AI-Drevet Kontrol & Informationssøgning i BIM

Integration af Kunstig Intelligens, Semantiske Webteknologier, Linked Data og BIM-Modeller til ACC og informationssøgning i naturligt sprog

20225405 Mads S. Martiny

20225408 Mili Becirbasic

20225409 Jonas Nedergaard

Aalborg Universitet
AAU BUILD
Institut for Byggeri, By og Miljø

Afleveret

09-01-2025

TITELBLAD

AAU BUILD

Institut for Byggeri, By og Miljø

Thomas Manns Vej 23

9220 Aalborg Øst

Uddannelse:

Cand. Tech. Byggeledelse og Bygningsinformatik

Projekttitel:

AI-drevet Kontrol & Informationssøgning i BIM

Projektperiode:

2/9/24 – 09/01/25

Semestertematik:

Kandidatspeciale

Vejleder:

Kjeld Svidt

Studienumre, forfattere og underskrifter:

20225405 Mads S. Martiny

20225408 Mili Becirbasic

20225409 Jonas Nedergaard

Anslag (m. mellemrum):	317093
Normalsider (2400 anslag):	109,24
Sideantal:	166
Antal bilag:	36

FORORD

Rapporten er skrevet som en del af det afsluttende semester på kandidatuddannelsen Cand.Tech i Bygningsinformatik på Aalborg Universitet. Kandidatspecialet omhandler en undersøgelse af, hvordan et system potentielt kan designes, så Kunstig Intelligens, Semantisk Web Teknologi, herunder Linked Data og BIM-værktøjer kan anvendes til at kontrollere BIM-data og understøtte projekteringsprocessen. For at opnå og eftervise interoperabilitet mellem disse teknologier og software er der udviklet både tekniske, visuelle og konceptuelle prototyper for at demonstrere systemets brugergrænseflader og kernefunktioner. Undersøgelsen er baseret på et litteraturstudie, der tager afsæt i danske og internationale kilder samt interviews afholdt med relevante fagpersoner, der har en relation til i Bygge- og Anlægsbranchen. Interviewpersonernes viden og generelle indblik i branchen har bidraget til en større forståelse af emnet og de tilhørende behov og krav, som det undersøgte system skulle imødekomme.

Vi ønsker at udtrykke vores taknemmelighed overfor de interviewpersoner, der har bidraget med deres indsigt, samt øvrige personer, der har ydet væsentlig støtte ifm. med udarbejdelsen af vores speciale. Desuden vil vi rette en stor tak til vores vejleder, Kjeld Svidt, for hans vejledning med specialet og de øvrige tre semestre.

LÆSEVEJLEDNING

I denne undersøgelse er IMRAD anvendt som rapportstruktur. Valget af IMRAD skyldes, at rapportstrukturen skaber en logisk sammenhæng for læseren. Det giver samtidig en kontinuerlig forståelse af det afdækkede område, som præsenteres i en kronologisk rækkefølge. IMRAD-strukturen består af følgende hovedafsnit: (1) *Indledning* (2) *Metode* (3) *Resultater* (4) *Analyse* (5) *Diskussion*. Afsnittenes indhold og formål er præsenteret i indledningen af hvert afsnit. Rapportstrukturen er tilpasset, så den yderligere består af det indhold, der er beskrevet i Contextual Design (CD), som udgør undersøgelsesdesignet i rapporten. Undersøgelsens indhold ses i nedenstående *Tabel 1*.

Rapportstruktur

IMRAD	Kapitel	Beskrivelse
INDLEDNING	Problemformulering	Beskrivelse af undersøgelsens problematik, hypotese og samlet problemformulering
	Afgræsning	Afgrænsning beskriver teoretiske og praktiske afgrænsning som ikke afdækket i undersøgelsen
METODE	Undersøgelsesdesign	Præsentation af anvendelsen af Contextual Design
	Datagrundlag	Metodisk fremgangsmåde for dataindsamling
	Dataanalyse	Fremgangsmetode for analyse og bearbejdning af data
RESULTATER	Begreber	Relevante begreber anvendt i undersøgelsen
	State of the Art	Litteraturstudie af relevante teori anvendt i branchen
	Primær empiri	Præsentation af indsamlet kvalitativ empiri
ANALYSE	Analyse	Komparativ analyse samt workmodels og systemkrav
	Prototyper	Præsentation af udarbejdet interface og teknisk prototype
	Udførelse af test	Feedback af udarbejdede prototyper
DISKUSSION	Diskussion	Diskussion af problemstillinger præsenteret i problemformuleringen
	Konklusion	Samlet konklusion af problemformulering

Tabel 1 – IMRAD Rapportstruktur – Med inspiration i (Collidu, 2024)

Den opstillede rapportstruktur opfordrer læseren til at læse rapporten kronologisk. Undersøgelsen anvender tekniske begreber, hvor de hyppigst anvendte er præsenteret i *Kapitel 3*. Hvis læseren har en større teknisk forståelse af undersøgelsens emner, kan dette afsnit undlades eller bruges som opslagsværk. Metodeafsnittet indeholder en forklaring af de metodiske tilgange og deres anvendelse i undersøgelsen. Afsnittet *Forkortelser* definerer de hyppigt anvendte forkortelser i undersøgelsen samt deres betydning.

Undersøgelsen angiver referencer på to forskellige måder. Begge tager udgangspunkt i Harvard-Anglia, men de adskiller sig ved, at kilder placeret før et punktum refererer til enkeltstående sætninger, mens kilder efter punktummet refererer til det foregående afsnit.

ABSTRACT

This report explores how Artificial Intelligence, Semantic Web and Linked Data can be integrated with BIM and regulatory frameworks to support the Danish construction industry. Globally, the adoption and acceptance of AI across industries are growing rapidly. While the Danish construction industry has embraced these technologies, there remains a significant need to fully understand and optimize their potential benefits. AI is recognized as a tool with immense potential, but a lack of understanding, competencies, and readiness hinders the identification of appropriate application areas for the technology.

This study employs the Contextual Design methodology to develop a system that leverages AI and Linked Data in conjunction with BIM-models to enhance Automated Compliance Checking and information retrieval within the Danish construction industry. The research involved interviews with industry professionals and a comprehensive review of theoretical literature to understand how AI and Linked Data can streamline specific workflows in the construction sector and potentially be applied in other contexts.

To support the application of Contextual Design and understand user needs and requirements for a future system, a Usability Test was conducted. This test provided valuable insights into user perceptions of the potential synergies between these technologies. Feedback from the test was used to contextualize the collected empirical data.

The findings indicate that AI can facilitate the interpretation of structured and unstructured data through natural language processing and highlight the significant potential of creating an integrated system to support users' workflows. However, as the industry currently lacks the necessary competencies and readiness, it is crucial to establish consensus on standards and tailor these technologies to specific tasks that AI should perform.

FORKORTELSER

FORKORTELSE	BETYDNING
DB	Database
GDB	Grafdatabase
VDB	Vektordatabase
RDB	Relationelle Databaser
HF	Huggingface
AI/KI	Artificial Intelligence/Kunstig Intelligens
ML	Machine Learning
RDF	Resource Description Framework
BIM	Building Information Model/Modelling/Management
LLM	Large Language Model
LD	Linked Data
ACC	Automated Compliance Checking
SMC	Solibri Model Checker
DL	Deep Learning
OWL	Web Ontology Language
SPARQL	SPARQL Protocol and RDF Query Language
SHACL	Shapes Constraint Language
SWRL	Semantic Web Rule Language
TTL	Turtle file
TS	Triple Store
NL	Natural Language / Naturligt sprog
NLP	Natural Language Processing
SW/SWT	Semantic Web / Semantic Web Technology
GI	Generative AI
RAG	Retrieval Augmented Generation
SWO	Stopwords
IFC	Industry Foundation Classes
CD	Contextual Design

INDHOLDSFORTEGNELSE

1 INDLEDNING

Dansk erhvervsliv er velpositioneret med en stærk konkurrenceevne og et solidt fundament for fremtidig vækst. Verdensordenen er præget af globale stridigheder, som udfordrer konkurrenceevnen, men samtidig skaber nye erhvervsmuligheder. Den seneste tid har vist, hvordan diverse kriser er opstået, og spørgsmålet er derfor, om Danmarks konkurrenceevne ift. omverdenen er så god, at den kan stå imod de fremtidsmæssige potentielle udfordringer, der kan opstå. Flere mulige tendenser er identificeret, som kan udfordre dansk konkurrenceevne, hvor specielt én tendens, i form af teknologiske gennembrud, er af særlig relevans. Teknologiske gennembrud anses for at være en af de primære katalysatorer for produktivitet og forandring i verden fordi, at teknologisk udvikling skaber nye markeder for produkter og services. Med digitale løsninger, automatisering og data kan virksomheder gentænke og rationalisere arbejdsgange, som kan forbedre deres konkurrenceevne. Et teknologisk kapløb er i gang, hvor målet er at være førende inden for udviklingen af nye teknologier, især med USA og Kina som frontløbere. Modsat har EU gennem de seneste årtier byttet sig til teknologier, der er udviklet i andre dele af verdenen, hvorefter EU er gået forrest ift. at fastlægge standarder og regulering for anvendelsen af de mange nye teknologier. En af de aktuelle fremtrædende teknologier er Kunstig Intelligens (KI), som EU nu introducerer reguleringer og standarder for. (Erhvervsministeriet, 2023).

Europa Kommissionen har i 2018 fremlagt en plan, der er udviklet i samarbejde med medlemsstaterne, som har til formål at fremme udviklingen og anvendelsen af KI i Europa. En plan som arbejder inden for fire centrale fokusområder: (1) Forøgelse af investeringer i KI (2) Tilgængeliggørelse af mere data og europæiske dataområder (3) Fremme talent og færdigheder i KI (4) Udvikle etisk og pålidelig KI. Dette bestod, at næsten alle medlemsstater i 2019 udviklede deres egne nationale strategier for anvendelsen af KI. (Europa-Kommissionen, 2018).

I 2018 udarbejdede Erhvervsministeriet en Strategi for Danmarks digitale vækst, hvor KI identificeres som en teknologi, der kan anvendes bredt med store og afgørende erhvervsmæssige potentialer på tværs af brancher og sektorer, og hvor et større samt alternativt brug af data bliver en stor kilde til vækst (Erhvervsministeriet, 2018). I forlængelse af Europa-Kommissionens plan om KI har den danske regering i 2019 udarbejdet en national strategi for KI, som tager sit udgangspunkt i de fire centrale områder udarbejdet af Europa-Kommissionen. Den nationale strategi har en vision om, at Danmark skal gå forrest med ansvarlig udvikling og anvendelse af KI. Heri fremgår det, at KI skal anvendes med danske fælles værdier om frihed, frisind, tryghed og ligeværd i centrum. Derudover skal det sikres, at dataene, som den kunstige intelligens benytter, ikke er fejlbehæftede. Danmark er et af de mest digitaliserede lande i verden, og denne styrkeposition skal bruges til at trække viden og teknologier til Danmark og præge udviklingen af ansvarlig KI i tæt samarbejde med de øvrige europæiske lande. Hvis ikke der handles hurtigt og

velovervejet, mister Danmark både konkurrenceevne og indflydelse på udviklingen af KI. KI skal samtidig hjælpe det danske erhverv med at analysere, forstå og træffe bedre beslutninger. Men teknologien kan og skal ikke erstatte mennesker eller træffe beslutninger. For at Danmark skal komme i mål med den nationale strategi, er der udfordringer, som der skal tages hånd om. Der er et behov for fælles retningslinjer og etiske rammer for KI. Dernæst er der behov for flere dansksprogede træningsdata til KI og medarbejdere med de rette digitale kompetencer. Danmark rangerer sig under de lande, som nationen normalt sammenlignes med, når det kommer til private investeringer i KI, hvilket kan gå hen og udfordre danske virksomheders konkurrenceevne. Brugen af offentlige data i danske virksomheder er forsat lav og flere offentlige datasæt er ikke tilgængelige for forskere og virksomheder. Derudover er de datasæt, som er tilgængelige i for ringe grad til at udvikle nye intelligente løsninger. (Finansministeriet og Erhvervsministeriet, 2019).

En sådan teknologisk udvikling kan have store konsekvenser ift. virksomheders og Danmarks konkurrenceevne. Det kan ydermere være problematisk for en branche som byggebranchen, som i forvejen er blandt de mindst digitaliserede sektorer i Danmark (Transport-, Bygnings- og Boligministeriet, 2019). I *Strategien for digitalt byggeri* beskrives byggebranchens produktivitet og udvikling som værende haltende ift. andre erhverv, hvor digitale værktøjer anses for at være én løsning på branchens udfordringer. Det understreges yderligere, at automatisering af byggeriets processer har et stort produktivitetspotentiale men også, at der er en mangel på digital parathed hos en eller flere aktører i et byggeprojekts værdikæde, hvilket kan udgøre en barriere for, at de digitale muligheder og produktiviteten udnyttes fuldt ud. Samtidig understreges vigtigheden af at anvende BIM i flere af byggeriets processer i fremtiden. Strategien for digitalt byggeri behandler delvist de samme fokusområder, som dem der er grundlæggende for Europa-kommissionens plan for anvendelsen af KI. Her er fem indsatsområder: (1) Bedre udnyttelse af digitale værktøjer (2) Åbne formater og fælles standarder (3) Bedre udnyttelse af data (4) Digitale kompetencer til hele værdikæden (5) Mere bæredygtigt byggeri gennem digitalisering. (Transport-, Bygnings- og Boligministeriet, 2019).

Det er tydeligt, at både Europa og Danmark, som selvstændigt land, er bevidste om det store potentiale ved anvendelse af KI. Det står samtidig klart, at de fokuspunkter og indsatsområder som Europa-Kommissionen, de nationale strategier for KI, digital vækst og digitalt byggeri arbejder med, bygger på de samme ideer, udfordringer og løsninger. Det er derfor indlysende, at KI vil spille en afgørende rolle i arbejdet med at forbedre byggeriets haltende produktivitet og udvikling samt manglende digitale parathed.

I denne forbindelse har *ConTech Lab (2024)* udgivet en publikation om anvendelsen af KI i byggeriet. Heri identificeres en væsentlig hindring for at realisere potentialet for innovation og effektivisering med KI ved, at mange af byggeriets grundlæggende

processer enten ikke er digitaliserede eller anvender digitale systemer, der mangler interoperabilitet og ikke understøtter effektiv dataoverførsel. Begrænset adgang til data, herunder dataene til træningsformål, kan udgøre en betydelig hindring for optimering af anvendelsen af KI. Ydermere kan fragmenteret og komplekse regulativer og tekniske krav, der enten ikke er opdaterede eller mulige at digitaliserer, skabe strukturelle barrierer for udnyttelsen af KI. Dog ses et stort potentiale i anvendelsen af KI, da teknologien kan bidrage til at styrke overholdelsen af de omfattende og komplekse krav, der er forbundet med byggebranchen. Teknologien kan bidrage til at sikre, at gældende regler og standarder efterleves, f.eks. ved at assistere brugeren med at identificere alle relevante regulativer for en given opgave, hvilket gør det lettere at overholde nødvendige regulativer og krav. Derudover opstår barrierer for KI-udviklingen i byggebranchen, hvilket kan omfatte: (1) Konkurrence mellem virksomheder (2) manglende tillid (3) databeskyttelsesproblemer (4) ikke-harmoniserede tekniske standarder (5) manglende kompetencer (6) lav digitalisering (7) manglede fælles mål. (ConTech Lab, 2024).

Selvom KI vil spille en afgørende rolle i arbejdet med at forbedre byggeriets haltende produktivitet og manglende digitale parathed, skal udfordringer som ovenfornævnte håndteres for, at KI udlever sit fulde potentiale. KI kan som benævnt assistere brugere med at identificere relevante regulativer for en given opgave, hvilket gør det lettere at overholde nødvendige bestemmelser. I denne forbindelse opstår begrebet Automatic Compliance Checking (ACC), som benytter software til at undersøge, om byggeprojekter overholder en række prædefinerede krav. ACC fokuserer på at automatisere størstedelen af krav- og regeltjek i byggeriets faser. Der er stadig en række udfordringer med at implementerer ACC i byggeriet i fuld skala, og ACC-teknologien har derfor brug for et skub i den rigtige retning, hvor den nyeste forskning peger på, at semantiske webteknologier kan give ACC-teknologien det nødvendige skub. Her understreges det i Byggeriets Automatiske RegelTjek-projektet (BART) af *Schlachter, et al. (2021*), som er et samarbejde mellem ni byg- og driftsherrer at:

> *De semantiske webteknologier og linked data tillader et udvideligt udviklingsmiljø som er fordelagtigt ved udvikling af nye regelsæt og funktionaliteter, fordi det bygger på et generelt, alsidigt standard sprog til at beskrive verden kaldet Ressource Description Framework*. (Schlachter, et al., 2021).

I BART-projektet er et litteraturstudie udført, som konkluderer, at mange forskere er enige om, at semantiske webteknologier potentielt kan være løsningen for fremtidens ACC. I denne forbindelse peges der på Shapes Constraint Language (SHACL), som en udvidelse til de semantiske webteknologier i og med, at den gør mere end at formalisere formuleringen af regler i form af afrapportering af resultaterne. Der eksisterer ikke meget viden om anvendelsen af SHACL inden for byggebranchen, men nogle studier har påvist, at teknologien kan have potentiale (Schlachter, et al., 2021). I BART-projektet fremhæves det, at med semantiske webteknologier kan ACC:

(1) Automatisere og effektivisere granskningsprocesser der traditionelt udføres manuelt (2) Forbedre datakvaliteten vha. implicit viden som beriger BIM-modellerne (3) Forsikre om at regler bygger på åbne standarder således, at de, uanset software, er tilgængelige for alle involverede aktører. (Schlachter, et al., 2021).

I og med at de forskellige initiativer og strategier, internationalt og nationalt, pointerer, at KI i fremtiden bliver en teknologi, som kommer til at blive integreret i diverse processer, vil der i denne rapport blive forsøgt at inkorporere KI i ACC-processen. Derudover vil det samtidig blive forsøgt at kombinere KI med SWT og LD og herunder SHACL fordi, at disse teknologier i forvejen forventes at kunne give ACC-processen et skub i den nødvendige retning. Disse forventninger identificeres, ud over i BART-projektet, i diverse videnskabelige artikler (Hagedorn, et al., 2023; Shin & Issa, 2021; Guo, et al., 2021; Ghannad, et al., 2019; Pauwels, et al., 2024). Det er ydermere belyst, at anvendelsen af BIM vurderes vigtig for byggeriets processer i fremtiden (Zhang & El-Gohary, 2015), hvorfor det i rapporten vil blive forsøgt at opnå interoperabilitet mellem KI, SWT herunder LD, SHACL og BIM. Pba. den opnåede viden fra tidligere undersøgelser indenfor emnet forventes det, at en sådan kombination vil kunne højne og rationaliserer ACC-processen. Undersøgelsen har desuden til formål, ud over optimering af ACC-processen, at gøre det muligt for brugere med begrænsede tekniske kompetencer at interagere med BIM-modellen udelukkende vha. NL. Dette vil bidrage til at gøre forespørgsler og informationssøgning både enklere og mere effektiv, hvilket potentielt kan afhjælpe den manglende digitale parathed i branchen.

1.1 Problemformulering

Hvorvidt er det muligt at udvikle et system, der kombinerer Kunstig Intelligens og Semantisk Webteknologi, herunder Linked Data, så brugeren kan interagere med projektinformation, BIM-modeller og gældende regulativer vha. naturligt sprog og derved rationalisere ACC og informationssøgning?

For at besvare problemformulering er følgende underspørgsmål formuleret og vil blive behandlet rapporten:

1. Hvad er den nuværende forskningsstatus og de nyeste teknologiske fremskridt indenfor anvendelsen af Kunstig Intelligens og Semantiske Webteknologier til ACC og informationssøgning inden for byggebranchen?
2. Hvordan er det nuværende workflow struktureret for udførelsen af kontroller vha. ACC-systemer og informationssøgning, samt hvilke holdninger eksisterer der til denne tilgang?
3. Hvad er byggebranchens generelle holdninger og synspunkter til anvendelsen af Kunstig Intelligens, Semantisk Webteknologier herunder Linked Data som

selvstændige teknologier, samt deres potentiale til at understøtte et fremtidigt ACC-system?
4. Hvilke brugerbehov og systemkrav er der til systemet og hvordan kan disse indarbejdes i et fremtidigt workflow?
5. Hvilke potentielle muligheder og udfordringer er der ved udviklingen af det nye system, og hvad er nødvendigt for at sikre dets anvendelse i byggebranchen?

1.2 Afgrænsning

Denne undersøgelse omhandler anvendelsen af Artificial Intelligence (AI), SWT, herunder LD i sammenhæng med BIM-software til understøttelse af projekterings- og kontrolopgaver. I undersøgelsen er der fokuseret på workflows, som foregår i den danske byggebranche. Undersøgelsen har, ud over få nationale kilder, anvendt internationale kilder, der er publiceret i lande, som har et betydeligt sammenligningsgrundlag med Danmark. Viden fra internationale kilder anvendes derfor i denne undersøgelse til at understøtte danske processer og arbejdsgange fordi, at der ikke foreligger tilstrækkeligt med nationale studier inden for emnet.

Emnerne AI, SWT herunder LD er omfattende og indeholder en lang række begreber og termer, der kræver en høj grad af teknisk viden. Undersøgelsen redegør for og anvender disse begreber og teknologier, men er primært begrænset til et teoretisk og konceptuelt niveau med en praktiskorienteret tilgang, hvilket afspejler sig i både anvendelsen og beskrivelsen heraf. Der er derfor et større fokus på at beskrive og synliggøre, hvilke potentialer teknologierne har både selvstændigt og i kombination med hinanden. Indholdet indebærer ikke fuldstændig detaljerede tekniske forklaringer. Dette afspejles i de udviklede prototyper, som har til formål at efterligne elementer af det udtænkte system. Prototyperne er i høj grad udarbejdet med udgangspunkt i eksisterende AI-modeller og teknologier, der er sammensat i et lokalt kontrolleret miljø.

Derudover har undersøgelsen et fokus på Large Language Models (LLM) til anvendelse af Natural Language Processing (NLP) og Generative AI (GI) funktioner. Dette skyldes, at denne type af AI-model understøtter de mest centrale systemfunktioner, som er tiltænkt det nye design.

2 METODE

Dette kapitel præsenterer de metodiske rammer og fremgangsmåder, der ligger til grund for udarbejdelsen af undersøgelsen. Formålet med kapitlet er at give læseren indsigt i de overvejelser og beslutninger, der har præget undersøgelsesprocessen for dermed at sikre gennemsigtighed og gøre det muligt at vurdere undersøgelsens validitet og pålidelighed. Kapitlet indledes med en beskrivelse af *Undersøgelsesdesignet*, hvor strukturen for undersøgelsen præsenteres. Herefter følger en gennemgang af *Datagrundlaget*, hvor både undersøgelsens teoretiske og empiriske data introduceres. Dette inkluderer en redegørelse af, hvordan disse data er blevet udvalgt, indsamlet og bearbejdet. Afslutningsvis beskrives *Dataanalyse,* hvor det redegøres for, hvordan dataene er blevet analyseret og efterfølgende anvendt i undersøgelsen.

2.1 Undersøgelsesdesign

I afsnittet vil undersøgelsesdesignet, som anvendes i rapporten, blive redegjort for. Det er inddelt i underelementerne: (1) *Contextual Design* (2) *Prototypeudvikling og Test*. I CD vil der blive redegjort for CD som designproces, og efterfølgende der blive argumenteret for, hvorfor denne fremgangsmetode er valgt, og hvordan de forskellige processer mere specifikt er anvendt i denne undersøgelse. *Prototypeudvikling og Test t*ager udgangspunkt i CD-processen, men med et skærpet fokus på strukturering og afprøvningen af prototypen, der blev udviklet baseret på resultaterne fra CD-processen.

Contextual Design

CD er en brugerdrevet designproces, der indebærer veldefinerede trin, som forfattere/forskere kan følge til at udvikle og designe nye systemløsninger på systematisk vis. Processen omfatter fastlagte metoder til at indsamle omfattende og detaljerede data om brugere, deres arbejdsmiljø, arbejdsopgaver og arbejdspraksisser. Ydermere inkluderer processen modeller og teknikker til at behandle, analysere og præsentere den indsamlede brugerdata. Den dybe indsigt i brugerens arbejdsgang og den efterfølgende strukturerede behandling af disse data anvendes som grundlag for idégenerering til systemløsninger og designvalg. I nedenstående *Tabel 2* er CD-processen overskueliggjort ved at præsentere processen opdelt i *Hovedfaser* og *Underfaser* med en tilhørende kort beskrivelse af de enkelte trin. (Holtzblatt & R. Beyer, 2005; Holtzblatt & Beyer, 2017).

Hoved-faser	Underfaser	Beskrivelse af underfaser
Requirements & Solution	Contextual Inquiry	Dataindsamling i form feltinterview.
	Interpretation Session	Processen hvorved centrale problemstillinger bliver identificeret og fortolket pba. indsamlede data fra interviews.
	Work Models and Affinity Diagramming	Konsolidering af den indsamlede data igennem modeller og diagrammer for fange og illustrerer brugerens arbejdsproces.
	Visioning	Redesign af brugerens arbejdsproces/workflow ved at fremføre nye teknologiske idéer.
Define & Validate Concepts	Storyboarding	Definering af den adfærd, struktur og detaljerede funktioner til det nye foreslåede system/workflow ved at skabe sammenhæng mellem arbejdsopgaver.
	User Environment Design	Illustration/beskrivelse af det nye systems/workflows struktur for at imødekomme et naturligt flow igennem arbejdsprocessen. Samtidig en beskrivelse/illustration af hvordan systemet understøtter brugerens opgaver, hvilke funktioner der er tilgængelig i den del af opgaven, og hvordan det kommer til og fra andre dele af systemet.
	Paper Mockup Interviews	Udarbejdelse af Mockup for interface af det nye workflow. Interaktionsmuligheder integreres i Mockup for at opnå testning med brugeren.
	Interaction & Visual Design	Test af systemudvikling/workflow med brugeren for at indhente feedback og opgøre tilfredsstillelsesgraden.

Tabel 2 - Contextual Design, Procesbeskrivelse – Inspiration fra (Holtzblatt & R. Beyer, 2005; Martiny, et al., 2024)

Ud over at CD generelt er en anerkendt og hyppig anvendt metode, blev den valgt i denne undersøgelse på grund af dens relevans ift. at besvare problemformuleringen bedst muligt. Det fremgår af rapportens indledning og problemformulering, at undersøgelsen behandler emnerne AI, SWT, LD og BIM. Det har generelt været hensigten at undersøge, samle og kombinere disse teknologiers tekniske egenskaber og potentialer i udviklingen af et nyt system. En væsentlig del af undersøgelsen fokuserer konkret på selve systemudviklingen, hvilket understøtter behovet for at vælge en metode som CD, der sikrer, at både undersøgelsen og udviklingen følger en tilgang, der kan forsvares teoretisk. Det fremgår af *Tabel 2* og den tilhørende litteratur, at CD er brugercentreret, hvilket betyder, at det udviklede system indebærer en høj grad af brugerinvolvering og -input. Dette sikres bl.a. igennem feltinterviews, test af systemet og indhentning af feedback fra testpersoner. Problemstillingen i undersøgelsen fokuserede på udviklingen af et nyt system til brugere i Bygge- og Anlægsbranchen, der arbejdet med kontroller af BIM-modeller og har behov for en mere effektiv metode til at søge og finde relevant information i både projektmateriale og BIM-modeller. Ved at anvende CD som undersøgelsesdesign muliggøres det at skabe

et system, der giver værdi for slutbrugeren ved at understøtte og forbedre deres arbejdspraksis. (Holtzblatt & R. Beyer, 2005; Holtzblatt & Beyer, 2017).

Prototypeudvikling & Test

Den sidste hovedfase i CD handlede om at definere det nye system og derefter bekræfte, at systemet ville skabe værdi for brugeren. Undersøgelsen har kronologisk fulgt CDs trin: (1) Storyboards (2) User Environment Design (UED) (3) Mockup (4) Interaction and Visual Design. Dette er gjort for at sikre, at der blev skabt et sammenhængende materiale, som på en overskuelig måde visualiserede og forklarede systemets tiltænkte funktioner samt konteksten af arbejdsopgaver, det skulle understøtte. De forskellige underfaser er alle sammen er nøje fulgt med henblik på at skabe et produkt, som brugeren kunne teste og derved give feedback ud fra dette. I den afsluttende del af CD, Mockup og Interaction and Visual Design, var der i metoden et stort fokus på at udarbejde materiale, hvor brugeren fik mulighed for fysisk at interagere med systemets designelementer. Derfor blev det i undersøgelsen forsøgt at opnå en mere detaljeret grad af interaktivitet for herved at styrke testen og feedbacken af systemet. Grundet det store fokus på interaktivitet, er der ud over CD, anvendt Interaction Design. CD og Interaction Design blev anvendt som iterative processer for at sikre, at systemer løbende blev tilpasset til brugernes behov og krav. Undersøgelsen er derfor testet med brugerne, hvor resultatet af den udførte test og feedback blev brugt som grundlag for at identificere problematikker og systemfejl. (Carroll, 2010; Holtzblatt & Beyer, 2017; Sharp, et al., 2007).

For at forstå konteksten hvori brugerne anvender systemet, blev Storyboards udarbejdet. Storyboards illustrer, hvordan brugerne kan udføre deres arbejdsopgave i det nye system, og hvilke systemfunktioner der understøtter brugeren i processen. Brugerens handlingsmønster i det nye system blev illustreret vha. Storyboards, der samtidig fremviser, hvordan deres handlinger udfolder sig i systemets funktioner. For at visualisere systemets arkitektur og skabe forståelse for hvordan brugergrænsefladen integreres med de bagvedliggende funktioner, blev der anvendt UED. UED blev brugt til at illustrere, hvordan systemets struktur understøttede brugerens arbejdsopgaver. I undersøgelsen blev UED udviklet med udgangspunkt i et funktionelt system, hvor der er sammenhæng mellem databaser (DB), Machine Learning-modeller (ML) og de bagvedliggende funktioner. (Holtzblatt & Beyer, 2017).

I undersøgelsen er der udarbejdet to prototyper, som består af hhv. én Mockup og ét teknisk system. Prototyperne blev udviklet for at give brugerne mulighed for at interagere med systemet og dets funktioner samt observere, hvordan deres handlinger afspejles i systemets respons. Det gjorde det muligt at identificere potentielle faldgruber, såsom manglende tekniske funktioner, ønskede forbedringer og yderligere nødvendige informationer.

Mockup'en blev udarbejdet i PowerPoint (PP), og den teknisk prototype fungerede som grundlag for udviklingen af et fremtidigt system. Formålet med Mockup'en var at demonstrere systemets funktioner og responser i et interface integreret i et BIM-software. Systemets funktioner blev simuleret vha. links, der illustrerer de tiltænkte funktioner. Gennem en grafisk fremstilling af interfacet får brugeren indsigt i, hvordan et virkeligt system vil reagere på input og andre former for interaktioner ved at navigere mellem de involverede slides i PP.

Den tekniske prototype blev udviklet med det formål at demonstrere systemets bagvedliggende funktioner i et kontrolleret miljø, hvor et samlet teknisk system ikke var nødvendigt Prototypen bestod overordnet af Python-scripts, Dynamo-grafer og filer, hvor både strukturerede og ustrukturerede data blev inkluderet. Ved at sammensætte elementer af systemets primære funktioner blev det illustreret for testpersoner, hvordan systemets funktionalitet blev udviklet og implementeret. Der blev anvendt prædefinerede scripts og funktioner fra Huggingface (HF) og Unsloth (Huggingface, 2024; Unsloth, 2024). Disse prædefinerede scripts krævede tilpasning til prototypeudviklingen, hvilket blev udført ud fra en kombination af manuel justering af værdier og troubleshooting vha. ChatGPT (OpenAI, 2024). Tilpasningen af scriptene foregik ved opsætning heraf i Microsoft Visual Code Studio og Google Colab. Når scriptene blev eksekveret uden fejlmeddelelser ved debugging, kunne disse implementeres i Dynamo Grafen. Dynamografen er primært baseret på anvendelsen af Out of the Box (OOTB) nodes kombineret med Python-scripts. I *Afsnit 7.2* beskrives fremgangsmåden for sammenkoblingen mellem software og Subprocess-biblioteket. Dynamo blev brugt for at muliggøre et automatiseret interface og fremvise systemarkitekturen.

Test af prototypen blev baseret på Interaction Design testing. Da både CD og Interaction Design er iterative processer, vil der i et virkeligt scenarie skulle udføres løbende test af prototypen for, at brugernes feedback kunne implementeres. Test-typen der anvendes, afhænger af, hvor i processen for systemudviklingen designteamet befinder sig. De test-typer der kan anvendes i Interaction Design er: (1) Usability Testing (2) Field Testing. I undersøgelsen er der blevet gjort brug af Usability Testing. Dette blev gjort fordi, at denne testform var baseret på test, der udføres undfer kontrollerede forhold, og hvor der bliver afprøvet specifikke funktioner i systemet. Formålet med anvendelsen af Usability Testing er at observere de typiske arbejdsopgaver, og hvordan brugeren interagerede med systemet. Her definerer designteamet, hvilke funktioner de vil have feedback på og kontrollerer dermed brugerens miljø. Den afsluttende del af Interaction Design testing fokusererede primært på test-typen Field Testing. I modsætning til Usability Testing fokuserer denne testtype på at overlade det fulde system til brugeren for at få en komplet oplevelse af dette. (Sharp, et al., 2007).

Feedbacken på det udviklede system og prototyper blev indsamlet gennem udførelsen af Usability Testing. Dette blev gjort, da prototypernes opbygning primært var baseret på enkelte funktionaliteter. Dertil blev prototypen udviklet i et lokalt miljø, som eliminerer muligheden for at brugeren kan opleve det samlede system. Undersøgelsen blev udført over en begrænset periode, med begrænsede ressourcer, hvilket betød, at testene ikke kunne gennemføres som en iterativ proces. Den givne feedback blev derfor vejledende for, hvordan en videreudvikling kunne udføres, men kunne ikke implementeres i et omfang, der afspejlede et retvisende resultat.

Udførelsen af Usability Testing kan findes i *Kapitel 8*. Testen blev udført ved fremvisning af Mockup'en af interfacet, hvor testpersonerne fik præsenteret de grundlæggende intentioner med funktionerne, og hvordan systemet vil reagere på deres inputs. For at understøtte Mockup'en blev dele af den tekniske prototype præsenteret som en integreret del af interfacet. Testen blev afholdt i et kontrolleret miljø, hvor brugerne fik fremvist systemets funktioner, og feedback blev indsamlet.

2.2 Datagrundlag

Dette afsnit omhandler selve datagrundlaget for undersøgelsen, og hvordan denne data blev indsamlet. Dataene består af teoretisk og empirisk data, og disse er indsamlet ud fra forskellige metoder. Metoderne til indsamling af de forskellige typer data blev nøje udvalgt, og der vil i afsnittet blive argumenteret for, hvorfor den enkelte metode blev valgt. Der vil i afsnittet afslutningsvist blive redegjort for, hvordan data efterfølgende er blevet analyseret og behandlet i undersøgelsen.

Teoretisk data

Den teoretiske data, der blev anvendt i denne rapport, præsenteres i *Kapitel 3 og 4* og består af sekundære kvalitative data. Selvom begge kapitler benytter sekundære kvalitative data, adskiller datagrundlaget sig mellem de to kapitler. Derfor er dataindsamlingen, der danner grundlag for kapitlerne, udført vha. to forskellige litteratursøgningsmetoder.

Kapitel 3 havde til formål at identificere og præsentere undersøgelsens centrale begreber for at give læseren en indledende forståelse. Dette sikrede, at de anvendte teknologier, principper og den relevante viden blev forklaret og gjort tilgængelig for læseren. Kilderne består af anerkendt faglitteratur, primært bøger, rapporter og artikler, og er fremfundet ved hurtige og effektive frisøgninger på internettet.

Kapitel 4 havde til formål at kontekstualisere undersøgelsen ift. den eksisterende litteratur på området. Det består af en mere omfattende litteratursøgning end litteratur henvendt til *Kapitel 3*. Dette indebærer en systematisk gennemgang af den tilgængelige viden inden for feltet for at sikre, at undersøgelsen baseres på et dokumenteret vidensgrundlag. Det medfører også, at der kan gennemføres velbegrundede

og relevante analyser, argumentationer og designprocesser. Kapitlet havde yderligere til hensigt at gøre det muligt at positionere undersøgelsen i relation til tidligere studier. Dette medførte samtidig, at system- og prototypeudviklingen, udført i rapporten, byggede på relevant viden fra allerede udførte studier. (Rienecker & Jørgensen , 2022).

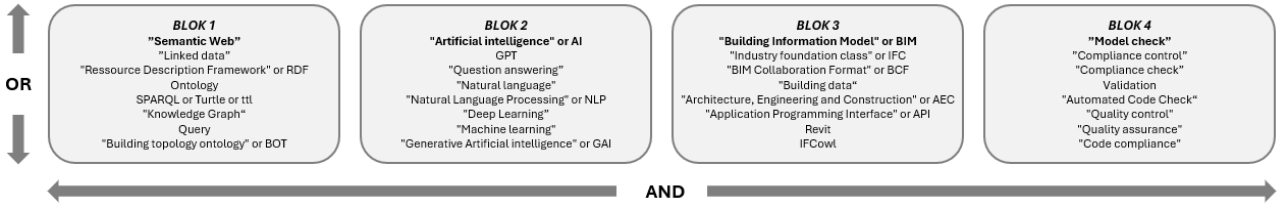
Indsamlingen af dataene blev foretaget med inspiration fra den systematiske litteratursøgning beskrevet i bøgerne af *Lund, et al (2014*) og *Rosenberg, et al (2016*). En sådan struktureret tilgang bidrog til øgede chancer for at fremfinde relevant litteratur. Derudover øgedes muligheden for at foretage robuste, veldokumenterede, transparente og troværdige konklusioner bestående af *non-biased* resultater (Lund, et al., 2014; Rosenberg, et al., 2016). Den systematiske litteratursøgning udført i denne rapport er opsummeret i *Tabel 3*.

Systematisk litteratursøgning

Trin	Forklaring
1. Baseret på allerede udarbejdet problemformuleringen defineres aspekter for litteratursøgningen	De vigtigste aspekter defineres pba. problemformuleringen. Aspekter er underemner, som sammen udgør forskellige delelementer af undersøgelsens emne.
2. Udvælgelse af relevante søgeord for hvert aspekt	Når aspekterne er identificeret, udvælges relevante søgeord, der specifikt relaterer sig til de definerede aspekter. Søgeordene kan bl.a. bestå af overordnede samt underordnede begreber, synonymer og akronymer. Disse kan findes ved brainstorm, titler og abstracts fra allerede fundet litteratur e.l.
4. Opbygning og efterfølgende udførelse af litteratursøgning	Indledningsvis udvælges én eller flere DB'er, der skal danne grundlag for litteratursøgningen. Bloksøgning oprettes og søgestrenge vha. booleske operationer (AND, OR, NOT) og frasesøgninger udføres. Samtidig anvendes inklusions- og eksklusionskriterier.
5. Udvælgelse og screening	Dubletter fjernes samt en sortering på pba. kriterier foregår. Der sorteres ud fra: Titel, abstract, første- og anden prioritet

Tabel 3 – Systematisk litteratursøgning – Proces for udvælgelse inspireret af (Martiny, et al., 2024)

Den systematiske litteratursøgning tog udgangspunkt i en grundig indledende frisøgning, da denne skulle bidrage med inspiration og viden til at fremfinde aspekter og tilhørende søgeord, som skulle agere grundlag for litteratursøgningen. Med udgangspunkt i problemformuleringen resulterede det i følgende fire aspekter: (1) SWT (2) AI (3) Building information model or BIM (4) Model check. Baseret på forudgående undersøgelser blev der fremfundet syv passende søgeord til hvert aspekt. Disse er illustreret på nedenstående *Figur 1*.



Figur 1 - Udvælgelse af søgeord til systematisk litteratursøgning

Herefter kunne den systematiske litteratursøgning igangsættes, hvilket foregik ud fra en bloksøgning. I undersøgelsen blev der, grundet tidsmæssig begrænsning, kun foretaget litteratursøgning i én DB, Scopus. Dette kunne i sig selv udgøre en begrænsning, men det forventedes, at den systematiske fremgangsmåde ville sikre dybden og troværdigheden af de fremfundne kilder (Lund, et al., 2014; Rosenberg, et al., 2016). Bloksøgningen omfattede i alt 14 søgestrenge (*Bilag 1*), hvor de forskellige blokke blev søgt individuelt og i kombinationer vha. booelske operationer (*AND, OR* og *NOT*) og frasesøgninger. Når søgestrengene blev søgt i Scopus, blev følgende inklusions- og eksklusionskriterier anvendt:

- År / Year: 2000-2025.
- Emne / Subject area: Computer Science & Engineering.
- Sprog / Language: English
- Land / Country (territory): United States, Germany, Sweden, Norway, Finland, Netherlands, Belgium & Denmark.
- Dokumenttype / Document type: Limited to conference paper & article

Disse blev først og fremmest valgt for at indskrænke resultaterne, som ellers ville have været ekstremt høje og uden relevans for undersøgelsen, hvis ikke kriterierne var inkluderet. Søgningen blev begrænset til kun at inkludere dokumenter fra år 2000 til 2025. Dette kriterie blev valgt fordi, at flere af aspekterne, som bloksøgningen bygger på, havde været aktuelle og anvendt i lang tid, også før år 2000. Baseret på indledende undersøgelser blev det identificeret, at de involverede aspekter og teknologier i undersøgelsen for alvor blev anerkendt og begyndte at tage fart omkring år 2000. (Coursera , 2024; Petkova & Kiryakov, 2021).

Da kriterierne og alle 14 søgestrenge blev udført, resulterede det i 637.240 artikler, hvilket vurderes at være for mange. Det var afgørende, at de artikler der blev identificeret gennem den systematiske litteratursøgning bedst muligt, afspejlede problemformuleringen, hvilket krævede, at søgeordene fra blokkene indgik på tværs i søgningen. Således indgår delementer af AI, SWT, LD og BIM på samme tid i de fremsøgte artikler. Der blev derfor lavet en afgrænsning, hvor en kombination af enten tre eller alle fire blokke, blev medtaget. Dette udgjorde fem søgestrenge, som resulterede i 1.051 artikler. I *Tabel 4* vises, hvordan én søgestreng med tre blokke samt den ene søgestreng med fire blokke blev kombineret.

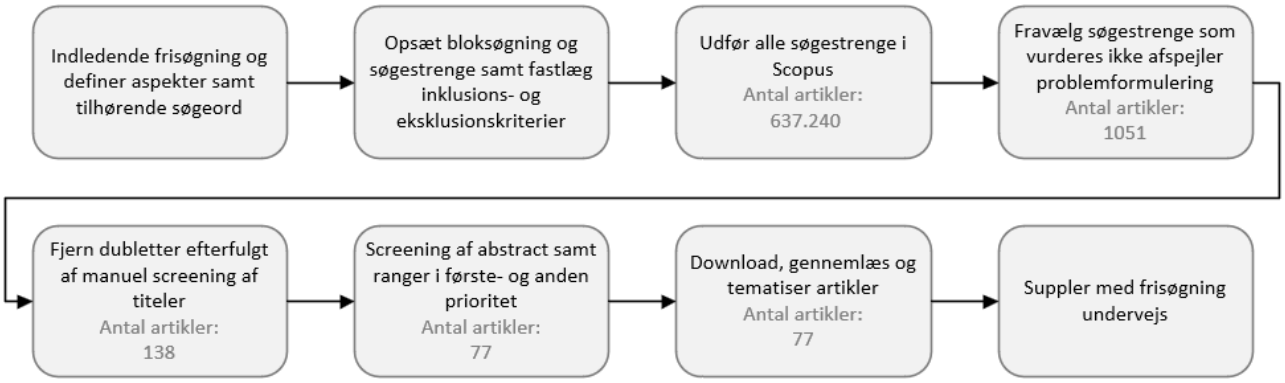
# Søge nr.	Søgestreng	Antal
Blok 1 + Blok 2 + Blok 3	("Semantic Web" or "Linked data" or "Ressource Description Framework" or RDF or Ontology or SPARQL or Turtle or ttl or "Knowledge Graph" or Query or "Building topology ontology" or BOT) and ("Artificial intelligence" or AI or GPT or "Question answering" or "Natural language" or "Natural Language Processing" or NLP or "Deep Learning" or "Machine learning" or "Generative Artificial intelligence" or GAI) and ("Building Information Model" or BIM or "Industry foundation class" or IFC or "BIM Collaboration Format" or BCF or "Building data" or "Architecture, Engineering and Construction" or AEC or "Application Programming Interface" or API or Revit or ifcOWL)	216
[…]	[…]	[…]
Blok 1 + Blok 2 + Blok 3 + Blok 4	("Semantic Web" or "Linked data" or "Ressource Description Framework" or RDF or Ontology or SPARQL or Turtle or ttl or "Knowledge Graph" or Query or "Building topology ontology" or BOT) and ("Artificial intelligence" or AI or GPT or "Question answering" or "Natural language" or "Natural Language Processing" or NLP or "Deep Learning" or "Machine learning" or "Generative Artificial intelligence" or GAI) and ("Building Information Model" or BIM or "Industry foundation class" or IFC or "BIM Collaboration Format" or BCF or "Building data" or "Architecture, Engineering and Construction" or AEC or "Application Programming Interface" or API or Revit or ifcOWL) and ("Model check" or "Compliance control" or "Compliance check" or Validation or "Automated Code Check" or "Quality control" or "Quality assurance" or "Code compliance")	13
I alt:		1051

Tabel 4 – Fremvisning af søgestrenge og kombination af blokke

Resultaterne af disse fem søgestrenge blev eksporteret fra *Scopus* til CSV med følgende metadata: (1) Titel (2) Cited by (3) Abstract (4) Link. Disse data organiseres efterfølgende i et Excel-ark (*Bilag 1*).

Dernæst foregik sidste trin af den systematiske litteratursøgning, hvor den fremsøgte litteratur blev udvalgt og screenet. Først blev dubletter fjernet, hvorefter en manuel screening fandt sted pba. titel. Dette resulterede i en fravælgelse af 913 dokumenter, hvilket gjorde, at der var 138 tilbageværende dokumenter. Dernæst blev der sorteret ud fra abstract, hvilket gjorde, at dokumenterne blev indskrænket til 77 dokumenter. Afslutningsvist skete yderligere en sortering, hvor de tilbageværende dokumenter blev rangeret efter enten første- eller anden prioritet. Her blev der lavet en prioritering, grundet tidsmæssige begrænsninger, hvor 53 dokumenter blev førsteprioritereret og 24 dokumenter anden prioriteret. Fuldteksten på de 77 dokumenter blev downloadet, gennemlæst, og vigtige pointer og citater blev markeret undervejs. Disse markeringer blev undervejs brugt til at udføre en kodning af den kvalitative data, som gennemgås i *Afsnit 2.3* om *Analyse af teoretisk data*. Forinden den systematiske litteratursøgning blev udført, blev en fremgangsmåde herfor udarbejdet, fordi det minimerer bias (Rosenberg, et al., 2016). Udarbejdelsen af en fremgangsmåde for litteratursøgningen sikrede, at undersøgelsen forholdt sig til den

tilgængelige viden inden for emnet. Det viste transparens, at artiklerne ikke blev tilfældigt udvalgt, og at undersøgelsen forholdt sig så objektivt som muligt, både undervejs men også forinden undersøgelsen blev igangsat (Rosenberg, et al., 2016). Denne fremgangsmåde er opsummeret i nedenstående *Figur 2*, hvor det er vist, hvordan antallet af artikler indsnævres løbende i processen.



Figur 2 – Fremgangsmåde for systematisk litteratursøgning

Efter den systematiske litteratursøgning blev der undervejs udført diverse frisøgninger i *Scopus*, når ny viden opstod i processen. Her blev der søgt på nye søgeord baseret på den nye viden, og funktionerne *Cited by* og *Related Documents*, blev anvendt. I designprocessen viste det sig f.eks., at DB-typer fik en afgørende rolle for systemudviklingen selvom, at søgeordene ikke var inkluderet i den systematiske litteratursøgning. For yderligere information om hvordan den systematiske litteratursøgning blev udført, henvises der til *Bilag 1*.

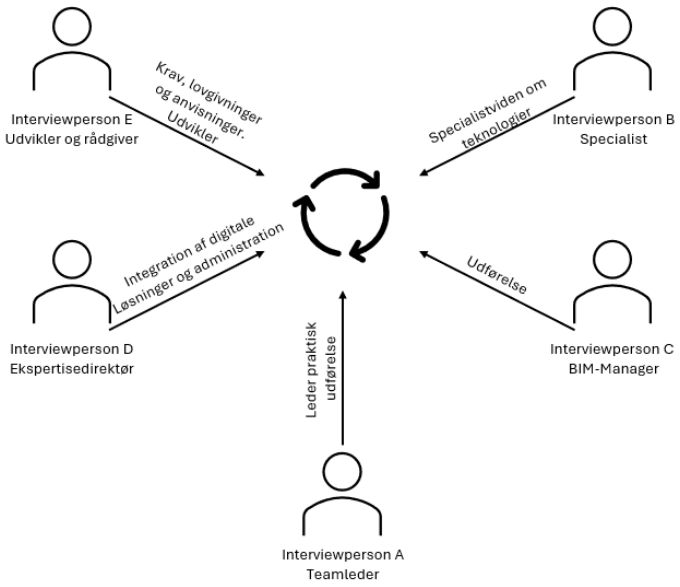
Empirisk data

Undersøgelses empiriske data bestod af kvalitative primære data indsamlet gennem interviews med relevante fagpersoner fra byggebranchen. Fokus var på IKT, innovation, udvikling og projekteringsprocesser, og interviewene relaterede sig til CD-trinnet Contextual Inquiry, som vist i *Tabel 1*. Ved udvælgelsen blev der lagt vægt på personer med procesforståelse inden for kvalitetssikring og kontrol samt erfaring med udvikling og implementering af nye systemer og standarder. Interviewpersonernes ekspertise bidrog til en holistisk forståelse af arbejdsprocesserne ifm. kontrol og kvalitetssikring, som understøtter CD-processen. Formålet var at opnå en forståelse for den nuværende udførelse af arbejdsprocessen, og hvordan den kunne rationaliseres med ny teknologi og standarder. Udvælgelsen af fagpersoner blev baseret på kausale sammenhænge og fokuserede på, at personer, der udfører denne proces, har relevante holdninger eller besidder kendskab til den adfærd og de teknologier, der blev undersøgt. (Aarhus Universitet, 2019).

Den oprindelig tilgang til Contextual Inquiry, som indebærer feltinterviews med fokus på observation og registrering af interviewpersonernes arbejdsproces, blev ikke anset som relevant i denne undersøgelse. Dette skyldes, at undersøgelsens

systemudvikling baseres på en helhedsændring af det samlede workflow, hvor flere typer af brugere og arbejdsopgaver er inkluderet. Formålet har derfor været at afdække de brugerbehov, der kan implementeres i systemet for at understøtte de arbejdsprocesser, der blev præsenteret af brugerne i interviewene.

En samlet forståelse af arbejdsprocessen samt de undersøgelsens teknologier, blev opnået gennem semistrukturerede interviews i forbindelse med indsamlingen af empirisk data. Semistrukturerede interviews blev valgt pga. fagpersonernes ekspertise inden for teknologier og standarder til understøttelse af kvalitetssikring og kontroller. Interviewformen gav mulighed for en interviewstruktur, hvor højere teknisk viden ikke var nødvendig. Fagpersonens viden bidrager til at hæve det taksonomiske niveau, udvide forståelsen af emnet og skabe paralleller til øvrige relaterede emner. Samtidig tillod anvendelsen af semistrukturerede interviews, at perspektiver af workflowet, der ikke var inkluderet i interviewguiden, kunne blive præsenteret af interviewpersonerne og dermed opnå elementer af feltinterviews. (Aarhus Universitet, 2019). Nedenstående *Figur 3* viser de anvendte interviewpersoner og deres bidrag af viden til undersøgelsen samt roller.



Figur 3 - Oversigt over interviewpersoner

Anvendelsen af semistrukturerede interviews i undersøgelsen gjorde det muligt at indfange interviewpersonernes personlige holdninger til branchens udførelse af kvalitetssikring og kontroller. Med variation i interviewpersonernes erfaringer, jobtitler og vidensområder blev der opnået en holistisk forståelse af branchens praktiske metoder, strategiske overvejelser og fremtidige teknologier til understøttelse af området. Grundet variationer i videns- og arbejdsområdet samt ekspertisen blev der udarbejdet interviewguides til hvert interview. Dette dannede grundlag for valget af semistrukturerede interviews som den mest relevante interviewform til at understøtte interviewpersonernes besvarelser. Valget blev truffet, da denne form gav mulighed

for, at personernes synspunkter og holdninger kunne udfoldes, selvom de mangler erfaring inden for deres fagområde relateret undersøgelsens emner. I og med at der ikke blev anvendt en fast struktur i interviewet, kunne interviewpersonerne referere til relevante områder og emner, som ikke var afdækket i interviewguiden. Interviewformen bidrog dog stadig til at skabe en overordnet retning og ramme, hvilket sikrede, at den indsamlede data var relevant for undersøgelsen. Grundet ønsket om supplerende informationer og perspektiver blev interviewguiden udformet med åbne spørgsmål, som lagde op til uddybende svar fra interviewpersonerne. Anvendelsen af lukkede spørgsmål er kun anvendt til bekræftelse af udsagn fra interviewpersonerne. (Justesen & Mik-Meyer, 2010; Aarhus Universitet, 2019).

2.3 Dataanalyse

Dette afsnit har til formål at gennemgå, hvordan hhv. den teoretiske og empiriske data er blevet analyseret og bearbejdet i undersøgelsen. Afsnittet består af to underafsnit, som opdeler analysen af det teoriske og empiriske data.

Analyse af teoretisk data

Den systematiske litteratursøgning blev afsluttet med en grundig gennemlæsning af de fremfundne artikler, hvor vigtige pointer og citater undervejs blev markeret med forskellige farver. Til dette blev der draget inspiration fra metoden *Kodning af kvalitative data* (Aarhus Universitet , 2024; Miles, et al., 2018). Disse farvemarkeringer blev brugt til at udføre en løbende kodning af datene fra den systematiske litteratursøgning, som indebar, at dataene blev opdelt i fem relevante temaer: (1) Intro/Blandet (2) RDF (3) SW/LD (4) Industry Foundation Classes/BIM (IFC/BIM) (5) AI. Denne kodning er udført for at overskueliggøre datasættet, bestående af 77 artikler samt artiklerne fra frisøgningen, som senere skulle konsolideres og sammensættes i *Kapitel 4.* Når en sådan kodning blev udført, var det vigtigt at være opmærksom på objektiviteten i selektionen af data og kodningen heraf for at undgå bias og misvisende konklusioner. Kodningen er samtidig en proces, hvor data fragmenteres, oversættes og generaliseres, hvilket kunne betyde, at data potentielt gik tabt eller blev forvrænget ift. den oprindelige kilde, hvilket er vigtigt at være opmærksom på undervejs. (Aarhus Universitet , 2024; Miles, et al., 2018). Denne kodning af dataene gjorde det muligt først og fremmest at skabe et overblik over datasættet. Derudover blev det muligt at skabe et mønster og en relation mellem dataene og efterfølgende samle datene i én samlet kontekst med relevans for undersøgelsen.

Empirisk data

Undersøgelsen empiriske grundlag består af primær kvalitative data. De kvalitative data er beskrevet i *Afsnit 2.2*. Dataene bestod, som beskrevet tidligere, af en række semistrukturerede interviews. Indsamlingen af interviewene blev foretaget med interviewguides *(Bilag 10-14)*. Interviewene blev under udførelsen optaget med

formålet om efterfølgende bearbejdning heraf. Optagelserne blev meningskondenseret, hvor de forberedte spørgsmål blev besvaret ud fra det kondenserede interview. Perspektiver, som ikke var omfattet af de planlagte spørgsmål, blev også inddraget. Anvendelsen af meningskondenseringer tillod, at de relevante emner kunne beskrives kortfattet. Meningskondenseringerne kan findes i *Kapitel 5* og anvendes i analysen i *Kapitel 6*. De sammenfattede kondenseringer af de individuelle interviewpersoner blev efterfølgende anvendt i analysen i *Kapitel 6* og i udviklingen af både systemet og prototyperne. I analysen fremhæves branchens nuværende arbejdsproces, hvilke muligheder der ses for fremtiden, brugernes behov og holdninger.

Baseret på den indhentede data blev en komparativ analyse udført for at skabe et overblik og identificere de systemkrav og brugerbehov, som det nye system skulle imødekomme. I den komparative analyse blev meningskondenseringerne sammenholdt med den teoretiske data indsamlet i *Kapitel 4*. Interviewpersonernes holdninger og erfaringer blev koblet sammen med artiklerne fra den systematiske litteratursøgning, hvor ét af inklusionskriterierne var baseret på landet, hvor artiklerne var publiceret i. Dette gjorde, at undersøgelsen og dermed analysen ikke kun forholdt sig til danske forhold, men kan sammenlignes med lande, som Danmark generelt sammenlignes med. Således blev både nationale og internationale principper inkluderet i analysen. Den komparative analyses indhold af teoretisk og empirisk data blev anvendt til at udarbejde en liste over identificerede brugerbehov og systemkrav, som blev præsenteret i *Afsnit 6.3*. Systemkravene og brugerbehovene blev anvendt som grundlag for den overordnede vision for det nye system, hvilket slutligt dannede rammerne for udarbejdelsen af prototyperne *Kapitel 7*.

3 BEGREBER

Dette kapitel præsenterer og definerer de centrale begreber, der er relevante for undersøgelsen og har til formål at etablere en fælles forståelsesramme. Introduktionen tager udgangspunkt i tre overordnede emner: (1) Artifcial Intelligence (2) Semantic Web (SW) (3) Databaser.

Under emnet AI vil kapitlet give en dybere indsigt i centrale metoder og teknologier indenfor KI, herunder ML, Deep Learning (DL) samt de overordnede tilgange: (1) Generative (2) Discriminative AI. I forlængelse heraf præsenteres LLM, som udgør en vigtig underkategori af GI og spiller en central rolle i moderne AI-udvikling.

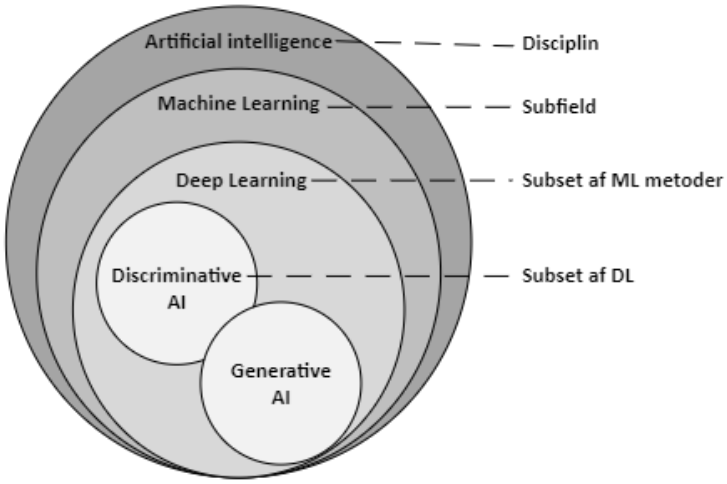
Inden for SW fokuseres der på teknologier og standarder, som muliggør repræsentation, strukturering og tilføjer semantiske relationer til data. Der redegøres for nøglebegreber som LD, Ressource Description Framework (RDF) og tilhørende teknologier såsom RDF Vocabulary Definition Language, SPARQL Protocol and RDF Query Language (SPARQL) og Ontologier. Desuden forklares forskellen mellem Abox og Tbox, samt hvordan regel- og valideringssprog, som Rule Markup Language, Semantic Web Rule Language (SWRL), triples og SHACL, anvendes til queries, specificering og validering af data.

Afslutningsvis behandles DB'er, hvor fokus er på grafdatabaser (GDB) og vektordatabaser (VDB). Disse repræsenterer innovative tilgange til lagring og organisering af komplekse datarelationer, hvilket spiller en central rolle i denne undersøgelse. Gennem kapitlet vil der blive etableret en grundlæggende forståelse for disse begreber og deres indbyrdes sammenhænge, hvilket danner fundament for de efterfølgende kapitler.

3.1 Artificial Intelligence

Artificial Intelligence, også kendt som Kunstig Intelligens på dansk, består af en række teknologier, der gør det muligt for computere at udføre avancerede opgaver. AI er et videnskabsområde, der omhandler at udvikle teknologier med funktioner, der giver maskiner mulighed for at ræsonnere, lære og handle på en måde, som normalvis ville kræve menneskelig intelligens. Disse funktioner omfatter bl.a. evnen til at analysere data, visualisere, forstå og give anbefalinger.
Fundamentet for AI-systemer er de data, som de bliver eksponeret for. Dette skyldes, at AI-systemer lærer og forbedres gennem fodring af store datamængder, hvor de genkender mønstre og sammenhænge. Det sker igennem konstruerede algoritmer, der behandler de store mængder data. Algoritmerne agerer som et sæt regler, som gør det muligt for AI-systemer at tilegne sig ny viden, hvilket medfører, at AI bliver bedre til at tilpasse sig nye situationer. (Google Cloud, 2024).

Som det fremgår af nedenstående *Figur 4*, betragtes AI som en overordnet disciplin eller kategori, hvor der herunder findes en række subsets.



Figur 4 - AI og dets underkategorier - Inspiration hentet fra (Mckinsey & Company, 2024)

Machine Learning er et subfield til AI og omfatter en række metoder, som differentierer sig ved deres forskellige formål og teknikker, som bruges til at udvikle AI. Et subset til ML-metoderne er f.eks. DL), hvor Discriminative AI og GI er underkategorier til DL. (Google Cloud, 2024; Mckinsey & Company, 2024; Mitchell, 2019). Der vil være en mere fyldestgørende beskrivelse af figurens begreber i de næste underafsnit.

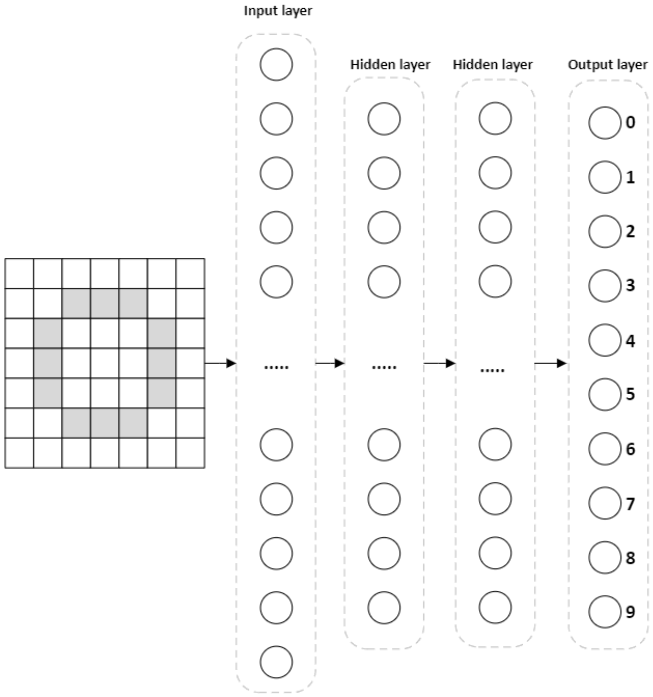
Machine Learning

ML er en type AI, der er kendetegnet ved at kunne tilpasse forskellige former for datainputs. Det kan være et system eller et program, der træner en AI-model ud fra datainputs. Disse data kan bl.a. være menneskelige input eller historiske data (Google Cloud, 2024). Ydermere bygger ML-modeller på algoritmer, der trænes på enten *Labeled data* eller *Unlabeled data*, hvilket opdeler ML-modeller i hhv. *Supervised* eller *Unsupervised Learning*. I ML trænes disse algoritmer i sådan en grad, at de kan komme med forudsigelser og kategorisere information baseret på nye data, hvilket gøres pba. den data, som ML-modellen allerede er blevet trænet på. *Supervised-modeller* anvender f.eks. *Labeled data* til at kigge på tidligere eksempler for at forudsige fremtidige værdier. *Supervised-modeller* genererer derfor et output, såsom en forudsigelse, og sammenligner det med de data, modellen er trænet på. *Unsupervised-modeller* anvender derimod *Unlabeled* rådata og undersøger, om de følger et mønster eller kan inddeles i grupper.
Visse ML-algoritmer er designet til at anvende både *Labeled* og *Unlabeled data* til at træne sig selv og ydermere til at udføre opgaver ved at behandle data for herefter at identificere mønstre og komme med forudsigelser. ML med disse algoritmer kan muliggøres gennem metoder, som karakteriseres som DL. (Mckinsey & Company, 2024; Google Cloud, 2024).

Deep Learning

DL-algoritmer kan opdage mønstre og, på baggrund heraf, forudsige resultater og lave anbefalinger, som ikke sker ved direkte programmering men ved analyse af data. Nogle algoritmer er udarbejdet med sådan en kompleksitet, at de kan udvikle sig over tid ved at tilpasse sig de data, som de løbende bliver eksporteret for og i takt med dette øge deres nøjagtighed. (Mckinsey & Company, 2024; Google Cloud, 2024).

DL og dens algoritmer anvender kunstige neurale netværk, som er direkte inspireret af, hvordan neuroner i menneskehjernen interagerer for at behandle og lære af data. Dette er også årsagen til, at DL også omtales som *Deep Neural Networks*. Nedenstående *Figur 5* illustrer princippet bag et kunstigt neuralt netværk.



Figur 5 - Princip bag Deep Neural Network - Inspiration hentet fra (Mitchell, 2019)

Figuren præsenterer et neuralt netværk, som i sin enkelhed er designet til at genkende tal. Der ses fire lag, de såkaldte *layers*, som hver indeholder cirkler/enheder, der repræsenterer neuroner. Inputbilledet, der visualiserer tallet 0, består af en række pixels, der hver især repræsenteres af enheder i *Input Layer*. Ydermere består netværket af to *Hidden Layers*, som indeholder skjulte enheder og et *Output Layer*, med ti output enheder svarende til de tal-udfald, der kan genkendes på input-billedet. Pilene angiver, at hver inputenhed har en forbindelse til hver *Hidden* enhed, og at hver *Hidden* enhed har en forbindelse til hver outputenhed. I Neural Network-litteratur betyder Hidden enheder/neuraler non-output enheder. Sammenlignet med menneskehjernen vil nogle enheder have direkte kontrol over outputet, såsom at

bevæge en muskel, men de fleste enheder kommunikerer hovedsageligt med andre enheder uden at have et output. (Mitchell, 2019).

Det er antallet af *Hidden layers*, der er med til at beskrive dybden, *the depth,* af Neurale Netværk. Har et Neuralt Netværk mere end et *Hidden Layer*, såsom den vist på *Figur 5*, refereres der til en *Multilayered Neural Network* eller *et Deep Network*. Som illustreret på figuren, skal alle inputdata igennem hvert *layer*. Det er i disse *layers*, hvor data behandles, og hvor resultaterne heraf sendes videre til det næste layer. Det er denne *layer*-struktur, der giver DL-modeller mulighed for at trække data, lære at behandle komplekse mønstre i data og finde forbindelser og mønstre i data. (Mitchell, 2019).

Generative og Discriminative AI

DL-modeller kan hovesagligt opdeles i undergrupperne GI og DI, hvilket betyder, at deres algoritmer bygger på *Neural Network*-princippet og derfor kan behandle både *Labeled* og *Unlabeled data* med *Supervised*-og *UnSupervised*-metoder. Discriminative-modeller benyttes til at klassificere, skelne eller forudsige Labels for data. Disse modeller vil typisk være trænet på et datasæt med *Labeled Data*, og hvor de lærer relationerne mellem dataegenskaber og Labels. En trænet Discriminative-model kan herefter anvendes til at levere et output, hvor den vil forudsige label/kategorien til det nye data fra inputtet. I modsætning til Discriminative-modeller er Generative-modeller designet til at skabe nyt data ud fra dem, som de er trænet på. Generative-modeller lære at kende forskellen mellem forskellige data og deres kategorier, men ydermere lærer disse modeller at forstå og genkende egenskaberne, strukturerne, karakteristikkerne og mønstrene for data i kategorierne. Generative modeller kan dermed skabe ny data, der har karakteristika, som ligner det data, modellen allerede er trænet på og kender. Generative-modeller kan bl.a. være med til at skabe indhold såsom koder, billeder, tekst og meget mere. (Google Cloud, 2024).

Når en generativ model modtager en prompt, vil den pba. statistiske mønstre forsøger at forudsige den ønskede respons. Dette resulterer i generering af nyt indhold baseret på tidligere eksempler og den information, som modellen tidligere er blevet eksponeret for. Der findes forskellige typer af GI-modeller, som primært adskiller sig ved typen af input i prompten, som modellen behandler og responderer på. Typerne kunne f.eks. være Generative Language Model og Generative Image Model, hvor input hhv. består af tekst og billeder, som navnet antyder. Outputtet, og derved indholdet, disse modeller genererer kan være i form af billeder, audio eller tekst. Denne undersøgelser vil primært anvende og referere til Generative Language modeller, herunder LLM'er, som i højere grad er i stand håndtere mere komplekse opgaver, der involverer tekst og sprogbehandling. (Google Cloud, 2024).

Large Language Model

Large Language Models er en type ML-model, der anvender Transformers-arkitekturen til at behandle og analysere input i NL. Modellen opdeler sproget tokens og repræsentere dem som numeriske, hvilket gør det muligt at forstå brugerens input og generere sammenhængende og logiske svar. Denne metode kaldes typisk for *Encoder-decoder arkitektur*. Encoderen bruges til at omsætte NL til numeriske værdier, mens decoderen anvender disse værdier til at generere svar, der tager højde for både semantikken i de resterende ord af inputtet og konteksten fra tidligere forespørgsler (Vaswani, et al., 2017). LLM er bl.a. designet til at forstå og generere NL, vha. flere funktioner, som overordnet kan betegnes som NLP. LLM'er anvender algoritmer til Natural Language Understanding for at analysere og forstå menneskeligt sprog. Dette sker typisk ved at vurdere ordenes semantiske værdi i deres kontekst og sammenhæng. LLM opbygger forståelsen gennem den numeriske placering af ordets vektor ift. de øvrige ord og deres placering i konteksten. Ud fra dette beregner modellen en statistisk sandsynlighed for det næste ord i LLM'ens respons. (Kamath, et al., 2024).

3.2 Semantic Web

Det er standardiseringsorganet, World Wide Web Concortium (W3C), som er med til at udbrede SW og dets principper, praksisser og initiativer. Tim Berners-Lee opfandt WWW, der ligeledes er drivkraften for SW-initiativerne. Berners-Lees SW-initiativer udspringer fra et ønske om at realisere hans oprindelige vision for Webben, hvor selve betydningen af information har en større rolle. (Antoniou & Harmelen, 2004).

Det er standardiseringsorganet, World Wide Web Concortium (W3C), som er med til at udbrede SW og dets principper, praksisser og initiativer. Tim Berners-Lee opfandt WWW, der ligeledes er drivkraften for SW-initiativerne. Berners-Lees SW-initiativer udspringer fra et ønske om at realisere hans oprindelige vision for Webben, hvor selve betydningen af information har en større rolle. (Antoniou & Harmelen, 2004) Indholdet, og dermed information på Webben, er historisk blevet kritiseret for at optræde som et medie til informationsudveksling mellem mennesker frem for maskiner. Det skal forstås sådan, at informationers betydning på websider er blevet lagret og kodet på sådan en måde, at de er let tilgængelige og forståelige for mennesker. Dette har resulteret i et Web, hvor maskiner har svært ved at forstå den underliggende betydning af information. Som konsekvens heraf bliver søgeresultater ofte upræcise og afhængige af det specifikke ordforråd, der benyttes i den specifikke søgning. (Breitman, et al., 2007). Det SW sigter mod er et mål om at skabe mere avancerede videns styringssystemer. Dette gøres ved at understøtte information således, at det bliver struktureret i *konceptuelle rum*, der direkte afspejler deres betydning. På den måde er det muligt at anvende automatiserede værktøjer til at udtrække ny viden og finde evt. uoverensstemmelser. Disse søgninger fungerer som

Query-Answering, hvor den efterspurgte information hentes, udtrækkes og præsenteres på en brugervenlig måde. (Antoniou & Harmelen, 2004).

SW referer ikke kun til en vision og et mål om data og informationer på nettet. Begrebet dækker også over en fælles ramme og struktur for, hvordan der skabes semantik på nettet således, at data kan deles og genbruges på tværs af forskellige applikationer, organisationer og fællesskaber. (W3C, 2024).

Linked Data

Som det det fremgår af overstående beskrivelse af SW, handler det ikke kun om at publicere data på nettet. Det handler om at gøre data og information semantiske, hvilket betyder, at data får kontekst og mening, der kan forstås af maskiner. Ligeledes handler det om at gøre det muligt for maskiner at forstå, hvordan data er relateret til andet data. Dette indebærer at skabe links mellem data, så både maskiner og mennesker kan udforske nettet for data. (Berners-Lee, 2006). LD-begrebet referer i denne sammenhæng til en række af prædefinerede praksisser for udgivelse og sammensætning af struktureret data på nettet. (W3C, 2023). Praksisserne muliggør at anvende Webben til at skabe forbindelser mellem data. Eftersom det er Webben, som anvendes til at skabe relationerne mellem data, er det yderligere muligt at skabe forbindelser mellem data, som kommer fra forskellige kilder. (Berners-Lee, et al., 2009).

På det traditionelle Web er de primære enheder opbygget som HTML-dokumenter, hvor disse dokumenter kan være forbundet via links uden en specifik type relation. Derimod anvender LD-dokumenter data, der er struktureret i det såkaldte RDF-format. LD bruger RDF til at forbinde dokumenter og bygge strukturerede udsagn, som forbinder forskellige ting i verden. Udfaldet heraf er det, der kaldes *Web of Data*, som mere simpelt kan beskrives som et netværk af ting i verdenen, der er beskrevet igennem data på Webben. (Berners-Lee, et al., 2009).

LD-praksisser indeholder overordnet fire regler, som skal følges ved udgivelse af data på Webben. Disse kaldes *The Linked Data Principles* og tilvejebringer en simpel fremgangsmåde til at publicere og forbinde data gennem Webbens infrastruktur. De fire regler er:

1. Uniform Resource Identifiers (URI) skal anvendes til at navngive ting.
2. HyperText Transfer Protocol (HTTP) URI'er skal anvendes, så mennesker kan slå disse navne op.
3. Nyttig information skal tilvejebringes med standarderne, RDF og SPARQL, når nogen slår en URI op.
4. Der skal inkluderes links til URI'er, så mennesker kan opdage nye informationer.
   (Berners-Lee, et al., 2009)

Webbens arkitektur består af en *Web Client*, almindeligvis kendt som en browser, og en *Web Server,* der tilvejebringer data og dokumenter til *Web Client*en, når en forespørgsel sker. *The Web Architecture* består af tre centrale komponenter, som er afgørende for, at *Web Serveren* og *Clienten* kan fungere. (Domingue, et al., 2011). Disse komponenter er:

1. *Worldwide addressing schema*. Dette gør det muligt at give dokumenter/hjemmesider en unik identifikator. Det gøres med Uniform Ressource Locators (URL), URI og Uniform Ressource Names (URN).
2. *A Transport* Layer. Dette fungerer som en protokol til dataoverførelse. HTTP sørger for, at informationer kan sendes frem og tilbage mellem computer og Webben.
3. *Platform-independent interface*. Dette er selve brugerinterfacet, hvor informationer og indhold præsenteres på en hjemmeside. Interfacet udarbejdes med HyperText Markup Language (HTML), så webbrowsere kan læse og dermed vise indholdet.
   (Domingue, et al., 2011).

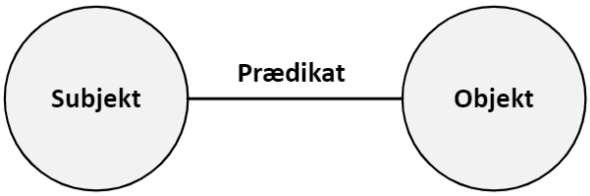
Som nævnt ovenfor omhandler første komponent, at ressourcer skal identificeres, med URL, URI, eller URN. URL'er identificerer og viser, hvor dokumenter og enheder er lokaliseret på Webben. Derimod giver URI'er en mere general og generisk måde at identificere enhver ting i verdenen. (Domingue, et al., 2011; Berners-Lee, et al., 2009). Enheder der bliver identificeret med en URI, og hvor *http://*-skemaet benyttes, er det muligt at følge URI'en gennem HTTP-protokollen. Dermed benyttes HTTP til at hente ressourcer over Webben. Sammenspillet mellem URI'er og HTTP muliggør at finde og hente data på Webben. De suppleres af den førnævnte teknologi, RDF, som er afgørende for *Web of Data*. (Berners-Lee, et al., 2009).

Resource Description Framework

Hvor HTML anvendes til at strukturere og forbinde dokumenter på Webben, benytter RDF en generisk grafbaseret datamodel til at strukturer og forbinde data, der beskriver forskellige ting i verdenen. (Berners-Lee, et al., 2009). RDF er et sprog, der bruges til at beskrive information om ressourcer på webben. Sproget er hovedsageligt designet til at repræsentere metadata om ressourcer på Webben. Ydermere kan sproget også bruges til at beskrive information om andre typer af objekter, der kan identificeres på Webben, selvom disse objekter ikke eksisterer på webben. I noget litteratur er RDF beskrevet som et *Lightweight ontology*-sprog, der er rettet mod at skabe interoperabilitet mellem systemer, som har behov for at kunne udveksle informationer i et maskinlæsbart format på Webben. (Breitman, et al., 2007).

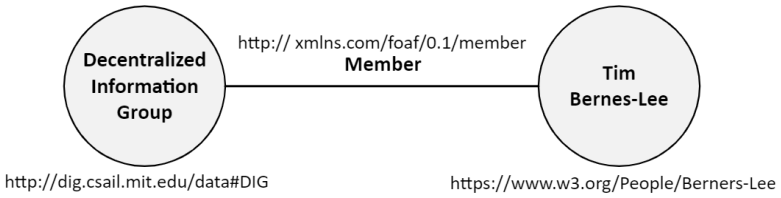
Selvom RDF omtales som et sprog, er RDF'er datamodeller. RDF-datamodeller opbygges på såkaldte *Statements*, der består af *subject-predicate-object*-værdier, kaldt triples, som beskriver data og deres relationer. Her er både subjektet og

objektet identificerede ressourcer med URI'er. Prædikatet er ligeledes repræsenteret af en URI, men den er det element, der beskriver, hvordan subjektet og objektet er relateret til hinanden. En illustration af forholdet mellem *subject, predicate og object* kan ses på nedenstående *Figur 6*.



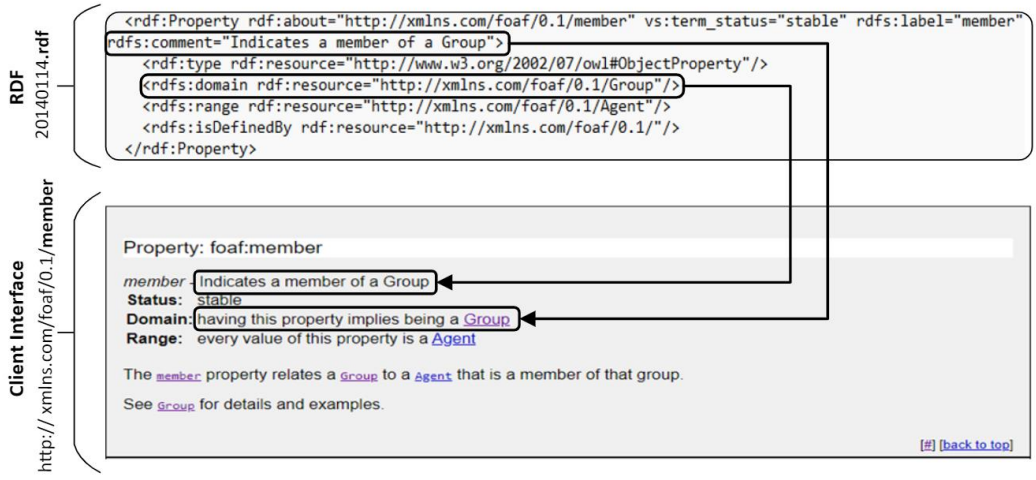
Figur 6 – RDF-triple/statement

Figur 7 viser et eksempel på et RDF-link, der er opbygget som en RDF-triple. Eksemplet beskriver, at ressourcen, Tim Bernes-Lee (Objektet), er Member (Prædikat) af ressourcen Decentralized Information Group (Subjektet). Begge ressourcer er identificeret med en URI.



Figur 7 – RDF-triple/statement, Eksempel – Inspiration fra (Breitman, et al., 2007)

På eksemplet skal det bemærkes, at Member (Prædikatet) ligeledes er identificeret med en URI. Ved at tilgå denne URI er det muligt at få definitionen af relationstypen, Member. Typen er blevet defineret og beskrevet med RDF ved at bruge RDF Vocabulary Definition Language og RDF Schema (RDFS). Nedenstående *Figur 8* viser et eksempel på, hvordan relationstypen er defineret i en .rdf-fil, med RDFS, og herefter præsenteret overskueligt på Web Clienten.



Figur 8 - Eksempel på Prædikat identificeret med oprettet URI (Brickley & Miller, 2014)

Eksemplet illustrer, hvordan forbindelser og relationer mellem data kan beskrives, så det både er læsbart for maskiner, men også for mennesker. Indholdet oplyser bl.a., at en ressource med prædikatet, Member, betyder, at ressourcen ligeledes er en del af en Group.

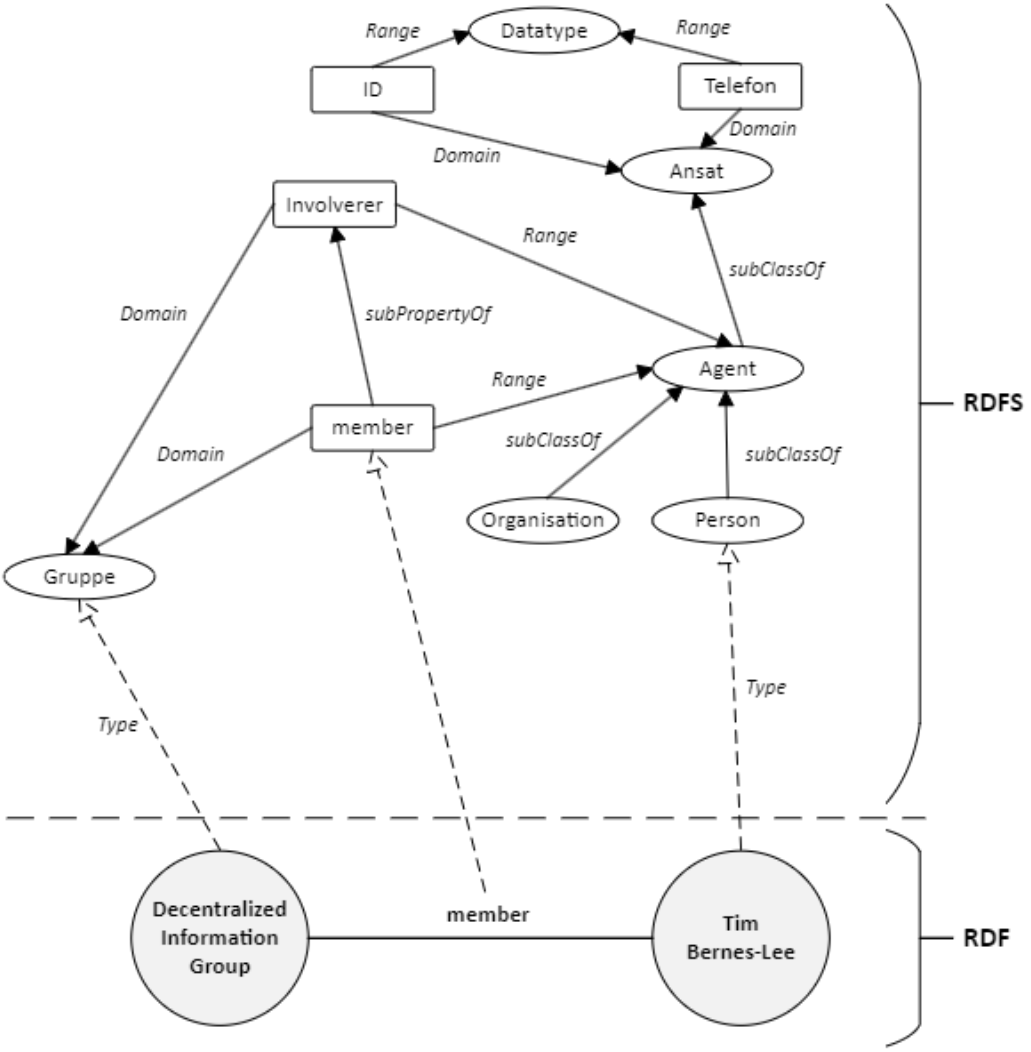
RDF Vocabulary Definition Language

RDF'er er hverken prædefinerede eller begrænsede til specifikke vidensområder. Ydermere bygger RDF ikke på forudbestemte antagelser af semantikken bag et domæne. RDF er derimod et universelt sprog, hvilket betyder, at det er muligt for brugere at beskrive ressourcer med deres egne definerede *Vocabularies* om deres domæne. Dette gøres igennem RDF Schema (RDFS). RDFS'er består af *Classes* og *Properties*, som tilsammen beskriver et *Vocabulary* for et domæne.

Classes definerer typer eller kategorier af ressourcer, som har nogle specifikke egenskaber eller funktioner tilfælles. Ressourcer, der er bestemt til at tilhøre en *Class,* kaldes instances af netop den *Class*. En vigtig funktion, som *Classes* har, er at skabe begrænsninger og derved en ramme for, hvilke RDF-*statements* der kan udtrykkes i et domæne. (Antoniou & Harmelen, 2004).

Properties beskriver relationen mellem objekter/ressourcer, defineret som *Classes* i RDFS, og hvordan objekter/ressourcer er relateret til specifikke værdier. Ifm. *Properties* anvendes egenskaberne *Domain* og *Range* til at definere, hvilke *Classes* eller værdityper en specifik Property kan tilknyttes til. Det er *Domain*-egenskaben, der indikerer, hvilken *Class* den enkelte Property kan anvendes på, mens *Range*-egenskaben anvendes til at angive værdien af en Property, der enten kan være en anden *Class* eller en datatype. (Breitman, et al., 2007).

Fælles for *Classes* og *Properties* er, at de opbygges RDFS i et defineret hierarki. Til at definere *Classes* hierarkiske position anvendes der i RDFS den prædefinerede *rdfs:subClassOf*-egenskab til at beskrive tilhørende underklasser af en *Class*. Relationen mellem to *Properties* defineres derimod med *rdfs:subPropertyOf*-egenskaben. I nedenstående *Figur 9* er der en illustration, som viser forholdet mellem et RDF-*statement* og begreberne relateret til RDFS: *Class, Property, Domain, Range, subClassOf og subPropertyOf*.



Figur 9 - Relation mellem RDF og RDFS – Inspiration fra (Antoniou & Harmelen, 2004)

Overstående figur tager udgangspunkt i RDF-*statementeksemplet*, som blev præsenteret på *Figur 7*, hvor det vises, hvordan denne triples relaterer sig til et domæne med et *Vocabulary* defineret med RDFS. Det ses bl.a., at Tim Bernes-Lee tilhører Classen, Person, som er en underklasse til *Agent*. Ydermere ses det, at Propertien, Member, er defineret med et *Domain* og *Range* svarende til Gruppe og Agent.

SPARQL

SPARQL er et query-language til RDF-grafer udarbejdet af W3C. SPARQL er designet specifikt til at hente og manipulere data, der er repræsenteret i RDF-format. Sproget giver bl.a. mulighed for at udforske data ud fra ukendte relationer, hente værdier fra data, og transformere data fra et *Vocabulary* til en anden. (Feigenbaum, 2009; W3C, 2013).

Ontology

Ontology stammer fra det græske sprog og betyder *Læren om væren*. Ydermere er det en filosofisk disciplin, hvor det handler om at udvikle og definere kategorisystemer, der beskriver en konkret opfattelse af verdenen. Ifm. SW kan ontologier beskrives som: *An ontology is a formal, explicit specification of a shared conceptualization* (Breitman, et al., 2007). Med *Formal* menes der, at ontologien skal være læsbar for maskiner. *Explicit* henviser til, at de indgående elementer skal være klart definerede. *Conceptualization* referer til en abstrakt model. Kombineret betyder det, at ontologier giver en struktureret måde at beskrive viden inden for en verden eller et domæne gennem definerede begreber og deres indbyrdes relationer, beskrevet med fastlagt Vocabulariy (Breitman, et al., 2007). Som førnævnt er RDF og RDFS ontologisprog, hvorfor de er værktøjer til at oprette ontologier. De kan udvides yderligere med Web Ontology Language (OWL).

Web Ontology Language

Igennem tiden blev der identificeret flere situationer, hvor *The Semantic Web* havde brug for mere avancerede muligheder end dem, som RDF og RDFS kunne understøtte. Der var blandt involverede aktører bred enighed om, at der var behov for et mere kraftfuldt Ontology Modeling Language. Resultatet heraf blev, at et mere rigt sprog ved navn DAML+OIL blev defineret af forskningsgrupper fra både USA og EU. W3C's gruppe, Web Ontology Working Group, brugte herefter DAML+OIL som et udgangspunkt i arbejdet om at definere OWL, som er designet til at være et standardiseret Ontology Language for SW (Antoniou & Harmelen, 2004). Ligesom RDF og RDFS er OWL også defineret som et Vocabulary, det er bare rigere på semantik. Ligeledes beskriver OWL Classes, Properties og Relations mellem objekter, men på sådan en måde at maskiner i højere grad kan fortolke indholdet af dem på Webben. (Antoniou & Harmelen, 2004; Breitman, et al., 2007).

Abox og Tbox

Abox og Tbox er begreber indenfor *Description Logics*, som anvendes til at definere *Knowledge Representation*. Begreberne Abox og Tbox er definition af, hvordan egenskaber og entiteter beskrives. Tbox eller *Terminology box* er den intentionelle viden, som beskriver menneskernes generelle viden om verdenen. Abox eller *Assertinal box* er den specifikke viden om et objekt eller individ. A-boksen definerer roller, koncepter og andre egenskaber, som beskriver terminologien anvendt i T-boksen. (Aberer, et al., 2007, p. 76; Rademaker, 2012).

Rule Languages - Rule Markup Language, Semantic Web Rule Language & triple

I modsætning til RDFS og OWL er Rule Languages (RL) designet til at definere og anvende logiske regler på data. Med reglerne er det muligt at kontrollere eller specificerer, hvordan data er relateret, udlede ny information og viden og automatisere

beslutningsprocesser. Ifm. SW er det muligt at anvende sprogene: Rule Markup Language (RuleML), SWRL eller triple. Sprogene har forskellige formål og anvendes i forskellige situationer.
RuleML bruges til at definere og udveksle regler på nettet, mens SWRL er et sprog, der kombinerer RuleML med OWL-ontologier. Dette muliggør udledning af ny viden baseret på den ontologiske information. Triples er designet til at kunne opstille regler, som kan arbejde og manipulere med RDF-data. Modsat regelsprog eksisterer valideringssprog til RDF-grafer.

Valideringssprog – Shapes Constraint Language

SHACL er et sprog til beskrivelse og validering af RDF-grafer, hvorfor syntaksen i SHACL også er opbygget af RDF-*Statements* – subjekt, prædikat og objekt. En SHACL-validering, der anvender én RDF-graf og én Shape-graf, resulterer i en valideringsrapport. En Shape-graf er en RDF-graf, som indeholder nul eller flere shapes, som anvendes i en SHACL-valideringsproces, hvor en RDF-graf valideres op imod de involverede shapes. SHACL består overordnet af tre komponenter: (1) *sh:Shape* (2) *sh:NodeShape* (3) *sh:PropertyShape*, hvor *sh:Shape* fungerer som grundlag for de to andre. Der arbejdes med nodes, som er de fundamentale enheder i RDF-grafen. SHACL bruger shapes til at definere regler for noder. *sh:nodeshapes* definerer valideringsregler for nodes og *sh:PropertyShape* definerer valideringsregler for specifikke egenskaber ved nodes. Targets anvendes til at definerer, hvilke nodes der skal valideres af en *sh:shape* baseret på deres klasse, relationer eller specifikke noder. (Klyne, et al., 2014).

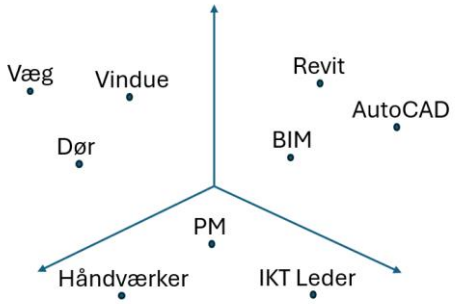
3.3 Vektor- og grafdatabaser

I takt med udviklingen af AI og LLM'er er behovet for nem adgang til og bearbejdning af både struktureret og ustruktureret data blevet større. Der er samtidig et øget behov for at inddrage ekstern information for at berige en LLMs respons og dermed reducere risikoen for modellens hallucinationer. For at opnå dette kan anvendelsen af ustruktureret og struktureret data, lagret i VDB- og GDB'er, være en løsning. (Sajid, 2024).

Vektordatabase

Hvor traditionelle DB'er, såsom relationelle databaser (RDB), primært organiserer og præsenterer data i tabeller baseret på struktureret data, giver VDB'er mulighed for at håndtere, gemme og søge i ustruktureret data samt etablere semantiske relationer mellem de tilgængelige data. VDB'er fungerer ved at gemme og omdanne data til numeriske formater, der repræsenterer længde- og breddegrader i en vektorrepræsentation. (Chaudhri, et al., 2024, pp. 3-4). Processen for at omdanne data til vektorer sker igennem anvendelsen af en Embeddings-model, som tildeler datastykker til vektorembeddings (Holdsworth & Kosinski, 2024). Opstillingen af numeriske

værdier, som vektorer, repræsenterer dataene som ensformige, og derfor vil vektorrepræsentationen af f.eks. Revit og BIM være placeret tættere end vektorrepræsentationen af væg og vindue. (Zilliz, 2022). En simplificeret opstilling af vektorrepræsentationen er vist på *Figur 10*.



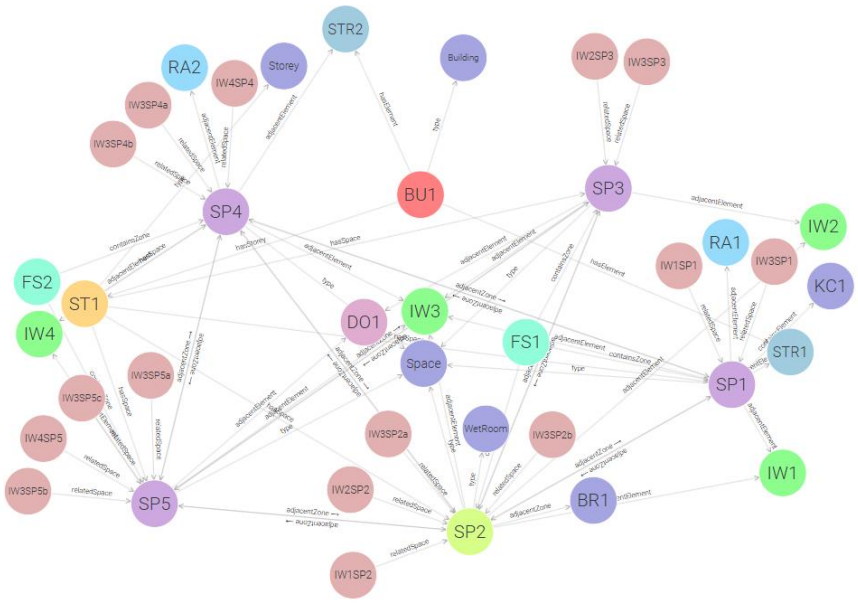
Figur 10 - Repræsentation af vektorembeddings - Inspireret af (Zilliz, 2022)

Brugerens forespørgsel omdannes til vektorembeddings, hvilket muliggør søgning blandt de lagrede embeddings og identificering af de genererede datafragmenter. Datafragmenterne og deres embeddings anvendes til Retrieval Augmented Generation (RAG) (Merrit, 2024). RAG-metoden adskiller sig fra andre typer ML-modeller, der udelukkende generer svar baseret på de data, modellen er trænet på. Ved at anvende RAG kan ML-modellerne inkludere eksterne kilder og data i deres svar, hvilket forbedrer og korrigerer responsen. RAG fungerer ved at identificere relevante dokumenter baseret på brugerens forespørgsel. Disse dokumenter anvendes som grundlag for den genererede besvarelse, hvilket gør det muligt for LLM'er at integrere eksterne data og dermed sikre pålidelige og præcise responser gennem effektiv informationssøgning. Ved at berige LLM'ens input med ekstern viden skabes mere nøjagtige svar, når brugerens forespørgsel behandles. RAG-metoden betragtes som en måde at augmentere for responsen fra LLM og dermed minimere risikoen for hallucinationerne ved at inddragelse af eksterne data. (Liu, 2024; Chaudhri, et al., 2024; Sajid, 2024).

Grafdatabaser

Mens VDB'en henvender sig til ustruktureret data, anvendes struktureret data i LD, som repræsenteres vha. RDF-triples i en GDB, hvor dataene organiseres som en graf gennem nodes og relationer. I GDB'er anvendes begreberne *vertices* og *edges,* som hhv. beskriver objekter som mennesker eller artefakter og deres relationer (Neo4j, 2024). GDB'er adskiller sig fra RDB'er ved at bruge *edges* til at definere relationer mellem data, i stedet for at der anvendes *primær- og fremmednøgler* som i traditionelle RDB'er. (Robinson, et al., 2015; Wang, 2024). En type af GDB er en såkaldt Triple Store (TS). TS adskiller sig fra den traditionelle GDB ved at anvende en struktur, der er målrettet SWT og LD. Dette opnås ved at erstatte *vertices* og *edges* med RDF-strukturen, som repræsenterer data i form af triples. (Duckham, et al., 2024;

Chambers, 2013). Nedenstående *Figur 11* illustrerer en genereret graf udledt af LD i *Bilag 34*.



Figur 11 - Graf genereret fra Bilag 34 - Udarbejdet i GraphDB (GraphDB, 2024)

4 STATE OF THE ART

Dette kapitel har til formål at kontekstualisere undersøgelsen ved at gennemgå den aktuelle forskningsstatus og de nyeste teknologiske fremskridt inden for problemfeltet. Kapitlet er struktureret i ni underafsnit, som indledes med en præsentation af traditionelle regeltjek, regulatoriske dokumenter, typer af regler og ACC-faser. Derefter undersøges byggebranchens aktører og deres kompetencer inden for IFC og ACC-processer. Efterfølgende præsenteres AI og relaterede teknologier og begreber ift. ACC-processer samt anvendelsen af SWT og LD. I forlængelse her redegøres der for anvendelsen RDF-grafer og ontologier og deres relation ift. SWT og LD. Herefter præsenteres litteratur omhandlende ML og LLM. Afslutningsvis præsenteres forskellige studier, hvor relevante prototyper er blevet udviklet, og hvor nogle af de nævnte teknologier er anvendt på forskellig vis i prototyperne.

4.1 Traditionelt regeltjek og regulatoriske dokumenter

Byggeriets mange processer bygger som regel på underliggende data og informationer, der sikrer, at beslutninger træffes på et informeret og velbegrundet grundlag. Disse data kan lagres på mange forskellige måder og lokationer, hvor et vigtigt element er BIM-modeller (Shin & Issa, 2021). BIM-modeller kan indeholde enorme mængder af data med komplekse relationer mellem objekter, som kan have diverse egenskaber og samtidig tilhøre en bestemt type (Ismali & Scherer, 2017; Pauwels, et al., 2024). Anvendelsen af BIM er en voksende kilde til lagring af data ift. geometri, topologi og semantik, og det er med til at muliggøre kommunikation, koordination, analyse og kvalitetskontrol og -sikring på tværs af involverede aktører i byggeriets livscyklus (Zheng & Fischer, 2023; Nepal, et al., 2012). I sin oprindelse var et hovedformål med at implementere BIM i datahåndtering at reducere tab af data og informationer under udveksling af disse (Pauwels, et al., 2024).

I denne forbindelse næves begrebet ACC i flere videnskabelige artikler, hvor det indikeres, at mange forskellige studier og metoder er udarbejdet og afprøvet ifm. med datahåndtering og BIM-modeller (Nuyts , et al., 2024). ACC giver mulighed for at kontrollere, om f.eks. en BIM-model overholder gældende regulativer, politikker og regler. I øjeblikket kræver ACC stadig mange manuelle processer, hvilket medfører betydeligt tids- og ressourceforbrug, hvor kun få studier har opnået fuld automatisering (Guo, et al., 2021; Zhang & El-Gohary, 2013). ACC anses for at have flere fordele end *Manuel Compliance Checking* fordi, at det forventes, at det kan reducere både tid, omkostninger og de potentielle fejl, som kan opstå ifm. manuelle regeltjek (Zhang & El-Gohary, 2015). Traditionelt foregår indhentning af data via manuelle søgninger, fra bl.a. PDF'er og andre dokumenter, som i sidste ende ikke er særlig effektivt (Anumba, et al., 2021). Det understreges samtidigt, at der er et praktisk og teknologisk hul, hvor de nuværende metoder til ACC, som tidligere nævnt, er

ufleksible og uigennemsigtige især ift. definitionen af regler og udførelsen heraf (Ghannad, et al., 2019). Derudover opleves det som en langsom proces at få udviklet disse ACC-systemer, hvilket der i artiklen af *Ghannad, et al. (2019)* er identificeret flere årsager til: (1) Der er en generel iboende kompleksitet (2) Branchen er af natur fragmenteret (3) Der er et stort antal af interessenter (4) Der er en modstand overfor forandring (5) Der er en manglende motivation til at adaptere nye teknologier (Ghannad, et al., 2019). Modsat ses flere fordele ved ACC såsom: (1) Tidlig identifikation af potentielle regelbrud (2) Brugen af BIM-modeller fremmes (3) Bedre integration af interessentinput i designprocesser (4) Færre regelovertrædelser på grund af hyppigere og mere effektiv kontrol (Zhang & El-Gohary, 2015).

Reglerne og kravene, som skal overholdes ifm. ACC, udspringer fra forskellige regulatoriske dokumenter såsom Bygningsreglementet (BR), tekniske dokumenter, kontrakter, normer, standarder eller generel juridisk information (Zhang & EL-Gohary, 2021; Ghannad, et al., 2019; Hjelseth, 2016). Disse regulatoriske dokumenter er struktureret på en måde, der skaber udfordringer ifm. udførelsen af ACC. Modsat BIM-data er størstedelen af disse regulatoriske dokumenter skrevet i udelukkende menneskeligt læsbare tekstformater, enten i NL, formel notation eller en kombination heraf. Dette medfører, at dokumenternes struktur typisk er kompleks og ustruktureret, hvilket gør det vanskeligt for aktører at finde relevante informationer (Ghannad, et al., 2019; Krijnen & van Berlo, 2016; Zhong & El-Diraby, 2024; Nuyts , et al., 2024). Forskere bemærker, at når regler udarbejdes, primært i NL, af institutioner og standardiseringsorganer, mangler der et klart og let forståeligt overblik over, hvilke konsekvenserne disse regler har for beregningsmæssig kompleksitet og beslutsomhed ifm. ACC, hvilket i sidste ende påvirker ACC-processen (Krijnen & van Berlo, 2016). Mange af de nuværende ACC-systemer kræver manuel håndtering for at udtrække krav fra de regulatoriske dokumenter, for derefter at omdanne dem til et format som er computerlæseligt, hvilket anses for at være et fundamentalt og udfordrende aspekt i ACC (Zhang & El-Gohary, 2013; Zhang & El-Gohary, 2015; Ghannad, et al., 2019). Et velfungerende ACC-system vil kræve, at al data, både BIM-data og regulatorisk data, er repræsenteret i computerlæseligt format (Nuyts , et al., 2024). Artiklen af *Ghannad et al., (2019)* foreslår en tilgang til formalisering af regler, hvor en samlet enhed er i stand til at indfange alle elementer af regler, semantikker, ontologier og logikker uanset kompleksiteten af reglerne (Ghannad, et al., 2019).

I forlængelse heraf fremhæver *Zhou & El-Gohary (2021),* at der er to forudsætninger, som er essentielle for at kunne opnå et niveau af fuld automatisering: (1) Regulativer skal transformeres til computerlæsbare repræsentationer (2) Egenskaber og relationer i BIM-modeller skal tilpasses således, at de stemmer overens med de begreber og relationer, der står beskrevet i de computerlæsbare regulativer. Dette vil sikre, at både BIM-modellen og regulativerne anvender samme struktur til at definere de samme objekter. En sådan proces anses for at være udfordrende fordi, at

BIM-modeller og regulativer anvender forskellige datarepræsentationer og terminologier (Zhou & El-Gohary, 2021).

4.2 Regeltjek, ACC-faser og regeltyper

I byggebranchen anvendes en række forskellige kontroltyper. Disse består hhv. af: (1) *Regulation* Checking som er kontroller af regulativer og krav (2) *Compliance Checking* som er overholdelseskontroller (3) *Quality Ckecking* som er kvalitetskontrol (Pauwels, et al., 2024). Når disse regler forsøges automatiseret, refereres der til ACC. Ifm. ACC omtales to overordnede typer af kontroller: (1) *Continuous Project Check* (2) *Hand-over Check*. *Continuous Project Check* er løbende kontinuerlige kontroller ofte i de tidlige faser af byggeriet, og Hand-over check er enkeltpunktkontroller ofte udført iht. kontraktuelle forhold. (Pauwels, et al., 2024).

Et ACC-system kan generelt inddeles i fire faser: (1) Regelfortolkning (2) Forberedelse af BIM-model (3) Eksekvering af regler (4) Rapportering af regler (Eastman, et al., 2009; Burggräf, et al., 2021; Pauwels, et al., 2024). Regler og krav er udarbejdet af mennesker og repræsenteres typisk i menneskeligt læsbare formater, såsom tekst og tabeller (Burggräf, et al., 2021; Eastman, et al., 2009; Pauwels , et al., 2021). Regelfortolkningsfasen indebærer, at regler skal oversættes til maskinlæsbare formater, hvilket kan foregå igennem forskellige metoder. Det kan være manuel kodning af regler, automatisk transformation af regler vha. f.eks. NLP eller brugen af specialudviklet software til at omdanne menneskeligt læselige regler til maskinlæsbare (Eastman, et al., 2009). Tilpasningen af BIM-modellen indebærer, at det sikres, at de opstillede regler og krav for projektet stemmer overens med BIM-dataene (Eastman, et al., 2009; Pauwels, et al., 2024). Dette er dog sjældent tilfældet, hvorfor en konvertering tit er nødvendig af både BIM-model og tilhørende regler og krav (Pauwels, et al., 2024). Forberedelsesfasen kræver, at modellerne forberedes, så de indeholder den nødvendige information i veldefinerede og aftalte strukturer (Eastman, et al., 2009). I eksekveringsfasen kombineres de to foregående faser, hvor regler sammenholdes med BIM-modellen (Burggräf, et al., 2021; Eastman, et al., 2009). Dette vil typisk indebære softwareprogrammering, da det skal give brugeren mulighed for at udvælge specifikke regelsæt og datasæt via et tilpasset interaktivt interface (Pauwels, et al., 2024). Den sidste fase i ACC er rapportering af resultaterne, hvilket kan forgå på forskellige måder (Burggräf, et al., 2021; Eastman, et al., 2009). Nogle applikationer præsenterer f.eks. grafiske rapporter for brugeren, som inkluderer de overtrådte regler og de involverede objekter (Burggräf, et al., 2021). *Eastman, et al (2009)* stiller krav til, at rapporten bør give et klart overblik over de regler, der er anvendt i kontrollen, samt de objekter og attributværdier der er blevet tjekket. Derudover skal der skelnes mellem regler, der er bestået, ikke bestået og regler der ikke kunne kontrolleres pga. fejl eller inkonsistens i dataene. Derudover skal resultat

omdannes til NL i et intuitivt brugerinterface, hvilket kan være en svær proces (Eastman, et al., 2009).

I artiklen af *Burggräf, et al. 2021* identificeres fire regeltyper: (1) *Interference Rules* (2) *Existence Rules* (2) *Compliance Rules* (4) *Calculation Rules*. *Interference Rules* bruges til at udføre geometriske kollisionskontroller mellem enkelte objekter eller flere BIM-modeller. *Existence Rules* bruges til at kontrollere, om objekter, attributter og attributværdier er til stede i modellen. *Compliance Rules* anvendes til at kontrollere modeller for overholdelse af attributværdier, dimensioner og forbindelsespunkters placering ved at sammenligne de respektive værdier fra alle modeller mod hinanden. Med denne regeltype kan, tidligere nævnte, regulatoriske dokumenter såsom BR kontrolleres. *Calculation Rules* repræsenterer matematiske beregninger, hvor attributværdierne, der anvendes til beregningerne, er tilgængelige i BIM-modellen. (Burggräf, et al., 2021).

Eastman, et al. 2009, skriver, at et regelbaseret vurderingsværktøj kan implementeres på forskellige platforme og skelner her imellem tre muligheder: (1) En applikation tæt knyttet til et designværktøj, såsom et plug-in, der giver designeren mulighed for kontrol, når personen ønsker det (2) En selvstændig applikation der fungerer på desktop parallelt med et designværktøj (3) En webbaseret applikation der kan modtage data fra forskellige kilder og behandle data ud fra dette. (Eastman, et al., 2009).

4.3 Aktørs kompetencer & IFC i ACC-processen

Traditionel håndtering af data og regler i byggebranchen fører tit tilbage til den enkelte aktør, hvis jobfunktion kan bestå af at udføre regeltjek pba. f.eks. en BIM-model og nogle regulatoriske dokumenter. Diverse videnskabelige artikler og studier udtrykker en skepsis angående den enkelte aktørs kompetencer, når det kommer til anvendelsen af BIM og generel software manipulation til f.eks. at udføre queries pba. af en stor datamodel (Anumba, et al., 2021). I takt med at BIM-modeller bliver hyppigere anvendt, bliver datamængden i BIM-modellerne større og mere komplekst. Dette medfører, at dataudtræk fra BIM-modeller kan være trættende og tidskrævende for ikke-tekniske aktører, som i forvejen også har begrænsede færdigheder i at navigere i BIM-software og BIM-modellen (Wang , et al., 2022; Anumba, et al., 2021). Et sådant dataudtræk vil kræve ekspertise og erfaring inden for BIM-data og -struktur (Issa, et al., 2022).

Ved udveksling af BIM-data, som kan indeholde omfattende informationer om et projekt, beskrives disse data generelt ved brug af IFC, da det anses for at kunne lette dataudvekslingen (Guo, et al., 2021). Dog er der nogle begrænsninger identificeret ved brugen af IFC ifm. ACC. ACC kræver mere detaljerede og specifikke data, som IFC ikke nødvendigvis understøtter, hvilket derfor kan gøre det svært at

automatisere regler og kontroller udelukkende baseret på en IFC (Guo, et al., 2021). Derudover fremhæves også interoperabilitetsproblemer med IFC, som fører til datatab ved konvertering mellem forskellige formater, standarder og software (Senthilvel & Beetz, 2020). Mange queries på dataudtræk fra f.eks. BIM-modeller kan være svære at gennemføre med nuværende softwareværktøjer, og de fleste af dem fungerer kun på specifikke IFC-datamodeller (Ismali & Scherer, 2017). *Ismali & Scherer (2017)* understreger, at IFC udgør af en fastlåst og kompleks hierarkisk struktur, som gør det vanskeligt at udføre en simpel manuel udtrækning af data. Derudover påpeges det, at en sådan udtrækning af data kræver dyb forståelse af selve IFC-datamodellen.

Særligt én udfordring påpeges i flere forskningsstudier. Her er udfordringen at tilpasse semantikken i BIM-modellerne, i formatet IFC, med regulativernes semantik, i NL, så udførelsen af overensstemmelseskontrol mellem BIM-modellen og reglerne muliggøres (Zhang & El-Gohary, 2023; Zhou & El-Gohary, 2021). I øjeblikket er denne tilpasning ikke fuldt automatiseret, men udføres delvist manuelt eller semiautomatisk, hvilket er en proces, der kan være vanskelig at undgå (Guo, et al., 2021; Zhou & El-Gohary, 2021). Forskningsstudierne understreger, at der skal findes en løsning, der kan bygge bro mellem BIM-data og regulativerne, hvor flere peger på, at der skal udvikles en ny semantisk tilgang for ACC. En tilgang der automatisk kan tilpasse de semantiske repræsentationer og terminologien i IFC til de naturlige sproglige regulativer (Zhang & El-Gohary, 2023). En sådan afstemning af data udføres i nuværende ACC-systemer, ofte gennem hardcoding, f.eks. ved brug af modellerings- eller query language ontologi- eller ordbogsbaseret matching eller søgemetoder (Zhang & El-Gohary, 2023).

4.4 Sprog til ACC – muligheder og begrænsninger

Eksisterende tilgange til ACC består bl.a. af at udtrække strukturerede bygningsdata fra BIM-modeller vha. et struktureret query language såsom SQL eller SPARQL (Anumba, et al., 2021). Disse eksisterende tilgange kræver, som tidligere belyst ift. andre typer software, at brugeren er fortrolig med de værktøjer, som anvendes til dataudtrækket. I denne forbindelse foreslår relevante forskningsprojekter anvendelsen af SQL-relaterede metoder til at udtrække information fra en BIM-model (Anumba, et al., 2021). Det vil dog kræve, at data fra BIM-modellen konverteres til IFC-formatet og derefter placeres i en anden DB-type (Anumba, et al., 2021). Samtidig understreger *Pauwels, et al. (2024)*, at ACC-feltet og diverse tilgange hertil udvikler sig hurtigt og er drevet af formaliseringsmetoder vha. logikbaserede deklarative sprog fra SWT såsom SHACL, SPARQL, RDF og OWL. Et logikbaseret deklarativt sprog indebærer, at et system der anvender disse sprog automatisk, kan lave logiske slutninger og udlede resultater (Pauwels, et al., 2024).

I artiklen af *Nuyts, et al. (2024)* er der udført et studie, hvor en større komparativ analyse og diskussion omhandlende diverse sprog og tilgange til ACC er udført. Dette studie vil derfor danne grundlag for nærværende afsnit og undervejs blive suppleret med relevante pointer fra andre studier. I studiet af *Nuyts, et al. (2024)* inddeles ACC i tre tilgange: (1) Tilgange der arbejder med IFC-data (2) Generelle åbne datastandarder og deres tilhørende skemadefinitionssprog (3) Tilgange der bygger på LD. De tre tilgange vurderes ud fra fem forskellige typer af krav og regler: (1) Krav og regler til datatilgængelighed (2) Krav og regler til værdier/værdigrænser (3) Krav og regler til relationer (4) Matematiske krav og regler (5) Betingede krav og regler. (Nuyts , et al., 2024).

Tilgangene, der arbejder med IFC-data, inkluderer både selvstændige softwareværktøjer og standarder fra organisationen BuildingSMART. Blandt softwareværktøjerne er bl.a. Solibri Model Checker (SMC), som er et af de mest anvendte i branchen, selvom det er begrænset til kun at arbejde inden for eget software, hvilket i sig selv udgør i barriere for softwarets anvendelse i et ACC-system (Nuyts , et al., 2024; Krijnen & van Berlo, 2016; Guo, et al., 2021; Senthilvel & Beetz, 2020). Dog kan SMC bruges til at definere og udføre tjek ud fra alle de førnævnte fem regler og krav. Ud over softwareværktøjer anvendes BuildingSMART-standarder såsom Model View Definition (MVD) til at tjekke, om IFC-data overholder kravene. MVD definerer specifikke arbejdsprocesser og anvendelser, men standarden bliver gradvist erstattet af Information Delivery Specifications (IDS), som er designet til at gøre dataoverholdelse lettere for slutbrugeren. IDS er stadig under udvikling, hvilket udgør en begrænsning, der gør avanceret ACC besværligt. Dette skyldes bl.a., at matematiske og betingede krav og regler i øjeblikket ikke kan håndteres med IDS. (Nuyts , et al., 2024).

Ud over BIM-specifikke datamodeller, i form af IFC, anvendes mere generelle dataformater såsom CSV, XML (som følger XSD-strukturen) og JSON ofte til dataudveksling. Disse formater er populære grundet deres udbredelse og alsidighed. Formaterne har hver deres fordele og ulemper og kan være nyttige inden for datavalidering, men de er alle begrænset af deres respektive format og struktur og mangler avancerede funktioner. *Yuexin, et al. (2023)* beskriver, at XML-tilgangen mangler semantiske beskrivelser, evner til at skabe genklang, forståelse og visualisering. XML er en W3C-standard og imødekommer alle fem krav og regler, hvor JSON ikke kan definere matematiske begrænsninger. (Nuyts , et al., 2024).

IFC kan repræsenteres i flere formater såsom STEP, XML og RDF (skrevet i ifcOWL) (Stolk & McGlinn, 2020). ifcOWL bygger på LD-principper og kan understøtte sammenkobling mellem andre datasæt, der er udtrykt i RDF (Stolk & McGlinn, 2020). LD-tilgangene anvender RDF til at repræsentere data i grafformat og gør det muligt at anvende avancerede semantiske sprog og værktøjer til ACC. Disse sprog omfatter bl.a. OWL, SWRL, SPARQL og SHACL. OWL kan bruges til at beskrive begreber og

deres relationer inden for en ontologi. OWL kan ikke selvstændigt antage, at det, der ikke er eksplicit nævnt i datasættet, er falsk. Dette kan være udfordrende for brugere, som forventer, at fravær af data betyder, at noget er falsk. OWL kan ikke bestemme, hvad der er korrekte og forkerte data i et datasæt, men kan påvise inkonsistens i datasættet og ontologien som helhed og aflede yderligere udsagn, hvis der ikke opdages nogen inkonsistensen. OWL-sproget er en W3C-standard, men sproget er ikke tilstrækkeligt til en ACC-proces og vil i den forbindelse kræve anvendelsen af SPARQL-queries. *Nuyts, et al. (2024)* beskriver, at en ulempe ved at arbejde med OWL er, at mange finder det kontraintuitivt fordi, hvis noget data mangler, er det nødvendigvis ikke en fejl eller falsk information. OWL definerer ikke matematiske operationer og kan derfor ikke imødekomme de matematiske krav og regler, som det eneste af de fem krav og regler. Sammenlignes ifcOWL-tilgangen med førnævnte XML-tilgang har den en forbedret maskinforståelse, kodning, evne til at skabe sammenhæng, forståelse og visualisering (Yuexin, et al., 2023). *Wang, et al. (2022)* og *Issa, et al. (2022)* understreger, at forskning inden for BIM/IFC kræver, at bygningsdata omdannes fra BIM/IFC til RDF eller OWL, men den proces er tidskrævende og kompliceret og kræver erfaring med formaterne. Derudover er filstørrelsen på en RDF-fil eller OWL-fil større end den tilsvarende BIM/IFC-fil (Wang , et al., 2022).

SWRL er et regelsprog, som kan skrives ud fra forskellige syntakser og er ikke en W3C-standard. SWRL er i stand til at definere de fem krav og regler, men det er, ligesom OWL, ikke tilstrækkelig i sig selv til en datavalideringsproces i og med, at det det er et sprog til inferensregler. Derfor afhænger sproget af SPARQL, når datavalidering skal foregå. I artiklen af *Zhou & El-Gohary (2021)* anvendes f.eks. semantiske regelsprog, såsom SWRL, til at repræsentere regulativerne og IFC og OWL til at repræsentere BIM-modellerne. SPARQL er, som tidligere beskrevet i *Afsnit 4.4*, et query language og en W3C-standard til at udføre søgninger i RDF-datasæt. Hvor de førnævnte sprog kræver udvidelser eller tilføjelser, kan SPARQL håndtere alle fem regler og krav uden yderligere udvidelser eller tilføjelser. Dog mangler tilgangen en standardiseret valideringsrapport, hvortil en udfordring bl.a. opstår i og med, at resultaterne for en SPARQL-queries ikke altid garanteres at være ensartede.

SHACL er er W3C-standard, som bruges til at validere RDF-grafer pba. regler og krav (Hagedorn, et al., 2023; Nuyts , et al., 2024). Syntaksen i SHACL understøtter brugen af SPARQL-queries, hvilket betyder, at det er muligt at skrive mere komplekse og dynamiske valideringsregler frem for enkeltstående SHACL-regler. SHACL imødekommer alle fem regler og krav. Desuden resulterer SHACL i en standardiseret valideringsrapport, hvori den kun nævner de data, som ikke er i overensstemmelse med de opsatte regler. *Eastman, et al. (2009)* og *Pauwels, et al. (2024)* understreger, at en SHACL-valideringsrapport er tilstrækkeligt struktureret til at kunne bygge et intuitivt brugerinterface ovenpå disse rapporter. I studiet af *Stolk & McGlinn (2020)* undersøges en datavalideringsproces, hvor der tages udgangspunkt i tidligere nævnte

ifcOWL. Her anvendes SHACL til at identificere fejl i en datamodel. Studiet konkluderer, at fremtidige versioner af OWL vil kunne drage fordel af at inkorporere SHACL, hvorfor det også anbefales, at der gradvist tages skridt mod at adaptere SHACL i datavalideringen og integrere det i selve processen (Stolk & McGlinn, 2020).

Overordnet set konkluderer *Nuyts, et al. (2024)*, at SMC, XML (XSD), SWRL, SPARQL og SHACL alle kan definere de fem regler og krav, hvor de andre tilgange kræver yderligere håndtering. Datainputtet til de forskellige tilgange kan opdeles efter IFC, JSON, XML og RDF. IFC-data står alene i denne sammenhæng fordi, at de metoder, der behandler disse datainput (SMC & IDS), ikke kan behandle andre datatyper. Modsat tilgangene der anvender IFC som datainput, anvendes JSON, XML og RDF stort set alle domæner. En anden problematik ved SMC er, at det er et lukket software. Dataoutputtet er også et væsentligt parameter ift. ACC. *Nuyts, et al. (2024)* påpeger, at det er blevet bemærket, at kun JSON Schema, SPARQL og SHACL resulterer i et standardiseret outputformat. I og med at IFC kun anvendes inden for byggebranchen, konkluderes det i studiet, at det medfører en langsommere udviklingsproces af de IFC-baserede tilgange i modsætning til, hvis det var en udbredt standard, som blev anvendt af flere brancher. Det modsatte er derimod tilfældet for de tilgange, som bygger på de generelle datastandarder og LD-tilgange. Afslutningsvis konkluderes det yderligere at ift. ACC og LD, er SHACL-tilgangen mest fordelagtig, hvor den mest åbenlyse fordel er, at den understøtter de fem regler og krav samt en standardiseret valideringsrapport, hvilket gør det muligt at implementere den i et ACC-system. Valideringsrapporten er en standardiseret RDF-graf, som består af selve valideringen i et maskinlæsbart format (Nuyts , et al., 2024; Pauwels, et al., 2024).

4.5 AI – Natural Language Processing, Tokenization & POS-tagging

I artiklen af *Shin & Issa (2021)* adresseres to centrale problematikker vedrørende udtræk af BIM-data. Den første problematik relaterer sig til de tidligere nævnte udfordringer om IFC og fremhæver, at IFC-datamodellen ikke nødvendigvis indeholder alle tilgængelige data fra BIM-modellen. Samtidig belyses det også, at nuværende forskning ikke har undersøgt mulighederne for at hente BIM-data direkte fra BIM-modeller, f.eks. ved brug af softwarets API, men i stedet, hvordan denne data kan konverteres til et andet format såsom IFC. Den anden problematik der omtales er, at de eksisterende tilgange ikke er i stand til at behandle NL. De eksisterende søgetilgange kræver, at brugeren har specifik viden om de præcise kommandoer og termer, der bruges i BIM-software. Dette udgør i sig selv en barriere for mange brugere i og med, at de skal have kendskab til de nøjagtige kommandoer for at udtrække de relevante data. (Shin & Issa, 2021).

For at gøre det nemmere for brugere, som ikke har kompetencerne inden for software manipulation eller BIM-kommandoer og -termer, anbefaler flere studier anvendelsen af NLP i en kombination med ACC. *Shin & Issa (2021)* anser det som en nødvendighed, at der udvikles en platform til udtræk af data, der semantisk kan forstå en brugeres spørgsmål. Her er NLP et vigtigt redskab i og med, at det kan tage højde for forskellige syntaksvariationer (Shin & Issa, 2021). *Shin & Issa (2021)* understreger derfor at BIM, semantisk forståelse og NLP er tæt forbundne, og ved at udnytte NLP til at transformere NLs semantik til maskinlæselige ontologier kan det være muligt at fremme interoperabiliteten mellem BIM-systemer. NLP er et underområde af AI og bygger på en computerbaseret tilgang til at fortolke, repræsentere og håndtere tale eller tekst formuleret som NL således, at resultatet heraf kan bearbejdes på en menneskelignende måde til diverse opgaver (Shin & Issa, 2021; Zhang & El-Gohary, 2015; Wang , et al., 2022; Xue, et al., 2024). Ifm. ACC kan AI og herunder NLP bruges til at udtrække regeltermer og logiske sammenhænge mellem disse termer fra tekstbaserede regulatoriske dokumenter, ved at regeltermerne transformeres til computerlæselige (Guo, et al., 2021; Ghannad, et al., 2019). De førnævnte regulatoriske dokumenter undergår hyppige ændringer og opdateringer, hvilket komplicerer den automatiserede oversættelse til maskinlæsbare formater. Her kan NLP og AI-teknikker muligvis bidrage til at opretholde en opdateret repræsentation af reglerne i en computerlæselig version (Ghannad, et al., 2019). *Wang, et al. (2022)* konkluderer i sit studie, at sammenlignes forespørgsler og respons, udført i NL, med queries, udført i SQL og SPARQL, er NL mere acceptable for ikke-ekspert BIM-brugere. Flere studier har undersøgt potentielle muligheder ved en kombination af SQL og SPARQL, kombineret med NL i processen, hvor forespørgsler udføres pba. BIM-data. Disse studier gennemgås i *Afsnit 4.9*. I studiet af *Shin et al (2022)* foreslås et system, som genererer SQL-query vha. NLP. Her konverteres en simpel NL-forespørgsel til en SQL-query, hvorefter systemet processerer denne query og returnerer et output til brugeren (Shin, et al., 2022). Den første del af databehandlingen, når der gøres brug NLP, handler om at forstå syntaksen bag brugerens forespørgsel (Nabavi, et al., 2023). I denne sammenhæng er Tokenization (TO) og Part-of-Speech tagging (POS) essentielle komponenter. TO er processen med at opdele en tekst i separate ord eller bogstavkombinationer, som omtales tokens, og samtidig eliminere tegnsætning samt Stopwords (SWO). SWO er ord, som ikke har nogen væsentlig betydning for selve forespørgslen. Ofte generes for mange tokens, hvilket opleves at kunne vanskeliggøre en præcis datasøgning af de tekstbaserede data, og derfor elimineres SWO for at kunne opnå en mere præcis datasøgning. I og med at POS fungerer på token-niveau, er det nødvendigt først at udføre TO på forespørgslen. POS indebærer, at alle resterende ord tagges ud fra ordklasse, såsom navneord, udsagnsord, adjektiver osv. (Nabavi, et al., 2023; Anumba, et al., 2021). Derefter oprettes via en parser en syntaktisk træstruktur af queries, som viser relationerne mellem ordene i query'en. Denne træstruktur tager sit udgangspunkt i de identificerede navneord i forespørgslen fordi, at navneordene anses for at være det centrale element og repræsenterer den vigtigste ordklasse i forespørgslen (Anumba, et al., 2021; Nabavi,

et al., 2023). Når nøgleordene er identificeret, og relationen imellem dem er oprettet, kan dette derefter anvendes til at fremfinde de relevante IFC-data og generer en respons i NL (Anumba, et al., 2021). I forlængelse heraf anvendes *Pattern-matching-based rules* (Zhang & El-Gohary, 2015). Det er regler, hvor der er defineret et sæt af resultater, der aktiveres, når et mønster af en specifik sekvens (bl.a. TO og POS) genkendes. Reglerne kan tilpasses og anvendes på forskellige måder alt efter formål og anvendelsesområde (Zhang & El-Gohary, 2015). Uanset hvordan reglerne er varieret af brugeren, følger de samme grundlæggende struktur, der er baseret på princippet om, at hvis et mønster identificeres, medfører det et forudbestemt resultat (Zhang & El-Gohary, 2015). *Zhang & El-Gohary (2015)* omtaler to typer informationsudtræk (1) Syntaksbaseret (2) semantiskbaseret. Her understreges det, at den semantiskbaserede tilgang har vist sig at opnå bedre resultater end, hvis udelukkende den syntaksbaserede tilgang blev anvendt.

4.6 Semantisk Webteknologi & Linked Data – muligheder og begrænsninger

SWT anvendes i stigende grad til digital repræsentation af bygninger inden for byggebranchen og supplerer samtidig eksisterende modelleringstilgange inden for BIM (Hamdan & Scherer, 2020; Perevalov, et al., 2022). Samtidig står SWT overfor en stor udvikling, hvor fokus er at give en mere direkte og softwareneutral adgang til bygningsdata (Pauwels, et al., 2024). Denne udvikling bygger bl.a. på et mål om bedre vidensdeling og repræsentation af viden, hvor webdata gøres maskinlæsbart og -forståeligt ved at beskrive koncepter, entiteter og relationer imellem dem (Perevalov, et al., 2022; Hjelseth, 2016). I denne forbindelse omtales datasiloer, som er data, bl.a. fra forskellige websteder, som mangler indsigt i de underliggende relationer fra udefrakommende data, og hvordan disse er forbundet, hvilket udgør en udfordring. Når disse datasiloer kombineres i en RDF-graf, vil hele processen for datahåndtering være mere effektiv og anvendelig. (Yuexin, et al., 2023).

LD fremhæves i adskillige studier, som et centralt princip i udviklingen af SWT, og anses for at have potentialet til at adressere og overvinde de nuværende udfordringer med interoperabilitet (Hagedorn & König, 2021; Pauwels, et al., 2024). *Hagedorn, et al. (2023)* understreger, i forlængelse af datasiloer, at når data fra flere informationskilder, der er forbundet med hinanden, skal valideres, bliver tilgange som semantisk regelkontrol og regelsystemer, der anvender SWT og LD afgørende. Dog identificerer *Pauwels, et al (2024)* diverse udfordringer og begrænsninger ved brugen af LD i ACC. Når det omhandler udførelsen af kravkontroller, er data typisk ikke samlet i ét format og én lokation. Dernæst er det udfordrende at kontrollere, hvad der omtales som *Vague Constraints* og *Computationally Constraints*. *Vague Constraints* er uklare krav og kontroller, som kan være svære at formalisere eller gøre eksplicitte. *Computationally Constraints* er krav og kontroller, som kræver

geometriske beregninger. (Pauwels, et al., 2024). Ud over problematikkerne påpeget af *Pauwels, et al (2024)*, identificerer *Pauwels, et al. (2021)* yderligere to generelle udfordringer for implementeringen af LD i byggebranchen. Den mest kritiske udfordring ved at implementere LD-tilgange er, at det vil kræve en betydelig omvæltning for byggebranchen. Branchen har traditionelt været præget af en filorienteret tankegang, men nu omhandler det selve dataopbevaringen. LD bygger på, at der ikke decideret skal udveksles filer, som tidligere, men, at data skal gemmes direkte i GDB'er. Et sådan skift vurderes nødvendigt for, at LD kan fungere (Pauwels , et al., 2021). Ud over dette nødvendige paradigmeskift er der en iboende begrænsning, som omhandler pålidelig lagring af detaljerede 3D geometriske data ved brug af LD-teknologi og herunder RDF-grafer. Dette er kritisk for en branche, som byggebranchen, hvor data i det fleste scenarier har tilknytning til et geometrisk objekt. *Pauwels, et al (2024)* understreger, at enten skal den geometriske udfordring løses for at gøre LD fuldt anvendeligt eller alternativt, må brugen af LD begrænses til ikke-geometrisk data, hvilket i sig selv er en stor begrænsning for fremtidens SWT. (Pauwels, et al., 2024).

4.7 RDF-grafer & Ontologier

SW kan anses for at være én stor RDF-graf (Perevalov, et al., 2022). BIM-modeller er typisk almindeligt tilgængelig i IFC, men modellerne kan lige så vel være tilgængelige i RDF-grafer. *Pauwels, et al. (2024)* understreger, at tilgængeligheden af RDF-grafer, og dermed LD, giver et bedre udgangspunkt for ACC sammenlignet med almindelige IFC-data på grund af dets bedre anvendelighed og logiske grundlag. Ved at konverterer BIM-modeller (IFC-data) til RDF-grafer, bliver det muligt, at denne kan bruges til at udføre data-queries, bl.a. ved brug af SPARQL, og dermed udføre avanceret søgning og analyse af BIM-modeller (Ismali & Scherer, 2017; Mohammed, et al., 2021; Ślusarczyk, et al., 2018). RDF-grafer er særdeles nyttige til at repræsentere og beskrive de komplekse relationer mellem bygningselementer og dataene i BIM-modeller (Ismali & Scherer, 2017). En SWT er som regel bygget ovenpå en RDF-graf, som grundlæggende beskriver entiteter, objekter, events, koncepter og deres indbyrdes relationer, hvor hver entitet identificeres med globale webidentifikatorer, som omtales IRI'er (Elshani, et al., 2023; Pauwels , et al., 2021). Disse gør det muligt at forbinde og integrere flere RDF-grafer (Elshani, et al., 2023). RDF-grafer betragtes som en måde at sætte data i kontekst ved at opbygge relationer og tilføje semantiske metadata (Pauwels , et al., 2021). Det betyder også, at RDF-grafer har et tydeligt potentiale i at forstå og tilgå komplekse og rige datasæt fra mange forskellige domæner (Ismali & Scherer, 2017). RDF-grafer har gennemgået aktiv udvikling i den seneste tid og anses for at udgøre en central bro mellem LD og slutbrugere bl.a. ved anvendelse af NLP til at generere strukturerede i RDF-queries (Gashkov, et al., 2022). Selvom GDB'er kan betragtes som et effektivt værktøj til at håndtere bygningsinformationer, er det stadig en udfordring. Brugere uden ekspertise i IFC og

GDB'er har udfordringer med at formulerer brugerspecifikke forespørgsler, da avancerede queries typisk kræver indsigt fra specialister (Ismali & Scherer, 2017).

Ud over at forbinde RDF-grafer inkluderer SWT også, tidligere nævnte, sprog såsom OWL til at berige data med semantik i form af ontologier (Elshani, et al., 2023). Brugen af en ontologi gør, at der kan søges efter meningsfulde og præcise resultater på kort tid og sikre, at søgningen giver mere relevante resultater baseret på ordenes betydning og relation, frem for specifikke nøgleord. IFC'en er også tilgængelig som OWL og kan konverteres til en RDF-graf, der er kompatibel med IFC-filen, hvilket gør det muligt at anvende de forskellige LD-sprog omtalt i *Afsnit 4.3* og 4.4 ifm. dataudtræk (Pauwels, et al., 2024). Det betyder, som belyst ovenfor, at andre IFC-filer kan kombineres med bygningsdata fra IFC-filen. En begrænsning ved IFC-filen er, at al geometri og visualisering er fuldt ud inkluderet heri, hvilket gør, at RDF-grafen, som konverteres pba. IFC-filen, bliver stor og kompleks (Pauwels, et al., 2024). For at overkomme dette og gøre bygningsdata og skemaer mere modulære og nemmere og bruge er der udviklet en række ontologier (Pauwels, et al., 2024). Ontologier er i stigende grad blevet anvendt til at validere forskellige typer data i byggebranchen (Elshani, et al., 2023). Ontologibaserede tilgange muliggør udviklingen af formelle og eksplicitte beskrivelser af domæneviden, hvilket kan bidrage til at sikre datakonsistens, fuldstændighed og nøjagtighed (Elshani, et al., 2023). *Nabiva, et al. (2023)* inddeler overordnet ontologier i to kategorier: (1) Generelle ontologier (2) domæneontologier. En domæneontologi beskriver hierarkisk essentielle begreber inden for et specifikt domæne og indeholder egenskaber for hvert begreb (Nabavi, et al., 2023). En generel ontologi relaterer sig til begreber, som anvendes i alle domæner, og disse kan føre til upræcise beskrivelser, hvis de anvendes inden for et specifikt domæne (Nabavi, et al., 2023). *Pauwels, et al. (2021)* belyser, at der er et behov for en bedre lagringsmodel, men hvor udgangspunktet stadig er den oprindelige kilde, hvor data er opstået. Dette er sammenfattet til tre centrale krav som består af: (1) Enkelhed (2) fleksibilitet til udvidelse (3) modularitet (Pauwels , et al., 2021). Det understreges af *Jackson, et al. (2019),* at ontologier er et afgørende element for behandlingen af data, men der forsat er mangel på værktøjer til videreudvikling af ontologier. Dette har medført, at aktører udvikler deres egne arbejdsgange, hvilket ofte resulterer i tidskrævende og ineffektive manuelle trin og ontologier uden systemer og med fejl ifm. senere anvendelse (Jackson, et al., 2019). *Liu, et al. (2017)* identificerer yderligere tre udfordringer ved anvendelsen af ontologibaserede metoder til dataudtræk, som samtidig opsummerer ovenfornævnte problematikker. (1) Der oprettes isolerede private ontologier, som modstrider tankegangen om fælles ontologier (2) Domæneontologier kræver, at den opbygges af eksperter, hvilket kræver ekstra arbejde og kommunikation og kan være besværligt (3) Eksisterende termer ændrer sig, og nye termer opstår over tid – især inden for BIM. Dynamiske ontologier kan have udfordringer med at følge med hurtige ændringer, mens eksisterende ontologier ofte har svært ved at opfylde de krav, der stilles (Liu, et al., 2017).

Schulz, et al. (2023) stiller yderligere krav til ontologier. Ontologier skal holdes små og modulære, da store ontologier der er skræddersyet til specifikke skemaer, som regel er vanskelige at genbruge til andre formål (Schulz, et al., 2023). Derudover oprettes mange ontologier for at bringe et nyt skema til SWT, hvor det i stedet burde undersøges, hvorvidt der er muligt at bygge ovenpå en eksisterende ontologi fremfor at oprette en ny (Schulz, et al., 2023). Samme artikel peger på BOT-ontologien, som en ontologi, der efterlever dette. BOT gør det muligt at repræsentere zoner og elementer. Zoner kan indeholde elementer, og elementer kan bestå af andre elementer. En zone kan f.eks. repræsentere en grund, bygning, etage eller rum. Dog understreger *Pauwels, et al. (2021)*, at det er nødvendigt at foretage nogle udvidelser af BOT-ontologien, hvilket omfatter følgende tre udvidelser: (1) Mere præcise semantiske definitioner af produkter og egenskaber (2) Reference til sensordata (3) Geometriske repræsentationer. Det er dog stadig styrken ved BOT-ontologien, at den er relativ simpel og alsidig (Pauwels , et al., 2021).

4.8 ML & LLM

Det er opstået en stigende interesse for at anvende AI til at forbedre valideringsprocessen (Teclaw, et al., 2024). AI anses bl.a. for at kunne afhjælpe nogle af de tidligere belyste udfordringer i *Afsnit 4.1 og 4.2*, men specielt også når det gælder validering af informationskvalitet og korrekthed (Teclaw, et al., 2024). ML er en tilgang til at processere et stort set rå data, og mange virksomheder og organisationer er engagerede i at implementere ML-metoder til at forbedre interne processer og systemer bl.a. ved at anvende og forstå NL (Anumba, et al., 2022; Wang , et al., 2022). De seneste bestræbelser, på at skabe sammenhæng mellem IFC-data og regulativer, har haft stor fokus på anvendelsen af ML til at udføre automatisering heraf. Her forsøges det at undgå hardcoding og manuelle regler, og i stedet anvende ML-modeller til automatisk at lære de underliggende semantiske og syntaktiske mønstre i både de regulatoriske dokumenter og IFC-dataene (Anumba, et al., 2022). Tidligere nævnte NLP, i *Afsnit 4.4*, har gjort betydeligt fremskridt i de seneste år, primært pga. udviklingen af LLM'er (Zheng & Fischer, 2023). LLM'er er designet til at forstå og generer menneskelignende sprog ved at blive pre-trained på enorme mængder tekstdata, og de har demonstreret imponerende evner til *in-context-læring* ved anvendelsen af NLP og tekstbaserede prompts. (Zheng & Fischer, 2023). Det er dog stadig udfordrende for LLM'er, såsom de velkendte GPT'er, at direkte omsætte en forespørgsel i NL til en struktureret query henvendt til et BIM-informationssystem, da ML-modellerne ikke er trænet til sådanne opgaver og på datastrukturen for BIM-modeller (Zheng & Fischer, 2023). ML og ML-forespørgselssystemer kræver store mængder tekstlig dialog data til træning, men sådanne træningsdata er begrænsede inden for byggebranchen (Wang , et al., 2022; Guo, et al., 2021; Issa, et al., 2022). Når det gælder regelkontrol og validering, kan et specifikt træningsdatasæt være nødvendigt. Det vil dog praktisk være yderst ressourcekrævende at oprette unikke

træningssæt for hvert enkelt regulativ. (Guo, et al., 2021). *Wang, et al. (2022)* understreger i denne sammenhæng at, om ét træningsdatasæt kan bruges til kontrol af flere regler og flere typer af regler vil kræve yderligere forskning. For at løse problemet med manglede træningsdata er der udviklet pre-trained LLM'er. Disse LLM'er er på forhånd trænet på store datasæt og kan anvendes i opgaver, som minder om den pågældende opgave, som en LLM skal løse (Issa, et al., 2022). Det reducerer behovet for store mængder træningsdata og sparer tid (Issa, et al., 2022). Derudover kræver træning af en LLM kraftfuldt hardwareudstyr, da både træning og test er tidskrævende processer, hvorfor mange virksomheder og organisationer har pre-trained deres AI-modeller, som kan genbruges til at løse lignende problemer (Issa, et al., 2022).

De tidligere nævnte RDF-grafer, i *Afsnit 4.6*, er en integreret del af mange ML-applikationer og tilbyder tit et SPARQL-forespørgselsinterface. Men eksisterende ML-værktøjer anvender ikke SPARQL-queries trods de åbenlyse fordele. Dette skyldes, at der er et misforhold mellem SPARQL og ML-værktøjerne ift. datamodel og programmeringsstil. ML-værktøjer arbejder med data i tabelformat, hvor SPARQL arbejder ud fra grafer og RDF-triples. I denne sammenhæng skelnes der mellem imperativ og deklarativ programmeringsstil, hvor ML-værktøjer er imperative, dvs. de arbejder ud fra direkte kommandoer, der fortæller præcist, hvordan noget skal udføres. SPARQL er derimod et deklarativt sprog, hvilket vil sige, at brugeren specificerer, hvad der ønskes opnået, hvorefter systemet automatisk afgør, hvordan dette skal udføres. I studiet af *Mohammed, et al. (2021)* bemærkes det, at ingen af de identificerede offentlige tilgængelige ML-værktøjer til RDF-grafer anvender SPARQL til at udtrække data fra RDF-grafer. Dette er bemærkelsesværdigt, da et sådant DB-system har klare fordele såsom datauafhængighed, deklarative forespørgsler og effektiv samt skalerbar forespørgselsbehandling. (Mohammed, et al., 2021).

4.9 Udviklede ACC-prototyper med AI, SWT, LD & BIM

Dette kapitel bygger på nøje udvalgte studier, der er identificeret gennem den systematiske litteratursøgning præsenteret i *Afsnit 2.2*. Udvælgelsen af studierne er foretaget pba. en grundig gennemlæsning, som er udført som en del af den systematiske litteratursøgning, med særlig et fokus på, at inkludere studier, der omfatter udviklede prototyper, og som på forskellige måder er relevante for undersøgelsen. Gennemgangen af de udvalgte studier har til formål at belyse designet og implementeringen af eksisterende prototyper inden for det pågældende område, herunder anvendte regelsprog, teknologier og software. Denne indsigt, sammen med resten af undersøgelsen, kunne bruges som inspiration til at udvikle en prototype i undersøgelsen.

Studiet af *Guo, et al. (2021)* arbejder med en semantisk tilgang til automatiseret kontrol af overholdelse af regler i byggeindustrien. Én af ACC-faserne i et ACC-

system indebærer forberedelse af BIM-modellen, og i denne prototype består fasen af at BIM-data konverteres til RDF-format. Disse data beriges derefter semantisk ved at integrere dem med ontologier. I regelfortolkningsfasen forklares, analyseres og udtrækkes regler fra regulatoriske dokumenter, og disse opbevares i en RDB. Her kan NLP og andre intelligente teknologier anvendes til regeludtrækning og danne logiske relationer mellem disse termer fra tekstbaserede regulatoriske dokumenter. Her gøres brug at TO og POS-tagging, som er benævnt i *Afsnit 4.5*. I studiet af *Zhang & El-Gohary (2023)* præsenteres en semantisk NLP-baseret informationsudtrækning fra regulatoriske dokumenter indenfor byggeriet til et ACC-system. Fremgangsmåden i studiet følger de samme faser og principper som studiet af *Guo, et al. (2021)*. Her forberedes f.eks. den rå tekstdata med bl.a. TO og *Sentence Splitting*, hvorefter der generes et sæt egenskaber, der beskriver teksten, hvilket bl.a. indebærer POS-tagging (*Bilag 4*). I andensidste fase i studiet af *Guo, et al (2021),* hvor regler eksekveres, bruges de semantisk berigede RDF-data og regel-DB'en til at genere og udføre queries i SPARQL og opstille regler i SWRL. I sidste fase, rapportering af regler, præsenteres reglerne som enten overholdt eller ikke-overholdt. Se *Bilag 2* for en visualisering af workflowet for prototypen.

Studiet af *Zheng & Fischer (2023)* præsenteres en dynamisk prompt-baseret *virtuel assistent*, der sigter mod at forbedre BIM-informationssøgning. Dette foregår via et interface baseret på NL og en visualisering af BIM-modellen. Prototypen består af tre overordnede moduler: (1) Et webbaserede brugergrænseflademodul (2) NLP-modul (3) Datastyringsmodul. Brugergrænseflademodulet giver brugeren mulig for at interagere med den virtuelle assistent og stille spørgsmål i NL og modtage respons i samme format samtidig med, at der vises en 3D visualisering af BIM-modellen. Når brugeren skriver en NL-forespørgsel, sendes dette til NLP-modulet, som består af to undermoduler: (1) En promptgenerator der skaber NL-prompts til at fortolke forespørgslerne vha. en ekstern GPT-server, som er trænet på massive tekstuelle data (2) En forespørgselsgenerator der strukturerer dataforespørgsler ud fra de fortolkede resultater og relevante oplysninger fra datastyringsmodulet. Datastyringsmodulet administrerer en cloud-baseret DB, der indeholder forbehandlet BIM- og NL-data bl.a. ved brug af en Revit API. (Zheng & Fischer, 2023). Se *Bilag 3* for en visualisering af workflowet for prototypen.

I studiet af *Zhong & El-Diraby (2024)* udvikles et projektspecifikt generativ *Question-answering-system*. Første del er databehandling, hvor data forberedes. Her konverteres tekniske dokumenter fra PDF til almindelig tekst, ikke-engelsk tekst og irrelevant indhold fjernes og sætninger opdeles. Derefter oprettes en VDB, hvor teksten fra første fase gennemgår TO og opdeles i små brudstykker, som hver transformeres til vektorer vha. domænespecifikke ordindlejringer. Vektoromdannelsen har den fordel, at den fanger diverse domænesemantikker vha. pre-trained LLM'er, hvilket forbedrer DB'ens dybde og kontekstforståelse. De små brudstykker opbevares

systematisk i en struktureret DB, hvilket gør det muligt at opnå hurtig og præcis adgang til information og giver kontekstuelt relevante responser. Næste fase er informationshentning. Når en bruger laver en forespørgsel, omdannes denne til *embeddings*, som er numeriske repræsentationer af teksten. Ud fra disse kan systemet identificere og udvælge relevante passager i DB ved at sammenligne embeddings og derved udvælge de vigtigste brudstykker til at besvare forespørgslen. Næste fase tilpasses LLM'en vha. en specialudarbejdet promptskabelon, som strukturer informationen af de vigtige udvalgte brudstykker. Til sidst kan brugeren stille spørgsmål og modtage præcise og velbegrundede responser, hvor flere datakilder kombineres. Her anvendes flere LLM'er, hvor: (1) Bedste og mest relevante data fremfindes (2) Data sammenlægges og opsummeres (3) Data forfines og tilpasses til konteksten. Således kan LLM'er arbejde effektivt og på store og komplekse datasæt. (Zhong & El-Diraby, 2024). Se *Bilag 5* for en visualisering af workflowet for prototypen.

I studiet af *Nabavi, et al. (2023)* udvikles en stemmebaseret assistent ved hjælp at NLP og Support Vector Machine (SVM) med henblik på at hjælpe ikke-tekniske brugere med at forespørge på information fra en BIM-model. SVM bruges til at bestemme brugerens type af forespørgsel fordi, at den har lært at genkende mønstre ud fra træningsdata, som den har fået til denne opgave. NLP anvendes ligesom i studierne af *Guo, et al (2021), Zhang & El-Gohary (2023)* og *Zhong & El-Diraby (2024)* til at analysere og forstå NL i brugeres forespørgsel. Der anvendes i dette studie diverse teknikker såsom, TO, POS, Stemming og N-Grams, som alt sammen hjælper med at forstå brugerens forespørgsel. I og med at der er forskellige synonymer og former for samme ord, bruges en NLP-teknik kaldet *Latent Semantic Analysis*, som bruges til at skabe vektorer, som repræsenterer hvert ords position i et flerdimensionelt rum, ligesom i VDB'en i studiet af *Zhong & El-Diraby (2024)*. Ydermere anvendes ontologier, såsom ifcOWL, som også bidrager med at udvide semantiske synonymer og relationer mellem termer og begreber. Derefter oprettes en DB med lister over objekter, materialer osv. fra BIM-modellen. Forespørgslerne foregår som plug-in i Naviswork, hvor en talegenkendelsesfunktion er indarbejdet (Google Web Speech API) således, at en bruger både kan udføre forespørgsler tekstuelt og stemmebaseret. (Nabavi, et al., 2023). Se *Bilag 6* for en visualisering af workflowet for prototypen.

Flere studier har undersøgt og udviklet prototyper baseret på den stemmebaserede tilgang. I studierne af *Shin, et al. (2022)* og *Shin & Issa (2021)* er der udviklet delvist ens to prototyper, som begge arbejder med automatisk stemmebaseret informationshentning fra BIM-software. I Studiet af *Shin & Issa (2021)* arbejdes der med prototypen BIMASR, som består af tre sammenkoblede moduler: (1) Først behandles den menneskelige stemme ved at konvertere den til tekstformat, så computeren forstår det (2) Her udføres syntaks og semantisk analyse af data, hvorefter det efterfølgende konverteres til SQL (3) Her knyttes BIM-data til et RDB-system (Oracle DB),

hvor SQL kan anvendes til forespørgsler. Til at omdanne menneskelig stemme til tekst anvendes Google Cloud Speech-to-Text API. I modul 2 udføres den syntaktiske analyse vha. NLP-teknikker, såsom TO og POS-tagging, og den semantiske analyse opnås ved at anvende ontologier. NL-forespørgsler kan i princippet udføres uden brug af den ontologibaserede semantiske analyse, men det vil kræve, at brugeren har kendskab til præcise BIM-termer og -kommandoer, når en NL-forespørgsel udføres. Når både den syntaktiske og semantiske analyse udføres, bliver det muligt for prototypens software at foretage selvstændige beslutninger, selvom der anvendes andre termer og kommandoer end BIM-specifikke. NL-forespørgslerne omdannes til SQL, som derefter kan interagere med BIM-modellen i RDB'en. BIM-data udtrækkes fra BIM-modellen, vha. Dynamo, og gemmes i et CSV-format, som lagres i RDB'en. Således kan en NL-forespørgsel, som omdannes til SQL, udtrække informationer fra en BIM-model vha. en RDB. (Shin & Issa, 2021; Shin, et al., 2022). Se *Bilag 7* for en visualisering af workflowet for prototypen.

I studiet af *Pauwels, et al. (2024)*, som tidligere har beskrevet de fire faser i et ACC-system, gennemgås den optimale fremgangsmåde for automatisering af en ACC-prototype baseret på SWT. Grundlæggende er hensigten at samle alle fire faser af et ACC-system i ét samlet system fordi, at det gør det mere simpelt for slutbrugeren, og manuelle handlinger undgås. Inputdata er den oprindelige BIM-model i IFC-format. Denne data konverteres til en RDF-graf, som kræver stabile ontologier for at fungere pålideligt. Der kører derefter et *Geometric Analysis Script*, bl.a. ved brug af Python, som analyserer geometriske aspekter i BIM-modellen, som derefter mappes til en RDF-struktur, der beriges med ontologier. Dette resulterer i en endelig RDF-graf. SHACL og ontologier bruges til at repræsentere krav og reguleringer. Her kan værktøjer, såsom Protégé, bruges til redigering af ontologier, og der findes diverse værktøjer til redigering af SHACL. Redigeringerne foregår ikke i det samlede system fordi, at det kræver en anden ekspertbruger end slutbrugeren til at udfører forespørgslerne. Derefter er de to sidste faser af et ACC-system, som indebærer, at regler eksekveres og rapportering af regler. Her tjekkes SHACL-regler op imod den endelige RDF-graf. Resultat præsenteres i en rapport, der dokumenterer, hvorvidt BIM-modellen opfylder de definerede krav. (Pauwels, et al., 2024). Se *Bilag 8* for en visualisering af workflowet for prototypen.

I studiet af *Zhou & El-Gohary (2021)* foreslås en metode til semantisk informationsjustering af BIM-modeller til computerfortolkelige regler vha. ontologier og DL. Denne metode inkluderer 4 faser: (1) Udtrækning af BIM-data (2) Udtrækning af data fra diverse regulativer (3) Semantisk informationsjustering (4) Evaluering. Ligesom i tidligere benævnte studier bliver BIM-modellen eksporteret til IFC, hvorefter IFC-entiteter kan udtrækkes og irrelevante informationer filtreres fra. Her bruges en JSDAI API, som gør det muligt at arbejde med IFC-data gennem en EXPRESS-baseret metode. Den regulatoriske data bliver udtrukket vha. en ontologibaseret metode, hvor

data repræsenteres ud fra ni semantiske informationselementer (SIE) såsom: (1) *Subject* (2) *Compliance checking attribute* (3) *Quantity restriction*. Hvor Subject referer til den primære enhed i et krav, *Compliance checking attribute* referer til en egenskab ved et *Subject* og *Quantity restriction* refererer til en begrænsning ved en Quantity Value. Her anvendes bSDD API og Protégé til at matche BIM-data og regulatoriske koncepter gennem standardiserede begreber fra bSDD og Protégé til at finde overordnede beslægtede begreber. Der sker herefter en semantisk informationsjustering, hvor BIM-data matches til de regulatoriske krav, repræsenteret i SIE'erne, hvilket består er flere processer. Først identificeres og matches BIM-objekter til regulatoriske objekter f.eks. et *Subject*. BIM-Properties matches til regulatoriske egenskaber såsom en *Compliance checking attribute*. Så anvendes der DL-modeller til at finde semantisk lighed mellem regulatoriske- og BIM-termer, og derefter udvælges de korrekte matches. Derefter organiseres de identificerede par i en graf, og disse linkes til de virkelige regulativer. Her bruges bl.a. Word2Vec-modellen, som repræsenterer ord som vektorer, og derved gør det muligt at analyserer termerne i BIM-data og regulatoriske krav for at finde semantiske ligheder. (Zhou & El-Gohary, 2021). Se *Bilag 9* for et eksempel samt en visualisering af metoden.

5 RESULTATER

Dette kapitel præsenterer de kvalitative empiriske data, der er indsamlet gennem de semistrukturerede interviews. Dataene er bearbejdet gennem meningskondenseringer for at fremhæve de vigtigste pointer samt præcisere betydningen og essensen af interviewpersonernes udsagn. Kapitlet relaterer sig til CD-trinene Contextual Inquiry og Interpretation Session, hvor data indsamles og anvendes til at identificere væsentlige aspekter af brugernes arbejdspraksis, som det nyt system skal understøtte. Kapitlet har til formål at give indsigt i brugernes nuværende arbejdspraksis og deres generelle holdninger til både deres arbejdsopgave og undersøgelsens emne. Disse resultater danner grundlag for den efterfølgende analyse, hvor dataene bearbejdes vha. Work Models for at udlede de brugerbehov og systemkrav, som det nye system skal imødekomme.

5.1 Meningskondenseringer

I dette afsnit præsenteres meningskondenseringerne foretaget pba. de udførte interviews. Interviewene blev optaget under udførelsen og er efterfølgende gennemlyttet for at sikre en præcis gengivelse af de indsamlede data. På baggrund heraf blev indholdet kondenseret og struktureret. Hver meningskondensering præsenteres særskilt for at give et klart og fokuseret indblik i det centrale indhold fra hvert interview.

Interviewperson A – Teamleder

Interviewperson A er teamleder i en afdeling, der arbejder med digital innovation. Teamet består af udviklere og forskellige typer af specialister, som støtter organisationen med både projekter og den digitale udvikling. Meningskondensering kan findes i *Bilag 10*.

A fortæller, at teamet er begyndt at undersøge, anvende og teste forskellige AI-løsninger. I den virksomhed, hvor A er ansat, fokuseres der på nuværende tidspunkt på de generative aspekter af AI, især i relation til tekstuelle input, samt AIs potentiale inden for dataanalyse. Kontrolprocesserne anses for at være tidskrævende, hvilket har økonomiske konsekvenser, beskriver A. Dette gælder både opsætning af kontroller, selve udførelsen og den efterfølgende opretning af fejl. Derfor fremhæver A vigtigheden af grundig projekteringsplanlægning, inden kontrollerne igangsættes. I organisationen anvendes SMC til kollisions- og konsistenskontrol. Ved opsætning af kontroller i SMC anvendes primært egne prædefinerede regelsæt. Disse regler er blevet udviklet og tilpasset over de seneste ti år og udgør forsat en central del af arbejdsprocessen. Reglerne er udarbejdet af specialister og bygger både på krav, informationer, regulativer og de erfaringer, der er tilegnet sig fra tidligere projekter.

Her fremhæver A en generel tanke om, at der er et behov for en digital *projekteringsassistent* i branchen. En sådan løsning vurderer A kunne hjælpe de projekterende

med at blive guidet til at vælge bedre og mere korrekte løsninger. A påpeger, at de nuværende kontroller foretages pba. IFC-datamodeller, der repræsenterer fagområdernes BIM-modeller og derfor det allerede udarbejdede og projekterede arbejde. Hertil mener A, at kontrol- og byggeprocessen kan forbedres, hvis fejl kunne opdages af de projekterende, mens modelleringen foretages. Det uddybes, at der i dag er fokus på kontrol mellem fagområder, men at teknologier, der kan assistere ved trække på viden og erfaringer, vil bidrage til bedre kontrolprocesser og dermed et bedre projekt. A mener, at resultatet herfra vil give kontrolprocessen mere værdi, hvis den udføres tidligere i *produktionskæden*, hvor fejl kan fanges og udbredes tidligere i processen.

Egenkontroller bliver yderligere diskuteret af A, som fremhæver, at der stadig er en vis uklarhed i organisationen om omfanget af de forskellige fagområders egenkontrol. A påpeger, at dette ikke er en optimal tilstand, da der mangler klare rammer for, hvad de enkelte fagdiscipliner skal kontrollere, før deres arbejde videregives. A forklarer, at samme problemstilling også er gældende i offentlige og fælles organisationer, hvor det forsøges at definere dette omfang. Der bemærkes, at nogle fagligheder, i højere grad end andre, har de nødvendige kompetencer og værktøjer til at foretage grundige egenkontroller. Dog skyldes manglerne i BIM-modellerne ofte sløvhed og faglige skyklapper. A understreger, at ukontrollerede modeller fra fagområderne kan have en negativ indvirkning på byggeprocessen, da mange arbejdsopgaver anvender modellerne som grundlag for udarbejdelse. Hvis dette grundlag ikke er tilstrækkeligt kontrolleret, vil det resultere i yderligere fejl, der senere i processen skal udbedredes, hvorfor fejl bør identificeres og håndteres allerede i projekteringen.

Efterfølgende understreger A, at branchen står over for en udfordring fordi, at krav og regler er spredt på tværs af flere organisationer og interessenter, hvilket gør det vanskeligt at indhente den nødvendige information. Derudover fremhæves det, at aktørernes tilgang til information i byggebranchen i stor udtrækning ikke har ændret sig hen over årene. Informationerne præsenteres forsat som døde dokumenter i form af bøger, PDF-filer mv.

Organisationen har udviklet interne ML-modeller, som kan håndtere større datamængder fra udbudsmateriale og give aktører mulighed for at fremsøge indhold ved anvendelsen af en LLM. Modellen assisterer ikke decideret modelleringen, men den informerer aktørerne om de krav, der skal overholdes i et specifikt scenarie ud fra et udbudsmateriale. A fremhæver dog en udfordring ved denne tilgang. Når ML-modellerne ikke nødvendigvis kan finde den korrekte respons, er de tilbøjelige til at give en forkert respons frem for intet. Der findes ingen arbejdsgang eller værktøj, som kan sikre, at modellen lever op til de påpegede fejl efterfølgende, hvilket må kontrolleres manuelt.

Interviewperson B – Linked Data specialist og produktudvikler

Interviewperson B har beskæftiget sig med, og specialiseret sig inden for, SWT og LD. Gennem årene har B undersøgt og udviklet systemløsninger, hvor potentialet i SWT og LD er udnyttet til innovative løsninger. I dag er B medstifter af en nystartet virksomhed, der i høj grad arbejder med databearbejdning gennem et samspil mellem AI, SWT herunder LD-teknologier. Meningskondensering kan findes i *Bilag 11*.

B forklarer, at en af fordelene ved LD er, at det ikke er nødvendigt at strukturere hele verden, som den er. I stedet er det muligt at pege på eksisterende elementer og bruge LD som limen til at binde dem sammen. Flere kunder har spurgt B, hvorfor det er nødvendigt at anvende og strukturer data efter LD-principper frem for RDB'er, som de er vant til. B forklarer, at forskellen ikke nødvendigvis er særlig stor, men at LD tilbyder en højere grad af fleksibilitet. LD består af en fleksibel datastruktur, der gør det muligt at modellere verden, som den fremstår. B understreger, at valget mellem en RDB og LD bør træffes pba. den type data, der skal håndteres efterfølgende. Hvis dataene blot omfatter én person med ét telefonnummer og én adresse, vil B foretrække en RDB, da denne type data nemt kan organiseres i tabelform. Hvis der derimod er tale om en situation, hvor der ønskes at modellere et scenarie med vægt på relationerne mellem data, er LD mere fordelagtigt. B pointerer, at RDB'er ofte kræver en forudbestemmelse af, hvordan data skal fremsøges i brugersammenhænge, men LD giver mulighed for en mere fleksibel efterbehandling. Selvom B fremhæver, at LD er en struktureret tilgang til data med stor fleksibilitet, bemærker B også, at nye teknologier som AI og vektorsøgninger giver mulighed for at arbejde med data uden nødvendigvis at have alt struktureret.

Der lægges vægt på, at der er en naturlig sammenhæng mellem AI og LD, hvor teknologierne kan understøtte hinanden. Hvis en graf beskriver objekter, subjekter og deres relationer, kan AI udlede resultater herfra. F.eks., hvis det er beskrevet, at *X* har en far, *Y*, og *Y* har en bror, *Z*, vil AI kunne udlede den næste implicitte relation, at Z er onkel til *X*, selvom dette ikke er eksplicit beskrevet i grafen. C fremhæver yderligere, at jo mere kontekst AI fodres med, desto bedre bliver modellens resultater. Denne kontekst kan LD tilbyde på grund af sin struktur. Derudover fungerer ML-modeller som en effektiv måde at interagere med data på, da modellerne kan omdanne NL til queries.

Organisationen, som B er medstifter af, anvender en kombination af SWT, LD og AI. De har udviklet en arbejdsgang og et system, hvor PDF-indhold opdeles i fragmenter. Disse fragmenter knyttes til den side, de tilhører, og denne side udgør et fragment af en specifik PDF-fil. Datene og relationerne gemmes i en RDF-graf. Fragmentet tildeles en *identifier*, som derefter lagres i en VDB. Når brugeren interagerer med datene gennem en ML-model, sikres det, at modellen leverer resultater, der angiver, hvor informationen stammer fra, hvilket giver validitet til responsen. B understreger, at

det ikke kan accepteres, at AI-modellen *hallucinerer*, når informationen skal bruges til at træffe beslutninger i et byggeprojekt. Produktet, som Bs organisation tilbyder, består af en række AI-modeller og services, der håndteres i en Cloud-løsning. Dette muliggør behandling af større datamængder, fordelt på flere instanser, frem for kun på en lokal computer. ML-modellerne, services og computerkraften lejes fra en ekstern virksomhed, hvilket medfølger, at flere services kan køre samtidig, da hver instans fungerer som en selvstændig computer. Hver service har sin specifikke funktion. En model trækker tekst ud, mens en anden håndterer billeder. Indholdet fra hver service bygges herefter som fragmenter i RDF-grafen. B anbefaler generelt, at den AI, der anvendes, er den, der bedst muligt understøtter den specifikke arbejdsopgave. Nogle opgaver løses med GBT, mens mindre opgaver kan håndteres fordelagtigt med gratis modeller som Llama.

B fremhæver vigtigheden af, at de, der udarbejder regulativer i Danmark, gør det i formater, der er computerlæsbare. På den måde bliver det muligt at udføre automatiske kontroller af bygningsmodeller baseret på krav. En udfordring, som påpeges, er, at krav og regler i regulativer ofte er uklart formuleret. F.eks. kan krav beskrive, at en specifik type af et element skal monteres i en betjeningshøjde, uden at betjeningshøjden er defineret. B påpeger, at hvis regulativer skal gøres computerlæsbare, kræver det, at reglerne først er klare, præcise og eksplicitte. Det vil ifølge B være ideelt, hvis reglerne fra de offentlige instanser var tilgængelige som SHACL, hvilket ville gøre det muligt at tjekke krav og regler i RDF-grafen, eller som IDS til validering af IFC-datamodeller og deres egenskaber. Dog understreges det, at der forsat er lange udsigter til, at dette scenarie vil blive en realitet. B foreslår fortsat, at BR udgiver regler formuleret som SHACL-regler og supplerer dem med en ontologi, der implicit inkluderer alle de egenskaber, som reglerne definerer.

Desuden fremhæves vigtigheden af at gøre disse teknologier let forståelige for brugerne i byggebranchen. Brugeren skal ikke nødvendigvis interagere direkte med modelgrafen, men skal i stedet tilbydes adgang til data gennem AI-modeller, der gør tilgangen enklere. Derfor spiller frontend og brugergrænsefladen en afgørende rolle for at sikre en god brugeroplevelse

Interviewperson C – BIM-manager

Interviewperson C arbejder som BIM-manager i en større rådgivende ingeniørvirksomhed. Ud over at varetage BIM-koordineringsopgaver er C også IKT-ansvarlig og delvist involveret i projekteringen af tekniske installationer. Meningskondensering kan findes i *Bilag 12*.

C beskriver, at der allerede ved projektopstart udføres kontrol af BIM-modeller. Disse kontroller retter sig mod overordnede aspekter, som sikrer, at modelleringsgrundlaget er korrekt udarbejdet, og at alle involverede parter arbejder ud fra de samme koordinater. Yderligere består denne tidlige kontrol af at sikre, at alle parter

eksporterer deres IFC-filer korrekt og med det aftalte dataindhold. Disse tidlige kontroller udføres gennem en visuel vurdering.

På Cs arbejdsplads udføres kontroller vha. SMC og påbegyndes allerede i dispositionsforslaget. Her understreges det, at der i denne fase primært tjekkes for mere simple forhold. Visse involverede aktører, der varetager specifikke videnområder og arbejdsopgaver, kræver data på et tidligt stadie i byggeprocessen. F.eks. nævner C, at arbejdsopgaver, der henvender sig til LCA-beregninger, økonomiske kalkulationer og energiberegninger kræver specifikke data og elementer. Derfor det er nødvendigt at sikre, at disse er til stede i IFC-datamodellerne. I projektforslaget, hvor koordineringen mellem de forskellige fagområder intensiveres, fortæller C, at kontrollerne bliver mere detaljerede og formelle, hvor fundne fejl oprettes som *Issues*, som dokumenteres og fremsendes til de ansvarlige.

Det understreges af C, at virksomheden ifølge Branchevejledningen er forpligtet til at kontrollere sit arbejde. Desuden foreslås det, at hvert fagområde påtager sig ansvaret for egenkontrol af deres BIM-modeller, både internt og i samspil med andre modeller. C forklarer, at de projekterende, for hvert fagområde, anvender indbyggede kontrolmetoder i deres interne modelleringsværktøjer til at identificere *Hard Clashes*. De tværfaglige kontroller udføres typisk af et mindre antal personer i organisationen, hvilket skyldes, at de ofte varetager store dele af projekteringsopgaverne ifm. et byggeprojekt. Hertil understreger C, at denne tilgang både giver økonomiske, kvalitetsmæssige og tidsmæssige fordele. C mener, at fordelene kommer af, at kontrolopgaverne er koncentreret om fagpersoner med specialisering indenfor området.

Det fremhæves, at der er to overordnede udfordringer ved kontroller af BIM-modeller. Den første udfordring er, at det stadig er en tidskrævende proces. F.eks. påpeger C, at det modtagne arbejde fra andre fagdiscipliner ofte kan være mangelfuldt. C anerkender, at dette i høj grad skyldes en utilstrækkelig kontrol af deres eget arbejde inden fremsendelse. Som et forsøg på at afhjælpe eller mindske denne udfordring afholdes der kontrolmøder med de involverede fagdiscipliner, hvor større mængder af ensartede fejl kommunikeres mundtligt. C understreger, at denne tilgang kan spare en masse tid, eftersom det er mindre omkostningstungt at behandle flere fejl og tendenser samlet i et åbent forum fremfor at håndtere individuelle og enkeltstående Issues gennem SMC.

Den anden problemstilling ved kontrol af BIM-modeller er, at både selve udførelsen og opsætningen af kontrollen i stor udtrækning er manuelle processer. Selvom der findes prædefinerede regler og templates, som kan anvendes som grundlag for kontroller i SMC, kræver det tilpasning til det specifikke projekt. C understreger, at projektkravene kan variere afhængigt af faktorer såsom aftalegrundlaget, bygningstypen, detaljeringsniveauet, bygherrekrav mm. C forklarer, at kontrollerne typisk

omfatter kollision- og konsistenskontrol. Visse opgaver kræver dog mere manuel indsats. F.eks. bliver brandegenskaber kontrolleret manuelt og stikprøvevist for at sikre, at de nødvendige data er til stede ved aflevering. C pointerer, at de ikke har kompetencer til at vurdere eller tjekke, om disse data er korrekte, men blot om de er tilstedeværende i BIM-modellerne. Nogle regler i SMC er baseret på kendte regulativer, men mange opgaver kræver stadig visuelle og manuelle tjek, da SMC vurderes til ikke at kunne håndtere alle typer regler med sine nuværende funktioner.

Senere i interviewet reflekterer C over, hvorvidt reglerne i SMC alligevel kunne tilpasses til samtlige regulativer. Det påpeges af C, at det må afhænge af de specifikke krav til afleveringen, hvilke interesser hver enkel aktør har og den tilgængelige tid til tilpasningen.

Interviewperson D – Ekspertisedirektør

Interviewperson D er ekspertisedirektør i en større rådgivende virksomhed med ansvar for at udvikle organisationens digitale kompetencer. Derudover er D aktivt medlem af flere nationale organisationer og initiativer inden for Bygge- og anlægsbranchen. D ser et stort potentiale i kombinationen af LD, AI og BIM. Organisationen har i løbet af årene forsøgt at udvikle prototyper og gennemføre testforsøg med systemer, der bygger på det potentiale, disse teknologier tilbyder. Særligt LD har haft organisationens opmærksomhed de seneste år, hvor det bl.a. har været anvendt til at udvikle et projekt og system kaldet BART. Projektet har givet mulighed for at undersøge store dele af området, men der opstod efterfølgende udfordringer med, at implementere idéen som en fast del af organisationens daglige praksis. D vurderer, at hovedårsagen til denne udfordring er, at LD for mange er en uvant måde at tilgå og forstå data på. Medarbejderne er mere fortrolige med at arbejde med data i tabeller, som i RDB'er, frem for at håndtere data som triples. Meningskondenseringen kan findes i *Bilag 13*.

Nogle af teknologierne anvendes i de nationale organisationer, som D er medlem af. Her bruges teknologierne til at kontrollere indholdet i filer, genkende filtyper og strukturer det tilgængelige indhold, primært vha. ML og ikke LD. D fremhæver behovet for flere personer med viden om LD og dets potentiale, især i kombination med ML, for at muliggøre udviklingen af brugertilpassede systemer til specifikke *use cases*, der er relevante for brugernes arbejdsopgaver. D påpeger samtidig, at der i byggebranchen er en tendens til at prioritere AI, mens LD skubbes i baggrunden. Ifølge interviewperson D beregner AI sin respons baseret på algoritmer for at give den rigtige respons, men LD kan i denne sammenhæng bidrage med mere præcise resultater fordi, at de data, der anvendes, er mere strukturerede. Derfor anses en kombination af de to teknologier som værende værdifulde.

D mener, at hvor BIM-modeller beriges med eksterne data, som f.eks. med dataene fra produktdatablade, vil LD kunne bidrage med at skabe en mere klar

sammenhæng. Samtidig understreges det, at LD dog ikke kan håndtere geometrier. D har vanskeligt ved at identificere præcist, hvor AI skal anvendes, men det pointeres, at AI kan automatisere visse standardprocesser og muliggøre sprogbehandlingen. Der ses derfor et potentiale i at bruge AI til at håndtere større datamængder og samtidig udtrække informationer fra LD på en mere brugervenlig måde.

Der ses ifølge D et stort potentiale i at gøre krav og regler computerlæsbare. Personen har bl.a. arbejdet med et projekt kaldet Helhedsorienteret Bygningsreglement, som handler om at forenkle BR, så det bliver mere computervenligt og regelbaseret. D mener, at oversættelse af regulativer til LD eller maskinlæsbare formater kan give mange nye muligheder. Det tilføjes, at AI ikke anses for at være den optimale løsning til denne type oversættelse, da LD tilbyder er en langt mere struktureret og præcis tilgang. Derfor at det er vigtigt at fastholde LD og dens relevans, selvom AI får mest opmærksomheden i branchen. Det er vigtigt at tydeliggøre, hvor og hvornår de to teknologier er stærke, samt hvilke typer information de hver især håndterer bedst. Det erkendes dog, at det kan være udfordrende at finde balancen i, hvordan teknologierne bedst anvendes både hver for sig og sammen. En god kombination kan give positive resultater, hvis deres respektive roller præcist identificeres. D uddyber, at for at teknologierne skal give mening i Bygge- og anlægsbranchen, skal de betragtes som en teknologistack, der understøtter hinanden. Denne stack bør bestå af LD, AI og BIM.

D diskuterer, hvorfor LD har haft succes i andre brancher, mens det ikke er blevet lige så udbredt i Bygge- og anlægsbranchen. D påpeger, at det kan skyldes, at data i branchen er ustruktureret. Dette rejser spørgsmålet om, hvorvidt branchen bør forsøge at skabe struktur i data eller, om det vil være mere gavnligt at behandle data i overensstemmelse med LD-principper for at skabe mere fleksibilitet. Det fremhæves, at byggebranchen ofte forsøger at skabe struktur gennem standardiserede klassifikationskoder for at sikre ensartethed. Hvis LD blev anvendt, ville det dog give brugerne mulighed for at navngive tingene, som de ønsker, hvilket kunne berige branchen. Det er imidlertid en udfordring for mange brugere at forstå og acceptere den frihed, som LD giver, fordi de ikke ser, hvordan data kan forbindes trods forskellige betegnelser.

Interviewperson E – Udvikler og rådgiver

Interviewperson E arbejder for en organisation, der udvikler materiale, som i Bygge- og anlægsbranchen accepteres som alment teknisk fælleseje. Materialet omfatter både krav og regler samt vejledninger, som branchen kan tilgå. E er ansvarlig for udvikling af digitale løsninger og rådgivning. En stor del af Es arbejde består i at hjælpe brugere med at forstå og finde den nødvendige information blandt organisationens materiale. Meningskondensering kan findes i *Bilag 14*.

Udfordringen, som organisationen dengang forsøgte at adressere, udsprang af, at brugerne manglede tid til at gennemgå en bog eller et blad for at finde et svar på deres problemstilling. Derfor efterspørger brugerne hurtige svar på deres spørgsmål. For at imødekomme dette behov har Es organisation arbejdet med at anvende LLM'er til at forstå store mængder tekst og har hertil udviklet en slags intern *Google*, der gør det muligt at søge på tværs af alt deres materiale. Målet med løsningen er at skabe et produkt, som gør det lettere for brugerne at få præcise og kontekstuelle svar hurtigt.

E forklarer, at traditionelle *keyword search-algoritmer* ofte ikke formår at forstå konteksten af en forespørgsel, hvilket kan resultere i upræcise svar. Derfor er der behov for en løsning, der både forstår, hvad der står i teksten, og hvad brugeren spørger om. E bliver spurgt om, hvorfor organisationen ikke anvender en LLM frem for en søgefunktion, hvor E svarer, at der er bevidste overvejelser bag dette valg. En chatrobot, der genererer tekst, udgør en risiko for at ændre eller fejlfortolke materialets oprindelige betydning, hvilket kan føre til potentielt misvisende rådgivning. Alt materiale er præcist og nøje beskrevet og er blevet løbende kontrolleret af mange fagpersoner. E understreger, at deres materiale der publiceres, er en del af Alment teknisk fælleseje og kan anvendes som dokumentation, idet den følger BR. Hvis en ML-model genererer en respons baseret på materialet, kan det give en forkert forståelse af indholdet især, hvis nuancer som forskellen mellem *bør* og *skal* ikke forstås eller medtages af modellen. I sådanne tilfælde bliver spørgsmålet, hvem der har ansvaret for potentielt misvisende rådgivning. E tilføjer, at en chatrobot altid vil forsøge at levere en respons, også selvom det er upræcist, og dette udgør en risiko for fejlinformation. Derfor prioriteres en løsning, hvor brugeren får nøjagtige og kontrollerede responser fremfor autogenereret tekst.

Herefter beskriver E, hvordan organisationens produkt fungerer. Systemet er baseret på en VDB, hvor tekst bliver konverteret til vektorer. Brugeren kan selv vælge søgeord og oprette forespørgsel, hvorefter systemet leverer de bedst mulige match. Denne tilgang gør brugeren til en aktiv del af søgeprocessen og præsenterer de ti bedste resultater baseret på forespørgslen, hvilket reducerer omfanget af materiale, brugeren skal gennemgå. Løsningen bevarer en grad af menneskelig vurdering. Brugerne, som stadig er eksperter, har ansvaret for at forstå og tolke den respons, systemet giver. På den måde støtter produktet brugerne, men fastholder dem som de afgørende beslutningstagere i at vurdere informationens relevans og præcision.

Da interviewpersonen bliver spurgt, om deres organisation har overvejet at udfase de traditionelle døde tekstdokumenter og i stedet skabe indhold i et computerlæsbart format, svarer E, at dette endnu ikke er blevet undersøgt nærmere, men at der kan være potentiale i det på længere sigt. E forklarer, at organisationen forholder sig til BR, hvor ikke alt kan omsættes direkte til regler eller praktiske anvisninger.

Derfor ser organisationen det som deres opgave at oversætte NL-regler til noget, der er mere forståeligt og håndgribeligt for brugerne.

Det fremhæves, at brugerne i Bygge- og anlægsbranchen ofte ønsker, at al nødvendig information kan tilgås samlet ét sted frem for at skulle søge i forskellige kilder. Dog ser E en væsentlig udfordring i, at store mængder af information er beskyttet bag betalingsmure. Dette vil nødvendiggøre nye forretningsmodeller for mange lignende organisationer, som arbejder med at fortolke regulativer og udarbejde vejledninger. E påpeger, at konkurrencemæssige hensyn kan gøre det svært at dele viden og dokumenter. Derfor kan ønsket om samlet adgang til informationer være vanskelige at opnå. Ikke udelukkende fra de tekniske barrierer, men også på grund af eksisterende forretningsmodeller og forskellige interesser på tværs af aktører.

6 ANALYSE

I dette kapitel analyseres de primære empiriske data, som blev præsenteret i *Kapitel 5*. De enkelte afsnit i kapitlet relaterer sig til CD-trinnet Work Modeling, hvor den indsamlede data konsolideres vha. metodens modeller og diagrammer. Kapitlets formål er at identificere og illustrere brugernes arbejdsstruktur og processer samt afdække de underliggende hensigter, strategier og motivationer. De udviklede modeller, som vil blive præsenteret, består af Workflow Modellen, Sekvensmodellen, den Kulturelle Model og Decision Point Modellen. Hver model bidrager med et unikt perspektiv på, hvordan brugerne udfører deres arbejdsopgaver og deres holdninger til disse. Den konsoliderede data danner efterfølgende grundlag for idegenereringen til et nyt system og dets funktionskrav, som skal hjælpe med at understøtte brugernes arbejdsopgaver.

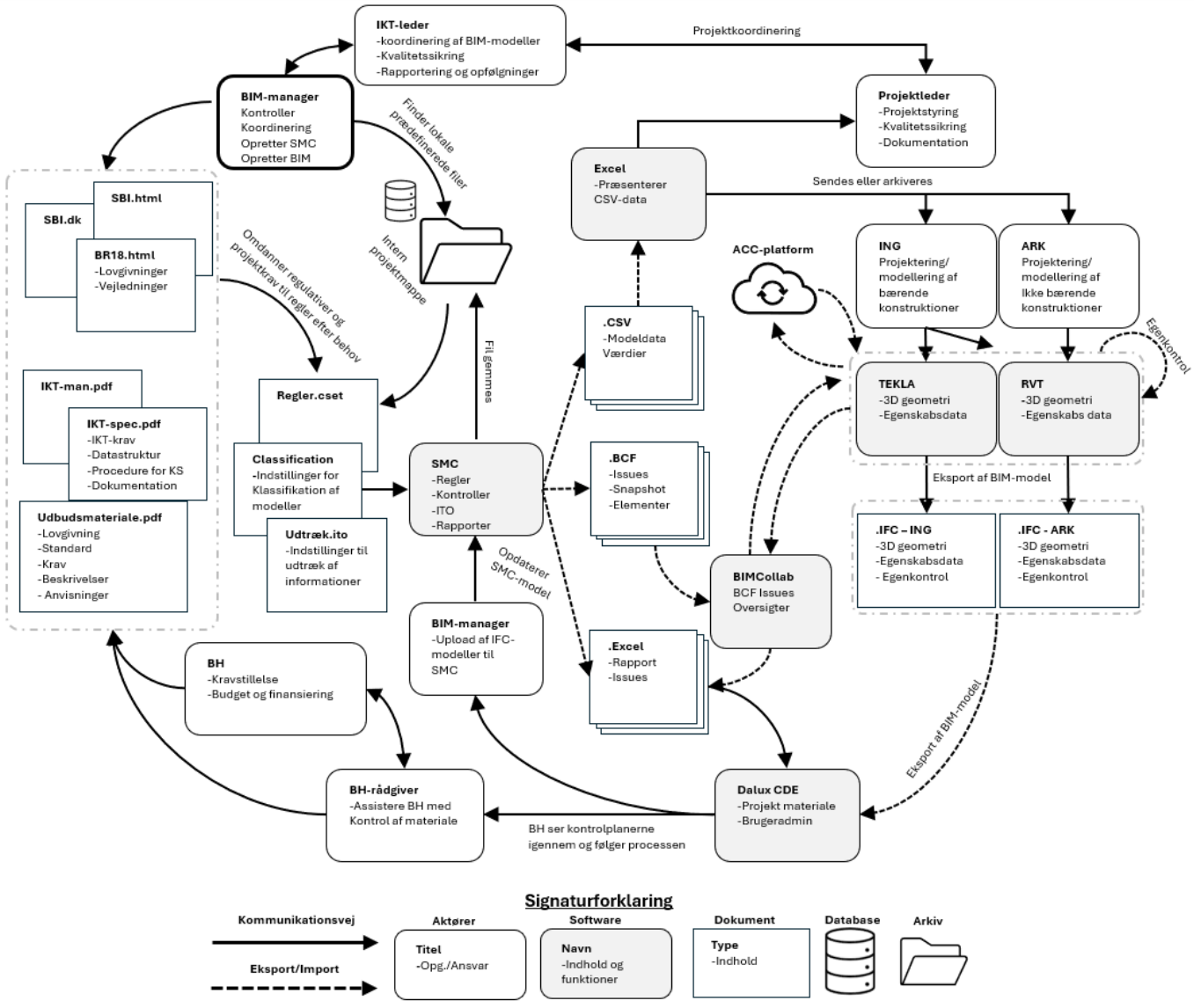
6.1 Workflow Modellen & Sekvensmodellen

Workflow Modellen er et værktøj der anvendes til at kortlægge og analysere komplekse arbejdsprocesser. Den bidrager til at skabe indsigt i, hvordan brugerne samarbejder, ved at fremhæve de roller og ansvarsområder, de påtager sig under udførelsen og koordineringen af arbejdsopgaverne. Modellen hjælper med at indfange koordineringen og kommunikationen mellem brugerne ved at fremhæve de kommunikationsstrømme, der foregår i arbejdsprocessen. Ved at bruge modellen til at belyse, hvordan arbejdsopgaver fordeles mellem involverede aktører, og hvordan disse roller interagerer, er det muligt at identificere essentielle detaljer om samarbejdet og strukturen arbejdsopgaven er underlagt. (Holtzblatt & Beyer, 2017; Holtzblatt & R. Beyer, 2005). Herfra er det muligt at identificere, hvor og hvilke processer i det samlede workflow der kan optimeres, og hvor optimering vil have størst effekt.

Dette afsnit indledes med en præsentation af Workflow Modellen, hvorefter Sekvensmodellen bliver gennemgået. I undersøgelsens analyse fungerer Sekvensmodellen som et forlængende og understøttende element til Workflow Modellen. I og med at der i Workflow Modellen er identificeret områder med uhensigtsmæssigheder og potentiale for forbedring og optimering, anvendes Sekvensmodellen til at præsentere disse afgrænsede områder mere detaljeret.

På *Figur 12* nedenfor vises den udarbejdede Workflow Model. Modellen illustrer den arbejdsproces, som brugerne i undersøgelsen gennemgår for at fuldføre opgaven af kontroller af BIM-modeller i et byggeprojekt. Figuren viser de involverede aktørers roller og ansvar samt de indbyrdes kommunikationsstrømme. Derudover illustrer modellen, hvordan typer af software og dokumenter indgår i disse kommunikationsstrømme. Dette fokus skyldes, at arbejdsopgaven og den tilhørende proces i høj grad bygger på kommunikation og koordinering, hvor software og dokumenter spiller

en central rolle. Derfor er der identificeret, hvilke funktioner og indhold de forskellige software og dokumenter bidrager med i processen.



Figur 12 – Workflow Model for nuværende arbejdsproces

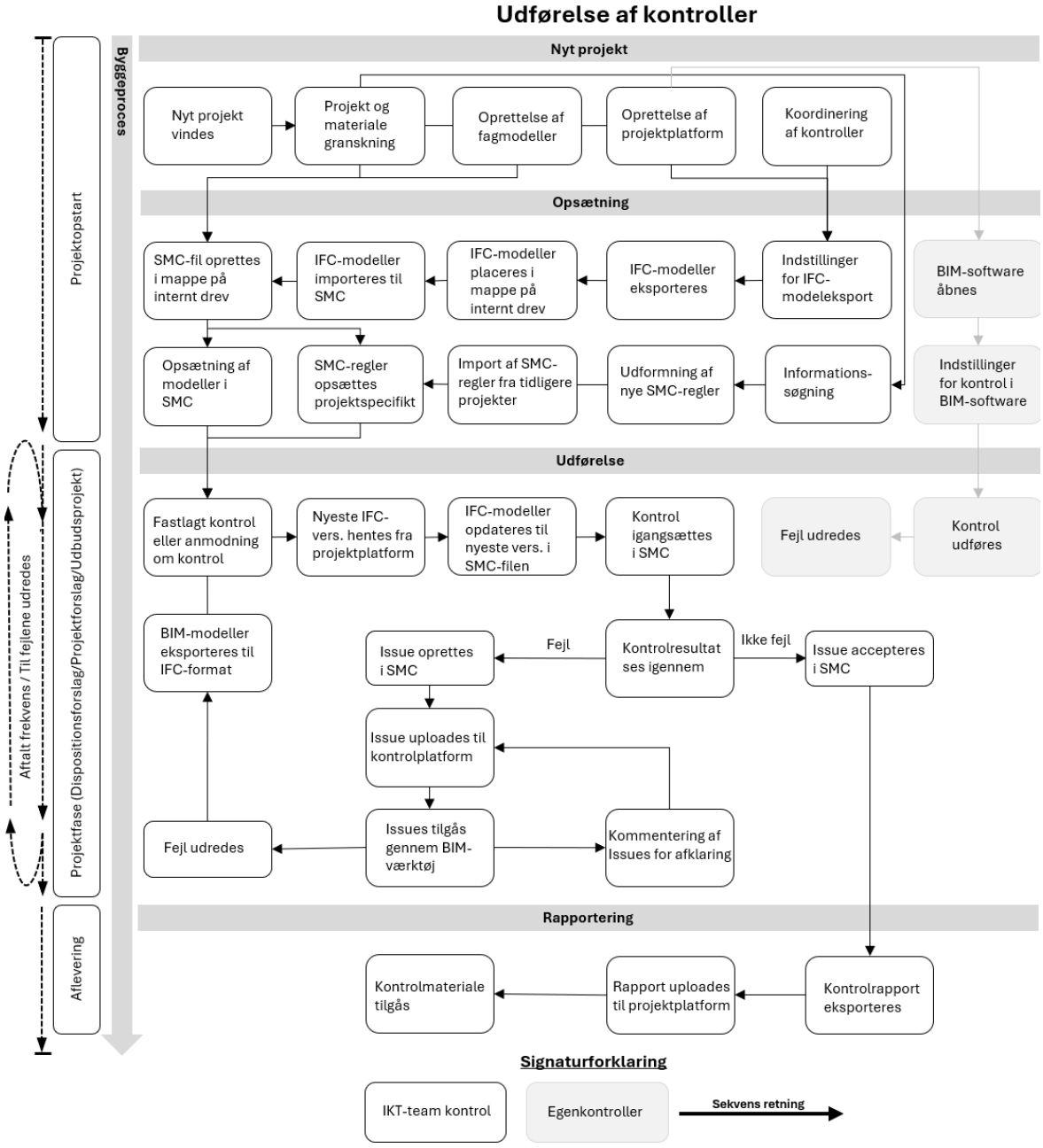
Som illustreret på Workflow Modellen har IKT-lederen det overordnede ansvar for den digitale koordinering. I tæt samarbejde med BIM-manageren sikrer IKT-lederen et effektivt digitalt samarbejde. BIM-manageren har medansvar og bistår IKT-lederen med at skabe og opretholde dette samarbejde. Blandt BIM-managerens opgaver er opsætning og udførelse af kontroller, hvilket kræver et dybdegående indblik i projektmaterialet udarbejdet af bygherre og bygherrerådgiver. Dette skyldes, at projektmaterialet, som typisk foreligger i PDF-format, beskriver bygherres ønsker og krav samt de regulativer og standarder, som rådgiverteamet kontraktuelt er forpligtet til at overholde. Efter en grundig gennemgang af projektmaterialet opretter BIM-

manageren en SMC-fil, som gemmes i IKT-teamets interne projektmappe, hvorefter opsætningen af kontroller i SMC påbegyndes. SMC anvender primært tre filtyper: (1) *CSET-filer* der indeholder reglerne for kontrollerne (2) *ITO-filer* som bruges til informationsudtræk (3) *Classification* som bruges til at kategorisere bygningskomponenter efter brugerbehov. BIM-manageren er hertil ansvarlig for at oprette eller redigerer disse filer således, at kontrollerne, der udføres i SMC, stemmer overens med projektets krav og standarder. BIM-manageren importerer først manuelt virksomhedens prædefinerede filer efterfulgt af relevante filer fra tidligere projekter, som kan genanvendes, og afslutningsvis udarbejdes filer baseret på projektets specifikke behov. Projektspecifikke SMC-filer, som BIM-manageren kan være nødsaget til at udarbejde, kan f.eks. være bygherrekrav, anvisninger eller regulativer, som skal omdannes til SMC-regler for at kunne være en del af kontrollen.

De projekterende aktører, såsom arkitekter og ingeniører, modellerer deres arbejde i deres respektive BIM-software og synkroniserer dagligt deres ændringer til centralfilen på projektets Cloud-platform. Her anvender IKT-teamet platformens funktioner til automatisk at publicere BIM-modellerne, eksportere dem til IFC-format og lagre dem på platformen med en aftalt fast frekvens. BIM-manageren downloader derefter IFC-filerne fra Cloud-platformen til sin lokale computer og uploader dem evt. til andre projektplatforme, hvor alle involverede aktører kan få adgang til dem. Opsætningen af SMC-filen, men også den senere udførelse af kontrollen, indebærer yderligere import af projektets modeller i IFC-format, da SMC ikke understøtter de filformater, som ingeniører og arkitekter anvender i deres projekteringsværktøjer. Der er ingen direkte forbindelse mellem SMC-filen, der er gemt på IKT-teamets interne drev og Cloud-platformen, hvor IFC-datamodellerne er lagret. Derfor sørger BIM-manageren, enten manuelt eller via en automatisk løsning, for at eksportere de nyeste BIM-modeller til IFC-format og gemme dem i den interne mappestruktur, inden kontrollen udføres. Denne proces gentager BIM-amageren hver gang der skal udføres en ny kontrol af projektet.

Som det fremgår af Workflow Modellen udføres to typer kontroller: (1) Den overordnede kontrol som IKT-teamet har ansvaret for (2) Egenkontrollen som de projekterende i de enkelte fagdiscipliner selv udfører i det BIM-software, der anvendes internt. Workflow-Modellen kan gøre det svært at afkode, hvornår og hvordan disse kontroller udføres, samt hvilke trin i arbejdsprocessen der er nødvendige for at gennemføre dem. Sekvensmodellen er valgt som supplement til Workflow Modellen for at detaljere og tydeliggøre dette. Sekvensmodellen fungerer grundlæggende som en task-analyse, der illustrerer, hvordan delaktiviteter udføres, og i hvilken rækkefølge de indgår i processen mod at fuldføre arbejdsopgaven (Holtzblatt & Beyer, 2017; Holtzblatt & R. Beyer, 2005). Med denne detaljerede indsigt og forståelse er det muligt at bruge Sekvensmodellen til at optimere udviklingen af det nye system og sikre, at løsningen understøtter brugernes behov og mål ifm. arbejdsopgaven.

Den udarbejdede Sekvensmodel ses nedenfor i *Figur 13*. Her vises det, hvordan IKT-teamet udfører den store og tværfaglige kontrol, og hvordan de enkelte fagdiscipliner udfører deres egenkontroller. Modellen adskiller sig fra CDs traditionelle Sekvensmodel ved at tilføje en ekstra dimension, der højere grad belyser arbejdsopgaven i relation til byggeprocessen og arbejdsfrekvenser. Dette valg er baseret på den indsamlede empiri, som understreger byggeprocessens centrale betydning for arbejdsopgaven og dens udførelse.



Figur 13 - Sekvensmodel

Sekvensmodellens grå bjælker opdeler delaktiviteterne i overordnede procesområder. Til venstre for bjælkerne ses en lodretgående grå pil, der angiver, hvornår aktiviteterne udføres undervejs i relation til byggeprocessen, og hvilke aktiviteter der gentages. Delaktiviteterne er illustreret som firkantede kasser og repræsenterer den konsoliderede data. Kasserne er yderligere inddelt i kontroltyperne: (1) IKT-teams kontrol (2) Egenkontrol. Procesområderne er inddelt i følgende faser: (1) Nyt projekt (2) Opsætning (3) Udførelse (4) Rapportering. Delaktiviteternes udførelsesmæssige afhængigheder er angivet med pile, der viser sekvensretningen. Hvor delaktiviteter kun er forbundet med en linje uden en retning, betyder det, at aktiviteterne kan udføres samtidig. Som det fremgår af modellen, påbegyndes arbejdsopgaven allerede i projektopstarten, hver gang et nyt projekt igangsættes. Fasen, Opsætning, dækker de aktiviteter, der gentages og færdiggøres for hvert nyt projekt for at skabe fundamentet og rammerne for kontrolarbejdet.

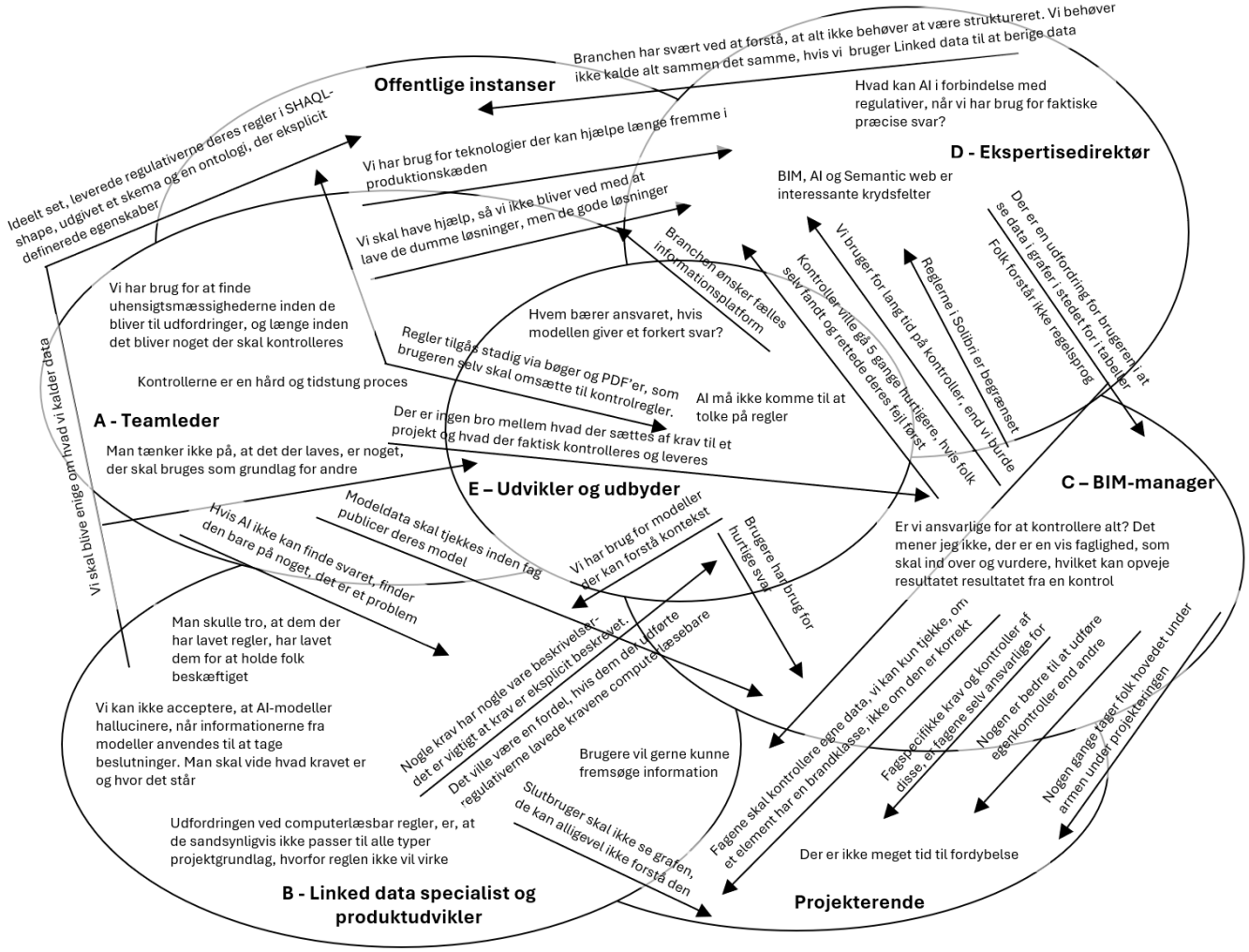
Fasen, Udførelse, referer til delaktiviteter, som skal gennemføres for at kunne eksekvere kontrollerne. Her skal det bemærkes, at delaktiviteter udføres gentagende gange i alle projekteringsfaser. Udførelsen påbegyndes enten som følge af den aftalte kontrolfrekvens eller grundet en anmodning fra projektaktører. Kontrolfrekvensen varierer typisk fra én gang om ugen til én gang i måneden. Årsagen til at disse delaktiviteter gentages, inden en kontrol kan udføres, er, at kontrollen skal tage udgangspunkt i det nyeste modelgrundlag. Dette gør det muligt at identificere både aktuelle fejl og fejl, der er blevet udbedret siden den seneste kontrol. Det endelig mål med arbejdsopgaven er at udrede fejl og dokumentere kontrolprocessen, hvilket sker gennem afrapportering af arbejdsopgavens udførsel ved Aflevering.

Sammenlignes IKT-teamets tværfaglige kontrol med projekternes egenkontrol, fremgår det, at den tværfaglige kontrol er betydelig mere omfattende. Derudover er de projekterende afhængige af IKT-teamet for at finde tværfaglige fejl og for at få adgang til kritiske informationer om BIM-modellerne.

6.2 Den Kulturelle Model & Decision Point Model

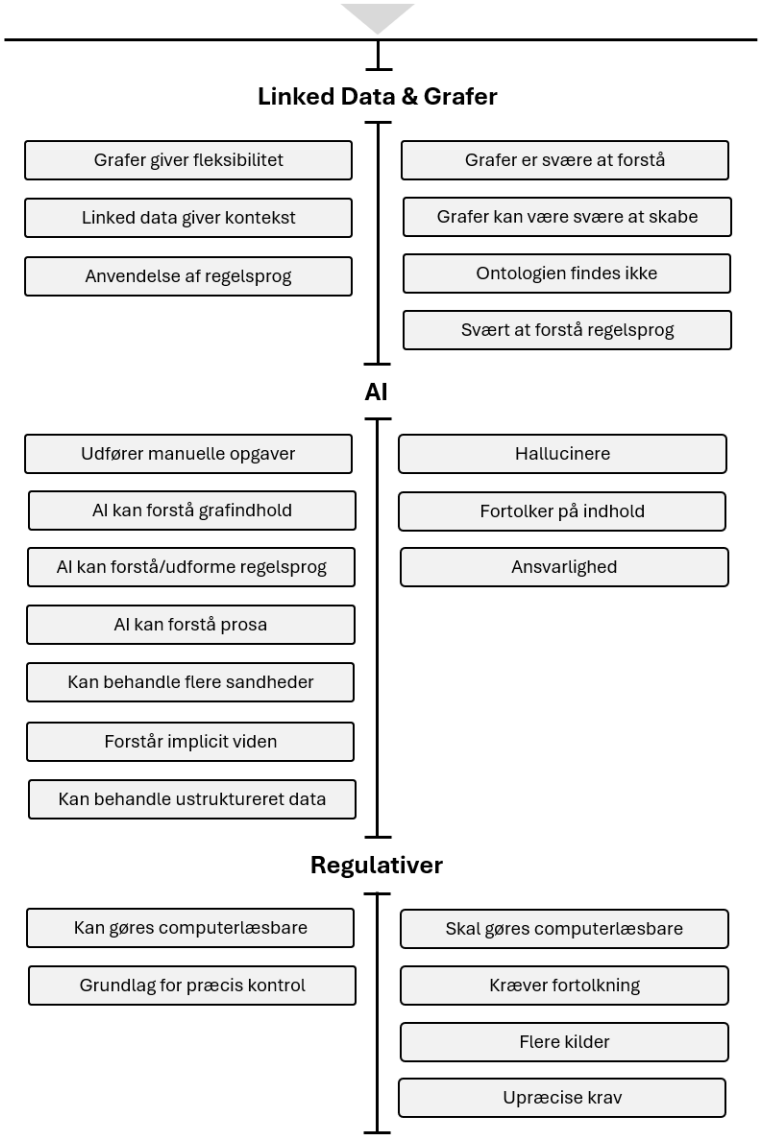
Den Kulturelle Model skildrer de kulturelle påvirkninger, der har indflydelse på brugerne, og giver indsigt i det kulturelle miljø, hvor det nye system skal implementeres. Modellen tager højde for både formelle og uformelle kulturelle faktorer, såsom juridiske påvirkninger, dynamikker og værdier. Den afspejler desuden de begrænsninger brugerne oplever ifm. udførelsen af arbejdsopgaven. (Holtzblatt & Beyer, 2017; Holtzblatt & R. Beyer, 2005). I denne undersøgelse anvendes modellen til at skabe et overblik over byggebranchens og dermed brugernes synspunkter, frustrationer og holdninger til arbejdsopgaven og til undersøgelsens emne. Ved at identificere og forstå disse faktorer er det muligt at udlede væsentlige brugerbehov og systemkrav, som skal indarbejdes for at sikre en succesfuld systemudvikling.
Som førnævnt fokuserer modellen ikke kun på kontroller, men også generelt på de

teknologier som ligger til grund for undersøgelsen i form af SWT, LD og AI. Dette skyldes, at det forud for interviewene blev vurderet væsentligt at forstå brugernes viden om og holdninger til disse teknologier for at sikre en vellykket implementering af dem i det nye system. Den Kulturelle Model er illustreret nedenfor i *Figur 14*.



Figur 14 – Kulturel Model udført med inputs fra Afsnit 5.1

Boblerne i modellen repræsenterer interviewpersonerne mfl., mens pilene med tilhørende tekst stammer fra den indsamlede empiriske data. Retning på pilene indikerer de kulturelle påvirkninger, og hvem de er henvendt til. Det kan være vanskeligt at danne et fuldkomment overblik over modellen, hvorfor Decision Point Modellen anvendes til at overskueliggøre indholdet og konsolidere dataene yderligere. Modellen er en variant af den Kulturelle Model, men den belyser bedre de faktorer, der påvirker mennesker i at gå mod en retning eller tage en bestemt beslutning (Holtzblatt & Beyer, 2017; Holtzblatt & R. Beyer, 2005). Decision Point Modellen er særlig anvendelig i tydeliggørelsen af de vigtigste faktorer, og derfor hvilke der særligt skal tages i betragtning til udviklingen af det nye system. Decision Point Modellen ses på nedenstående *Figur 15*.



Figur 15 – Decision Point Model uddraget fra Sekvensmodel.

Modellen illustrer, hvorvidt interviewpersonerne vurderer, at SWT, LD og AI er teknologier, der fordelagtigt kan implementeres i deres arbejdsopgave med kontrol af BIM-modeller. På venstre side præsenteres de faktorer, der understøtter brugernes vilje til at adoptere teknologierne og deres opfattelse af dem. På højre side vises derimod de faktorer, der udgør barrierer for teknologiernes anvendelse. Ved at analysere og tilpasse det nye system til at adressere de identificerede udfordringer, kan brugernes behov imødekommes mere effektivt. For at skabe en overskuelig struktur er faktorerne på både venstre og højre side organiseret i overordnede begreber.

6.3 Identificering af brugerbehov og systemkrav

Næste trin i CD-processen efter Work Modeling er Visioning, hvor de konsoliderede data fra foregående trin anvendes til at udvikle en vision for, hvordan et nyt system

kan bidrage til at rationalisere brugernes arbejdsopgave og det eksisterende workflow (Holtzblatt & R. Beyer, 2005). Visionen præsenterer en fortælling om, hvordan brugeren fremover vil udføre sit arbejde (Holtzblatt & Beyer, 2017). For at udforme visionen og identificere systemkrav og brugerbehov anvendes både de konsoliderede data samt viden fra *Kapitel 4 og 5*, som er sammenfattet i en komparativ analyse. Således anvendes både det videnskabelige grundlag i rapportens undersøgelse i en kombination med holdninger og processer fra erhvervslivet. Den komparative analyse resulterer i en samlet liste over identificerede brugerbehov og systemkrav, som udgør fundamentet for redesignet af det eksisterende workflow med det nye system og samtidig danner rammen for visionen for systemet.

Komparativ analyse af teoretisk og empirisk data

Interviewperson A efterspørger først og fremmest en digital projekteringsassistent, som kan give brugerne hurtig respons på deres spørgsmål. Med udgangspunkt i Workflow Modellen og Sekvensmodellen identificeres to grundlæggende problematikker, som begge stiller krav til systemet. Først identificeres det i Workflow Modellen, at en BIM-managers indledende opgaver er at få et dybdegående indblik i projektmaterialet, hvilket kan være vanskeligt i og med, at dette typisk foreligger i PDF-format. Dette udgør første problematik selvom, at det muligvis ikke opleves som en problematik for brugeren. Dette belyses også i *Afsnit 4.1*, hvor det i studiet af *Anumba, et al. (2021)* beskrives som en proces, der ikke er særlig effektiv grundet manuelt arbejde. Interviewperson A fremhæver i forlængelse heraf, at aktørernes tilgang til information i byggebranchen i stor udtrækning ikke har ændret sig hen over årene, men primært anvender døde dokumenter såsom PDF-formatet. Det er altså et indlysende krav, at systemet skal kunne håndtere døde formater, hvor brugerne på en nem og effektiv måde kan fremsøge information i et ellers omfattende projektmateriale. Den anden problematik, som identificeres i Work Models, er, at der er mange manuelle processer forbundet med opsætningen og udførelsen af kontroller gennem byggeprocessen. Dette afspejles bl.a. i brugernes håndtering af projekter med SMC, hvor regler udarbejdes manuelt pba. omfattende granskning af information angivet i projektmaterialet. I *Afsnit 4.4* påpeges både muligheder og begrænsninger ved anvendelsen af SMC. Overordnet konkluderes det i samme afsnit i undersøgelsen af *Nuyts, et al. (2024),* at SMC, sammen med andre software og sprog, kan definere alle fem regler og krav, som studiet arbejder ud fra. Ét af problemerne med de software som anvender IFC-data som datainput er, at de ikke kan behandle andre formater. Derudover er SMC begrænset til kun at arbejde inden for eget software, hvilket i sig selv udgør en barriere for softwarets anvendelse i et potentielt ACC-system. Det vil derfor, ift. systemkrav og eksisterende workflow, kræve at andre processer og software anvendes frem for SMC. I forlængelse heraf ser diverse interviewpersoner, specielt Interviewperson B, D og E, samt diverse studier inkluderet i *Kapitel 4*, et stort potentiale i en kombination af SWT, LD, AI og BIM. Der er bred enighed om, at disse

teknologier kan understøtte hinanden positivt og bl.a. imødekomme udfordringerne med håndteringen af de døde formater og den ustrukturerede projektdata, hvilket kan bidrage til nem informationssøgning og validering af BIM-data. Derfor skal disse teknologier anvendes i det nye system.

De mange manuelle processer i Workflow Modellen er standardprocesser, som ifølge Interviewperson A kan automatiseres vha. AI. AI består af diverse teknologier, hvor nogle er mere omtalt end andre ifm. ACC f.eks. LLM'er. Interviewperson B anbefaler først og fremmest at anvende den type AI, der bedst muligt understøtter den specifikke arbejdsopgave. Vedkommende foreslår brugen af enten open source-modeller eller AI-modeller fra konventionelle udbydere, da disse ofte er fine-tunet på store datasæt. F.eks. arbejder studiet af *Zheng & Fischer (2023)* med en promptgenerator, der skaber NL-prompts til at fortolke forespørgslerne vha. en online LLM, som er trænet på massive tekstuelle data. Det samme foregår i studiet af *Zhong & El-Diraby (2024)*, hvor pre-trained modeller anvendes til at fange domænesemantikker. Der identificeres samtidig nogle problematikker, som systemet skal kunne håndtere, når det kommer til LLM'er. Den væsentligste begrænsning, som der er generel enighed om blandt interviewpersonerne og studierne i *Kapitel 4*, er, at ML-modeller kan hallucinere. Dette fremgår også i Den kulturelle Model og Decision Point Modellen som væsentlige udfordringer, der kan have direkte indflydelse på, om brugerne vil anvende systemet. Derfor er det et klart systemkrav og brugerbehov, at dette undgås i udviklingen af systemet. Flere studier anvender ML-modeller til automatisk at identificere og forstå de underliggende semantiske og syntaktiske mønstre i både de regulatoriske dokumenter og IFC-dataene. Derfor bliver anvendelsen af LLM'er relevant for udviklingen af systemet.

Ydermere påpeges i *Afsnit 4.3* én af hovedudfordringerne i ACC. Udfordringen er at tilpasse semantikken i BIM-modellerne, i formatet IFC, med regulativernes semantik, i NL således, at udførelsen af overensstemmelseskontrol mellem BIM-modellen og reglerne muliggøres. Denne udfordring bliver noget systemet skal kunne håndtere. Her efterspørger interviewperson A, at anvendelsen af IFC til ACC udgår, og at kontrollen foregår direkte fra BIM-modellen imens, at modelleringen foregår. En anden problematik, der understøtter dette, er nævnt i *Afsnit 4.5*. Mange af de eksisterede tilgange til ACC er ikke i stand til at behandle NL, hvorfor det kræver at brugeren har specifik viden om de præcise kommandoer og termer, der bruges i BIM-software, og hvordan disse relatere sig til termer og begreber i regulativerne og projektmateriale. Her opstår et brugerbehov pointereret af *Shin & Issa (2021)* om en platform til udtræk af data, der semantisk kan forstå en brugeres spørgsmål. Diverse studier og allerede udviklede prototyper, opsummeret *Afsnit 4.9*, peger i denne sammenhæng på anvendelsen af NLP, herunder TO og POS-tagging, som essentielle værktøjer til at forespørge og få respons i NL ifm. ACC. Anvendelsen af NLP optræder derfor som et systemkrav for at gøre det nemmere og mere forståeligt for

brugeren og anvende det endelige system. Interviewperson B understreger at brugeren skal interagere med en RDF-graf, men adgangen skal tilbydes gennem AI-modeller. For at sikre god brugeroplevelse stiller dette store krav til frontend og brugergrænsefladen i det endelige system. Dette stemmer overens med den platform, som foreslås af *Shin & Issa (2021),* hvor frontenden i det kommende system vil skulle kunne håndtere en brugers spørgsmål. I *Afsnit 4.7* understreger *Pauwels, et al. (2024)*, at tilgængelighed af RDF-grafer og dermed LD giver et bedre udgangspunkt for ACC sammenlignet med almindelige IFC-data, hvorfor anvendelsen af RDF-grafer indgår som et systemkrav. Der efterspørges et fokus på at give en mere direkte og softwareneutral adgang til bygningsdata. Ifm. med RDF-grafer identificeres diverse fordele ved anvendelsen af ontologier og specifikt domæneontologier. Fordelen ved en ontologi er, at den muliggør søgning efter præcise og mangelfulde resultater på kort tid samtidig med, at den sikrer mere relevante søgeresultater. Det indgår derfor også som systemkrav, at domæneontologier, såsom BOT, skal anvendes i udviklingen af systemet. Det kræver dog diverse udvidelser af ontologierne, hvor de bl.a. skal tilpasses regulativerne indhold, som ACC arbejder ud fra.

Der er tidligere beskrevet en problematik om døde dokumenter i NL, hvilket skaber en begrænsning ift. automatisering og ACC. Derfor er et af de vigtigste systemkrav, at systemet skal kunne håndtere disse dokumenter. Dokumenterne er visualiseret i Workflow Modellen, på *Figur 12*, og består bl.a. af udbudsmateriale, IKT-specifikation, IKT-manual, projekteringsmateriale og meget mere. Derudover eksisterer der også døde dokumenter i form af regulativer såsom BR18 og alment teknisk fælleseje. Angående de døde dokumenter påpeges det af interviewpersonenerne, at data og projektkrav i byggebranchen generelt er ustruktureret og tvetydigt formuleret. Disse kravs karakteristika omtaler *Pauwels, et al (2024)* bl.a. som *Vague Constraints* og *Computationally Constraints*, hvilket er krav der er svære at kontrollere og anvende i et ACC-system. Det medfører et behov om at regler skal være klare, præcise og eksplicitte. I *Afsnit 4.1* forslås en tilgang, i studiet af *Ghannad, et al. (2019),* til formalisering af regler, hvor en samlet enhed er i stand til at indfange alle elementer af regler, semantikker, ontologier og logikker, uanset kompleksitet af reglerne. Dette opsummerer flere af pointerne fra diverse studier inddraget i *Kapitel 4*, men også interviewpersonernes synspunkter, hvorfor det indgår som et relevant systemkrav. Her er det vigtigt at undersøge, som D påpeger, om branchen bør forsøge at skabe struktur i bygningsorienteret data, eller om det vil være mere gavnligt at behandle data i overensstemmelse med LD-principper og samtidig acceptere arbejdet med ustrukturerede data. Interviewperson B understreger, at det ville være ideelt, hvis alle regler fra de offentlige instanser var tilgængelige som SHACL. Dette understøttes af undersøgelsen udført i studiet af *Nuyts, et al. (2024)*, hvor det først belyses, at SHACL-valideringsrapport er tilstrækkeligt struktureret til at kunne bygge et intuitivt brugerinterface ovenpå disse rapporter. Dernæst konkluderes det ift. ACC og LD, at en tilgang, der bygger på SHACL, er den mest fordelagtige og anvende fordi, at SHACL-

syntaksen understøtter brugen af SPARQL-queries og returnerer en standardiseret valideringsrapport. Derudover kan SHACL håndtere regler og krav omhandlende datatilgængelighed, værdier/værdigrænser, relationer, matematiske beregninger og betingelser.

Interviewperson B og E har begge arbejdet med VDB'er i hver deres udviklede systemer, hvor NL-tekst bliver konverteret til vektorer og derefter lagret i en VDB. De beskriver begge diverse fordele ved at anvende en VDB. Samme princip anvendes i to at de eksisterende prototyper, beskrevet i *Afsnit 4.9* i studierne af *Zhong & El-Diraby (2024)* og *Nabavi, et al. (2023)*. VDB'en har den fordel, at den kan opbevare fragmenter systematisk i en struktureret DB på en anden måde end konventionelle RD'er. Den gør det muligt at opnå hurtig og præcis adgang til information og samtidig opnå kontekstuelt relevante svar. I kombination med LLM'er medfører dette validitet til responsen fordi, at det kan fremvises, hvorfra information kommer fra. VDB'en gør det muligt at arbejde med data uden, at data nødvendigvis er struktureret, hvilket anses som en væsentlig tilgang til lagring af data, hvorfor en VDB fremgår som et systemkrav til udviklingen af systemet. E tilføjer yderligere et krav om, at brugere ofte ønsker, at al nødvendig information kan tilgås fra ét samlet sted fremfor, at en persone skal fremsøge informationer fra flere kilder.

Baseret på den ovenstående komparative analyse kan følgende liste udarbejdes. Listen opsummerer analysens resultaterne af analysen og fremhæver de systemkrav og brugerbehov, der er afgørende for udviklingen af et nyt system der understøtter brugernes arbejdsopgaver og processer:

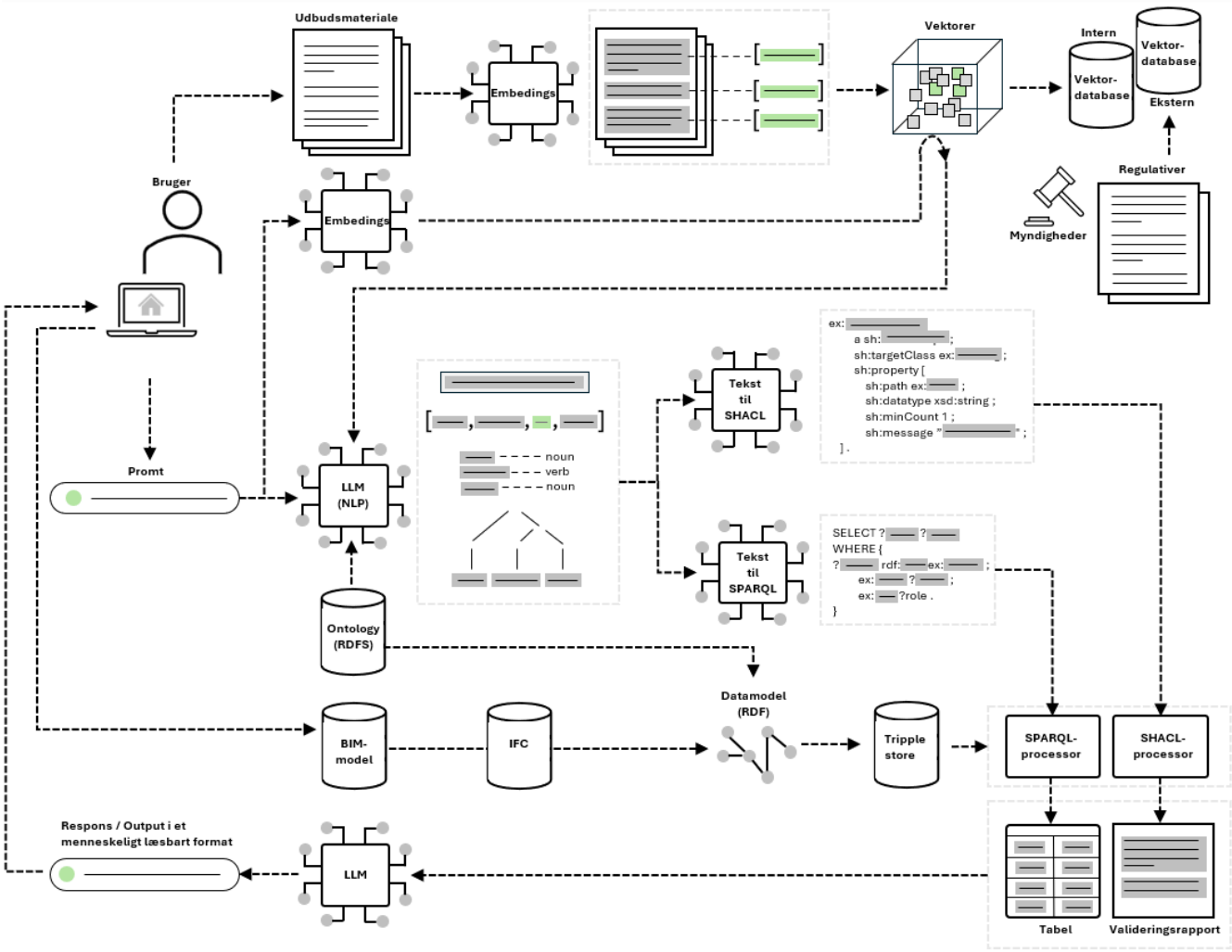
Pkt.	Krav.
1.	Overordnet kombination af SWT, LD, AI og BIM.
2.	Manuelle processer bør i videst muligt omfang elimineres.
3.	Alt nødvendig information tilgås fra ét samlet sted.
4.	Digital projekteringsassistent – hurtig respons på forespørgsler og kontroller.
5.	Én samlet enhed der er i stand til at indfange alle elementer af regler, semantikker, ontologier og logikker uanset kompleksitet af reglerne.
6.	Håndtering af døde dokumenter såsom PDF og ustrukturerede data.
7.	Anvend VDB til lagring af ustrukturerede data såsom regulativer og projektmateriale.
8.	Der skal være direkte og softwareneutral adgang til bygningsdata.
9.	Anvend RDF-grafer til at repræsentere BIM-modeller.
10.	Undgå IFC og interager direkte med BIM-model.
11.	Tilpas semantik mellem BIM-modeller (IFC) og regulativerne i NL.
12.	Erstat SMC i workflow og i systemet.
13.	Anvend gratis og pre-trained LLM'er til specifikke opgaver.
14.	Anvend LLM'er der kan udføre NLP-opgaver, herunder TO og POS-tagging, til at processerer og forstå brugernes NL-forespørgsler og regulativerne skrevet med NL-sprog for herved at: a. Omdanne NL-forespørgsler til SPARQL til at udtrække informationer. b. Omdanne NL-forespørgsler til regler, skrevet i SHACL, for at validere BIM-modeller.
15.	Undgå hallucinationer fra LLM'er og sikre valide resultater ved at give modellerne kontekst bl.a. vha. SW- og LD-principperne.
16.	Anvend domæneontologier, såsom BOT, med udvidelser efter behov.
17.	Håndtering af *Vague Constraints* og *Computanitonally Constraints*.
18.	Anvende SW- og LD-principper til at skabe overensstemmelse mellem data-præsentationerne og terminologierne i BIM-modeller og regulativerne.
19.	Mulighed for at udvælge specifikke regelsæt.
20.	Afrapportering skal indeholde reglen, der er anvendt, og hvilke objekter og attributværdier der bliver tjekket.
21.	Afrapportering skal fremvises som enten: Bestået, ikke-bestået eller fejl der ikke kunne tjekkes.
22.	Afrapportering skal foregå i NL-sprog og tage udgangspunkt i SHACL-valideringsrapport i et intuitivt brugerinterface.

Tabel 5 – Bruger og systemkrav

Fremtidigt workflow med et nyt system

Visioning i CD omhandler skabelsen af nye teknologier og systemløsninger, der kan forbedre og effektivisere brugernes arbejdsopgaver. De præsenterede brugerbehov og systemkrav, i *Afsnit 6.3*, fungerer som retningslinjer for, hvad et nyt system skal opfylde for at skabe værdi for brugerne. Indtil nu har undersøgelsen fokuseret på at afdække eksisterende teknologier og deres potentialer samt opnået en forståelse af brugeren og branchen. Dette kapitel illustrerer og beskriver, hvordan der kan skabes synergi mellem teknologierne og brugerne, så de understøtter arbejdsprocessen. Denne vision er visualiseret i *Figur 16*, og fungerer som en fremtidsfortælling om, hvordan brugerne kan arbejde på en ny måde i det nye system. Figuren tager afsæt i brugernes nuværende workflow, præsenteret i *Afsnit 6.1* og *Figur 12*, og danner grundlaget for et redesign af deres arbejdsproces. Målet er at skabe et fremtidigt og forbedret workflow, der er let at forstå og videreformidle. Yderligere er grundtanken bag det nye system integreret i det fremtidige workflow for at illustrere systemets struktur og de dele af brugernes arbejdsopgaver, som det skal understøtte og bidrage til.

Selvom *Figur 16* er et redesign af *Figur 12* kan det være svært at genkende brugernes nuværende workflow. Dette skyldes, at flertallet af aktører, samarbejdsplatforme, dokumenter og arbejdsgange er erstattet af tekniske elementer i det nye system. Formålet er at skabe en systemløsning, der kombinerer teknologier inden for SWT, LD, AI og BIM. Systemet er designet til at reducere antallet af involverede aktører, minimere manuelle processer og skabe nem adgang til al projektinformation for herved at give brugeren en effektiv og digital projekt- og projekteringsassistent. Assistenten, der bygger på kombinationen af de ovenstående teknologier, giver brugeren mulighed for at interagerer med projektinformation, BIM-modeller og regulativer med NL og derved udføre ACC. Redesignet illustrerer derfor i højere grad analysen af, hvilke elementer systemet skal bestå af, og hvordan disse elementer skal interagere indbyrdes i et workflow, så systemet opfylder systemkravene og brugerbehovene. Visionen og det fremtidige workflow er udarbejdet pba. den viden og erfaring, der er opnået gennem de forrige kapitler. Dette inkluderer interviewpersonernes indsigt, tidligere undersøgelsers konklusioner og erfaringer samt tidligere udviklede prototyper, der har forsøgt at realisere lignende systemer vha. de samme teknologier. *Figur 12* illustrerer brugernes fremtidige workflow med det nye system.



Figur 16 – Fremtidig workflow med et nyt system

På venstre side af modellen ses et ikon, der repræsenterer brugeren. Brugeren kan i denne sammenhæng være BIM-manageren, den projekterende eller projektlederen. Workflowet starter ved at brugeren formulerer en forespørgsel i systemets prompt. Herfra bliver prompten bearbejdet af en LLM, der anvender NLP-funktioner til at analysere brugerens forespørgsel. Systemet skal kunne håndtere forespørgsler, der henvender sig til projektinformation, BIM-modellen, kontrol/validering af data og eksterne kilder. Systemet skal derfor i høj grad anvende RAG-metoden til at hente data og generere et svar. Yderligere skal *Pattern-matching-based rules* anvendes for, at LLM'erne i systemet kan genkende, hvilken type af forespørgsel brugeren efterspørger. Når systemet identificerer, hvad brugeren efterspørger, vil LLM'en vide, hvor den skal hente sin information fra, og hvad informationen efterfølgende skal anvendes til. Efterspørger brugeren et svar der henvender sig til dataene, der fremgår af en BIM-model, vil brugerens forespørgsel behandles af en LLM, der kan oversætte tekst til en SPARQL-query. Hvis brugerens forespørgsel omhandler eller indikerer, at der er et ønske om at validere eller kontrollere dataene i BIM-modellen, vil forespørgslen behandles af en LLM, der kan omsætte tekst til en SHACL-validering.

Afhængigt af om NL-forespørgslen er omsat til en SPARQL eller en SHACL, vil systemet indeholde funktioner, der kan processere og eksekvere begge typer forespørgsler på projektets RDF-graf for derved at returnere en respons baseret den strukturerede BIM-data. BIM-modellen eksporteres til IFC-format og konventers efterfølgende til en grafbaseret model, i RDF-format, og lagres i en TS, som systemet kan sende queries til. Optimalt bør systemet skulle kunne håndtere queries på live BIM-data, hvilket gør det ønskværdigt, at softwareudbyderen muliggør direkte adgang til BIM-data gennem en RDF-graf.

Når en SPARQL-query eller SHACL sendes til TS'en, vil systemet returnere en respons på forespørgslen i enten tabelform eller som en valideringsrapport. Responsen fungerer som resultatet på forespørgslen og lagres derfor i en DB for at sikre, at brugerne kan tilgå resultaterne på et senere tidspunkt. Eftersom ikke alle brugere har kompetencerne til at forstå SPARQL-queries eller SHACL, anvendes LLM'er til at omdanne resultaterne til NL. Resultaterne ender tilbage hos brugeren og fremvises på interfacet.

Workflowet og systemet er designet til at håndtere ustrukturerede data på en måde, der gør det muligt for brugeren nemt at fremsøge information vha. forespørgsler formuleret i NL gennem systemets prompt. Grundtanken med systemet er, at brugeren kan uploade projektspecifikt materiale, såsom PDF-dokumenter med ustruktureret eller ikke-maskinlæsbar tekst, f.eks. udbudsmateriale og aftaledokumenter. Herefter anvender systemet en embeddings-model, der repræsenterer tekstindholdet som numeriske vektorer og opdeler indholdet i mindre tekstdele/fragmenter. Vektorerne lagres i organisationens interne VDB, som muliggør semantisk søgning i dataene. Denne systemopbygning giver brugeren mulighed for hurtigt og præcist at finde relevant information på tværs af store mængder data uden behov for kompetencer i komplekse søgekommandoer eller DB-arkitektur. Ved at tilbyde brugeren et system der understøtter vektorbaserede søgninger med NL-forespørgsler, opnås en semantisk forståelse af dokument- og tekstindholdet, hvor systemet kan identificere og skabe relation mellem data baseret på betydningsmæssig lighed frem for nøgleord. Når systemet identificerer de fragmenter, der har størst lighed med brugerens forespørgsler, vil fragmenterne blive behandlet af en LLM, som, kort og præcist, gengiver indholdet og angiver, hvor kildens indhold stammer fra. Hvis fragmentet indeholder et krav, der stilles til bygningen eller BIM-modellen, vil indholdet blive viderebehandlet af en LLM, der omdanner teksten til en SHACL-repræsentation, som herefter kan anvendes til valideringsprocessen.

Det skal bemærkes, at systemet ikke kun har adgang til en intern VDB men også en ekstern. Interviewperson E nævner, at de selv er i gang med at udnytte VDB'er til at tilbyde deres brugere effektiv informationssøgning af ustrukturerede og ikke-maskinlæsbart data. I forlængelse heraf påpeger interviewperson E, at deres regulatoriske informationer er placeret bag en betalingsmur. Det anses som usandsynligt, at

brugere ifm. denne undersøgelse kan undgå betaling for denne viden. Derfor er ideen, at brugeren igennem systemet kan købe adgang til disse eksterne VDB'er og herfra lave forespørgsler, der relaterer sig til viden, der er placeret heri.

6.4 Storyboard

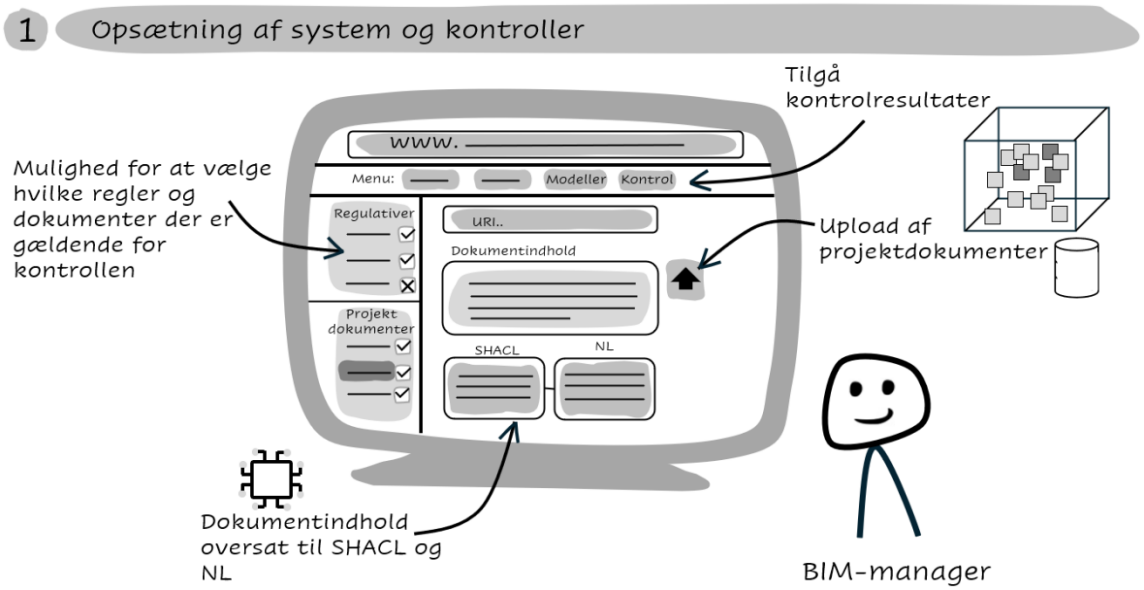
I forrige *Afsnit 6.2 og 6.3*, blev brugernes behov og krav til det nye system identificeret samt den overordnede vision. Dette afsnit præsenterer trinnet i CD, som beskriver udarbejdelsen af Storyboards, hvor visionen omsættes til konkrete systemfunktioner og strukturer, der tager hensyn til brugernes arbejdsopgaver og behov. Storyboards anvendes i denne sammenhæng til at visualisere og planlægge, hvordan brugerne vil løse deres arbejdsopgaver med det nye system.

Hvert Storyboard illustrerer de trin, som brugeren følger for at udføre arbejdsopgaven samt de specifikke systemfunktioner, der understøtter hvert trin. Dette sikrer, at de relevante systemfunktioner placeres korrekt i det nye system, hvilket både bidrager til en høj grad af brugervenlighed og sikrer, at brugernes arbejdsproces og behov understøttes. Storyboardsene illustrerer overvejelserne om, hvordan der kan skabes et samspil mellem brugerne, systemhandlingerne, systemfunktionerne og brugergrænsefladerne. Brugergrænsefladerne præsenteres på et overordnet niveau og repræsenterer de indledende tanker til udviklingen af systemets interface.

Der er udarbejdet tre selvstændige Storyboards. Storyboardsene repræsenterer og illustrere hhv. forskellige scenarier af hvordan og hvornår brugerne vil udføre arbejdsopgaven med det nye system. Scenarierne er hhv.: (1) Opsætning af system og kontroller (2) Forespørgsler på model- og projektinformation (3) Kontrol af BIM-modeller med ACC.

1 - Opsætning af system og kontroller

Det første Storyboard viser brugeren i et scenarie, hvor et nyt projekt er opstartet, og projektets kontroller skal klargøres gennem systemet. Brugeren, der er ansvarlig for denne arbejdsopgave, vil formodentlig være BIM-manageren eller IKT-lederen. Hidtil i undersøgelsen er det blevet argumenteret for, at systemet skal fungere som en form for projekteringsassistent, hvor systemet kommer tæt på brugeren under udførelsen af arbejdsopgaven. Dog ses det mest hensigtsmæssigt, at de ansvarlige for opsætningen ikke skal gennem BIM-softwareprogrammer for at tilgå systemfunktionerne men derimod, at disse tilgås gennem en Webpage. Funktionerne, som vil være tilgængelige for brugeren ifm. denne opgave, ses på nedenstående *Figur 17*.



Figur 17 – Storyboard 1 - Opsætning af system og kontroller

BIM-manageren skal, som i sit nuværende workflow, kunne vælge hvilke regler og krav der er gældende for det enkelte projekt, og som dermed skal kontrolleres. Som det fremgår af figuren, vil der forsat være en manuel fremgangsmåde i at til- og fravælge diverse regler. Systemet giver brugeren mulighed for at markere, om den enkelte regel er gældende for projektet, og om den skal indgå i kontrollen, vha. en afkrydsningsboks ud fra hver regel. Det illustreres på venstre side af brugergrænsefladen, at der er en klar opdeling af de regler, der stammer fra regulativer og de regler eller informationer, der stammer fra projektdokumenter. Reglerne fra regulativerne vil være angivet med et kort navn, der antyder reglens omfang. Klikker brugeren ind på reglerne, er det muligt at få oplysninger om, hvor reglen stammer fra. Hertil vil der være en URI, som brugeren kan følge for at tilgå reglen fra den aktuelle kilde og hermed forstå reglen i en større kontekst. Derudover vil brugergrænsefladen tilbyde en visning af den SHACL, der repræsenterer den pågældende regel samt en beskrivelse af den i NL. Dette skyldes, at der en antagelse om, at der vil være brugere, der både har og ikke har kompetencerne til forstå SHACL. Men på samme tid vurderes det vigtigt, at hvis brugeren sidder inde med disse kompetencer, at systemet imødekommer kompetenceniveauet. Det skal bemærkes, at der er relativ få funktioner tilgængelige for brugeren at interagere med, hvilket skyldes, at systemet og de bagvedliggende ML-modeller udfører store dele af arbejdet.

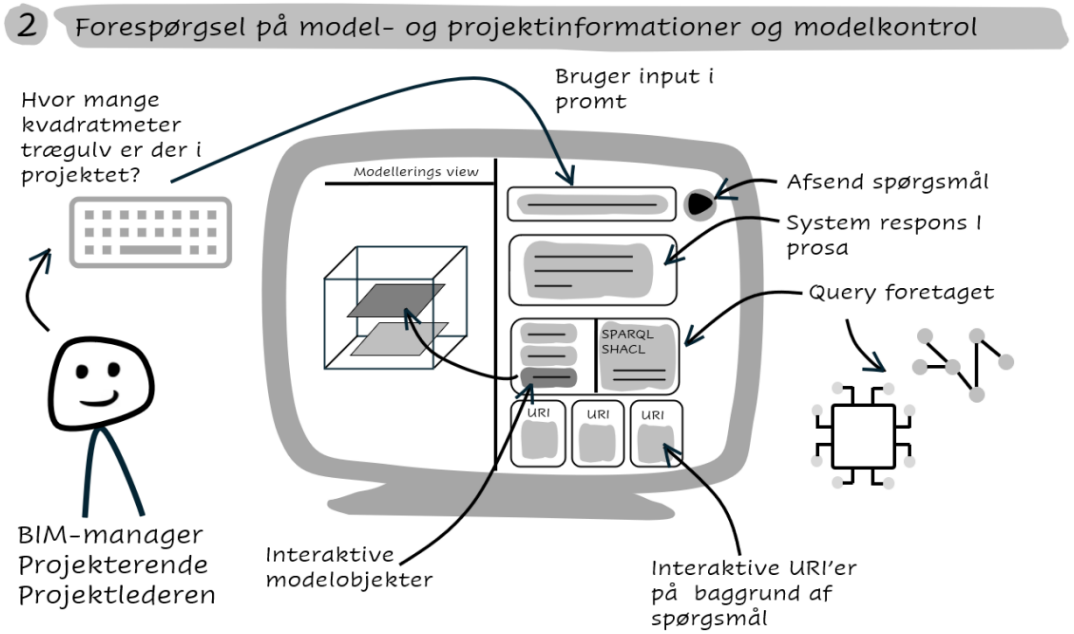
De samme overordnede principper gør sig også gældende for regler, der stammer fra projektdokumenter. Systemet vil inkludere en funktion, der giver brugeren mulighed for at uploade et dokument til den interne VDB. Tanken er, at system herfra selv generer en URI, der indikerer, hvilket projekt som dokumentet tilhører, og hvad navnet er på projektet. På den måde giver det systemet mulighed for at skabe en semantisk forbindelse mellem dokumentindholdet og projektet. Brugergrænsefladen vil tilbyde

en forhåndsvisning af dokumentindholdet, den SHACL som systemet har udarbejdet baseret på dokumentindholdet og afslutningsvist en tekst i NL, der forklarer, hvad denne SHACL har til formål at validere.

Eftersom systemet er tiltænkt som et ACC-system, vil systemet naturligvis udføre en kontrol. Hensigten er, at brugeren herefter kan tilgå kontrolresultaterne, som er vist på figuren og interagere med disse for at beslutte, om nogle af fejlene er relevante i projektkonteksten.

2 – Forespørgsel på modelinformation og modelkontrol

Nedenstående Storyboard, *Figur 18*, viser et scenarie, som flere af brugerne kan opleve ifm. deres arbejdsopgave. Både BIM-manageren, den projekterende eller projektlederen kan sidde i en situation, f.eks. i et bygherre- eller et entreprenørmøde, hvor der bliver stillet komplicerede spørgsmål, hvor svaret kan eller skal findes i projektets BIM-modeller. Nedenfor vises det, hvordan systemet understøtter brugeren i dette tilfælde.



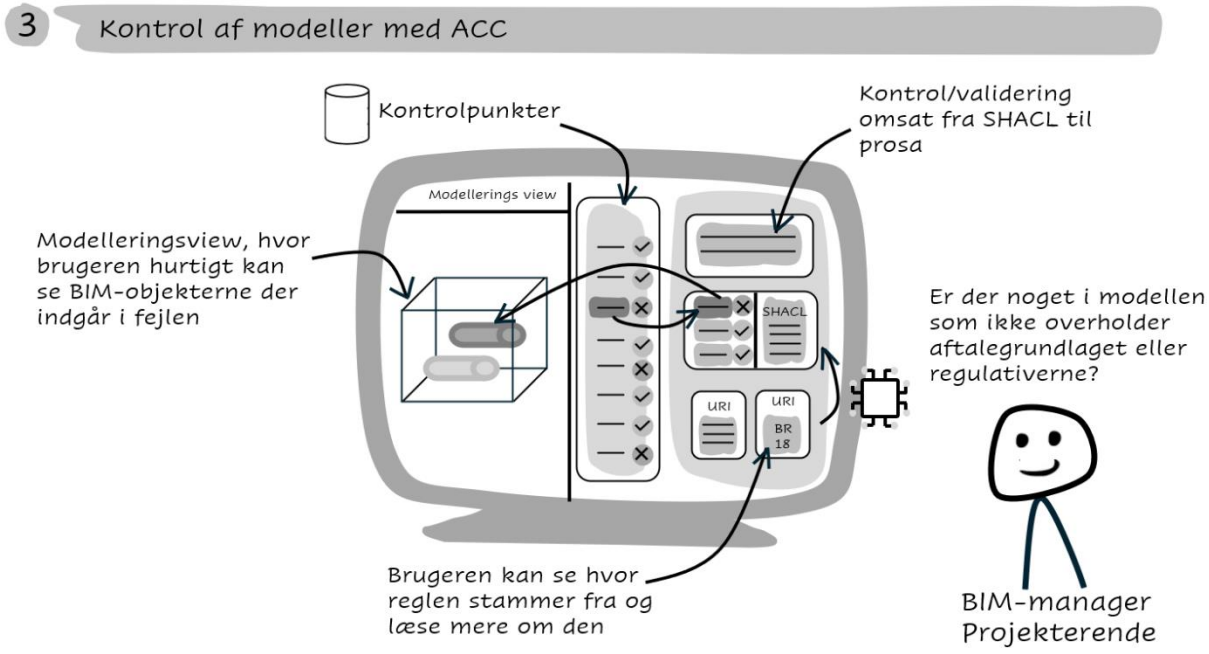
Figur 18 – Storyboard 2 - Forespørgsel på modelinformation og modelkontrol

Som det vises på figuren, kan brugeren igennem sit BIM-softwareprogram tilgå systemet gennem et Plug-in. Herfra kan brugeren tilgå en prompt, hvor der kan stilles spørgsmål i NL. I scenariet er der en bruger, der efterspørger: Hvor mange kvadratmeter trægulv er der i projektet? Spørgsmålet kunne modsat være: Er der 34.000 kvadratmeter trægulv i projektet? Der er en klar forskel på de to spørgsmål. Det første spørgsmål kræver, at systemet laver og udfører en SPARQL-query, eftersom spørgsmålet søger et svar, der kræver en udtrækning af data. Det sidste indikerer en

validering af projektets data, hvorfor der skal udarbejdes en SHACL. Systemet kan håndtere og identificere begge typer af spørgsmål. Systemet tilvejebringer en tekst der forklarer SPARQL-query'en og SHACL-valideringen i NL og fremviser disse på brugergrænsefladen. Hertil vil systemet også fremvise, hvilke URI'er der relaterer sig til brugerens forespørgsel og systemets respons. På den måde er det muligt for brugeren at se, hvilke kilder der danner grundlag for systemets respons. Ligeledes giver det brugeren mulighed for at tilgå disse URI'er, hvis brugeren ønsker at validere eller undersøge den bagvedliggende information. Slutligt vil systemet fremvise, hvilke BIM-objekter der anvendt som grundlag for responsen. Brugergrænsefladen giver brugeren mulighed for at interagere med disse objekter ved at isolere objekterne i et isoleret *View*.

3 – Kontrol af modeller med ACC

Det sidste Storyboard omhandler arbejdsopgaven og et scenarie, hvor den projekterende skal tilgå og udrede de issues, som BIM-manageren har identificeres gennem systemet. Det vises på nedenstående *Figur 19*, at brugeren tilgår systemet og kontrollerne gennem et plug-in i sit BIM-softwareprogram.



Figur 19 – Storyboard 3 – Kontrol af modeller med ACC

Brugeren kan se kontrolpunkterne og resultaterne direkte på brugergrænsefladen. Kontrolpunkter, der har udløst issues, er tydeligt markeret, og antallet af fundne issues fremgår. Dette giver brugeren et hurtigt overblik over kontrollens omfang og identificerede uoverensstemmelser. Fra brugergrænsefladen kan brugeren klikke på de enkelte kontrolpunkter og få vist de BIM-objekter, der udløser issues. Derudover

kan brugeren klikke på de specifikke objekter for at få dem fremhævet i BIM-softwarens modelleringsview. Denne visning kan åbnes direkte gennem systemet.

Systemet viser også den relevante SHACL, der fungerer som fundament for kontrolprocessen, hvilket tidligere er illustreret i Storyboards. Denne SHACL præsenteres sammen med en beskrivelse i NL, hvilket gør det lettere for brugeren at forstå, hvad reglen indebærer. Brugergrænsefladen fremhæver desuden de URI'er, der ligger til grund for kontrollen, så brugeren hurtigt kan få adgang til yderligere informationerne. Dette øger forståelsen af de fundne issues og hjælper brugeren med at finde mulige løsninger. Da systemet arbejder direkte med BIM-data og bygger på ACC-principper, er tanken og hensigten, at brugeren i realtid kan se, om det Issue brugeren forsøger at løse faktisk bliver løst.

6.5 Systemdesign med User Environment Design (UED)

Systemarkitekturen er udarbejdet med udgangspunkt i de foregående analyser Work Models, identificering af brugerbehov og systemkrav og en tekniske fremstilling af visionen for det nye system. UED'en udarbejdes for at kunne fremvise de tekniske funktioner i systemet samt hvilke brugerinputs og responser, som systemet kan håndtere og afgive. Systemarkitekturen er vist som hhv. et add-in til eksisterende BIM-software og en ekstern webside, der kan interagere med systemets backend. UED bruges til at give brugeren en samlet forståelse af de funktioner, der præsenteres i Storyboards, og hvordan disse funktioner forbindes for at udvikle det samlede system, som understøtter det fremtidige workflow, og hvordan brugeren navigerer i systemet. UED'en kan ses på nedenstående *Figur 20*.

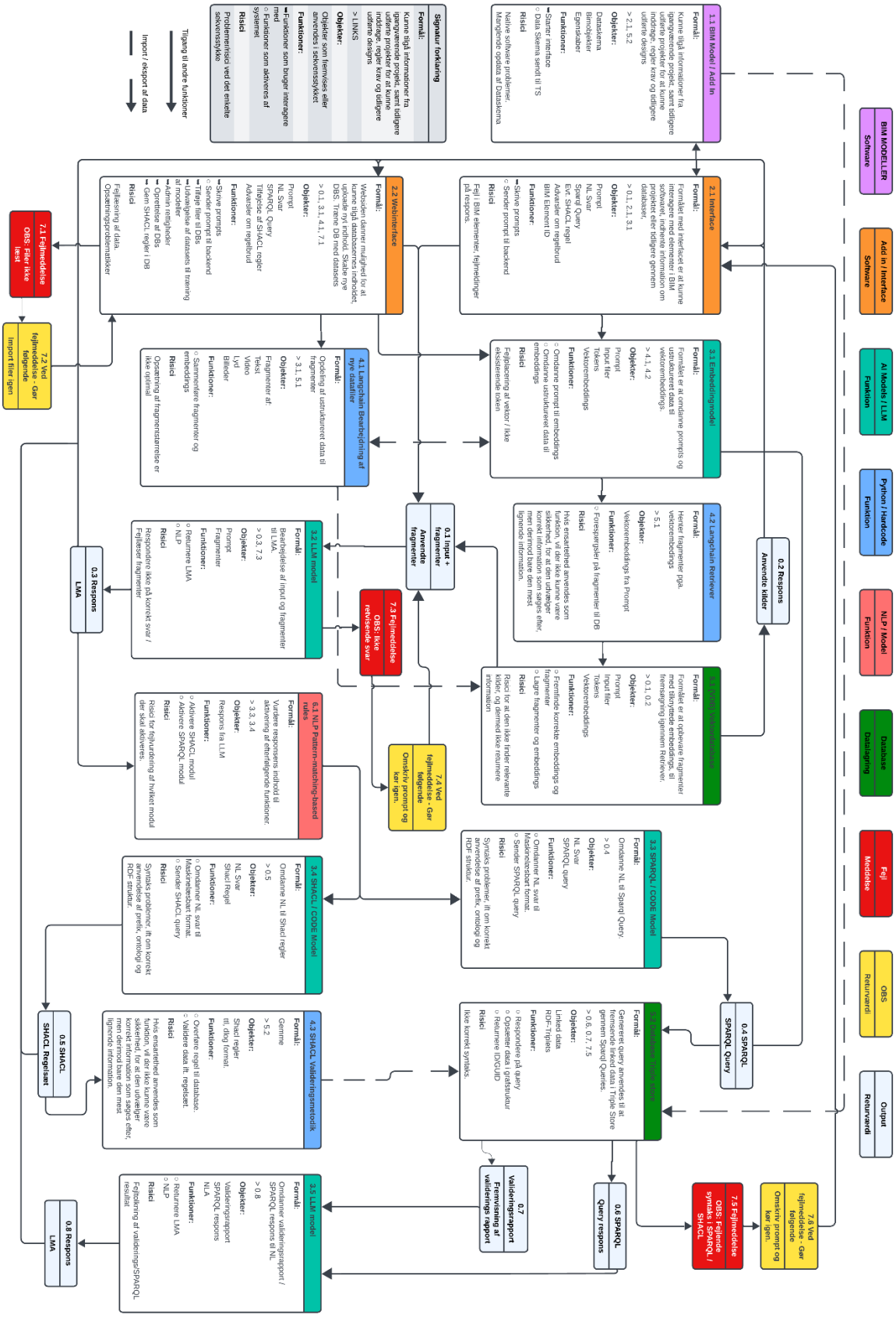
Adgangen til systemet sker via enten BIM-Add-in eller webbrowser afhængig af brugerens rolle og behov. Interfacet vælges ud fra, om systemet skal bruges som et kontrolværktøj eller til opsætning af systemet og DB'er. UED'en viser to interfacenoder: (1) 2.1 BIM-Add-in (2) 2.2 Website. Disse adskilles pba. deres forskellige funktioner, dog giver begge adgang til VDB og GDB og muliggør queries. BIM-Add-in'et tilføjer funktioner, der forbinder software og DB'er ved at hente element-ID'er. Dette muliggør manipulation af både geometri og egenskaber. Brugerens inputkrav vil variere afhængigt af den funktion, som systemet skal udføre. Ved opsætningen af systemet skal det kunne håndtere både strukturerede og ustrukturerede data. Den strukturerede data omfatter: (1) Projektspecifikke attributter (2) Geometriske repræsentationer, herunder referenceprojekter, der kan anvendes som grundlag og referencer i projekteringsprocesser. Den ustrukturerede data omfatter: (1) Tekst (2) Billeder (3) Videoer (4) PDF-dokumenter samt udvalgte standarder (5) Kravspecifikationer og lovgivning. De ustrukturerede data integreres i systemet, som en del af grundlaget for RAG, hvor de kombineres med LD og BIM-data for at berige responsen fra

LLM'en ved af indhente fragmenter fra VDB'en, som bearbejdes gennem embeddings-modellen og LangChain.

Ved at anvende RAG kan fragmenterne bruges som input til LLM'en, i sammenhæng med forespørgslen, hvilket gør det muligt at generere et forbedret og mere præcist svar til brugeren. Efterfølgende aktiverer NLP-funktionen *Pattern-matching-based rules* til at identificere valideringsfunktioner eller queries i responsen. NLP-modulet omdanner NL til enten SPARQL-forespørgsler eller SHACL-regler, der bruges til at validere og sende quries til TS'en. NLP anvendes til at konverterer NL til hhv. SPARQL-queries eller SHACL-regler. SPARQL-queries og SHACL-regler kan sendes til TS'en, som overordnet returnerer valideringsrapporter eller en respons på querien. Interfacet giver efterfølgende brugeren mulighed for at kunne vælge om valideringsrapporterne skal downloades. Da responsen fra TS, ikke er NL, bearbejdes den af LLM'en så den returnerer i NL.

Formålet med at bearbejde data ud fra funktioner og teknologier som LangChain, LLM'er og DB'er er at skabe et fundament for at besvare brugerens forespørgsler baseret på den udførte prompt. UED'en illustrerer, hvordan systemet er tilpasset til at validere både regler i NL og strukturerede regler samt udvælge projektspecifikke krav og relevante regulativer. Systemarkitekturen er designet til at kunne returnere relevante responser til brugeren, hvor hallucinationer minimeres (Sajid, 2024).

Systemet bruger LLM'er til at generere maskinlæsbare responser, som kan anvendes til at forespørge direkte i DB'erne. Disse responser bliver derefter omdannet til NL vha. yderligere brug af LLM'er. Brugerens prompt vil, baseret på den strukturerede og ustrukturerede data kunne: (1) Opdele ustruktureret data i fragmenter (2) Returnere data fra BIM-modeller (3) Anvende queries og valideringsrapporter og oversætte dem til NL. Disse overordnede funktioner hjælper brugeren med at navigere i projektet information, og udføre kontroller af det tilgængelige data. Den udarbejdede UED er fremvist på nedenstående *Figur 20 og* *Bilag 15*.



Figur 20 - User Environment Design

7 PROTOTYPEUDVIKLING

I dette kapitel præsenteres de udarbejdede prototyper, som baseres på *Kapitlerne 4, 5 og 6*. Kapitlet præsenterer den tekniske prototype samt Mockup'en af interfacet. Dette omhandler beskrivelser af kernefunktionerne, der findes i prototyperne, og giver en overordnet beskrivelse af funktionerne og den bagvedliggende tekniske fremstilling af disse. En detaljeret teknisk beskrivelse af den tekniske prototype, kan findes i *appendiks, Kapitel 13*. Beskrivelsen af prototyperne anvendes til testen af prototyperne, hvor der modtages feedback af designet og de indlejrede funktioner. De identificerede muligheder og begrænsninger ved systemets tekniske funktioner og design inddrages afslutningsvist i diskussionen om udviklingen af et fremtidigt system.

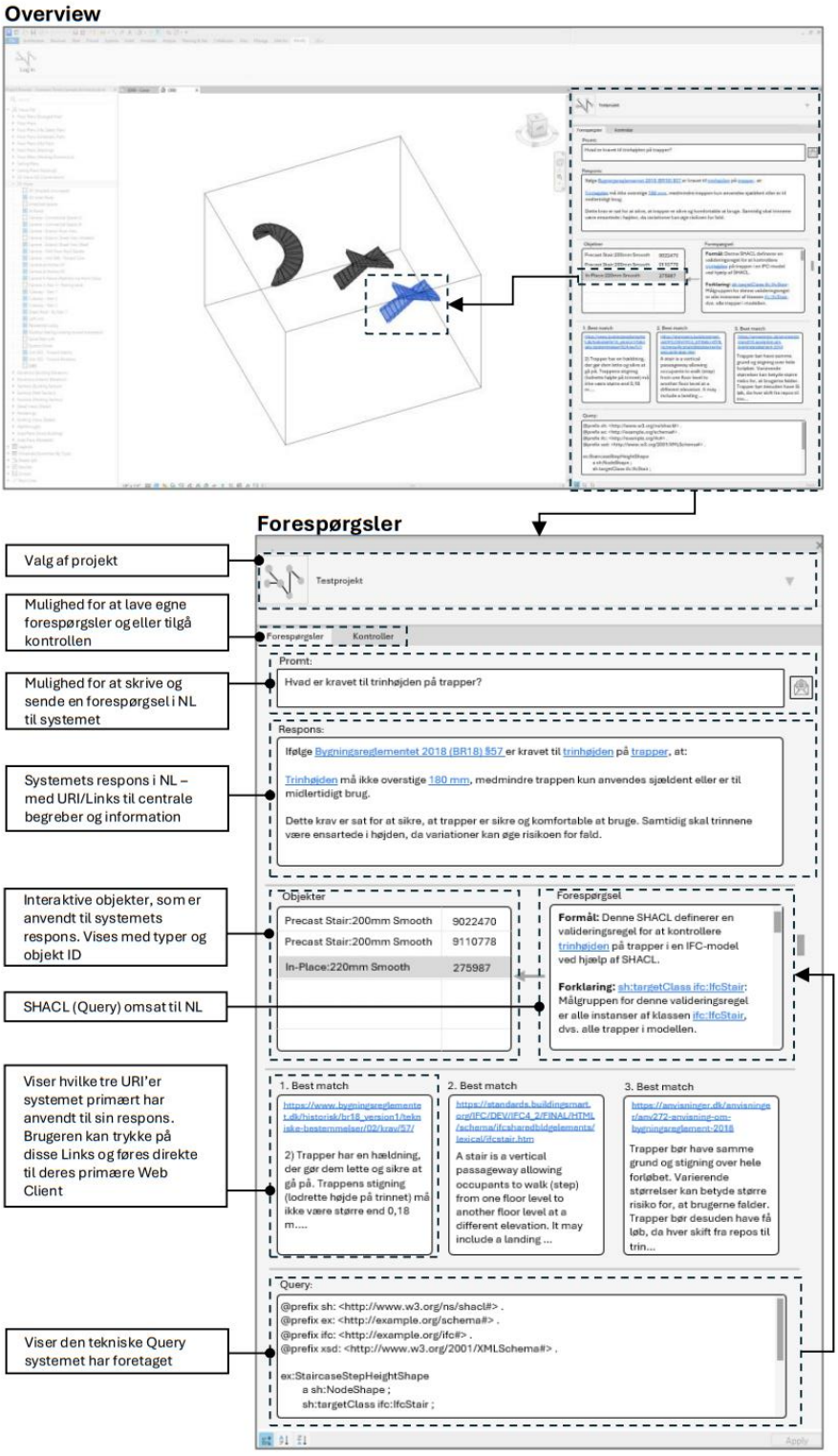
7.1 Mockup af interface

Udarbejdelsen af Mockup af interface til det nye system bygger på det fundament, der blev skabt gennem Storyboards og UED. Storyboardene visualiserede, hvordan det nye system kunne understøtte brugernes arbejdsopgaver ved at fremhæve centrale tekniske funktioner og illustrere, hvordan disse kunne udformes i et interface. UED'en samlede kravene fra de udviklede Storyboards og organiserede dem i en systematisk struktur med tilhørende beskriver af formål og funktionalitet. Mockup'en af interfacet er en grafisk fremstilling af UED'en og de indledende ideer til brugergrænseflader, som blev beskrevet og visualiseret i Storyboards. Formålet med Mockup'en er at skabe en prototype, der præsenterer et intuitivt og letforståeligt interface. Den skal give brugeren en klar fornemmelse af de tiltænkte systemfunktioner og tilhørende arbejdsgange. Mockup'en fungerer desuden som et centralt værktøj i testforløbet og i dialogen med brugerne for at verificere, at systemet og dets design lever op til de fastlagte systemkrav og opfylder de identificerede brugerbehov

Der er udarbejdet en Mockup med fokus på *Storyboard 2 og 3*. Dette er udført fordi, at disse Storyboards adresserer både undersøgelsens og brugernes centrale behovet, som er et system, der fungerer som en projekteringsassistent. Systemet og interfacet skal gøre det muligt for brugerne hurtigt og nemt at kontrollere deres BIM-modeller og finde nødvendig information. Mockup'en har til formål at skabe et testgrundlag for brugerne, hvor det illustreres, hvordan systemet tilgås gennem et plug-in i deres BIM-software. Via dette plug-in kan brugerne interagere med systemets funktioner, udføre kontroller og få adgang til relevant BIM-data og projektinformation. Alt dette understøttes af et brugervenligt og intuitivt interface.

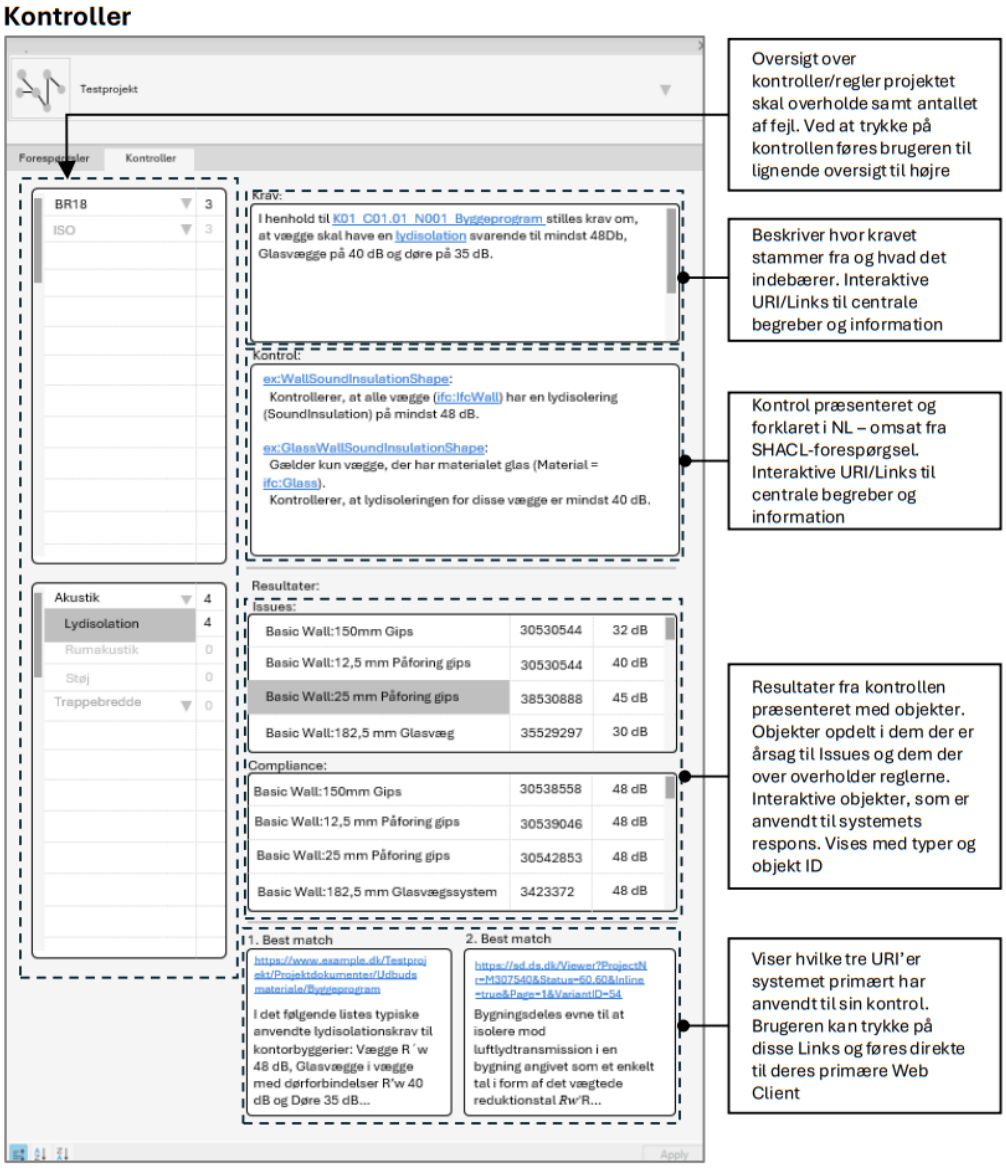
Som beskrevet i *Kapitel 2* er interfacet udarbejdet i Microsoft PP. Figurer, ikoner og symboler, i PP, er anvendt til at designe interfacet, det overordnede layout og systemets knapper og funktioner. Yderligere er PP brugt til at udarbejde reelle knapper og funktioner ved at linke disse på tværs af layouts/slides. Interfacet findes i *Bilag 16*,

men er også vist og kort beskrevet i nedenstående *Figur 21 og 22*. *Figur 21* demonstrerer en bruger, der ønsker at finde ud af, hvad kravet er til trinhøjden på trapper er og samtidig kontrollere, om dette krav er overholdt i BIM-modellen. Figuren refererer til *Storyboard 2*, hvor brugeren med NL kan lave forespørgsler.



Figur 21 - Interface – Forespørgsler

På nedenstående *Figur 22* ses den del af interfacet, som referer til *Storyboard 3*. Her tilgår brugeren den faktiske projektkontrol, hvilket vil sige, at indholdet og informationerne heri fremhæver de potentielle afvigelser, brugen skal udrede inden aflevering. Figuren efterviser et eksempel, hvor en bruger tilgår et projektkrav, som ikke stammer fra regulativer, men derimod fra det projektspecifikke materiale, som i dette tilfælde er det udleverede byggeprogram.



Figur 22 - Interface Kontroller

7.2 Teknisk prototype

Den tekniske prototype er sammensat af forskellige scripts og grafer, der vha. Python-programmeringssproget og supplerende biblioteker, udgør delelementer af den samlede prototype. Disse delelementer udgør et samlet billede af, hvordan et fuldt fungerende system kan opbygges samtidig med, at der opnås interoperabilitet, høj

hastighed og mere valide resultater. Prototypens tekniske dele består af en række forskellige Python-scripts, terminal prompts og Dynamo Grafer, som muliggør oprettelsen af egne ML-modeller, fine-tuning af egne og eksisterende ML-modeller. Sammenkoblingen mellem LD, ML og BIM sker igennem en Dynamo Graf, som anvender hhv. Ollama/llama 3.1 eller OpenAI/GPT-4o, som LLM, til bearbejdningen af information. Prototypen fremviser, hvordan brugeren kan lave forespørgsler i NL, der omdannes til SHACL og SPARQL og indhenter data fra VDB og GDB.

Gennemgangen af den tekniske prototype er struktureret ud fra de samlede funktioner og præsenteres i forskellige detaljeringsgrader. De følgende funktioner kan overordnet opdeles og beskrives som følgende og inkluderer de opstillede bilag:

Oprettelse af ML-modeller og datasæts samt fine-tuning af modeller*:*

- *Bilag 17, 18, 19, 20 og 21*

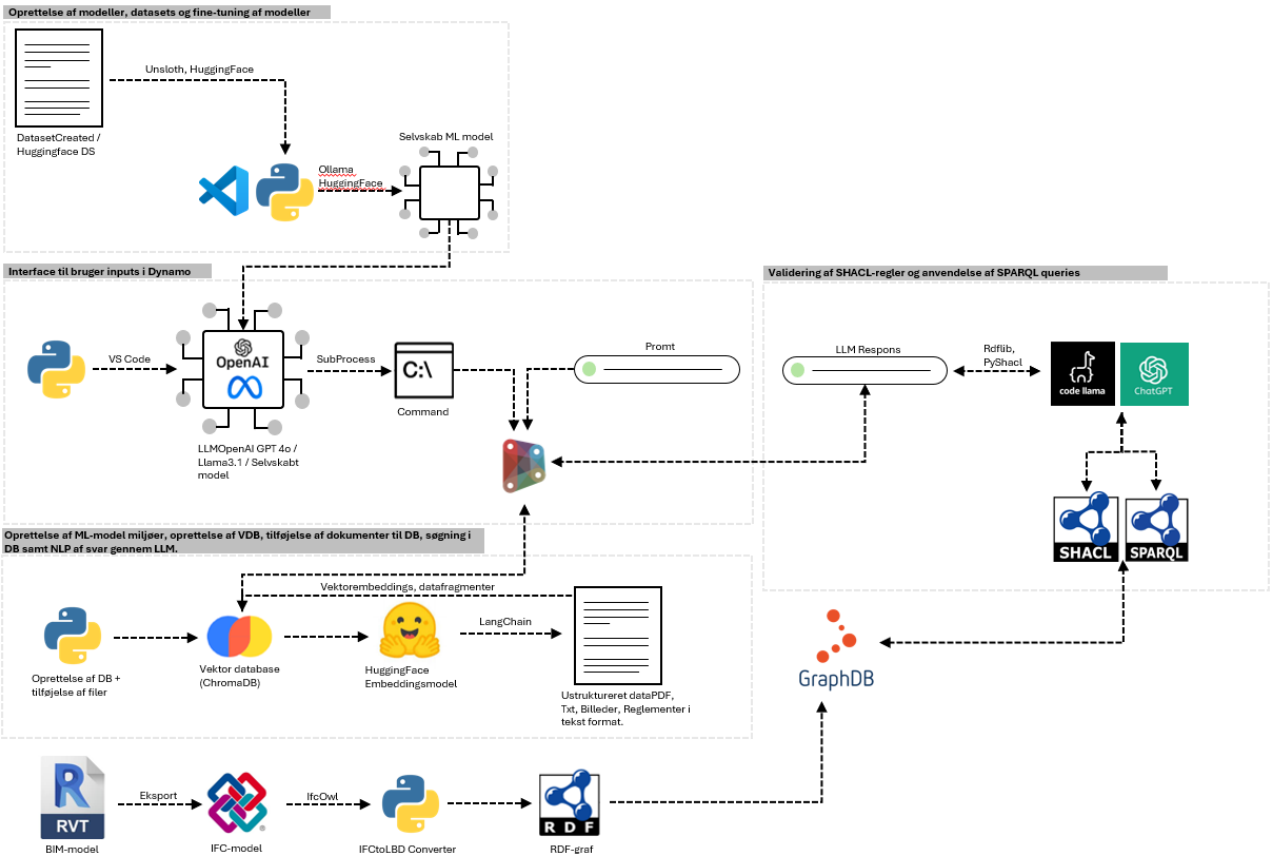
Oprettelse af ML-model-miljøer, oprettelse af VDB, tilføjelse af dokumenter til DB, søgning i DB samt NLP af respons gennem LLM:

- *Bilag 22, 23, 24, 25, 26, 27, 28, 29, 32, 34 og 35*.

Dynamo graf med adgang til oversættelsen af SHACL/SPARQL:

- *Bilag 30, 31, 33, 34 og 35*.

De ovenstående opdelinger af bilag vil overlappe hinanden, når funktionerne sammensættes i prototypen. Den tekniske beskrivelse af hver funktion kan læses i *appendiks*. Heri foretages en uddybende beskrivelse af indholdet i prototypen, der kan give læseren en bedre forståelse for de tekniske komponenter. I nedenstående *Figur 23*, er der opstillet en overordnet visualisering af de involverede teknologier og softwares, som viser, hvordan de forskellige funktioner, fra overstående bilag, er sammensat i systemet.



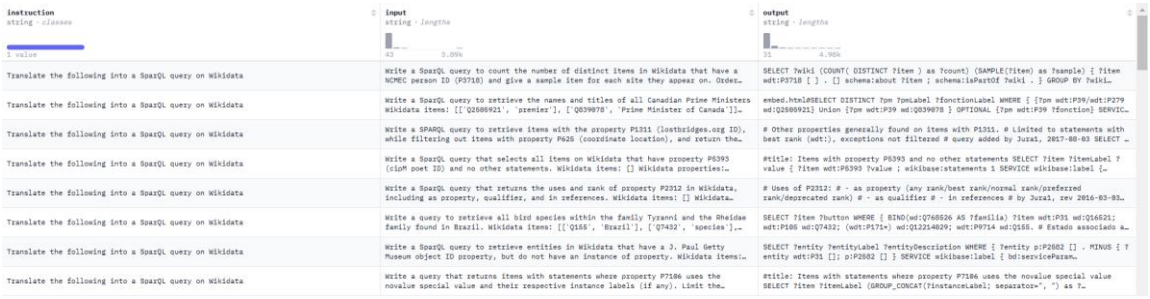
Figur 23 - Anvendte teknologier i teknisk prototype

Oprettelse af modeller, datasæts og fine-tuning

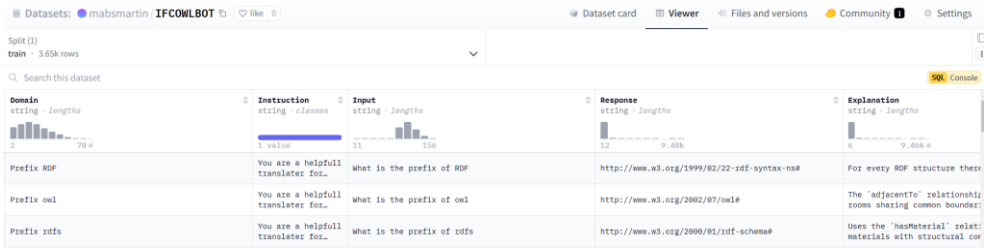
Når ML-modeller anvendes til udførelsen af opgaver, vil der igennem udvælgelsen skulle fokusere på anvendt data ved pre-training, funktioner og mulighed for fine-tuning. Ved at anvende en fine-tuningsproces bliver det muligt, at den enkelte ML-model kan tilpasses til den givne arbejdsopgave, som modellen skal udføre. Ved at sammenføre selvskabte eller hentede datasæts, muliggøres træningen af modeller til at opnå en forståelse for den udførte respons. Den primære fremgangsmåde, for oprettelsen af modeller i denne undersøgelse, består af eksisterende ML-modeller, som igennem fine-tuning (Huggingface, 2024) opdateres med dataene fra det udvalgte datasæt. Yderligere er der oprettet en ny ML-model, hvilket er foregået ud fra eksisterende modeller, hvor disse er valgt ud fra HFs model hub samt Ollama (Huggingface, 2024; Ollama, 2024). Fine-tuningsprocessen er udført vha. Python-biblioteket Unsloth (Unsloth, 2024), som tillader fine-tuning med eksisterende- og selvskabte datasæts.

Datasæt bruges til træning af ML-modeller for at definere viden og instruktioner til modellerne. Disse anvendes og præsenteres primært som prædefinerede prompts, responser og instruktioner, som ML-modellerne bruger til at opnå og udvikle deres forståelse. Hvert datasæt består af dataceller, som fremvist på nedenstående *Figur 24 og 25,* der består af et hentet datasæt, der består af et hentet datasæt, som

indeholder SPARQL-kommandoer samt et selvskabt datasæt med oversigt over ontologier og RDF-triples. I visse tilfælde vil datasættet bestå af flere kolonner, hvor det er nødvendigt at udvælge de input, som ML-modellerne potentielt vil have behov for at anvende.



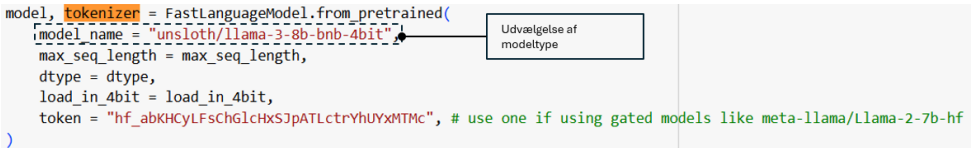
Figur 24 - Udklip af datasæt htriedman/wiki-sparql (Triedman, 2023)



Figur 25 - Udklip af selvudviklet datasæt (Bilag 19), (Martiny, 2024)

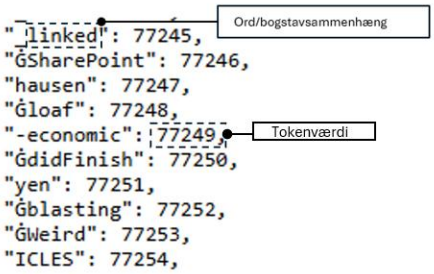
Hvis der anvendes datasæt, der ikke uploades til HF-platformen eller hvis der ikke er videreudviklet på eksisterende datasæt, kan der være en behov for at træne lokalt på computeren. *Bilag 17 og 18* indeholder to Python-scripts. *Bilag 17* henter eksisterende datasæt lokalt til en computer, og *Bilag 18* omdanner datasæt, hvor kolonnerne endnu ikke indeholder instruktion, prompt og respons til det ønskede format med korrekt navngivning.

Prototypens træningsproces er baseret på metoder udviklet af Unsloth (Unsloth, 2024). Her findes en række Python-script-skabeloner, som anvendes til fine-tuning af modeller vha. lokale eller online datasæt. Scriptet behandler datasættet og tilpasser kolonnerne til det korrekte format, så de kan anvendes i den videre proces. Således muliggøres fine-tuning af ML-modellen, og det understøtter både online publicering og lokal anvendelse. *Bilag 20* indeholder en tilpasset version af Unsloth-scriptet (Unsloth, 2024). Scriptet bruges til at omsætte datasæt til tokens og kan dermed identificere sammenhænge i de sekvenser, der er defineret i det anvendte datasæt. Ved brug af Unsloth specificeres den anvendte tokenizer, som tilpasses til den model, der skal fine-tunes, hvilket er fremvist på nedenstående *Figur 26* (Mittal, 2024). Formålet med processen er at forsøge at få ML-modellerne til at opnå en bedre forståelse for query sprog. Prototypen anvender Llama 3.1 8b, hvilket kan defineres som en mindre model med begrænsede muligheder for at opnå en kompliceret forståelse.



Figur 26 - Udvælgelse af model og tokenizer – Udklip af Bilag 20

Brugen af Llama 3.1 8b-modellen medfører, at processeringen af datasæt resulterer i mindre komplekse responser og kræver færre ressourcer for at opnå en forståelse af de tokens, der genereres fra det anvendte datasæt. Samtidig vil det være oplagt, når specifikke ML-modeller anvendes, at undersøge, hvordan modellens tokenizer har opbygget eksisterende tokens, og hvordan disse stemmer overens med datasættets indhold. Brugen af Llama 3.1-tokenizeren som en del af LLM'en muliggør effektiv og præcis processering af NL. Pre-trainingen indeholder ikke tokens, der naturligt relaterer sig til byggebranchens termer, IKT, BIM-data. I et virkeligt system kræves det, at den korrekte tokenizer anvendes og tilpasses til indholdet og strukturen af det anvendte datasæt. Nedenstående udklip i *Figur 27* fremviser et udklip af anvendte tokens fundet i Llama 3.1's tokenizer-fil.



Figur 27 - Udklip af Llama 3.1 tokenizer file. (Meta, 2024)

Afslutningsvist publiceres LLM'en til lokalt eller online brug. Ved anvendelse af HF, sker udgivelsen automatisk under træningen, mens Ollama benytter en push-funktion, som er illustreret i *Bilag 21*.

Sammenkobling BIM, Data og ML/AI

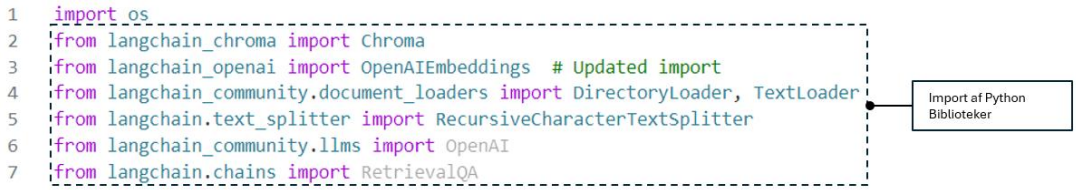
Den tekniske prototype, der integrerer BIM, ML og LD, er udviklet vha. Dynamo og Dynamo Player, som fungerer som Add-ins i projektet. Prototypen indsamler data ved at anvende VDB'er, der oprettes og behandles via LangChain og viderebearbejdes af en LLM. I forsøget anvender prototypen to forskellige LLM'er til at analysere og behandle data fra VDB'en og prompten fra brugerne. Prototypen anvender OpenAI 4o og Llama 3.1 som LLM'er. Grundlaget for oprettelsen af VDB'er, tilføjelse af dokumenter og søgning i VDB'erne er blevet specifikt udarbejdet og tilpasset til hver af de anvendte modeltyper. I gennemgangen af modellerne tages der udgangspunkt i OpenAIs LLM. *Bilag 23* viser oprettelsen af VDB'er, og følger samme fremgangsmåde som *Bilag 24*, der anvender Ollama- og HF-modeller. På samme måde er *Bilag 25 og*

26 samt *Bilag 27 og 28* struktureret og følger tilsvarende fremgangsmåde, hvor forskellen er anvendelsen af LLM og embeddings-model.

I prototypen er der anvendt et virtuelle miljø igennem Anaconda (Anaconda, 2017). Virtuelle miljøer er anvendt for at sikre, at Python-bibliotekerne har de korrekte versioner, der matcher kravene for de enkelte scripts og funktioner, som anvendes. Terminalkommandoerne til oprettelse af de virtuelle miljøer er vist i *Bilag 22*. Disse kommandoer opretter et nyt virtuelt miljø, aktiverer dem og installerer de nødvendige biblioteker til brug i prototypen. En oversigt over processen er illustreret i *Figur 28*

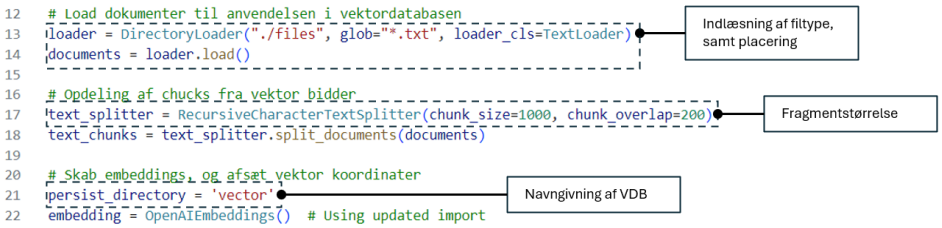
Figur 28 - Oprettelse af virtuelt miljø gennem Conda, udklip af Bilag 22.

Første del af Python-scriptet, vist på *Figur 29,* importerer de installerede biblioteker, som anvendes til oprettelsen af VDB'en.



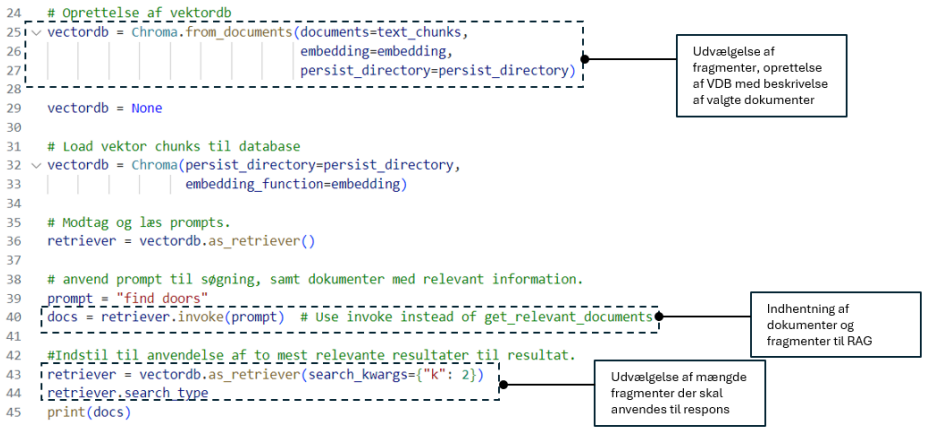
Figur 29 - Import af Python-biblioteker - Udklip af Bilag 23

VDB'ens indhold opstår ved upload af dokumenter. I den anvendte prototype er der taget udgangspunkt i .txt-dokumenter og Turtle-filer (.ttl), der indeholder både strukturerede og ustrukturerede data. Dokumenternes indhold bearbejdes ved at opdele det i fragmenter, hvor der i forsøget er afprøvet med en størrelse på 1000 tegn med et overlap på 200 tegn, vist på *Figur 30*. Disse fragmenter omdannes til vektorembeddings, som derefter lagres i VDB'en.



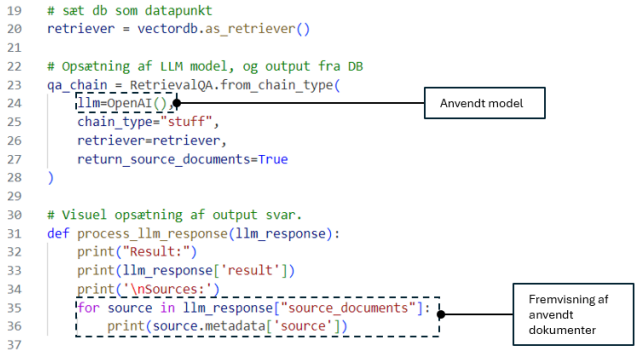
Figur 30 - Fragment splitter - Udklip af Bilag 23

Funktionen *retriever.invoke* anvendes til at fremsøge information baseret på brugerens prompt. Søgefunktionen er konfigureret til at returnere de to mest ensartede fragmenter ved at matche promptens vektorrepræsentation mod de lagrede embeddings i DB'en, som er fremvist på nedenstående *Figur 31*.



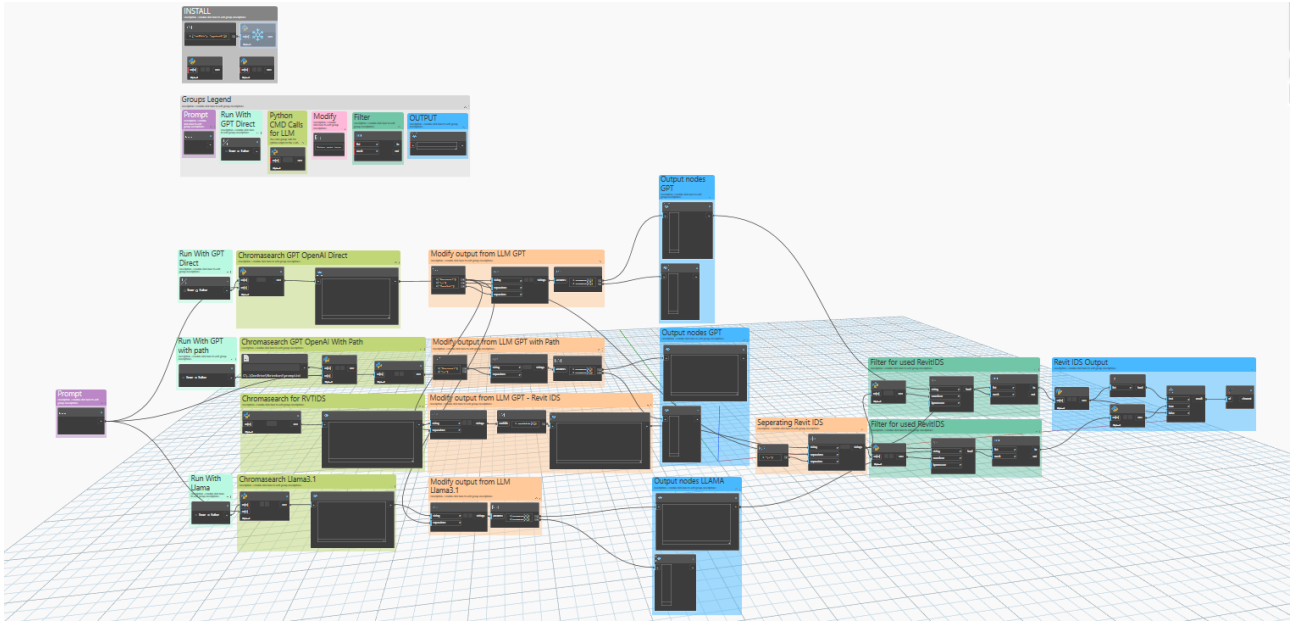
Figur 31 - Oprettelse af V DB, søgningsfunktion - Udklip fra Bilag 23

Den oprettede VDB vil herefter kunne tilgås af LLM'en, når brugeren laver forespørgsler. Fremgangsmåden for sammensætningen af VBD'en og LLM'en kan ses i *Bilag 27 og 28*. Ligesom andre tilsvarende scripts kræver søgningen i VDB'en, at de nødvendige Python-biblioteker importeres, API-koden indtastes, og den oprettede VDB indlæses, som illustreret nedenfor *Figur 32*.



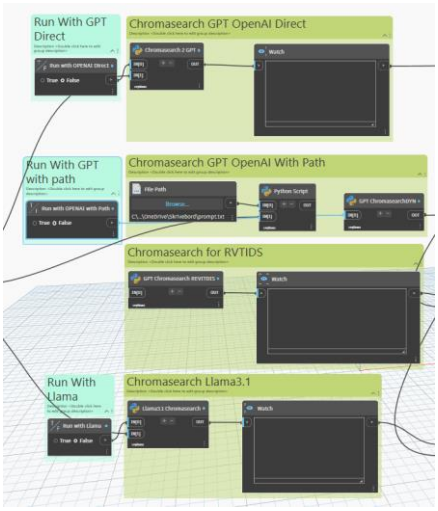
Figur 32 - Opsætning af LLM-respons - Udklip af Bilag 27

Når VDB'en anvendes til informationssøgning, omdannes brugerens forespørgsel til en query, der identificerer de mest relevante tekststykker i VDB'en. VDB'en returnerer derefter de to mest relevante tekstfragmenter. Da disse tekstfragmenter alene ikke nødvendigvis udgør en brugbar respons, bearbejdes de yderligere af LLM'en. Gennem NLP-processer analyserer og sammenfatter LLM'en de returnerede tekststykker og genererer en samlet og forståelig respons baseret på den hentede informationer og relevante kilder.



Figur 33 - Samlet Dynamo graf Bilag 32

Bilag 32 fungerer som en integreret softwareløsning, der baseret på prompt-input, kan aktivere ML-modellerne til at behandle forespørgsler i VDB'en. Hele Dynamo-grafen er fremvist på ovenstående *Figur 33*. Grafens opsætning er designet, så input- og outputværdier vises i et brugervenligt interface, hvilket i dette tilfælde er implementeret vha. Dynamo Player. Dynamo-grafens struktur er udviklet som et eksperiment, der undersøger metoder til effektiv kommunikation mellem LLM'en og VDB'en. I nedenstående *Figur 34* fremvises metoden for udvælgelsen af LLM'en samt, hvordan der kommunikeres med Python-script og ML-modeller. Gennem Dynamo Player anvendes *booleans* til at bestemme, hvilken LLM som anvendes.



Figur 34 - Udvælgelse af LLM - Udklip af Bilag 32

Kommunikationen mellem Dynamo, VDB'en og LLM'en etableres gennem to processer. En udfordring ved brug af flere modeltyper er, at hver ML-model kræver sit eget virtuelle miljø, hvilket skaber problemer at integrere dem direkte i Dynamo-grafen.

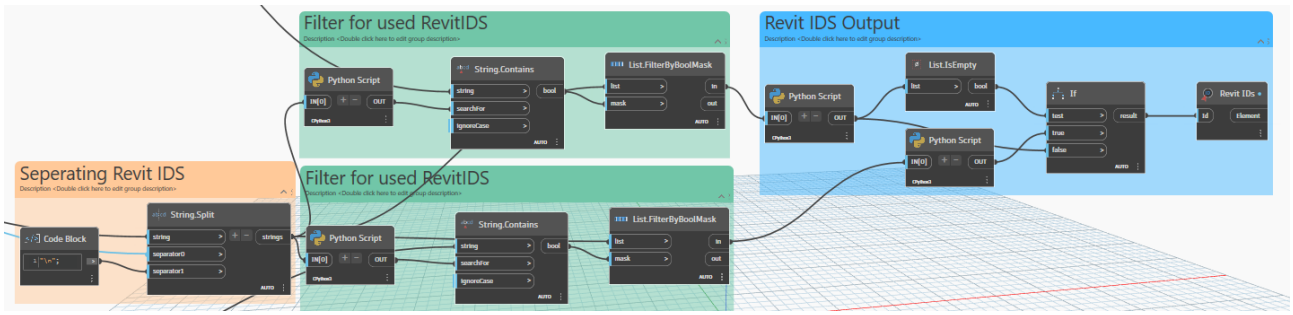
For at løse dette problem anvendes Python-biblioteket Subprocess, som gør det muligt at eksekvere Python-scripts via computerens kommandoprompt (CMD) eller Visual Studio Code.

Denne tilgang muliggør brugen af flere forskellige LLM'er, der afhænger af forskellige versioner af Python-bibliotekerne, hvilket sikrer fleksibilitet og kompatibilitet på tværs af miljøer og modeller. Subprocess-scriptet er fremvist på *Figur 35*.

```python
import os
import subprocess

# Boolean condition input to determine if the script should run
run_condition = IN[0]  # Boolean input (True = run, False = stop)

# Check if the condition is False
if not run_condition:
    # Stop execution if condition is False
    OUT = "Execution skipped: run_condition is False."
else:
    # Proceed with the script if condition is True
    # Define paths
    directory_path = r"C:\Users\madss\OneDrive\Skrivebord\chroma"
    python_file_path = r"C:\Users\madss\OneDrive\Skrivebord\chroma\chromasearch2.py"
    python_executable = r"C:\Users\madss\AppData\Local\python-3.9.12-embed-amd64\python.exe"

    # Prepare input
    try:
        input_text = IN[1]  # Input from Dynamo
    except Exception:
        input_text = "default question"  # Fallback input

    # Set PYTHONPATH
    env = os.environ.copy()
    env["PYTHONPATH"] = r"C:\Users\madss\AppData\Local\python-3.9.12-embed-amd64\Lib\site-packages"

    # Run the subprocess
    command = [python_executable, python_file_path]
    process = subprocess.Popen(
        command,
        cwd=directory_path,
        env=env,
        stdin=subprocess.PIPE,
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
        text=True
    )
    # Pass input_text to subprocess and get output
    output, error = process.communicate(input=input_text)
    result = output if output else error

    # Output the result
    OUT = result
```

Figur 35 - Anvendelse af Subprocess til eksekvering af Python-script gennem CMD – Udklip af Bilag 32.

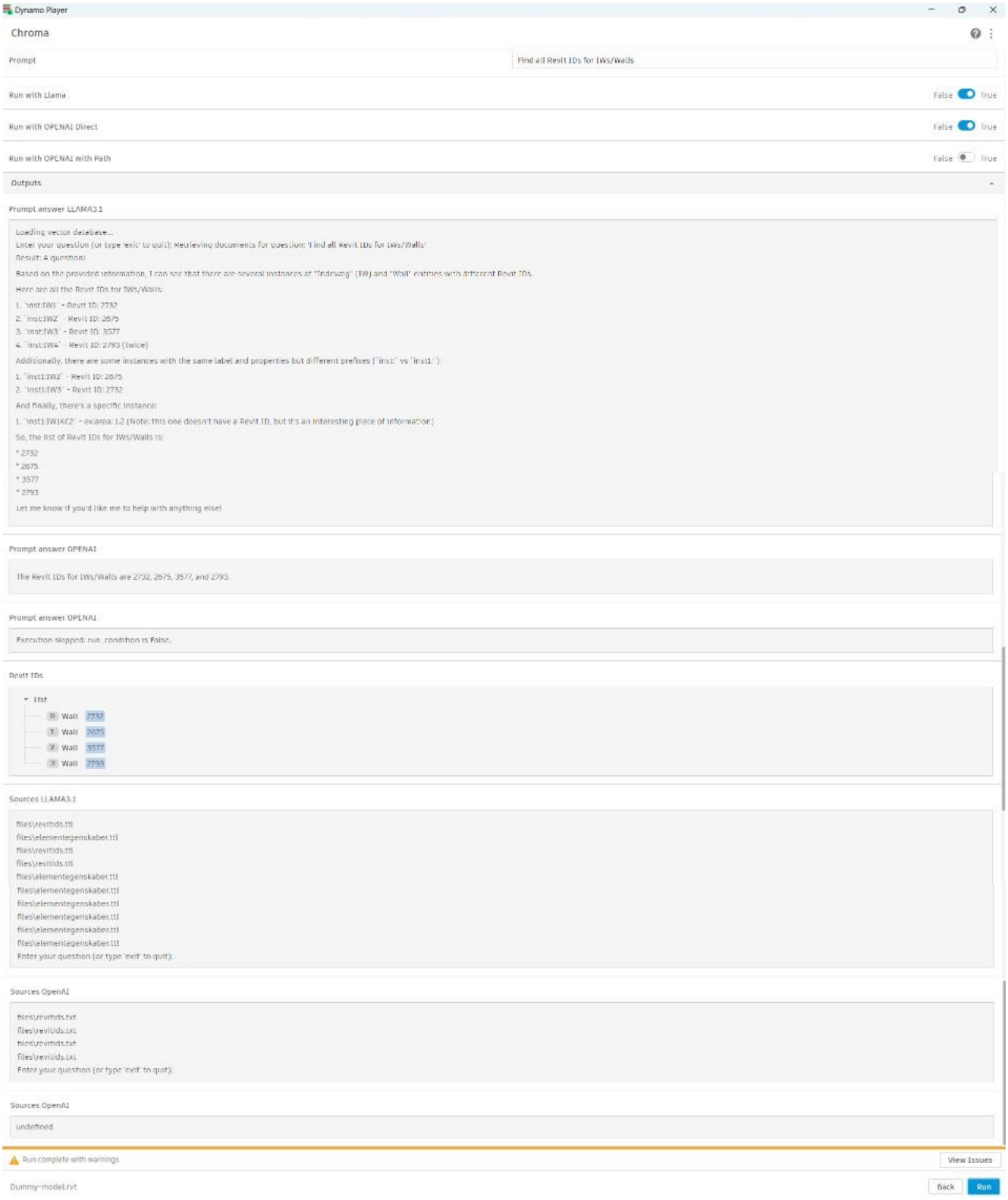
For at sammenkoble LLM'erne, VDB'en og BIM-modellen kræver det en direkte integration, der muliggør automatisk eller semiautomatisk manipulation af BIM-elementer efter en potentiel validering. Dette testes i prototypen vha. en LLM-søgning med en prædefineret query, Find all RevitIDs. De fremfundne RevitIDs kan derefter sammenlignes med de værdier, der returneres fra NL-prompten og LLM'ens respons. Således kan elementers ID identificeres igennem NLP. Metoden fremvises på nedenstående *Figur 36*.



Figur 36 - Sortering af RevitIDs - Udklip af Bilag 32

Prototypens interface præsenteres med mulighed for at anvende de forskellige LLM'er hver for sig. Det blev forsøgt at identificere alle Revit IDs for indervægge, der

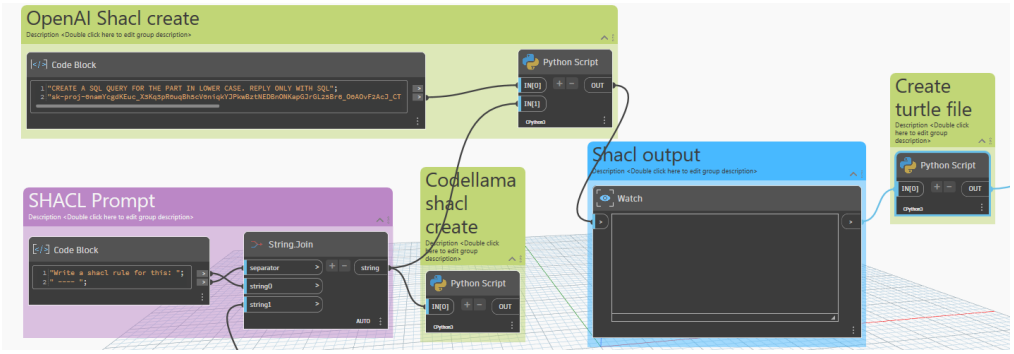
er placeret i den strukturerede data. Søgningen viser, at både Llama 3.1 og OpenAI fremfinder det korrekte antal indervægge og deres IDs. Den efterfølgende sortering af IDs er ikke tilfredsstillende nok, da der ikke kan interageres med BIM-modellen. Nedenstående *Figur 37* viser Dynamo Player ved udførelsen af grafen med Llama 3.1 og OpenAI. Her fremvises både responsen fra LLM'erne og de kilder, der er anvendt til at udarbejde responsen samt de Revit IDs, som er fundet. Behovet for at udspecificere, hvilke informationer der efterspørges, gør at systemets primære funktioner forsvinder, da der skal bruges viden om elementernes specifikke navne og egenskaber. Samtidig er der varians i om responsen indeholder den korrekte information.



Figur 37 - Dynamo Player ved kørsel af Bilag 32

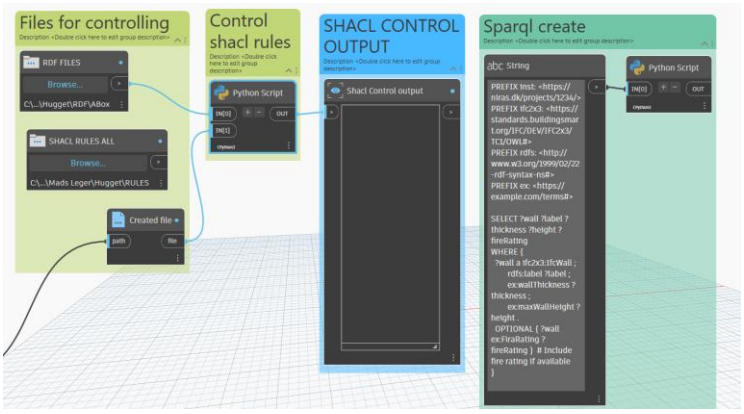
Bilag 33 præsenterer en modificeret udgave af Dynamo-grafen fra *Bilag 32*. Her undersøges det yderligere, om ML-modeller kan bruges til at omdanne NL til maskinlæsbare queries og regler, som kan anvendes på den lagrede BIM-data i VDB'en. *Figur 38* illustrerer, hvordan nye ML-modeller kan integreres til kodegenerering.

I denne opsætning kan LLM'ens respons anvendes til at generere SHACL-regler f.eks. vha. modeller som CodeLlama eller OpenAI 4o. De genererede regler kan derefter konverteres til .ttl-filer, som lagres i VDB'en for yderligere brug. Opsætningen vises på nedenstående F*igur 38*.



Figur 38 - Omdannelse af respons til SHACL-regler - Udklip af Bilag 33

Den oprettede .ttl-fil kan anvendes til at kontrollere BIM-elementernes egenskaber. For at indlæse SHACL-reglerne og udføre validering mod RDF-strukturen anvendes Python-bibliotekerne RDFlib og pySHACL. Prototypen bygger på ontologier og information, der i forsøget er baseret på fiktive semantikker. Derfor er syntaksen, i de opstillede SPARQL-queries og SHACL-regler, ikke funktionelt anvendelig i denne fase. Dette medfører et behov for mere avancerede LLM'er eller yderligere fine-tuning for at opnå tilstrækkelig driftssikkerhed og sikre, at regler og queries kan udføres korrekt i praksis. Et endeligt system vil derfor kræve en opsætning, hvor modeller kan omdanne NL-respons til korrekt SHACL og SPARQL, som fremvist på *Figur 39*.



Figur 39 - Udførelse af kontrol med SHACL, SPARQL queries - Udklip af Bilag 33

I prototypen er .ttl-filerne, der indeholder RDF- og SHACL-data, placeret lokalt. I en reel implementering bør disse data lagres i en GDB for at understøtte korrekt udførelse af SPARQL-queries. Da de nuværende data ikke er organiseret i en grafstruktur, er de eksisterende RDF-forbindelser muligvis utilstrækkelige ift. at sikre fuld understøttelse af queries og regler i en netværksbaseret kontekst. Integration i en GDB vil forbedre sammenhængen og muliggøre mere avancerede queries.

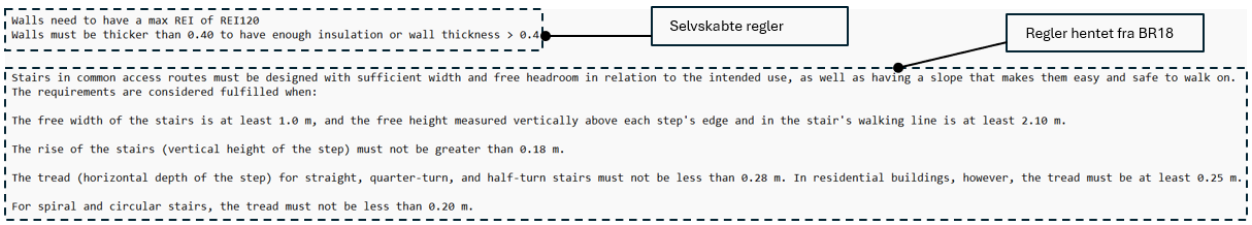
Validering af SHACL-regler og anvendelse af SPARQL queries

I *Bilag 34 og 35* er en række filer opstillet i hhv. .txt- og .ttl-format. Filerne der anvendes i projektet, er hentet fra ConTech Labs Pioner projekt. Her fremvises en række .ttl-filer med RDF-data, som samlet beskriver et mindre BIM-projekt og de interne sammenhænge. Filerne tager udgangspunkt i beskrivelser af egenskaber, rumprogram, bygningstopologi og produkter. Disse er repræsenteret i RDF og med tilhørende ontologier. Hertil er der yderligere opstillet regler omhandlende konsistens, geometri, lyd, brand- og fugtkrav. (Rasmussen, 2021). Et udklip af den anvendte LD, er fremvist på nedenstående *Figur 40*.

```
inst:IW2
    rdfs:label  "Indevæg 2"@da ,
                "Interior wall 2"@en ;
    ex:wallThickness 0.15 ;
    ex:maxWallHeight 2.7 ;
    rvt:RevitID 2675 .

inst:IW3
    rdfs:label  "Indevæg 3"@da ,
                "Interior wall 3"@en ;
    ex:wallThickness 0.15 ;
    ex:maxWallHeight 6.2 ;
    rvt:RevitID 2732 .
```

Figur 40 - Udklip af elementegenskaber.ttl - Bilag 34

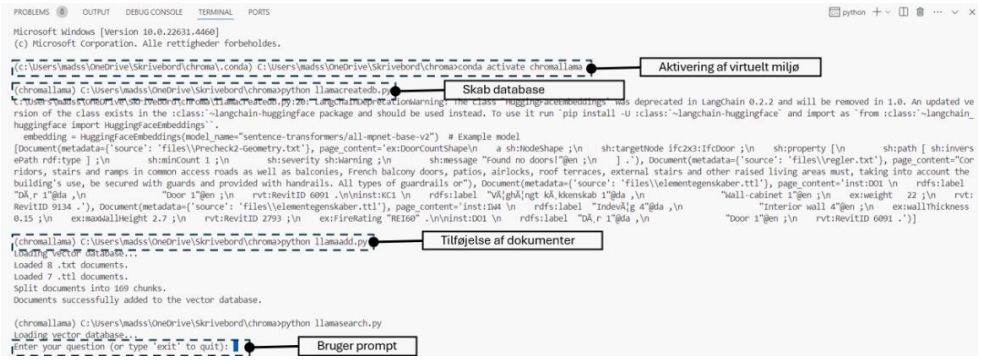
I forsøget er der opstillet en række fiktive projektkrav, hvor dele af BRs paragraffer er inddraget, til at undersøgelse, om NL kan bruges til validering. Da de anvendte LLM'er ikke er trænet på dansk, er kravene oversat. De opstillede regler og de eksisterende SHACL-regler udarbejdet af ConTech Lab kan ses på *Figur 41* (ConTech Lab, 2024; Social- og Boligstyrelsen, 2018).



Figur 41 - Udklip af regler.txt - Bilag 34

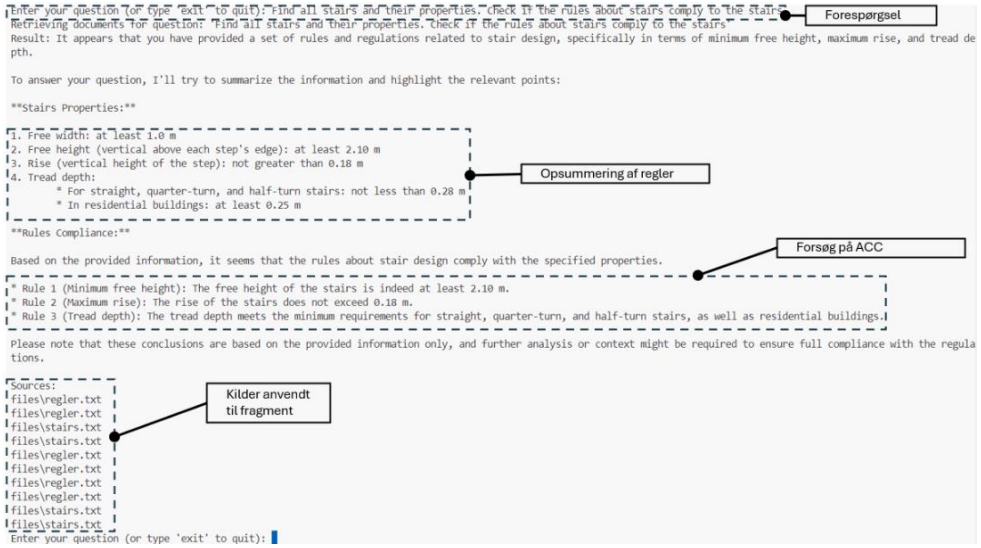
Formålet med prototypen er at kunne validere, om både strukturerede og ustrukturerede data på samme tid vil kunne anvendes til kontrol.

Til udførelse af Usability Testing er indholdet i *Bilag 28* anvendt i Visual Code Studio. Testforsøget blev udført for at verificere, om der vha. søgningen i VDB'en kan gennemføres kontroller baseret på både strukturerede og ustrukturerede data. Der anvendes det virtuelle miljø som fremvises i *Bilag 22*. Miljøet tillader oprettelse af VDB'er, tilføjelse af filer og efterfølgende søgning i VDB'en. Indlæsningen af det virtuelle miljø og fragmenter ses nedenfor på *Figur 42*.



Figur 42 - Opstart af Bilag 22, 24, 26 og 28.

Ved afprøvningen af prototypen blev der udført prompts (*Figur 43*), som kan betragtes som generelle og mindre specifikke ift. et element, dets regler og egenskaber.

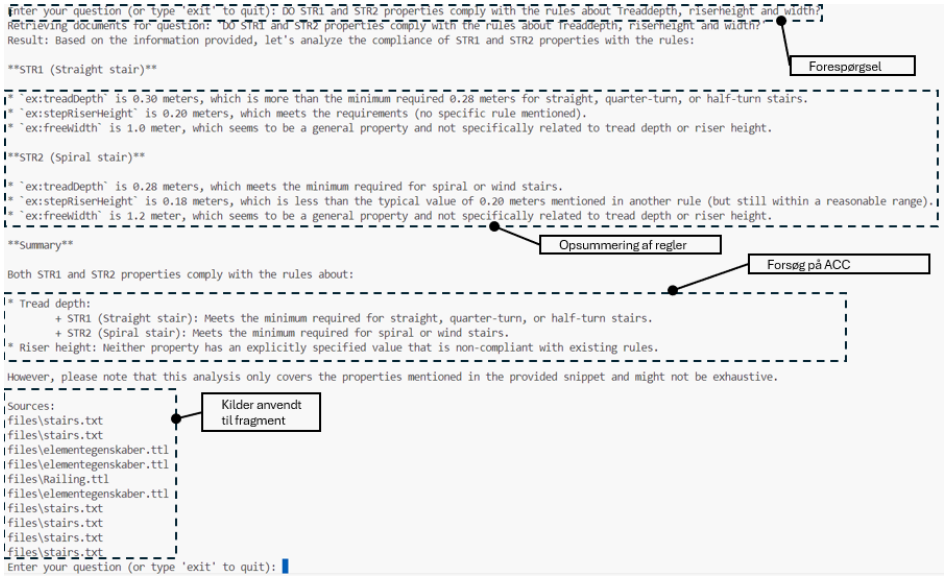


Figur 43 - Prompt forespørgsel på trapper og dertilhørende egenskaber

Forespørgslen er formuleret som en anmodning med det formål at indhente information om alle trapper og de tilknyttede regler hertil i projektet. Responsen fra LLM'en returnerer dog kun en generel opsamling af regler vedrørende trapper, som er formuleret i NL og ikke en korrekt kontrol. Den returnerede respons er korrekt ift. reglerne

om trapper, men den tager ikke højde for datastrukturens sammenhæng og validerer dermed ikke BIM-dataene.

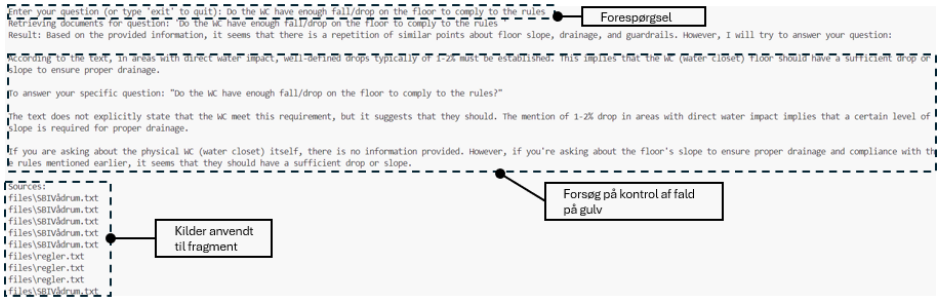
For at teste om LLM'en kan returnere korrekt information, blev de efterspurgte elementer, *STR1* og *STR2*, specificeret i forespørgslen sammen med de ønskede egenskaber. Egenskaberne blev dog ikke angivet med deres korrekte navne i forespørgslen, for at teste om LLM'en kunne opfange den korrekte navngivning. Dette fremgår af nedenstående *Figur 44*.



Figur 44 - Forespørgsel til LLM, med specifikke elementer og egenskaber

Prompten består af en efterspørgsel på krav til for en grund og en stigning for trapper samt bredden på trappen. Responsen fra LLM'en giver de korrekte elementer og egenskaber, som er efterspurgt. Yderligere er der, igennem systemets søgningen i dokumenter, fremfundet, at trappetypen har relevans for reglen for stigninger. Responsen fremhæver, at for typen *Spiral Type STR2* er højden på stigningen ikke opfyldt, som ses i kassen *Opsummering af regler*. Den efterfølgende opsummering giver dog en modstridende respons og godkender højden for stigningen. De anvendte regler i søgningen tages både fra regler i NL og fra RDF-datastrukturen.

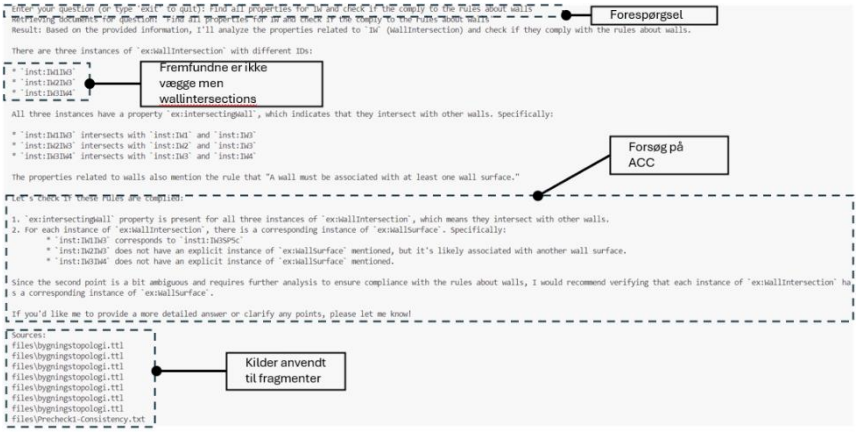
Som kontrol testes det om regler, som udelukkende beskrives som ustrukturerede data i NL, kan anvendes til kontrol. I filen *rumprogram (Bilag 34)* er *SP2* beskrevet som *Wetroom* og *WC*. Det testes, hvorvidt en beskreven egenskab, *ex:drop 2%,* kan kontrolleres igennem .txt-filen fra *Anvisninger* (Anvisninger, 2021). Forsøget vises i nedenstående *Figur 45*.



Figur 45 - Forespørgsel på fald i vådrum

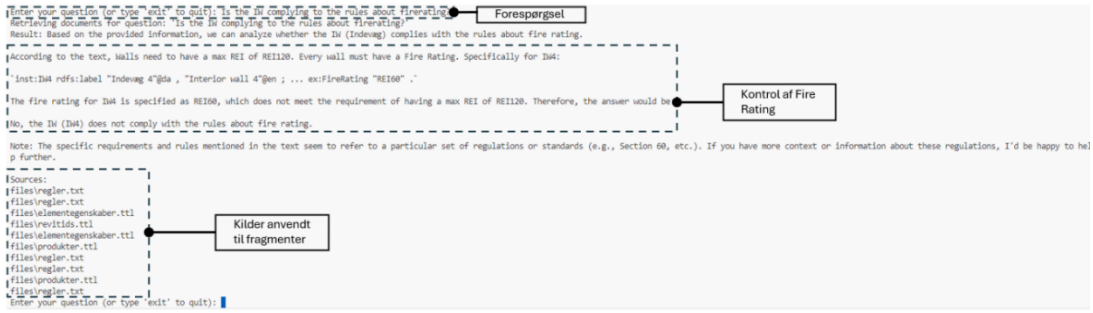
LLM'ens respons viser, at søgningen udelukkende er baseret på tekst og ikke inkluderer en kontrol af, om egenskaben findes i .ttl-filen for rumprogrammet. Der foretages et nyt forsøg for at undersøge, om responsen bliver mere præcis, hvis der søges på label og rumnummer. Prompten giver imidlertid den samme respons og udfører ikke kontrollen, men gengiver i stedet kravene som NL.

De ovenstående forsøg viser behovet for specifikke forespørgsler for at opnå brugbare responser. Når kontroller udføres uden at specificere elementer og egenskaber, tvinges LLM'en til at generere en respons ud fra ensartetheden i forespørgslen og teksten fundet i VDB. Dette illustreres i nedenstående *Figur 46*, hvor forespørgslen, om vægge overholder reglerne, returnerer en respons baseret på de fragmenter, der indeholder *IWs* flest gange. Her ses det i de anvendte kilder, at den kun har taget udgangspunkt i filen bygningstopologi.



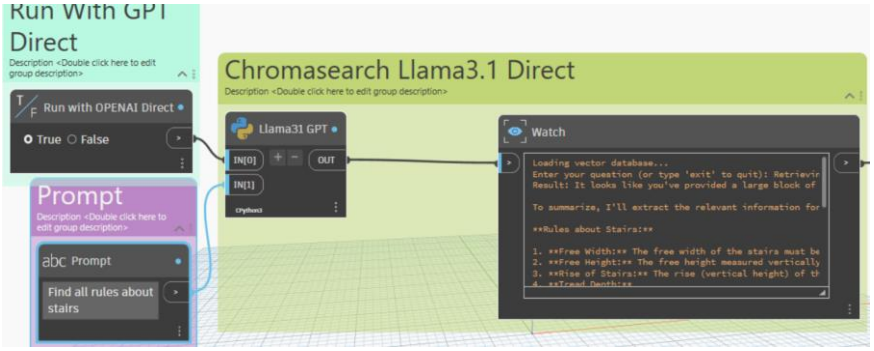
Figur 46 - Respons for forespørgsel om vægge og regler

Forsøget med trapper kan med præcisering af elementet og den tilhørende egenskab fører til en mere specifik og målrettet respons som vist i *Figur 47*.



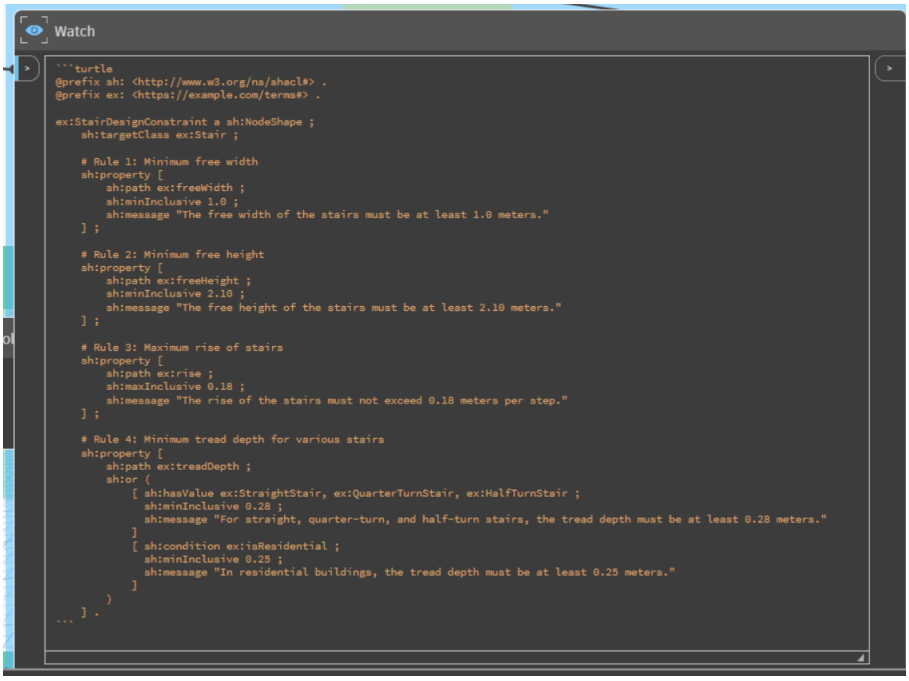
Figur 47 - Respons vedrørende IW4 og FireRating egenskab

Prototypen muliggør interaktion med fragmenterne i VDB'en og giver, vha. forespørgsler, mulighed for at udføre kontroller. Der er dog en usikkerhed om, hvorvidt kontrollerne udføres korrekt. Forespørgsler med større præcisering har vist sig at give mere nøjagtige responser, da disse skaber bedre relationer mellem informationerne, på tværs af filerne. For at konvertere prompts, som ikke er skrevet specielt præcise, anvendes *Bilag 33* til opsætning af regler i tekstformat til SHACL. De udførte regler er baseret på de overstående forsøg og anvender derfor ikke de korrekte ontologier, syntakser m.m., som fremvist på *Figur 48*.



Figur 48 - Anvendelse til LLM til SHACL-grundlag

Ved korrekt anvendelse af ontologier og syntaks kan en passende ML-model med relevant træning omsætte LLM-responsen til anvendelige SHACL-regler. På nedenstående *Figur 49* fremvises, hvordan indholdet af *Bilag 33* omsætter LLM responsen til SHACL-regler. Grundlaget for det anvendte materiale er præfikset *ex:*, *hvor ex* er en fiktiv ontologi kaldet example, der anvendes ved uspecificerede værdier. I et virkeligt system skal det sikres at anvende korrekt syntaks og passende ontologier. Eksemplet er udelukkende baseret på anvendelsen af SHACL og *ex*. Dette betyder, at de korrekte forbindelser til øvrige ontologier ikke er inkluderet, hvilket resulterer i, at reglen ikke kan udføre en fyldestgørende kontrol af dataene. Dette skyldes, at indlæsningen af *ex* ikke er defineret i sammenhæng med øvrige relevante ontologier såsom BOT, og derfor ikke kan danne parallel til disse.



Figur 49 - Genereret SHACL-kode af Bilag 33

8 TEST OG FEEDBACK AF DET NYE SYSTEM

I dette kapitel beskrives udførelsen af Usability Testing fra Interaction Design og den afsluttende del af CD, som er Interaction & Visual Design. Testen er udført ved, at Mockup'en af interfacet er fremvist for testpersonerne sammen med grundlæggende funktioner af den tekniske prototype. Derudover fremlægges de løbende overvejelser, som undersøgelsen har givet anledning til. Testen er foretaget med interviewperson C samt testperson F og G. Testperson F og G varetager de tekniske ansvarsområder som hhv. IKT-leder og Team Lead for BIM- og IKT-personale i samme virksomhed. Testpersonerne er udvalgt grundet deres tekniske færdigheder, og deres forståelse for udvikling og anvendelse af digitale værktøjer. Formålet med testen er at opnå en dybere forståelse af brugernes behov ved at analysere den modtagne feedback samt at evaluere systemet ved at teste dets funktioner gennem prototypen. Meningskondenseringen af testen kan findes i *Bilag 36*.

Gennem anvendelsen af Usability Testing, i dette kapitel, er den indsamlede feedback og dialog opdelt i to hovedkategorier: (1) Interface og (2) Tekniske prototype. I undersøgelsen er denne test den eneste, som er udført. Processen for CD og Interaction Design defineres gennem en iterativ proces, hvor feedback fra brugeren løbende indsamles. Denne data indarbejdes derefter i systemet og vil på et tidspunkt skabe grundlag for Field Testing. Da den iterative proces ikke har været mulig, indsamles feedbacken uden at blive indarbejdet i systemet. Feedbacken vil i stedet blive anvendt i diskussionen om systemet, protypen og fremtiden herfor.

8.1 Feedback

Interface

Testpersonerne blev først præsenteret for interfacet. I præsentationen blev det fremvist, hvordan brugeren skal åbne add-in'et i BIM-softwaret. Herefter blev systemets funktioner præsenteret, og hvordan forespørgslen til systemet skulle skrives ind, og hvordan outputtet, som LLM'erne responderer, opstår. Derudover eftervises, hvordan relevante BIM-elementer præsenteres som geometriske repræsentationer med tilhørende IDs/GUIDs, forespørgsler og relevante fragmenter anvendt til RAG, samt den genererede SHACL-regel ifm. valideringsforespørgsler. C og G påpegede, at de mener, der vil være behov for, i udarbejdelsen af systemet, at brugerne hjælpes på vej, når forespørgsler udføres. Systemet skal ud fra ord, genkendt i forespørgslen, kunne komme med kontraspørgsmål, som vil hjælpe brugeren på vej mod en mere retvisende forespørgsel. Dette skyldes, at hvis der udføres forespørgsler om krav eller regulativer, f.eks. vedrørende trapper, vil relationerne til projektet afhænge af den kontekst, hvori de er tegnet. Hertil understreges det, at hvis der skal undgås hallucinationer, vil der være et behov for, at der ikke laves formuleringsfejl, når der efterspørges på information. Efterfølgende blev der kommenteret på,

hvordan responsen og valideringen blev vist grafisk. Om dette uddyber G, at der er et behov for mere synlighed omkring de elementer og regler, der er anvendt til valideringen, og om disse bliver overholdt. Det er både svært at identificere, hvilke krav der anvendes i valideringen, og samtidig vurdere, om kravene overholdes i projektet. Derfor bør der anvendes advarselstegn eller andre markeringer, der gør brugeren mere opmærksom på fejl og mangler. Hertil understreges det, at det er dataene, der skal fremvises og ikke nødvendigvis den geometriske repræsentation af relevante BIM-elementer. G fremhæver, at evnen til at skabe relationer mellem døde dokumenter og live BIM-data kan tilføre stor værdi. Dog påpeger G, at det bør overvejes, om systemet udelukkende skal indføres i BIM-software, da dette kan ekskludere personer uden digitale kompetencer eller adgang til en licens.

Teknisk prototype

Den efterfølgende del af testen fokuserede på de tekniske funktioner samt strukturen i den tekniske prototype. F udstrålede en usikkerhed ift. sikkerheden ved besvarelser med udført pba. fragmenter. Her understreges det af testpersonerne, at når LLM'erne responderer, er behovet korte og præcise responser og ikke en sammenkobling af mange elementer, som vil medføre, at modellen vil danne vildledende besvarelser. Hertil uddyber F, hvorvidt mængden af fragmenter enten ville kunne danne bedre respons, eller om det ved en større mængde af data, vil danne fejltolkning af informationerne. Hertil efterspurgte F muligheden for at kunne justere på fragmentstørrelsen pba. datatypen og indholdet. Det uddybes med, at hvis der arbejdes med jura, eller projektspecifikke krav og oplysninger, som f.eks. blev oplyst i punktform, om disse vil kunne opdeles i mindre fragmenter og omvendt, at større tekststykker vil kunne defineres i større fragmenter.

G påpeger desuden, at regulativer, krav og vejledninger generelt ikke bør omdannes fra NL, men i stedet leveres direkte af de instanser og virksomheder, der udarbejder dem. Det handler mere om en brancheændring, der skal gennemføres via organisationer som Molio og DiKon, som skal bane vejen for en ny tilgang til udformningen af regler og regulativer. Her mener G, at ændringer i regelopsætningen bør baseres på enighed inden for branchen og ikke på individuelle virksomheders beslutninger om at ændre procedurerne. Indhentningen af krav, som enten strukturerede eller ustrukturerede data, skal dermed komme fra udgivernes egne DB'er og ikke en selvskabt løsning. Dermed vil kravene heller ikke kunne fejltolkes i samme omfang som egne genererede regler. Der er konsensus fra C, F og G om, at det vil kræve en modning i branchen og markedet, før det fulde potentiale i teknologien opnås.

C og G påpeger, at de, data der anvendes, bør være projektspecifikke, og at det kræver forsigtighed at kombinere data fra for mange forskellige projekter. Dette kan skabe usikkerhed om, hvordan responsen er dannet og øge risikoen for fejltolkninger. G placerer sig kritisk ift. om det ikke vil blive en manuel proces at udvælge den

data, som skal anvendes og sikre sig, at dette er korrekt og kan anvendes i den sammenhæng, der ønskes. Hertil spørger C, hvordan BIM-dataene skal danne grundlag i GPB'en. Alle tre ser en problematik i at skulle anvende data, som kommer direkte fra et Native dataskema, da det vil hindre muligheden for at inddrage andre BIM-software. En direkte inddragelse af BIM-dataene anses som en udfordring, og det vurderes, at der vil være behov for en oversættelse af dataskemaet til IFC og derfra til en RDF-struktur. Hertil pointerer F også, at det er nødvendigt, at der skal være enighed i branchen om, hvilket IFC-skema der skal anvendes.

Generelt ser C, F og G stort potentiale i teknologierne. Dog understreges det, at det bør betragtes som et understøttende værktøj og ikke som en beslutningstager. Her påpeger G, at det er fint, hvis det kan hjælpe dem på vej, men der skal sikres, at det aldrig gør noget, der ikke regnes med. Derfor mener G også, at det er svært at anvende systemet til kontroller, da dette ikke nødvendigvis vil give retvisende responser. Her pointeres det, at hvis f.eks. projektledere fik muligheden for at interagere med BIM-modellerne på en simpel måde, vil det kunne skabe en stor værdi. Normalt kræver det inddragelse af andet fagpersonale for at hente en mængde fra Revit, ved oprettelse af schedules. Denne proces kan optimeres og dermed bidrage til øget effektivitet. Det ville være nyttigt, hvis en almindelig medarbejder uden digitale kompetencer kunne stille spørgsmål om f.eks. betydningen af en klassifikation. Desuden skulle det være muligt at generere navngivninger uden at skulle sætte sig ind i de bagvedliggende funktioner eller forklaringer. G påpeger, at han ikke mener at et virkeligt system ville skulle etableres i et BIM-software som start, men derimod i et software der er tilgængeligt for alle brugere, der ikke vil kræve en type af licens. F påpeger, at det skal anses som værende en rygdækning for projektledere eller andre medarbejdere, som ville kunne stille spørgsmål og få informationer uden at have kompetencerne til at indhente disse på anden vis. Dette vil både være henvendt til en hurtigere søgning i projektmaterialets døde dokumenter, men det vil også fungere som direkte kommunikationsplatform med BIM-modellerne.

9 DISKUSSION

Bearbejdningen af litteraturstudiet og den indsamlede empiri har udgjort et omfattende analysearbejde, der dannede grundlaget for at identificere de centrale brugerbehov og systemkrav, som det nye system skulle imødekomme. Disse behov og krav blev anvendt til at formulere og illustrere en klar vision for systemets funktionalitet og dets rolle i brugernes arbejdsproces. Visionen fremhævede, hvordan systemet kunne integreres, som et redskab i brugernes fremtidige workflow, for herved at understøtte arbejdsopgaven mere effektivt, ved at imødekomme de identificerede udfordringer. De tiltænkte tekniske elementer i systemet blev illustreret gennem diagrammer og testet vha. udviklede prototyper. I dette kapitel vil det blive diskuteret og vurderet, hvorvidt det er muligt at udvikle et system, der kombinerer AI, SWT og herunder LD med den intention, at brugeren kan interagere med projektinformation, BIM-modeller og gældende regulativer vha. NL og derved rationalisere ACC og informationssøgning. Diskussionen er opdelt i tre underafsnit, som diskuterer AI, SWT, LD og BIM som løsning på identificerede ACC-problematikker, muligheder og begrænsninger ved prototypeudviklingen og afsluttende et fremtidsperspektiv ift. det udviklede system.

9.1 ACC, AI, SWT, LD, og BIM som løsning?

Det fremhæves i indledningen, at SWT og LD understøtter et fleksibelt og udvideligt udviklingsmiljø, hvilket kan være fordelagtigt for udviklingen af nye regelsæt og funktionaliteter. Et relevant spørgsmål i denne sammenhæng er, hvad der ligger til grund for anvendelsen af LD, både generelt, men specielt ifm. et ACC-system, der bygger på SWT, LD, AI og BIM. En fordel ved LD er, at det giver brugeren mulighed for at modtage en respons på en forespørgsel og samtidig se, hvilken datakilde LLM'en har benyttet til at udarbejde responsen. En udfordring som tidligere har bremset mange forsøg på at udvikle ACC-systemer er reglernes opbygning, som kan ses som værende komplekse og ustrukturerede. Derudover består den traditionelle informationssøgning af døde dokumenter, hvilket i sidste ende ikke er specielt effektivt ift. ACC-tankegangen, hvorfor ændringer er nødvendigt. Denne problematik imødekommes af et ACC-system, der bygger på førnævnte teknologier. Når det gælder regler og krav, bliver det muligt vha. forskellige LLM'er og en VDB at arbejde direkte med komplekse, ustrukturerede og døde dokumenter, såsom PDF'er, uden at disse data først skal forarbejdes og klargøres. LD-elementet i dette ACC-system opstår, når BIM-modellen (IFC) omdannes til LBD. Her omskrives IFC-datamodellen til ifcOWL, som repræsenteres som en RDF-graf, som derefter kan tilgås i en TS. Med en VDB som repræsenterer regulativer og projektkrav og en RDF-graf, som repræsenter en BIM-model, bliver det nu muligt at udføre ACC og lave queries på projektspecifikke data. Selvom et sådant system kan håndtere ustrukturerede og strukturerede dokumenter, efterspørger testpersonerne stadig, at branchen opfordres til at udarbejde

reglerne i et format, der er mere anvendeligt for de virksomheder, der skal efterleve dem. Det indebærer, at regulativer, krav og vejledninger ifm. et potentielt ACC-system ikke skal omdannes fra tekst i NL af virksomhederne selv. Det skal i stedet leveres direkte fra de offentlige instanser, der udarbejder dem. Dette vil medføre, at regulativerne ikke kan fejltolkes. Objektivt bliver et sådant ønskescenarie svært i en branche, som i forvejen er blandt de mindst digitaliserede sektorer i Danmark, med haltende produktivitet og manglede digital parathed, som Strategien for digitalt byggeri pointerer. I forlængelse af testpersonerne fremhæver Interviewperson C vigtigheden af, at de der udarbejder regulativer i Danmark, gør det i formater, der er maskinlæsbare. Her fremhæves det, at det ville være ideelt, hvis reglerne fra de offentlige instanser var tilgængelige som SHACL fordi, det vil muliggøre tjek af RDF-grafer. Dette underbygges også i *Afsnit 4.4*, hvor SHACL-tilgangen anses for at være mest fordelagtigt og anvende i et ACC-system sammenlignet med de resterende sprog inkluderet i afsnittet. Interviewperson B underbygger pointen fra Strategien for digitalt byggeri, med at understreges, at der er lange udsigter til at et sådant scenarie bliver en realitet.

Hvad er rationalet for at vælge en VDB frem for en RDB, som er mere udbredt og velkendt blandt aktører inden for branchen? Interviewperson B understreger, at forskellen ikke nødvendigvis er særlig stor, men at der generelt ligger en større fleksibilitet i VDB'en og LD-principperne især, når det omhandler ustruktureret data. Samtidig påpeger interviewperson D, at LD for mange aktører er det en uvant måde at tilgå og forstå data på. Aktører er generelt mere fortrolige med at arbejde med data i tabeller, som RDB'er, frem for at håndtere data som triples. I *Afsnit 4.9* bruger flere prototyper VDB'er til håndtering af regulatoriske dokumenter. Modsat bruger prototyperne baseret på den stemmebaserede tilgang en RDB til håndtering af BIM-data. BIM-dataene udtrækkes fra BIM-modellen vha. Dynamo til et CSV-format, som lagres i den RDB, hvor SQL anvendes til queries. Der er mange måder at lagre de forskellige typer data på, hvorfor det på sin vis også kan være svært, uden yderligere forskning, at fremføre en velbegrundet konklusion på, hvilken DB der er bedst til formålet. Dog vurderes det pba. litteraturstudiet, systemudviklingen og erfaringer i den forbindelse, at der på nærværende tidspunkt er flest fordele i at håndtere regulatoriske dokumenter med VDB'er og BIM-data som RDF-grafer. Til at understøtte denne konklusion fremviser interviewperson E en velfungerende VDB, kombineret med en LLM, der består af virksomhedens materiale. Dette system har den fordel, at brugeren selv udgør en aktiv del af søgeprocessen, hvor VDB'en præsenterer de ti bedste resultater baseret på forespørgslen og materialet i VDB'en. Derved bevares den menneskelige vurdering stadig. Derudover påpeger interviewpersonerne, at der kan være et stort potentiale i at koble de ustrukturerede data, i døde dokumenter, med BIM-modeller, hvilket muliggøres ved anvendelsen af AI, SWT herunder LD. Af denne grund vurderes det, at disse teknologier har potentiale til at bidrage til øget

digitalisering og produktivitet i den danske byggebranche, hvilket kan styrke konkurrenceevnen i det danske erhvervsliv.

Der er dog udfordringer med implementeringen af disse teknologier. Udfordringerne tager sit udgangspunkt i den manglende digitale parathed og haltende produktivitet. En af de væsentligste problematikker opstår i relation til det manuelle arbejde, som i princippet elimineres gennem implementeringen af det nye system. Spørgsmålet er imidlertid, om der ikke blot opstår en ny form for manuelt arbejde i det nye workflow således, at det manuelle arbejde fortsat eksisterer, men blot i en anden form? Det understreges i *Kapitel 4*, *Afsnit 5.1* og af testpersonerne, at den menneskelige vurdering ikke blot kan fjernes. Derudover påpeger testpersonerne, at værktøjet skal fungere som en rygdækning for projektlederen eller et understøttende værktøj og ikke et decideret ACC-værktøj. Det vil kunne afhjælpe aktører i deres normale arbejdsgang og på den eliminere diverse manuelle arbejdsprocesser fordi, at informationer nu kan fremsøges med en enkelt forespørgsel. VDB'en lagrer data, hvor testpersonerne er nervøse for det manuelle arbejde, som vil opstå med at skulle udvælge, hvilken data virksomheden ønsker i VDB'en. Her det understreges, at de informationer der placeres i VDB'en udelukkende skal være virksomheds- eller projektspecifikke.

Ud over udfordringerne ved de manuelle processer opstår en væsentlig barirrer ift. de manglede kompetencer, som primært identificeres i litteraturstudiet. Dette er et af de fire centrale fokusområder i Europa Kommissionens samordnede plan for AI, hvor der sættes fokus på at fremme talent og færdigheder i anvendelsen af AI. Det samme fokus afspejler sig i både den danske *strategi for digital vækst* og *digitalt byggeri*. Det betyder derfor, at aktører inden for byggebranchen skal tilegne sig kompetencer til at kunne håndtere teknologier såsom AI, SWT herunder LD og BIM. Dette er en nødvendighed for, at dansk erhvervsliv skal kunne bibeholde en stærk konkurrenceevne i og med, at teknologiske gennembrud er en af de primære katalysatorer for produktivitet og forandring i verdenen. Dette understreges også af testpersonerne, som pointerer, at branchen skal modnes, hvis et system som dette skal anvendes i virkelige arbejdsgange. Dog ser testpersonerne systemet som noget, der vil skabe stor værdi for de medarbejdere, som ikke besidder digitale færdigheder. Normalt ville en medarbejder i testpersonernes afdeling udføre et informationsudtræk vha. et *Schedule* i BIM-modellen, hvilket kan være udfordrende for personer uden digitale færdigheder. Systemet vil kunne understøtte deres normale arbejde fordi, at en bruger, vha. NL, kan fremsøge informationer og udføre kontroller på egen hånd.

Det er tidligere belyst i *Afsnit 4.4*, at SMC er i stand til at håndtere de fem forskellige krav og regler, som programmerne blev evalueret ud fra i daværende studie. Problematikken ved denne type ACC-system er imidlertid, at det besidder visse begrænsninger, hvorfor andre software og teknologier anbefales. Et ACC-system der består af teknologierne AI, SWT, LD og BIM anses for at kunne imødekomme nogle af disse begrænsninger. Her udtaler testpersonener, at ACC efter deres mening ikke skal

indgå som en del af systemet, og at SMC, som de anvender til at udføre ACC i deres organisation, umiddelbart ikke foreløbigt vil kunne erstattes af et system, som det der udvikles i rapporten. De ser kun et behov for en assistent, som kan være behjælpelig med at fremsøge informationer og derved afhjælpe den enkelte medarbejder i sine daglige arbejdsopgaver.

9.2 Prototypeudvikling – Muligheder og begrænsninger

Formålet med udarbejdelsen af prototyperne var at fremvise de grundlæggende funktioner, som et virkeligt system skulle indeholde samtidig med, at der blev skabt paralleller til et fuldendt system. De grundlæggende forskelle på den tekniske prototype og et virkeligt system udformer sig i valget af ML-modeller, funktioner og træning, samt i udvælgelsen af software og programmer, der anvendes i prototypen, for at skabe en ramme for funktionerne

I den tekniske prototype er der taget udgangspunkt i brugen af alsidige LLM'er, som, med deres generaliserede træning, kan opfylde kravet om at demonstrere tekniske funktionalitet for testpersonerne. Yderligere er de anvendte LLM'er udvalgt med fokus på fleksibel anvendelse, hvilket betyder, at de kan benyttes både lokalt og gennem eksterne servere. Fælles for disse LLM'er er, at de er forholdsvis små, med en størrelse på 8 milliarder parameter. Dette medfører en mere begrænset forståelse af de tokens og inputs, som brugeren anvender. Valget af disse mindre modeller har desuden gjort det muligt at anvende dem lokalt. Udviklingen af et virkeligt system vil naturligvis involvere anvendelse af tjenester, der benytter eksterne modeller og processorkraft for dermed at imødekomme systemets kompleksitet. Anvendelsen af mindre modeller i prototypen har desuden haft betydning for, hvor godt LLM'erne har behandlet og forstået de forespørgsler, der blev anvendt. I *Afsnit 7.2 Oprettelse af modeller, datasæts og fine-tuning* fremvises et udklip af Llama 3.1 tokens. Ved gennemsyn findes der ikke direkte tokens, der refererer til dataene, som ville kunne understøtte bygningsdata eller ontologier. Dette kan i processen, hvor modeller skal bearbejde og forstå NL-forespørgsler og omdanne disse til f.eks. SHACL, SPARQL, resulterer i fejl, hvor modellerne hallucinerer. De alsidige modeller skaber samtidig en problematik ved ikke at være designet til specifikke opgaver eller funktioner. Dette understreges også af interviewperson B, som påpeger, at anvendelsen af AI-modeller kræver nøje og korrekt udvælgelse for at matche et systems specifikke funktioner gennem pipelines og med målrettet træning. Den manglende træning af LLM'erne har i prototypen gjort det umuligt at etablere en direkte forbindelse til en GDB. Prototypen anvender derfor LD i en sammenhæng, der ikke anses for optimal. LD er lagret i VBD'en, hvilket betyder, at valideringer eller forespørgsler på BIM-data, som LD, ikke udføres gennem faktiske validerings- eller forespørgselssprog, men i stedet som fragmentsøgninger. Prototypen danner derfor udelukkende grundlag for at fremvise funktioner men ikke for en korrekt bearbejdning af input og

respons herpå. Et virkeligt system vil derfor kræve et mere driftssikkert setup, som ikke opererer lokalt, og som anvender korrekte skræddersyede modeller og funktioner, der vil kunne udnytte det fulde potentiale af dataene lagret i TS'en.

Ved at implementere det nye system i brugernes arbejdsproces forventes en optimeret og effektiviseret udførelse af den undersøgte arbejdsopgave. Testpersonerne fremhæver dog i *Kapitel 8*, at der er flere problematikker forbundet med at fokusere på anvendelsen af systemet i et lukket miljø som BIM-software. Det eksisterende workflow, i *Afsnit 6.1*, viser et stort behov for inddragelse af IKT-teamet, som besidder digitale kompetencer til at hente informationer og data samt validere dem med udgangspunkt i BIM-modellen. En naturlig måde at kommunikere med bygningsdata og ustruktureret data gennem systemet vil overvejende kunne minimere antallet af personer, der skal involveres i arbejdsprocessen. Testpersonerne udtrykker dog en skepsis for ML-modellers konsistens, og derfor vil en egentlig eliminering af visse fagpersoner i workflowet muligvis ikke være aktuel, da det stadig vil kræve menneskelig validering af den information, som LLM'erne returnerer. Samtidig viser systemets kompleksitet ift. driftssikkerhed, funktionernes validitet, respons og træning, at det vil kræve flere kompetencer at vedligeholde og opsætte systemet. Det vil derfor kræve en betydelig mængde ressourcer at sikre, at både de anvendte systemer og teknologier udnyttes til fulde, og i denne sammenhæng at de relevante fagpersoner tilegner sig de nødvendige kompetencer.

En endelig udvælgelse af typer af LLM'er og funktioner samt evt. træning vil derfor skulle vurderes ud fra relevansen af modellernes pre-training. Interviewperson B fremhæver, at de ikke anvender fine-tuning af modeller i deres software, men specifikke funktioner igennem Pipeline-funktionaliteter. Det vil derfor kræve en omfattende mængde kompetencer at sikre, at systemets respons er korrekt ift. queries til VBD'en og TS'en. Dette understreges også i feedbacken fra testen ift. behovet for kontraspørgsmål ved forespørgsler. Når den almene bruger stiller spørgsmål, som muligvis ikke er dækkende ift. den givne sammenhæng, skabes der usikkerhed om, hvorvidt ML-modellerne vil danne fejltolkninger eller misforståelser. Samme usikkerhed opstår også ift., om systemets funktioner skal simplificeres og opdeles for at kunne understøtte enkeltstående opgaver. Testperson F mente, at behovet for geometrisk repræsentation er unødvendigt, da dette allerede findes i deres eksisterende softwares. Testpersonernes indikation om, at den grafiske repræsentation af interfacet bør fokusere mere på information og mindre på geometrisk repræsentation. Der sættes derfor spørgsmålstegn ved, hvad intentionen med systemet egentligt er, og der bør tænkes over, hvad de primære funktioner skal kunne for bedst muligt at understøtte brugerne.

Det vil dog ikke udelukkende være en virksomhedsdefineret beslutning, hvordan systemets backend opbygges. Interviewperson A og *Afsnit 4.1 og 4.2* fremhæver, at en direkte anvendelse af BIM-data fra Native software ville være at foretrække, da det

ville muliggøre brugen af live BIM-data, hvor systemets respons vil kunne blive baseret på aktuelle data. Under testen udtrykte testpersonerne mistillid til anvendelsen af data fra Native dataskemaer, da de ikke naturligt vil kunne danne sammenhæng med øvrige BIM-software. Holdningen var derfor, at de nuværende tendenser kun muliggør anvendelsen af IFC som grundlag for konvertering af BIM-modeller til RDF-grafer. Ifølge testerpersonerne kræver det en præcisering af den type af IFC-skema, der udvælges på brancheniveau, for at identificere det mest hensigtsmæssige skema til repræsentation af LD.

De teknologier, der anvendes i prototypen, minder om dem fra tidligere prototyper beskrevet i *Afsnit 4.9*. Brugen af LLM'er til interaktion med GDB'er eller VDB'er fremstår som en metode til informationssøgning og kontrol af BIM-modeller. De opstillede prototyper i *Afsnit 4.9* repræsenterer brudstykker, som fremhæver relevansen og dermed argumentet for udviklingen af et virkeligt system. Det tekniske systems fremstilling og håndtering af de teknologiske brudstykker vil primært varier og afhænge af systemets arkitektur. De præsenterede prototyper understøtter anvendelsen af AI, SWT, LD og BIM og hjælper med at forstå, hvordan disse kan anvendes i et ACC-system, som det der er udarbejdet i undersøgelsen.

I *Afsnit 4.1* identificeres fire faser for et ACC-system: (1) Regelfortolkning (2) Forberedelse af BIM-model (3) Eksekvering af regler (4) Rapportering af regler. Et ACC-system, som det, der udført i denne undersøgelse, omfatter principielt alle fire faser, hvor størstedelen af processerne foregår automatisk. Det centrale i denne sammenhæng er, at alle faser udføres i backenden, mens brugeren, i frontenden, kun skal indtaste en forespørgsel i NL, hvorefter brugeren modtager en respons, hvilket svarer til den sidste ACC-fase. Således bliver det nemt for brugeren med få digitale kompetencer hurtigt at fremsøge projektspecifikke data og samtidig kontrollere dem. I samme afsnit defineres tre muligheder for, hvordan et regelbaseret vurderingsværktøj kan implementeres, hvor prototypen og interfacet tager udgangspunkt i at være et Add-in til Revit, som er én af de tre muligheder. Ifm. testen af prototypen understreges det, at det er væsentligt, at systemet ikke integreres i et lukket program som Revit, da dette begrænser adgangen for mange brugere. I stedet foreslås platforme som Microsoft-Teams, da disse er bredt tilgængelige og samtidig nemme at anvende for alle brugerne. Dette harmonerer med diskussionen i *Afsnit 9.1*, hvor det påpeges, at ACC-funktionen, set ift. testpersonerne, ikke bør integreres, som en del af systemet, men derimod begrænses til informationssøgning. I en sådan kontekst giver det mening at placere systemet på en platform, der er tilgængelig for alle. Omvendt hvis ACC-funktionen skal være en del af systemet, vil det være hensigtsmæssigt at implementere det i et program som Revit, hvor brugere med licens og ekspertise kan anvende systemet som en del af deres normale arbejdsgange.

Hvorvidt systemets endelige funktionalitet skal understøtte informationssøgning, ACC eller andet, vil derfor være afhængig af branchens og dermed brugerens behov

samt kompetencerne til at udvikle systemet. Testpersonerne anser systemets validitet som mere retvisende, hvis det anvendes som understøttende værktøj til de eksisterende workflows, hvilket kan øge effektiviteten og hastigheden på de allerede eksisterende workflows.

9.3 Fremtidsperspektiv

Der er grundlæggende identificeret diverse muligheder og begrænsninger i at udvikle et system, der kombinerer AI, SWT og herunder LD med den intention, at brugeren kan interagere med projektinformation, BIM-modeller og gældende regulativer vha. NL og derved rationalisere ACC og informationssøgning. Udviklingen af prototypen imødekommer både intentionerne fra Europa Kommissionen og de forskellige nationale strategier, hvorfor et sådant system er et skridt i den rigtige retning. Den nuværende status for anvendelsen af disse teknologier er grundlæggende på prototypeniveau, og kun få færdigudviklede systemer har gjort deres indtog i byggebranchen. Der er derfor også et stort potentiale for dansk erhvervsliv, og specielt byggebranchen, i at investere både penge, tid og forskning i disse teknologier. Det er samtidig også væsentligt at spørge, om nuværende standarder og reguleringer forbeholdt byggebranchen, er tilstrækkelige til at understøtte en teknologisk innovation som ACC bestående af førnævnte teknologier, eller om der er behov for yderligere harmonisering og tilpasning. Først og fremmest, som benævnt, er det nødvendigt at branchen går sammen om at definere standarder, hvorpå der i fremtiden skal defineres regulativer på. I *Afsnit 4.2* identificeres fire regeltyper og i *Afsnit 4.4* identificeres fem krav og regler, som et ACC-system vil skulle kunne håndtere. En problematik opstår om *Interference rules*, som vil kræve yderligere forskning i fremtiden i og med, at systemet har svært ved at håndtere disse regler. *Interference rules* refererer til geometriske kollisionskontroller mellem enkelte eller flere BIM-modeller og kan udgøre en udfordring, da LD-principperne ikke understøtter data repræsentationen for dette. De resterende regeltyper kan håndteres som alfanumeriske værdier, hvilket imødekommer LD-principper, hvorfor disse ikke er et problem for ACC-systemet. Dette stemmer overens med begrebet *Computationally Constraints* identificeret i *Afsnit 4.6*, der beskrives som udfordrende at kontrollere fordi, der opstår tekniske begrænsninger i f.eks. at håndterer geometriske data. Denne begrænsning underbygges yderligere i *Afsnit 4.6*, hvor det pointeres at, der er problematikker ved pålidelig lagring af detaljerede 3D geometriske data ved brug af LD-teknologier og herunder RDF-grafer. Dette er kritisk for en branche som byggebranchen, hvor data i det fleste scenarier har tilknytning til et geometrisk objekt. En væsentlig pointe fremhæves i denne kontekst, som opsummerer og adskiller to forskellige anvendelsesmuligheder af LD. Enten skal den geometriske udfordring løses, så LD kan anvendes i sin fulde kapacitet, eller også skal anvendelsen af LD begrænses til områder, der ikke omhandler geometriske data. Sidstnævnte løsning vil dog udgøre en betydelig begrænsning for formålet med det samlede system og prototypen, hvis et system, som dette,

potentielt skal erstatte et software som SMC, og dets anvendelse som ACC-system. Derfor vil den optimale løsning være at løse problematikkerne om geometrisk data og LD frem for fuldstændigt at udelukke geometrisk data fra systemet. Derudover fremhæves *Vague Constraints*, som ligeledes udgør en udfordring at kontrollere. Det indebærer uklare og upræcise regler og krav, der ikke er klart definerede og kan fortolkes på forskellige måder. Disse kan ikke kontrolleres ud fra en alfanumerisk værdi og er derfor svære at håndtere i et ACC-system, da en præcis regelfortolkning, hvor tvetydige definitioner gøres klare og målbare, vil være nødvendig. Derfor vil det kræve, at der i fremtiden forskes i, hvordan disse problematikker om regeltyper potentielt kan løses.

Ovenfor understreges et stort potentiale for dansk erhvervsliv, og specielt byggebranchen, ved at investere både penge, tid og forskning i disse teknologier. I denne forbindelse er der problematisk, at Danmark som nation placerer sig under de lande, som nationen normalt sammenlignes med, når det kommer til private investeringer i AI. Det kræver, at Danmark som nation og danske virksomheder tager et opgør med den manglende digitale parathed og bevidst vælger at investere i nye teknologier. På den måde kan de samtidig styrke virksomhedernes konkurrenceevne. For at opnå det fulde potentiale af især LD-teknologien vil branchen fremover skulle gøre op med den traditionelle filorienterede tankegang, nævnt i *Afsnit 4.6*, og i stedet fokusere mere på selve dataopbevaringen. For at et ACC-system baseret på AI, SWT, LD og BIM, kan implementeres og anvendes i fremtiden, vil det nødvendiggøre en række tilpasninger på flere niveauer – både nationalt, på diverse organisatoriske hierarkier i virksomheder og institutioner samt blandt de relevante aktører.

10 KONKLUSION

Hvorvidt det er muligt at udvikle et system, der kombinerer AI og SWT, herunder LD, hvor brugeren kan interagere med projektinformation, BIM-modeller og gældende regulativer, vha. NL, og derved rationalisere ACC og informationssøgning afhænger af flere faktorer. Undersøgelsen i rapporten viser, at det overordnet set er muligt at udvikle et sådant system. Konklusionen baseres på rapportens litteraturlitteraturstudie samt system- og prototypeudvikling. Systemet muliggøres først og fremmest ved at regulatoriske dokumenter opdeles i fragmenter og lagres i en VDB. Dernæst omdannes BIM-modellen til LBD, hvor IFC-datamodellen eksporteres til ifcOWL, som repræsenteres om en RDF-graf, som derefter kan anvendes i en TS. Derudover anvendes udvalgte programmer, værktøjer og biblioteker såsom, diverse LLM'er, Dynamo, Python, Visual Code Studio, Fine-tuning, LangChain, OpenAI, Embeddings, Subprocess, RDFlib og pySHACL. Anvendelsen af disse programmer, værktøjer og biblioteker har medført at, det har været muligt at udvikle en prototype, som udgør et grundlag for fremsøgning af projektinformationer og udførelse af ACC ud fra RAG-metoden, som bidrager til at minimere hallucinationer. Systemets opbygning muliggør, at brugere med manglede digitale og teknologiske færdigheder kan udføre forespørgsler og modtage respons i NL. Den begrænsede digitale parathed, der kendetegner byggebranchen, behøver i princippet ikke at udgøre en væsentlig udfordring, da fremsøgning af information kan foretages uden at ændre brugerens normale arbejdsprocesser. I og med at systemet udelukkende fungerer på prototypeniveau, og ikke udgør et færdigt system, kræver det yderligere systemudvikling, hvis et sådant system skal realiseres. Det indebærer, at udviklere skal besidde de rette kompetencer inden for programmering og håndtering af teknologier som AI, SWT og herunder LD. Derudover skal der udvælges LLM'er, der understøtter udvalgte Pipelines til at udføre ønskede funktioner således, at systemet i fremtiden simplificeres og strømlines. På nuværende tidspunkt udfører ACC-systemet kontroller baseret på BIM-data repræsenteret som IFC uden mulighed for direkte interaktion med en BIM-model. Dette skyldes, at direkte interaktion ikke har været teknologisk muligt, samt vigtigheden af at systemet skal baseres på software, der er tilgængeligt for alle involverede parter.

AI repræsenterer en fremtrædende teknologi, og fremtiden forventes at byde på betydelige teknologiske gennembrud og transformationer, som i sidste ende kan have en væsentlig indvirkning på konkurrenceevnen i dansk erhvervsliv. Både EU og Danmark er bevidst om potentialet i AI. Derfor kan det konkluderes, at det er afgørende for byggebranchen at integrere AI i relevante processer for at kunne følge med den globale udvikling. Samtidig er byggebranchen sammenlignet, med andre brancher, en af de mindst digitaliserede brancher, med haltende produktivitet og begrænset udvikling, hvorfor der er et klart behov for rationalisering. Det kan konkluderes, at det system, der er udviklet i denne rapport, vurderes og forventes at kunne

adressere disse problematikker og dermed bidrage til at øge byggebranchens samlede konkurrenceevne.

Afslutningsvist er det vigtigt at konkludere, at det ikke kun er den teknologiske del af systemet, der kræver yderligere udvikling. Der er et grundlæggende behov for, at branchen opnår enighed om, hvordan regulatoriske dokumenter skal udgives og i hvilket format. Derudover er det nødvendigt at arbejde på den digitale parathed, hvor branchen bør adoptere LD-principperne og fokusere mere på dataopbevaring frem for den filorienterede tilgang. Europa-Kommissionen og nationale strategier har allerede prioriteret anvendelsen af AI og nye teknologier. Det er nu op til virksomheder og relevante aktører at integrere disse teknologier og gøre dem til en del af deres eksisterende arbejdsgange.

11 REFERENCER

Aberer, K., Cudré-Mauroux, P., Choi, K.-S. & Noy, N., 2007. *The Semantic Web,* Busan: s.n.

Anaconda, 2017. *Conda*. [Online]
Available at: https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html
[Senest hentet eller vist den 2024].

Anon., 2016. *Methodologies for requirement checking on building models,* s.l.: Technische Universiteit Eindhoven..

Antoniou, G. & Harmelen, F. V., 2004. *A Semantic Web Primer*. 1 red. USA: Massachusetts Institute of Technology.

Anumba, C. J., Issa, R. R. A. & Wang, N., 2022. *Transfer learning-based query classification for intelligent building information spoken dialogue,* s.l.: ScieceDirect.

Anumba, C. J., Wang, N. & Issa, R. R. A., 2021. *A Framework for Intelligent Building Information Spoken Dialogue System (iBISDS),* s.l.: ResearchGate.

Anvisninger, 2021. *Anvisninger.dk*. [Online]
Available at: https://edu.anvisninger.dk/anvisninger/anv276-praktisk-udforelse-af-vadrum
[Senest hentet eller vist den 2024].

Berners-Lee, T., 2006. *W3C*. [Online]
Available at: https://www.w3.org/DesignIssues/LinkedData
[Senest hentet eller vist den 27 september 2024].

Berners-Lee, T., Bizer, C., Universität, F. & Heath, T., 2009. *Linked Data - The Story So Far,* Hershey: IGI Global.

Breitman, K. K., Casanova, M. A. & Truszkowski, W., 2007. *Semantic Web: Concepts, Technologies and Applications*. 1 red. London: Springer-Verlag.

Brickley, D. & Miller, L., 2014. *FOAF Vocabulary Specification 0.99*. [Online]
Available at: http://xmlns.com/foaf/spec/
[Senest hentet eller vist den 2 oktober 2024].

Burggräf, P., Dannapfel, M., Ebade-Esfahani, M. & Scheidler, F., 2021. *Creation of an expert system for design validation in BIM-based factory design through automatic checking of semantic information,* s.l.: s.n.

Carroll, J. M., 2010. *Interaction-design*. [Online]
Available at: https://www.interaction-design.org/literature/book/the-encyclopedia-

of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro
[Senest hentet eller vist den 2024].

Chambers, S., 2013. *Catelogue 2.0*. 1 red. London: Faces Publishing.

Chaudhri, A., Comet, A., Hanson, E. & Kumar, M., 2024. *SingleStore*. [Online]
Available at: Vector Databases & AI Applications
[Senest hentet eller vist den 2024].

Collidu, 2024. *Collidu*. [Online]
Available at: https://www.collidu.com/presentation-imrad-method
[Senest hentet eller vist den 2024].

ConTech Lab, 2024. *AI i byggeriet,* s.l.: ConTech Lab - en del af Molio.

ConTech Lab, 2024. *Molio*. [Online]
Available at: https://molio.dk/nyheder-og-viden/netvaerk/contech-lab

Coursera , 2024. *The History of AI: A Timeline of Artificial Intelligence*. [Online]
Available at: https://www.coursera.org/articles/history-of-ai?utm_medium=sem&utm_source=gg&utm_campaign=B2C_EMEA__coursera_FTCOF_career-academy_pmax-multiple-audiences-country-multi&campaignid=20858198824&adgroupid=&device=c&keyword=&matchtype=&network=x&devicemodel=&a
[Senest hentet eller vist den 9 December 2024].

Domingue, J., Fensel, D. & A. Hendler, J., 2011. *Handbook of semantic Web Technologies*. 1 red. Berlin: Springer-Verlag.

Duckham, M., Sun, Q. & Worboys, M. F., 2024. *GIS. A Computing Perspective*. 3 red. Boca Raton: Taylor & Francis group.

Eastman, C., Lee, J.-m., Jeong, Y.-s. & Lee, J.-k., 2009. *Automatic rule-based checking of building designs,* s.l.: s.n.

Elshani, D., Hernández, D., Lombardi, A. & Wortmann, D., 2023. *Building Information Validation and Reasoning Using Semantic Web Technologies,* s.l.: ResearchGate.

Erhvervsministeriet, 2018. *Strategi for Danmarks digitale vækst ,* København K: Erhvervsministeriet.

Erhvervsministeriet, 2023. *Redegørelse om vækst og konkurrenceevne ,* København K: Erhvervsministeriet .

Europa-Kommissionen, 2018. *Kunstig intelligens i Europa*.... [Online]
Available at: https://denmark.representation.ec.europa.eu/news/kunstig-intelligens-i-europa-2018-12-07_en?prefLang=lv
[Senest hentet eller vist den 12 December 2024].

Feigenbaum, L., 2009. *W3*. [Online]
Available at: https://www.w3.org/2009/Talks/0615-qbe/
[Senest hentet eller vist den 9 oktober 2024].

Finansministeriet og Erhvervsministeriet, 2019. *National strategi for kunstig intelligens ,* København K: Regeringen (Finansministeriet og Erhvervsministeriet).

Gashkov, A., Perevalov, A., Eltsova, M. & Both, A., 2022. *Improving Question Answering Quality Through Language Feature-Based SPARQL Query Candidate Validation,* s.l.: ResearchGata.

Ghannad, P., Lee, Y.-C., Dimyadi, J. & Solihin, W., 2019. *Automated BIM data validation integrating open-standard schema with visual programming language,* s.l.: Elsevier.

Google Cloud, 2024. *Google Cloud*. [Online]
Available at: https://cloud.google.com/learn/what-is-artificial-intelligence
[Senest hentet eller vist den 19 september 2024].

GraphDB, 2024. *graphdb.ontotext*. [Online]
Available at: https://graphdb.ontotext.com/
[Senest hentet eller vist den 2024].

gretelai, 2024. *Huggingface*. [Online]
Available at: https://huggingface.co/datasets/gretelai/synthetic_text_to_sql
[Senest hentet eller vist den 2024].

Guo, D., Onstein, E. & La Rosa, A. D., 2021. *A Semantic Approach for Automated Rule Compliance Checking in Construction Industry,* s.l.: IEEEAcces.

Hagedorn , P. & König, M., 2021. *BPMN-related Ontology for Modeling the Construction Information Delivery of Linked Building Data,* s.l.: ResearchGate.

Hagedorn, P., Pauwels, P. & König, M., 2023. *Semantic rule checking of cross-domain building data in information containers for linked document delivery using the shapes constraint language,* s.l.: Elsevier.

Hamdan, A.-H. & Scherer, R. J., 2020. *Integration of BIM-related bridge information in an ontological knowledgebase,* Dresden: Institute of Construction Informatics.

Hjelseth, E., 2016. *CLASSIFICATION OF BIM-BASED MODEL CHECKING CONCEPTS,* s.l.: ITCon.

Holdsworth, J. & Kosinski, M., 2024. *IBM*. [Online]
Available at: https://www.ibm.com/topics/vector-database
[Senest hentet eller vist den 2024].

Holtzblatt , K. & R. Beyer, H., 2005. *Interaction Design Foundation*. [Online]
Available at: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/contextual-design
[Senest hentet eller vist den 24 Oktober 2024].

Holtzblatt, K. & Beyer, H., 2017. *Contextual Design*. 2. red. Cambridge: Elsevier.

Huggingface, 2024. *Huggingface*. [Online]
Available at: https://huggingface.co/docs/transformers/en/training
[Senest hentet eller vist den 2024].

Huggingface, 2024. *Huggingface*. [Online]
Available at: https://huggingface.co/models
[Senest hentet eller vist den 2024].

Ismali, A. & Scherer, R., 2017. *Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC standard,* Dresden: ResearchGate.

Issa, R. R. A., Wang, N. & Anumba, C. J., 2022. *Query Answering System for Building Information Modeling using BERT NN Algorithm and NLG,* s.l.: ResearchGata.

Jackson, R. C. et al., 2019. *ROBOT: A Tool for Automating Ontology Workflows,* s.l.: BMC.

Justesen, L. & Mik-Meyer, N., 2010. *Kvalitative metoder*. s.l.:Hans Reitzels Forlag.

Kamath, U., Keenan, K., Somers, G. & Sorensen, S., 2024. *Large Language Models: A Deep Dive*. Cham: Springer.

Klyne, G., Carroll, J. J. & McBride, B., 2014. *W3C*. [Online]
Available at: https://www.w3.org/TR/rdf11-concepts/#dfn-node
[Senest hentet eller vist den 12 December 2024].

Krijnen, T. & van Berlo, L. A. H. M., 2016. *Methodologies for requirement checking on building models,* s.l.: Technische Universiteit Eindhoven.

Liu, F., 2024. *Zilliz*. [Online]
Available at: https://zilliz.com/learn/what-is-vector-database
[Senest hentet eller vist den 2024].

Liu, H. et al., 2017. *Enhanced Explicit Semantic Analysis for Product Model Retrieval in Construction Industry,* s.l.: ResearchGate.

Lund, H., Juhl, C., Andreasen, J. & Møller, A., 2014. *Håndbog i litteratursøgning og kritisk læsning*. 1 red. København : Forfatterne og Munksgaard.

Martiny, M., 2024. *Huggingface*. [Online]
Available at: https://huggingface.co/datasets/mabsmartin/IFCOWLBOT/viewer
[Senest hentet eller vist den 2024].

Martiny, M., Becirbasic, M., Nedergaard, J. & Risager, E. H., 2024. *Digital Kravsstillelse,* Aalborg: s.n.

Mckinsey & Company, 2024. *Mckinsey & Company*. [Online]
Available at: https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-ai
[Senest hentet eller vist den 24 september 2024].

Merrit, R., 2024. *Nvidea Blog*. [Online]
Available at: https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/
[Senest hentet eller vist den 2024].

Meta, 2024. *Huggingface.co*. [Online]
Available at: https://huggingface.co/unsloth/llama-3-8b-bnb-4bit/raw/main/tokenizer.json
[Senest hentet eller vist den 2024].

Miles, M. B., Huberman, A. M. & Saldana, J., 2018. *Qualitative Data Analysis*. s.l.:SAGE Publications.

Mitchell, M., 2019. *Artificial Intelligence A Guide for Thinking Humans*. 1 red. Milton Keynes: Straus and Giroux.

Mittal, A., 2024. *Unite.AI*. [Online]
Available at: https://www.unite.ai/da/at-forst%C3%A5-store-sprogmodelparametre-og-hukommelseskrav-et-dybt-dyk/?utm_source=chatgpt.com
[Senest hentet eller vist den 2024].

Mohammed, A., Abuoda, G. & Ghanem, A., 2021. *RDFFrames: knowledge graph access for machine learning tools,* Berlin: Technische Universität .

Molio, 2022. *Anvisninger Molio*. [Online]
Available at: https://anvisninger.molio.dk/gratis-vaerktojer/bygningsdelsspecifikationer/bds_shared_parameters

Nabavi, A., Ramaji, I. J., Sadeghi, N. & Anderson, A., 2023. *Leveraging Natural Language Processing for Automated Information Inquiry from Building Information Models,* s.l.: ResearchGate.

Neo4j, 2024. *Neo4j*. [Online]
Available at: https://neo4j.com/docs/getting-started/appendix/graphdb-concepts/
[Senest hentet eller vist den 2024].

Nepal, M. P., Staub-French, S., Pottinger, R. & Webster, A., 2012. *Querying a building information model for construction-specific spatial information,* s.l.: Elsevier.

Nuyts , E., Bonduel, M. & Verstraeten, R., 2024. *Comparative analysis of approaches for automated compliance checking of construction data,* s.l.: Elsevier.

Ollama, 2024. *Ollama*. [Online]
Available at: https://ollama.com/search
[Senest hentet eller vist den 2024].

OpenAI, 2024. *Openai*. [Online]
Available at: https://openai.com/api/pricing/
[Senest hentet eller vist den 2024].

Pauwels , P., Costin, A. & Rasmussen, M. H., 2021. *Knowledge Graphs and Linked Data for the Built Environment,* s.l.: Springer Nature Link.

Pauwels, P., van den Bersselaar, E. & Verhelst, L., 2024. *Validation of technical requirements for a BIM model using semantic web,* s.l.: s.n.

Perevalov, A., Gashkov, A., Eltsova, M. & Andreas, B., 2022. *Towards Knowledge Graph-Agnostic SPARQL Query Validation for Improving Question Answering,* s.l.: ESWC.

Petkova, T. & Kiryakov, A., 2021. *The Semantic Web: 20 Years And a Handful of Enterprise Knowledge Graphs Later*. [Online]
Available at: https://www.ontotext.com/blog/the-semantic-web-20-years-later/
[Senest hentet eller vist den 9 December 2024].

Python, 2024. *docs python*. [Online]
Available at: https://docs.python.org/3/library/venv.html
[Senest hentet eller vist den 2024].

Rademaker, A., 2012. *A Proof Theory for Description Logics*. Rio de Janeiro: Springer.

Rasmussen, M. H., 2021. *Github*. [Online]
Available at:
https://github.com/ConTechLabDK/PionerProjekt5/blob/master/readme.md
[Senest hentet eller vist den 2024].

Rienecker , L. & Jørgensen , P. S., 2022. *Den gode opgave*. 6 red. s.l.:Forfatterne og Samfundslitteratur.

Robinson, I., Webber, J. & Eifrem, E., 2015. *Graph Databases*. 2 red. Sebastopol: O'Reilly.

Rosenberg, J., Andresen, K. & Burcharth, J., 2016. *Systematisk review og meta-analyse*. 2 red. s.l.:Forskerkurser.dk .

Sajid, H., 2024. *Unite*. [Online]
Available at: https://www.unite.ai/da/hvad-er-retrieval-augmented-generation/
[Senest hentet eller vist den 2024].

Schlachter, A., Rasmussen, M. H. & Ernstsen, S. N., 2021. *Automated Rule Checking (ARC),* København: Niras A/S.

Schulz, O., Oraskari , J. & Beetz, J., 2023. *Lessons Learned from Designing and Using bcfOWL,* s.l.: ResearchGate.

Senthilvel, M. & Beetz, J., 2020. *A Visual Programming Approach for Validating Linked Building Data,* s.l.: ResearchGata.

Sharp, H., Rogers, Y. & Preece, J., 2007. *INTERACTION DESIGN beyond human-computer interaction*. 2 red. West Sussex: John Wiley & Sons.

Shin, S. & Issa, R. R. A., 2021. *BIMASR: Framework for Voice-Based BIM Information Retrieval,* s.l.: ASCE.

Shin, S., Lee, C. & Issa, R. R. A., 2022. *Framework for Automatic Speech Recognition-Based Building Information Retrieval from BIM Software,* s.l.: ASCE.

Ślusarczyk, G., Ismail, A. & Strug, B., 2018. *Building Knowledge Extraction from BIM/IFC Data for Analysis in Graph Databases,* s.l.: ResearchGate.

Social- og Boligstyrelsen, 2018. *Bygningsreglementet*. [Online]
Available at: https://www.bygningsreglementet.dk/tekniske-bestemmelser/02/krav/
[Senest hentet eller vist den 2024].

Stolk, S. & McGlinn, K., 2020. *Validation of IfcOWL datasets using SHACL,* Dublin: ADAPT Centre, Trinity College.

Teclaw, W., Kind, R. & Labonnote, N., 2024. *IFC properties validation using deep graph neural network,* s.l.: ResearchGate.

Textservice , 2023. *Kodning af kvalitative data*. [Online]
Available at: https://textservice.dk/kodning-af-kvalitative-data/
[Senest hentet eller vist den 12 December 2024].

Transport-, Bygnings- og Boligministeriet, 2019. *Strategi for digitalt byggeri ,* København K: Transport-, Bygnings- og Boligministeriet.

Triedman, H., 2023. *HuggingFace*. [Online]
Available at: https://huggingface.co/datasets/htriedman/wiki-sparql/tree/main
[Senest hentet eller vist den 2024].

Unsloth, 2024. *Colab*. [Online]
Available at: https://colab.research.google.com/drive/1WZDi7APtQ9VsvOrQSSC5DDtxq159j8iZ?usp=sharing
[Senest hentet eller vist den 2024].

Unsloth, 2024. *Github*. [Online]
Available at: https://github.com/unslothai/unsloth
[Senest hentet eller vist den 2024].

Vaswani, A. et al., 2017. *Attention Is All You Need,* Long Beach: Google.

W3C, 2013. *SPARQL Query Language for RDF*. [Online]
Available at: https://www.w3.org/TR/rdf-sparql-query/
[Senest hentet eller vist den 9 oktober 2024].

W3C, 2023. *W3C*. [Online]
Available at: https://www.w3.org/wiki/LinkedData
[Senest hentet eller vist den 27 september 2024].

W3C, 2024. *The Semantic Web Made Easy*. [Online]
Available at: https://www.w3.org/RDF/Metalog/docs/sw-easy
[Senest hentet eller vist den 27 september 2024].

Wang , N., Issa, R. R. A. & Anumba, C. J., 2022. *NLP-Based Query-Answering System for Information Extraction from Building Information Models,* s.l.: ASCE.

Wang, S., 2024. *puppygraph*. [Online]
Available at: https://www.puppygraph.com/blog/graph-database-vs-relational-database
[Senest hentet eller vist den 2024].

Xue, X., Zhang, J. & Chen , Y., 2024. *Question-answering framework for building codes using fine-tuned and distilled pre-trained transformer models,* s.l.: Elsevier .

Yuexin, H. et al., 2023. *Design knowledge graph-aided conceptual product design approach based on joint entity and relation extraction,* s.l.: Delft University of Technology.

Zhang , J. & El-Gohary, N. M., 2015. *Automated Information Transformation for Automated Regulatory Compliance Checking in Construction,* s.l.: ASCE.

Zhang, J. & El-Gohary, N. M., 2013. *Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking,* s.l.: ASCE.

Zhang, R. & EL-Gohary, N., 2021. *Semantic Representation Learning and Information Integration of BIM and Regulations,* s.l.: ASCE.

Zhang, R. & El-Gohary, N., 2023. *Transformer-based approach for automated context-aware IFC-regulation semantic information alignment,* s.l.: Elsevier.

Zheng, J. & Fischer, M., 2023. *Dynamic prompt-based virtual assistant framework for BIM information search,* s.l.: Elsevier.

Zhong, Y. & El-Diraby, T., 2024. *Generative project question answering system: triangulating three approaches for project authoring,* Marakesh, Morrocco: itc.scix.net.

Zhou, P. & El-Gohary, N., 2021. *Semantic information alignment of BIMs to computer-interpretable regulations using ontologies and deep learning,* s.l.: Elsevier.

Zilliz, 2022. *Zilliz*. [Online]
Available at: https://zilliz.com/learn/introduction-to-unstructured-data

Aarhus Universitet , 2024. *Aarhus Universitet*. [Online]
Available at: https://metodeguiden.au.dk/kodning-af-kvalitative-data
[Senest hentet eller vist den 12 December 2024].

Aarhus Universitet, 2019. *Metodeguiden*. [Online]
Available at: https://metodeguiden.au.dk/interviews
[Senest hentet eller vist den 2024].

Aarhus Universitet, 2019. *Metodeguiden*. [Online]
Available at: https://metodeguiden.au.dk/semistruktureret-interview
[Senest hentet eller vist den 2024].

Aarhus Universitet, 2019. *Metodeguiden*. [Online]
Available at: https://metodeguiden.au.dk/spoergsmaal-aabne-og-lukkede
[Senest hentet eller vist den 2024].

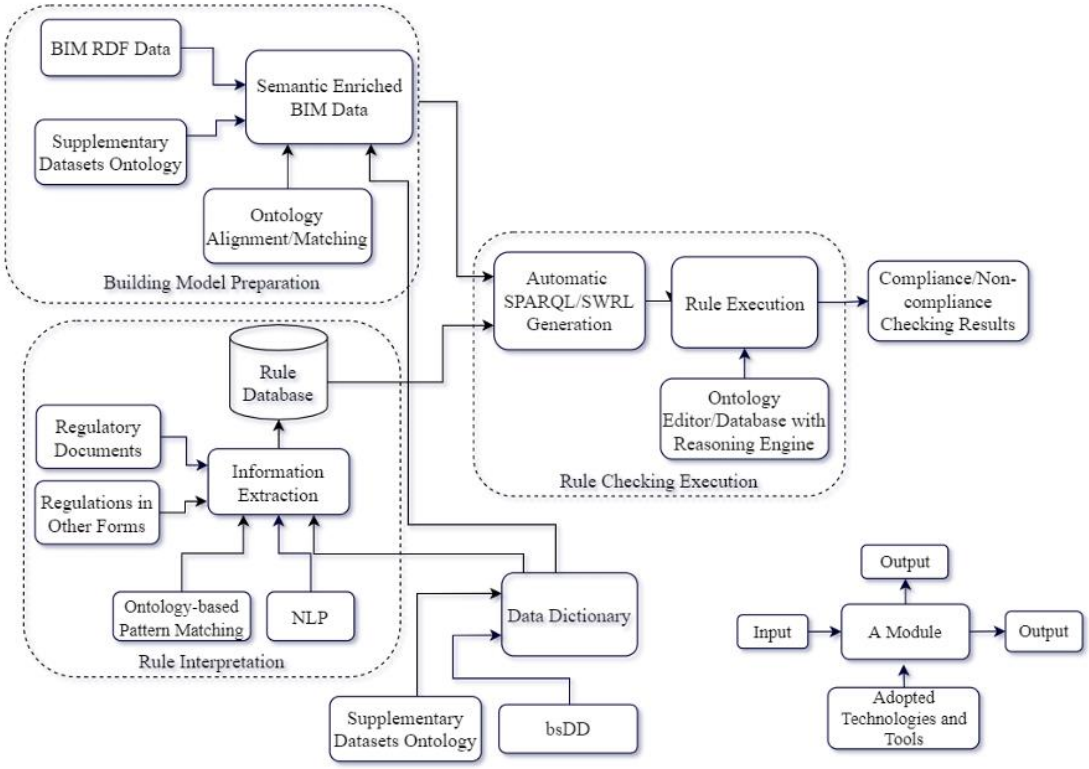
12 BILAG

12.1 Bilag 1 – Systematisk litteratursøgning, bloksøgning og tematisering

Bilaget er et Excel ark indeholdende tematisering af den systematiske litteratursøgning, bloksøgning.

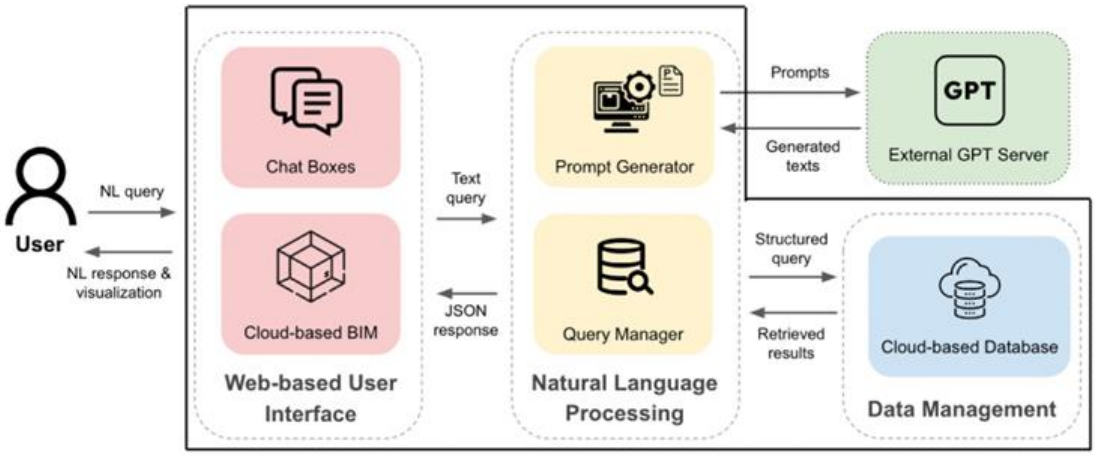
12.2 Bilag 2 – Prototypevisualisering (Guo, et al., 2021)

Diagrammet fremviser systemarkitekturen for prototypen for Guo, et al.



12.3 Bilag 3 – Prototypevisualisering (Zheng & Fischer, 2023)

Diagrammet fremviser systemarkitekturen for prototypen for Zheng & Fisher.



12.4 Bilag 4 – Prototypevisualisering (Zhang & El-Gohary, 2013)

Diagrammerne fremviser systemarkitekturen for prototypen for Zhang & El-Gohary.

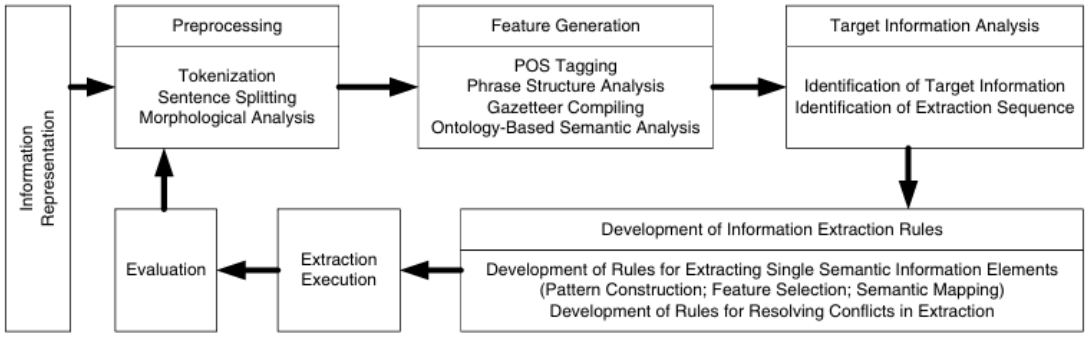
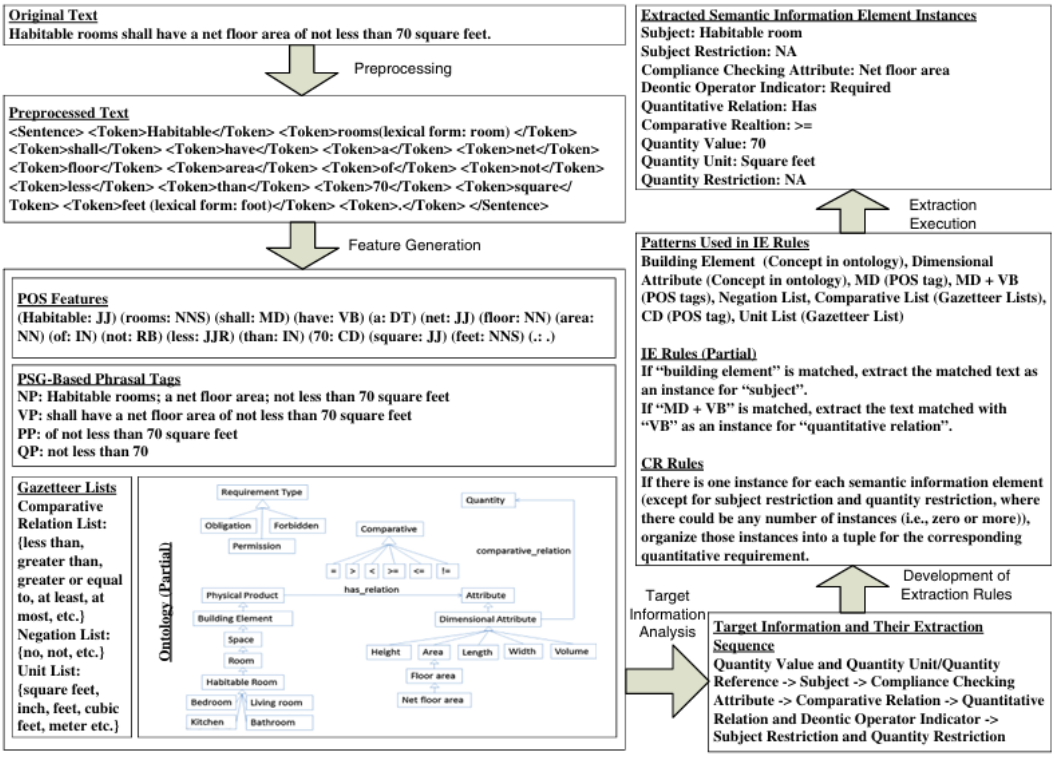


Fig. 2. Proposed information extraction methodology



12.5 Bilag 5 – Prototypevisualisering (Zhong & El-Diraby, 2024)

Diagrammet fremviser systemarkitekturen for prototypen for Zhong & El-Diraby.

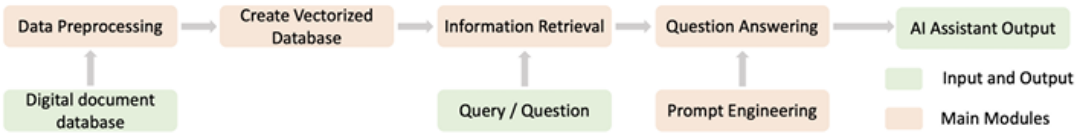
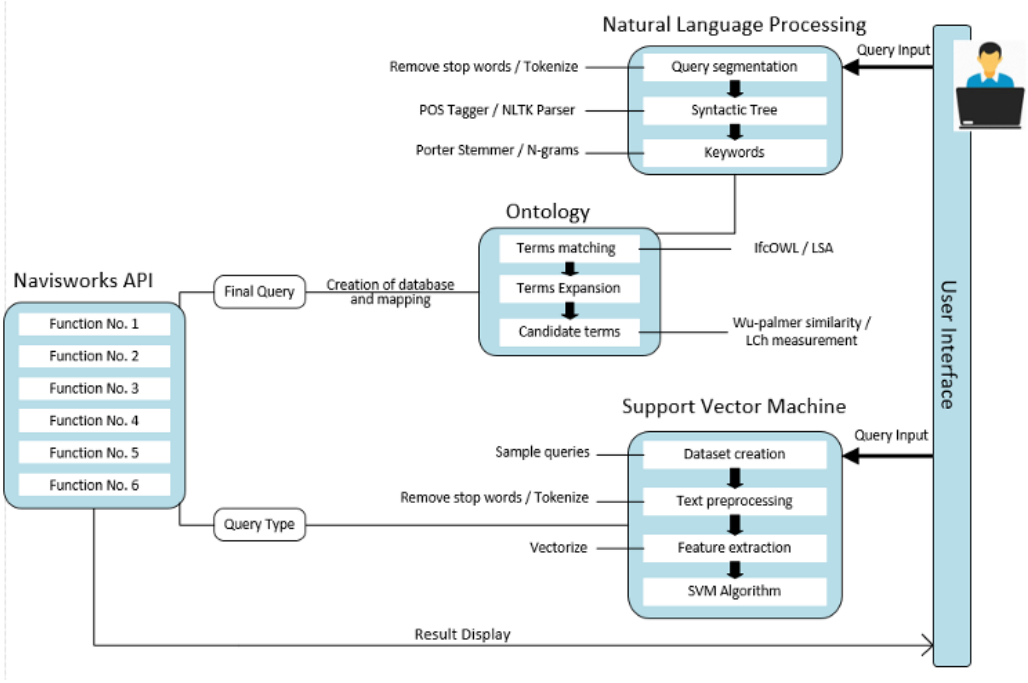


Figure 1: Overview of Methodology

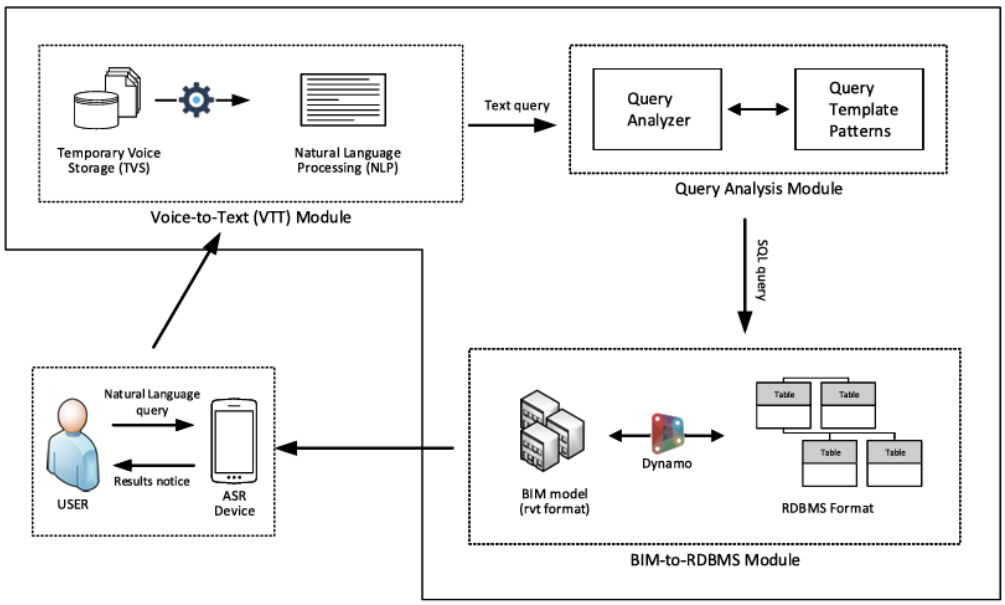
12.6 Bilag 6 – Prototypevisualisering (Nabavi, et al., 2023)

Diagrammet fremviser systemarkitekturen for prototypen for Nabavi, et al.



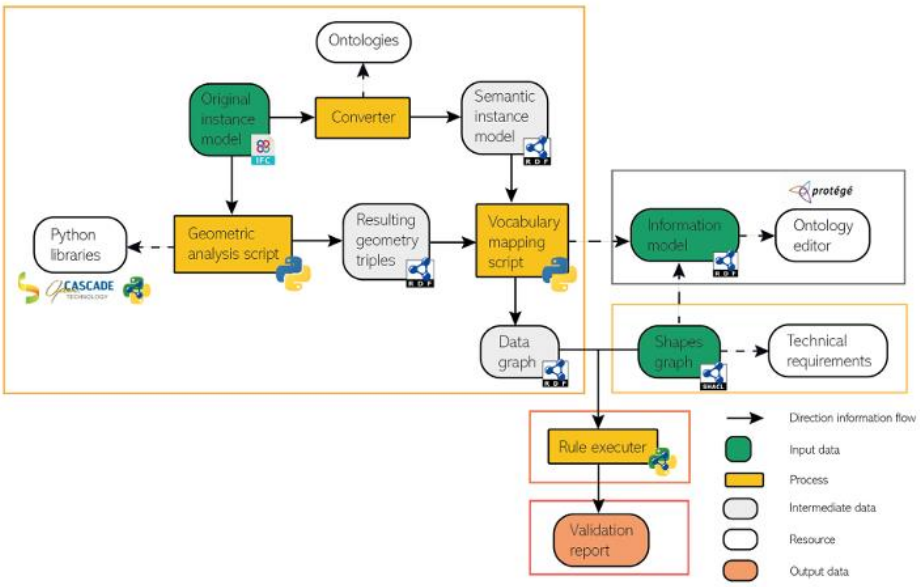
12.7 Bilag 7 – Prototypevisualisering (Shin & Issa, 2021)

Diagrammet fremviser systemarkitekturen for prototypen for Shin & Issa.



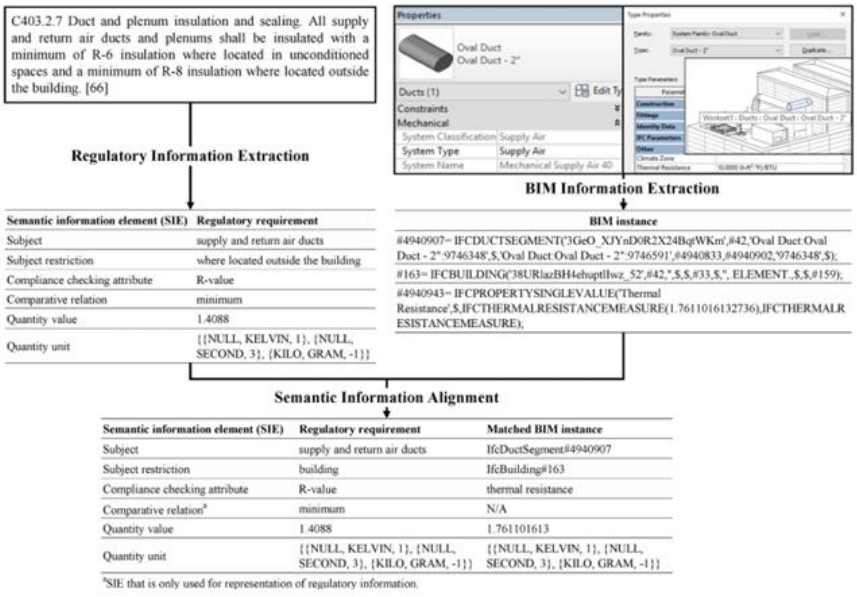
12.8 Bilag 8 – Prototypevisualisering (Pauwels, et al., 2024)

Diagrammet fremviser systemarkitekturen for prototypen for Pauwels, et al.



12.9 Bilag 9 – Metodevisualisering (Zhou & El-Gohary, 2021)

Diagrammet fremviser systemarkitekturen for prototypen for Zhou & El-Gohary.



12.10 Bilag 10 – Meningskondensering 1 – IP A

Interviewguide og meningskondensering af interviewperson A.

12.11 Bilag 11 – Meningskondensering 2 - IP B

Interviewguide og meningskondensering af interviewperson B.

12.12 Bilag 12 – Meningskondensering 3 - IP C

Interviewguide og meningskondensering af interviewperson C.

12.13 Bilag 13 – Meningskondensering 4 - IP D

Interviewguide og meningskondensering af interviewperson D.

12.14 Bilag 14 – Meningskondensering 5 - IP E

Interviewguide og meningskondensering af interviewperson E.

12.15 Bilag 15 – User Environment Design

Bilaget indeholder en billedfil af det udarbejdede User Environment Design.

12.16 Bilag 16 – Prototype af Interface

Bilaget består af et interaktivt PowerPoint som agerer som interface i undersøgelsens prototype.

12.17 Bilag 17 – Save to CSV Python-script

Dette Python script gemmer datasæt til CV, til efterfølgende fine-tuning

12.18 Bilag 18 – Fine-tuning converter Python-script

Dette Python script gemmer datasæt til CV, til efterfølgende fine-tuning

12.19 Bilag 19 – Datasæt parquet fil IFCBOT

Datasæt i formatet Parquet, indeholdende data vedrørende IFC, BOT og andre ontologier.

12.20 Bilag 20 – Fine-tuning for llm ipynb script

Dette Python script fine-tuner udvalgte modeller.

12.21 Bilag 21 – Push Ollama model terminal prompt

Bilaget indeholder kommandoer som skal anvendes i en terminal for at oprette en Ollama model lokalt.

12.22 Bilag 22 – Terminal prompt for Python Evn skabelse

Dette bilag indeholder kommandoer til oprettelse af virtuelle miljøer I Python.

12.23 Bilag 23 – Python-script VDB OPENAI

Dette Python script opretter en VDB med modeller skabt af OpenAI

12.24 Bilag 24 – Python-script VDB Llama 3.1

Dette Python script opretter en VDB med modeller skabt af Meta Llama

12.25 Bilag 25 – Python-script tilføjelse af filer til VDB OPENAI

Dette Python script tilføjer filer til VDB med modeller skabt af OpenAI

12.26 Bilag 26 – Python-script tilføjelse af filer til VDB Llama 3.1

Dette Python script tilføjer filer til VDB med modeller skabt af Meta Llama

12.27 Bilag 27 – Python-script søgning i VDB med OPENAI

Dette Python script indhenter fragmenter og anvender dem til en prompt søgning med modeller skabt af OpenAI.

12.28 Bilag 28 – Python-script søgning i VDB med Llama

Dette Python script indhenter fragmenter og anvender dem til en prompt søgning med modeller skabt af Meta Llama.

12.29 Bilag 29 – Python-script søgning i directory, med anvendelse af CodeLlama

Dette Python script søger i en samlet mappe, indholdende filer med relevans og omdanner denne til kode vha. modellen CodeLlama af Meta.

12.30 Bilag 30 – Python-script SPARQL query

Dette Python script anvendes stil at indlæse data i LD eller RDF-struktur, og opsætte det som grafer, som der kan udføres forespørgsler til. Denne tager udgangspunkt i RDFlib.

12.31 Bilag 31 – Python-script SHACL control

Dette Python script anvendes stil at indlæse data i LD eller RDF-struktur, og opsætte det som grafer, som der kan udføres forespørgsler til. Denne tager udgangspunkt i RDFlib.

12.32 Bilag 32 – Dynamo graf samlet vektor kald og llm

Grafen udvælgelse af de ønskede anvendte modeller. Hertil at kunne fremvise modellernes respons og videre bearbejdning.

12.33 Bilag 33 – Dynamo graf samlet vektor kald og llm, SPARQL/SHACL

Grafen tager udgangspunkt i Bilag 22. Grafen arbejder videre med at kunne lave SPARQL queries og validering igennem SHACL.

12.34 Bilag 34 – Zip fil, indeholder RDF, .ttl, .txt-filer

Disse filer anvendes som LD og ustruktureret data i prototyperne. (Rasmussen, 2021)

12.35 Bilag 35 – Zip fil, indeholder, .ttl, SHACL-filer

Disse filer anvendes som regler til validerings af LD fundet i Bilag 24. (Rasmussen, 2021)

12.36 Bilag 36 – Meningskondensering 6 – Test og feedback

Test og feedback med interviewperson C, og testpersoner F og G.

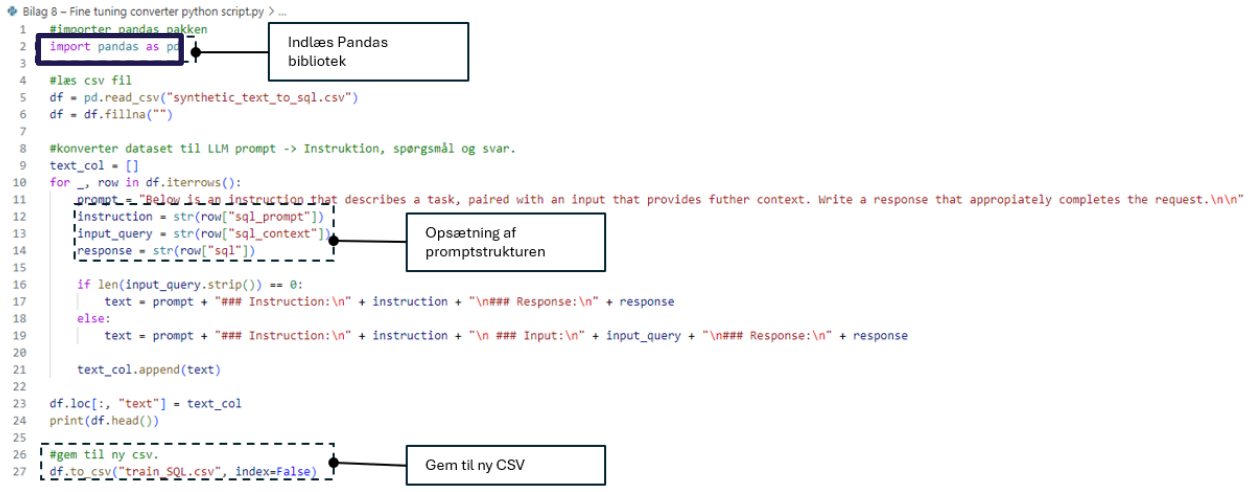
13 APPENDIKS

Teknisk beskrivelse af Bilag 17.

Figur 50 – Indlæsning af HF DB.

Adgangen til HF-DB'en for datasæt anvender Python-biblioteket Datasets, der tillader interaktion til de eksisterende datasæt. Funktionen der anvendes fra biblioteket, er *load_datasets*, som indhenter informationerne placeret i datasættet. For at kunne transformere datasættet til csv-format, anvendes Pandas biblioteket, der omdanner datasættet til en datatabel, som videre kan konverteres til csv.

Teknisk beskrivelse af Bilag 18.



Figur 51 - Indlæsning og opdeling af datasæt

Bearbejdningen og konverteringen af csv filen til promptstrukturen vil typisk bestå af en opdeling af *instruction, input/query* og *reponse/answer*. Dette er varierende fra model til model, og kræver derfor rettelse efter modeltypen. Opsætningen af den eksisterende datatabel fra csv indholdet fremstiller en ny tabel, der med udvalgte kolonner fra kan omdannes til promptstrukturen. Opsætningen kan herefter anvendes til fine-tuning af modellerne med udgangspunkt i de valgte kolonner i datasættet.

Teknisk beskrivelse af Bilag 19.

Bilag 19, består af et oprettet datasæt, der kan skabe forståelse for sammensætningen af forskellige ontologier, RDF-strukturer fra disse ontologier og deres sammenkobling til NLP. Datasættet er oprettet som *parquet* fil type, som understøttes af HFs online DB, til visuel fremvisning af indholdet. Datasættet er opdelt i kolonnerne, *ID, Ontology, Domain, Instruction, Input, Response, Explanation*. Kolonnerne *Ontology* og *Domain,* bruges til at repræsentere de relevante ontologier, såsom BOT, ifcOWL, OWL, SHACL etc. *Domain* anvendes til ontologiernes domainbeskrivelser, som f.eks. Ifc:*IfcWall, rdfs:list, bot:space* etc. De samlede ontologier er importeret til datatabellen og bruges derefter til at skabe instruktioner for ML-modellens respons samt til generering af spørgsmål og svar.

ID	Ontology	Domain	Instruction	Input	Response	Explanation
1	Prefix	Prefix RDF	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of RDF	http://www.w3.org/1999/02/22-rdf-syntax-ns#	For every RDF structure there are a need for an URI
2	Prefix	Prefix owl	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of owl	http://www.w3.org/2002/07/owl#	The `adjacentTo` relationship in BOT helps find rooms sharing common boundaries.
3	Prefix	Prefix rdfs	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of rdfs	http://www.w3.org/2000/01/rdf-schema#	Uses the `hasMaterial` relationship to link materials with structural components.
4	Prefix	Prefix xsd	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of xsd	http://www.w3.org/2001/XMLSchema#>	Leverages the hierarchical structure in BOT to navigate and count floors in a building.
5	Prefix	Prefix dcterms	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of dcterms	http://purl.org/dc/terms/	This is the prefix of dcterm
6	Prefix	Prefix vann	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of vann	http://purl.org/vocab/vann/	This is the prefix of vann
7	Prefix	Prefix voaf	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of voaf	http://purl.org/vocommons/voaf#	This is the prefix of voaf
8	Prefix	Prefix vs	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of vs	http://www.w3.org/2003/06/sw-vocab-status/ns#	This is the prefix of vs
9	Prefix	Prefix dce	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of dce	http://purl.org/dc/elements/1.1/	This is the prefix of dce
10	Prefix	Prefix dbo	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of dbo	http://dbpedia.org/ontology/	This is the prefix of dbo
11	Prefix	Prefix vcard	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of vcard	http://www.w3.org/2006/vcard/ns#	This is the prefix of vcard
12	Prefix	Prefix schema	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of schema	https://schema.org/	This is the prefix of schema
13	Prefix	Prefix bot	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of bot	https://w3id.org/bot#	This is the prefix of bot
14	Prefix	Prefix qudt	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of qudt	http://qudt.org/schema/qudt/	This is the prefix of qudt
15	Prefix	Prefix ex	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of ex	https://example.com/tezne#	This is the prefix of ex
16	Prefix	Prefix AAU	You are a helpfull translater for ontologies, so when asked i normal type of building element or related..	What is the prefix of AAU	https://eaubuild.com/speciale#	This is the prefix of AAU

Figur 52 – Udklip af selvskabt datasæt

Teknisk beskrivelse af Bilag 20 og 21.

Fine-tuning scriptet fra Unsloth er skrevet i Python, men i formatet ipynb som opstilles som Notebook format, der kan defineres som en funktionsopdeling af de enkelte celler. Formatet tillader terminal prompts, som anvendes til opsætning af Python miljøer til eksterne anvendte biblioteker (Python, 2024). Den præsenterede version, er den modificerede version af Unsloth scriptet (Unsloth, 2024).

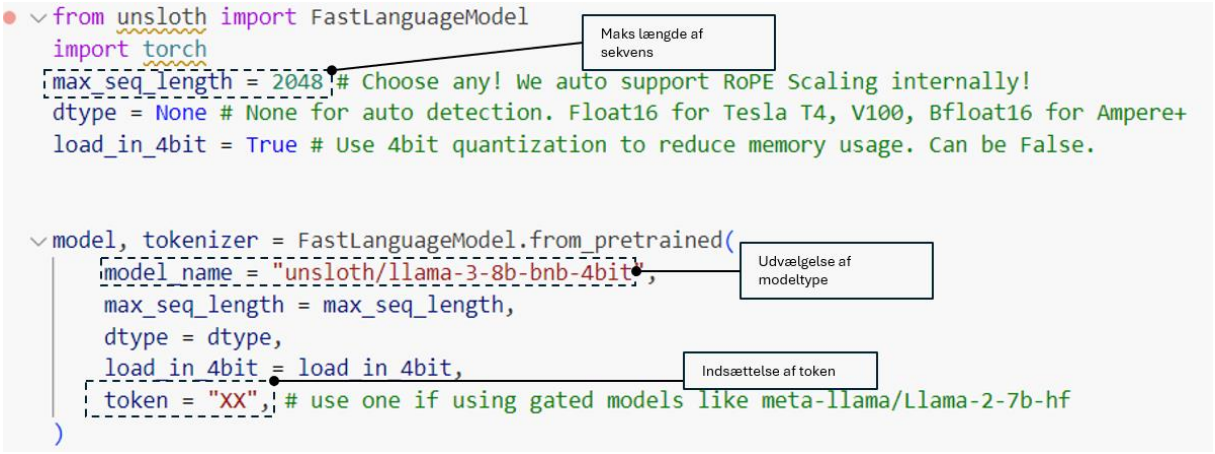
```
%%capture
# Installs Unsloth, Xformers (Flash Attention) and all other packages!
!pip install unsloth
# Get latest Unsloth
!pip install --upgrade --no-deps "unsloth[colab-new] @ git+https://github.com/unslothai/unsloth.git"
```

Installation af unsloth

Python

Figur 53 - Installation af Unsloth

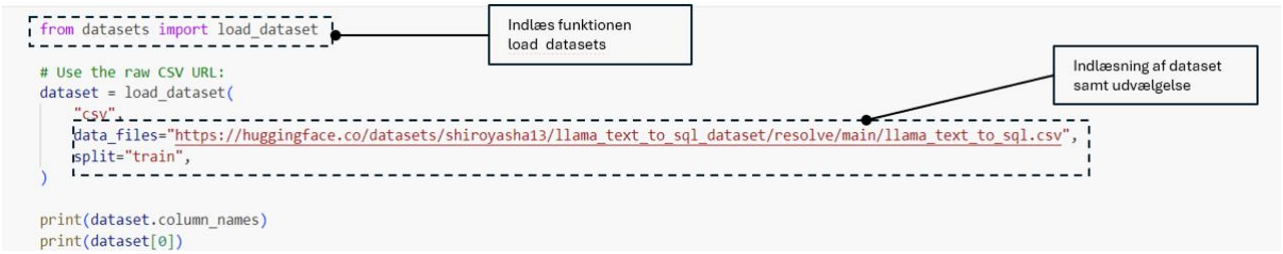
Første celler, anvendes som terminal prompt, hvor biblioteket Unsloth installeres. Dette anvendes igennem *pip install* funktionen. Hertil anvendes funktionen *–upgrade* med henvisning til Unsloth.git filen, som indeholder korrekte de Python-biblioteker, som understøtter de anvendte funktioner gennem scriptet.




```
from unsloth import FastLanguageModel
import torch
max_seq_length = 2048 # Choose any! We auto support RoPE Scaling internally!
dtype = None # None for auto detection. Float16 for Tesla T4, V100, Bfloat16 for Ampere+
load_in_4bit = True # Use 4bit quantization to reduce memory usage. Can be False.

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/llama-3-8b-bnb-4bit",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    token = "XX", # use one if using gated models like meta-llama/Llama-2-7b-hf
)
```

Figur 54 – Indlæsning af modeltype

Som en del af fine-tuningsprocessen kræves opsætning af maksimal længde for inputsekvenser og responsen, valg af ML-model og HF-token. Opsætningen udføres vha. Unsloth- og Torch-bibliotekerne. Under opsætningen genererer Torch en række config.json-filer, som fungerer som grundlag for modellens anvendelse, specifikationer og matematiske beregninger. Disse vil i anvendelsen ved HF-modeller, kunne anvendes til oprettelsen af egne ML-model.




```
from datasets import load_dataset

# Use the raw CSV URL:
dataset = load_dataset(
    "csv",
    data_files="https://huggingface.co/datasets/shiroyasha13/llama_text_to_sql_dataset/resolve/main/llama_text_to_sql.csv",
    split="train",
)

print(dataset.column_names)
print(dataset[0])
```

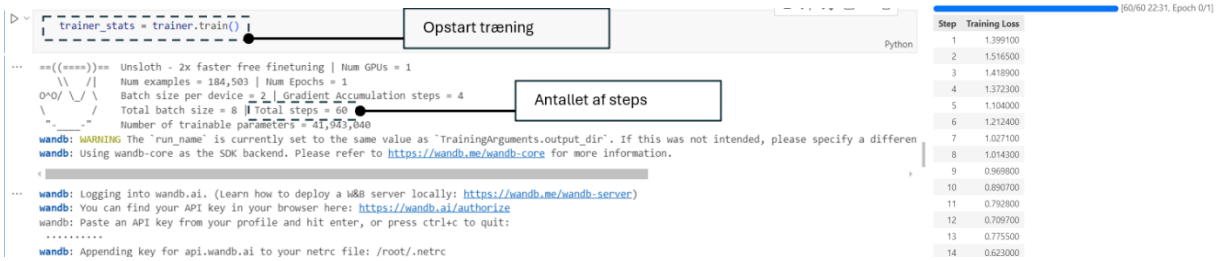
Figur 55 - Indlæsning af CSV-fil

Det udvalgte datasæt opstilles, så det kan splittes i kolonner, der bruges til promptstrukturen. Dette udføres vha. funktionen *load_datasets*.

```
from unsloth import to_sharegpt
dataset_simple = to_sharegpt(
    dataset,
    merged_prompt = "[[the question is {input}.\n]][[then answer ${output}.\n]]",
    output_column_name = "output",
)
```

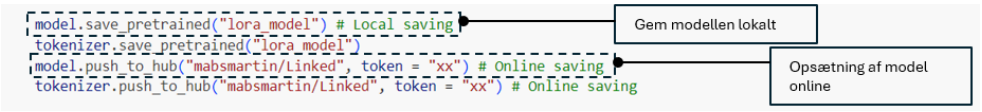
Figur 56 - Opsætning af Input output funktion

Promptstrukturen opstilles ved at kombinere input fra det valgte datasæt med det tilsvarende output for at skabe strukturerede sætninger, som ML-modeller bruger som grundlag for deres respons.



Figur 57 - Træning og training loss

Funktionen *trainer.train()* starter fine-tuningsprocessen for modellen. I opsætningen er der etableret en proces, hvor datasæt-informationerne opdeles i 60 trin. Igennem ML-modellernes tokenization proces, vises det igennem stepsne hvordan modellens tokenization af informationerne, tillære modellen informationer.



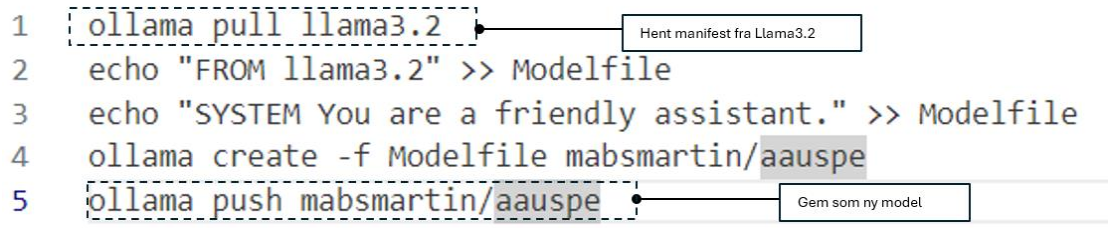
Figur 58 - Publish af model til HF-DB

Efter træningen kan modellen enten publiceres på HF-platformen til online brug, eller gøres tilgængelig som en lokal version.

```
!ollama create unsloth_modelcsv -f ./model/Modelfile
```

Figur 59 - Oprettelse af Unsloth model til lokalt brug

Modellen kan også samtidig blive published til Ollama serveren, og lokalt brug gennem Ollama applikationen. Modellen kan derfor oprettes og publiceres via *Bilag 21* som et terminalprompt.



Figur 60 - Indhent systemmanifest fra Llama 3.2

Systemmanifestet hentes fra den valgte model, og danner grundlaget for den modificerede model.

Teknisk beskrivelse af Bilag 22

Bilag 22 fremviser oprettelsen af virtuelle miljøer i Python, og anvendes til brugen af de to typer af LLM/vektor kald der anvendes i prototypen. Miljøerne oprettes i Anaconda Prompt, og anvender *conda* funktionen til oprettelsen. For prototyperne

oprettes miljøerne Chromaenv til brug med OpenAI 4.0 og Chromallama til brug med Ollama Llama 3.1.

```
//create with anaconda prompt and conda

conda create -n chromaenv
conda activate chromaenv
pip install Chroma langchain_chroma langchain_huggingface langchain_community openai tiktoken sentence-transformers chromadb langchain

conda create -n chromallama
conda activate chromallama
pip install Chroma langchain_chroma langchain_huggingface langchain_community ollama tiktoken sentence-transformers chromadb langchain
```

Figur 61 - Skabelse af virtuelle miljøer

Den primære forskel i miljøerne, er bibliotekerne OpenAI og Ollama. Disse biblioteker anvender afhængige underbiblioteker, som kan variere i versioner og derfor være afgørende for, om scripts kan køres korrekt.

Teknisk beskrivelse af Bilag 23

Bilag 23 opretter VDB'er baseret på OpenAI, og som anvender GPT 4o som LLM til bearbejdning af vektor outputs. De installerede biblioteker som installeres i *Bilag 22*, importeres til Python-scriptet, hvor de anvendte funktioner hentes.

```
import os
from langchain_chroma import Chroma
from langchain_openai import OpenAIEmbeddings  # Updated import
from langchain_community.document_loaders import DirectoryLoader, TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.llms import OpenAI
from langchain.chains import RetrievalQA
```

Figur 62 - Import af Python-biblioteker

For at bruge OpenAI-modeller kræves en købt API-nøgle, hvor betalingen afhænger af antallet af anvendte tokens. Ligesom med HF og Ollama findes der flere modeller, der er pre-trained og fine-tuned på forskellige datasæt. For at bruge en OpenAI-model skal API-nøglen inkluderes i scriptet.

```
# API kode til OpenAI
os.environ["OPENAI_API_KEY"] = "xx"
```

Indsættelse af API token

Figur 63 - Indsættelse af OpenAI Token

Før VDB'en kan oprettes, skal filmappen, der indeholder de data, der skal tilføjes, først defineres. Dette sker igennem funktionen DirectoryLoader, hvor ./files definere placeringen, og *glob* beskrives formaterne der skal læses i VDB'en. Dette gøres igennem funktionen TextLoader.

```
12  # Load dokumenter til anvendelsen i vektordatabasen
13  loader = DirectoryLoader("./files", glob="*.txt", loader_cls=TextLoader)
14  documents = loader.load()
```

Indlæsning af TextLoader

Figur 64 – Indlæsning af filer

VDB'en fungerer ved at de importerede tekstfragmenter opdeles og dannes til vektor repræsentationer. Dette gøres ved funktionen *RecursiveCharacterTextSplitter*, som splitter teksten i fragmenter på 1000 tegn, som laves med overlap på 200 til de øvrige fragmenter.

```
16  # Opdeling af chucks fra vektor bidder
17  text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
18  text_chunks = text_splitter.split_documents(documents)
```

Opsætning af fragmentstørrelse

Figur 65 - Splittelse af fragmenter

Herefter oprettes VDB'en ignnem *persist_directory*, som nævnes som Vector. Herefter sættes vektor dataene på fragmenterne. Dette gøres igennem OpenAIEmbeddings.

```
20  # Skab embeddings, og afsæt vektor koordinater
21  persist_directory = 'vector'
22  embedding = OpenAIEmbeddings()  # Using updated import
23
24  # Oprettelse af vektordb
25  vectordb = Chroma.from_documents(documents=text_chunks,
26                                   embedding=embedding,
27                                   persist_directory=persist_directory)
28
29  vectordb = None
30
31  # Load vektor chunks til database
32  vectordb = Chroma(persist_directory=persist_directory,
33                    embedding_function=embedding)
```

Udvælgelse af Embeddingsmodel

Figur 66 - Indlæsning af embeddings-model

For at sende forespørgsler til VDB'en konfigureres den som modtager. Dette sikrer ensartethed mellem scriptets prompt og de fragmenter eller dokumenter, der skal returneres som respons.

```
35  # Modtag og læs prompts.
36  retriever = vectordb.as_retriever()
37
38  # anvend prompt til søgning, samt dokumenter med relevant information.
39  prompt = "find doors"
40  docs = retriever.invoke(prompt)
41
42  #Indstil til anvendelse af to mest relevante resultater til resultat.
43  retriever = vectordb.as_retriever(search_kwargs={"k": 2})
44  retriever.search_type
45  print(docs)
```

Udvælgelse af antal fragmenter brugt til RAG

Figur 67 - Indstillingen af antal fragmenter til anvendelsen i prompten

Teknisk beskrivelse af Bilag 24

Bilag 24 er tilsvarende modstykke af *Bilag 23*, der anvendes Llama 3.1 og HF. Scriptet adskilles fra *Bilag 23*, ved at tillade indlæsningen af .ttl-filer, ud over det normale brug af .txt filer.

Importen af biblioteker varierer fra *Bilag 23*, da de embeddings der anvendes er HuggingFaceEmbeddings fremfor OpenAI.

```
1  import os
2  from langchain_chroma import Chroma
3  from langchain_community.embeddings import HuggingFaceEmbeddings
4  from langchain_community.document_loaders import DirectoryLoader, TextLoader
5  from langchain.text_splitter import RecursiveCharacterTextSplitter
```

Anvendelse af HuggingFace Embeddingsmodel

Figur 68 - Indlæsning af HF Embeddings-model

Der opsættes stier til anvendelsen af .txt og .ttl, som indlæses til opdeling af fragmenter.

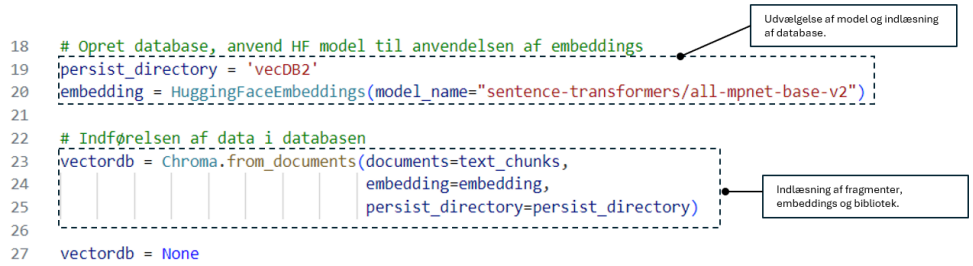
```
7  # Import af filer fra path, anvender både txt og ttl formater
8  loader_txt = DirectoryLoader("./files", glob="*.txt", loader_cls=TextLoader)
9  loader_ttl = DirectoryLoader("./files", glob="*.ttl", loader_cls=TextLoader)
10
11 # Indlæs både ttl og txt
12 documents = loader_txt.load() + loader_ttl.load()
13
14 # Split filerne til chunks i karakter størrelser
15 text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=200)
16 text_chunks = text_splitter.split_documents(documents)
```

Indlæsning af txt og ttl filer

Opsætning af fragmentstørrelse

Figur 69 - Opdeling af fragmenter

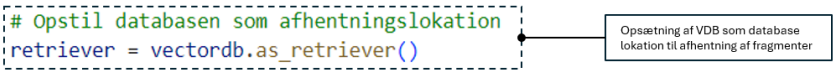
Herefter oprettes VDB'en, og HF-modellen indlæses for at bearbejde embeddings på fragmenter.




```
# Opret database, anvend HF model til anvendelsen af embeddings
persist_directory = 'vecDB2'
embedding = HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-v2")

# Indførelsen af data i databasen
vectordb = Chroma.from_documents(documents=text_chunks,
                                 embedding=embedding,
                                 persist_directory=persist_directory)

vectordb = None
```

Figur 70 - Indlæsning af embedding-model

Hertil indlæses fragmenterne i VDB'en. Den opsættes som afhentningspunkt for data igennem funktionen *vectordb.as_retriever*.




```
# Indlæs databasen med embeddings
vectordb = Chroma(persist_directory=persist_directory,
                  embedding_function=embedding)

# Opstil databasen som afhentningslokation
retriever = vectordb.as_retriever()

# Anvend afhentningen
prompt = "find doors"
docs = retriever.invoke(prompt)

# Opstilling af mængden af fragmenter der skal anvendes
retriever = vectordb.as_retriever(search_kwargs={"k": 10})

# Print resuktatet
print(docs)
```

Figur 71 - Opsætning af retriever og fragmenter splitter

Teknisk beskrivelse af Bilag 25

Ved tilføjelsen af dokumenter til eksisterende VDB, anvendes Python-scriptet fra *Bilag 25*, for OpenAI GPT 4o og *Bilag 26* for Ollama Llama 3.1


```
import os
from langchain_chroma import Chroma
from langchain_openai import OpenAIEmbeddings
from langchain_community.document_loaders import DirectoryLoader, TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.llms import OpenAI
from langchain.chains import RetrievalQA

# Set OpenAI API Key
os.environ["OPENAI_API_KEY"] = "xx"

# Initialize OpenAI Embeddings and Chroma for persistence
persist_directory = 'vector'
embedding = OpenAIEmbeddings()
```

Figur 72 - Indlæsning af Python-biblioteker og OpenAI Token

For at tilføje filer til den eksisterende VDB kræves import af Python-biblioteker for at bruge load-funktionerne. Ved opstart indlæses det eksisterende directory, hvor

fragmenter, der ikke allerede findes i VDB'en, identificeres. Disse behandles til nye fragmenter og tilføjes derefter til VDB'en.

```python
# Function to add documents from the 'files' directory to the existing Chroma database
def add_documents_to_db(vectordb, directory_path, glob_pattern="*.txt"):
    # Load documents from the specified directory
    loader = DirectoryLoader(directory_path, glob=glob_pattern, loader_cls=TextLoader)
    new_documents = loader.load()

    # Split the new documents into chunks
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
    new_text_chunks = text_splitter.split_documents(new_documents)

    # Add new chunks to the existing Chroma database
    vectordb.add_documents(new_text_chunks)
    print(f"Documents from '{directory_path}' have been added successfully.")

# Load additional documents from the 'files' directory
add_documents_to_db(vectordb, "files")
```


Figur 73 - Indlæsning af VDB

Teknisk beskrivelse af Bilag 26

Bilag 26 omhandler samme funktionalitet som *Bilag 25*. Tilføjelse af dokumenter og .ttl-filer til eksisterende VDB.

Import af Python-biblioteker kan ses i *Bilag 24*, hvor samme biblioteker anvendes. Embeddings anvendes som beskrevet i *Bilag 24*, hvor HuggingFaceEmbeddings bruges i stedet for OpenAIEmbeddings. Modellen fra *Bilag 24* benyttes og kan justeres til at behandle tilføjet data baseret på filtypen, der tilføjes. Derudover indlæses Python-biblioteket RDFlib for yderligere databehandling.

```python
import os
import subprocess
from langchain_chroma import Chroma
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_community.document_loaders import DirectoryLoader, TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.schema import Document
from rdflib import Graph, Namespace

# Definer anvendt database
persist_directory = 'dbttltxt'

# Indlæs embeddings i database
embedding = HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-v2")

# Indlæs database
print("Loading vector database...")
vectordb = Chroma(persist_directory=persist_directory, embedding_function=embedding)

# Verificer at database er læst
if vectordb is None:
    print("Failed to load the vector database.")
    exit()
```


Figur 74 - Indlæsning af Python-biblioteker og embeddings-model

For at kunne bearbejde .ttl-filerne, anvendes *graph* funktionen, til at indlæse relationerne der findes i RDF-dataene.

```python
# Indlæs RDF grafer til brug i database
class TurtleLoader:
    def __init__(self, file_path):
        self.file_path = file_path

    def load(self):
        graph = Graph()

        # Bind navne og inst sammen.
        graph.bind("RevitID", Namespace("https://help.autodesk.com/view/RVT/2024/ENU/?guid=GUID-DAF7AE25-23F3-4CAD-BAD8-6CD901306AEE#"))
        graph.bind("inst", Namespace("https://niras.dk/projects/1234#"))

        try:
            # Analyser RDF grafen
            graph.parse(self.file_path, format="ttl")

            # Omsæt graf til ttl format, til indlæsning i database
            content = graph.serialize(format="turtle")
            return [Document(page_content=content, metadata={"source": self.file_path})]
        except Exception as e:
            print(f"Error loading Turtle file {self.file_path}: {e}")
            return []
```


Figur 75 - Indlæsning af graf funktion og opstilling af format

De indledte informationer indlæses med *textloader*. Dette gøres med .ttl og .txt filerne, som efterfølgende kan bearbejdes gennem fragmentopdelingen.

```python
# Definer funktion til load af elementer
def load_documents_to_vectordb(vectordb, directory_path):
    try:
        all_documents = []

        # Indlæs tekstfiler
        txt_loader = DirectoryLoader(directory_path, glob="*.txt", loader_cls=TextLoader)
        txt_documents = txt_loader.load()
        print(f"Loaded {len(txt_documents)} .txt documents.")
        all_documents.extend(txt_documents)

        # Indlæs turtle filer
        ttl_files = [os.path.join(directory_path, f) for f in os.listdir(directory_path) if f.endswith(".ttl")]
        for ttl_file in ttl_files:
            ttl_loader = TurtleLoader(ttl_file)
            ttl_documents = ttl_loader.load()
            all_documents.extend(ttl_documents)
        print(f"Loaded {len(ttl_files)} .ttl documents.")

        # Split dokumenter til fragmenter
        text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
        document_chunks = text_splitter.split_documents(all_documents)
        print(f"Split documents into {len(document_chunks)} chunks.")

        # tilføj dokumenter til database
        vectordb.add_documents(document_chunks)
        print("Documents successfully added to the vector database.")

    except Exception as e:
        print(f"Error while loading documents: {e}")
```

Indlæsning af tekstfiler igennem TextLoader

Splittelse af dokumenter til fragmenter i valgt størrelse

Figur 76 - Opdeling af fragmenter

Teknisk beskrivelse af Bilag 27

Bilag 27 anvendes til indlæsning af VDB, søgning efter vektor fragmenter, og videre bearbejdning af denne gennem brugen af OpenAI GPT 4o.

Scriptet skal ved brugen af OpenAI, som *Bilag 23 og 25*, indlæse anvendte Python-biblioteker. Hertil indlæses VDB'en og de skabte embeddings / vektor koordinater.

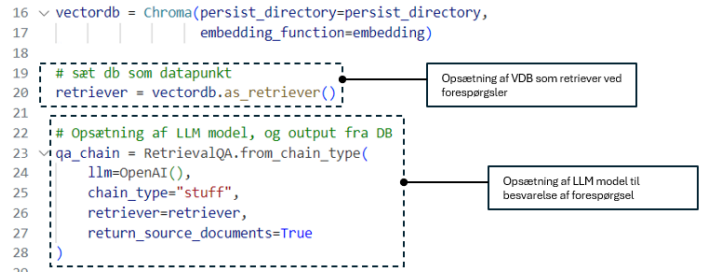
```python
import os
from langchain.vectorstores import Chroma
from langchain.embeddings import OpenAIEmbeddings
from langchain.document_loaders import DirectoryLoader, TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.llms import OpenAI
from langchain.chains import RetrievalQA

# Openai api kode
os.environ["OPENAI_API_KEY"] = "xx"

# Indlæs database
persist_directory = 'vector'
embedding = OpenAIEmbeddings()
```

Indlæsning af vektordatabase

Figur 77 - Indlæsning af Python-biblioteker og embedding-model

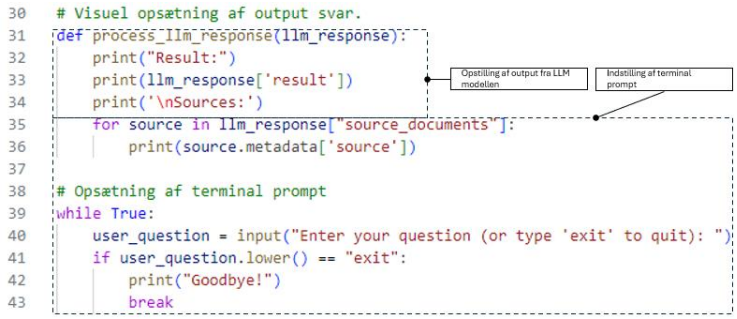
Den efterfølgende proces omhandler opsætningen af de NL-forespørgsler, der sendes, samt hvordan VDB'en og LLM'en skal respondere. Opsætningen indebærer at

definere VDB'en som retriever i kombination med de numeriske repræsentationer af tekstfragmenterne, der genereres gennem embeddings. Derudover opstilles en funktion til både at indhente og bearbejde informationerne vha. OpenAI.



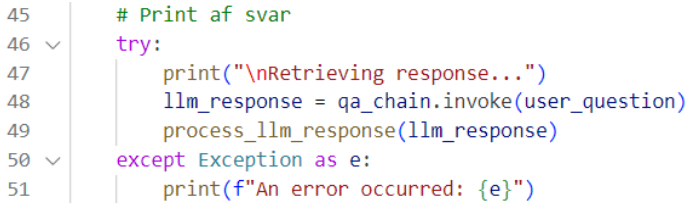
Figur 78 - Opsætning af retriever

Med opsætningen af *qa_chain* fremvises responsen fra LLM'en som modellens resultat, sammen med de anvendte dokumenter, der er fundet gennem *Retrieval*-funktionen. Den efterfølgende opsætning omhandler brugen af scriptet via cmd/terminal. Vha. en *if*-statement sikres det, at scriptet fortsætter med at køre, indtil en *exit*-prompt indtastes, hvor *break*-funktionen derefter anvendes til at afslutte processen.



Figur 79 - Opsætning af output

Ud fra brugerens input vil LLM herefter kunne levere en bearbejdet respons. Dette betyder, at *qa_chain* anvendes, når der stilles et spørgsmål, hvilket udføres gennem *invoke*-funktionen.



Figur 80 - Opsætning af output

Teknisk beskrivelse af Bilag 28

Bilag 28 er tilsvarende version af *Bilag 27*, med anvendelsen af Ollama, Llama 3.1 og HF. De gennemgående principper for scriptet er tilsvarende *Bilag 27*, dog tilpasset til anvendelsen af Ollama modeller.

Som grundlæggende del af scriptet, indlæses Python-biblioteker. VDB'en indlæses og opsætningen af embeddings vha. HF-modellen placeres.

```python
import subprocess
from langchain_chroma import Chroma
from langchain_huggingface import HuggingFaceEmbeddings

# Definer sti til database
persist_directory = 'dbttltxt'

# Indlæs embeddings til læsning i DB
embedding = HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-v2")

# Indlæs databasen
print("Loading vector database...")
vectordb = Chroma(persist_directory=persist_directory, embedding_function=embedding)

# Verficer at databasen er indlæst
if vectordb is None:
    print("Failed to load the vector database.")
    exit()
```

Figur 81 - Indlæsning af DB og model

Opsætningen til antallet af fragmenter der skal anvendes til LLM responsen, opsættes igennem *as_retriever*-funktionen og *kwargs*. Hertil kan der defineres funktionen *print*, som fremviser resultatet. Den efterfølgende proces, som beskrevet i *Bilag 27*, indebærer opsætning af *while*- og *if*-statements, der lader scriptet køre kontinuerligt, mens prompts sendes som input til LLM'en. De indhentede fragmenter kombineres med prompten og bruges som input til Llama 3.1.

```python
# Opsætning af mængden af anvendte fragmenter
retriever = vectordb.as_retriever(search_kwargs={"k": 5})

# Opstil funktion til output med respons og kilder
def process_llm_response(result_text, sources):
    print("Result:", result_text)
    print('\nSources:')
    for source in sources:
        print(source.metadata.get('source', 'Unknown'))

# Kør loop med prompt til LLM
while True:
    user_question = input("Enter your question (or type 'exit' to quit): ")
    if user_question.lower() == "exit":
        break

    # Giv dokumentation for kilder anvendt til respons fra LLM
    try:
        print(f"Retrieving documents for question: '{user_question}'")
        relevant_documents = retriever.invoke(user_question)

        if not relevant_documents:
            print("No relevant documents found in the vector database.")
            continue

        # Indsæt relevante fragmenter til prompt til udarbejdelsen af respons
        relevant_text = "\n\n".join([doc.page_content for doc in relevant_documents])
        prompt = f"Relevant Information:\n{relevant_text}\n\nQuestion: {user_question}\nAnswer:"
```

Udvælgelse af antal fragmenter anvendt i RAG

Opsætning af loop for prompt forespørgsler

Sammensætning af prompt og fragmenter til LLM bearbejdning

Figur 82 - Opsætning af indhentning af fragmenter og respons fra model

Ved anvendelsen af Ollama, kan Subprocess anvendes til at kalde en proces via cmd/terminalen. Her opsættes processen til at starte Ollama, og køre Llama 3.1 modellen.

```
49      # Anvend subprocess til kørsel af Llama3.1
50      result = subprocess.run(
51          ["ollama", "run", "llama3.1"],
52          input=prompt,
53          capture_output=True,
54          text=True,
55          check=True,
56      )
```

Opsætning af SubProcess kommando, og udvælgelse af ML model

Figur 83 - Opsætning af Ollama model

Scriptet sender outputtet fra Llama 3.1 og anvendes som responsen af tekstfragmenterne.

```
58          # Processer resultatet og output
59          llm_response_text = result.stdout.strip()
60          process_llm_response(llm_response_text, relevant_documents)
61
62      except subprocess.CalledProcessError as e:
63          print("Error while calling Ollama CLI:", e.stderr)
64      except Exception as e:
65          print(f"An error occurred during retrieval or processing: {e}")
```

Figur 84 - Indsættelse af fragmenter i prompten

Teknisk beskrivelse af Bilag 29

Bilag 29 anvender samme tilgang som *Bilag 18* til at kalde Ollama gennem terminalen vha. Subprocess-biblioteket. Scriptet foretager dog ikke kald til VDB'en, men udfører i stedet en søgning i lokale mapper for at finde data, der skal anvendes som input. Opsætningen af scriptet indebærer import af de nødvendige biblioteker, angivelse af stien til den relevante mappe og indlæsning af alle relevante filer, der findes i den angivne mappe.

```
1   import os
2   import subprocess
3
4   # Opstil sti til relevante dokumenter
5   directory_path = r"C:\Users\madss\Aalborg Universitet\Aarhus til AAU - General\4. Semester\RDF\ABox"
6
7   # find alle filer placeret i mappen.
8   file_paths = [
9       os.path.join(directory_path, file_name)
10      for file_name in os.listdir(directory_path)
11      if os.path.isfile(os.path.join(directory_path, file_name))
12  ]
13
```

Opsætning af sti til valgte filer

Sammensæt stinavn og filnavn

Figur 85 - Indlæsning af sti til directory

Filerne loades herefter med *r* funktionen. Dette efterfølges af *continue*-funktioner, som håndterer fejl i filerne eller situationer, hvor der ikke findes filer i den angivne mappe. Disse funktioner sikrer, at scriptet kan fortsætte sin eksekvering uden afbrydelser, selv hvis der opstår problemer med de indlæste filer.

```python
# load filerne til søgning
document_content = ""
for path in file_paths:
    try:
        with open(path, 'r', encoding='utf-8', errors='ignore') as file:
            document_content += file.read() + "\n"
    except FileNotFoundError:
        print(f"File not found: {path}")
        continue
    except OSError as e:
        print(f"Error reading file {path}: {e}")
        continue

if not document_content:
    print("No documents loaded. Please check the directory path and its contents.")
    exit()
```

Anvendelse af read kommando (r) til indlæsning af filer

Forsæt indlæsning selv med fejl i filer

Figur 86 - Indlæsning af filer

Den resterende del af scriptet anvender en tilsvarende fremgangsmåde til at kalde Ollama via terminalen, som beskrevet i *Bilag 28*.

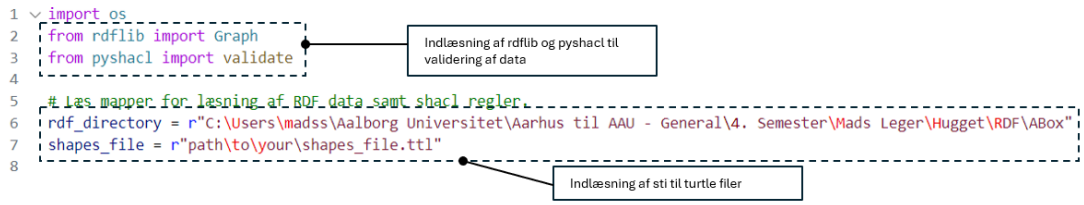
```python
# Input spørgsmål
while True:
    user_question = input("")
    if user_question.lower() == "exit":
        break

    # Indstil instruktion til ML modellen
    prompt = f"Answer in sql queries please.\n\n{document_content}\n\nQuestion: {user_question}\nAnswer:"

    # Anvend subprocess til kald til Ollama.
    try:
        result = subprocess.run(
            ["ollama", "run", "codellama"],  # Adjust the model name as needed
            input=prompt,                    # Pass the prompt directly via stdin
            capture_output=True,
            text=True,
            encoding='utf-8',                # Ensure UTF-8 encoding
            errors='ignore'                  # Ignore any characters that cannot be encoded
        )
        print(result.stdout)
    except subprocess.CalledProcessError as e:
        print("Error while calling Ollama CLI:", e.stderr)
```

Figur 87 - Opsætning af model og respons

Teknisk beskrivelse af Bilag 30

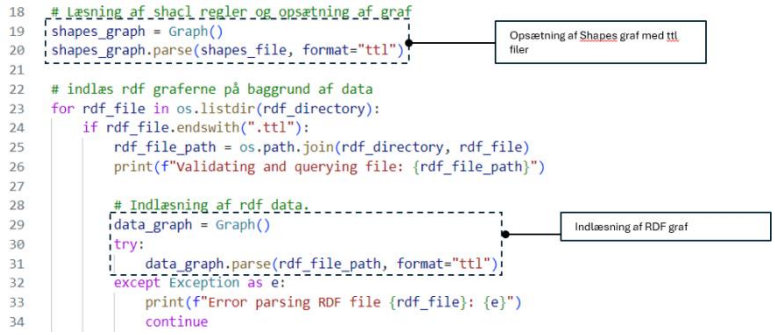
Bilag 30 bruges som grundlag for at sende SPARQL-forespørgsler til RDF-data uden brug af en TS. Dette demonstrerer, hvordan generering af forespørgsler kan kommunikere med DB'er og, i prototypens tilfælde, med lokale mapper, der indeholder RDF-data.

Scriptet anvender Python-bibliotekerne RDFlib og pySHACL. RDFlib bruges til at indlæse RDF-data i grafstrukturer, mens pySHACL muliggør validering af RDF-data vha. SHACL-regelsæt. Indlæsning af data i det lokale miljø sker ved at importere RDF-data som kildeinformation samt en supplerende mappe, der indeholder de nødvendige SHACL-regler.



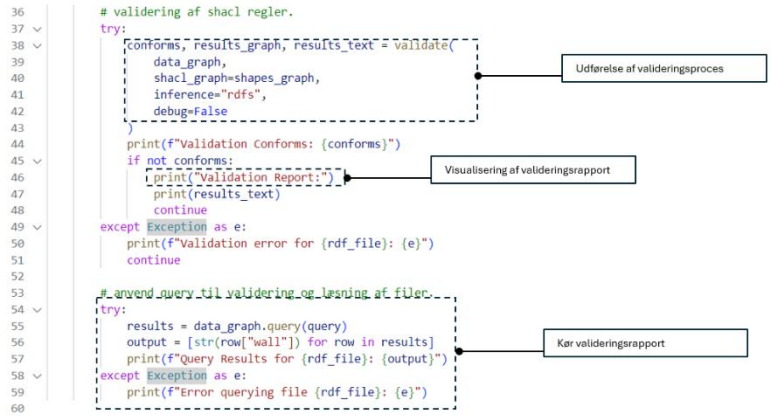
Figur 88 - Importer Python-biblioteker og sti

De importerede pakker anvendes til at opsætte RDF dataene i en grafstruktur. I forsøget anvendes der kun .ttl-filer som format. Dette vil kunne ændres til at læse andre RDF kompatible formater.



Figur 89 - Indlæsning af .ttl-filer til graf

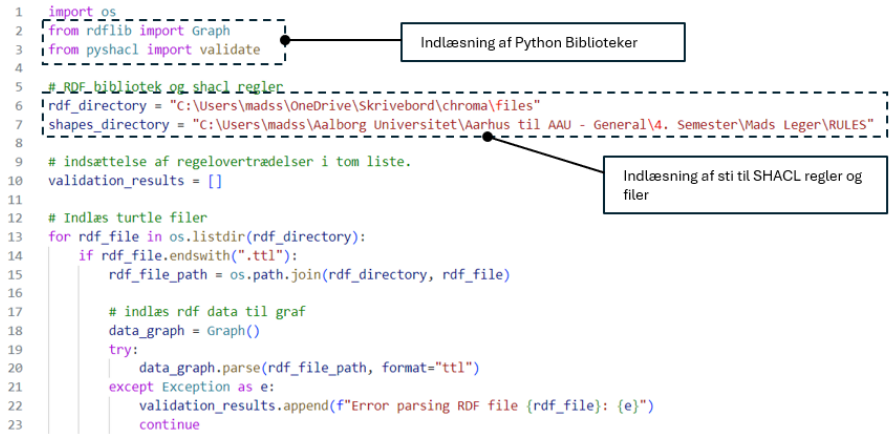
Muligheden for at kombinere SPARQL-forespørgsler med SHACL-regler implementeres via *Validation*-funktionen fra pySHACL. *Validation* bruges til at generere valideringsrapporter, der identificerer og beskriver eventuelle issues mellem RDF-dataene og SHACL-reglerne. Denne anvendes nærmere i *Bilag 31*.



Figur 90 - Opsættelse af valideringsrapport

Teknisk beskrivelse af Bilag 31

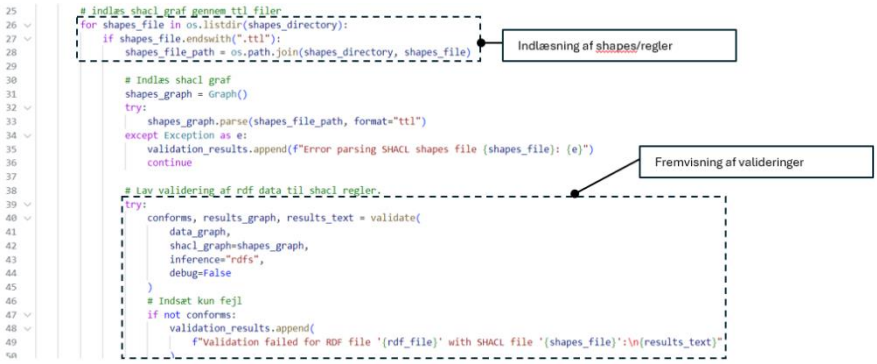
Fremgangsmåden for indlæsning af biblioteker, RDF-data og SHACL-regler, indlæses med samme fremgangsmetode som *Bilag 30*.




```python
import os
from rdflib import Graph
from pyshacl import validate

# RDF bibliotek og shacl regler
rdf_directory = "C:\Users\madss\OneDrive\Skrivebord\chroma\files"
shapes_directory = "C:\Users\madss\Aalborg Universitet\Aarhus til AAU - General\4. Semester\Mads Leger\RULES"

# indsættelse af regelovertrædelser i tom liste.
validation_results = []

# Indlæs turtle filer
for rdf_file in os.listdir(rdf_directory):
    if rdf_file.endswith(".ttl"):
        rdf_file_path = os.path.join(rdf_directory, rdf_file)

        # indlæs rdf data til graf
        data_graph = Graph()
        try:
            data_graph.parse(rdf_file_path, format="ttl")
        except Exception as e:
            validation_results.append(f"Error parsing RDF file {rdf_file}: {e}")
            continue
```

Figur 91 - Import af Python-biblioteker og indlæsning af SHACL-regler

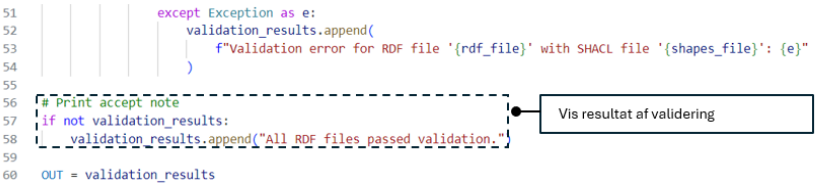
Indlæsningen af RDF-dataene og SHACL-reglerne opsættes som grafer. *Validation*-funktionen bruger disse to grafer til at kontrollere informationerne. Valideringsfunktionen identificerer derefter de data, der ikke overholder SHACL-reglerne, og opstiller dem som issues.




```python
# indlæs shacl graf gennem ttl filer
for shapes_file in os.listdir(shapes_directory):
    if shapes_file.endswith(".ttl"):
        shapes_file_path = os.path.join(shapes_directory, shapes_file)

        # Indlæs shacl graf
        shapes_graph = Graph()
        try:
            shapes_graph.parse(shapes_file_path, format="ttl")
        except Exception as e:
            validation_results.append(f"Error parsing SHACL shapes file {shapes_file}: {e}")
            continue

        # Lav validering af rdf data til shacl regler.
        try:
            conforms, results_graph, results_text = validate(
                data_graph,
                shacl_graph=shapes_graph,
                inference="rdfs",
                debug=False
            )
            # Indsæt kun fejl
            if not conforms:
                validation_results.append(
                    f"Validation failed for RDF file '{rdf_file}' with SHACL file '{shapes_file}':\n{results_text}"
```

Figur 92 - Indlæsning af SHACL-graf

Hvis der ikke resulteres i nogle issues fra SHACL-reglerne, viser *print* at RDF-dataene overholder kravene.

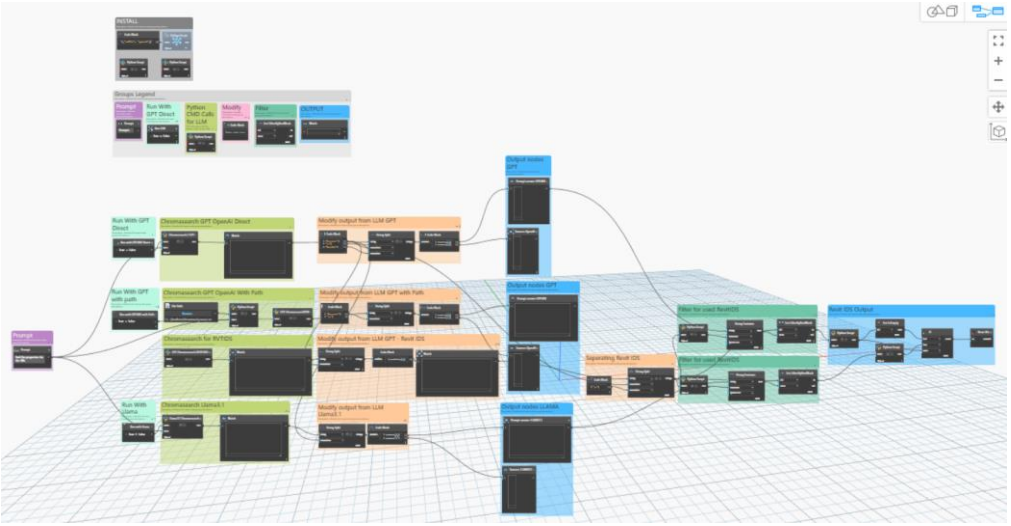



```python
            except Exception as e:
                validation_results.append(
                    f"Validation error for RDF file '{rdf_file}' with SHACL file '{shapes_file}': {e}"
                )

# Print accept note
if not validation_results:
    validation_results.append("All RDF files passed validation.")

OUT = validation_results
```

Figur 93 - Fejlsøgning i valideringsrapport

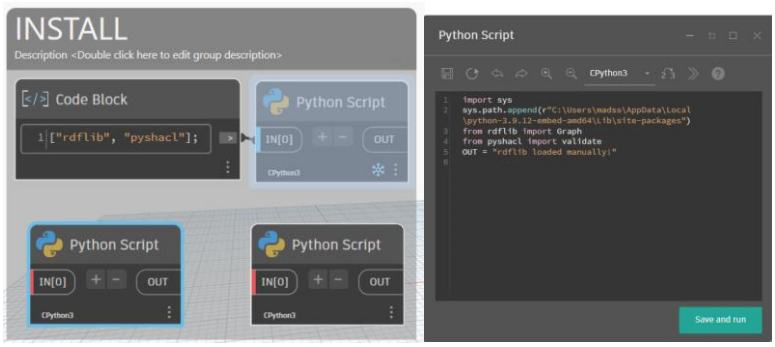
Teknisk beskrivelse af Bilag 32

Bilag 22 er en samlet Dynamo graf, der anvendes til at sammensætte funktionerne fra de øvrige bilag.



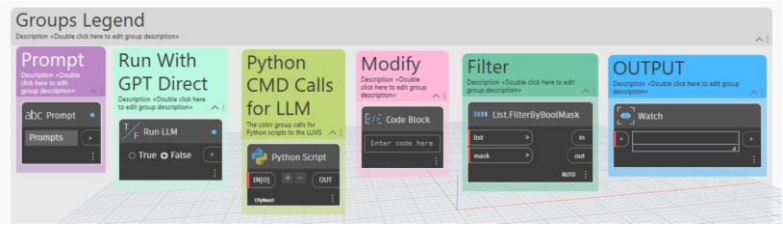
Figur 94 - Samlet Dynamo Graf

Ved brug af Dynamo kan udfordringer med ekstra Python-biblioteker føre til fejl i grafens funktionalitet. For at undgå dette anvendes Python-scripts, der manuelt installerer de nødvendige biblioteker. Denne løsning sikrer, at alle krævede biblioteker er tilgængelige, selv hvis der opstår kompatibilitetsproblemer i Dynamo.



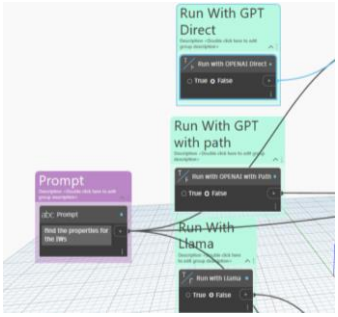
Figur 95 - Installation af Python-biblioteker

Grafen består af forskellige typer af nodes, hvor opdelingen af disse er fremvist i grupper ud fra funktionaliteten.



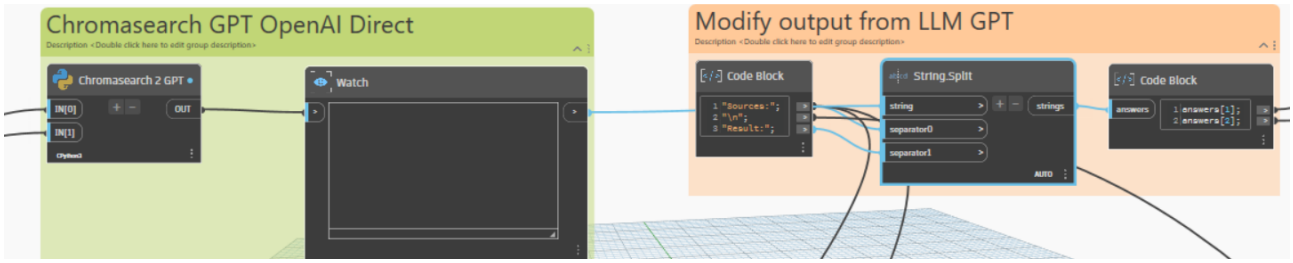
Figur 96 - Beskrivelse af gruppetyper

Grafens anvendelse baseres på valget af LLM, der skal bearbejde VDB'ens resultater. Opsætningen inkluderer en *boolean*-variabel, der muliggør valg af LLM gennem *true/false*.



Figur 97 - Boolean til udvælgelse af anvendte modeller

Hver er modellerne er sat op gennem et Python-script, der anvender VDB'erne og deres LLM-bearbejdning. Resultatet fra LLM'en splittes på Sources og Results, som efterfølgende vises i separate lister.



Figur 98 - Søgning gennem LLM og manipulation af respons

Resultaterne kommer fra anvendelsen af *Bilag 27*. Dette gøres igennem anvendelsen af Subprocess biblioteket, som kan køre Python-scriptet igennem terminalen og efterfølgende returnere outputtet.

Inputtet indhentes igennem IN[0], hvor booleanen fortæller om scriptet skal aktiveres. Hvis værdien læses som true aktiveres scriptet. IN[1] anvendes til NL-forespørgsel som skal bearbejdes af LLM/VDB'en.

```python
import os
import subprocess

# Boolean condition input to determine if the script should run
run_condition = IN[0]  # Boolean input (True = run, False = stop)

# Check if the condition is False
if not run_condition:
    # Stop execution if condition is False
    OUT = "Execution skipped: run_condition is False."
else:
    # Proceed with the script if condition is True
    # Define paths
    directory_path = r"C:\Users\madss\OneDrive\Skrivebord\chroma"
    python_file_path = r"C:\Users\madss\OneDrive\Skrivebord\chroma\chromasearch2.py"
    python_executable = r"C:\Users\madss\AppData\Local\python-3.9.12-embed-amd64\python.exe"

    # Prepare input
    try:
        input_text = IN[1]  # Input from Dynamo
    except Exception:
        input_text = "default question"  # Fallback input

    # Set PYTHONPATH
    env = os.environ.copy()
    env["PYTHONPATH"] = r"C:\Users\madss\AppData\Local\python-3.9.12-embed-amd64\Lib\site-packages"

    # Run the subprocess
    command = [python_executable, python_file_path]
    process = subprocess.Popen(
        command,
        cwd=directory_path,
        env=env,
        stdin=subprocess.PIPE,
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
        text=True
    )
    # Pass input_text to subprocess and get output
    output, error = process.communicate(input=input_text)
    result = output if output else error

    # Output the result
    OUT = result
```

Kører kun hvis input er True

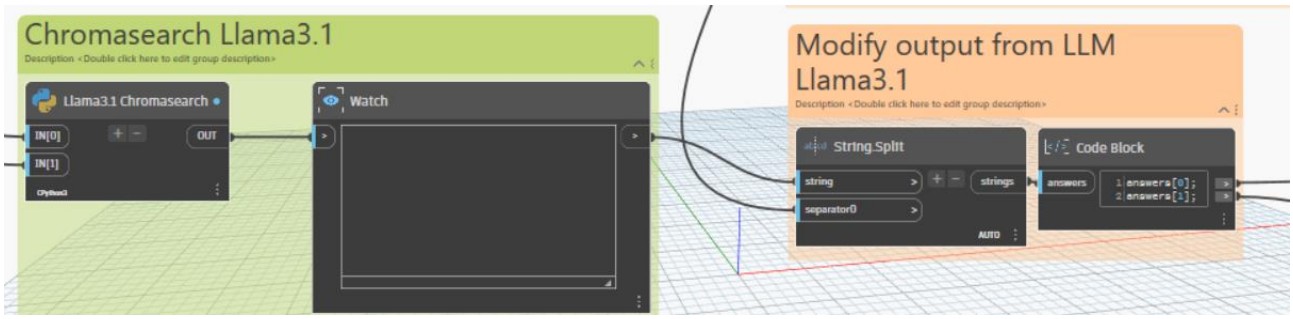
Indlæsning af stier til Python og script

Indlæser prompt fra bruger

Indlæsning af CMD, kørsel af SubProcess

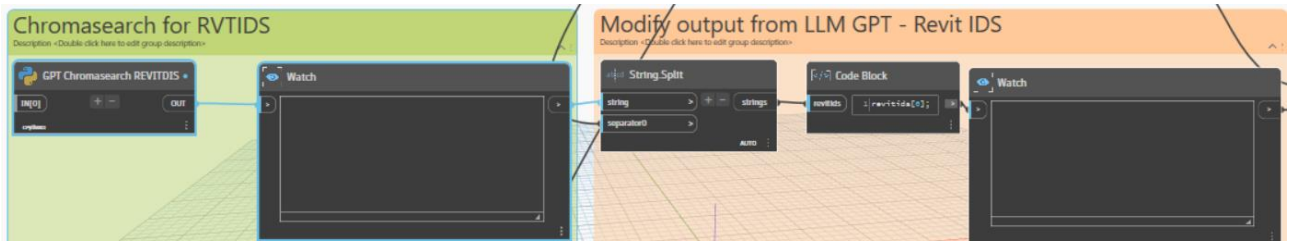
Figur 99 - Anvendelsen af Subprocess til indlæsning af Python-script

Denne metode bruges også til at kalde Llama 3.1-modellen.



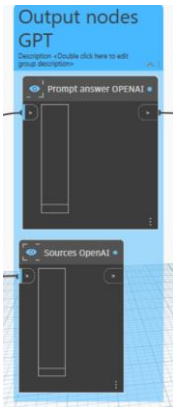
Figur 100 - Respons fra Llama 3.1

Det grundlæggende princip anvendes også til søgning efter RVT-IDs. Her adskiller det sig ved at inkludere en fast defineret prompt, der efterspørger alle tilgængelige elementers RVT-IDs, som er defineret i de anvendte data.



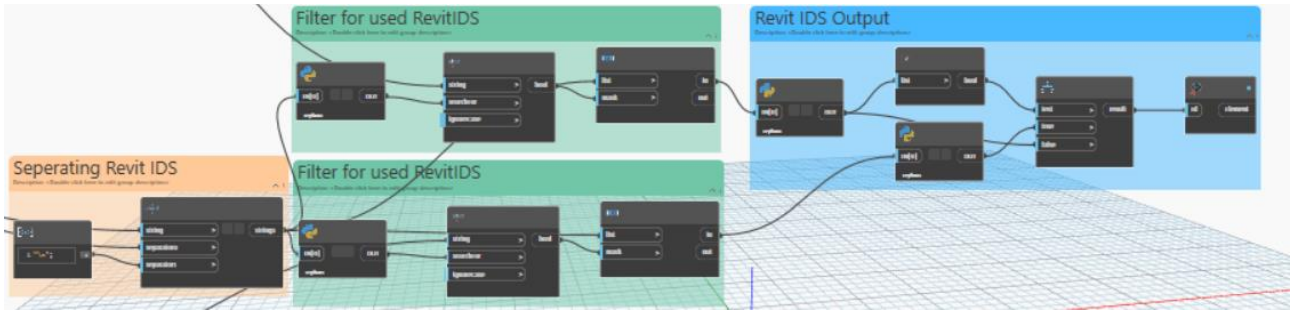
Figur 101 - Indhentning af Revit IDs

De genererede resultater vises i Watch-noder og defineres som respons. Disse præsenteres vha. Dynamo Player.



Figur 102 - Output fra Revit IDs

Den afsluttende del af Dynamo Grafen omhandler fremvisning af BIM elementernes ID, som vil kunne resultere i interaktion med elementer i deres Native software.

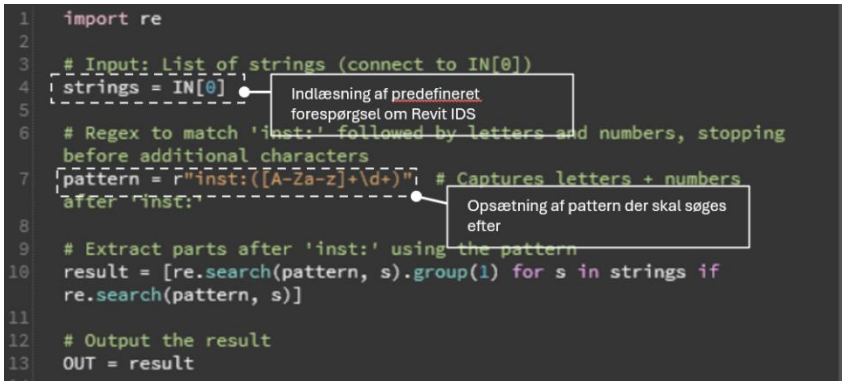


Figur 103 - Søgning efter anvendte BIM-elementer

Separationen af Revit IDs sker som respons fra LLM'en, der søger efter og identificerer Revit IDs. Disse IDs returneres typisk som en lang streng, der ikke er indekseret. Derfor anvendes en *split*-funktion til at opdele strengen baseret på linjeskift *\n*.

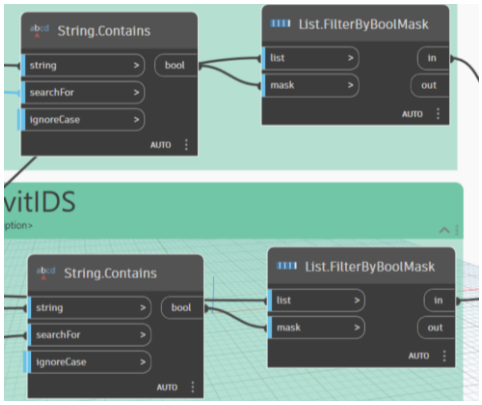
I prototypen anvendes RDF-data, hvor BIM-elementer benytter præfikset *inst:*. Dette gør det muligt at fremsøge elementer som f.eks. *inst:iw1*, der definerer en indervæg. De opsplittede *strings* bruges derefter som input i et Python-script, hvor præfikset *inst:* bevares, men *suffixet* ignoreres. Dette muliggør søgning og sammenligning af elementer med resultaterne fra OpenAI/Llama 3.1s respons.

.



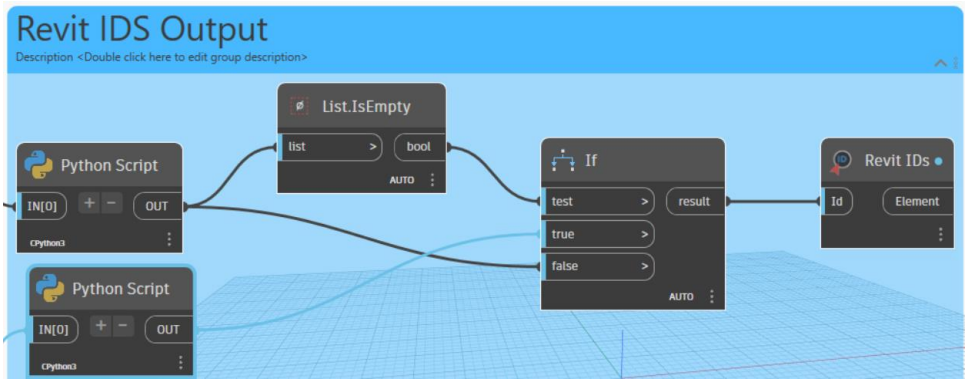
Figur 104 - Sortering efter anvendte BIM-elementer

Søgningen efter disse anvendes til at søge i den respons de to LLM'er giver. Dette gøres igennem *String.Contains* funktionen som giver en *boolean*, om de fundne *inst:* værdier kan findes i responsen. Denne tages videre i *FilterByBoolMask*, som sortere efter *boolean* værdierne.



Figur 105 - Sortering af Revit IDs

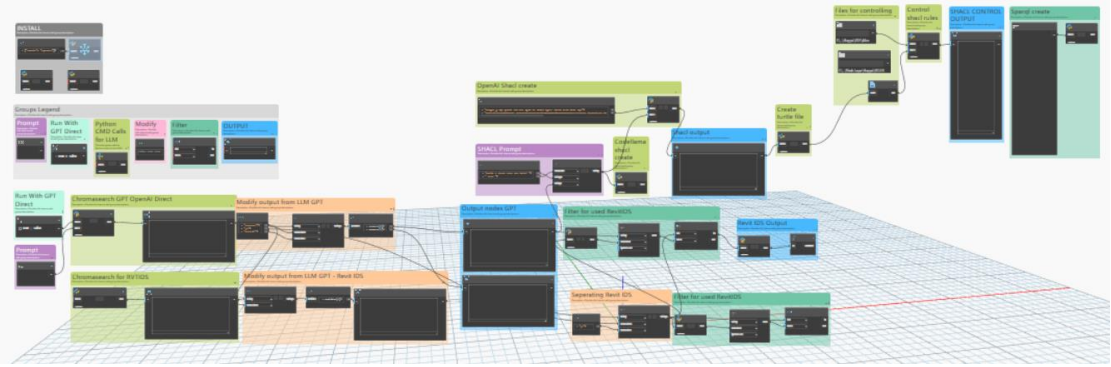
Outputtet anvendes til søgning efter de relevante Revit IDs. Dette gøres efter et tilsvarende script, som søgte på *Inst:* værdien, udskiftet med RevitID, der anvendes som Prefix i RDF-dataene, som giver elementernes ID property. For ikke at give dobbelt output, anvendes en kontrol af om listens output er tom. Dette anvendes efterfølgende til input i et *if statement*, som resulterer i en søgning af elementer.



Figur 106 - Opstilling af IF-statement til sortering

Teknisk beskrivelse af Bilag 33

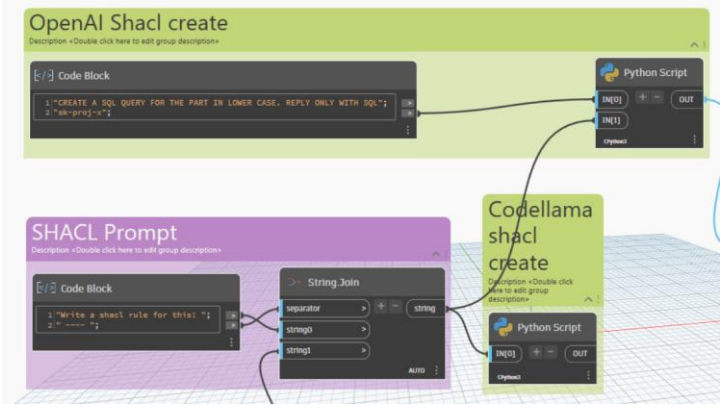
Bilag 33 er en modificering af *Bilag 32*, hvor der ikke anvendes alle LLM'er. I forsøget er der til søgning i VDB'en kun anvendt OpenAI 4o, som processerer informationen hurtigst af de anvendte modeller.



Figur 107 - Samlet Dynamo Graf

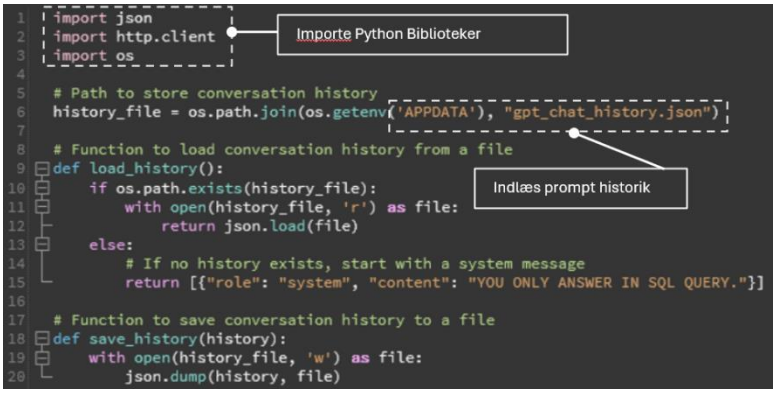
Responsen fra LLM'en kan bruges til at generere SHACL-regler og SPARQL-forespørgsler. De anvendte modeller er dog ikke trænet med de korrekte ontologier, hvilket medfører, at syntaksen ofte ikke stemmer overens med de forespørgsler og regler, der skal anvendes i grafen.

I forsøget med at omsætte responsen til kode er OpenAI 4o og Ollamas kodebaserede model Codellama blevet brugt. Codellama-modellen kaldes ved den fremgangsmåde, der er beskrevet i *Bilag 32*, mens det udførte script er det samme som præsenteret i *Bilag 29*.



Figur 108 - Omdannelse af respons til SHACL-regel

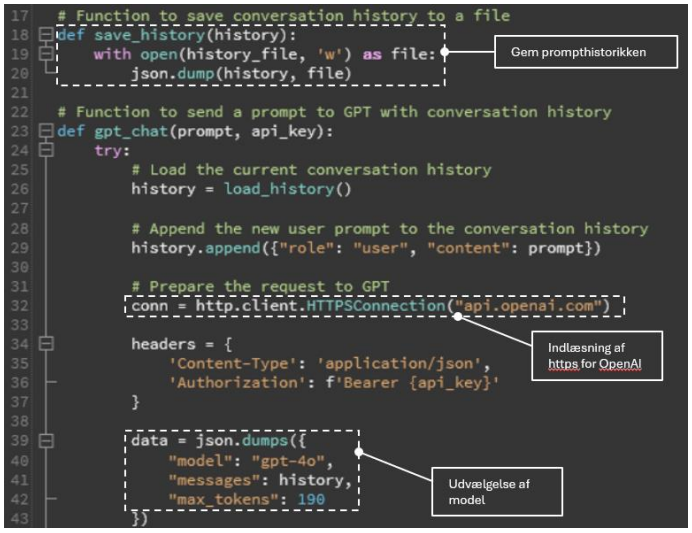
Ved anvendelsen af OpenAI 4o er der benyttet en anden tilgang. Her er der forsøgt at fine-tune modellen i et mindre omfang, hvor modellen anvender en fil i JSON-format som en slags hukommelse. Scriptet bruger HTTPS til at lave direkte kald til openai.com.



Figur 109 - Indlæsning af GPT-historik

Opsætningen af JSON-filen med chat-historikken udvides med nye spørgsmål hver gang modellen anvendes. JSON-filen giver mulighed for at specificere felterne *role*, *system* og *content*, hvilket gør det muligt at skabe en manuel fine-tuning proces. Denne opsætning kan justere modellen til at følge en specifik instruktion for responsens type og struktur samt definere, hvordan responserne skal udformes.

Problemet med denne tilgang opstår ved brugen af *max_tokens*-begrænsningen, som definerer det maksimale antal tokens, der kan anvendes ved hver prompt. Ved anvendelsen, skal hukommelsen også tokeniseres, og dermed bliver antallet af tokens hurtigt brugt, når der anvendes fine-tuning. Derfor er denne løsning ikke praktisk anvendelig i en reel prototype, medmindre begrænsningen på det maksimale antal tokens fjernes.



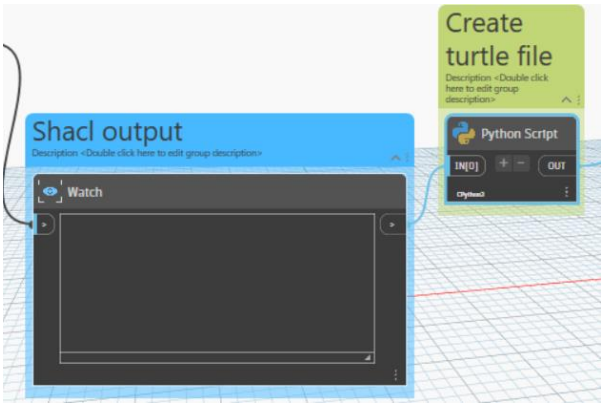
Figur 110 - Indlæsning af GPT 4o model

Den resterende del af scriptet, anvenders til at returnere NL og promptresponsen til json filen.

```
        print("Sending request to OpenAI...")
        conn.request("POST", "/v1/chat/completions", body=data, headers=headers)

        # Get the response
        response = conn.getresponse()
        response_data = response.read()

        print(f"Response Status: {response.status}")
        print(f"Response Data: {response_data}")

        if response.status == 200:
            response_json = json.loads(response_data)
            gpt_response = response_json["choices"][0]["message"]["content"]

            # Append GPT's response to the history
            history.append({"role": "assistant", "content": gpt_response})

            # Save the updated history back to the file
            save_history(history)

            return gpt_response
        else:
            return f"Error: {response.status}, {response_data.decode()}"

    except Exception as e:
        # If any error occurs, return the error message
        return f"Error occurred: {str(e)}"

# Inputs from Dynamo
api_key = IN[0]  # OpenAI API key as input
user_prompt = IN[1]  # User's question or request as input

# Get response from ChatGPT
chatgpt_response = gpt_chat(user_prompt, api_key)

# Output
OUT = chatgpt_response
```

Figur 111 - Opsætning af respons størrelse

Outputtet i form af SHACL-reglerne kommer ud som *string*, og er nødsaget til at blive omdannet til .ttl format, for at det kan læses og anvendes i grafforbindelsen.



Figur 112 - Output fremviser SHACL-regel og gemmer den

Python-scriptet bruges derfor til at omsætte *stringen* fra NL til .ttl-formatet. Dette gøres ved IN[0] som definere *stringen* fra NL fra modellen. Dette skrives til en ny fil med *w* funktionen.

```python
import os

# Input Turtle string from IN[0]
turtle_string = IN[0]  # Input Turtle content from Dynamo

# Define the directory where the Turtle file will be saved
output_directory = r"C:\Users\madss\OneDrive\Skrivebord\chroma\RULES"  # Ensure this directory exists

# Define the output Turtle file name with proper extension
output_file = os.path.join(output_directory, "shaclrules.ttl")  # Combine directory and file name

# Write the Turtle string to the output file
try:
    with open(output_file, "w", encoding="utf-8") as f:
        f.write(turtle_string)
    # Output success message with the full file path
    OUT = f" {output_file}"
except Exception as e:
    # Handle any errors and output the error message
    OUT = f"Error creating Turtle file: {e}"
```

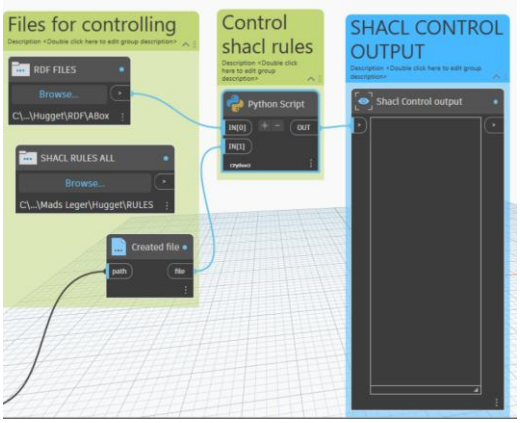
Indlæs Turtle fil

Indlæsning af regelsæt

Læs Turtle filer

Figur 113 - Indlæsning af SHACL-regel og valideringsproces

Den afsluttende del af SHACL-kontrollen bruges til at validere RDF-dataene mod SHACL-reglerne. Dette udføres med en modificeret udgave af *Bilag 31*, som understøtter brugen i Dynamo.



Figur 114 - Output af valideringsrapport

Teknisk beskrivelse af Bilag 34 og 35

Bilag 34 og 35 er bestående af to zip filer indeholdende med .txt- og .ttl-filer. Materialet der er taget udgangspunkt i ConTech Labs Pionerprojekt 5 (Rasmussen, 2021). Projektet indeholder en række RDF-data i .ttl-format, som danner grundlag for en samlet grafenhed. Datafilerne er dog udvidet med egenskaber på de eksisterende elementer i filerne samt tilføjelser af filer med ny data. Der er også indsat regler og tekst i NL, som bruges som kontrol i VDB'en. *Bilag 35* indeholder SHACL-regelsæt fra projektet, hvor der er lavet yderligere tilføjelser i disse filer.

Files mappen indeholder som følgende:

Brandsektioner.ttl – Indeholder RDF data beskrivende af brandsektioner og dertilhørende rum, samt navne.

Bygningstopologi.ttl – Indeholder RDF data beskrivende af elementers relationerne forhold til øvrige elementer, samt beskrivelse igennem BOT.

Elementegenskaber.ttl – Indeholder RDF data vedrørende egenskaber for elementer i bygningen.

```
75  # Stairs
76  inst:STR1
77      rdfs:label  "Trappe 1"@da ,
78                  "Stair 1"@en ;
79      ex:type "Straight" ;
80      ex:freeHeight 2.0 ;
81      ex:stepRiserHeight 0.20 ;
82      ex:treadDepth 0.30 ;
83      ex:freeWidth 1.0 .
84
```

Figur 115 - Udklip af Elementegenskaber.ttl

Precheck1-Consistency.txt – Indeholder SHACL regel for konsistens kontrol, generelle regler for f.eks. etager indeholder rum – Storey include minimum one space.

Precheck2-Geometry.txt – Indeholder SHACL regler vedrørende geometriske forhold i bygningen.

Precheck3-Noise.txt – Indeholder SHACL regler vedrørende krav om støj til egenskaber indlagt i Elementegenskaber.ttl.

Precheck4-Fire.txt – Indeholder SHACL regler vedrørende krav om brand til egenskaber indlagt i Elementegenskaber.ttl.

Precheck5-Moisture.txt – Indeholder SHACL regler vedrørende krav om fugt til egenskaber indlagt i Elementegenskaber.ttl.

Produkter.ttl – Beskriver relationer fra elementegenskaber og Bygningstopologi til IFC-elementer.

Railing.ttl – ChatGPT genereret SHACL regel taget fra Bygningsreglementet, omhandlende krav til gelænder. (Social- og Boligstyrelsen, 2018)

Regler.txt – Beskrivelse af projektkrav i NL samt udklip fra BR18 (Social- og Boligstyrelsen, 2018).

Revitids.ttl – RDF Data indhentet Revit IDs fundet i Revit fil, til sammenkobling mellem LD og BIM-software.

rumprogram.ttl – RDF-data beskrivelse af spaces/rum der findes i projektet, samt egenskaber forbundet til rum.

SBIVådrum.txt – Kopieret beskrivelse i NL fra SBI (Anvisninger, 2021)

Stairs.ttl - ChatGPT genereret SHACL regel taget fra Bygningsreglementet, omhandlende krav til trapper. (Social- og Boligstyrelsen, 2018)