GRUPPE NR.
14

TILGJENGELIGHET
ÅPEN

OsloMet – storbyuniversitetet

Institutt for Bygg- og energiteknikk – Energi og miljø i bygg
EMTS 3900
Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
Besøksadresse: Pilestredet 35, Oslo

Telefon: 67 23 50 00
www.hioa.no

BACHELOROPPGAVE

BACHELOROPPGAVENS TITTEL	DATO
DIGITALT FORTRINN Effektivisering av prosjekteringsoppgaver innen VVS ved bruk av visuell programmering.	22/05/2018
	ANTALL SIDER / ANTALL VEDLEGG 78/8
FORFATTER Ilgen, Murat Nordeidet, Markus Tørholen, Terje Prestby Økland, Pawel	VEILEDER Hjelseth, Eilif

UTFØRT I SAMMARBEID MED	KONTAKTPERSON
Sweco Norge AS, Oslo	Joakim Myren Svendsen Stian Orvedal

SAMMENDRAG

I denne oppgaven ble det sett på Dynamo, og muligheten dette verktøyet har for å effektivisere VVS prosjektering. Skript har blitt utviklet for dette formålet og det har også blitt sett på hvordan utvikling av egne skript kan bidra til å frigjøre seg fra programutviklere. Gjenbrukhetsgraden av hvert skript til andre prosjekteringsoppgaver er vurdert og det vil bli lagt vekt på synliggjøring av god samhandlingen for å kunne bruke verktøyet effektivt.

Resultatene viser at verktøy som støtter dynamisk modellering har et stort potensial til å effektivisere oppgaver innen VVS prosjektering og at slike verktøy kan redusere avhengighet av systemutviklere. Resultatene peker på at fokus på god datastruktur er essensielt for effektiv utnyttelse av verktøyet. Det er også verdt å nevne at programmer som blir utviklet ikke sirkulerer i det frie markedet og dermed ikke blir tilgjengeliggjort for konkurrenter. Dette skaper et konkurransefortrinn til de bedrifter som har effektive egendefinerte programmer. Teknologien er moden for bruk og bygger på det samme fokuset på bedre samhandling ved innføring av BIM.

3 STIKKORD
Dynamo
BIM
VVS-Prosjektering

Forord

Dette er den avsluttende oppgave for bachelor-/ingeniørstudiet, energi og miljø i bygg, ved OsloMet – storbyuniversitet. Den ble gjennomført vår 2018 i samarbeid med Sweco Norge AS, avdeling Oslo og tilsvarer 20 studiepoeng.

Byggenæringen er inne i et digitalt skifte. Vårt ønske med denne oppgaven er å være med på denne utviklingen, samt bidra til digitaliseringen som nå foregår. Vi vil være med på å effektivisere bransjen og se nærmere på programmeringsverktøyet Dynamo.

Temaet for oppgaven er å kartlegge hvordan man best kan benytte Dynamo som effektiviseringsverktøy for tidskrevende og repetitive oppgaver innenfor VVS prosjektering. Det benyttes programvarer som Revit, Dynamo og Excel knyttet opp mot hverandre for kontroll på dataflyten.

Vi vil takke vår interne veileder, professor Eilif Hjelset, for veiledning og oppfølging. Samtidig vil vi takke eksterne veiledere hos Sweco, Joakim Myren Svendsen (Gruppeleder/ingeniør IKT & Sikkerhet) og Stian Orvedal (Avdelingsleder for tekniske Installasjoner), for videreutvikling av problemstillinger og spissing av oppgaven. Det må videre rekkes en takk til alle intervjuobjekter som har bidratt med både gode innspill på problemområder og tilbakemeldinger på skriptene.

Sammendrag

Byggebransjen er inne i et digitalt skifte, og med innføringen av BIM har mange muligheter for effektivisering blitt presentert. Vi er en gruppe på fire personer som studerer Energi og miljø i bygg ved OsloMet – storbyuniversitetet, hvor noen av oss også studerer BIM ved NTNU. Denne bakgrunnen er grunnen til at vi har valgt å kombinere BIM og VVS i vår bacheloroppgave.

I denne oppgaven ble det sett på Dynamo, som er et visuelt programmeringsverktøy, og muligheten dette verktøyet har for å effektivisere VVS prosjektering. Det har blitt utviklet en rekke mer eller mindre avanserte skript i forbindelse med dette. Hensikten var å utvikle hjelpemidler som kan bidra til effektivisering, men også for å belyse hvor enkelt det kan være, med begrenset programmeringskunnskap, å bruke Dynamo. Videre var tanken å vise at ved gjenbruk av skript så kan en overføre mye av ressursene brukt ved utvikling av skript til nye problemstillinger samt utvikle større og mer komplekse skript etter behov. Det ble også sett på muligheten utvikling av egne skript kan bidra til å frigjøre seg fra programvareleverandører. Videre ble det satt som mål å synliggjøre avhengighetsgraden av samarbeid med andre fagfelt, samt hvilket fokus på kravstilling av riktig informasjon som må til, for å utnytte verktøyet mest mulig effektivt.

Resultatene viser at verktøy som støtter dynamisk modellering har et stort potensial til å effektivisere oppgaver innen VVS prosjektering. Slike verktøy vil også redusere avhengighet av systemutviklere ved at ingeniør har mulighet til å utvikle egne programmer som kan effektivisere en oppgave. Videre synliggjorde resultatene at fokus på god datastruktur er essensielt for effektiv utnyttelse av verktøyet. Kompetanse i bedriften innen slike verktøy er nødvendig for å analysere resultater, avdekke fallgruver samt optimalisere eksisterende programmer. Dynamo krever ikke ekstensiv programmeringskompetanse ettersom det bygger på bruk av visuelle virkemidler og er begrenset til programmet Revit. Det er også verdt å nevne at programmer som blir utviklet ikke sirkulerer i det frie markedet og dermed ikke blir tilgjengeliggjort for konkurrenter. Dette skaper et konkurransefortrinn til de bedrifter som har effektive egendefinerte programmer.

Teknologien er moden for bruk, og bygger på det samme fokuset på bedre samhandling ved innføring av BIM. Ved overgang fra statisk til dynamisk BIM kan man trekke samme paralleller som til overgangen fra CAD til BIM i forhold til implementering av nye arbeidsmetoder samt oppnåelse av effektiv bruk i forhold til kompetansesatsing.

Innhold

1 Leveranse

Denne rapporten er en del av en total pakke. Totalleveransen vil bli levert på minnepenn og inneholder følgende elementer:

Rapport
Skriptfiler
Testmodeller klargjort for kjøring av skript
Excel maler for skriptene
Oversiktsbilde i PDF over hvert enkelt skript
Presentasjonen i form av Power Point
Introduksjonsvideo av temaet som vist i presentasjonen
Videoer over utvalg av skript

Denne rapporten er den skriftlige delen av bacheloroppgaven. Det er lite hensiktsmessig å levere kun et skriftlig dokument når en har arbeidet med skriptutvikling. Derfor blir også skriptfilene levert som en del av besvarelsen. Leveransen av skriptene er ment som et supplement til den programtekniske delen. Det blir også levert et PDF dokument til hvert skript som inneholder et oversiktsbilde av skriptene, med mulighet for å zoome inn på ønskede områder. Modeller og Excel maler vil også bli levert for test av skriptene.

Power Point presentasjonen vil også bli overlevert. Presentasjonen består også av en video som skal fungere som en introduksjonsvideo av oppgaven som også følger med leveransen. Presentasjonen og introduksjonsvideoen skal ikke distribueres videre på grunn av kopirettslige grunner.

2 Oppbygning

Denne rapporten er delt opp i to hoveddeler grunnet tilpasning til ulike målgrupper.

Del 1 er hoveddokumentet, rettet mot *ledelsen*, hvor det store overblikket blir presentert og tar for seg diskusjonen rundt nødvendige arbeidsprosesser for effektiv bruk av automatiseringsverktøy som Dynamo. Skriptene blir presentert på overordnet nivå, hvor resultat, bruksområde og begrensninger skal være i hovedfokus. Strukturen på denne delen er basert løst på artikkelen *«Positioning and presenting design science research for maximum impact»* [1]

Del 2 er det programtekniske rettet mot *ingeniøren,* hvor vi ser nærmere på oppbygning av skriptene og metode for programmeringen. Dette er ment som en brukerveiledning for videre bruk og utvikling av skript. Det blir presentert en grundig beskrivelse av innhold og oppbygning av de ulike deler av skriptene, del for del, node for node, på et høyere detaljeringsnivå.

Del 1 – Organisatorisk

3 Innledning

I denne delen av rapporten gjennomgås det generelt rundt valg av tema, fokusområde, hva som er ønskelig å undersøke og hvordan dette er utført. Dette vil bli presentert som bakgrunn, problemstilling, mål og begrensninger.

3.1 Bakgrunn

Byggebransjen er inne i et omfattende digitalt skifte, noe som endrer måten vi jobber på, og vil jobbe på i fremtiden. Andre næringer som eksempelvis fly- og bilindustrien har benyttet digitalisering og automatisering i produksjon og utvikling av produkter i flere tiår, og nå kommer byggenæringen etter, med stormskritt. Dette skiftet byr naturligvis på mange muligheter, men også problemstillinger man må ta hensyn til i overgangen fra tradisjonelle til nye prosesser.

Bruk av BIM er på god vei til å være veletablert i de fleste store selskap som driver med VVS prosjektering per dags dato, men det handler naturligvis ikke bare om å lære seg å bruke tilgjengelige BIM verktøy, men hvordan man bruker verktøyet effektivt. Hensiktsmessig bruk vil resultere i økt effektivitet for både bedriften og bransjen. Etablering av god modelleringskultur og kompetanse i bruk av verktøyet er en ressurskrevende prosess som fører til at man ikke merker gevinst av BIM, før de ansatte i bedriften er fortrolige med verktøyet og klarer å benytte det på en effektiv måte. [2, p. 255]

Implementering av BIM verktøy til prosjektering av VVS løsninger i bransjen har ført til at en rekke prosesser er effektivisert, samtidig som bruk av teknologien setter lys på delprosesser som er tidskrevende. Videre fokus er da på hvordan man kan redusere tiden det tar å utføre tidskrevende delprosesser for å utnytte BIM maksimalt.

For å få til dette, er det mulig å benytte seg av programmeringsverktøy ved prosjektering, slik at brukeren har muligheten til å utvikle egne programmer hvor man kan løse repetitive og tidskrevende arbeidsoppgaver. Et eksempel på et slikt verktøy er Dynamo, som denne oppgaven retter fokuset mot.

«*På generell basis er fordelen med Dynamo at det er noen konstruksjoner og geometrier som nesten ikke er praktisk gjennomførbart å modellere uten Dynamo. I tillegg gir Dynamo muligheter for bedre optimalisering da man kan parametrisere modellen og gjøre den mer fleksibel med tanke på endringer.*»

Sivilingeniør Andreas Lyngtveit Lindland, Norconsult

Avhengighetsgraden av programvareutviklere innen BIM verktøy har vært stor ved bruk av de ulike verktøy som leveres. Ved manglende funksjoner har man tidligere ventet på at en eventuell løsning skal bli utviklet og levert av programvareutviklere, og denne avhengigheten eksisterer i stor grad per dags dato. Dynamo gir bedriften mulighet til å redusere denne avhengigheten hvor brukere med begrenset programmeringskunnskap har mulighet til å lage programmer som løser en rekke oppgaver.

Kunnskapen om bruk av slike verktøy til å effektivisere prosesser i prosjekteringen, er relativ liten. Byggebransjen er en konservativ bransje, hvor man ikke trenger å gå mange år tilbake før selve

bruken av BIM var noe nytt i arbeidsmetodene hos de prosjekterende. Dermed er krav til digital kompetanse ved valg av nyansatte i fokus, som i hovedsak gjelder teknisk tegning og kjennskap til BIM. Dette har flere av gruppemedlemmer erfart og videre brukt til sin fordel ved jobbsøking og nettverksbygging med bransjen. Samtlige av gruppens medlemmer har valgt å bygge på utdanningen sin med BIM årsstudium på NTNU som følge av det tydelige fokuset på digital kompetanse blant nyutdannede. Tilegnet kunnskap fra begge utdanninger er grunnlaget for denne oppgaven.

3.2 Problemstilling

Oppgaven sitt fokus er parametrisk behandling av informasjon i BIM, ved bruk av et visuelt programmeringsverktøy. Det legges vekt på prosjekteringsprosesser som kan effektiviseres innen fagfeltet VVS. Det er mange funksjoner i Revit som er tilgjengelige for de prosjekterende. Derimot er det en rekke prosesser som fremdeles er tidskrevende og repetitive grunnet manglende eller ufullstendige funksjoner. Alternativet i slike tilfeller er å utføre manuell input eller betale leverandør av tjenester, eksempelvis NTI CAD, som utvikler et program som løser den aktuelle problemstillingen. Å vente på at leverandører skal utvikle et program som løser problemer, tar derimot tid og medfører en kostnad for bedrifter. Videre kan det være at det spesifikke problemet som skal løses er unikt for prosjektet som medfører at program som blir utviklet har mindre sannsynlighet for gjenbruk.

Bruk av Dynamo bør ikke være noe nytt for byggingeniører eller arkitekter, ettersom verktøyet er brukt i hovedsak til å manipulere geometri. Gruppen skal derimot undersøke hvordan man kan benytte verktøyet til å manipulere annen form for data i modellen. I VVS prosjekteringen er det mye data som behandles som ikke krever manipulering av geometri, og ettersom verktøyet sin hovedstyrke ligger i data-/listebehandling, vil gruppen undersøke hvordan man kan utnytte denne styrken til å effektivisere prosesser i VVS prosjekteringen.

3.3 Mål

Gruppen sitt mål er å synliggjøre *hvordan* man kan: benytte programmer som Dynamo til å løse problemer relevante for VVS prosjektering, redusere andel begrensede funksjoner, øke effektiviteten samt redusere avhengighet av programvareutviklere.

Resultatene skal synliggjøre avhengighetsgraden av samarbeid med andre fagfelt, samt hvilket fokus på kravstilling av riktig informasjon som må til for å utnytte verktøyet mest mulig effektivt. Synliggjøring av viktige faktorer som sikrer effektiv bruk skal danne oversikt for ledelse over hvordan informasjon må struktureres for å effektivisere sin drift ved bruk av et slikt verktøy.

3.4 Begrensninger

De utvalgte skriptene er utviklet basert på konkrete problemstillinger fra prosjektingeniører i Sweco sin VVS-avdeling i Vækerø. Skriptene som ble utviklet ble testet på enkle testmodeller modellert av gruppen sine medlemmer. Etter bekreftet funksjon ble skriptene brukt på BIM modeller fra Sweco sine prosjekter, for å undersøke funksjon på modeller med større kompleksitet. Resultater fra tester på prosjektspesifikke modeller blir ikke synliggjort i dette dokumentet, grunnet konfidensiell informasjon fra prosjekter som ikke ønskes å offentliggjøres. Evalueringen blir derimot tatt med i diskusjonen.

Disponibel tid til utføring av oppgaven var på underkant av fem måneder. Det er en rekke problemstillinger som kan gjennomføres mer effektivt ved bruk av koding med programmeringsspråket Python innad i Dynamo. [3] Gruppens medlemmer skal da i løpet av den

tilgjengelige tiden lære seg å bruke Dynamo som verktøy, samt lære Python programmering på et enkelt nivå. Kunnskapen skal videre brukes til å løse problemstillinger som har blitt kartlagt ved intervjuer av prosjektingeniører hos Sweco. Noe arbeid ble utført i forkant av prosjektet i forhold til programmering med Python. Derimot hadde ikke gruppen kapasitet til å lære seg å bruke Python i Dynamo på et avansert nivå da dette innebærer ekstensiv forståelse av de ulike databasene i Revit [4] samt logikken i programmeringsspråket, som er et eget studie.

4 Faglig rammeverk

I dette kapitlet presenteres de ytre forhold som er relevante og som setter den faglige rammen til dette prosjektet. Produktene som er utviklet, tar stilling til en rekke prosesser hvor gruppen har benyttet seg av IDM standarden for utvikling av prosesskart. Andre standarder relevante for fagfeltet VVS er også benyttet i utviklingen. Beskrivelse av de standardene gruppen har tatt stilling til i utformingen av denne oppgaven, er gitt nedenfor.

4.1 NS-3031:2014

Beregning av bygningers energiytelse – Metode og Data

Standarden omfatter tre beregningsalternativer av bygninger energiytelse: - månedsberegning (stasjonær metode); - forenklet timeberegning (dynamisk metode); - detaljerte beregningsprogrammer (dynamisk metode). [5]

4.2 SN/TS 3031:2016

Bygningers energiytelse – Beregning av energibehov og energiforsyning

Teknisk spesifikasjon som omfatter regler for beregning av bygningers energibehov, behovet for levert energi til bygningen og eksportert energi til energinettet for dynamiske beregningsprogrammer validert etter NS EN 15265. SN/TS 3031:2016 er et supplement for helhetlige energiberegninger for bygg og energiforsyningssystemer, men erstatter ikke standarden NS 3031:2014. [6]

4.3 NS-EN 12831-1:2017

Bygninger energiytelse – Metode for beregning av dimensjonerende effektbehov til varme- Del 1: Effektbehov til oppvarming

Denne europeiske standarden dekker metoder for beregninger av effektbehov på romnivå, hvor effektbehovet er definert som den varmen som kreves tilført rommet for å opprettholde den temperaturen som er ønsket basert på utetemperaturen. [7]

4.4 NS 3055:1989

Dimensjonering av ledninger for vann- og avløpsanlegg i bygninger

Standard som fastsetter regler for dimensjonering av vann- og avløpsledninger i bygninger og stikkledninger, som bygger på Normalreglement for sanitæranlegg med SBI-nomogram utarbeidet av det danske Statens Byggeforskningsinstitut. [8]

4.5 ISO 29481-1:2016

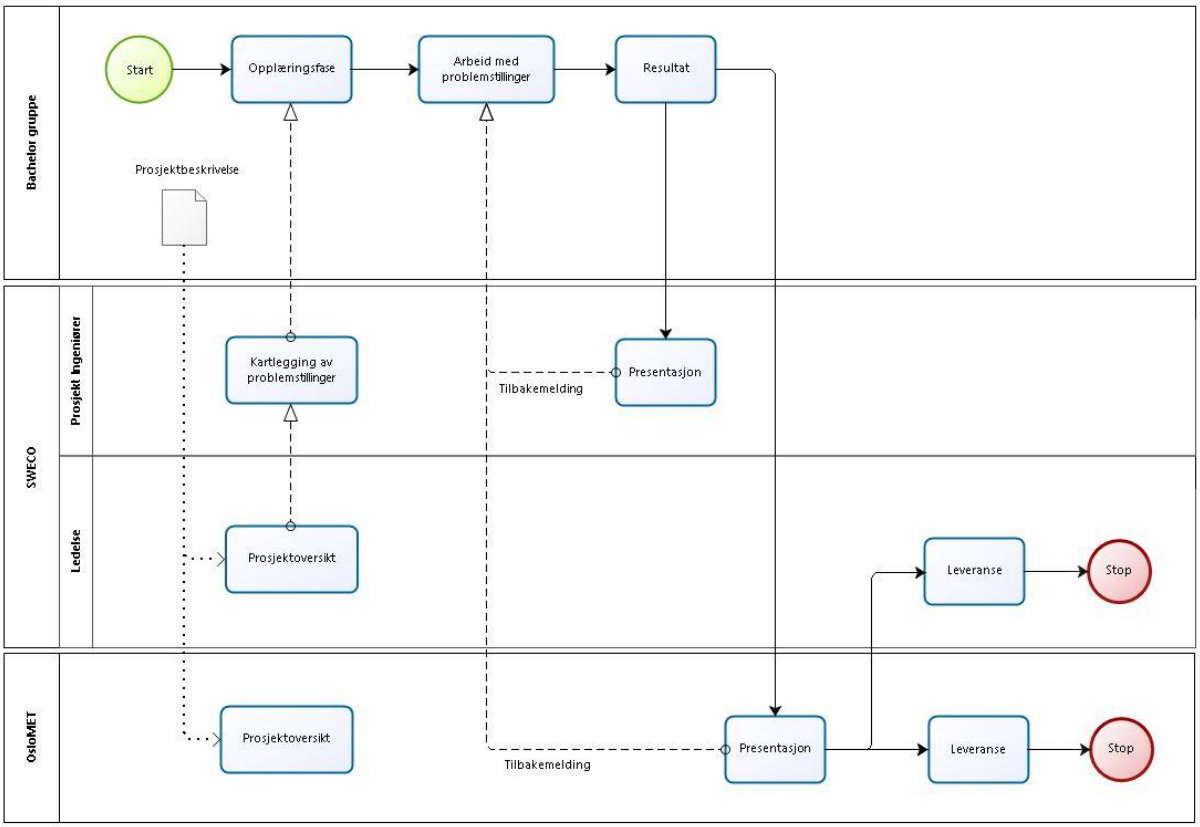
IDM standarden

IDM står for Information Delivery Manual, som definerer retningslinjer og prinsipper for å beskrive krav til leveranser av informasjon til/fra bygningsinformasjonsmodeller (BIM). IDM-standarden er grunnleggende både for programvareutvikling og for bruk av BIM i prosjekter, eller som sikring av kvalitet på informasjonsutvekslinger. [9]

5 Metode

Her presenteres de ulike metoder for utføring av arbeid, for å videre kunne tilnærme seg de mål som er satt i forkant av oppgaven. Det vil bli forklart kort og generelt om ulike metodene, samt hvordan disse er benyttet og presentert i oppgaven.

IDM standarden har blitt benyttet for utviklingen av egne prosesskart som er presentert i denne oppgaven. Det har blitt sett på som hensiktsmessig å dele opp arbeidsprosessen i løpet av prosjektet i flere faser, videre definert som; opplæring, arbeid med problemstillinger og testing av resultater. *Figur 1* illustrerer hele arbeidsprosessen gjennom prosjektets forløp.



Figur 1 – Prosessforløp for arbeid med prosjektet

5.1 Forberedelser

Grunnet oppgavens omfang begynte gruppen med planlegging i høstsemesteret 2017. Dette ble

utført for å få en generell forståelse for Dynamo samt en grunnforståelse for bruken av programmet. For oversikt over fremdriften til gruppen henvises det til *Vedlegg I*

5.2 Dynamo workshop & Python

Samarbeid med andre brukere av Dynamo har også vært med å øke forståelsen for bruken av verktøyet. Gruppen har brukt Dynamo.org forumet på nettet, hvor det er mange eksperter som kan gi råd rundt problemstillinger. Videre har gruppen også holdt workshop sammen med student fra NTNU som også skriver om Dynamo, da i forhold til hovedoppgave i årsstudiet BIM.

Python er vesentlig for å kunne tilpasse egne noder, og en forståelse for dette programmerings-språket ble sett på som nyttig for arbeidet med egne skript. Det ble derfor arbeidet med et online programmeringsskurs for Python gjennom webområdet Code Academy [10]. Dette for å få et fundament for å prøve å utvikle egne Python noder som er ofte brukt i egne tilpassede noder. Også under opplæringen av Python ble det brukt opplæringsvideoer fra Youtube.

5.3 «Learning by doing»

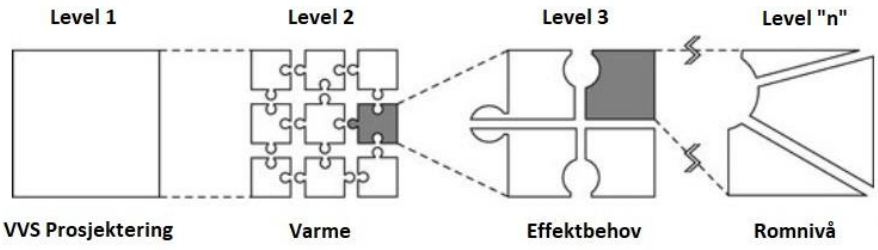
Learning by doing er et slagord hvor tanken er at læring skal være både relevant og praktisk og ikke bare passiv og teoretisk. [11] Programmering er noe som må praktiseres. I opplæringsfasen har denne metoden vært essensiell for å forstå de ulike nodene i Dynamo samt hvordan man hensiktsmessig kobler opp nodene ved logikk for å få frem ønsket resultat. Tilsvarende har metoden vært viktig for å benytte bruk av Python i skriptutviklingen hvor man har en formening om hvordan man vil sette opp logikken i en kode.

5.4 «Trial and Error»

Trial and error, baserer seg på prøve og feile. I møte på et problem, så kan en ved gjentatte forsøk redusere feil og tilslutt eliminere de. På denne måten lærer en seg å overvinne problemet. [12] Prøve og feile forbindes ikke bare med læring, men også som metode for å løse problemer. I utviklingen av skriptene har denne metoden vært sentral for å komme frem til ønsket resultat. Ulike løsninger på filtrering av informasjon har blitt benyttet til å oppnå ønsket løsning hvor valget faller på den mest robuste funksjonen.

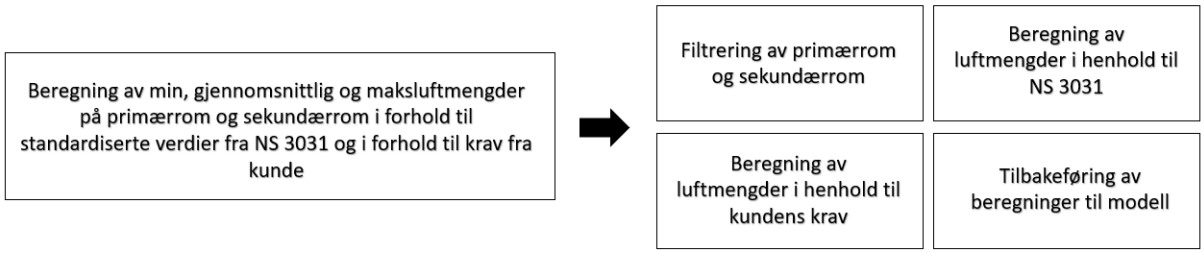
5.5 «Work breakdown structure»

Work breakdown structure baserer seg på å bryte ned større oppgaver i mindre biter, for å gjøre oppgavene mer overkommelige. Ved å bryte de ned kan en lettere se hvilke oppgaver som må løses for å oppnå det endelige målet, og det kan være lettere å delegere arbeid. Et prosjekt kan deles opp i flere deler, som igjen kan deles opp i mindre biter.



Figur 2 - Work breakdown structure, redigert av forfatter [13]

De fleste skript består av delprosesser som må ha en robust funksjon for at det totale produktet skal fungere optimalt. Skriptet blir gruppert hvor det legges fokus på funksjon i hver delprosess. Når funksjonalitet oppnås kobles delprosessene sammen til man tilslutt har det ønskede resultatet. *Figur 3* illustrerer hvordan denne metoden ble benyttet på utvikling av *skript 02 – Luftmengdeberegning.*



Figur 3 – Det totale produktet avhenger av delprosesser

5.6 Intervjuer

Intervjuformen benyttet faller inn under kvalitativintervju. Den kvalitative intervjuformen baserer seg på ustrukturerte intervjuer hvor intervjuet baserer seg på en emneliste over temaer samtalen skal dreie seg om. [14] I denne oppgaven har ingeniører ved Sweco blitt intervjuet for å diskutere de problemstillinger som de mener det er muligheter til å effektivisere. **[Vedlegg II]**

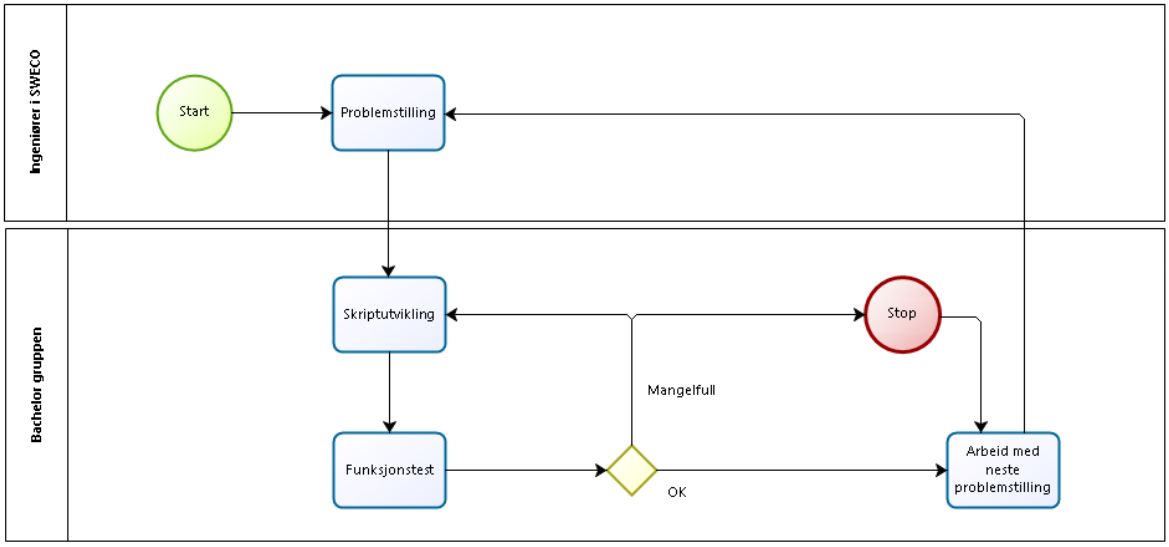
6 Litteratur- og casestudie

Her presenteres det kort hvilke ulike tilnærminger som er utført for å tilegne seg kunnskap og erfaring med programmet Dynamo, samt hvordan problemstillingen ble spisset og definert etter samtaler med personell i Sweco.

6.1 Kartlegging av problemstilling

Det ble gjennomført intervjuer med personell fra Sweco, hvor utgangspunktet var å finne ut hvilke oppgaver de mente var tidskrevende under prosjektering, og som kan effektiviseres. Dette for at en kunne implementere noen av disse i utarbeidelsen av vår problemstilling. Ettersom bedriften har ingeniører med erfaring innen bruk av BIM verktøy kunne vi kartlegge reelle problemstillinger som vi som studenter ikke var klare over.

For å utvikle skript er det ofte hensiktsmessig å se på hva som allerede er blitt gjort og som er tilgjengelig. Dette kan ofte brukes som et utgangspunkt. Skriptene som blir brukt som utgangspunkt, er ment som en informasjonskilde til noder en kanskje ikke var klar over, og i mange tilfeller har en oppdaget pakker med forskjellige noder en kan få bruk for i sine egne skript. Skript brukt som utgangspunkt vil bli referert til i resultatdelen der det er aktuelt. *Figur 4* beskriver utviklingsprosessen av skript.

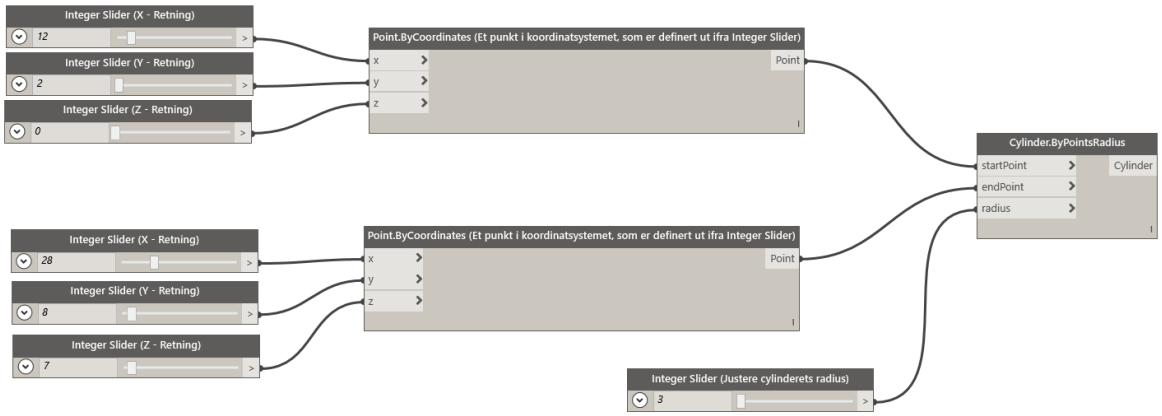


Figur 4 – Prosesskart som beskriver arbeid med skriptutvikling

7 Dynamo og visuell programmering

Dynamo er et visuelt programmeringsverktøy. Programmet er en utvidelse av Revit, utviklet av Autodesk. Konvensjonell programmering og visuell programmering, har samme mål, som er å gi brukeren mulighet til å definere en rekke algoritmer. Dette kan gjøres ved tekstkode eller ved bruk av visuelle virkemidler. I Dynamo er dette noder som er definerte og utfører en rekke med kommandoer. Eksempel i *Figur 5* og *Figur 6* viser et program som danner en sylinder i rommet ved bruk av koding med tekst og ved visuelle virkemidler i Dynamo. Gruppen benyttet Dynamo versjon 1.3.2.2480 ved utvikling av skriptene i oppgaven.

Visuell Programmering



Figur 5 – Koding som danner en sylinder i rommet ved bruk av visuell programmering i Dynamo

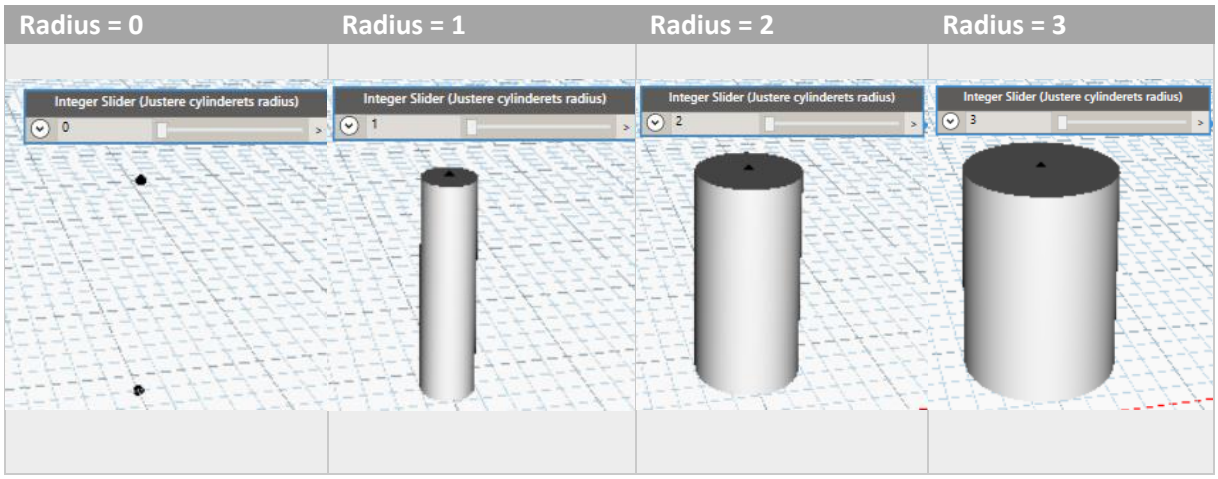
Koding med tekst

```
xpoint_start = 12; // Startpunktets koordinat i x er 12
ypoint_start = 2; // Startpunktets koordinat i y er 2
zpoint_start = 0; // Startpunktets koordinat i z er 0
point1 = Point.ByCoordinates(xpoint_start, ypoint_start, zpoint_start); // Noden danner et punkt ut ifra ønsket x, y og z koordinater, som er definert av nodene Integer Slider. En integer slider fører til justerbare heltall.
xpoint_end = 28; // Endepunktets koordinat i x er 28
ypoint_end = 8; // Endepunktets koordinat i y er 8
zpoint_end = 7; // Endepunktets koordinat i z er 7
point2 = Point.ByCoordinates(xpoint_end, ypoint_end, zpoint_end); // Noden danner et nytt punkt ut ifra ønsket x, y og z retning, som er definer av nodene Integer Slider. Avstanden mellom punkt 1 og 2 er da gitt i millimeter.
radius = 5; // Ved å benytte seg av noden Integer Slider kan man justere radiusen til sylinderet, samt se resultatet i bakgrunnen.
cylinder1 = Cylinder.ByPointsRadius(point1, point2, radius); // Her blir sylinderen dannet.
```

Figur 6 – Koding med tekstforklaring

Parametrisk kraft

Bransjen benytter hovedsakelig statisk modellering i prosjekteringsarbeidet sitt per dags dato. Bruk av Dynamo baserer seg på modellering basert på algoritmer og er i større grad dynamisk. Integer Slider er et eksempel på en kraftig node hvor lengden kan justeres, samt figuren omformes, se *Figur 7* for illustrasjon. Figuren viser at ved en enkel justering ved bruk av noden, kan en endre radiusen til for eksempel en søyle. I praktisk sammenheng kan en tenke seg et prosjekt med mange søyler, hvor endring i ønsket radius enkelt kan justeres.



Figur 7 - Parametrisk kraft til noden Integer Slider

7.1 Dynamo - Fagstoff

Rene opplæringsbøker i Dynamo eksisterer ikke da Dynamo er i konstant utvikling. Derfor var det fornuftig å se på alternative opplæringsmetoder for å kunne tilnærme seg de problemstillinger som var blitt satt i forkant av oppgaven. Følgende er en beskrivelse av de forskjellige tilnærmingene.

Dynamo Primer [15], som er en nettbasert veiledning til Dynamo, fungerer som et springbrett for Dynamo verdenen. Den inneholder korte beskrivelser til emner som inngår i Dynamo, for eksempel noder og pakker, som Dynamo er bygget opp av. Dette er ment som en introduksjon og for å gi et grunnlag for videre arbeid med forståelsen og arbeidet med Dynamo. En av de større fordelene med Dynamo primer er listen som beskriver de mest populære pakkene til Dynamo og som kan gjøre arbeidet lettere med egne skript. Dynamo primer anbefales derfor som en introduksjon for emnet.

Bruken av opplæringsvideoer er en stadig mer brukt metode for å tilegne seg kunnskap om forskjellige temaer, hvor kanskje Khan Academy [16] er det mest kjente eksemplet for hvordan slik pedagogikk er i praksis. Autodesk har utviklet webområdet dynamobim.org [17] som er et webområde ment som en ressurs og læringssenter for brukere av programmet, og inneholder mange opplæringsvideoer laget av eksperter. Disse fungere fint på et grunnleggende plan, og viser eksempler på bruk og utvikling av skript, som er nyttige for nye brukere for å bil kjent med programvaren. Det er ikke bare på dynamobim.org en kan finne opplæringsvideoer. Blant annet har flere eksperter lagt ut videoer på Youtube, hvor en har muligheten til å aktivt søke etter videoer som ligger nærmere de problemstillinger det arbeides med.

Det å lese seg opp og få en fundamental forståelse for Dynamo og opplæringsvideoer for å øke forståelse er vel og bra, men for å virkelig kunne anvende Dynamo må en arbeide med programmet. «*Learning by doing*» er en glimrende metode for å nærme seg mestring av Dynamo og derfor har mye av opplæringsfasen gått med på å prøve og lage egne skript på mer begrensede problemstillinger, enn de som er presentert i denne oppgaven.

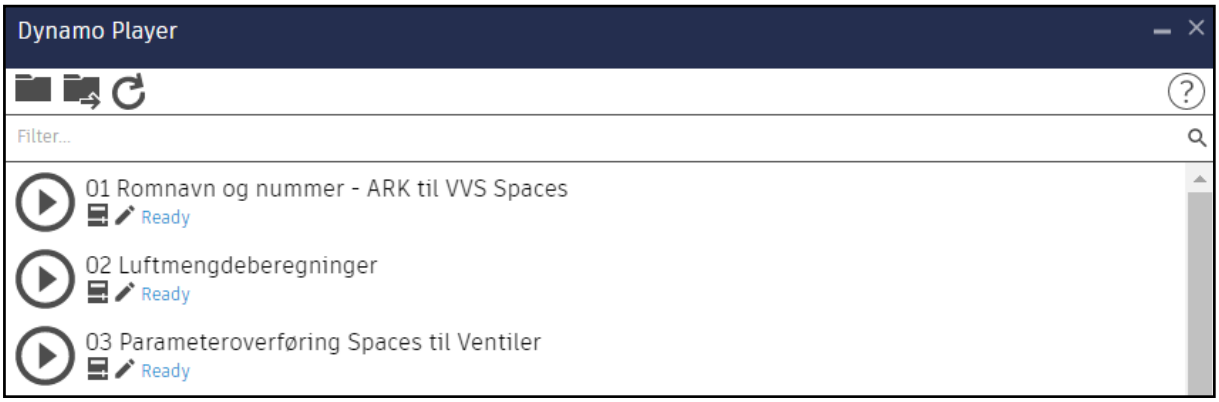
7.2 Dynamo Player

Det er viktig at de skriptene som blir utviklet har brukervennlighet for prosjektingeniører som skal benytte de i prosjekter. Arbeid i selve skriptet kan være krevende for brukere med begrenset kunnskap til programmet. Dette gjør at mange spørsmål dukker opp som stjeler tid i prosjektet.

«Når folk åpner Dynamo og ser den jungelen av koder, så får de migrene og går hjem og gråter i dusjen»

Håvard Vasshaug – Siv.Ing i bygg samt BIM og programmeringsutvikler i Bad Monkeys [18]

Dynamo Player er et tillegg i Revit som gir brukeren et enkelt brukergrensesnitt ved bruk av de utviklede Dynamo skriptene. *Figur 8* nedenfor gir et eksempel om hvordan skriptene blir vist i Dynamo Player.



Figur 8 – Dynamo Player

Dersom man trykker på symbolet blir man tatt videre til inputverdier i Dynamo skriptet. Eksempelet nedenfor er *Skript 02 - Luftmengdeberegninger* og viser deler av de input som er definert for skriptet.

Figur 9 – Dynamo Player viser deler av input som er definert for skriptet

Det er allikevel noen tilfeller der hvor man er avhengig av å gå inn i selve skriptet for å utføre en oppgave. Da har man muligheten i Dynamo Player til å trykke på ikonet som åpner opp selve skriptet i Dynamo.

7.3 Kvalitetssikring av skript

Som en del av kvalitetskontrollen av arbeidet vi har gjort, har vi kjørt ulike skript på forskjellige testmodeller som vi selv har modellert. Bakgrunnen for kvalitetssikring av skript er å evaluere mangler ved skriptet eller kartlegge feil, for så å endre på disse. Feil som ble kartlagt underveis ble videreformidlet til skriptutvikler, som en del av prosessen med optimalisering frem mot leveranse.

8 Resultat

I denne delen vil hvert skript bli kort oppsummert slik at en kan få et innsyn i hva skriptet gjør og hvilke effektiviseringsmuligheter en kan oppnå. Det vil videre beskrives kort hvordan skriptet fungerer, hvilken avhengighet skriptet har til modell, hvilke resultat som kan forventes og evaluering av hvert skript.

Det vil ikke bli redegjort i dybden hvordan skriptene fungerer. Det henvises til Del 2 – Programteknisk, for en grundigere gjennomgang av skriptene.

8.1 Luftmengder

Følgende er en rekke av skript som tar stilling til ulike faser av prosjekteringen med tanke på luftbehandling. Prosessforløpet er illustrert i *Figur 100*.

Figur 10 – Prosessforløpet for luftbehandling

Skriptene har stor grad av avhengighet i forhold til hverandre, noe som gjør at en høy grad av datastruktur i modellen gir gode forutsetninger for automasjon av delprosesser. En lav grad av datastruktur derimot kan medføre at noen delprosesser ikke fungerer optimalt, som igjen medfører større andel med manuelt arbeid. Dermed er det viktig at parametere i modellen og objekter er vel definerte.

RIV kan utnytte stor grad av automasjon dersom relevant info fra ARK er tilstede i sin modell. Fokus på hvilken informasjon som skal tildeles til hvilken tid i prosjektet, vil gi prosjekterende gode forutsetninger til å utnytte disse skriptene.

8.1.1 Skript 01 – Overføring av navn og nummer fra ARK modell til VVS modell

Beskrivelse

Skriptet tar for seg overføring av riktig navn og nummer fra rommene i arkitekt modellen til «Spaces» basert på samme geometrien i VVS modellen. Meningen med skriptet er å spare tid ved å slippe dobbeltarbeid med å plotte samme informasjon inn flere ganger. I tillegg slipper en forvirring med ulikheter i navnsettingen på «Spaces» i forhold til «Rooms».

Avhengighet

For at skriptet skal fungere må arkitekt ha plassert ut rom i sin modell, med riktige avgrensninger.

Resultat

Følgende figur er resultatet av skript, med oppdaterte navn og nummer.

Før skriptet kjøres				Etter skriptet har blitt kjørt			
01 Etasje	Space	1	29.76 m²	01 Etasje	Kontor	1	29.76 m²
01 Etasje	Space	2	22.13 m²	01 Etasje	WC	2	22.13 m²
01 Etasje	Space	3	36.00 m²	01 Etasje	Fellesrom	3	36.00 m²
01 Etasje	Space	4	26.76 m²	01 Etasje	WC	4	26.76 m²
01 Etasje	Space	5	23.69 m²	01 Etasje	Kontor	5	23.69 m²
01 Etasje	Space	6	48.76 m²	01 Etasje	Resepsjon	6	48.76 m²
01 Etasje	Space	7	31.86 m²	01 Etasje	Lager	7	31.86 m²
01 Etasje	Space	8	31.15 m²	01 Etasje	Lager	8	31.15 m²
01 Etasje	Space	9	47.67 m²	01 Etasje	Gang	9	47.67 m²
01 Etasje	Space	10	23.16 m²	01 Etasje	Kontor	10	23.16 m²

Figur 11 – Resultat av Skript 01, overføring av navn og nummer

Evaluering

Skriptet oppdaterer alle de aktuelle parametere i forhold til endringer utført i ARK modellen og kan benyttes fortløpende når endringer gjennomføres. Skriptet er robust, hvor eneste avhengighet er at arkitekt har definert sine rom i modellen.

8.1.2 Skript 02 - Luftmengdeberegninger

Beskrivelse

Formålet med dette skriptet er å beregne maksimum, minimum og gjennomsnittlige luftmengder, og tilbakeføre disse verdiene tilbake til «Spaces» under definerte parametere.

Brukeren har mulighet til å velge egne verdier for person og materialbelastning. Videre kan bruker velge egen bygningskategori som påvirker persontettheten, eller velge egne verdier for persontetthet samt minste tilstedeværelse.

Antall personer blir beregnet i forhold til standardiserte verdier for persontetthet med gitt bygningskategori i henhold til tabell H.2 i NS 3031 [5].

Bygningskategori	Minste primær-areal	Største sekundær-areal	Minste persontetthet, primærareal m²/person	Minste tilstedeværelse, primærareal [b]
Småhus	– [a]	– [a]	– [a]	100 % [a]
Boligblokk	– [a]	– [a]	– [a]	100 % [a]
Barnehage	70 %	30 %	5	60 %
Kontorbygning	65 %	35 %	5	60 %
Skolebygning	70 %	30 %	2,5	60 %
Universitets- og høgskolebygning	70 %	30 %	4	70 %
Sykehus	75 %	25 %	5	70 %
Sykehjem	75 %	25 %	5	70 %
Hotellbygning	60 %	40 %	6	50 %
Idrettsbygning	80 %	20 %	5	60 %
Forretningsbygning	70 %	30 %	4	75 %
Kulturbygning	70 %	30 %	4	60 %
Lett industribygning, verksted	70 %	30 %	4	60 %

Figur 12 – Standardiserte verdier for persontetthet i henhold til tabell H.2 i NS 3031 [5]

Avhengighet

Parametere som danner grunnlag for funksjon må være definert i modellen for at skriptet skal fungere. Det er mulig å velge egne parametere basert på bedriftens prosjektmal, men da er det viktig at skriptet tilpasses denne endringen for funksjon.

Resultat

Figur 13 illustrerer manglende beregnede verdier for luftmengder før skriptet blir kjørt.

A	B	C	D	E	F	G	H	I	J
Etasje	Name	Number	Areal	Tilstedeværelse	Funksjon	Personer	Maks Luftmengde	Min Luftmengde	Gjennomsnittlig Luft
02 Etasje	Lager	20	23.16 m²						
02 Etasje	Kontor	21	20.08 m²	Opphold					
02 Etasje	Stillerom	22	19.63 m²	Opphold					
02 Etasje	Kontor	23	41.75 m²	Opphold					
02 Etasje	Møterom	24	58.74 m²	Opphold	Annet				
02 Etasje	Møterom	25	27.63 m²	Opphold					
02 Etasje	Kontor	26	28.26 m²	Opphold					
03 Etasje	WC	27	34.72 m²		Annet				
03 Etasje	WC	28	42.00 m²		Annet				

Figur 13 – Før skriptet har blitt kjørt

Figur 14 viser deler av resultatet etter at skriptet har blitt kjørt. Her ser vi at rom som er definerte som oppholdsrom tar stilling til persontetthet, sekundærrom ekskluderer persontettheten og rom definert som «Annet» blir utelatt og må beregnes manuelt.

A	B	C	D	E	F	G	H	I	J
Etasje	Name	Number	Areal	Tilstedeværelse	Funksjon	Personer	Maks Luftmengde	Min Luftmengde	Gjennomsnittlig Luft
02 Etasje	Lager	20	23.16 m²				58 m³/h	58 m³/h	58 m³/h
02 Etasje	Kontor	21	20.08 m²	Opphold		8	258 m³/h	50 m³/h	175 m³/h
02 Etasje	Stillerom	22	19.63 m²	Opphold		8	257 m³/h	49 m³/h	174 m³/h
02 Etasje	Kontor	23	41.75 m²	Opphold		17	546 m³/h	104 m³/h	370 m³/h
02 Etasje	Møterom	24	58.74 m²	Opphold	Annet				
02 Etasje	Møterom	25	27.63 m²	Opphold		11	355 m³/h	69 m³/h	241 m³/h
02 Etasje	Kontor	26	28.26 m²	Opphold		11	357 m³/h	71 m³/h	242 m³/h
03 Etasje	WC	27	34.72 m²		Annet				
03 Etasje	WC	28	42.00 m²		Annet				

Figur 14 – Deler av resultatet etter at skriptet er kjørt

Evaluering

For at funksjonen til skriptet skal være gyldig er det viktig at nødvendige parametere er definert i modellen. Dette gjøres ved å definere parametere i «Spaces» for hvert prosjekt, eller å ha de definerte i prosjektmalen som benyttes for å unngå å gå gjennom definisjonsprosessen fra prosjekt til prosjekt.

Skriptet er avhengig av at arkitekt har definert romnavn og romnummer i forkant av beregninger samt en kravspesifikasjon fra kunde over hvor mange personer er tenkt i de ulike rom hvor standardiserte verdier ikke skal benyttes.

Skriptet eliminerer behovet for å ha et separat regneark som benyttes til luftmengdeberegninger. Alle beregninger kan gjennomføres i modellen og ingeniøren har mulighet til å velge egne verdier for material og personbelastning. Ingeniøren har videre mulighet til å legge inn beregningsrelevant informasjon i modellen basert på kundens ønsker samt standardiserte verdier for bygningskategorien. Luftmengdeskjema kan i etterkant eksporteres til Excel når dette etterspørres. Behovet for fagspesifikk kompetanse er fremdeles nødvendig hvor vurderinger gjennomføres av

spesialist. Derimot elimineres det dobbeltarbeid ved at ingeniøren kan jobbe med beregningene direkte i modellen.

8.1.3 Skript 03 – Overføring av parametere fra «Spaces» til ventiler

Beskrivelse

Skriptet har som oppgave å overføre informasjon over til ventiler som befinner seg i hvert rom. Hensikten ved overføring av romnavn og romnummer til ventiler er å danne oversikt over beliggenhet til teknisk utstyr i prosjektet. Ventiler er et eksempel, men skriptet kan benyttes til all type utstyr. Skriptet undersøker hvilke objekter som befinner seg i et romvolum og sorterer informasjon deretter.

Avhengighet

Følgende parametere må være definerte for ventiler for at informasjonen skal overføres fra Spaces:

Tabell 1 – Oversikt over parametere

Navn	Discipline	Type of parameter	Input
Luftmengde rom	HVAC	Air Flow	[m³/h]
Romnummer	Common	Integer	Nummer
Romnavn	Common	Text	Tekst

Videre forutsettes det at «Spaces» i modellen har samme parametere definert i *Skript 02.* «Spaces» må også fylle volumet definert med vegg avgrensinger og start på overkantdekke gulv og underkantdekke til overliggende etasje. Utstyr må ligge i romvolumet for å kunne bli tildelt informasjonen som ligger i rommet.

Resultat

Tabell 2 – Resultat for skript 03 Overføring av parametere fra Spaces til ventiler

Før skriptet kjøres

Type	System Classification	Romnummer	Romnavn	Luftmengde rom
Orion-LOV-TA-160	Supply Air			
Orion-LOV-TA-160	Supply Air			
Orion-LOV-TA-160	Supply Air			
Orion-LOV-TA-160	Supply Air			
Orion-LOV-TA-160	Supply Air			
Orion-LOV-TA-160	Supply Air			
Orion-LOV-TA-160	Supply Air			
Orion-LOV-TA-160	Supply Air			
Orion-LOV-TA-160	Supply Air			

Etter skriptet har blitt kjørt

Type	System Classification	Romnummer	Romnavn	Luftmengde rom
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h
Orion-LOV-TA-160	Supply Air	16	Personalrom	842 m³/h
Orion-LOV-TA-160	Supply Air	16	Personalrom	842 m³/h
Orion-LOV-TA-160	Supply Air	16	Personalrom	842 m³/h

Evaluering

Skriptet fungerer så lenge objekter befinner seg i volumet til «Spaces». Det ble ikke avdekket noe feil i dataoverføringen i testmodellen. Skriptet vil være meget tidsbesparende i forhold til TFM merkesystemet.

8.1.4 Skript 04 – Fordeling av total luftmengde til ventiler

Beskrivelse

Fordelingen av luftmengdene til de ulike ventilene blir gjennomført i Excel. Dette gjennomføres ved skript som overfører informasjon fra modellen til Excel hvor endringer blir gjennomført.

Avhengighet

Ventilene må ha en definert parameter med informasjon om den totale luftmengden i rommet. Det anbefales å lage en egen Excel mal som blir benyttet til dette hvor første rad er definert for de parametere man ønsker å jobbe med. En egen Excel mal for dette skriptet er med i leveransen.

Resultat

Informasjonen overføres og ingeniør fordeler totale luftmengden på de ventilene som er i rommet basert på type ventil. Fordelingen i dette eksempelet er markert med rødt.

Ventiltype	Romnummer	Romfunksjon	Total luftmengde i rom	Luftmengde ventil
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	18	Kantine	1199	250
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	18	Kantine	1199	250
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	18	Kantine	1199	250
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	18	Kantine	1199	150
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	18	Kantine	1199	150
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	18	Kantine	1199	149
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	16	Personalrom	842	150
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	16	Personalrom	842	150
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	16	Personalrom	842	150
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	16	Personalrom	842	150
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	16	Personalrom	842	100
Family Type: Orion-LOV-TA-160, Family: Orion-LOV-TA	16	Personalrom	842	142

Figur 15 – Resultat for Skript 04 fordeling av totale luftmengde til ventiler

Evaluering

Det hjelper å ha en definert Excel mal i forkant av uttaket. Informasjonen plasseres riktig i Excel fra modellen. Det er mye kjappere å fordele luftmengdene i Excel enn å jobbe direkte i modellen. Det kan derimot argumenteres mot om dette er en optimal måte å jobbe på. Man har ikke oversikt over ventilens plassering i rommet med hensyn på kastelengder. Det er derimot en interessant prosess som kan benyttes til å optimalisere en rekke andre arbeidsoppgaver hvor man er mindre avhengig av å jobbe direkte i modellen.

8.1.5 Skript 05 – Tilbakeføring av luftmengder til Revit

Beskrivelse

Etter at ventiler har blitt tildelt luftmengde basert på den totale luftmengden i rommet ved å følge arbeidsprosessen i forrige skript, blir informasjonen tilbakeført til modellen og ventilinformasjonen blir oppdatert.

Avhengighet

Et viktig hensyn å ta stilling til er at dersom man endrer eksisterende ventiler i modellen så må man gjenta prosessen i *Skript 04*. Dette er pga. at den tidligere modellen inneholder en ID som ikke lenger finnes. Dette skaper en ny ventilliste som ikke tar stilling til den som ble definert i tidligere skript basert på endrede ventiler. Det beste i et slikt tilfelle er å sette inn luftmengden manuelt i den nylig plasserte ventilen i modellen.

Resultat

Type	System Classification	Romnummer	Romnavn	Luftmengde rom	Flow
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h	250 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h	250 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h	250 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h	150 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h	150 m³/h
Orion-LOV-TA-160	Supply Air	18	Kantine	1199 m³/h	149 m³/h
Orion-LOV-TA-160	Supply Air	16	Personalrom	842 m³/h	150 m³/h
Orion-LOV-TA-160	Supply Air	16	Personalrom	842 m³/h	150 m³/h
Orion-LOV-TA-160	Supply Air	16	Personalrom	842 m³/h	150 m³/h
Orion-LOV-TA-160	Supply Air	16	Personalrom	842 m³/h	150 m³/h
Orion-LOV-TA-160	Supply Air	16	Personalrom	842 m³/h	100 m³/h
Orion-LOV-TA-160	Supply Air	16	Personalrom	842 m³/h	142 m³/h

Figur 16 – Resultat for Skript 05 - Tilbakeføring av luftmengder til Revit

Evaluering

Her er evalueringen lik som for *Skript 04*

8.2 Effektbehovsberegninger

Skriptene presentert nedenfor, består i utgangspunktet av to skript, hvorav *skript 06* består av fire deler som har som hensikt å hente ut alle arealer relevant for beregning av effektbehov på romnivå etter NS-EN-12831. [7] *Skript 07* dreier seg om beregning av ventilasjonsvarmetap.

8.2.1 Skript 06 – Del 1 og 2 - Hente ut dør- og vindusareal tilknyttet Spaces/Rooms.

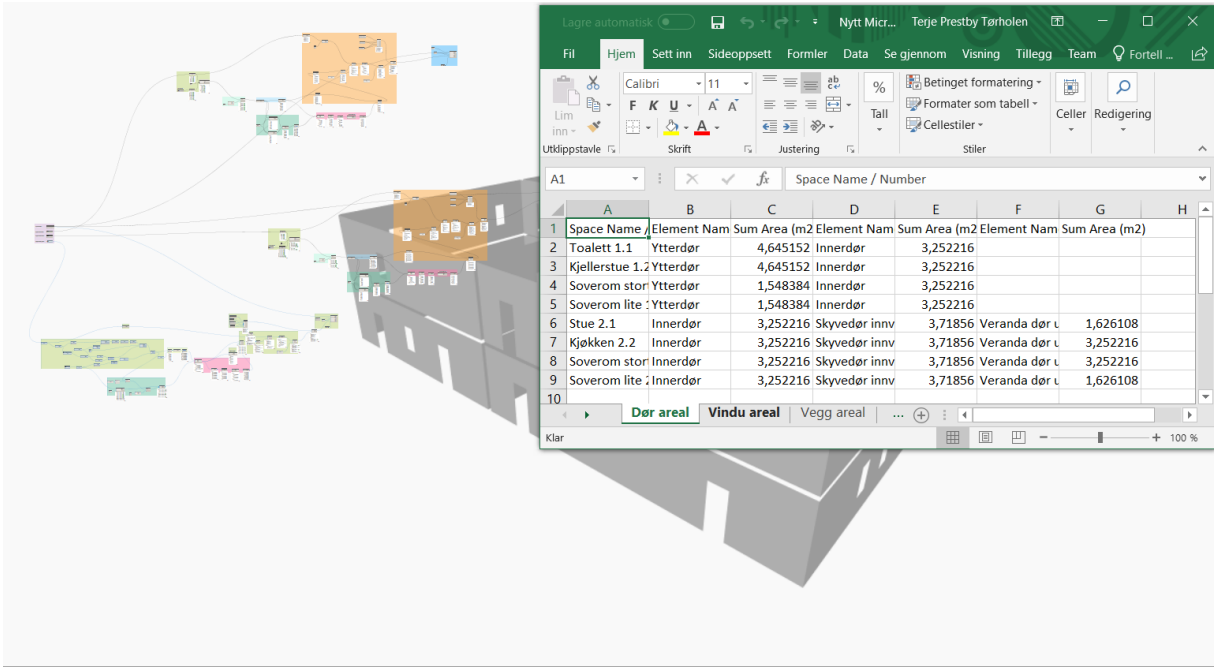
Beskrivelse

Del 1 og 2 i *skript 06* for uthenting av arealer for vindu-/dører er helt like i utforming, bortsett fra en enkel node som velger elementtype, så begge vil kommenteres felles. Skriptet har som oppgave å samle vindu- og dørarealer på romnivå, for bruk i effektbehovsberegninger. Dette skriptet vil samle alle elementer av samme type, summere arealene og presentere dette i et Excel dokument, slik at man enkelt kan hente ut nødvendige areal til de ulike rommene i bygget.

Avhengighet

For at skriptet skal fungere, er det avhengig av at det finnes objekter av type dør/vindu, og at disse har dimensjoner som høyde og bredde. Dette vil si at objektene har parametriske egenskaper, slik at det faktisk er mulig å hente ut data for videre beregninger. Det er videre avhengig av definerte «Spaces», eventuelt «Rooms».

Resultat



Figur 17 – Resultat for skript 06 - Del 1 dørareal tilknyttet Spaces/Rooms

Evaluering

Del 1 og 2 i skriptet fungerer godt i de fleste scenarioer, og har liten avhengighet til andre fagfelt da det er relativt innarbeidet å legge inn de krevde parameterne. Det en må være observant på er skissemodeller, hvor arkitekten gjerne ikke har modellert objektene inn i modellen, men heller lagd åpninger der det skal være dører og vinduer. Skriptet vil naturligvis ikke virke på disse modellene.

8.2.2 Skript 06 – Del 3 - Hente ut veggareal tilknyttet Spaces/Rooms

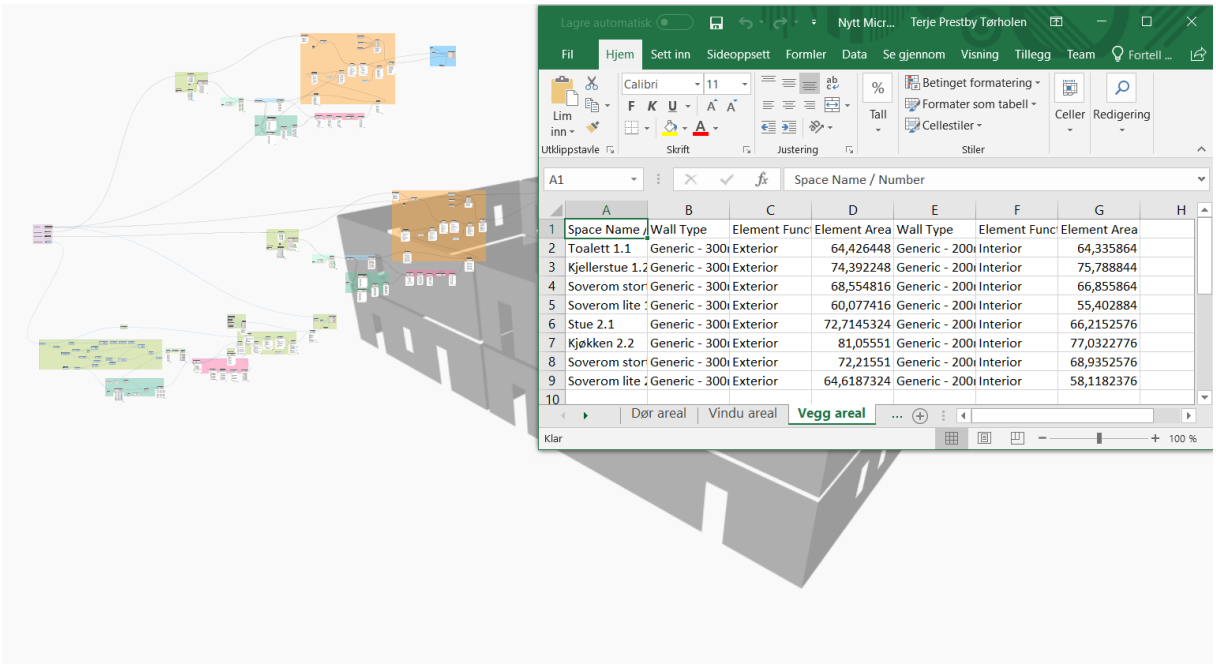
Beskrivelse

Del 3 har som oppgave å hente ut veggareal, slik at disse kan brukes i effektbehovsberegninger. Tanken er at det skal automatisk fjerne areal fra vinduer og dører, eventuelt andre komponenter, slik at en ikke trenger å subtrahere vekk disse arealene manuelt, samt samle arealer på romnivå basert på inner- eller yttervegger.

Avhengighet

Del 3 er avhengig av å ha «Rooms» eller «Spaces» definert i modellen, da skriptet baserer seg på å hente ut de veggene som avgrenser disse. Da følger også avhengigheten av at «Rooms» eller «Spaces» er definert på en riktig måte, for at en ikke skal få med vegger fra andre deler av modellen. Videre er det hensiktsmessig at arkitekten definerer hva som er yttervegg og hva som er innervegg i modellen slik at ingeniøren lett kan skille disse og velge de arealene som er relevante for beregningene. Det er også nødvendig at modellen har veggobjekter definert som kategori «*Wall*».

Resultat



Figur 18 – Resultat for skript 06 Del 3 Hente ut nødvendig veggareal tilknyttet Spaces/Rooms

Evaluering

Del 3 er ikke stabilt eller ferdigutviklet da det må tas stilling til veggtykkelse på innervegg, for å beregne fasadearealet riktig i henhold til NS-EN-12831. Del 3 gir kun arealet til fasadeveggen basert på den aktuelle flaten til «Rooms» eller «Spaces» og mangler derfor noe på fasadearealet. Det har blitt testet flere måter å løse dette problemet på, men etter vurderinger har en kommet frem til at dette krever koding i Python på et høyere nivå. Del 3 er også usikkert å bruke da mange modeller ikke er modellert godt nok, slik at en ofte får null verdier. Tilbakemeldinger fra ingeniører etter demonstrasjon var at det kan være lurt å hente ut arealene fra hvert enkelt «Room» eller «Space», slik at en har bedre kontroll på hva en høster av informasjon fra skriptet.

8.2.3 Skript 06 – Del 4 - Hente ut gulv- og takareal tilknyttet Spaces/Rooms

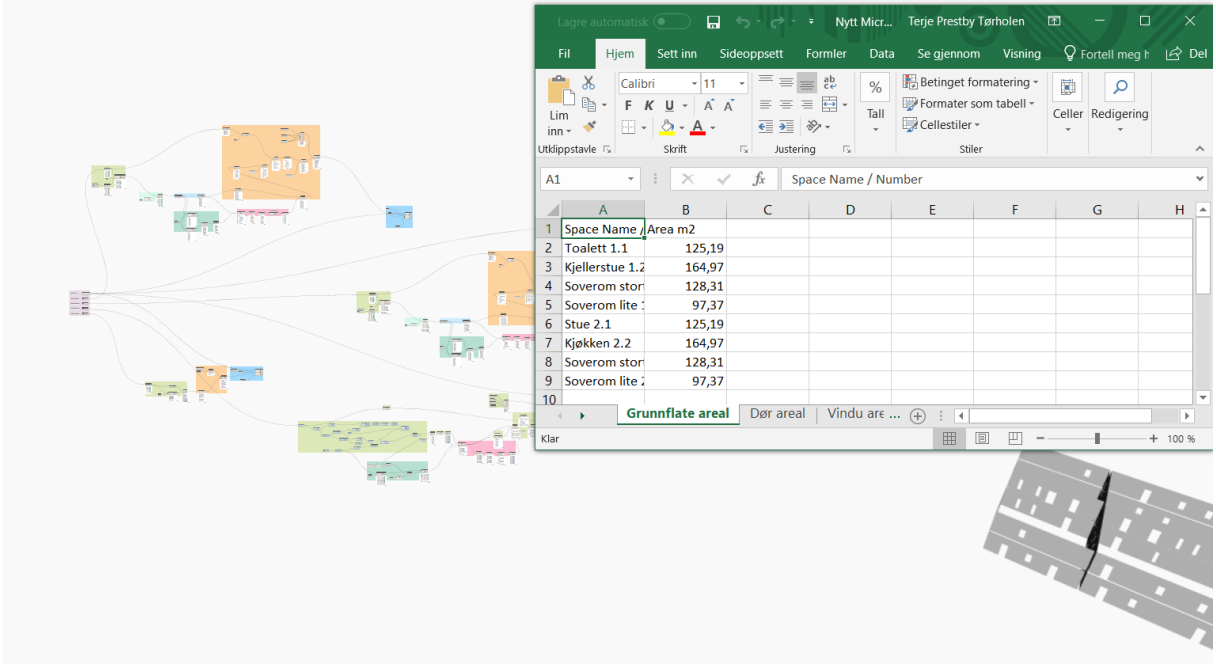
Beskrivelse

Oppgaven til Del 4 er å hente ut og samle gulvarealer fra «Rooms» eller «Spaces»

Avhengighet

Denne delen tar kun stilling til et eller flere plasserte «Rooms» eller «Spaces» i modellen. Dette modelleres ofte automatisk av ARK eller RIV, hvor man henter ut parameteren *«Area»* fra de overnevnte objektene. Resultatet av å kjøre denne delen, blir helt og holdent avhengig av at grensene er definert presist for hvert enkelt rom. For at dette skal være gyldig, og kunne brukes videre i beregninger, må dette være definert riktig i modellen. Gulv- og takareal forutsettes å være identiske ved bruk av denne delen.

Resultat



Figur 19 – Resultat av skript 06 Del 4 Hente ut nødvendig gulv- og takareal tilknyttet Spaces/Rooms

Evaluering

Muligheten til å hente ut gulvarealer fra «Rooms» eller «Spaces» finnes allerede i Revit med mengdeuttak. Det er allikevel hensiktsmessig å implementere dette i det totale skriptet, da en kun er avhengig av å hente ut arealene i en operasjon, sammen med de andre nødvendige areal for å utføre effektberegninger.

8.2.4 Skript 07 - Ventilasjonsvarmetap på romnivå

Beskrivelse

Dette skriptet beregner effektbehovet til oppvarming grunnet undertemperert tilluft fra ventilene. Beregningen tar stilling til tabell A.9 i SN/TS 3031 [6] i forhold til settpunkttemperatur for oppvarming i forhold til romtemperaturen. Skriptet er bygget opp med samme logikk som *Skript 02.*

Tabell A.9 – Normerte settpunkttemperaturer

Bygningskategori	Settpunkttemperatur for oppvarming[a] °C		Settpunkt-temperatur for kjøling[a, b, c] °C
	i driftstiden	utenfor driftstiden	
Småhus	22	20	24
Boligblokk	22	20	24
Barnehage	22	20	24
Kontorbygning	21	19	24
Skolebygning	21	19	24
Universitet/høyskole	21	19	24
Sykehus	21	19	24
Sykehjem	22	20	24
Hotellbygning	21	20	24
Idrettsbygning	19	17	24
Forretningsbygning	21	19	24
Kulturbygning	21	19	24
Lett industri/verksteder	21	19	24

Figur 20 – Tabell A.9 SN/TS 3031 [6] – Normerte settpunkttemperaturer

Brukeren har også muligheten til å velge egen settpunkttemperatur for oppvarming men da må feltet «Bygningskategori» stå tomt. Valg av tilluftstemperatur er også tilgjengelig for brukeren.

Avhengighet

For at skriptet skal fungere er det viktig at alle «Spaces» i modellen inneholder informasjon om luftmengder i en definert parameter. Det er mulig å definere egne parametere basert på bedriftens prosjektmal, men da er det viktig at skriptet tilpasses denne endringen for funksjonalitet. «Spaces» må ha avgrensning og representere de rom som står i kravspesifikasjonen. Dette forutsetter at ARK har satt inn romavgrensningen i modellen i forhold til byggherrens ønsker.

Resultat

Figur 21 viser deler av informasjonen for bygget før skriptet blir kjørt.

Etasje	Name	Number	Areal	Tilstedeværelse	Funksjon	Personer	Maks Luftmengde	Varmetap Ventilasjon
02 Etasje	Lager	20	23.16 m²				58 m³/h	
02 Etasje	Kontor	21	20.08 m²	Opphold		8	258 m³/h	
02 Etasje	Stillerom	22	19.63 m²	Opphold		8	257 m³/h	
02 Etasje	Kontor	23	41.75 m²	Opphold		17	546 m³/h	
02 Etasje	Møterom	24	58.74 m²	Opphold	Annet			

Figur 21 – Viser informasjon før skriptet er kjørt

Figur 22 viser resultater som fremkommer etter at skriptet har blitt kjørt.

Etasje	Name	Number	Areal	Tilstedeværelse	Funksjon	Personer	Maks Luftmengde	Varmetap Ventilasjon
02 Etasje	Lager	20	23.16 m²				58 m³/h	79 W
02 Etasje	Kontor	21	20.08 m²	Opphold		8	258 m³/h	351 W
02 Etasje	Stillerom	22	19.63 m²	Opphold		8	257 m³/h	349 W
02 Etasje	Kontor	23	41.75 m²	Opphold		17	546 m³/h	743 W
02 Etasje	Møterom	24	58.74 m²	Opphold	Annet			

Figur 22 – Resultat av skriptet [07] Ventilasjonsvarmetap på romnivå

Evaluering

Skriptet er brukervennlig og har en robust funksjon når alle avhengigheter er dekket. Avhengighetene er ikke store for RIV sitt ansvarsområde, men større i forhold til hvordan ARK har definert sine rom som «Spaces» drar sine geometriske egenskaper fra.

8.3 Spesifikke

I dette underkapitlet vil en se på en rekke skript som ikke har noen direkte tilknytning til hverandre.

8.3.1 Skript 08 – Kvalitetskontroll av trykkfallsberegninger i MagiCAD

Beskrivelse

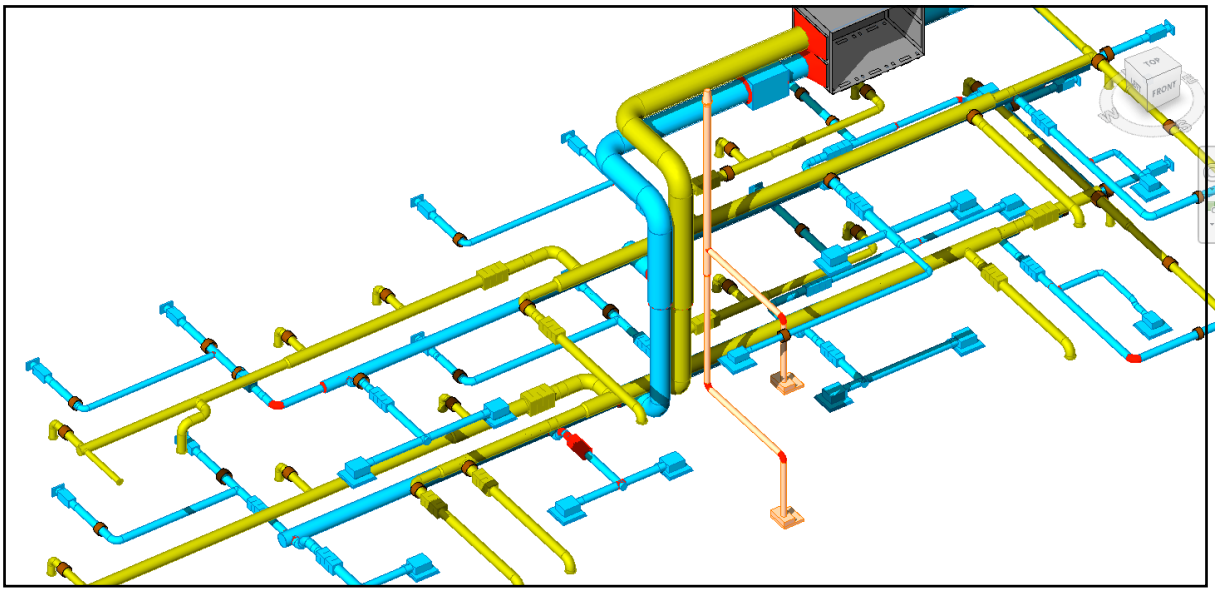
Skriptet undersøker ventilasjonsmodellen for trykkfallsberegninger. Det er en utvidelse av [19] som fargelegger de komponenter som eventuelt ikke har blitt inkludert i trykkfallsberegningene i MagiCAD med fargen rød. Det anbefales å lage separate 3D visninger for området som skal sjekkes ettersom skriptet overskriver fargen på elementene som ikke har blitt trykkfallsberegnet i den aktive visningen.

Avhengighet

Det må ha blitt gjennomført trykkfallsberegninger på anlegget før dette skriptet blir tatt i bruk med MagiCAD. Dersom trykkfallsberegninger har blitt gjennomført ved bruk av Revit MEP erstattes parameternavn hvor trykkfallsverdien blir lagret med relevant parameter for Revit MEP.

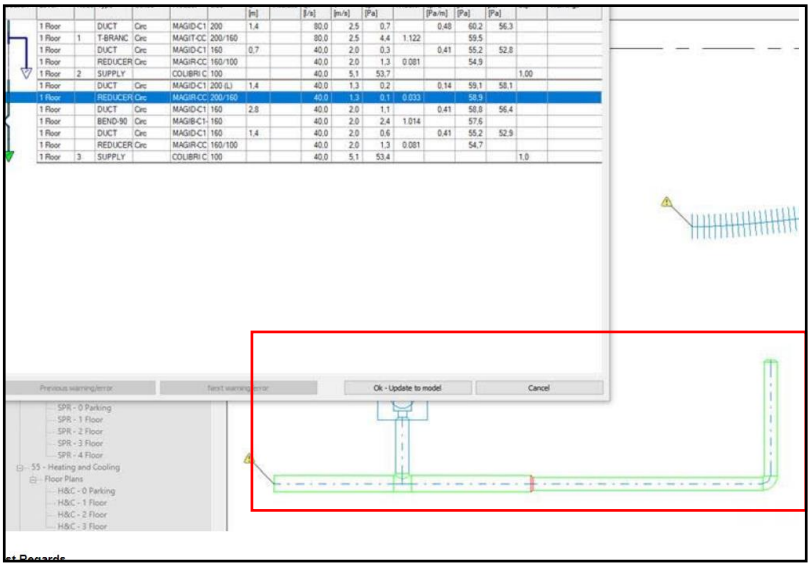
Resultat

Nedenfor er en illustrasjon av de elementer som hadde 0 Pascal i trykkfall basert på modell fra gruppen sitt VVS prosjekt, høstsemesteret 2017.



Figur 23 – Resultat av skript 08 Kvalitetskontroll av trykkfallsberegninger i MagiCad

Føringer ble koblet til på nytt og omprosjektert for å se om nullverdiene ble borte. Det resulterte med samme feil som gir mistanke om at det er elementer som programmet ikke klarer å tolke i beregningene. Kontakt ble opprettet med tekniske ingeniører hos MagiCAD **[Vedlegg III]** som mener at programmet skal kunne beregne alle deler. Derimot var levert eksempel, som var feilfritt, et veldig forenklet anlegg som ikke kan sammenlignes med et større anlegg.



Figur 24 – Tilbakemelding fra Mikael Engstrøm

Evaluering

Skriptet effektiviserer kvalitetskontrollen av trykkfallsberegningene ettersom MagiCAD unnlater noen elementer i systemet. Dette gir en oversikt for ingeniøren i prosjekteringen av anlegget.

Resultatene fra *Figur 23* viser at behovet for et skript som tydeliggjør objekter som ikke er trykkfallsberegnet kan være et nyttig verktøy.

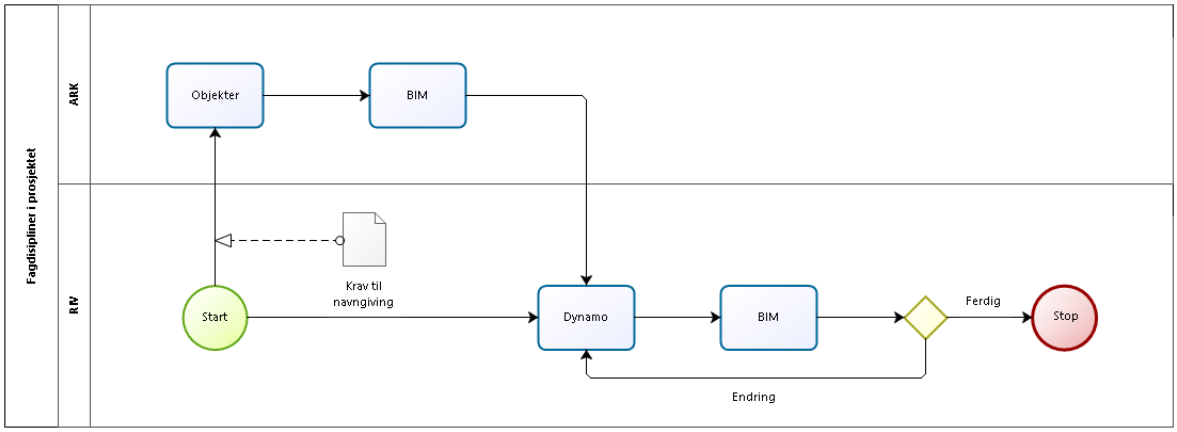
8.3.2 Skript 09 –Utstyrstelling og beregning av sannsynlige vannmengder

Beskrivelse

Følgende skript ble utviklet etter samtale med prosjektingeniør hos Sweco ettersom utstyrtelling for beregning av normalvannmengder er en tidskrevende prosess. Skriptet tar kun stilling til den linkede ARK modellen og filtrerer utstyr basert på ulike rom og gir et mengdeuttak i Excel. Tilslutt utføres en beregning av størst sannsynlige vannmengder for summen av alle normalvannmengdene i bygget, dvs. reelt for bunnledning hvor alle vannmengdene møtes.

Avhengighet

For at skriptet skal kunne filtrere utstyr riktig er det essensielt at BIM objektene som definerer utstyret er logisk navngitt. Et forslag for å løse dette er at RIV definerer familier for teknisk utstyr og tar seg av parameterbehandlingen. Etter at korrekt navngiving er satt, sammen med ønskede parametere og definisjon av familiekategori av utstyrstype, overføres filene til arkitekten som plasserer utstyret i sin modell. *Figur 25* viser et prosesskart for hvordan samspill mellom de ulike fagfeltene kan gjennomføres.



Figur 25 – Prosesskart som viser hvordan samspill mellom de ulike fagfeltene kan gjennomføres

Resultat

Resultatet på uttaket fremkommer i *Figur 26*.

	A	B	C	D	E
1	**Romnavn og romnummer**	**Antall toaletter**	**Sone**	**NVM/stk**	**SUM NVM**
2	WC 46	5		1,8	9
3	WC 47	4		1,8	7,2
4	Bad 30	2		1,8	3,6
5	WC 28	6		1,8	10,8

Toalett | Dusj | Vask | Urinal | Vaskemaskin | Størst maksimal samtidighet

Maskimal samtidighet basert på all utstyr i bygget
5,164444366

Figur 26 – Resultat av skript 09 Sanitær

Evaluering

Skriptet sørger for rask beregning for bunnledningen. Videre dannes det oversikt over utstyr i de ulike rommene. Derimot er dette skilt på de ulike fanene og man har ikke samme informasjonen i en fane. Ulempen er at man ikke har oversikt over alt utstyr på en fane. Man bør fokusere på videreutvikling hvor man klarer å plassere informasjonen på en fane.

8.3.3 Skript 10 – Kontroll av parametere i «Spaces»

Beskrivelse

For at enkelte skript skal fungere er det viktig at parametere er definerte i forkant. Eksempelvis vil *Skript 02* bare fungere dersom man definerer parametere gitt i *Tabell 1* i «Spaces».

For å undersøke om disse parameterne er definert for kategorien i modellen kan det utvikles skript som eksempelet i *Figur 27*. En slik sjekk vil kartlegge om modellen er klar for aktuelle skript eller ikke.

Dette skriptet undersøker om parameteren «Funksjon» er tilstede i kategorien «Spaces». Dersom parameteren ikke er tilstede vil Sjekk noden gi beskjed om dette. Skriptet tar stilling til en parameter, men kan bygges ut til å inkludere alle parametere definert i *Tabell 3*.

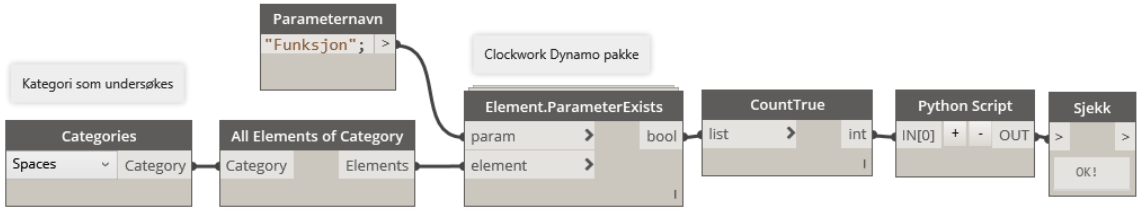
Tabell 3 – Liste over parameter som må være definert for at skriptet skal kjøres

Listenr	0	1	2	3	n
Parameter	Funksjon	Tilstedeværelse	Maks Luftmengde	Min Luftmengde	N

Avhengighet

Regelsjekkeren må gjennomføres i Dynamo og ikke Dynamo Player. Slike skript er avhengige av at den som utvikler de har veldefinerte parametere fra prosjektingeniører som skal arbeide med modellen.

Resultat



Figur 27 – Resultat av skript 10 Kontroll av parametere i Spaces

Evaluering

Regelsjekkeren gir raskt tilbakemelding på om nødvendig informasjon er tilstede i modellen. Skriptet er bare et eksempel i forhold til en problemstilling, men kan tilpasses til en rekke med BIM relaterte prosesser i prosjekter. Iterasjonstiden er lav ettersom en slik sjekk gjerne gjennomføres i tidligfase prosjektering.

8.3.4 Skript 11 – Kollisjonskontroll

Beskrivelse

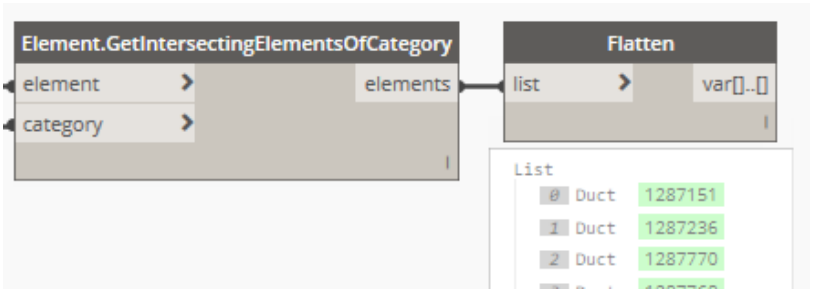
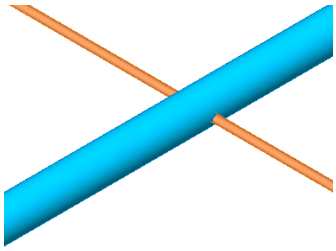
I prosjekteringen er det viktig at man unngår kollisjoner som resulterer i større kostander for prosjektet i byggefasen. Det finnes veldig gode verktøy for å sjekke BIM modellen for kollisjoner som eksempelvis Solibri Model Checker. Det tar derimot tid å eksportere en IFC fil til Solibri og er heller ikke hensiktsmessig dersom man skal undersøke kollisjoner mellom VVS føringer i egen modell.

Skriptet ble utviklet for å sjekke kollisjoner mellom rør og kanaler i modellen. På denne måten kan man undersøke kollisjoner som oppstår i egen prosjektering, før man eksporterer modell til Solibri for sjekk mellom andre fagfelt. Kolliderende elementer blir synliggjort i en egen node i Dynamo. Når man klikker på en av disse så blir man tatt direkte til kollisjonen i modellen.

Avhengighet

For å benytte skriptet er man avhengig av å ha selve skriptet åpent i Dynamo og ikke Dynamo Player slik at man har muligheten til å velge kolliderende elementer. Skriptet tar kun stilling til elementer inne i modellen og ikke elementer fra linkede modeller. I dette skriptet undersøkes kun kollisjoner mellom rør og kanaler. Dette kan derimot endres ved å endre kategorier.

Resultat

Resultat i Dynamo	Resultat i Revit
	

Figur 28 – Resultat fra skript 11 Kollisjonskontroll

Evaluering

Skriptet er en begrenset versjon av kollisjonssjekkeren i Revit men brukeren har en større fleksibilitet for å undersøke kollisjoner mot en rekke objekter dersom det videreutvikles.

8.3.5 Skript 12 – Farging av sprinkler i forhold til avstand fra vegg

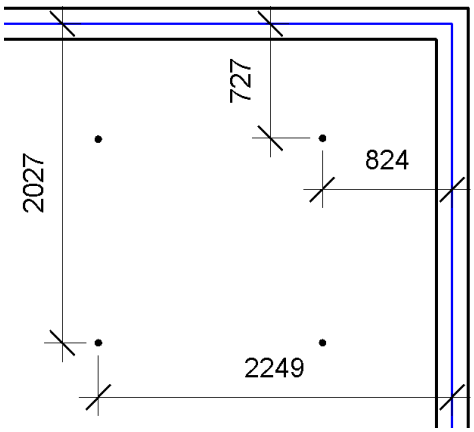
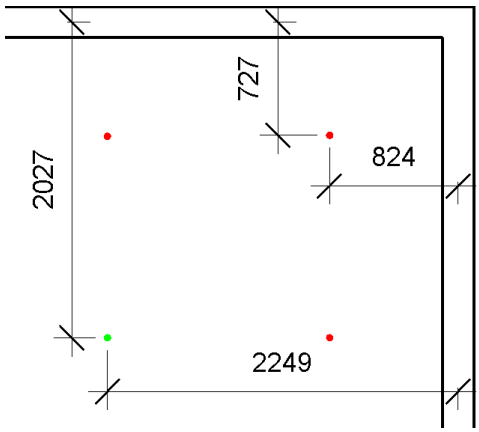
Beskrivelse

Det fullstendige skriptet vil automatisk overskride farge på sprinkler med tanke på avstand i fra veggen som er definert i skriptet. I skriptet har vi definert sprinkel avstand på utspredning til en meter. Dersom sprinkelen er for nær en vegg, vil det bli fargelagt til rødt, og dersom den er i tilstrekkelig avstand, til grønt.

Avhengighet

For at skriptet skal kunne filtrere lengdene mellom sprinklene og veggene, er det viktig å hente inn plassering til sprinklene og veggene. Dette skriptet baserer seg på at ARK er linket til RIV modellen.

Resultat

Før skriptet kjøres	Resultat etter kjørt skript
	

Figur 29 – Resultat av skript 12 Farging av sprinkler i forhold til avstand fra vegg

Evaluering

Farging av sprinklere i forhold til feilplassert avstand mellom objekt kan også gjøres i Solibri Model Checker. Men dette krever at modellen eksporteres til IFC som er tidskrevende. Skriptet vil bidra til tidsbesparelser. Skriptet nåværende tilstand tar kun stilling til avstand til vegger, men kan eventuelt videreutvikles til å inkludere flere elementer. Skriptet som ble programmert var krevende å programmere og det ble benyttet hjelp fra Dynamo forums. [20]

8.3.6 Skript 13 – Dimensjonsteksting av diametere for kanaler

Beskrivelse

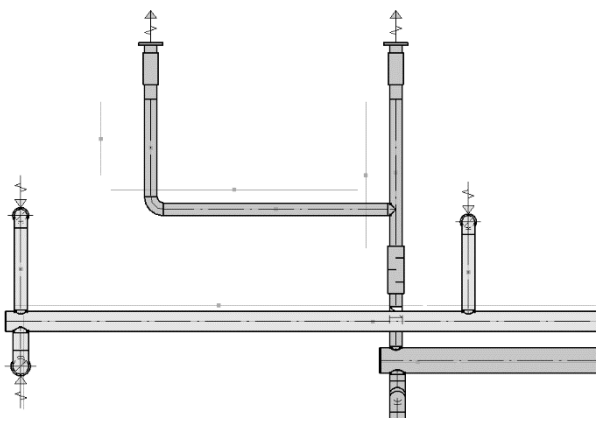
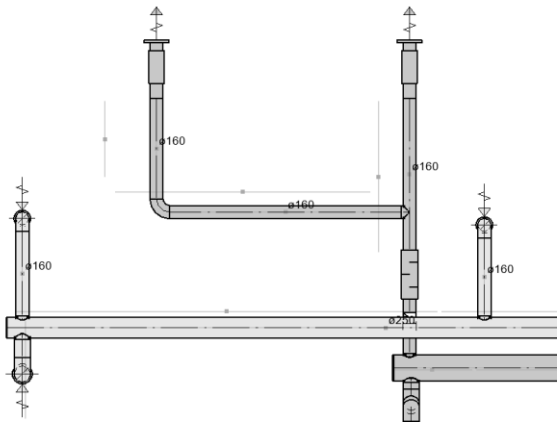
Skriptet vil effektivisere dimensjonsteksting av kanaler, da det ved plantegninger er nødvendig med dimensjonsteksting. Skriptet vil plassere teksting sentralt på kanallengden, samt unngå å angi dimensjon på kanalene som er mindre enn 900mm. Dette skriptet er ment som en erstatning for Revit funksjonen Tag All, som tekster alle komponenter uavhengig av lengden. *Figur 30* viser problematikken med Revit funksjonen Tag All.

Figur 30 – Revit funksjonen Tag All

Avhengighet

For at skriptet skal fungere optimalt kreves det at lengden er definert som parameter i kanalen, samt at riktig visningsvindu er valgt. Kanallengder under satt verdi blir ikke tekstet.

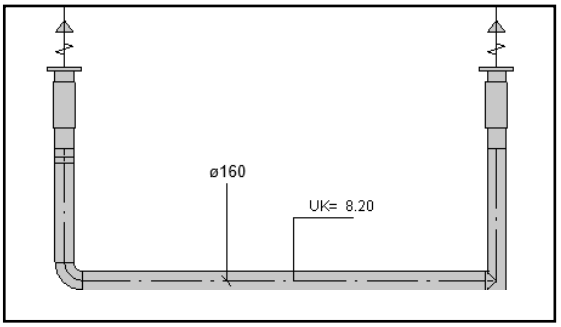
Resultat

Før skriptet kjøres	Resultat etter kjørt skript
	

Figur 31 – Resultat av skript 13 Effektiv dimensjonsteksting av diametere for ventilasjonsanlegget

Evaluering

Som vist på resultat er dimensjonen på selve kanalen. For å få en linje ifra kanalen og deretter dimensjonen som vist under kreves det programmering på et høyere nivå. Det er absolutt mulig å få til dette ved videreutvikling av skriptet.



Figur 32 – Eventuelt resultat ved videreutvikling av skriptet

9 Diskusjon

9.1 Krav til kompetanse

Man kan stille seg selv spørsmålet: «Bør/vil vi forvente at alle prosjekterende VVS ingeniører har kompetanse innen bruk av verktøy som Dynamo?». Eller er det mer hensiktsmessig at hver avdeling i en bedrift har spesifikke personer med programmeringsbakgrunn i tillegg til spesialiseringen innen sitt fagfelt?

Her kan det tenkes at det mest hensiktsmessige er å ha en superbruker i hver avdeling. Denne tilnærmingen gir muligheter for fagpersonen å diskutere programmeringstekniske problemstillinger, med fagpersoner på andre avdelinger ettersom slike diskusjoner er uavhengige av fagfeltet som ingeniørene har spesialisert seg i.

Programvareutviklere og BIM teknikere har også muligheten til å levere Dynamo skript som løser en spesifikk oppgave. Dette setter lys på om en bedrift da har behov for ingeniører som har kompetanse i utvikling av skript. Hvorfor skal en ressurssterk bedrift ha behov for prosjektingeniører som skal sette seg inn i skriptutvikling når en BIM teknikker kan løse problemet eller at en leverandør kan levere det spesifikke produktet som etterspørres? Her er det tre faktorer som bør tas hensyn til – faglig kunnskap, tid og tilpasninger.

En ingeniør har mye kunnskap om sitt fagfelt noe som er en stor fordel når et skript skal utvikles. Formålet med skriptet er at den skal automatisere en prosess. Faglig kunnskap, samt erfaring, vil gjøre at ingeniøren lettere kan identifisere fallgruver som oppstår i utviklingen, slik at endeproduktet er mer robust når det benyttes i prosjekter. En utvikler eller BIM teknikker har ikke nødvendigvis denne kompetansen, og problemer vil bli kartlagt når skriptet først er i bruk, noe som ikke er fordelaktig for prosjektet. Videre er det en mer tidskrevende prosess å sende tilbakemelding hvor funksjonsproblemer blir kartlagt i forhold til om fagkyndig kan gå inn i Dynamo og optimalisere skriptet selv.

Motargumentet her er at en utvikler vil ha større kompetanse innen programmering, enn en VVS ingeniør som kun har programmeringskunnskap innen visuell programmering, eventuelt tilleggskompetanse om Python på et begrenset nivå. Dette motargumentet kan også være med på å gjøre ledere i bedrifter skeptiske til å benytte en teknologi som blir utviklet av personer med begrenset forståelse for programmering. Slik bekymring er naturlig og viktig å ta stilling til når man utvikler et produkt som skal benyttes i prosjekter med stor verdi. Dersom man ser på verktøyet som blir benyttet så har det et begrenset formål, nemlig å gjennomføre prosesser i Revit. Man må ikke inneha ekstensiv kompetanse innen programmering for å utvikle et produkt.

Videre er Dynamo et visuelt programmeringsverktøy som eliminerer behov for koding med tekst. På denne måten har byggingeniører mulighet til å utvikle velfungerende skript med begrenset programmeringskompetanse. Det bør settes fokus på feil og fallgruver før man tar skriptet i bruk i et prosjekt, men denne typen kvalitetskontroll må også gjennomføres av andre utviklere av lignende produkter.

9.2 Mindre avhengighet av programvareutviklere

Gruppen har valgt å lage en større andel med skript for å vise frem skript av større og mindre kompleksitet. Formålet med dette er å synliggjøre hvor lite det skal til for å få frem et velfungerende

skript ved bruk av verktøyet. Andre rapporter innen Dynamo, som f.eks. «Automation of information flow from Revit to BSIM using Dynamo» [21] og «Productivity improvement of reinforcement drawing generation: a DSR perspective» [18], tar for seg skript med stor kompleksitet. Dette kan virke uoverkommelig for ingeniører, i forhold til implementering i arbeidsmetoden. Resultat kan dermed bli at ingeniøren fortsatt henvender seg til eksterne programutviklere for løsninger på problemstillinger.

Et fortrinn er at de skriptene som utvikles i bedriften ikke sirkulærer i det frie markedet og tilgjengelig gjøres for konkurransen. Dette gjør at bedrifter, som satser på gode løsninger ved bruk av Dynamo, vil få ytterligere konkurransefortrinn som følge av satsningen forutsatt at skriptene som utvikles er effektive i bruk.

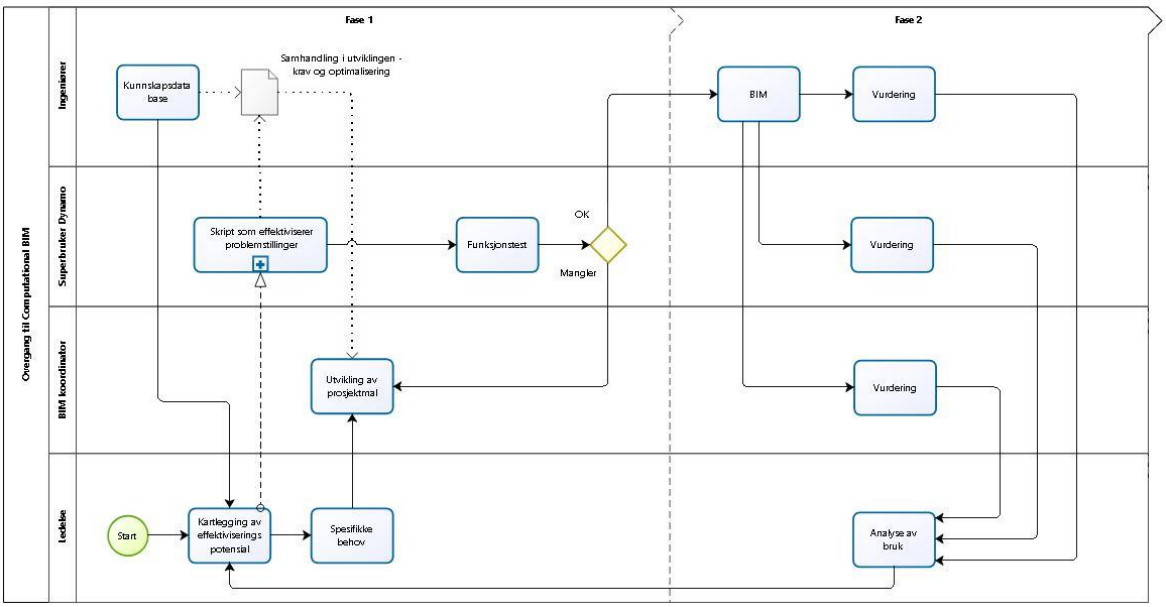
9.3 Kravstilling og fokus på god datastruktur

En interessant observasjon i utviklingen av skriptene, er hvilke parametere som skaper avhengighet for å ivareta funksjonalitet og gyldighet til produktet. Dette påvirker ikke bare hvilke elementer som skal inkluderes av rådgiver i sin BIM modell, men også hvilke krav som må stilles til andre fagfelt for å kunne utnytte skriptene. Fokus på god datastruktur er essensielt for bruk av skript med stor, eller liten avhengighet av andre fagdisipliner.

Et eksempel er *Skript 02* hvor RIV er avhengig av at ARK har plassert ut objekter som definerer rom med navn og nummerering for å etablere oversikt. Videre er det viktig at krav fra byggherren angående antall personer i rom er spesifisert. Dette bør ligge inne i ARK sin modell eller i romlisten fra byggherren. Fordelen med at informasjonen ligger i ARK modellen, er at man har mulighet til å overføre informasjonen hurtig. Dersom man skal overføre informasjon fra regneark er det naturligvis en meget tidskrevende prosess. *Skript 09* er et annet eksempel som setter krav til hvordan objekter navngis for effektivt mengdeuttak for videre beregninger.

Synliggjøring av hva som må ligge til grunn for å øke effektiviteten til den prosjekterende, vil øke motivasjonen til å sørge for at nødvendig informasjon fra andre fagfelt ligger til grunn i prosjektet. Dermed kan den prosjekterende vektlegge fokus på spesifikke krav tidlig i prosjektets forløp, og kommunisere dette tydelig til fagfeltene det angår.

Derfor er det viktig med en plan for hvordan man kan legge til rette for implementering og videreutvikling av teknologien i en bedrift, hvor god datastruktur er en hjørnestein. Prosesskartet vist i *Figur 33* er en sammenstilling over hva gruppen mener er en god tilnærmingsmåte til å implementere teknologien, som er et resultat basert på de funn kartlagt i utviklingsprosessen samt testing av skriptene.



Figur 33 – Prosesskart for god tilnærmingsmåte til å implementere teknologien

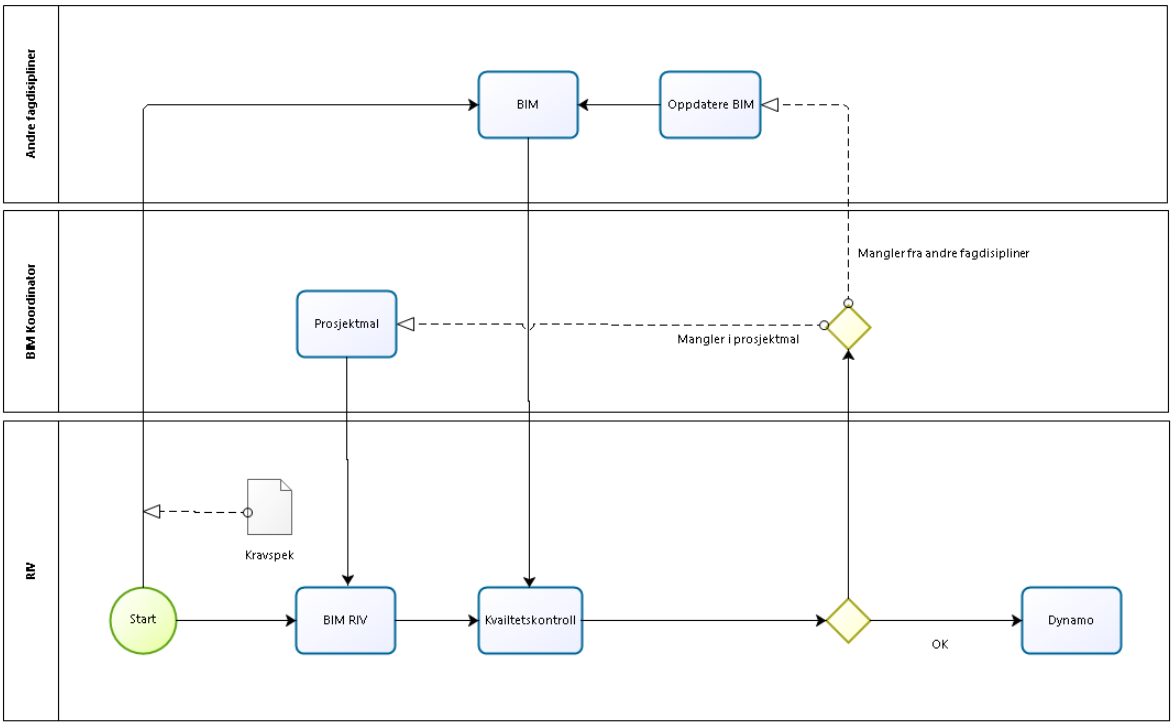
9.4 Kontroll av modell

For å kunne utnytte skript effektivt og sømløst i fremtidige prosjekter, vil det være nødvendig med kontrollsjekk av modeller i forkant av å «kjøre» skript. Ved å sette opp krav til objekter eller krav til innhold av spesifikk informasjon og parametere til objektene, vil man tidlig i prosjektet kunne kontrollere at de satte krav til modellinformasjon fra kravspesifikasjonen, faktisk er tilstede.

Dette vil kunne fungere som en kontrollsjekk opp mot kravspesifikasjonen, samtidig som man vil begrense unødvendig tidsbruk til å «kjøre» skript på modeller, som ikke inneholder den nødvendige informasjonen. Dette kan gjøres med programmer som Solibri Model Checker, som har en rekke *maler* for informasjonskontroll, blant annet BIM regelsjekk for bygg- og arkitektmodeller.

Men disse *maler* vil ofte kontrollere mye mer enn spesifikk ønsket informasjon, hvis ikke noen med kunnskap og erfaring kan definere disse regelsjekkene til hvert spesifikt prosjekt. Da kan en annen løsning for å utføre denne kontrollen være å definere og bruke et eget utviklet skript, som eksempelvis *Skript 10*, som sjekker definerte parametere til et ønsket objekt.

En regelsjekker kan bli utviklet til å undersøke modellen som en del av en arbeidsprosess i tidligfase prosjektering. Et eksempel på en slik arbeidsprosess er illustrert i *Figur 34*.



Figur 34 – Prosesskart, arbeidsprosess for informasjonssjekk

9.5 Brukervennlighet

Dynamo Player gir ingeniører som ikke er komfortable med arbeid direkte i skriptet, muligheten til å benytte et enkelt brukergrensesnitt for å definere inputverdier. Muligheten til å benytte dette enkle brukergrensesnittet er kun tilgjengelig i Revit 2018, og ikke tidligere versjoner.

Iterasjonstiden for å kjøre hvert skript er lenger i Dynamo Player, ettersom skriptet blir åpnet i bakgrunnen som en delprosess. Dette er derimot ikke en begrensning, da man uansett må åpne hvert skript manuelt dersom man ikke benytter Dynamo Player. Videre har man en skriptliste basert på mappevalg som gir oversikt for brukeren.

9.6 Gjenbruk av skript

Det er ønskelig at et skript kan dekke flere prosjekteringsoppgaver. Gjenbruk av et skript, til å dekke flere formål, vil også medføre ytterligere gevinst ved bruk av Dynamo. Matrisen i *Tabell 4* illustrerer et utvalg av skript hentet fra resultater og hvilke muligheter de har til å bli gjenbrukt i de ulike delfagene innunder VVS. Kolonnen «Manuelt» sier noe om det fremdeles gjenstår manuelt arbeid i en delprosess som skriptet skal effektivisere. Kolonnen «Utvikling» beskriver om det er muligheter til å redusere andel manuelt arbeid ved å videreutvikle skriptet.

Tabell 4 – Utvalg av skript fra resultater og muligheter for gjenbruk

	Gjenbruk av skript				
Skript	Varme	Ventilasjon	Sanitær	Manuelt	Utvikling
Skript 02		✓		✓	✓
Skript 03	✓	✓	✓		
Skript 04/05	✓	✓	✓	✓	✓
Skript 06	✓			✓	✓
Skript 07		✓			
Skript 09			✓	✓	✓
Skript 10	✓	✓	✓	✓	✓
Skript 12		✓		✓	✓
Skript 13	✓	✓	✓	✓	

Det er også slik at man kan bryte ned prosjekteringsoppgaver, innen eksempelvis varme, ned i en rekke mindre oppgaver. Gjenbruk av skript trenger ikke å innebære at hele skriptet skal benyttes til å utføre en annen oppgave, men kan også innebære gjenbruk av deler av skriptet til å utvikle et nytt. *Tabell 5* viser et utvalg av tre deloppgaver innen prosjektering av varmeanlegg, samt en oversikt over hvorvidt deler av skript kan gjenbrukes til å løse en annen problemstilling.

Tabell 5 – Deloppgaver innen prosjektering

	VARME		
Skript	Effektbehov romnivå	Effektbehov bygningsnivå	Dimensjonering
Skript 03	✓	✓	
Skript 06	✓	✓	✓
Skript 07	✓	✓	✓
Skript 10	✓	✓	✓

Eksemplene ovenfor illustrerer hvilken fleksibilitet Dynamo gir, og setter lys på hvorfor det er hensiktsmessig å ha kompetanse innen bruk av verktøyet innad i bedriften. Kompetansen er en forutsetning for effektiv gjenbruk av utviklede skript.

9.7 Dynamos påvirkning på samhandling på tvers av bedrifter

Dynamo øker kravet til god samhandling mellom de ulike fagdisiplinene, og byggherren har et ansvar for å legge til rette for dette. I dagens situasjon, kan det tenkes at ulike bedrifter ikke er interessert i å bruke ekstra ressurser for å legge til rette for andre, som kan være i direkte konkurranse i markedet, og dermed spoleres mye av den samhandlingen som kreves, for at effektiviseringsverktøy skal brukes til sitt fulle potensiale. Hvordan byggherren kan legge til rette for dette, blir ansett som en digresjon i forhold til de rammer som er satt for oppgaven, så det vil ikke bli diskutert mer i dybden her. Allikevel er dette en problemstilling som oppfattes som relevant og det oppfordres at andre tar opp hansken og undersøker problemstillingen nærmere.

9.8 Videreutvikling per dags dato

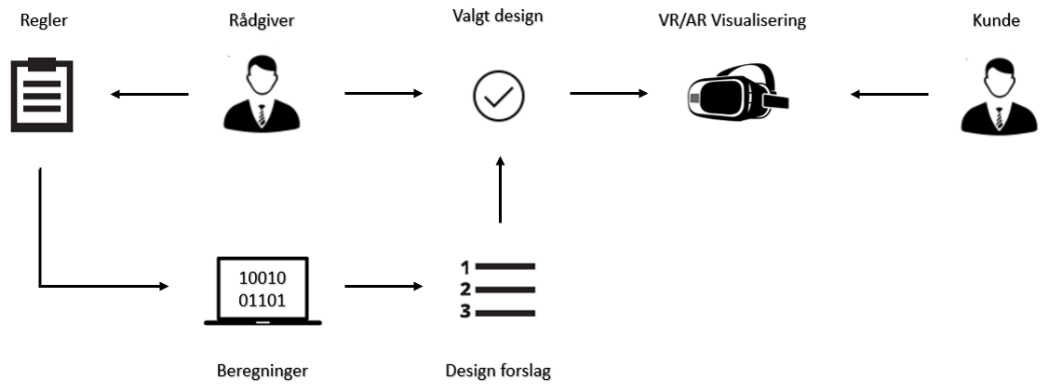
En del av skriptene har sine begrensninger. Dette skyldes manglende noder til å oppnå ønsket funksjonalitet, samt manglende fundament innen programmering. Videreutviklingen kan legge fokus på å løse problemstillinger innen de skriptene som gruppen har arbeidet med, hvor fokuset faller på redusert avhengighet.

Videreutviklingen bør også ta stilling til effektiv programmering som reduserer iterasjonstiden, som øker vesentlig når man kjører skript på store BIM modeller. Arbeid med prosjektspesifikke modeller resulterer i lengre iterasjonstid, enn ved utviklingen av skriptene. Dette er grunnet at testmodeller i utviklingsfasen ikke er like komplekse som prosjektspesifikke modeller. Her har dataingeniører også en mulighet til å komme med bidrag, grunnet solid bakgrunn med koding.

9.9 Videreutvikling i fremtiden

Bruk av Dynamo gir oss en dynamisk måte å manipulere data på. Hvilken påvirkning vil dette ha på byggebransjen i fremtiden? Måten verktøy som Dynamo blir benyttet på per dags dato, er et samspill mellom maskin og menneske, hvor man endrer input verdier og datamaskinen utfører bearbeidelsen av informasjonen. På denne måten kan vi kjøre ulike inputverdier, få et resultat og kjøre dette ut som et forslag rimelig raskt.

I fremtiden vil vi kunne definere mer kompliserte algoritmer hvor AI-baserte noder kan være en mulighet. Dette åpner opp for å legge opp reglene og få datamaskinen til å gjennomføre design basert på krav. Eksempelvis kan man ønske x antall forslag på ventilasjonsanlegg, hvor man tar hensyn til SFP-faktor og installasjonskostnader. Datamaskinen genererer forslag basert på input og rådgiver kan opprette funksjon basert på energieffektivitet og installasjonskostnad, hvor riktig balanse i forhold til byggherrens ønsker velges basert på fremstillingen av data. Videre benyttes VR/AR til å visualisere valgt løsning for kunden, som illustrert i *Figur 35.*



Figur 35 – Fremtidig prosjektering

Det finnes nå mye dokumentasjon om effektiv prosjektering ved hjelp av Dynamo, og kanskje kan det forventes fremover at fokus flyttes over fra å dokumentere hvilke muligheter dette gir, og heller legger vekt på å gjennomføre dette i praksis. Store byggherrer som Statsbygg har et stort ansvar her, men bransjen ellers burde også forsøke å implementere dette inn i sine bedrifter. Det ligger et stort konkurransefortrinn i å beherske Dynamo i sine prosjekter. Det er rimelig å anta at dette skaper et skille mellom de som henger med i utviklingen, og de som ikke følger med på utviklingen.

10 Tilbakemelding

Det ble holdt et møte med prosjektledere og ledelsen i VVS avdelingen hos SWECO. Her ble resultatene presentert med diskusjoner rundt hvert skript. Vurderingen var i forhold til skript med stor avhengighet av andre fagdisipliner og skript med liten avhengighetsgrad. Skript med stor

avhengighetsgrad er et utfordrende område ettersom det krever både god datastruktur i flere fagdisipliner, samt god kommunikasjon i prosjektet.

Videreutvikling bør fokusere på noen utvalgte skript som har lav avhengighetsgrad, er tidsbesparende og har høy grad av gjenbruk. Et eksempel er *Skript 13* som har elementer som ikke blir tekstet og noen forekomster av dobbel teksting på vertikale føringer. Dette krever en mer omstendelig kontrollsjekk i etterkant. En videreutvikling for å eliminere lignende svakheter er nødvendig for å sikre god funksjonalitet i prosjekter.

Skript som *Skript* 06 har stort potensiale, men burde muligens begrenses til et rom og ikke beregning av alle rom samtidig. Dette gir muligheten til å undersøke resultatet og gyldigheten av skriptet fortløpende. *Skript 02* har også stort potensiale men faller også inn under de skript hvor prosesser bør defineres tydelig for brukeren.

11 Konklusjon

Dynamo er et verktøy som har et stort potensial for å effektivisere oppgaver innen VVS prosjektering. Dette er under forutsetninger at man har riktig kompetanse i bruken av verktøyet innad i bedriftene.

Superbrukeren må ha tilstrekkelig kompetanse i bruken av Dynamo for å utvikle logiske skript med kortest mulig iterasjonstid. Videre er det nødvendig med oversikt over hvilke fallgruver programmet har og hvordan man skal sikre funksjon ved alternative tilnærmingsmetoder til en eventuell problemstilling. Generell kompetanse blant andre i prosjekteringsteamet er et element som kan bidra til å effektivisere kommunikasjonen i forhold til utvikling av skript. Det er ikke en nødvendighet, men vil danne et større eierskap til teknologien som motiverer involvering.

Et annet viktig poeng, er organisatorisk struktur i forhold til bruk av teknologien. For at teknologien skal bli benyttet effektivt er det viktig at ledelse støtter opp under bruken i prosjekter. Det er både nødvendig og viktig at det settes opp en prosjektmal, som skal sikre at nødvendig informasjon som utføring, fremgangsmåte og parametere, videreføres til modellen for å sikre effektiv samhandling og utførelse av prosjekteringsoppgaver. Fokus på datastruktur er essensielt.

Teknologien er moden for bruk i en rekke områder men vil kreve forskning, videreutvikling og jevnlig vurdering. Man kan trekke samme paralleller til overgangen fra CAD til BIM i forhold til implementering av nye arbeidsmetoder samt oppnåelse av effektiv bruk i forhold til kompetansesatsing. Dynamo vil derimot dra nytte av god datastruktur i BIM-en som øker fokus på viktige forbedringsområder i bygningsinformasjonsmodelleringen grunnet økt tidsbesparelse ved bruk av skript. Dette vil gagne samarbeidet og effektiviteten i prosjektet på flere områder, og som vi har undersøkt i gjennom hele denne oppgaven, vil dette være kritisk for at skript og andre effektiviseringsprosesser som omhandler BIM vil kunne brukes i fremtiden. Vi trenger å sette krav til hverandre i et prosjekteringsteam, noe som bør starte i tidligfase, slik at selve «grunnmuren» i prosjektet er solid, som kan resultere i en økonomisk gevinst.

DEL 2 – Programteknisk

12 Skript og utforming

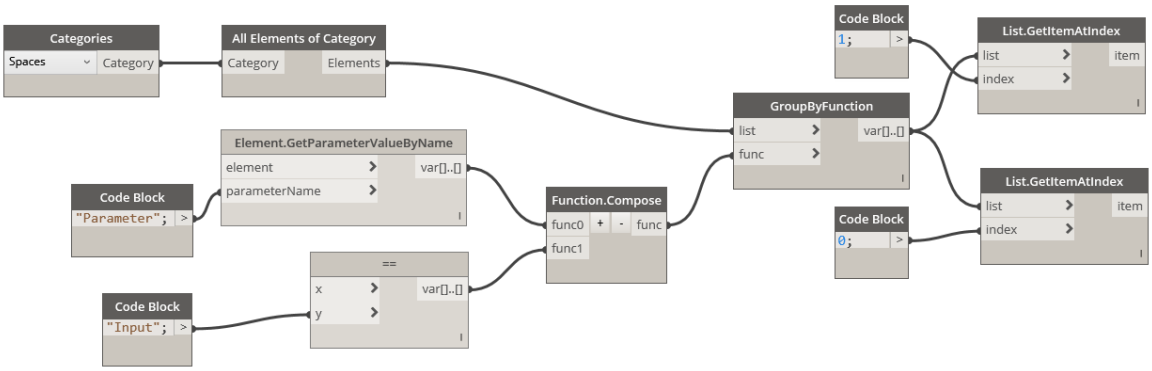
Dette kapittelet tar for seg metodikk og metode for utforming av skriptene på et programteknisk nivå.

12.1 Metode i skriptutvikling

Fokuset for bruk av Dynamo til å effektivisere prosjekteringsoppgaver har vært på behandling av informasjon i modellen og ikke til å skape geometri. I VVS faget er det store mengder med data som må bearbeides, noe som krever effektiv sortering av data i modeller. Følgende er en oversikt over de ulike oppsett av noder som er benyttet i utvikling av skript. Det presenteres ulike metoder for filtrering benyttet til å sortere data effektivt.

12.1.1 Function Compose

Følgende metode er en tilnærming brukt til å filtrere elementer i skript. Her har man muligheten til å koble funksjoner som blir videre gruppert i forhold til krav. *Figur 36* illustrerer hvordan kategorier (i dette tilfellet «Spaces») blir gruppert i forhold til en definert input.

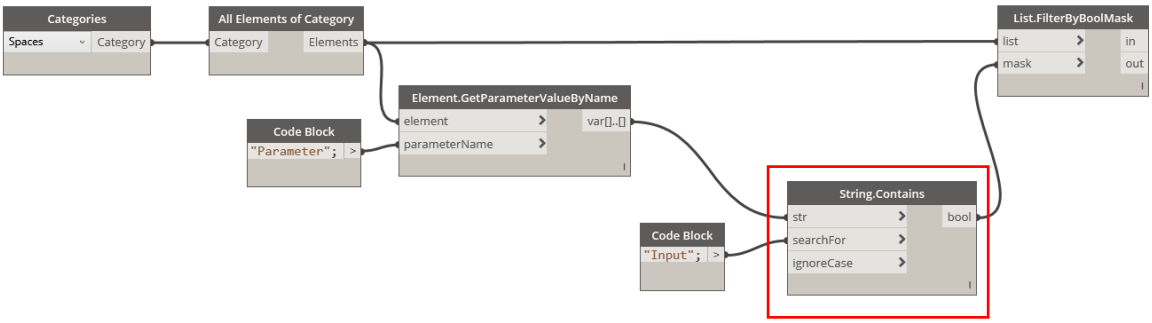


Figur 36 – Illustrerer hvordan kategorier blir gruppert i forhold til definert input

Dersom parameteren i kategorien inneholder den definert inputen, så blir elementene gruppert i liste 0, hvor det benyttes «List.GetItemAtIndex» node for å ta ut den ønskede listen. Funksjonene kan defineres med en rekke logiske eller matematiske operatører i kombinasjon med andre noder.

12.1.2 String Contains

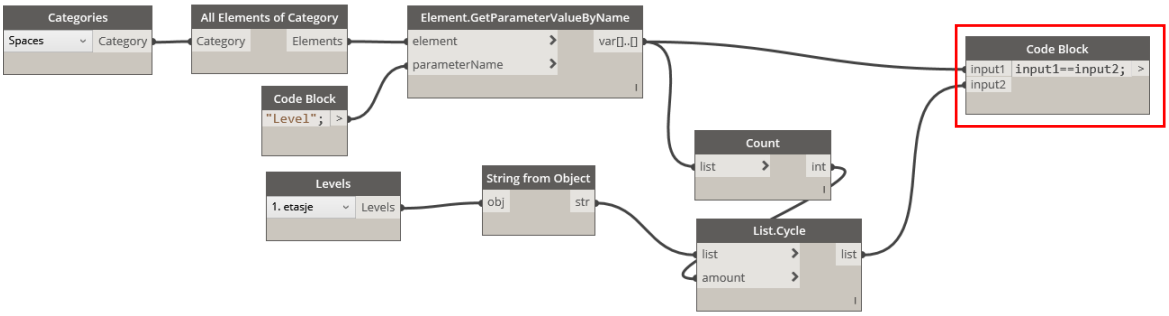
«String.Contains» noden gir muligheten til å undersøke om innholdet i et objekt har en gitt tekst eller deler av en tekst i seg. *Figur 37* viser et eksempel på bruk av noden.



Figur 37 – Illustrerer bruken av «String.Contains»

12.1.3 Code Blocks

Code Blocks kan benyttes til å utvikle alt fra enkle til avanserte funksjoner. *Figur 38* nedenfor illustrerer et enkelt eksempel på bruk av «Code Block» til å opprette en enkel funksjon.



Figur 38 – Illustrerer bruken av Code Blocks

Den enkle koden i «Code Block» noden markert med rød ramme, sjekker om listen inneholder 1. etasje i parameteren hvor etasjen er definert. Vi får en rekke «True» og «False» verdier som kan videre filtreres. Her er det også benyttet en «List.Cycle» node i kombinasjon med en «Count» node som sørger for at alle elementer blir sjekket opp mot 1. etasje. Et annet eksempel på bruk av «Code Block» illustreres i *Skript 02* hvor ulike luftmengdeberegninger blir gjennomført.

Avansert bruk av skriping ved bruk av «Code Block» ble ikke gjennomført av gruppen ettersom dette krever større kompetanse med koding.

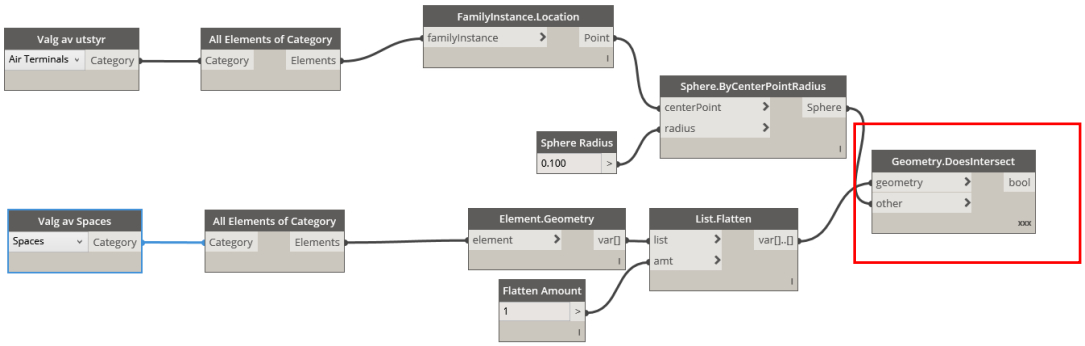
12.1.4 Lacing

«Lacing» er et begrep i Dynamo som definerer hvordan to eller flere lister med ulikt antall elementer er prosessert, basert på en definert kommando. Det er en veldig kraftig funksjon som har en rekke anvendelsesområder. I Dynamo har vi følgende lacing muligheter:

1. Shortest: Hvert element i den korteste listen er sjekket opp mot hvert element av det første, av x antall elementer i den lengre listen.
2. Longest: Samme funksjon som Shortest, men det siste elementet i den korteste listen blir også sjekket opp mot de gjenværende elementene i den lengre listen.
3. Cross Product: Hvert av elementene i hver liste blir sjekket opp mot hverandre.

Et eksempel på hvordan gruppen har benyttet Lacing er illustrert i *Figur 39* markert med rød ramme. Her blir ventiler gjort om til små sfærer og sjekket opp om de eksisterer i «Spaces» som er gjort om

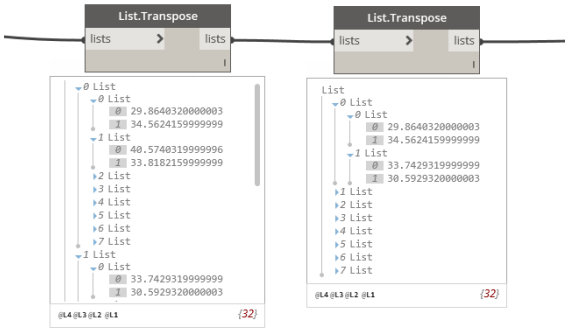
til geometri. «Geometry.DoesIntersect» noden har kryssprodukt Lacing (markert med tre krysser i bunn av noden) og alle ventiler blir sjekket mot tilhørighet i alle «Spaces».



Figur 39 – Illustrerer funksjonen av lacing

12.1.5 List Transpose

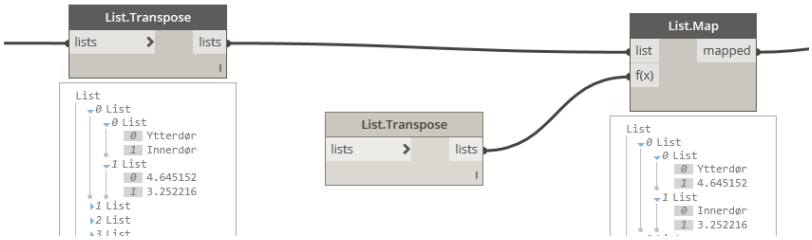
«List.Transpose» er en node i Dynamo knyttet til behandling av lister. Denne brukes i sammenheng med listehåndtering, som skifter rader og rekker i en liste av lister. Denne noden blir ofte brukt for å sikre at riktig data blir sortert på ønsket måte, for å koble ulike lister av data opp mot hverandre. Hvis det er ulike listelengder, blir nullverdier satt inn i listene for å alltid sikre kvadratiske lister.



Figur 40 – Illustrerer funksjonen av List Transpose

12.1.6 List Map

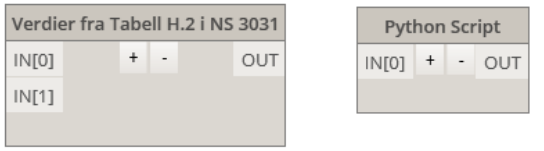
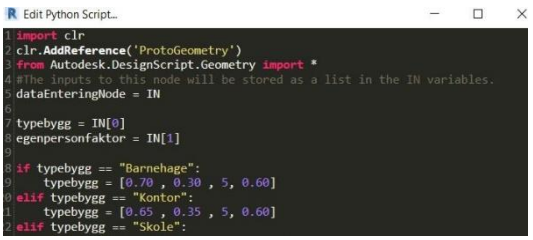
«List.Map» er en funksjonsnode, hvor det anvendes en funksjon til alle elementer i listen, og videre genererer en ny liste basert på resultatene av funksjonen. Det er mulig å bruke flere ulike noder som funksjon etter ulike behov, men som vist i *Figur 41*, er det satt i en praktisk sammenheng med å anvende en «List.Transpose» node som funksjon til en liste, som kobler data fra listen med element og areal i ulike lister, sammen i en ny liste.



Figur 41 – Illustrerer bruken av List.Map

12.1.7 Bruk av Python i Dynamo

Bruk av programmeringsspråket Python ble benyttet på et enkelt nivå i utviklingen av skriptene i denne oppgaven. I Dynamo er det en egen node som heter «Python Script». Ved å dobbeltklikke på noden kommer man inn i grensesnittet hvor man kan kode i Python. Avanserte brukere av Dynamo benytter gjerne «Code Block» og «Python Script» i utviklingen av sine skript, som gir funksjoner som krever mindre prosesseringstid, enn ved bruk av definerte noder i Dynamo.

Python node og bruk i *Skript 02*	**Tekstkode i Python node**
	

Figur 42 – Bruk av Python i Dynamo

12.2 Kvalitetssikring av skript

For å kvalitetssikre skriptene og finne eventuelle svakheter, har det blitt modellert egne testmodeller, i tillegg til å bruke modeller som har blitt gjort tilgjengelige for oss i ulike emner under studieløpet på OsloMet.

Figur 43 illustrerer en tilbakemelding på kvalitetssikring av alle skript, samt en beskrivelse og hvilken testmodell som ble benyttet ved testingen.

Kvalitetssikring av skript		
Skript beskrivelse	Test modell	Status
01 Overføring av navn og nummer fra ARK Room tuk VVS Soaces	Simpel arkitektmodell	Ok
02 Luftmengdeberegninger	testbygg_vvs + testbygg_ark	Ok
03 Overføring av parametere fra Spaces til ventiler	testbygg_vvs + testbygg_ark	Ok
04 Fordeling av total luftmengde til ventiler	testbygg_vvs + testbygg_ark	Ok
05 Tilbakeføring av luftmengder til Revit	testbygg_vvs + testbygg_ark	Ok
06 Effektbehovsberegninger del 1,2,3 og 4 tilknyttet areal	Simpel arkitektmodell	Ok
07 Ventilasjonsvarmetap på romnivå	testbygg_vvs + testbygg_ark	Ok
08 Kvalitetskontroll av trykkfallsberegninger i MagiCad	sykehjem_RIVv	Ok
09 Sanitær - Avløp - Utstyrtelling og beregninger av sannsynlige vannmengder	testbygg_vvs + testbygg_ark	Ok
10 Kontroll av parametere i Spaces	testbygg_vvs	Ok
11 Kollisjonskontroll	Kollisjons_modell	Ok
12 Farging av sprinkler i forhold til avstand fra vegg	Sprinkler testmodell + Linket ARK fil	Ok
13 Effektiv dimensjonsteksting på kanaler	VVS_testmodell	Ok

Figur 43 – Kvalitetssikring av skript

En annen viktig faktor som ble lagt merke til ved kvalitetssikring, er at dersom noen av skriptene har benyttet seg av tilleggspakker som ikke har blitt installert, vil ikke skriptet fungere. Dermed er det krav til å inneha alle tilleggspakker som blir presentert i *Tabell 6,* i kapittel *Nødvendige tilleggspakker*.

Skriptene har blitt testet i både Revit versjon 2017 og 2018, for å undersøke funksjonalitet på tvers av versjoner. Det har vist seg at noen skript har blitt påvirket av overgang til ny versjon som ugyldiggjorde skriptet. Å kartlegge slike påvirkninger har vært viktig for å tilpasse løsninger som sikrer

fremtidig funksjonalitet på tvers av versjoner. Videre er en bevisstgjøring på at dette er viktig kunnskap med tanke på en kvalitetssikring av skript, når en ny versjon av Revit og/eller Dynamo kommer på markedet og blir benyttet i prosjekter.

12.3 Dynamo Player

For at skriptet skal være mest mulig oversiktlig for brukeren i det forenklede brukergrensesnittet i Dynamo Player, er det viktig at den som utvikler skriptet tenker på hvilke noder som blir synliggjort. Noder som ikke vises i Dynamo Player er «Code Blocks», hvor man både kan benytte tall eller tekst som input. Noder som vises er blant annet «String» og «Number». «Code Block» bør benyttes til filtrering og bør da ikke gis mulighet til å endres av bruker. Derimot bør all input som bruker selv skal definere, være tilgjengelig i Dynamo Player ved bruk av «String» og «Number» noder i skriptene. Det påpekes igjen at valgmulighet for input i Dynamo Player er begrenset til Revit 2018 versjonen.

12.4 Nødvendige tilleggspakker

Flere av skriptene benytter seg av tilleggspakker. Det er viktig at disse pakkene er installert på maskinen hvor skriptene skal kjøres. Mangel på disse vil medføre svikt i funksjonalitet, hvor *Tabell 6* gir oversikt over hvilke Dynamo pakker, samt versjoner som ble benyttet i utformingen av skriptene og må installeres på maskin før test.

Tabell 6 – Liste over tilleggspakker som er benyttet

Tilleggspakke	**Versjon**
archi-lab.net	2018.0.8
LunchBox for Dynamo	2017.10.4
SteamNodes	1.2.4
Clockwork for Dynamo 1.x	1.31.1

13 Skript

I dette kapitlet vil alle skriptene bli presentert. Grundige beskrivelser av skriptene finnes etter hvert emne, samt en oversikt over alle skript i *Tabell 7.*

Tabell 7 – Tabell liste over skript

Skript	Beskrivelse
01	*Overføring av navn og nummer fra ARK Rooms til VVS Spaces*
02	*Luftmengdeberegninger*
03	*Overføring av parametere fra «Spaces» til ventiler*
04	*Fordeling av total luftmengde til ventiler*
05	*Tilbakeføring av luftmengder til Revit*
06	*Hente ut nødvendige areal tilknyttet Spaces/Rooms*
07	*Ventilasjonsvarmetap på romnivå*
08	*Kvalitetskontroll av trykkfallsberegninger i MagiCAD*
09	*Sanitær – Avløp – Utstyrstelling og beregning av sannsynlige vannmengder*
10	*Kontroll av parametere i Spaces*
11	*Kollisjonskontroll*
12	*Farging av sprinkler i forhold til avstand fra vegg*
13	*Effektiv dimensjonsteksting av diametere for kanaler*

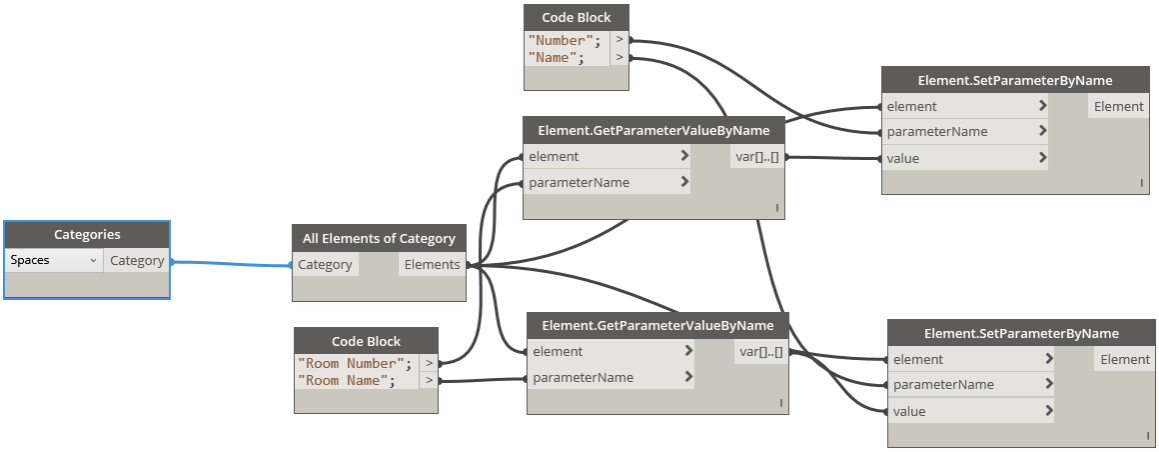
13.1 Luftmengder

Skript 01 til 05 er en samling av flere skript med avhengighet av hverandre. Disse brukes til å klargjøre og utføre luftmengdeberegninger, samt fordele disse på ventiler i modellen.

13.1.1 Skript 01 – Overføring av navn og nummer fra ARK Rooms til VVS Spaces

Følgende skript tar for seg overføring av riktig navn og nummer fra rommene i arkitekt modellen til «Spaces» i VVS modellen. Problemstillingen i forhold til nummerering og navngiving av «Spaces» blir illustrert først. Vi begynner med testmodellen, som er en ARK modell med 4 etasjer, hvor rommene har blitt navngitt og nummerert. Modellen åpnes som linket modell i RIV modellen, hvor det tekniske modelleres. Etter at den linkede modellen har blitt merket som avgrensende i forhold til geometri, så plasserer vi ut «Spaces» basert på dette.

Dette skriptet tar stilling til romnavn og nummer som ligger i «Spaces», men ikke kommer med i mengdeuttak og planvisning. Skriptet overfører denne informasjonen til parametere i «Spaces» fra «Room Number» til «Number» og «Room Name» til «Name».



Figur 44 – Skript 01 Overføring av navn og nummer fra ARK Room til VVS Spaces

Skriptet tar utgangspunkt i valgte kategorier, som er «Spaces», og får tak i alle instanser i modellen. Deretter får noden «Element.GetParameterValueByName» tak i de ønskede parameterne i hvert objekt, og overfører dette videre til valgt parameter i «Spaces» ved bruk av «Element.SetParameterByName».

Når alle «Spaces» har blitt oppdatert med riktig nummer og navn basert på ARK modellen, gir dette oss et godt grunnlag til å arbeide effektivt med videre delprosesser.

13.1.2 Skript 02 - Luftmengdeberegninger

Når alle «Spaces» har riktig nummerering og navngiving, så er neste delprosess å beregne luftmengdene til rommene. Formålet med dette skriptet er å beregne maksimum, minimum og gjennomsnittlige luftmengder, og tilbakeføre disse verdiene tilbake til «Spaces» under definert parameter.

Følgende parametere må defineres for å benytte funksjonen i skriptet. Alt grupperes under kategorien «Spaces».

Tabell 8 – Parametere som må være definert

Navn	Discipline	Type of parameter	Input
Tilstedeværelse	Common	Text	Opphold **ELLER** «Tomt felt»
Personer	Common	Number	-
Min. Luftmengde	HVAC	Air Flow	[m³/h]
Maks. Luftmengde	HVAC	Air Flow	[m³/h]
Gj.snittlig Luftmengde	HVAC	Air Flow	[m³/h]
Funksjon	Common	Text	Annet **ELLER** «Tomt felt»

Gjennomgang av skriptet

<u>Del 1 – Inputverdier og parametere</u>

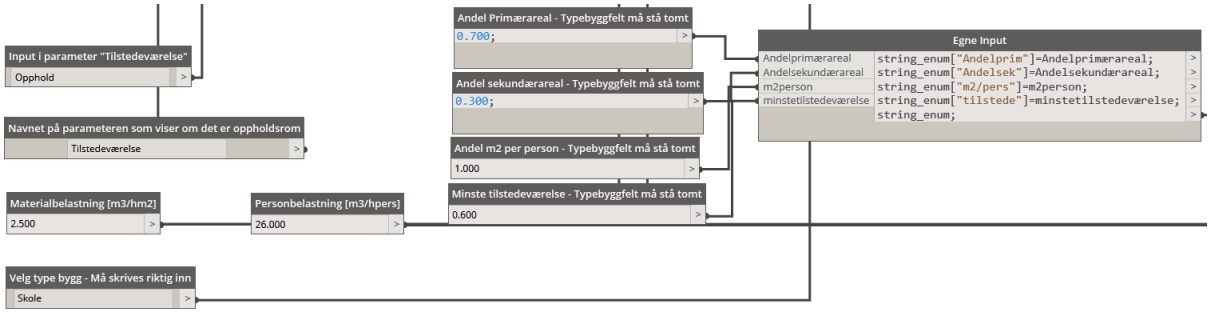
I første delen av skriptet definerer vi hvilken type bygg dette er, basert på Tabell H.2 i NS 3031. Dette danner forutsetninger for å få ut listene som inneholder verdier for den spesifikke bygnings-kategorien som blir tatt med i videre beregninger. Skriptet tar utgangspunkt i verdier for «Persontetthet» og «Minste tilstedeværelse». Verdiene for «Minste Primærareal» og «Største Sekundærareal» blir filtrert vekk, men inkluderes dersom man ønsker å bygge ut skriptet i fremtiden hvor det er behov for disse parameterne.

Bygningskategori	Minste primær-areal	Største sekundær-areal	Minste persontetthet, primærareal m²/person	Minste tilstedeværelse, primærareal [b]
Småhus	_ [a]	_ [a]	_ [a]	100 % [a]
Boligblokk	_ [a]	_ [a]	_ [a]	100 % [a]
Barnehage	70 %	30 %	5	60 %
Kontorbygning	65 %	35 %	5	60 %
Skolebygning	70 %	30 %	2,5	60 %
Universitets- og høgskolebygning	70 %	30 %	4	70 %
Sykehus	75 %	25 %	5	70 %
Sykehjem	75 %	25 %	5	70 %
Hotellbygning	60 %	40 %	6	50 %
Idrettsbygning	80 %	20 %	5	60 %
Forretningsbygning	70 %	30 %	4	75 %
Kulturbygning	70 %	30 %	4	60 %
Lett industribygning, verksted	70 %	30 %	4	60 %

Figur 45 – Tabell H.2 i NS 3031 [5]

Andre inputverdier er materialbelastning [m^3/hm^2] og personbelastning [m^3/h] og «Opphold» som blir sendt til parameteren «Tilstedeværelse». Denne parameteren danner forutsetning for å dele opp rom i forhold til beregninger for primærrom og sekundærrom. Det er da viktig at parameteren er lagt til i «Spaces» og input «Opphold» er definert i forkant før dette skriptet blir kjørt.

Skriptet gir også mulighet for egen input for verdier av persontetthet, minste primærareal, største sekundærareal og minste tilstedeværelse. Dersom dette er ønskelig **SKAL** feltet i noden «Velg type bygg» stå tomt eller ha en annen input enn definert bygningskategori. Ellers vil ikke egendefinerte verdier bli vurdert i beregningene, men standardiserte verdier for bygningskategorien som er definert i noden.



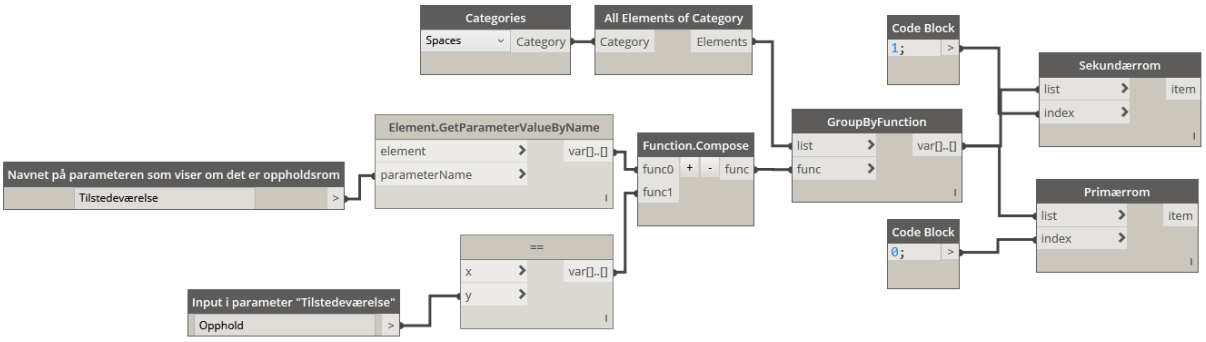
Figur 46 – «Egne input» som en Code Block

«Egne input» noden er en «Code Block» som danner en liste basert på egendefinerte inputverdier og sender denne listen videre til Python koden, som filtrerer lister basert på definert (eller ikke definert) bygningskategori.

<u>Del 2 - Filter</u>

Filterdelen av skriptet skiller mellom hvilke rom som skal ta stilling til materialbelastning og hvilke rom som skal ta stilling til material og personbelastning. Filteret filtrerer også de rom som man ikke ønsker beregning for. Ettersom det kan være spesialrom i bygget som har en rekke andre prosesser å ta stilling til i forhold til luftmengdeberegninger, er det viktig at man har en måte å skille disse rommene fra rom med standardiserte beregninger.

Metoden benyttet for å filtrere de ulike rommene basert på definerte parametere i hele skriptet, er bygd opp på samme logikk som illustrert i *Figur 47*. «Spaces» blir sjekket om de tilfredsstiller en definert input i en parameter og blir filtrert deretter.



Figur 47 – Filtrering av Spaces

I denne delen filtreres «Spaces» ut i forhold til hva som er primærrom og hva som er sekundærrom. Denne delen sjekker om parameteren «Tilstedeværelse» har input «Opphold» eller ikke. De «Spaces» som har denne parameteren definert, blir sendt videre til primærrom delen og de som ikke har den definert, blir sendt til beregninger for sekundærareal som bare tar stilling til materialbelastningen.

På samme måte blir «Spaces» filtrert i forhold til en parameter kalt «Funksjon». Dersom «Annet» er inputen i «Funksjon» for et «Space», blir dette rommet sendt videre til en nullstilling av luftmengder.

<u>Del 3 – Sekundærromberegninger</u>

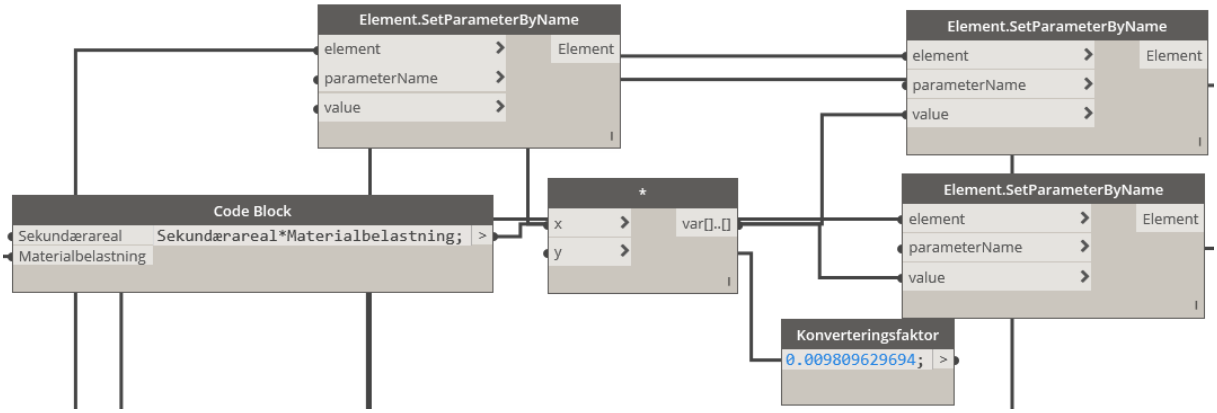
I denne delen blir tilluftsmengden beregnet på de rommene som skal ha standardisert beregning, og rom som har spesielle funksjoner som krever en annen beregningsmetode, blir utelatt. Alle parametere nevnt ovenfor er koblet til sine respektive plasser i inputen «parameterName» på «Element.SetParameterByName».

En viktig ting å ta stilling til her er konverteringsfaktoren som er en del av beregningene. Problemet er at Dynamo ikke forstår strømningstekniske enheter når de blir sendt tilbake til Revit, og sender verdien multiplisert med en faktor som gir feil verdier. Løsningen på dette problemet var å benytte *Formel 1*.

$$k \cdot \dot{q}_{Dynamo} = \dot{q}_{Revit} \rightarrow k = \frac{\dot{q}_{Revit}}{\dot{q}_{Dynamo}} = 101.9406472205209$$

Formel 1 – Konvertering til m³/h

Dette betyr at vi må multiplisere 1/k, med den verdien som ser tilsynelatende riktig ut i skriptet for å få ønsket verdi. Faktoren gjelder for konverteringer til m^3/h og ikke andre enheter. Forløpet er illustrert i *Figur 48*.

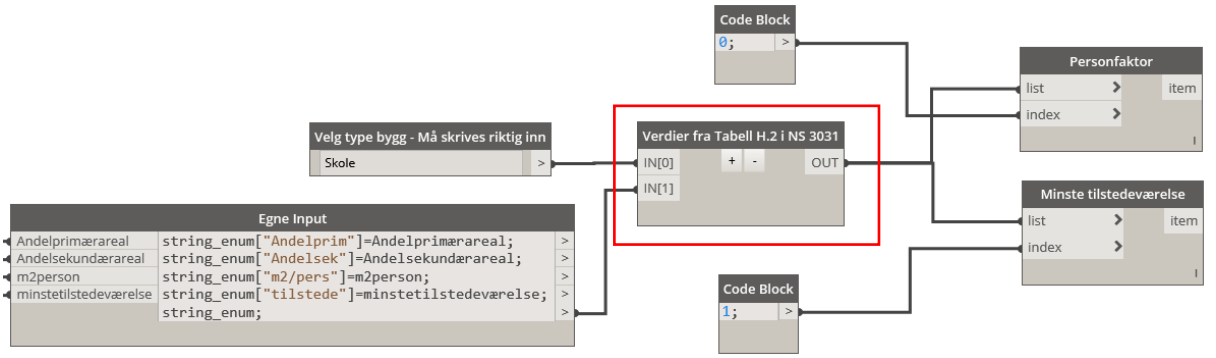


Figur 48 – Illustrering for konvertering

Del 3 - Primærromberegninger

Denne delen har samme prinsipp som beregningene i sekundærromdelen. Forskjellen her er at beregningene også tar stilling til antall personer og minste tilstedeværelse. Det er også et annet element som er nytt her. Python Script noden inneholder et eget program, som sender verdier videre basert på om du har valgt en bygningskategori i input delen av skriptet, eller om du har satt egendefinerte verdier. Koden for skriptet er å finne i *Vedlegg IV*.

Nodene «Personfaktor» og «Minste tilstedeværelse», henter ut verdier fra listen som kommer ut fra Python skriptet for videre beregninger. *Figur 49* illustrer Python noden markert med rød ramme.



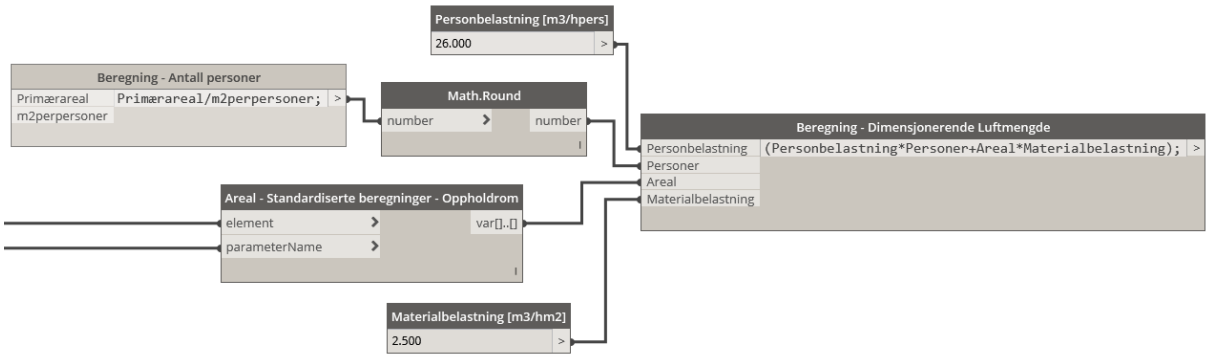
Figur 49 – Illustrering av Python noden

Formlene som er benyttet for beregninger av luftmengdene fremkommer i *Tabell 9*.

Tabell 9 – Oversikt over formler og symbolforklaring

Formler	**Symbolforklaring**
$\dot{q}_{Maks} = n_{pers} \cdot k + A \cdot m$ $\dot{q}_{Snitt} = n_{pers,snitt} \cdot k + A \cdot m$ $\dot{q}_{Min} = A \cdot m$	$k = personbelasntingsfaktor\ [m^3/h \cdot person]$ $m = materialbelastningsfaktor\ [m^3/h \cdot m^2]$ $A = arealet\ til\ rommet\ [m^2]$ $n_{pers,snitt} = gjennomsnittelig\ antall\ personer$ $n_{pers} = antall\ personer$

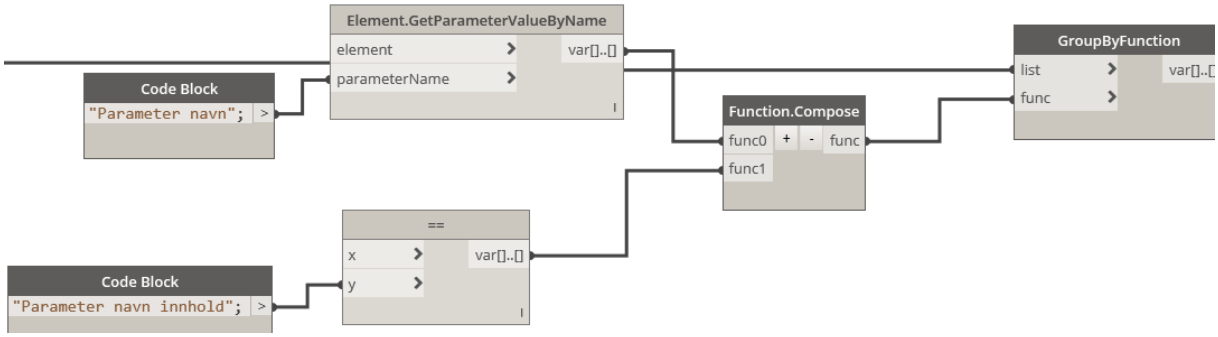
Figur 50 illustrerer forløpet i forhold til beregninger. På samme måte som i beregningene for sekundærrommene, blir de beregnede verdiene sendt tilbake til alle definerte parametere i «Spaces» som er i modellen. Figuren illustrerer beregningene av dimensjonerende luftmengde for ett rom. Samme fremgangsmåte er benyttet for «Gjennomsnittelig Luftmengde» og «Min Luftmengde», med gjeldende inputverdier.



Figur 50 – Illustrering av beregningene av dimensjonerende luftmengde for et rom

Feil filtrering ved bruk av GroupByFunction

Det ble registrert at skriptet hadde problemer med å filtrere informasjonen riktig når oppsett i *Figur 51* ble benyttet.



Figur 51 – Feil filtrering ved bruk av GroupByFunction

Dette oppsettet ga problemer i måten elementer ble filtrert i skriptet. Logikken tilsier at dersom «Parameter navn» inneholder «Parameter navn innhold», så vil listen på indeks 0 inneholde elementer som inneholder «Parameter navn innhold» og indeks 1 innehar resten av elementene. Dette vekslet Dynamo på i forhold til Revit versjoner. Listene ble feil filtrert i forhold til indeks, hvor løsningen var å ta den indeksen som inneholdt de rette elementene videre.

I utgangspunktet er dette feil i forhold til logikk, men gav riktige resultater i Revit 2017 versjonen, grunnet endring av indeks. Derimot inneholdt skriptet to filter i forhold til sekundærrom beregninger og primærrom beregninger. Feilen som oppstod i Revit 2017, ga feil filtrering og primærfilteret fikk derimot feil indekser i forhold til logikk. Ved å endre indeksen kan en løse dette problemet avhengig av Revit versjon.

13.1.3 Skript 03 – Overføring av parametere fra «Spaces» til ventiler

Følgende parametere skal overføres fra «Spaces», ved å benytte dette skriptet:

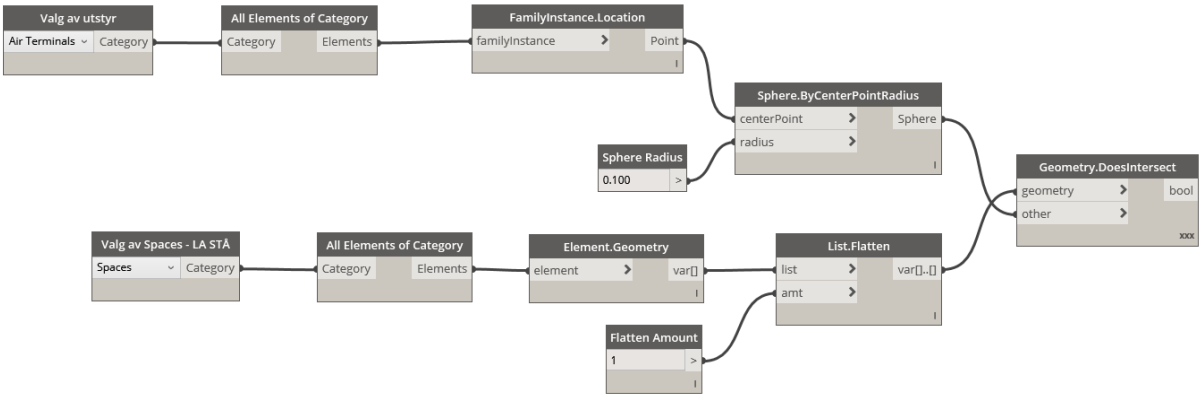
- Name
- Number
- Maks Luftmengde

Parametere til ventilene defineres på følgende måte:

Tabell 10 – Ventilers parametere

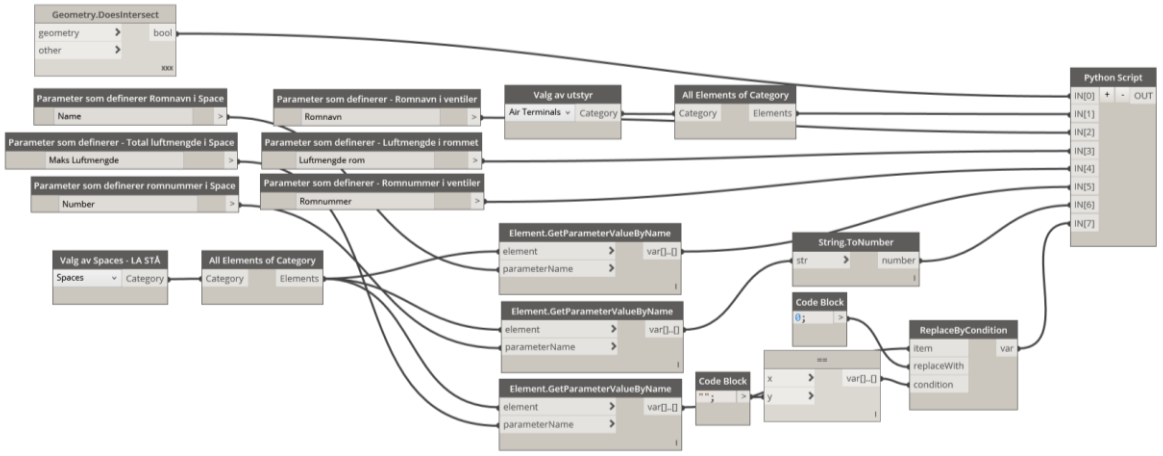
Navn	Discipline	Type of parameter	Input
Luftmengde rom	HVAC	Air Flow	[m^3/h]
Romnummer	Common	Integer	Nummer
Romnavn	Common	Text	Tekst

Skriptet tar stilling til valgt utstyr, som i dette tilfellet er ventiler, samt «Spaces». Alle ventiler blir omgjort til et punkt som det skapes en kule fra, med noden «Sphere.ByCenterPointRadius». «Spaces» blir omgjort til geometri, og det undersøkes om geometriene overlapper hverandre med en «Geometry.DoesIntersect» node med et kryssprodukt. *Figur 52* viser logikken i oppbygningen av denne delen.



Figur 52 – Logisk oppbygning av del 1, Skript 03

Parametere som skal hentes fra «Spaces», defineres og overføres til ventilene som befinner seg i de ulike rommene. Til slutt går listene over «Spaces» og ventiler, samt tilhørende parametere, inn i en Python node. Python noden tar også stilling til resultatene fra «Geometry.DoesIntersect» noden. *Figur 53* viser alle koblingene som er relevante for Python noden. «Element.GetParameterValueByName» noden henter inn verdier fra «Spaces» for overføring. «ReplaceByCondition» noden blir benyttet ettersom tomme felt i parameter «Maks Luftmengde» gjør at Python Scriptet ikke klarer å tildele verdier til ventilene. Tomme felt blir da erstattet med verdien 0 m^3/h.

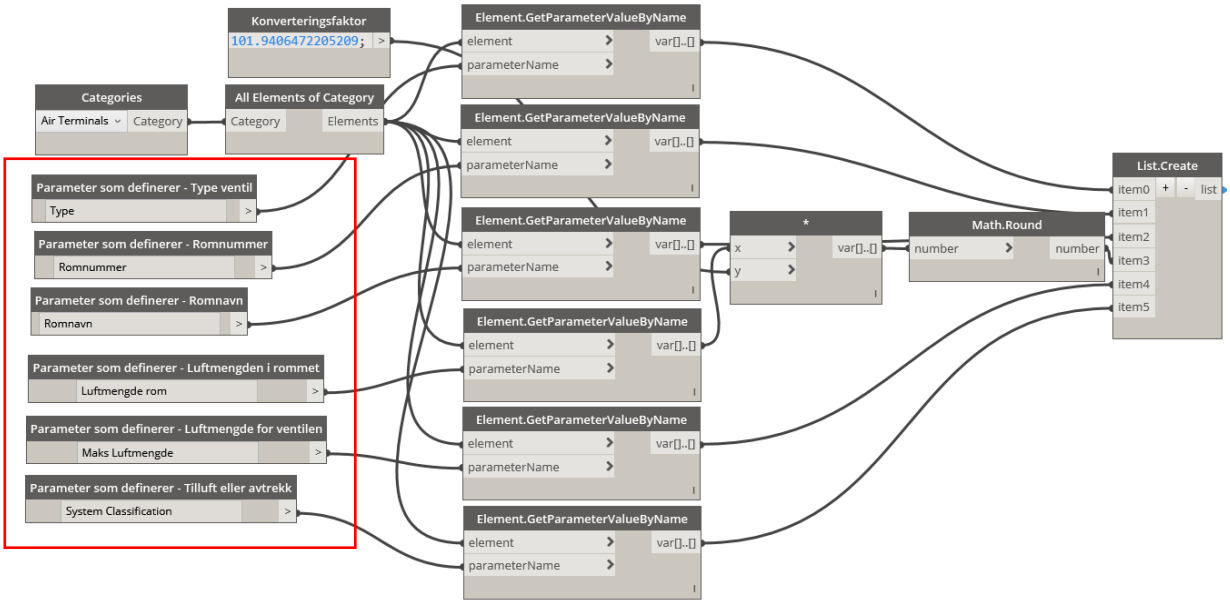


Figur 53 – Alle koblinger som er relevante for Python noden for skriptet

I Python noden blir listene sjekket opp mot hverandre i forhold til et filter. Dersom en ventil er tilstede i et rom, vil den få overført verdien til ønsket parameter. For oversikt over skriptkoden henvises det til *Vedlegg V*.

13.1.4 Skript 04 – Fordeling av total luftmengde til ventiler

Data hentes fra ventilen ved å benytte «Element.GetParameterValueByName» noden. Dette gjelder alle ønskede parametere. I dette skriptet tas det stilling til parametere markert i rød boks i *Figur 54*.



Figur 54 – Skript 04, Fordeling av total luftmengde til ventiler

Den transponerte listen sendes som data til «Excel.WriteToFile» noden. Her defineres kolonne- og radnummer, samt fane hvor data skal skrives til.

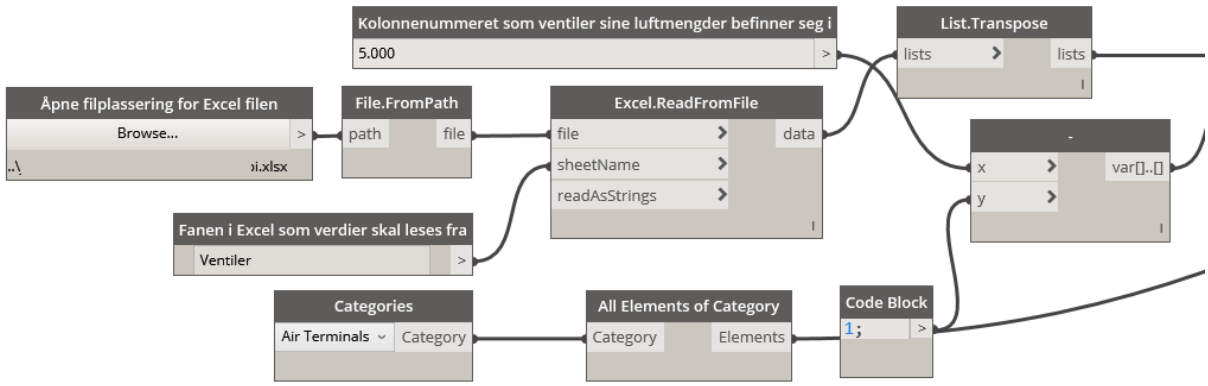
Figur 55 – Definering av kolonne og rad nummer samt fane

Riktig Excel fil velges ved «filePath». Her er det opprettet en malfil, hvor første raden definerer informasjonen med filtervalg. Dette ligger vedlagt i den totale leveransen.

Luftmengden fordeles i «Luftmengde ventil» kolonnen. Når dette er gjort er vi klare for å tilbakeføre den fordelte luftmengden tilbake til ventilene i modellen.

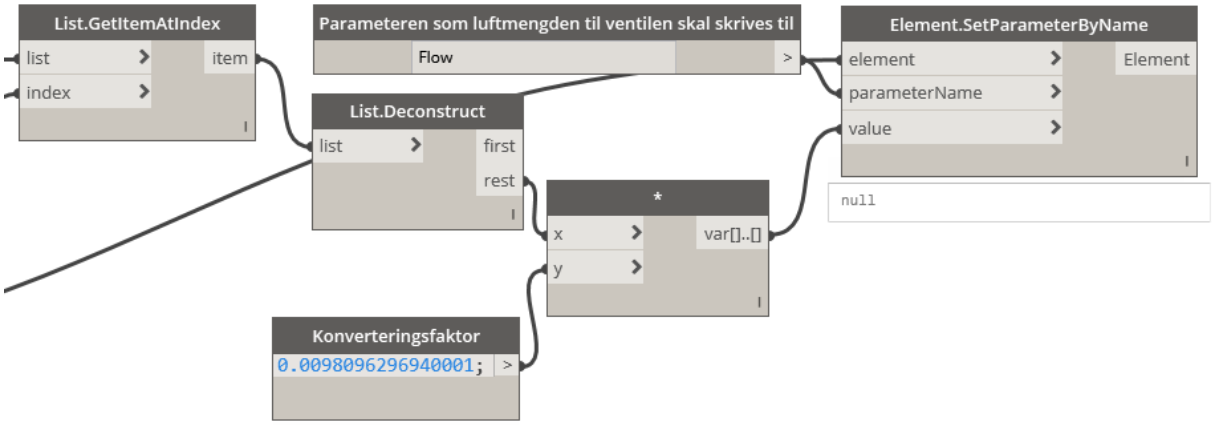
13.1.5 Skript 05 – Tilbakeføring av luftmengder til Revit

Etter at luftmengdene er fordelt i *Skript* 02 – *Luftmengdeberegninger,* overføres verdiene tilbake til ventilene fra regnearket til modellen. Kolonnenummeret som luftmengden til hver ventil befinner seg i, er definert i en «String» node. Denne verdien blir sendt videre til en subtraksjonsnode, ettersom listehåndteringen i Dynamo begynner på 0.



Figur 56 – Tilbakeføring av luftmengder til Revit

«List.Deconstruct» noden tar vekk den første raden i listen, ettersom dette var beskrivelsen av parametere og ikke verdiene. Parameteren som verdiene blir skrevet til, er da «Flow» og konverteringsfaktoren blir inkludert i beregningene før tilbakeføring, slik at vi får den verdien vi ønsker i modellen.



Figur 57 – utnytte konverteringsfaktor i skriptet tilbakeføring av luftmengder til Revit

13.2 Effektbehovsberegninger.

Skriptet for å hente ut nødvendige areal og luftmengder for videre beregninger av effektbehov på rom-/bygningsnivå, blir beskrevet i detalj i dette kapitlet. Skriptene skal fungere slik at det skal samles inn nødvendige arealer fra modellen for bruk ved beregning av effektbehov, samt et skript som baserer seg på beregning av effektbehovet til oppvarming av tilluft i ventilasjonsanlegget.

Skriptene forklart i dette dokumentet vil ta for seg hvordan man effektivt kan bruke informasjonen i en BIM modell, for å hente ut de nødvendige areal for effektbehovsberegninger på rom-/bygningsnivå i et bygg, direkte overført til Excel fra modell, for raske beregninger.

Det vil være flere ulike deler i *Skript 06,* som da danner et totalt skript. I hovedsak tar vi utgangspunkt i «Spaces» og elementer i disse. Videre vil det da hentes ut dimensjoner, samt utføres summering av areal i skriptet, for like elementer i samme rom.

Data vil da sendes videre til et Excel dokument, med definert informasjon om de ulike parametere, som er hentet fra modellen. Det viser seg at å hente ut arealer fra vegger tilknyttet rom er meget avhengig av å ha modeller riktig definert, slik at uthenting av informasjonen er mulig. Flere av modellene vi har testkjørt skriptet på, viser seg å ha utilstrekkelige utplasseringer av «Spaces» og dermed fører til nullverdier i mange av tilfellene. En foreløpig løsning på dette problemet, er å programmere om skriptet slik at en kan hente ut arealer fra valgt «Space», og dermed ha en betydelig større kontroll på informasjonsmengden som blir høstet ut.

På denne måten kan ingeniøren enkelt kvalitetssikre informasjonen, og dermed foreta en avgjørelse på om dette er riktig informasjon, eller om informasjonen må forkastes og eventuelt hentes ut manuelt.

13.2.1 Skript 06 - Hente ut areal tilknyttet Spaces/Rooms.

Det som videre presenteres er en mer detaljert fremgangsmåte på hvordan skriptene er bygget opp og hvilke noder som er nødvendige for at dette skal fungere.

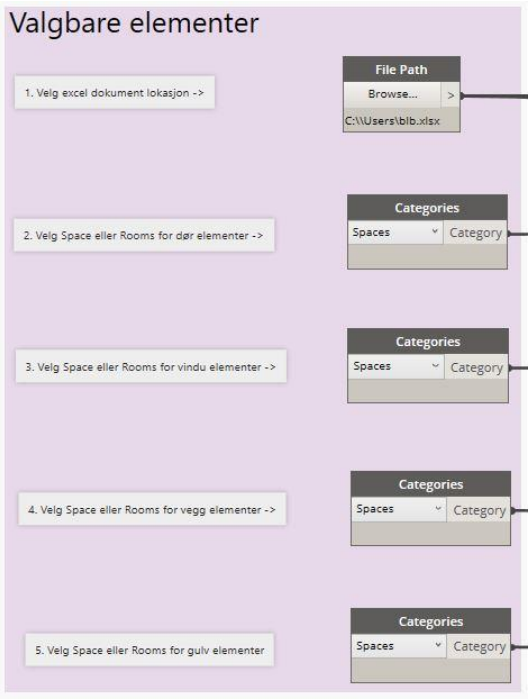
Alle grupperte noder, databehandling og essensen av hvordan skriptene fungerer, vil bli forklart nærmere i de neste deler av dette kapitlet.

13.2.1.1 Inputs/Outputs

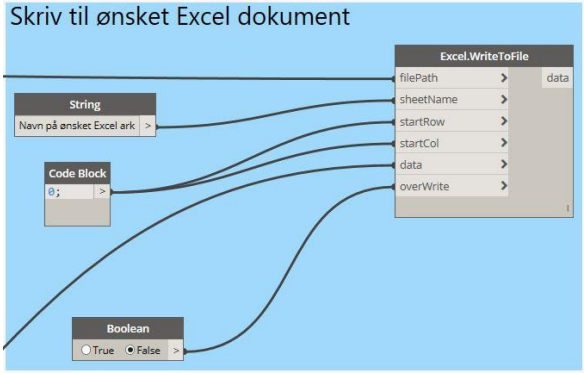
Det er nødvendig å definere noen inputs for alle deler av skriptene som blir presentert på løpende bånd nedenfor. Det er også viktig med et definert lokalt Excel dokument hvor data blir overført, samt å definere om man ønsker å hente informasjon fra ulike typer elementer tilknyttet «Spaces» eller «Rooms» i modellen.

«File Path» og «Categories» i *Figur 59* er felles for alle deler i skriptet.

«Excel.WriteToFile» i *Figur 58*, som er den siste individuelle noden i hvert skript, noe som sikrer at data blir skrevet til et spesifikt dokument og ark. Denne noden krever flere input som «filePath» hvor dokumentet befinner seg på maskinen, «sheetName» hvor man definerer ønsket ark/sheet i dokumentet man ønsker å skrive til, samt kolonne og rad man ønsker å overføre data til. Input som er definert som «data» er hentet fra de ulike listehåndteringene i skriptet. «OverWrite» er et True/False valg, hvor man kan velge å overskrive data som allerede ligger i Excel filen.



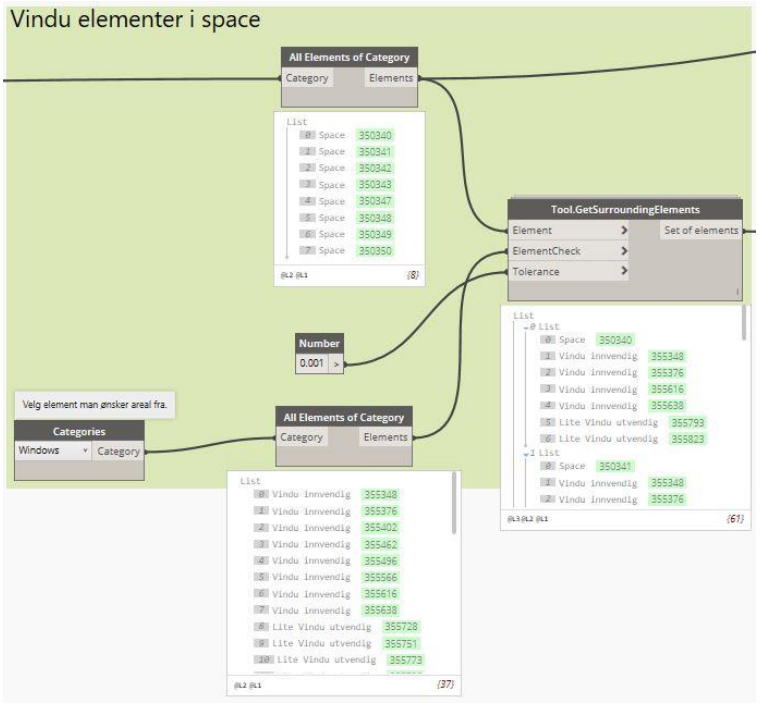
Figur 59 – Felles input til alle delskript



Figur 58 – Den avsluttende node for alle deler

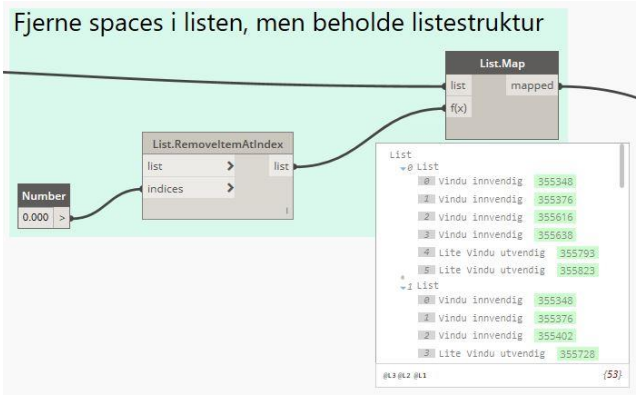
13.2.2 Skript 06 – Del 1 og 2 - Hente ut dør- og vindusareal tilknyttet Spaces/Rooms.

Den første delen av skriptet viser hvordan man kan hente ut areal til elementer av ønsket kategori, som er modellert i modellen. Det er brukt «Tool.GetSurroundingElements» som tar utgangspunkt i to ulike elementer, og sjekker om det ligger elementer av ønsket type i omgivelsene, med en satt toleransegrad for å oppdage nærliggende objekter. Dette verktøyet ble brukt for å finne elementer som dører og vinduer, som var i eller rundt «Spaces» i modellen. Dette ble videre brukt til å finne ut eksakt hvilke elementer som er tilknyttet de ulike «Spaces». Resultatet er en liste som inneholder informasjon om hvilke, og hvor mange elementer som er plassert i de definerte «Spaces».



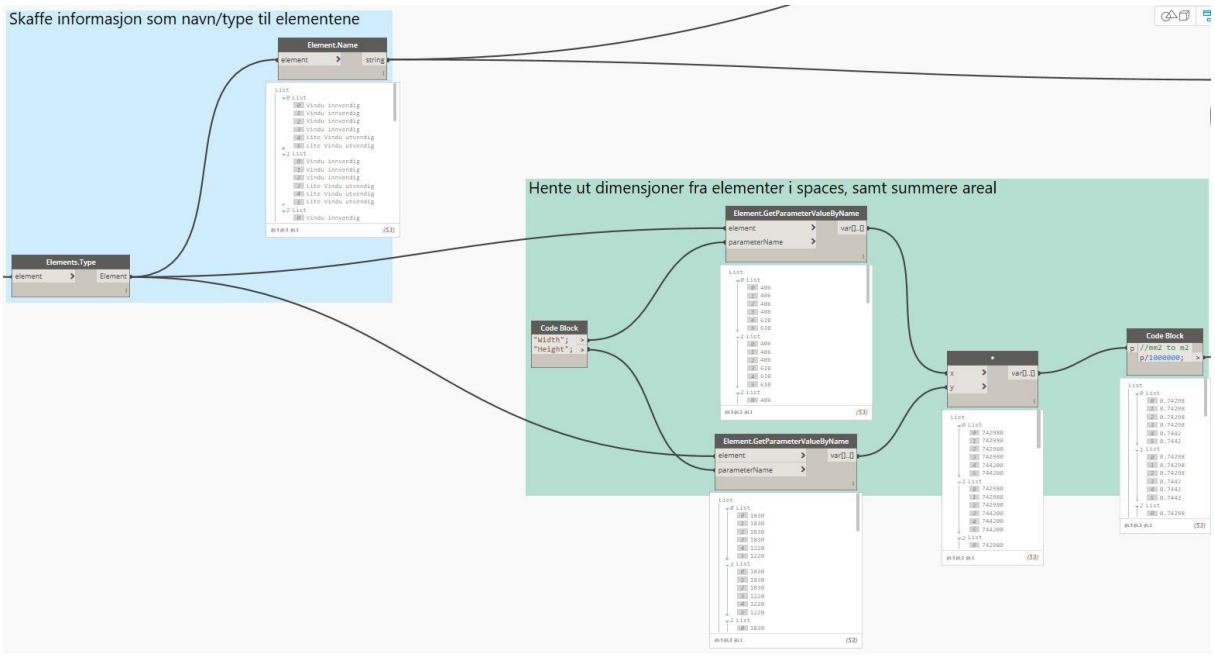
Figur 60 – Definerte rom og omliggende elementer

Det er ønskelig å beholde listestrukturen for elementer tilknyttet de ulike rommene videre i skriptet, for å kunne knytte de ulike elementene til riktige *rom*. Elementenes parameter, som eksempelvis dimensjoner og type, blir hentet ut for videre bearbeidelse.



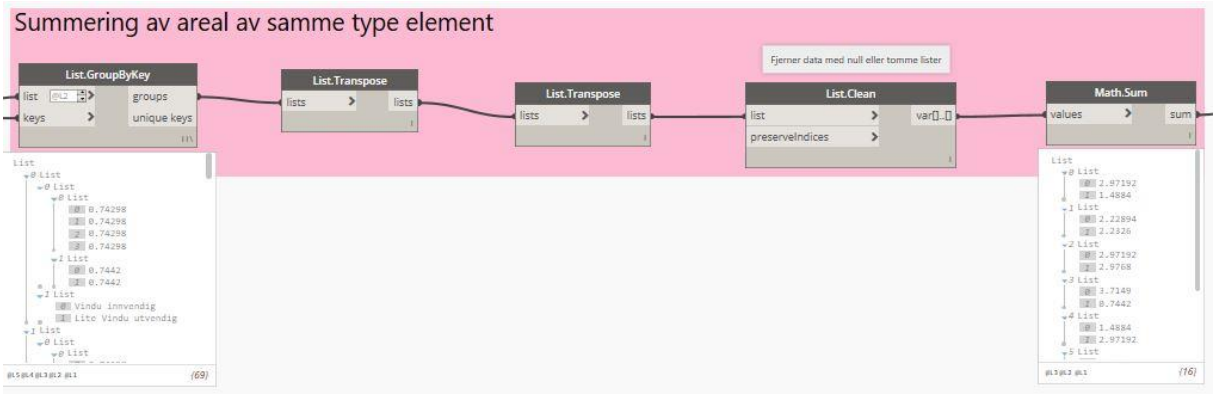
Figur 61 – Listehåndtering for videre arbeid med elementene

Videre blir listen med elementer brukt som utgangspunkt for å hente ut informasjon som type, navn og parametere som høyde/bredde til de ulike elementene. «Element.GetParameterValueByName» brukes for å hente ut definerte parameter fra et element, som da videre brukes til å beregne ut areal for hvert element, ved å addere x med y i en matematisk node (kalt *). Den ene parameteren er da høyden til elementet, mens det andre er bredden. Her er det da *meget* viktig at elementer har definerte dimensjoner, for at dette skal fungere. En kan altså ikke hente ut informasjon fra generiske objekter, uten definerte parametere som nevnt. Det er også viktig å påpeke at Revit bruker millimeter som default, så det er lagt inn en *«Code Block»* som overfører arealene fra mm^2 til m^2.



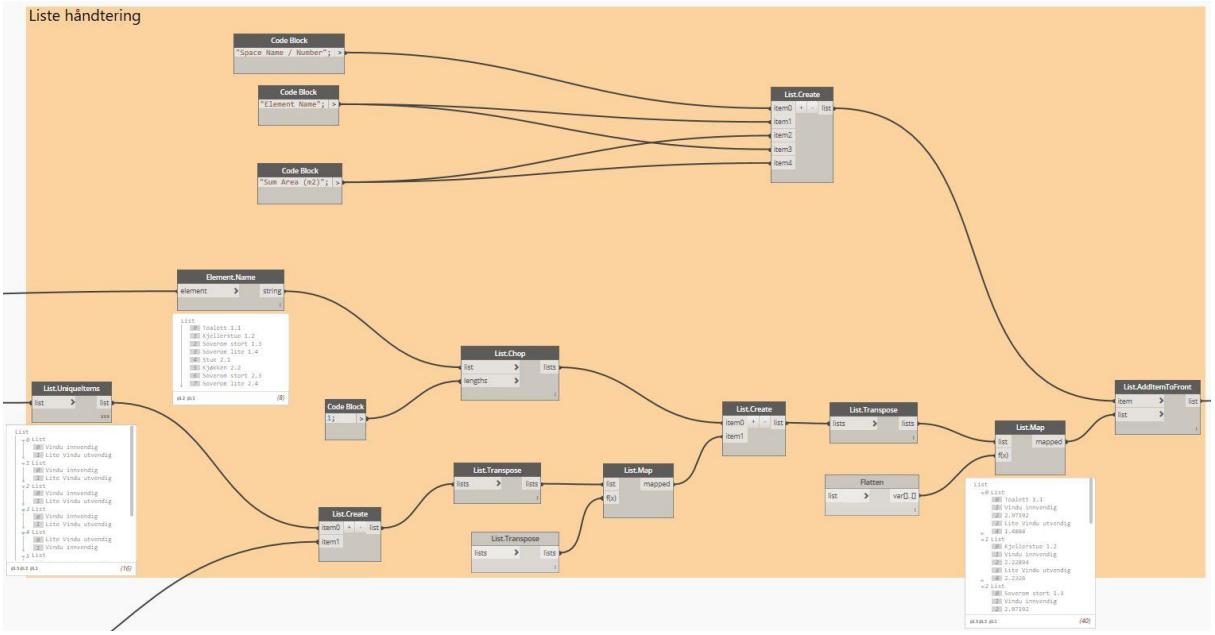
Figur 62 – Informasjonsinnhenting fra elementer som navn, type og dimensjoner

Ved summering av areal av samme type element i samme rom, er det brukt «List.GroupByKey» som da overfører listestrukturen fra tidligere arealberegninger, og samler areal som har like navn på elementene. Videre er det brukt ulike noder for listestrukturering som «List.Transpose» og «List.Clean», for å renske listene for nullverdier. «Math.Sum» summerer verdier som ligger i listene, og skal videre i scriptet kobles mot de riktige elementer.



Figur 63 – Summering av arealer med samme type element i hvert «Space»

Denne delen av skriptet er brukt til listehåndtering av data for å kunne skrive data videre til ønsket definert Excel dokument. Det er viktig at riktig data blir overført, slik at definerte *rom* med tilhørende elementer og arealer blir koblet sammen, og videre at dette blir skrevet til Excel på en respektiv og oversiktlig måte. Det er lagt til overskrifter som da legges til i listen på de allerede forklarte parametere, som henholdsvis er *Space Name / Number, Element Name* og *Sum Area (m2).*



Figur 64 – Listehåndtering for å samle alle lister med riktig struktur og korrekt tilkobling

Resultatet fra skriptet som er kjørt på en testmodell, vises i en utskrift fra Excel i *Figur 65* og *Figur 66*.

	A	B	C	D	E
1	Space Name / Number	Element Name	Sum Area (m2)	Element Name	Sum Area (m2)
2	Toalett 1.1	Vindu innvendig	2,97192	Lite Vindu utvendig	1,4884
3	Kjellerstue 1.2	Vindu innvendig	2,22894	Lite Vindu utvendig	2,2326
4	Soverom stort 1.3	Vindu innvendig	2,97192	Lite Vindu utvendig	2,9768
5	Soverom lite 1.4	Vindu innvendig	3,7149	Lite Vindu utvendig	0,7442
6	Stue 2.1	Lite Vindu utvendig	1,4884	Vindu innvendig	2,97192
7	Kjøkken 2.2	Lite Vindu utvendig	2,9768	Vindu innvendig	3,7149
8	Soverom stort 2.3	Lite Vindu utvendig	2,9768	Vindu innvendig	2,97192
9	Soverom lite 2.4	Lite Vindu utvendig	0,7442	Vindu innvendig	2,22894

Dør areal | **Vindu areal** | Vegg areal | Grunnflate areal | Sheet1

Figur 65 – Resultat av kjørt skript for å anskaffe vindusarealer – Del 1

	A	B	C	D	E	F	G
1	Space Name / Number	Element Name	Sum Area (m2)	Element Name	Sum Area (m2)	Element Name	Sum Area (m2)
2	Toalett 1.1	Ytterdør	4,645152	Innerdør	3,252216		
3	Kjellerstue 1.2	Ytterdør	4,645152	Innerdør	3,252216		
4	Soverom stort 1.3	Ytterdør	1,548384	Innerdør	3,252216		
5	Soverom lite 1.4	Ytterdør	1,548384	Innerdør	3,252216		
6	Stue 2.1	Innerdør	3,252216	Skyvedør innvendig	3,71856	Veranda dør utvendig	1,626108
7	Kjøkken 2.2	Innerdør	3,252216	Skyvedør innvendig	3,71856	Veranda dør utvendig	3,252216
8	Soverom stort 2.3	Innerdør	3,252216	Skyvedør innvendig	3,71856	Veranda dør utvendig	3,252216
9	Soverom lite 2.4	Innerdør	3,252216	Skyvedør innvendig	3,71856	Veranda dør utvendig	1,626108

Dør areal | Vindu areal | Vegg areal | Grunnflate areal | Sheet1

Figur 66 – Resultat av kjørt skript for å anskaffe dørarealer – Del 2

Skriptet fungerer både for elementer som vinduer og dører **hvis** de er definert i modellen som denne typen, med tilhørende definerte dimensjoner som høyde og bredde. Dette skriptet kan kjøres på både elementer tilknyttet «Rooms» eller «Spaces», avhengig av ønsket kategori.

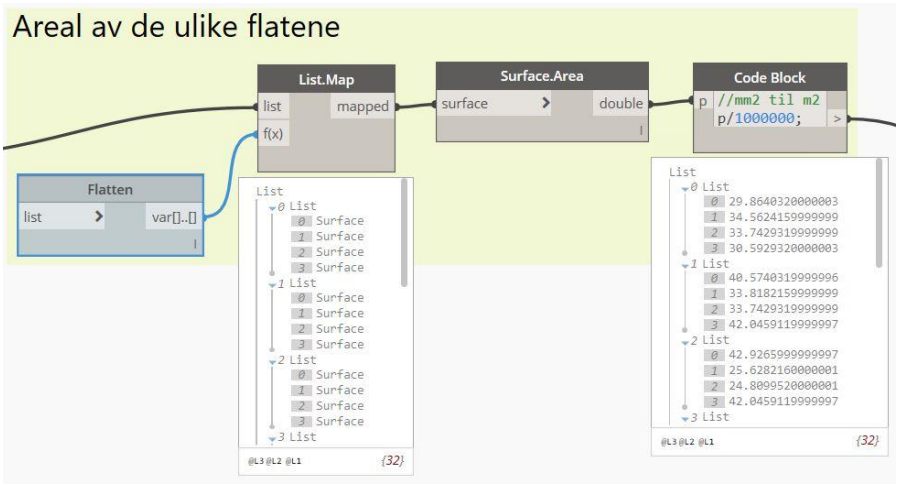
13.2.3 Skript 06 – Del 3 - Hente ut veggareal tilknyttet Spaces/Rooms

I dette skriptet er det ønskelig å anskaffe areal til vegger tilknyttet de ulike rom i modellen. Det er formet til å fungere til en spesifikk modell, og vil således ikke være fungerende for alle arkitektmodeller. Men hvis man tar utgangspunkt i logikken og oppbygningen, vil man kunne bruke dette til å raskt hente ut areal til ulike type vegger, med tilhørighet til ulike rom i modellen.

Det er tatt utgangspunkt i et skript lagt ut på dynamobim.org sine sider, hvor det ble lagt ut en fil av Vikram Subbaiah som heter «RoomWallArea.dyn». [22] Skriptet ble videreutviklet til ønsket funksjon.

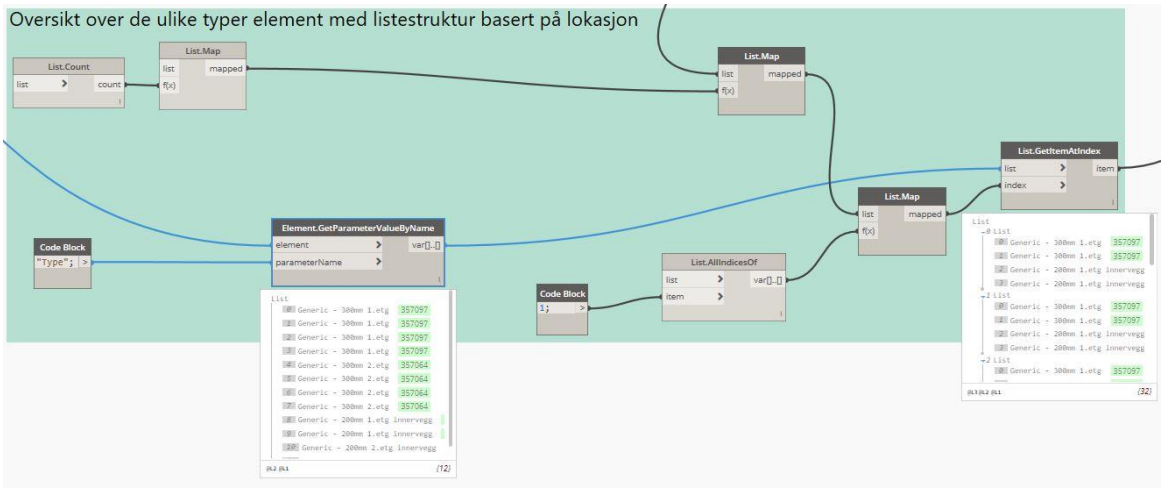
Den første del av skriptet tar utgangspunkt i «Spaces» i modellen, og oppretter en kobling mellom geometrien til alle rommene i modellen, og geometrien til alle veggene i modellen. Deretter brukes dette som grunnlag til å opprette «flater» i Dynamo, som baseres på lokasjon, og danner grunnlag for beregning av arealer. Oversiktsbilde over dette finnes i vedlagt PDF dokument over *Skript 06*.

Den neste delen, eller gruppen i skriptet, henter ut overflateareal til de ulike flatene ved hjelp av noden «Surface.Area». Videre summeres og konverteres disse arealene fra millimeter til meter, ved hjelp av «Code Block» som tidligere definert. Det er også utført listehåndtering i forkant, slik at areal til de ulike flatene er basert på lokasjon til de ulike rom.



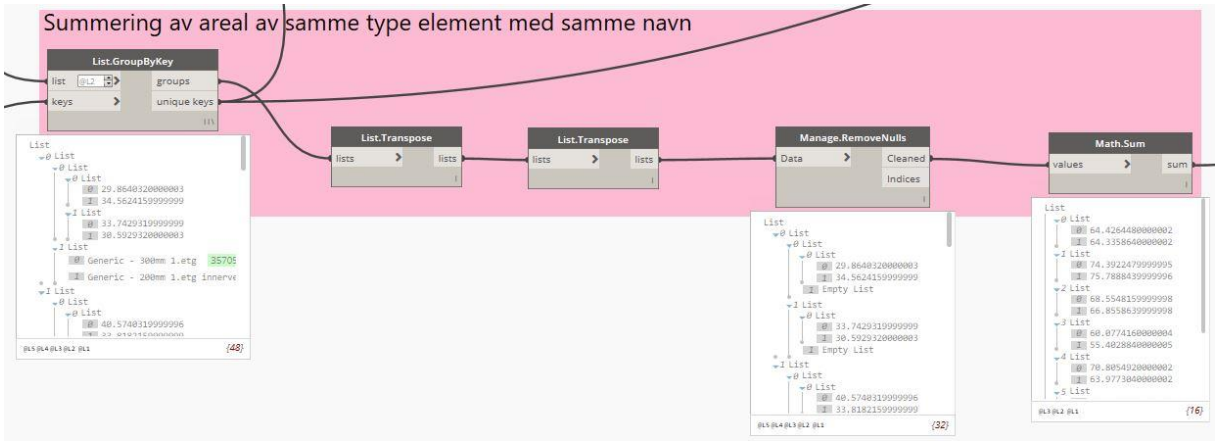
Figur 67 – Areal av de ulike flatene

Videre i skriptet, er det nødvendig å koble flatene til de ulike vegger i modellen. Dette vil si at man kobler flatene som er hentet ut og kalkulert, videre til de tilhørende veggtyper som befinner seg i modellen. Det er hentet ut parameter som «Type», som viser til typenavnet til objektet, og videre «List.GetItemAtIndex» for å gruppere disse på romnivå.



Figur 68 – Oversikt over de ulike typer element med listestruktur basert på lokasjon

Etter å ha koblet elementtypen til de ulike flatene, vil det være ønskelig å summere areal av samme type veggelement som befinner seg i samme rom. *Figur 69* viser grupperinger ved hjelp av noden «List.GroupByKey», og tar utgangspunkt i summerte areal med «keys» som er objektenes typenavn. Da får vi koblet summerte areal, flater og objekter til riktig rom i modellen.



Figur 69 – Summering av areal av samme type element med samme navn

Den siste gruppen av skriptet, brukes for å kombinere de ulike listene som er hentet ut i de ulike delene. Her vil vi i tillegg anskaffe informasjon som veggobjektenes sin funksjon, ved hjelp av «WallType.Function», som vil beskrive om veggen er definert som innvendig eller utvendig i modellen. Videre vil vi lage en liste med alle de ulike parametere som navn på rom/space, veggtype, funksjon og areal til en samlet liste ved hjelp av «List.Create», som også brukes til å definere noen overskrifter til Excel dokumentet, slik at resultatet blir oversiktelige. Det henvises igjen til leveransen hvor oversiktsbilde over skriptet, ligger vedlagt som PDF.

Resultatet fra skriptet som er kjørt på en testmodell vises i *Figur 70*.

	A	B	C	D	E	F	G
1	Space Name	Wall Type	Element Func	Element Area	Wall Type	Element Func	Element Area
2	Toalett 1.1	Generic - 300	Exterior	64,426448	Generic - 200	Interior	64,335864
3	Kjellerstue 1.2	Generic - 300	Exterior	74,392248	Generic - 200	Interior	75,788844
4	Soverom stor	Generic - 300	Exterior	68,554816	Generic - 200	Interior	66,855864
5	Soverom lite	Generic - 300	Exterior	60,077416	Generic - 200	Interior	55,402884
6	Stue 2.1	Generic - 300	Exterior	72,7145324	Generic - 200	Interior	66,2152576
7	Kjøkken 2.2	Generic - 300	Exterior	81,05551	Generic - 200	Interior	77,0322776
8	Soverom stor	Generic - 300	Exterior	72,21551	Generic - 200	Interior	68,9352576
9	Soverom lite	Generic - 300	Exterior	64,6187324	Generic - 200	Interior	58,1182376
10							

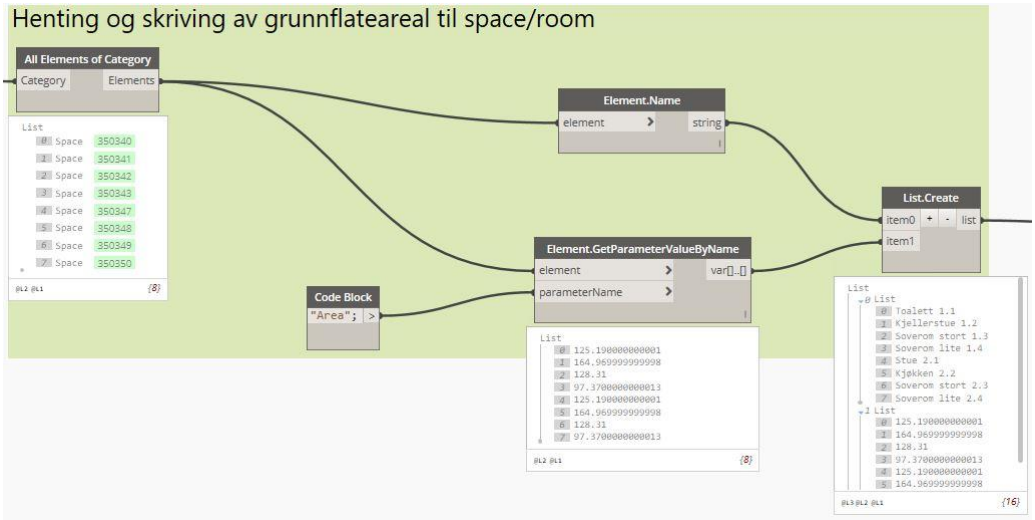
Dør areal | Vindu areal | **Vegg areal** | ...

Figur 70 – Resultat av kjørt skript på en testmodell

Parameterne presentert i Excel, som overskrifter og tilhørende verdier til de ulike parametere som space/rom, veggtype, funksjon og areal, er hentet rett fra modellen. Det er meget viktig med riktig listehåndtering i skriptet, for de ulike objekter, areal, rom og funksjon, for at dette skal presenteres som ovenfor. Ved udefinerte objekter kan det oppstå nullverdier eller tomrom i Excel dokumentet.

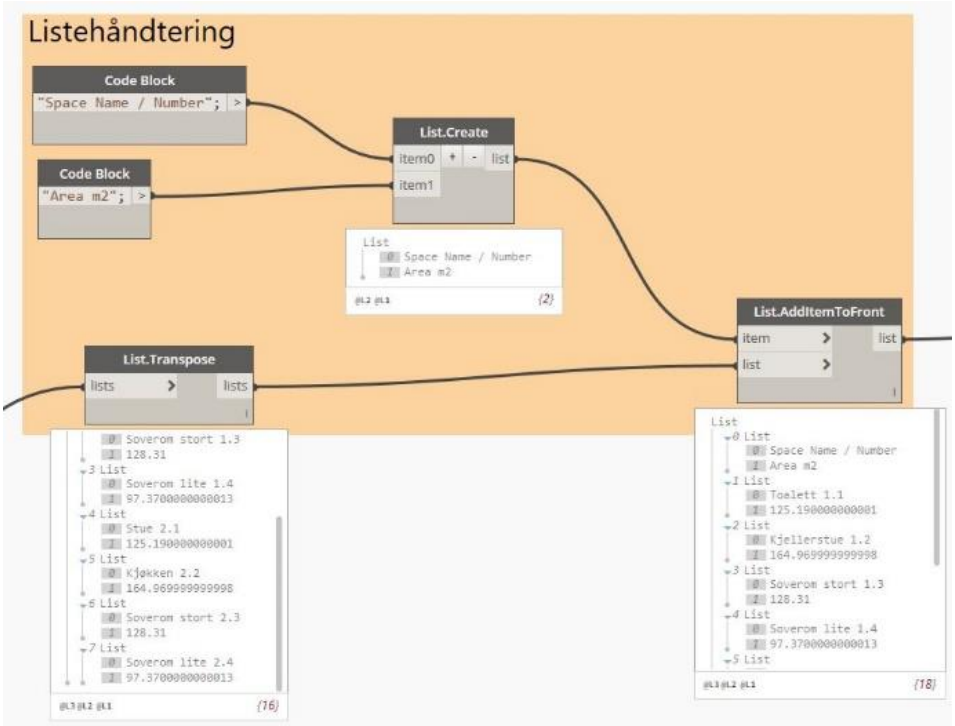
13.2.4 Skript 06 – Del 4 – Hente nødvendig gulv- og takareal tilknyttet Spaces/Rooms

Dette er et relativt kort og enkelt skript sammenlignet de andre skriptene, hvor det hentes ut areal til de ulike rommene som er modellert. Dette er naturlig å legge ved som ekstra informasjon ved videre beregninger, og hører med som en nødvendig parameter ved fullstendig effektbehovsberegninger. Dette kan også gjøres ved mengdeuttak i Revit, men det er hensiktsmessig å gjøre alt i en operasjon.



Figur 71 – Henter ut areal fra alle rommene i modellen

Til slutt er det nødvendig med listehåndtering for å koble areal til rom, sammen med et par overskrifter, for å sørge for at formateringen i Excel dokumentet blir som ønsket.



Figur 72 – Listehåndtering

Resultatet fra skriptet som er kjørt på en testmodell vises i *Figur 73*.

	A	B
1	Space Name / Number	Area m2
2	Toalett 1.1	125,19
3	Kjellerstue 1.2	164,97
4	Soverom stort 1.3	128,31
5	Soverom lite 1.4	97,37
6	Stue 2.1	125,19
7	Kjøkken 2.2	164,97
8	Soverom stort 2.3	128,31
9	Soverom lite 2.4	97,37
10		

Grunnflate areal | Dør areal | Vindu areal | Vegg areal | Ark1

Figur 73 – Resultat av kjørt skrift for å anskaffe gulvareal – Del 4

Skriptet opererer ved å hente areal direkte fra definerte «Spaces» i modellen, så det er viktig at disse er definert riktig i modellen. Hvis det er feil på avgrensninger eller at grensene går inn i vegger eller andre forstyrrende elementer, vil disse arealene ikke være gyldige.

13.2.5 Skript 07 - Ventilasjonsvarmetap på romnivå

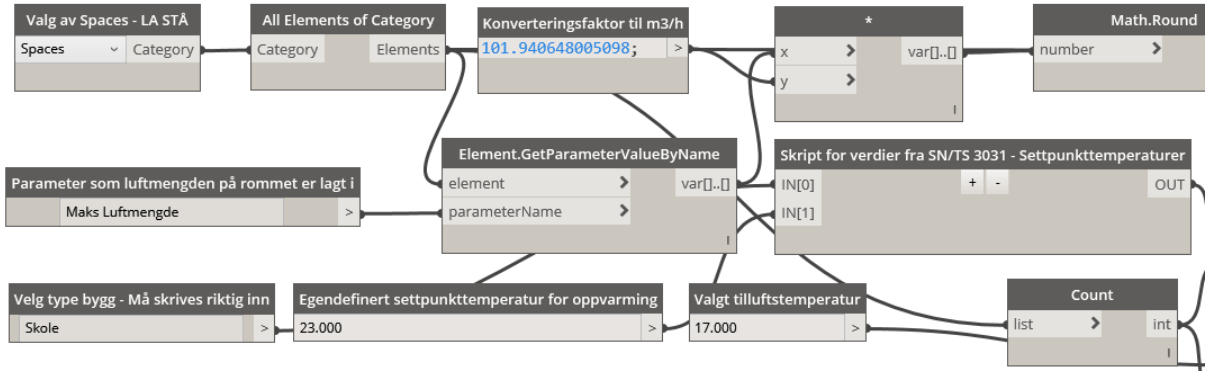
Følgende skript beregner effektbehovet til oppvarming grunnet undertemperert tilluft fra ventilene.

Tabell 11 – Parametere som blir lagt til i Spaces er definert på følgende måte

Navn	Discipline	Type of parameter	Input
Varmetap ventilasjon	HVAC	Power	[W]
Maks Luftmengde	HVAC	Air Flow	[m³/h]

Del 01 – Valg av inputverdier og innhenting av luftmengder

I denne delen hentes alle «Spaces» fra modellen, med tilhørende luftmengder som ble beregnet tidligere, og konverteringsfaktor som ble diskutert i tidligere skript. Videre er det input basert på bygningskategorier og muligheter for egendefinert settpunkttemperatur for oppvarming med samme fremgangsmåte som *Skript 02*. Brukeren har også mulighet for å sette inn ønsket verdi for tilluftstemperatur.



Figur 74 – Valg av inputverdier og innhenting av luftmengder

Noden med IN[0] og IN[1] er en Python node med samme logikk som i *Skript 02*. For oversikt over oppbygningen av koden henvises det til *Vedlegg VI.*

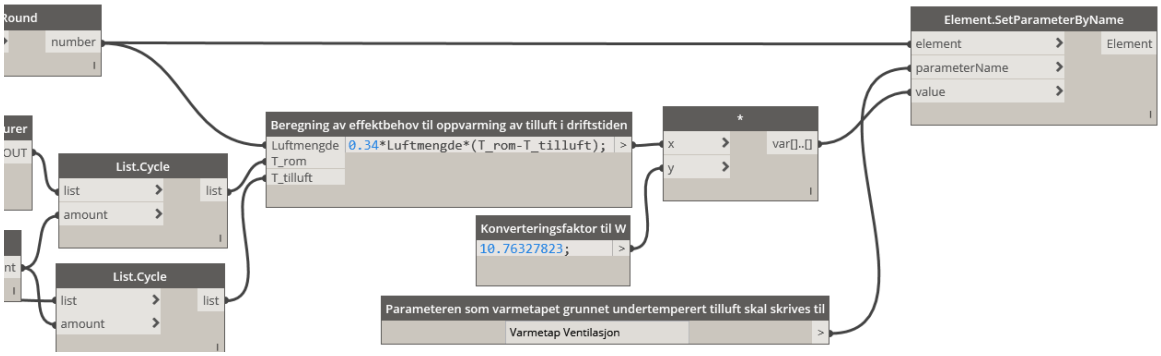
Del 2 – Beregninger av effektbehovet til oppvarming

I denne delen blir effektbehovet beregnet. Node «Beregning av effektbehov til oppvarming av tilluft i driftstiden», se *Figur 75*, tar stilling til følgende formel:

$$\varphi_{ventilasjon} = \dot{m} \cdot c_p \cdot \Delta T \ \rightarrow 0.34 \cdot \dot{V} \cdot \left(T_{rom} - T_{tilluft}\right)$$

$$\dot{V} = luftmengde\ i\ m^3/h$$

Formel 2 – Beregning av effektbehov



Figur 75 – Beregning av effektbehov til oppvarming av tilluft i driftstiden

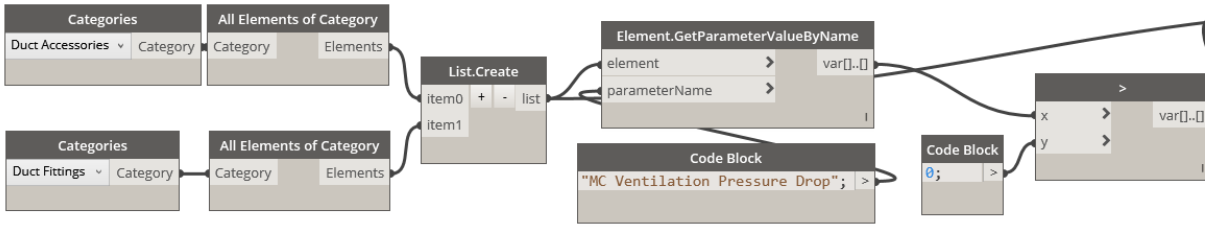
Her må vi også ta stilling til en konverteringsfaktor for å få en riktig verdi for effektbehovet ettersom Dynamo tar kun stilling til enheter benyttet til geometrisk vurdering, og ikke strømnings og varmetekniske enheter. Beregningene sendes videre til «Element.SetParameterByName» noden som er koblet til «Spaces» hvor beregningene blir overført og tildelt til parameteren «Varmetap Ventilasjon».

13.3 Spesifikke

13.3.1 Skript 08 – Kvalitetskontroll av trykkfallsberegninger i MagiCAD

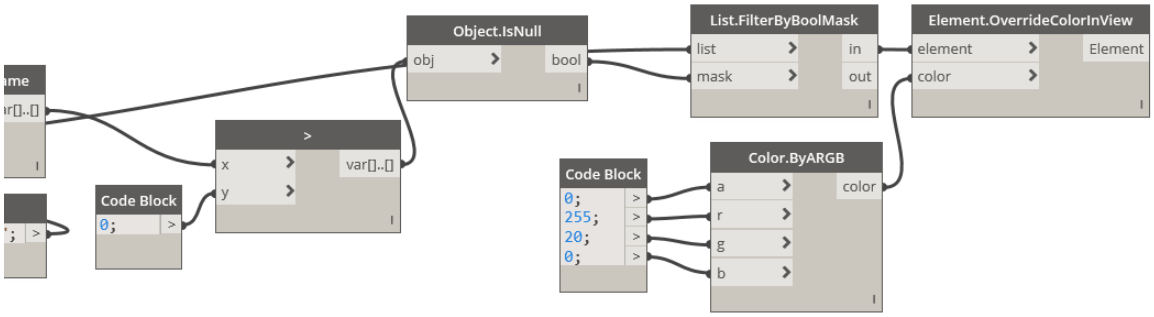
Følgende skript undersøker elementer i ventilasjonsanlegget for trykkfallsverdier. Skriptet fargelegger alle elementer som har verdien 0 Pa i parameteren «MC Ventilation Pressure Drop». En må sørge for å kjøre trykkfallsberegning i forkant av skriptet, ellers blir alle elementer (unntak av kanaler som ikke er med) fargelagt til rødt.

Skriptet henter først ut to kategorier som er «Duct Fittings» og «Duct Accessories». Videre hentes verdier lagret i parameter ut og det undersøkes om den har en verdi større enn 0 Pa. Forløpet er illustrert i *Figur 76*.



Figur 76 – Resultat for Skript 08 Kvalitetskontroll av trykkfallsberegninger i MagiCad

Deretter blir alle elementer som ikke har beregnet trykkfallsverdi, sendt til et filter og videre fargelagt ved bruk av «Element.OverrideColorinView» noden. Forløpet er illustrert i *Figur 77*.

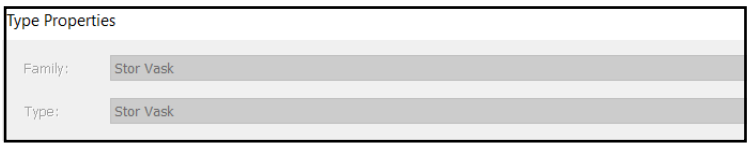


Figur 77 – Farging av elementer til rødt

13.3.2 Skript 09 - Utstyrstelling og beregning av sannsynlige vannmengder

Skriptet filtrerer utstyr basert på ulike rom og gir et mengdeuttak i Excel og en beregning av største sannsynlige vannmengder for summen av alle normalvannmengdene i bygget.

For at skriptet skal kunne filtrere utstyr riktig er det essensielt at BIM objektene som definerer utstyret er logisk navngitt. Dette gjøres i ARK sin modell hvor «Type» i *Figur 78* må inneholde den teksten som benyttes i skriptet for filtrering.



Figur 78 – Type properties som må være tilstede

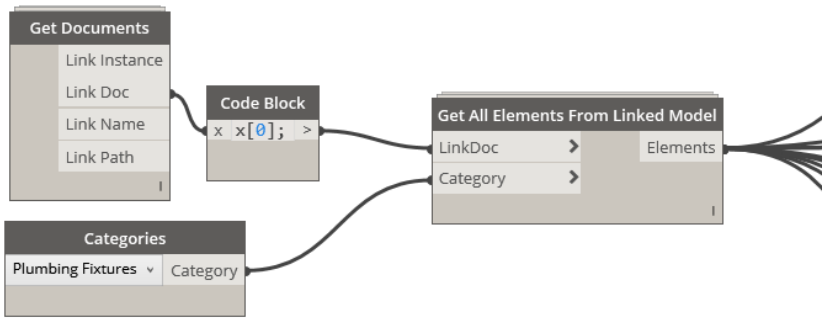
I dette skriptet benyttes følgende tekster for å filtrere utstyr i modellen:

- Vask – Typenavn på alle vasker i ARK modell
- Vaskemaskin – Typenavn på alle vaskemaskiner i ARK modell
- Toalett – Typenavn på alle toaletter i ARK modell
- Dusj – Typenavn på alle dusjer i ARK modell
- Urinal – Typenavn på alle urinaler i ARK modell

Forklaring av skriptets oppbygging

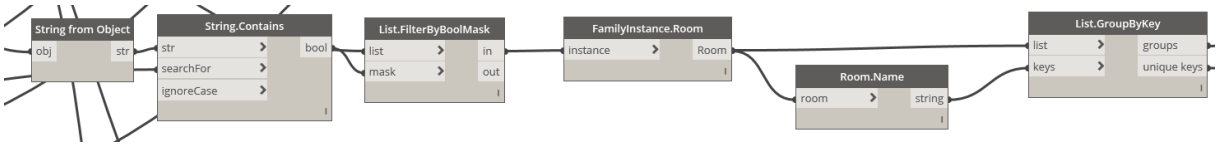
Følgende er en oversikt over hvordan skriptet er bygd opp. Her blir logikken i skriptet gjennomgått og oversikt over hele skriptet supplert i leveransen.

«Get Documents» og «Get All Elements from Linked Model» nodene som blir brukt i skriptet kommer fra Archi-lab_Grimshaw Dynamo pakke. Denne pakken henter alle sanitære utstyr som er plassert i ARK modellen som er linket til VVS modellen.



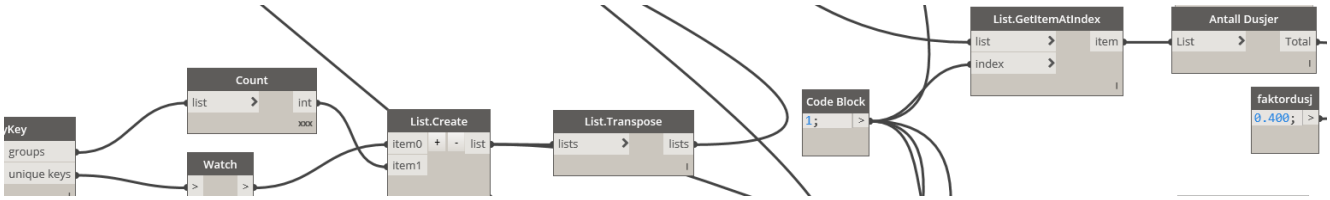
Figur 79 – Hvordan få sanitære utstyr fra linket VVS modell

Kategorien som er valgt er da «Plumbing Fixtures» og det er viktig at den som definerer objektene sørger for at alt sanitært utstyr tilhører kategorien «Plumbing Fixtures». Elementene blir koblet sammen til «List.FilterByBoolMask» og «String from Object» nodene for hvert utstyr type som skal filtreres. «String Contains» noden har en input «SearchFor». Denne delen blir koblet med en «String» node som inneholder den teksten RIV har definert utstyret som. Er det eksempelvis snakk om et toalett så skal BIM objektet som inneholder toalett ha teksten «Toalett» inkludert i «Family» og «Type» egenskapene i Revit for å kunne benytte skriptet.



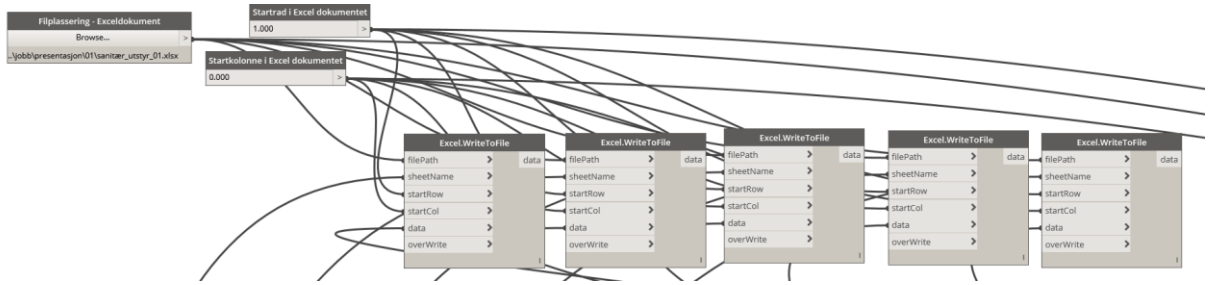
Figur 80 – Filtrering og gruppering av utstyr i forhold til romtilhørighet

«Count» noden er satt som kryssprodukt i forhold til hvor mange instanser det er, av hver utstyrstype i hvert rom. «List Create» noden sammenstiller da navn og nummer på rom og hvor mange enheter av utstyret det rommet inneholder.



Figur 81 – Liste håndtering

Videre blir informasjonen sendt i to retninger. Den ene retningen er mot utskrift i Excel fil hvor utstyr og romnavn samt nummer skrives ut i en Excel mal. Den andre retningen er mot en beregning av størst samtidig vannmengde for hele bygningen. *Figur 82* er oppsettet for Excel uttaket av utstyrene og rommene.

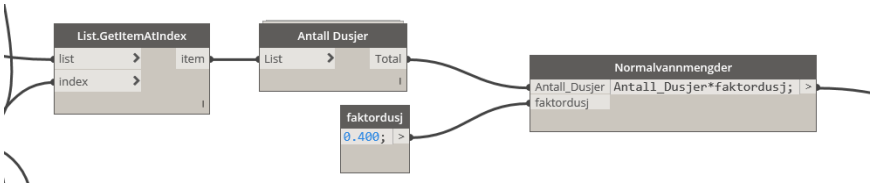


Figur 82 – Oppsettet for Excel uttaket av utstyrene og rommene

«SheetName» inputen er til fanen i Excel arket som inneholder utstyret hvor vi da har fem faner i vår mal. Fanenavnene får de samme navn som utstyret, eksempelvis «Toalett». Videre er det en input for start rad og kolonne hvor startraden er satt til 1, for å få et område hvor vi kan filtrere i regnearket.

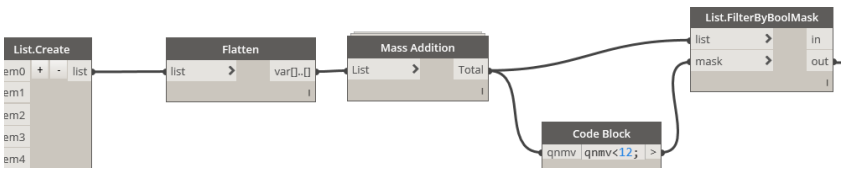
Beregning av normalvannmengder

Figur 83 viser forløpet for beregningene av største samtidige vannmengde for anlegget. Alle typer utstyr blir summert og multiplisert med faktor for normalvannmengde for hver utstyrstype i forhold til verdier gitt i NS 3055.

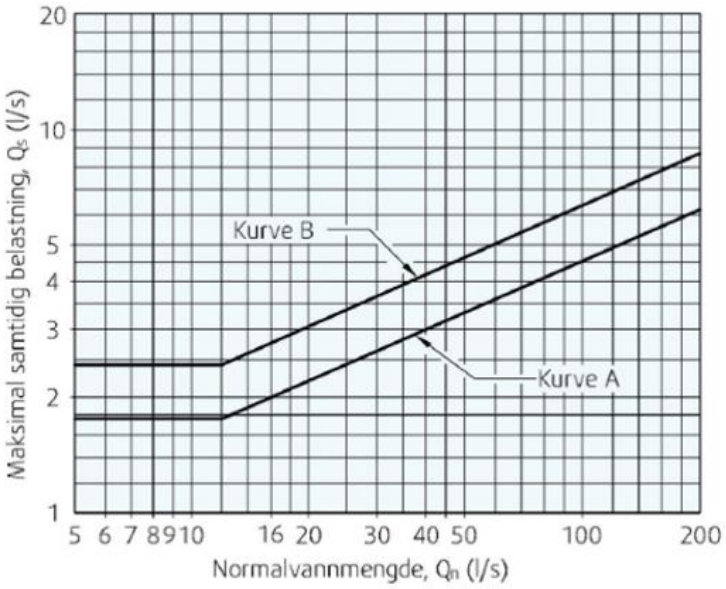


Figur 83 – Forløpet for beregningene av størst samtidig vannmengder for anlegget

Denne delen av skriptet tar for seg beregningen av maksimal samtidighet for bunnledningen, og tar stilling til summen av alle normalvannmengdene. «List.Create» noden lengst til venstre i *Figur 84,* er en liste av normalvannmengder til alt utstyr fra arkitekt modellen. Videre blir normalvannmengdene summert med «Mass Addition» noden og et filter som sørger for å sende verdien videre, dersom summen av normalvannmengdene er mindre enn 12 l/s, basert på kurven angitt i *Figur 85*.

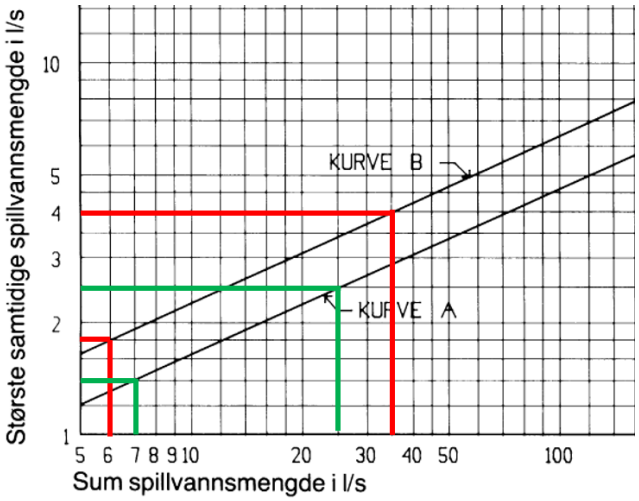


Figur 84 – Summering av normalvannmengde



Figur 85 – Fig. 42 i Byggdetaljer 553.004 [23]

Funksjonsuttrykkene som blir benyttet i skriptet til å beregne største sannsynlige vannmengde, er et resultat av interpolasjon av bestemte punkter på kurvene i *Figur 86*, ettersom funksjonsuttrykk ikke er definert i NS 3055. Funksjonsutrykket er vist i *Tabell 12.*



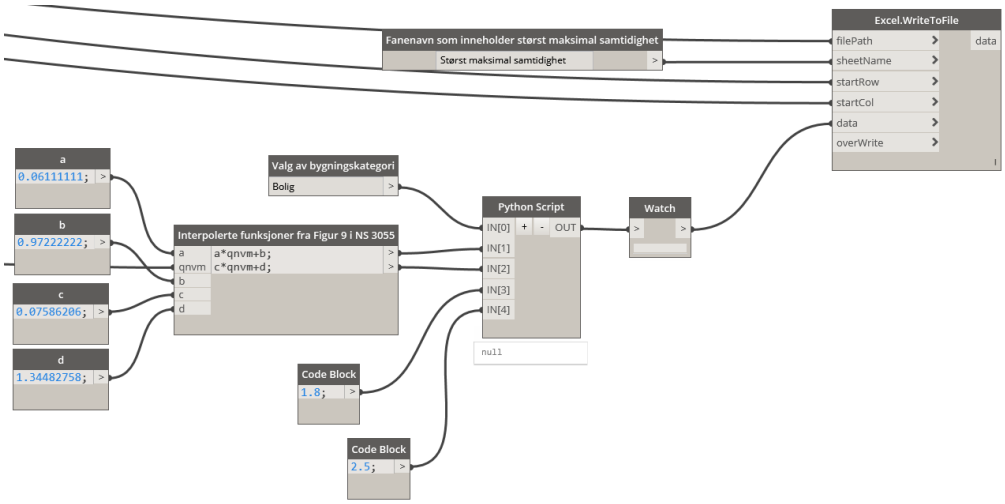
Figur 86 – Bestemte punkter på kurvene i NS 3055 [8] før interpolering

Tabell 12 – Funksjonsutrykket for Kurve A og B

Kurve A	**Kurve B**
$q_{sA} = a \sum q_{nvm} + b$ $a = 0.06111111$ $b = 0.97222222$	$q_{sB} = c \sum q_{nvm} + d$ $c = 0.07586206$ $d = 1.34482758$

«Python Script» noden bestemmer hvilken verdi som blir sendt videre basert på bygningskategori og sum av normalvannmengder. Dersom summen er 12 l/s eller større blir de interpolerte kurvene

sendt videre for beregning av største samtidige vannmengde. For nærmere beskrivelse av oppbyggingen av Python koden, henvises det til *Vedlegg VII*.

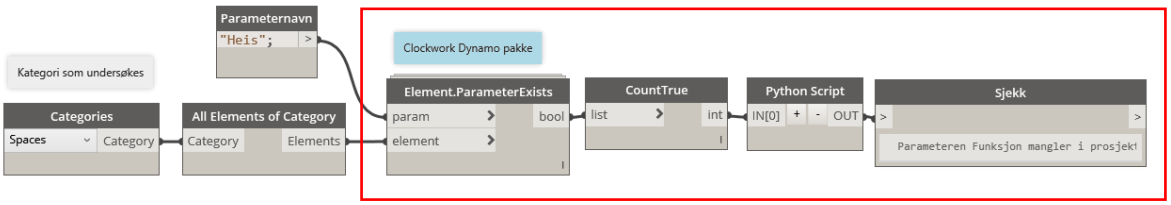


Figur 87 – Beregning av største sannsynlige vannmengder

Beregningen av størst maksimal samtidighet blir sendt videre til det siste arket i Excel dokumentet.

13.3.3 Skript 10 – Kontroll av parametere i «Spaces»

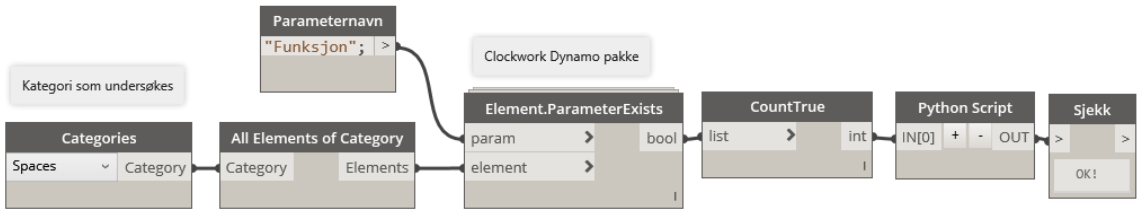
Følgende skript undersøker om en parameter er tilstede i BIM modellen. I dette tilfellet undersøkes en parameter i kategori «Spaces», hvor samme fremgangsmåte kan benyttes for alle typer kategorier i prosjektet. Skriptet kan utvides til å dekke et mye større antall parametere ved å kopiere noder markert i rødt. I *Figur 88* hvor «Element.ParameterExists» nodens «param» input kobles til en ønskelig parameter.



Figur 88 – Skript undersøker om en parameter er tilstede i BIM

Vi ser at parameter «Heis» ikke er tilstede i kategorien «Spaces» i «Sjekk» resultatet. Skriptet gir «False» verdier dersom en parameter ikke er tilstede i kategorien. Videre telles antall «True» fra resultatet og Python skriptet gir ut en beskjed basert på om det er null antall «True» verdier eller ikke, fra resultatet av «CountTrue» noden. For tekstkoden i Python noden henvises det til *Vedlegg VIII.*

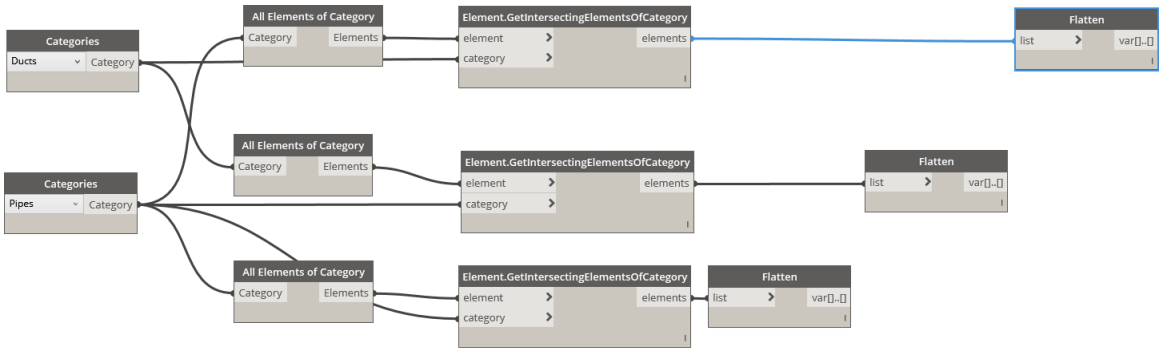
Skriptet undersøker om parameteren «Funksjon» er tilstede i modellen. Resultatet fra sjekken viser at parametere er satt til en kategori, noe som gir klarsignal om at skript som er avhengig av denne parameteren, kan brukes.



Figur 89 – Klarsignal for å kjøre skriptet

13.3.4 Skript 11 – Kollisjonskontroll

Skriptet tar stilling til elementer som kolliderer med hverandre. Man velger kategorier som man ønsker å undersøke mot andre elementer. Resultatet kommer frem i «Flatten» noden, hvor bruker har mulighet til å trykke på elementer som resulterer i at man blir tatt til den planvisningen og lokasjonen hvor kollisjonen fremkommer. Forløpet er illustrert i *Figur 90*.



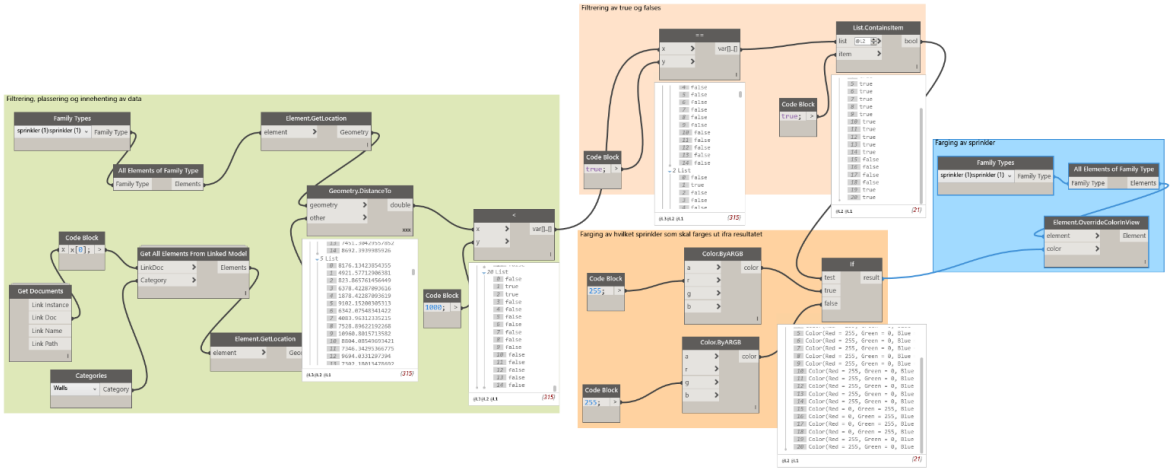
Figur 90 – Resultat for Skript 11 Kollisjonskontroll

Skriptet er en forenklet versjon av Revit sin kollisjonskontroll, hvor bruker har muligheten til å definere de kategorier som ønskes å undersøkes.

13.3.5 Skript 12 – Farging av sprinkler i forhold til avstand fra vegg

Beskrivelse

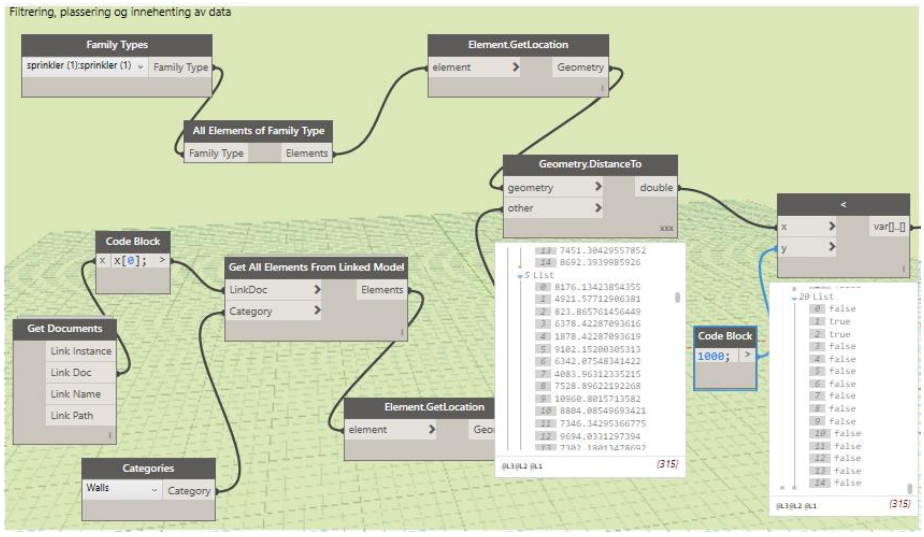
Skriptet i *Figur 91* vil automatisk farge sprinklere med en avstand fra veggen som er definert i skriptet. Sprinklerens avstand på utspredning, er definert til 1000 millimeter. Dersom sprinkelen er for nær en vegg, vil det bli fargelagt rødt, og dersom de er i god avstand fra veggen vil det bli farget til grønt. Skriptets nåværende tilstand tar kun stilling til avstand til vegger, men kan eventuelt videreutvikles til å inkludere flere elementer.



Figur 91 – Skript 12 Farging av sprinklere

Del 1

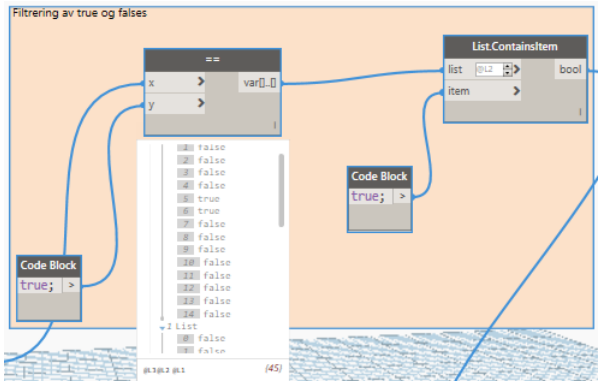
Den første delen av skriptet, *Figur 92,* er å få lokasjonene til sprinkler samt veggene med noden «*Element.Getlocation»*. Siden ARK-modellen er brukt som en link, kunne vi ikke få tak i veggene. For å løse dette ble det nodene «*Get Documents», «Code Block», «Categories», «Get All Elements From Linked Model»* brukt, for å få lokasjonene til veggene. Noden «*Geometry.DistanceTo»* benyttes for å finne avstanden mellom sprinkler og veggene, her er det viktig å sette noden til kryssprodukt på *Lacing*.



Figur 92 – Innhenting, filtrering og plassering av data

Del 2

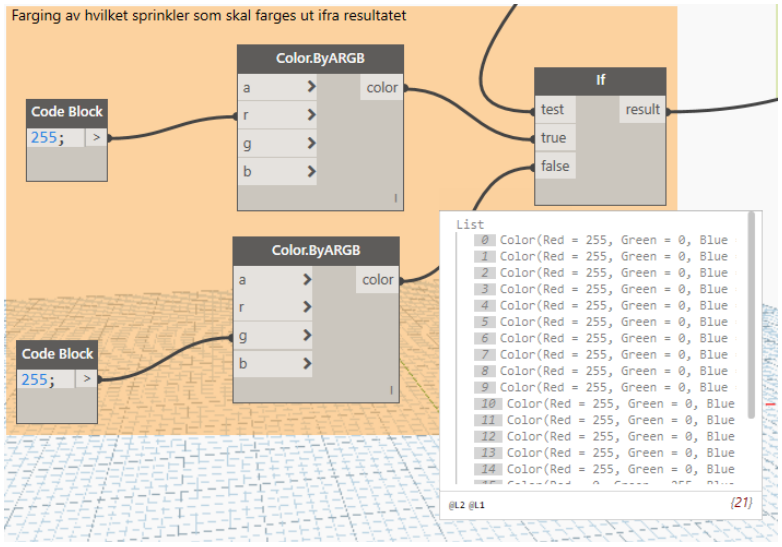
Hensikten bak denne delen, *Figur 93,* er å filtrere falske og sanne verdier. Dersom avstanden mellom en sprinkler og en vegg er kortere enn 1000 mm, vil verdien i noden «==» tilsvare sant. Dersom noen av listene inneholder en eller flere sanne verdier, har sprinkleren for kort avstand til en vegg. Falsk verdi vil si at sprinkleren er i god avstand i forhold til vegg. En kan da se om listene i noden «==» inneholder flere sanne eller falske verdier. Noden «*List.ContainsItem»* benyttes for å filtrere listene, slik at hvis en av listene inneholde en sann verdi, skal hele listen bli sann.



Figur 93 – Filtrering av sanne og falske verdier

Del 3

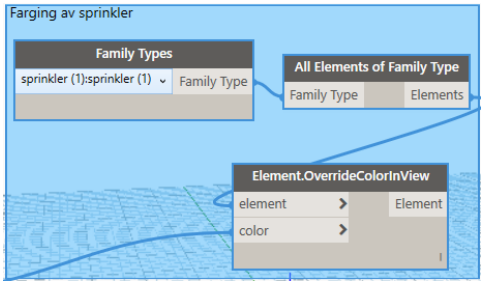
I denne delen av skriptet benyttes «*If*» noden for å avgjøre hvilken sprinkler som skal farges. Testinput til noden kobles fra output noden «List.ContainsItem». Sann verdier blir farget til rødt og falsk verdi blir farget til grønt ved hjelp av nodene «*Color.ByARGB*» *og «Code Block»,* og dette kobles til input True og False i noden. *«Code Block»* definerer fargene som brukes for visualisering av kollisjon.



Figur 94 – Bruk av «If» noden for å definere farger

Del 4

I den siste delen av skriptet, benyttes noden «*Element.OverrideColorInView*», for å farge sprinklene ut i fra resultatet på «*If*» noden.



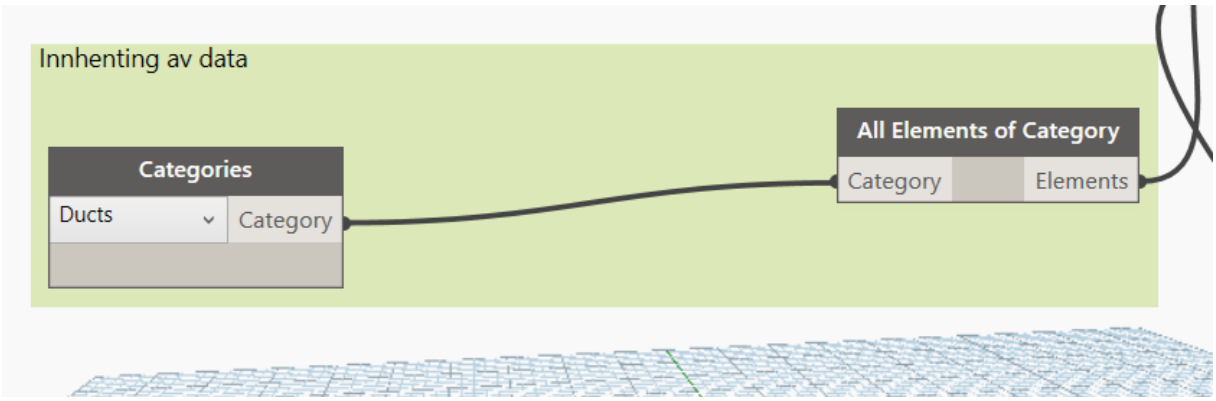
Figur 95 – Farging av sprinklene

13.3.6 Skript 13 – Dimensjonsteksting av diameter for kanaler

Skriptet vil plassere dimensjonstekst på kanaler som er lengre enn 900 mm, og vil plassere det på midtpunktet av kanalen. Grunnen til at vi har valgt kanaler som er lengre enn 900 mm, er for å unngå teksting på små kanallengder.

Del 1 - Kategori

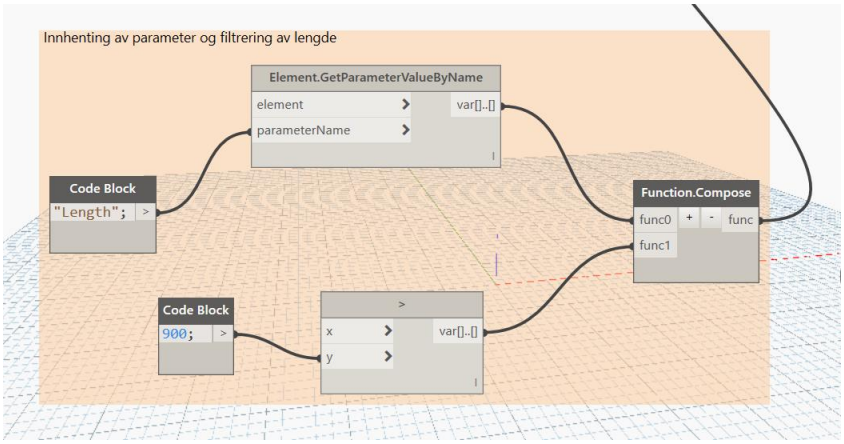
Den første delen vist i *Figur 96,* er en samling av hvilken kategori som skal behandles. I dette tilfellet er det kanaler som skal analyseres i prosjektet. Nodene som har blitt brukt er «*Categories» og «All Elements of Category».*



Figur 96 – Innhente elementer av kategori

Del 2 – Parameter og filtrering

Neste del, *Figur 97,* er å samle lengdene av kanalene ved å bruke noden «Element.GetParameterValueByName». Lengdene skal dog filtreres slik at kanaler med mindre enn 900 mm, blir plassert i en egen liste. For filtrering benyttes noden «Code Block».

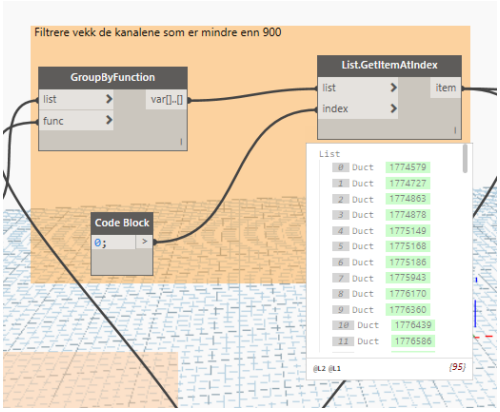


Figur 97 – Innhenting av parametere og filtrering av lengder

Del 3 – Fjerning av ønsket kanallengde

Videre fjernes kanaler som er mindre enn 900 mm, ved å benytte noden «GroupByFunction». Noden definerer to lister, hvor liste 0 er de kanalene som er under 900 mm og liste 1 er de kanalene som er over 900 mm.

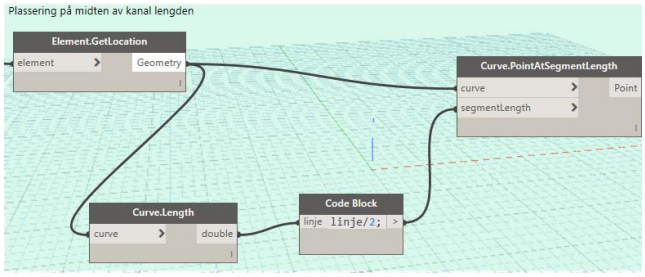
Noden «List.GetItemAtIndex» benyttes for å fjerne liste 0 ved hjelp av «Code Block».



Figur 98 – Filtrering av kanaler

Del 4 – Plassering av teksting

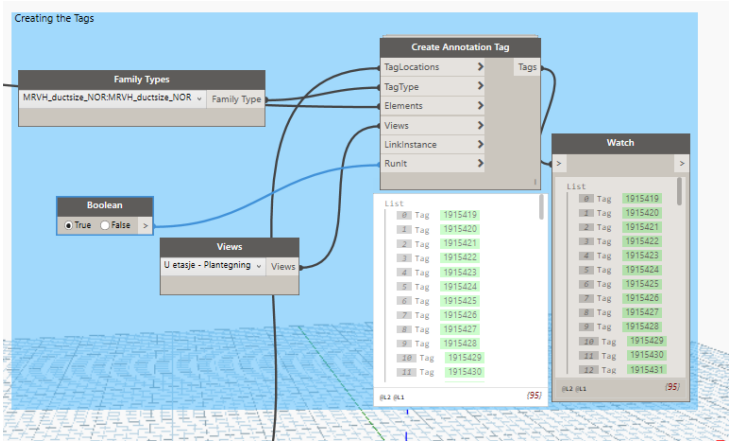
Denne delen definerer dimensjonstekstingens lokasjon, og sørger for at den plasseres sentralt på kanalen. For å utføre dette, brukes en «Code Block».



Figur 99 – Definerer lokasjon for teksting på kanaler

Del 5 – Utplassering av dimensjonstekst

Siste del kobler alle gruppene til noden «*Create Annotation Tag*». «*TagLocations*» blir tilkoblet fra outputen i noden «*Curve.PointAtSegmentLength».* Noden *«Family Types»,* benyttes for å avgjøre type teksting. Inputen «*Views*» bestemmer hvilken visning kanalene skal dimensjonstekstes i.



Figur 100 – Resultat skript 13

14 Kildeliste

[1] A. R. H. Shirley Gregor, «Positioning and presenting design science research for maximum impact,» *Reasearch essay,* vol. 37, nr. 2, pp. 337-355, 2013.

[2] C. Eastman, P. Teicholz, R. Sacks og K. Liston, BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors, 2nd Edition, WILEY, 2011.

[3] M. Lutz, Learning Python, Fifth Edition., California: O'Reilly Media, Inc., 2013.

[4] Revit API Docs, «Revit API Docs - Online Documentation for the Revit API,» [Internett]. Available: http://www.revitapidocs.com/. [Funnet 10 1 2018].

[5] Standard Norge, «NS 3031,» Standard Norge, 2018. [Internett]. Available: https://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=702386. [Funnet 12 mars 2018].

[6] Standard Norge, «SN/TS 3031,» Standard Norge, 2018. [Internett]. Available: http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=859500. [Funnet 12 mars 2018].

[7] Standard Norge, «NS-EN 12831,» Standard Norge, 2018. [Internett]. Available: https://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=941524. [Funnet 12 mars 2018].

[8] Standard Norge, «NS 3055,» Standard Norge, 10 1 1989. [Internett]. Available: http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=133346. [Funnet 16 3 2018].

[9] Standard Norge, «NS-EN ISO 29481,» Standard Norge, 2018. [Internett]. Available: https://buildingsmart.no/article360.html. [Funnet 28 mars 2018].

[10] Code Academy, «Code Academy,» Code Academy, 2018. [Internett]. Available: https://www.codecademy.com/learn. [Funnet 13 Mars 2018].

[11] K. E. Tranøy, «Store norske leksikon,» 14 Februar 2009. [Internett]. Available: https://snl.no/John_Dewey. [Funnet 14 Mars 2018].

[12] Psychestudy, «Psychestudy,» 2018. [Internett]. Available: https://www.psychestudy.com/behavioral/learning-memory/trial-error-learning. [Funnet 14 Mars 2018].

[13] Work breakdown structure, «Work breakdown structure,» 2018. [Internett]. Available: www.workbreakdownstructure.com/. [Funnet 20 Mars 2018].

[14] K. Sander, «estudie,» [Internett]. Available: https://estudie.no/kvalitative-metoder/. [Funnet 20 Mars 2018].

[15] «The Dynamo Primer,» Autodesk, 2017. [Internett]. Available: dynamoprimer.com/.

[16] Khan Academy, «Kahn academy,» Kahn academy, 2018. [Internett]. Available: https://www.khanacademy.org/. [Funnet 13 Mars 2018].

[17] Autodesk inc, «Dynamobim,» Autodesk inc, 2016. [Internett]. Available: dynamobim.org/. [Funnet 13 Mars 2018].

[18] H. K. E. A. D. M. D. Algunerhan, «Productivity improvement of reinforcement drawing generation: a DSR perspective,» Høgskolen i Oslo og Akershus, 2016.

[19] Cansel, «YouTube - Using Dynamo to override the graphics of Revit elements that are Missing Pressure Drop,» 4 1 2016. [Internett]. Available: https://www.youtube.com/watch?v=7icfG-_NvLw. [Funnet 14 1 2018].

[20] D. Forum, «Forum.dynamobim,» [Internett]. Available: https://forum.dynamobim.com/t/override-color-based-on-true-false-values/19965/3. [Funnet 13 Mars 2018].

[21] P. Pavlov, «AUTOMATION OF INFORMATION FLOW FROM REVIT TO BSIM USING DYNAMO,» Ålborg Universitet, Ålborg, 2015.

[22] V. Subbaiah, «dynamobim.org,» [Internett]. Available: http://dynamobim.org/forums/topic/take-area-of-each-wall-inside-rooms/. [Funnet 16 Februar 2018].

[23] SINTEF - Byggforsk, «553.004 - Dimensjonering av avløpsrør,» Byggforskserien, 2011. [Internett]. Available: https://byggforsk.no/dokument/543/dimensjonering_av_avloepsroer. [Funnet 26 3 2018].

15 Figurliste

16 Tabelliste

17 VEDLEGG

Vedlegg I – Selvstudie

Dynamo - Selvstudie

Leksjonene er å finne ved å følge følgende lenker:

Guide - Lesestoff	Video tutorials
http://dynamoprimer.com/en/	http://dynamobim.org/learn/#res

BASIC TUTORIALS

Uke	Guide - Lesestoff	Video tutorials
41	1.1 – 1.3	Getting started with Dynamo
42	2.1 – 2.4	The anatomy of a definition
43	3.1 – 3.4	Data Managment
44	4.1 – 4.5	Nested list managment
45	5.1 – 5.8	Computational logic part 1
46	6.1 - 6.4	Computational logic part 2
47	7.1 - 7.4	Parametric assembely part 1
48	8.1 – 8.6	Parametric assembely part 2
49	9.1 – 9.5	Code Blocks
50	10.1 – 10.5	Custom Nodes
51	11.1 – 11.2	-
52	12.1 – 12.3	-
1	13.1 – 13.4	-

ADVANCED TUTORIALS

Uke	Guide - Lesestoff	Video tutorials
41	1.1 – 1.3	Unit 1: Introduction & Editing elements in Revit
42	2.1 – 2.4	Unit 1: Editing elements in Dynamo & Editing multiple parameters
43	3.1 – 3.4	Unit 1: Editing with formulas & Unit 2: Structural framing part 1
44	4.1 – 4.5	Unit 2: Structural framing part 2 & Family by point part 1
45	5.1 – 5.8	Unit 2: Family by point part 2 & Family by point part 3
46	6.1 - 6.4	Unit 2: Adaptive components part 1 & 2
47	7.1 - 7.4	Unit 2: Adaptive levels & Unit 3: Working with sats part 1
48	8.1 – 8.6	Unit 3: Working with sats part 2 & 3
49	9.1 – 9.5	Unit 3: Dynamo geometry to Revit & Unit 4: Import sats into Dynamo
50	10.1 – 10.5	Unit 4: Excel Read part 1 & 2
51	11.1 – 11.2	Unit 4: Excel Read part 3 & 4
52	12.1 – 12.3	Unit 5: Excel write & Unit 6: Solar Orientation
1	13.1 – 13.4	Unit 7: Code block part 1, part 2 & Python

Fremgangsmåte

Alt kursmateriell er veldig rettet mot geometri. Her er det viktig at vi klarer å overføre informasjonen lært i hver tutorial til HVAC. Dette innebærer at alle lager en egen vri av gjennomgått materiell i

HVAC sammenheng, altså i en MagiCAD fil med definerte objekter og grenser. Bruk gjerne Sykehjem ARK modellen og MagiCAD i dette arbeidet.

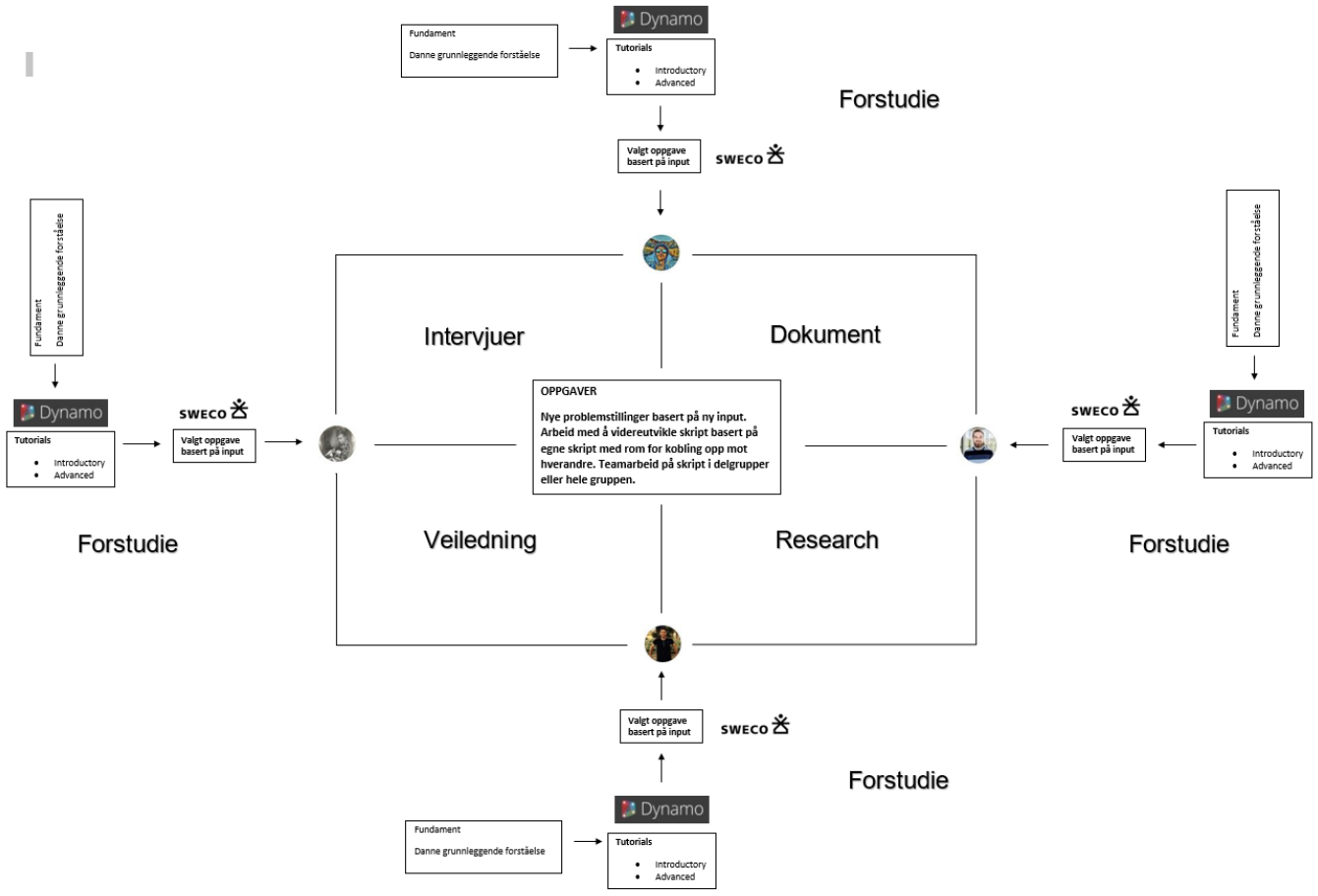
Arbeidsoppgave til hver enkel

- Lag en egen HVAC vri på gjennomgått tutorial og last den opp på: OneDrive/ Effektiv_Prosjektering_Dynamo/Opplæring/Unit (X)/VVS script
- Navngi filen slik: Unit 4_Excel_Read_part _1_navn for gjennomgått kurs slik tutorialen er og en Unit 4_HVAC_Excel_Read_part _1_navn
- Både script fra opplæringen og eget script med VVS vri lastes opp med navn

HVAC Opplæring

Følgende er opplæring som skal gjennomgås til spesifisert tid

Uke	Fagstoff	Lenke
45	Using Dynamo for MEP Design – Part 1	Klikk her
48	Using Dynamo for MEP Design – Part 2	Klikk her
52	An MEP Engineer's Guide to Dynamo	Klikk her



Vedlegg II – Intervjuer

Effektivisering av prosjekteringsoppgaver innen VVS ved bruk av visuell programmering
Bransjen er inne i et digitalt skifte som påvirker måten vi prosjekterer på. Nye programvarer og prosjekteringsverktøy som støtter BIM har kommet på markedet og blir implementert i hverdagslige oppgavene til rådgivere og arkitekter. Selv om programvarer effektiviserer måten vi jobber på er det fremdeles mange tidskrevende oppgaver som kan effektiviseres. Slike oppgaver kan automatiseres ved bruk av visuell programmering. Bruk av teknikken gir mange anvendelsesområder.
Oppgaven går ut på bruk av parametrisk modellering ved bruk av Dynamo for Revit som er et visuelt programmeringsverktøy. Anvendelsesområde er innen VVS sammenheng. Per dags dato er verktøyet mye brukt av til geometrisk behandling i en BIM modell av arkitekter som eksempelvis Snøhetta. VVS fagfeltet er fremdeles relativt lite utforsket og byr på mye potensiale.

Oppgave består av å:

- Gjennomføre en forstudie ved research og intervjuer av rådgivere for å avdekke hvilke problemstillinger som foreligger per dags dato som kan automatiseres.

- Effektivisere prosesser som er tidskrevende men kan automatiseres for VVS ingeniører i prosjekteringsarbeidet ved å utvikle skript som effektiviserer prosjekteringsarbeidet.

Hva trenger vi?
Innspill fra erfarne rådgivere hvilke prosesser som per dags dato utføres manuelt i Revit som kan automatiseres. Det vi trenger er mest mulig innspill fra dere som har erfaring med utfordringer i prosjekteringen og å sette lyst på gjentakende arbeidsoppgaver som er tidskrevende. Mest mulig innspill gir oss bedre forutsetninger til å løse problemer.

Vedlegg III – Mail Fra MagiCad

Gmail Pawel <pawel.okland@gmail.com>

Pressureloss MagiCAD - Missing values

4 messages

Pawel <pawel.okland@gmail.com> Thu, Feb 8, 2018 at 2:04 PM
To: mail@magicad.com

I have a inquiry about the pressure loss report and addition of pressure loss to components in the Calculations «Balancing» in MagiCAD.

After I run the calculation I get the report which includes pressureloss in most components. View image_01 . The dpt value gets transferred to the parameter name «MC Ventilation Pressure Drop».

After updating the values to the model I wish to have the values stored in the «MC Ventilation Pressure Drop» parameter for ALL duct components and ducts. However they are not stored.

View image_02. All the red parts dont have any value transferred to «MC Ventilation Pressure Drop» parameter. What seems to be recurring in the lack of pressureloss value is on the parts which split the flow, so a duct which is connected to several smaller networks where flow gets seperated.

Is there any way to fix this in the future? Like a built in function that shows the accumulated pressure loss at a given lenght for calculation control procedures preformed by other team members?

I also notice that reducers dont get calculated as well. At least I dont see the value for dpt in the pressure loss raport.

Regard

Pawel Økland

2 attachments

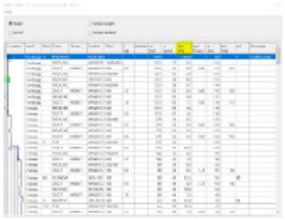


image_01.JPG
199K

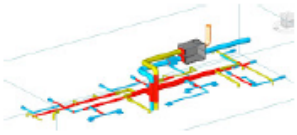


image_02.JPG
91K

Mikael Engström <mikael.engstrom@magicad.com> Fri, Feb 9, 2018 at 9:50 AM
To: "pawel.okland@gmail.com" <pawel.okland@gmail.com>
Cc: Lesley Kiernan <lesley.kiernan@magicad.fr>

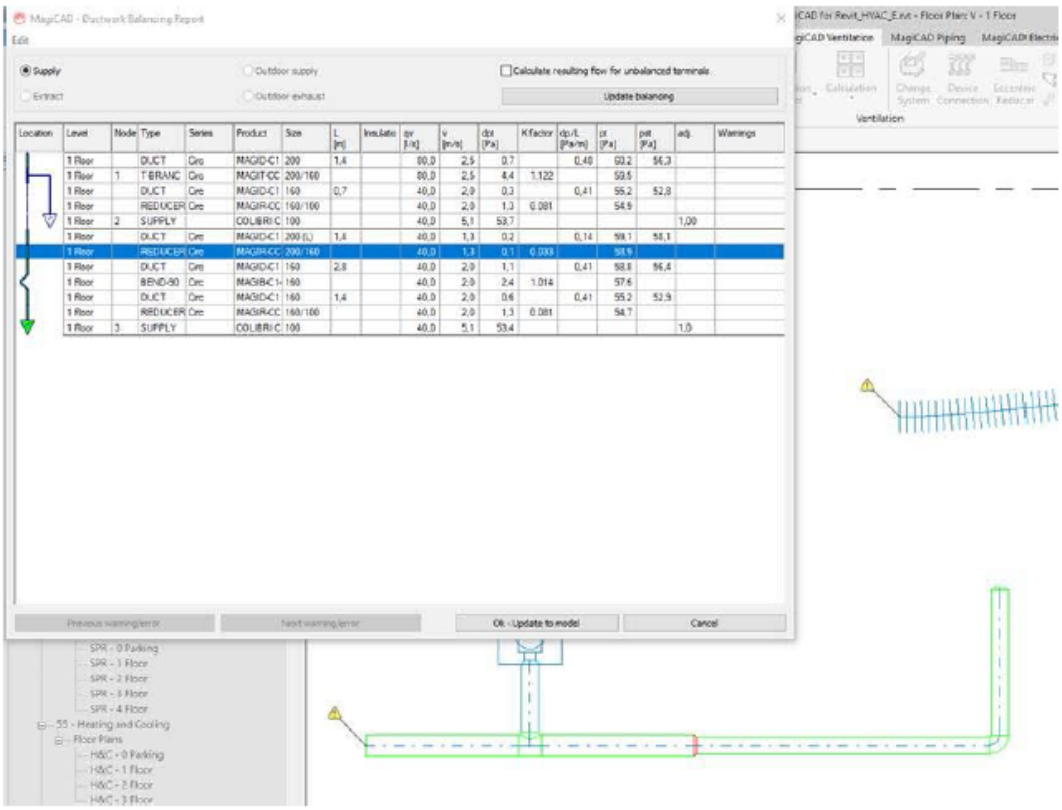
Hi Pawel,

Lesley forwarded to this one to us.

-Yes you are right about that and correct about the reason. The reason is, as you probably guessed, that there are many different pressure losses over the duct but the duct itself is one element.

Your suggestion did I however get a bit curious about: Do you think that we would, for example, add a parameter which instead showed the accumulated pressure loss over the entire duct?

-Regarding the reducers I think that they should be calculated normally. I did a fast test and get the value here. In case you see to have some issues then could you send me a copy of the project and the dataset, as well as some screenshot showing where you run these functions?



Best Regards,

Mikael Engström

Technical support engineer

Help-Desk

tuki@magicad.com

Phone **+358 2 83876022**

Progman Oy | Nortamonkatu 1 | 26100 Rauma, Finland | **www.magicad.com**

MagiCAD – *The professional's choice for Building Services - For AutoCAD and Revit*

From: Pawel [mailto:pawel.okland@gmail.com]
Sent: 08 February 2018 15:05
To: Web request <mail@progman.fi>
Subject: Pressureloss MagiCAD - Missing values

[Quoted text hidden]

Pawel <pawel.okland@gmail.com> Fri, Feb 9, 2018 at 3:04 PM
To: Mikael Engström <mikael.engstrom@magicad.com>

Hi Mikael and thanks for the quick reply. Yes that could work. I see that the value for the entire duct I had for 14 m has been calculated however the value does not get sent to "MC Ventilation Pressure Loss". The pressure loss to different terminals from AHU will ofcourse include segments of a lagre duct with several branches when calculated.

I use Dynamo to override colors of elements which do not have a pressureloss value in the paramter where its stored to check the model for irregularities. And sometimes a manual calculation is done to verify that the pressure loss calculation is correct. If the duct could have added the accumulated pressureloss value for the element to the same parameter as the others.

I can check about the project and get back to you with a link and screenshots when possible.

Regards

Pawel Økland
[Quoted text hidden]

Mikael Engström <mikael.engstrom@magicad.com> Fri, Feb 9, 2018 at 4:33 PM
To: Pawel <pawel.okland@gmail.com>

Hi Pawel,

Hopefully you get it to work with the reducers.

I right now don't know how this visual part of Revit can be connected to such a parameter but maybe it would make the planning in general easier if you had access to that data somewhere, maybe you can in that case use dynamo for that as well.

I will run some test and then in case it seems OK, then I'll make a suggestion to add such a parameter to the program.

Best Regards,

Mikael Engström

Technical support engineer

Help-Desk

tukl@magicad.com

Phone **+358 2 83876022**

Progman Oy | Nortamonkatu 1 | 26100 Rauma, Finland | **www.magicad.com**

MagiCAD – *The professional's choice for Building Services - For AutoCAD and Revit*

From: Pawel [mailto:pawel.okland@gmail.com]
Sent: perjantai 9. helmikuuta 2018 16.05
To: Mikael Engström <mikael.engstrom@magicad.com>
Subject: Re: Pressureloss MagiCAD - Missing values

[Quoted text hidden]

Vedlegg IV – Bruk av Python node for skript 02

```python
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN

typebygg = IN[0]
egenpersonfaktor = IN[1]

"""Inputverdier er fra Tabell H.2 fra NS 3031 er Bygningskategori med følgende parametere
som representerer verdiene i hver liste nedenfor:

| Minste primærareal |
| Største sekundærareal |
| Minste persontetthet, primærareal |
| Minste tilstedeværelse, primærareal | """

if typebygg == "Barnehage":
    typebygg = [0.70 , 0.30 , 5, 0.60]
elif typebygg == "Kontor":
    typebygg = [0.65 , 0.35 , 5, 0.60]
elif typebygg == "Skole":
    typebygg = [0.70 , 0.30 , 2.5, 0.60]
elif typebygg == "Universitet/høgskole":
    typebygg = [0.70 , 0.30 , 4, 0.70]
elif typebygg == "Sykehus":
    typebygg = [0.75 , 0.25 , 5, 0.70]
elif typebygg == "Sykehjem":
    typebygg = [0.75 , 0.25 , 5, 0.70]
elif typebygg == "Hotell":
    typebygg = [0.60 , 0.40 , 6, 0.50]
elif typebygg == "Idrett":
    typebygg = [0.80 , 0.20 , 5, 0.60]
elif typebygg == "Forretning":
    typebygg = [0.70 , 0.30 , 4, 0.75]
elif typebygg == "Kultur":
    typebygg = [0.70 , 0.30 , 4, 0.60]
elif typebygg == "Lett industri":
    typebygg = [0.70 , 0.30 , 4, 0.60]
else:
    typebygg = egenpersonfaktor

#Assign your output to the OUT variable.
OUT = typebygg
[VEDLEGG] – EFFEKTBEHOV TIL OPPVARMING PYTHON
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN

typebygg = IN[0]
egenverdi = IN[1]

```

Vedlegg V – Bruk av Python node for skript 03

```python
import clr
clr.AddReference('ProtoGeometry')
clr.AddReference("RevitNodes")
from Autodesk.DesignScript.Geometry import *

#Definerte inputverdier

Filterliste = IN [0]
Ventiler = IN [1]
Luftromnavn = IN [2]
Luftventil = IN [3]
Luftromnummer = IN [4]
NavnSpaces = IN [5]
NummerSpaces = IN [6]
LuftSpaces = IN [7]


"""Dette skriptet gjelder for overføring av informasjon fra Spaces til parameter i ventiler"""

#Romnummer

for info in Ventiler:
    info.SetParameterByName(Luftromnummer, int())

for filter, space in zip(Filterliste, NummerSpaces):
    for objekt, tilstede in zip(Ventiler, filter):
        if tilstede:
            objekt.SetParameterByName(Luftromnummer, space)

#Romnavn

for info in Ventiler:
    info.SetParameterByName(Luftromnavn, "")

for filter, space in zip(Filterliste, NavnSpaces):
    for objekt, tilstede in zip(Ventiler, filter):
        if tilstede:
            objekt.SetParameterByName(Luftromnavn, space)

#Luftemngde rom

for info in Ventiler:
    info.SetParameterByName(Luftventil, int())

for filter, space in zip(Filterliste, LuftSpaces):
    for objekt, tilstede in zip(Ventiler, filter):
        if tilstede:
            objekt.SetParameterByName(Luftventil, space)


#Assign your output to the OUT variable.

OUT = Ventiler

```

Vedlegg VI – Bruk av Python node for skript 07

```python
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN

typebygg = IN[0]
egenverdi = IN[1]


"""Inputverdier er fra Tabell A.9 fra SN/TS 3031 er Bygningskategori med følgende parametere som representerer verdiene i hver liste nedenfor:

| Settpunkttemperatur for oppvarming i driftstiden |"""

if typebygg == "Barnehage":
    typebygg = [22 ]
elif typebygg == "Kontor":
    typebygg = [22]
elif typebygg == "Skole":
    typebygg = [21]
elif typebygg == "Universitet/høgskole":
    typebygg = [21]
elif typebygg == "Sykehus":
    typebygg = [21]
elif typebygg == "Sykehjem":
    typebygg = [22]
elif typebygg == "Hotell":
    typebygg = [21]
elif typebygg == "Idretts":
    typebygg = [19]
elif typebygg == "Forretning":
    typebygg = [21]
elif typebygg == "Kultur":
    typebygg = [21]
elif typebygg == "Lett industri":
    typebygg = [21]
else:
    typebygg = egenverdi

#Assign your output to the OUT variable.
OUT = typebygg


```

Vedlegg VII – Bruk av Python node for skript 09

```
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN

typebygg = IN[0]
kurveA = IN[1]
kurveB = IN[2]
standard = IN[3]
standard1 = IN[4]

if typebygg == "Sykehus":
    if len(kurveB) == 0:
        typebygg = standard1
    else:
        typebygg = kurveB
elif typebygg == "Hotell":
    if len(kurveB) == 0:
        typebygg = standard1
    else:
        typebygg = kurveB
elif typebygg == "Kino":
    if len(kurveB) == 0:
        typebygg = standard1
    else:
        typebygg = kurveB
elif typebygg == "Forsamlingshus":
    if len(kurveB) == 0:
        typebygg = standard1
    else:
        typebygg = kurveB
elif typebygg == "Skole":
    if len(kurveB) == 0:
        typebygg = standard1
    else:
        typebygg = kurveB
elif typebygg == "Bolig":
    if len(kurveA) == 0:
        typebygg = standard
    else:
        typebygg = kurveA
elif typebygg == "Forretningsbygg":
    if len(kurveA) == 0:
        typebygg = standard
    else:
        typebygg = kurveA
elif typebygg == "Aldershjem":
    if len(kurveA) == 0:
        typebygg = standard
    else:
        typebygg = kurveA
else:
    typebygg = "Error"

#Assign your output to the OUT variable.
OUT = typebygg
```

Vedlegg VIII – Bruk av Python node for skript 10

```
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN
a=IN[0]
x = []
if a == 0:
    x = "Parameteren Funksjon mangler i prosjektet!"
else:
    x = "OK!"

#Assign your output to the OUT variable.
OUT = x
```