Slutligen är listhantering,

nivåer och längder av data en utmaning för att automatisera flödesprocesser mellan arkitekter och konstruktörer.

6 Slutsats

Utifrån de framtagna resultaten kan slutsatsen dras att Dynamo har möjligheter att automatisera och således effektivisera datautbytet mellan (K) och (A) under projektering. Genomförbarheten hos automatisering av flödesprocesser styrs delvis av den lagrade informationen i vederbörande modeller. Vid fler faktorer att ta hänsyn till ökar komplexitet för hantering av varierad underlag. Vidare styrs svårighetsgraden av i vilket skede i projekteringsprocessen skripten används. Optimala skedet för arbetets skript är att dessa körs i tidigare skeden då modellen är mindre känslig för informationsförluster vid förändringar. Slutligen styrs svårighetsgraden av hur stort moment som ska utföras det vill säga storlek och komplexiteten i datastrukturer. Att automatisera en mindre komplex uppgift, såsom ändring av alla gemener till versaler hos tags i Revit är mindre komplext än att automatisera uppgifter där fler faktorer bör beaktas och underlag kan variera. Vid en övervägning om en process ska automatiseras eller inte spelar skalan en avgörande roll. Automatisering innebär vanligtvis en initial investering i tid och resurser för att skapa en automatiserad lösning, vilket kan vara ekonomiskt fördelaktigt eller ineffektivt beroende på omfattningen av uppgiften. Om automatiseringen sker i större skala blir det ofta mer fördelaktigt att automatisera en uppgift.

Resultaten visar att Dynamoskripten kan ersätta manuellt arbete såsom skapandet av objekt utifrån A-modellens geometrier, överföring av metadata eller identifiering av pelare som syns i A-modellens fönster. Ersättningen sker med skript vars körtid är ett fåtal sekunder. Detta kan bidra till tidsbesparing vid återanvändning av skript, mindre risk för mänsklig fel, eliminering av mindre givande och repetitiva uppgifter vilket kan bidra till en effektivare projekteringsprocess mellan konstruktörer och arkitekter.

6.1 Förslag på framtida studier

För att vidareutveckla arbetet och undersöka Dynamos roll inom projektering i byggbranschen, presenteras en punktlista för olika utvecklingsområden och framtida studier.

- En av begränsningarna vid fall 1, 3 och 5 är användandet av externa familjer vid utplacering av exempelvis ursparningar. För att göra skripten oberoende av externa familjer kan skapandet av familjer genom Dynamo undersökas.
- För att ge en god grund för framtida dynamoskript kan definitioner på abstrakta geometriska begrepp programmeras, exempelvis kod som definierar byggnaders hörn, insidor eller rum geometriskt och oberoende av byggnadens utformning.
- Anpassa skripten för fallen till förflyttning av befintliga objekt istället för nyskapande av objekt.
- Programmera en accelerationsstruktur för de olika fallen. Till exempel används kryssprodukt av kollisionskontroller i varje fall. En accelerationsstruktur som sorterar relevanta objekt innan kollisionskontrollen hade optimerat körtiden väsentligt.

- Undersöka automatisering hos processflöden mellan två andra discipliner.
- Programmera instickspprogram i C# för samma fall och jämföra med dynamoskript.

7 Referenser

Ahnler, N & Dahl, G. *Effektivisering av Byggprojektering med hjälp av Grafisk programmering*. Hämtad 2024 05-21 från: https://www.diva-portal.org/smash/get/diva2:1223046/FULLTEXT01.pdf

Akademiska hus. (2024). *Hur går byggprocessen till?* Hämtad 2024-05-13 från: https://www.akademiskahus.se/om-oss/vanliga-fragor/hur-gar-byggprocessen-till/

Autodesk. (2024.a). *Dynamo player.* Hämtad 2024-05-13 från: https://help.autodesk.com/view/RVT/2024/ENU/?guid=RevitDynamo_Dynamo_Player_html

Autodesk. (2024.b). *About Parameters.* Hämtad 2024-04-19 från: https://help.autodesk.com/view/RVT/2024/ENU/?guid=GUID-AEBA08ED-BDF1-4E59-825A-BF9E4A871CF5

Autodesk. (2024.c). *Revit vs. AutoCAD.* Hämtad 2024-05-31 från: https://www.autodesk.com/solutions/revit-vs-autocad

Autodesk. (u.å). *Revit categories and families.* Hämtad 2024-05-13 från: https://www.autodesk.com/learn/ondemand/tutorial/revit-categories-and-families

Autodesk. (2015). *About Family templates.* Hämtad 2024-05-13 från: https://help.autodesk.com/view/RVT/2015/ENU/?guid=GUID-E36987A9-A68F-4121-A391-907306BAA60A

Boverket. (2021a). *Olika skeden i byggandet*. Hämtad 2024-01-26 från: https://www.boverket.se/sv/PBL-kunskapsbanken/teman/ekosystemtjanster/metod_byggande/skeden/

Boverket. (2021b). *I förstudien sätts målen*. Hämtad 2024-02-20 från: https://www.boverket.se/sv/PBL-kunskapsbanken/teman/ekosystemtjanster/metod_byggande/forstudie/

Byggindustrin. (2022). *NCC projektstudion är årets leanbyggare 2022.* Hämtad 2024-02-20 från: https://www.byggindustrin.se/affarer-och-samhalle/affarer-i-byggsektorn/ncc-projektstudion-ar-arets-leanbyggare-2022/

DynamoBIM. (2024). *What is Dynamo?* Hämtad 2024-05-13 från: *https://primer.dynamobim.org/01_Introduction/1-2_what_is_dynamo.html*

Hansson, B. et.al. (2015). *Byggledning – Projektering.* Studentlitteratur AB

Hujdur, A & Karlsson, E. (2020). *En analys av arbetsflödet för 3D-armering mellan Revit och Robot. Hämtad 2024-05-21 från: https://odr.chalmers.se/items/2be5a863-2154-4b21-b25d-d3724a9fc51b*

Microsol Resources. (2021). *What is Revit used for?* Hämtad 2024-05-13 från: https://microsolresources.com/tech-resources/article/what-is-revit-used-for/

Nordstrand, U. (2008). *Byggprocessen.* Liber Ab.

Rhino3D. (2024.a). *Features.* Hämtad 2024-05-31 från: https://www.rhino3d.com/features/

Rhino3D. (2024.b). *Introducing Rhino.Inside.Revit v1.0.* Hämtad 2024-05-31 från: https://www.rhino3d.com/inside/revit/1.0/

Schmied, L. & Strömberg, P. (2019). *Armering i pålfundament - Effektivare byggprojektering med grafisk programmering. Hämtad 2024-05-21 från: http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1352939&dswid=-3143*

Svensk Byggtjänst. (u.å.). *BSAB-för bättre kommunikation*. Hämtad 2024-05-13 från: https://bsab.byggtjanst.se/bsab/om

Tekla. (2024). *Tekla och Trimble.* Hämtad 2024-05-31 från: https://www.tekla.com/se/om/tekla-trimble

Terrol, C. 2020. *Revit families deep dive 2: Element and their classification* Hämtad 2024-05-13 från:

https://globalcad.co.uk/revit-families-deep-dive-2-elements-and-their-classification/#:~:text=When%20it%20comes%20to%20Revit%20families%2C%20there%20are,these%20represent%20the%20bottom%20of%20the%20hierarchy.%20

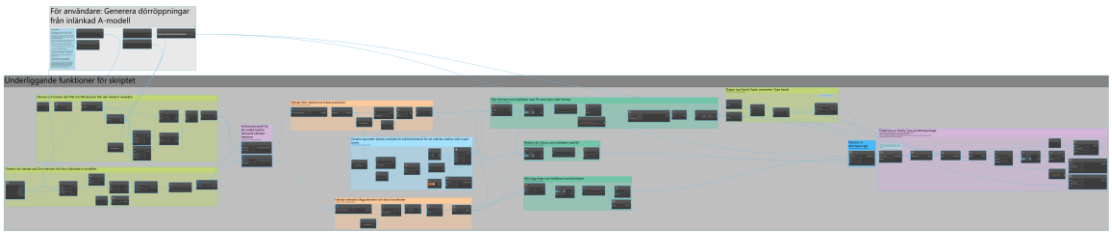
Trimble. (2022). *What is BIM (Building Information Model)*. Hämtad 2024-05-31 från: https://constructible.trimble.com/construction-industry/what-is-bim-building-information-modeling

Wikland, M. & Karlgren, J. (2022). Automatisering av informationshantering inom BIM. Hämtad 2024-05-21 från: https://odr.chalmers.se/items/9e318e3e-52d4-44b2-be2c-ab29fc4d3f39

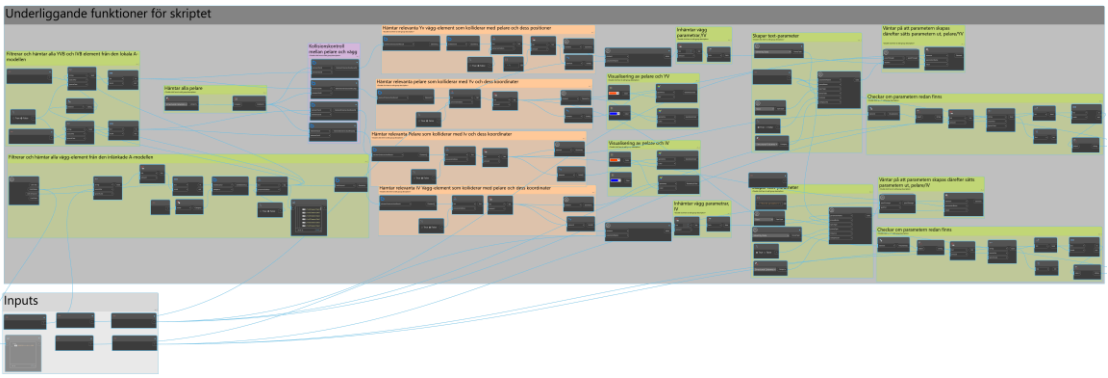
Öberg, R. (2024). *Intervju om VBK:s projekteringsprocess och Dynamo.* Intervjufrågor bifogas i bilagor.

8 Bilagor

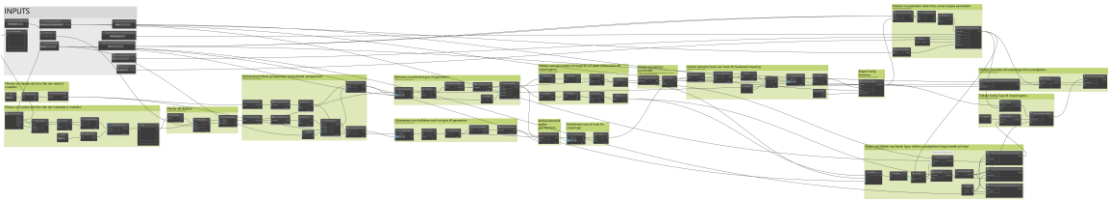
Bilaga 1. Dynamoskript för fall 1



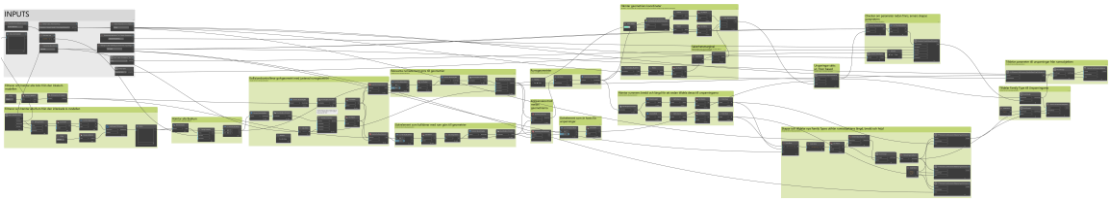
Bilaga 2. Dynamoskript för fall 2



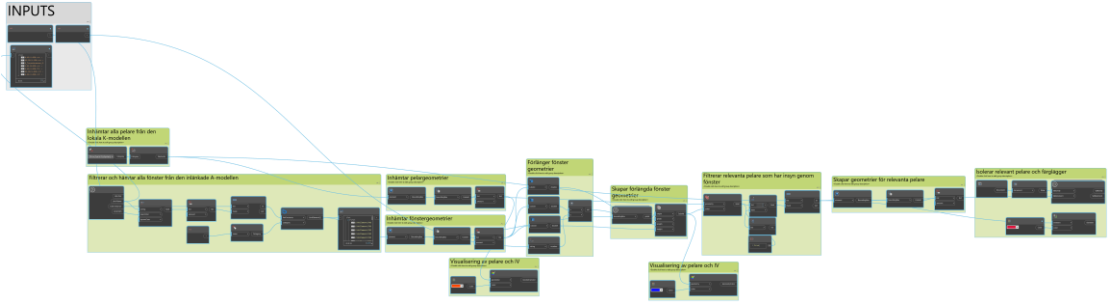
Bilaga 3. Dynamoskript för face-based skript, fall 3.



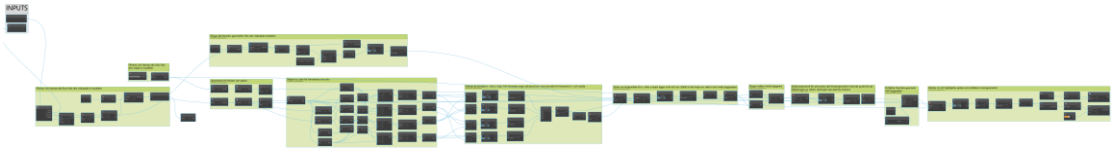
Bilaga 4. Dynamoskript för floor-based skript, fall 3.



Bilaga 5. Dynamoskript för fall 4, skript 1.



Bilaga 6. Dynamoskript för fall 4, skript 2.



Bilaga 7. Dynamoskript för fall 5.

Intervjufrågor:

Introduktion

- Vad heter du och vad är din yrkesroll?
- Hur ser en vanlig arbetsdag ut på VBK?
- Vad är den största utmaningen på ditt jobb?

Projekteringsprocessen hos VBK

- Hur ser projekteringsprocessen ut på VBK?
- Styrs processflödena av projektets typ? Påverkar kontraktsformer processflödena?
- Hur ser informationsutbytet och datahanteringen generellt ut på VBK internt och externt? Kan de variera beroende på projekttyp?
- Vilka flöden upplever du som förbättringsområden?
- Vad finns det, enligt dig, ”hotspots” där dataöverföringen resulterar i informationsförluster? Någon speciell typ av dataöverföring som är överrepresenterad?

Inventering av nuvarande processflöde

- Hur ser processflödena ut i projekteringen kopplat till datahantering i inlänkade modeller?
- Hur ser processflödet ut när dörraröppningar (A), eller plint/ingjutningsgods (K) modelleras?

Dynamo

- Vad använder företaget för metoder och processer för att skapa bättre arbetsflöden? Finns det redan en implementerad Dynamo automatisering i er process?
- I vilken utsträckning används Dynamo på VBK?

- I vilket skede av projekteringen finns det störst möjlighet att implementera automatisering/förädling med Dynamo?
- Vilka kunskaper och kompetenser anser du vara viktiga för att arbeta effektivt med Dynamo?

- Finns det något system eller programvara som kan integreras med Dynamo för att skapa bättre arbetsflöden som ni har använt er utav på VBK?
- Har du haft nytta av att kunna återanvända skript från tidigare projekt? Exempel?
- Vilka fördelar ser du med att använda Dynamo jämfört med andra automatiseringsverktyg? T.ex (Rhino, Grasshopper)

Utmaningar/begräsningar

- Kan informationsförluster undvikas och processflöden förbättras mellan olika discipliner om man automatiserar informationsöverföringen med programmet Dynamo?
- Vad har ni på företaget gjort för att förhindra informationsförluster samt utveckling av förbättrade arbetsflöden?
- Känner du att ni lägger ner tillräckligt med tid på företaget för att förbättra processflöden? Om inte, vad kan förändras?
- Har tekniska uppdateringar någonsin gjort era tidigare skrivna skript ej-fungerande?
- Behöver underlag bearbetas innan de kan användas som indata för dynamo?
- Kan det finnas ”gömda fel” (såsom vägg-väggkollisioner) som är svårare att upptäcka/åtgärda eller är det tvärtom, att många ”gömda fel” kan undvikas?
- Finns det krav på LOD (Level of detail) på modellen och når Dynamo denna nivå eller krävs det oftast handpåläggning?

Automatisering

- Vilka upprepande uppgifter utförs regelbundet under projekteringsprocessen som har potential att automatiseras
- I automatiserade processflöden kan en 100% automatisering vara svårt att uppnå. Vad för typ av handpåläggning brukar vara nödvändigt att göra?
- Vilka lärdomar tar ni med er från VBK från tidigare automationsinitiativ?
- Vad för strategier kan vara tillämpbara på VBK för att utveckla en bredare kompetens bland anställda som är mindre bekanta med automatisering?
- Ser du några svårigheter att involvera intressenter och kunder i diskussionen om automatisering av projekteringsprocessen?
- Vad ser du för möjligheter med att använda Dynamo/andra automatiserings verktyg för att bidra till hållbarhetsinitiativ eller minska miljöpåverkan inom olika sektorer?
- I vilket skede av byggprocessen ser du störst potential för att automatisera arbetsflöden?
- Vilka möjligheter och lösningar tror du att förbättrade och förädlade arbetsprocesser öppnar upp?

- Vilka framsteg förväntar du dig inom automatisering av projekteringsprocessen i framtiden?
- Finns det utmaningar, hinder eller begräsningar som du förutser vid införandet av automatisering i projekteringsprocessen?
- Vad ser du för framtidsmöjligheter för att använda Dynamo/andra automatiserings verktyg i senare skeden? T.ex vid ombyggnad och underhåll?
- Ser du några trender inom automatisering och Dynamo som du tror kommer att vara viktiga framöver?

Alternativa lösningar

- Vad för alternativa lösningar finns för att skapa bättre och förädlade processflöden? Vilka möjligheter öppnar denna lösning upp? Vad är utmaningen med lösningen?

INSTITUTIONEN FÖR ARKITEKTUR OCH SAMHÄLLSBYGGNAD
AVDELNINGEN FÖR CONSTRUCTION MANAGEMENT
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2024
www.chalmers.se

CHALMERS