Norges miljø- og biovitenskapelige universitet

Masteroppgave 2020 30 stp
REALTEK – Fakultet for realfag og teknologi

Kartlegging av beste strategi for 3D modellering og oppbygging av digitale tvillinger gjennom testing av etablert teori

Exploration of Strategies for 3D CAD Modelling and Construction of Digital Twins through Testing Established Theory

Maja Gunvor Erlandsen
Morten Sæther
Maskin, prosess- og produktutvikling

Maja Gunvor Erlandsen
Morten Sæther

Forord

Prosjektet er gjennomført som en masteroppgave ved Norges miljø- og biovitenskapelige universitet, NMBU, våren 2020 og gjøres i samarbeid med Multiconsult sin enhet for fornybar energi. Målet har vært å finne den beste strategien for å sette opp parametriserte modeller og skape digitale tvillinger med 3D DAK.

I dette prosjektet har vi fått en mulighet til å utforske en svært spennende og framtidsrettet måte å jobbe med dataverktøy for ingeniører. Vi tror at ferdighetene og kunnskapen vi har opparbeidet oss vil kunne komme til nytte svært mange ganger gjennom arbeidslivet og derfor har det vært motiverende å jobbe med dette gjennom hele prosjektets varighet. Det har kanskje vært spesielt motiverende å oppdage at dette er et felt som er relativt lite utforsket i norsk sammenheng og at vi kanskje er blant de først i Norge til å gjøre et slik studie.

Vi har hele tiden samarbeidet tett gjennom hele prosjektets varighet, men allikevel er arbeidsoppgavene fordelt etter våre styrker. Siden Morten var ungdom har han lekt seg med programmering og er godt kjent med programmer og dataverktøy. Dermed var det svært naturlig at han ledet den praktiske delen av prosjektet. Maja er svært strukturert av natur og har dermed tatt mer styring i planlegging og utforming av rapporten.

Rapporten er skrevet slik at Multiconsult skal kunne benytte den som en standard for intern opplæring. Leseren er antatt å være en ingeniør med grunnleggende kjennskap til 3D-modellering og analyseverktøy uten at det er et betydelig behov for forkunnskaper.

Videre ønsker vi å rette en stor takk til alle de som har hjulpet oss med å lage denne rapporten som vi nå er ganske så stolte av. Vi har fått svært god oppfølging av våre to veildere Bjarne Børresen fra Multiconsult og Tor Anders Nygaard fra NMBU. Takk for at dere har holdt oss på den smale stien og stadig presset oss framover. Videre har Kristine Bøe, Amalie Risholm og Henrik Aamodt lest gjennom rapporten for å bidra med sine svært ærlige tilbakemeldinger på presentasjonen av det faglige innholdet. Så har Inger Johanne Helth Sveen og Erna Erlandsen hjulpet oss med språk og struktur. Tiden på Hemsen ble alt for kort dette semesteret da korona begrenset oss alle til hjemmekontor. Likevel retter vi en takk til alle på Hemsen for at dere gjør hver arbeidsdag litt bedre.

Til sist er det både med stolhet og vemod at vi avslutter vår utdanning ved NMBU. Dette har vært både svært lærrerikt og det vi vil se tilbake på som noen av de flotteste årene i våre liv. Dette er utvilsomt det beste stedet å være student i Norge.

Maja Gunvor Erlandsen
Morten Sæther

Sammendrag

De aller fleste som har drevet noe med modellering er kjent med problemstillingen der de kun skal endre en liten detalj og hvor det ender opp med at hele modellen kræsjer. Da en skal lære seg DAK-programmer lærer en om hvordan bruke de ulike verktøyene i programmet, men ingenting om strategier for å bygge robuste modeller.

For å bygge en 3D-modell trenger en i utgangspunktet ikke tenke så mye over strategier. Det finnes svært mange ulike framgangsmåter som ville gitt akkurat den samme 3D-modellen. Det er i det øyeblikket en skal begynne å gjøre endringer at det vil bli stor forskjell på de ulike framgangsmåtene. Er modellen robust vil den tåle disse endringene godt. For de mindre gode modellene vil det i beste fall være tungvindt å endre parametre, men i svært mange tilfeller vil modellen kræsje.

I denne rapporten er det gjort et forsøk for å finne beste mulige strategi for å bygge parametriserte modeller. Modelleringsarbeid er ofte tidkrevende og en kan spare inn ressurser dersom man kan gjenbruke en modell fra et tidligere prosjekt. For at dette skal være mulig kreves det en robust modell som tåler relativt store endringer av parametre. Det er også ønskelig at denne modellen skal være intuitiv å overta for en annen ingeniør, slik at det kun er behov for å bygges én slik modell i den gjeldende bedriften. I tillegg vil det i de aller fleste tilfeller oppstå et behov for å endre på 3D-modellen underveis i et prosjekt. Er modellen bygget på en strategisk måte vil denne endringen ta kortere tid.

Digitale tvillinger er et spennende konsept som brukes i stadig flere sammenhenger og som også er aktuelt for 3D-modellering. Ved å koble sammen regneark og analyseprogrammer med en 3D-modeller vil en ha mulighet til å automatiser manuelle og tidkrevende prosesser for å ytterligere spare inn ressurser. På grunn av teknologiske framskritt blir bedrifter stadig mer effektive og det er viktig å bruke slike verktøy for å ikke tape i sitt marked. Digitale tvillinger er også et godt verktøy for å optimalisere konstruksjoner.

For å kunne etablere en best mulig strategi ble det først sett på hvilke strategier som allerede er utarbeidet og hvilken forskning som er gjort på dette. Videre ble det gjennomført tester av disse for å se hvilke strategier som skilte seg ut.

Det som raskt blir tydelig i arbeidet er at det viktigste for å spare ressurser internt i en bedrift er ikke hvilken strategi en velger, men at alle i bedriften modellerer etter samme strategi. På denne måten vil modellene være lett gjenkjennelige og det er lettere å overta en modell etter en annen ingeniør. Likvel er *Resilient Modeling strategy* det som anbefales å implementere i bedriften. Dette er en helhetlig metode som i større grad endrer DAK-ingeniørenes tankesett i stede for å styre med rigide regler.

Videre anbefales det å etablere *Design Intent* som et naturlig steg i alle modelleringsprosesser. Dette innebærer at det må legges en plan i forkant og ingeniøren må tenke gjennom hvilke endringsbehov som mest sannsynlig vil inntreffe. For å kunne forutse slike ting kreves det et visst erfaringsbehov og derfor anbefales det at det er de erfarne DAK-ingeniørene som bygger de parametriserte modellene for bedriften.

Maja Gunvor Erlandsen
Morten Sæther

Abstract

Most people who have done CAD modelling are familiar with the problem of a model breaking after changing one minor detail. One of the reasons why this problem arises is because, up until now, the focus in school has been to learn the functions of the CAD program rather than focus on strategies and routines for planning the model.

The choice of strategy concerned when only constructing a part does not matter that much. This is because either strategy will result in the same model. In addition, there is no particular difference in difficulty using them. On the other hand, when you try to alter the model, minor flaws in your modelling technique will shine through and in the worst case cause the model to break on update. Proper planning and strategy use will result in a robust model and handle changes in an easy and predictable way.

In this thesis there is conducted tests to find the best parametric model building strategy. Constructing CAD models is time-consuming work and therefore a company could save resources on reusing existing models. This means a company would have to invest some extra resources upfront, but would save for each time the engineer uses the old model for new projects. This is where a robust model that is as flexible as possible saves the most. However, it is easy for one engineer to come up with a proper robust model, but the work may be for nothing if it is impossible for a second engineer to use and make changes to it. The changes should be possible to make fast and easy. This is where the need for a companywide strategy is important.

Digital twins is an exciting consept and it could be used in many different areas. One of these areas is 3D modelling, where the concept is based around connecting early calculations with FEA and CAD. The successful making of a Digital Twin will result in the automation of trivial, repetitive and time-consuming tasks. Optimization is an example of time-consuming work that is possible to automate. Ultimately, all this automation helps the company to stay ahead in its industry.

To be able to establish a best practise strategy, we searched for existing strategies and if there were any studies done around the effectiveness of these strategies. Further, we conducted tests of our own to explore these researched techniques.

It quickly became apparent that the most important thing for efficient CAD-modelling is not the choice of strategy, but rather to settle on one strategy and stick with it throughout the organisation. This ensures that the companies' engineers understands each other's work even if the original creator of the model is not there to explain it. The paper did end up with a recommendation for a strategy for components: "Resilient Modelling Strategy". This is because its focus is towards guiding the engineers' actions instead of controlling the actions with firm rules.

The concept of Design Intent has shown its value and it is recommended that this be implemented in all CAD processes. In short, this entails that there is to be a plan prepared ahead of all CAD work. It should at a minimum cover some thoughts about how the models intended use is and basic information about parameters and structure. Therefore, it forces the designing engineer to think about choices that otherwise would be swept under the rug. This demands a high level of knowledge by the engineer and should be done by an experienced engineer to secure the models usefulness.

Innhold

Forord i

Sammendrag ii

Abstract iii

Innhold v

Figurer viii

Tabeller ix

Begrepsliste x

1 Innledning 1
- 1.1 Oppdragsgiver . . . 1
- 1.2 Bakgrunn . . . 1
- 1.3 Oppgave . . . 2
  - 1.3.1 Problemformulering . . . 2
  - 1.3.2 Valg av oppgave . . . 2
  - 1.3.3 Resultatmål . . . 2
- 1.4 Veiledning . . . 3
- 1.5 Informasjonskilder . . . 3
- 1.6 Avgrensning . . . 3
- 1.7 Oppbygging av rapporten . . . 4
- 1.8 Oppbygging av tillegg . . . 4

2 Teori 5
- 2.1 Om modellering . . . 5
  - 2.1.1 Hva er en parametrisert modell? . . . 5
  - 2.1.2 Avhengigheter . . . 5
  - 2.1.3 Kompleksitet . . . 5
  - 2.1.4 Design intent . . . 6
- 2.2 Strategi for oppbygging av komponenter . . . 6
  - 2.2.1 Generelt . . . 6
  - 2.2.2 Delphi´s Horisontal Modeling . . . 7
  - 2.2.3 Explicit Reference Modeling . . . 8
  - 2.2.4 Resilient Modeling . . . 8
  - 2.2.5 Sammenligning av strategier . . . 10
- 2.3 Organisk formede komponenter . . . 11
  - 2.3.1 Loft . . . 11
  - 2.3.2 Sweep . . . 11
- 2.4 Strategi for oppbygging av sammenstillinger . . . 12
  - 2.4.1 Top-Down Design . . . 12
  - 2.4.2 Bottom-Up Design . . . 17
  - 2.4.3 Middle-Out Design . . . 18
  - 2.4.4 Sammenligning av strategier . . . 19
- 2.5 Digitale tvillinger . . . 19
  - 2.5.1 Optimalisering av modell . . . 20
  - 2.5.2 Datadeling mellom programvarer . . . 21

2.6 Inventor . . . 21
2.6.1 Modelleringsfunskjoner . . . 22
2.6.2 Optimaliseringsfunksjoner . . . 23
2.6.3 Datadelingsfunksjoner . . . 23
2.7 Beregningsprogrammer . . . 23
2.7.1 Mathcad . . . 24
2.7.2 Jupyter notebook . . . 25
2.7.3 Excel . . . 25
2.7.4 Sammenligning av beregningsprogrammer . . . 27
2.8 Analyseprogrammer . . . 27
2.8.1 Inventor . . . 27
2.8.2 SolidWorks Simulation . . . 28
2.8.3 ANSYS . . . 28
2.8.4 Sammenligning av analyseprogrammer . . . 29

3 Ståstedsanalyse **31**
3.1 Tidligere masteroppgaver . . . 31
3.2 Kurs om strategi 3D-modellering . . . 31
3.3 Patentregisteret . . . 31
3.4 Oppsummert . . . 31

4 Metode **33**
4.1 Prosesstrinn . . . 33
4.2 Kvalitetsikring . . . 34
4.3 Innsamling av data . . . 35
4.4 Databehandling . . . 35

5 Resultater **37**
5.1 Oppbygging av komponent . . . 37
5.2 Organisk utformet geometri . . . 38
5.2.1 Loft . . . 39
5.2.2 Sweep . . . 39
5.3 Oppbygging av sammenstilling . . . 39
5.3.1 Top-Down Design . . . 40
5.3.2 Bottom-Up Design . . . 41
5.3.3 Middle-Out Design . . . 41
5.3.4 Oppsummert . . . 41
5.4 Inventor . . . 41
5.5 Beregningsprogrammer . . . 43
5.6 Analyseprogrammer . . . 44

6 Drøfting **45**
6.1 Utredningsfasen . . . 45
6.1.1 Plan . . . 45
6.1.2 Teori . . . 46
6.2 Utviklingsarbeid . . . 46
6.2.1 Bearbeiding av teorien . . . 46
6.2.2 Testing . . . 47
6.3 Endelig produkt . . . 47
6.3.1 Resultater . . . 47
6.3.2 Design Intent . . . 47

7 Konklusjon **49**
7.1 Resultatmål . . . 49
7.2 Måloppnåelse . . . 50
7.2.1 Hovedmål 1 . . . 50

7.2.2 Hovedmål 2 . . . 50
7.3 Videre arbeid . . . 51
7.3.1 Design Intent . . . 51
7.3.2 Testing . . . 51
7.3.3 Programmering . . . 51
7.3.4 Optimalisering . . . 52

Bibliografi **55**

Tillegg **59**

Tillegg A Design Intent **60**
A.1 . . . 60

Tillegg B Ståstedsanalyse **61**
B.1 Erfaringsskriv Multiconsult . . . 61
B.2 Modelleringsprogram for organisk utformet geometri . . . 63

Tillegg C Forsøksrapporter **64**
C.1 Strategi for komponent: Horizontal . . . 64
C.2 Strategi for komponent: Explicit . . . 65
C.3 Strategi for komponent: Resilient . . . 66
C.4 Organisk utformet geometri: Loft . . . 67
C.5 Organisk utformet geometri: Sweep . . . 68
C.6 Organisk utformet geometri: CAESES og metasurface . . . 69
C.7 Strategi for sammenstilling: Top Down: Single Skeleton . . . 70
C.8 Strategi for sammenstilling: Top Down; Multi Skeleton . . . 71
C.9 Strategi for sammenstilling: Top Down: Multibody . . . 72
C.10 Strategi for sammenstilling: Bottom-Up . . . 73
C.11 Strategi for sammenstilling: Middle-Out . . . 74
C.12 Inventor: iPart/iAssembly . . . 75
C.13 Inventor: API . . . 76
C.14 Inventor: Program for test av parametre . . . 77
C.15 Beregningsprogram: Mathcad . . . 78
C.16 Beregningsprogram: Mathcad kommunikasjon . . . 79
C.17 Beregningsprogram: Jupyter Notebook . . . 80
C.18 Beregningsprogram: Excel . . . 81
C.19 Analyseprogram: Inventor . . . 82
C.20 Analyseprogram: Solidworks . . . 83
C.21 Datadeling: Mathcad og Inventor . . . 84
C.22 Datadeling: Jupyter Notebook og Inventor . . . 85
C.23 Datadeling: Excel og Inventor . . . 86
C.24 Datadeling: Overføring til ANSYS . . . 87

Figurer

2.1 Sitat om *Design intent* . . . 6
2.2 Sitat om planlegging . . . 6
2.3 Horizontal modeling . . . 7
2.4 Forskjellen på klasse 1 og klasse 2 referering . . . 8
2.5 Sammenligning av *Design tree* til komponent . . . 10
2.6 Organisk utformet komponent med *Loft* . . . 11
2.7 Organisk utformet komponent med *Sweep* . . . 12
2.8 Prinsippet i et *Top-Down* design . . . 12
2.9 Skeleton modeling . . . 13
2.10 Sammenheng mellom PS og DS i *multi skeleton modelling* . . . 15
2.11 Eksempel på en sammenstilling hvor du typisk nytter *Multi-body part* . . . 17
2.12 Prinsippet i et *Bottom-Up* design . . . 17
2.13 Prinsippet i et *Middle-Out* design . . . 18
2.14 Alternativer til hvordan bygge opp en optimaliseringsprosess . . . 20
2.15 Prinsipper for deling av data mellom ulike programvarer . . . 21
2.16 Parametervindu i Inventor . . . 22
2.17 Eksempel på et regneark i Mathcad . . . 24
2.18 Eksempel på regneark laget med Jupyter Widgets . . . 25
2.19 Eksempel på oppsett av formel i Excel . . . 26
2.20 Eksempel på bruk av *Excel name range* . . . 26

4.1 Prosessdiagram for prosjektarbeidet . . . 33
4.2 Kvalitetsikring av rapporten ved bruk av HOQ . . . 34

5.1 Parametrisert komponent . . . 38
5.2 Parametrisert glideluke . . . 39
5.3 Vinkel avstivet av ribber . . . 42
5.4 Applikasjon for å kommunisere mellom Mathcad og Inventor . . . 42
5.5 Egen fane i Excel som styre kommunikasjon med SolidWorks . . . 44

Tabeller

2.1 Inndeling av fastholdninger i *Explicit reference modeling* 8
2.2 *Features*-grupper i *resilient modeling* strategien [1] 9
2.3 Sammenlikning av strategier for oppbygging av komponenter 10
2.4 Skjelettinndeling i *Multi-Skeleton* 14
2.5 Parameterinndeling i *multi-skeleton* 16
2.6 Sammenlikning av strategier for oppbygging av sammenstillinger 19
2.7 Sammenligning av beregningsprogrammer 27
2.8 Sammenligning av analyseprogrammer 29

4.1 Søkeord benyttet i litteratursøk 35

5.1 Resultater oppbygging av komponent 37
5.2 Resultater test av ulike strategier for TTD 40
5.3 Resultater beregningsprogrammer 43
5.4 Resultater analyseprogrammer 44

Begrepsliste

I denne listen er det samlet alle begreper hvor det er et behov for å tydeliggjøre dens betydning. Noen fordi begrepene kan ha en annen betydning i en annen sammenheng, andre fordi de kan oppleves som fremmedord.

Det er også en del engelske ord i teksten. I dataverden regjerer dette språket og der hvor de engelske faguttrykkene er blitt en integrert del av det norske arbeidsspråket, er det valgt å ikke oversette disse.

Generelt

Adaptiv	Kan tilpasse seg.
Designautomatisering	En parametrisert modell koblet opp mot en matematisk modell som muliggjør optimalisering av designet.
Designfleksibilitet	Muligheten til å nytte eksisterende geometri i andre sammenhenger og applikasjoner.
Digital tvilling	Beskrevet i avsnitt 2.5
Gjenbrukbarhet	Se designfleksibilitet.
Komponent	En enkelt del. Kan være en del av en sammenstillingen eller en enkeltstående komponent.
Modell	Samlebegrep for komponent og sammenstillinger.
Modelleringsstrategi	Strategi eller en plan for hvordan en modell skal bygges.
Parameter	Kan deles inn i to: Konstant størrelse og variabel størrelse
Parametrisering	Beskrevet i avsnitt 2.1.1
Parametrisert modell	En konstruksjon som styres av et begrenset antall parametre.
Sammenstilling	Flere komponenter satt sammen til en modell.
Skjelett	Overbygning eller struktur av en modell.
Topologi	Gren av moderne geometri. Hvordan ulike geometrier samhandler og er arrangert i forhold til hverandre.
3D-modell	Se modell.

Modellering

Chamfer	Overgangen mellom to overflater på et objekt.
Design Tree	Arrangementet av *features* i en komponent eller en sammenstilling.
Extrude	Skape solid av en skisse (i sammenhenge med modellering).
Fastholdinger	Definere hvordan ulike komponenter samhandler med hverandre.
Feature	Samlebetegnelse for alle handlinger en kan gjøre i prosessen da man skaper en 3D-modell.
Node	Et knutepunkt/forgreningspunkt
Null tykkelse	Et element som ikke har noe tykkelse. Eksempler er plan og overflater laget av Surface.
Path	En sti. Ofte definert med en linje i rommet.
Pattern	Gjentagende mønster av en *feature*.
Solid	En geometri med masse.
Surface	En overflate med null tykkelse.

Forkortelser

2D	Todimensjonal
3D	Tredimensjonal
API	Application Programming Interface (NO: Programmerings Grensesnitt)
DAK	Dataassistert konstruksjon
DS	Design skeleton model
FEA	Finite element analysis
FEM	Finite element method
GP	General product design parameter
KP	Key product design parameter
LP	Interface and location design parameter
LS	Location skeleton model
NIPO	Norwegian Industrial Property Office (NO: Patentregisteret)
NMBU	Norges miljø- og bioviteskapelige universitet
PS	Published Skeleton Model
RMS	Resilient modeling strategy
STEP	Standard for Exchange of Product Model Data
TDD	Top-down design

1. Innledning

I dette kapittelet presenteres oppgaven og oppdragsgiver. Det er beskrevet valg av oppgave, hva prosjektet omhandler, avgrensinger og lignende, slik at leseren kan forberede seg på rapportens videre innhold.

1.1 Oppdragsgiver

Multiconsult er et av Norges ledende konsulentselskaper og har røtter helt tilbake til 1908. Selskapet tilbyr multifaglig rådgiving, design, prosjektering, arkitektur, prosjektoppfølging, ledelse, verifikasjon og kontroll - både nasjonalt og internasjonalt. Multiconsult har kontorer svært mange steder rundt om i Norge og har spesialisert seg innenfor syv ulike forretningsområder. Disse er bygg & eiendom, industri, olje & gass, samferdsel, fornybar energi, vann & miljø og by & samfunn [2].

Denne oppgaven gjennomføres i samarbeid med maskinsektoren i enheten for fornybar energi. Sektoren har 17 ansatte som i all hovedsak sitter på kontoret enten i Oslo eller i Drammen. De jobber med prosjekter innen vannkraft både nasjonalt og internasjonalt. Typiske prosjekter kan være enten nybygg av vannkraft eller vedlikehold og inspeksjon av eksisterende kraftverk.

1.2 Bakgrunn

Bruk av 3D DAK og moderne beregningsverktøy betraktes som *state of the art* for mekaniske konstruksjoner. Samtidig er det fortsatt ønskelig å forbedre arbeidsprosessen ved å knytte konstruksjon, beregninger og analyse tettere sammen i integrerte modeller, også kalt «digitale tvillinger». I en parametrisert modell er det viktig at alle elementer styres av et konsistent sett av nøkkelparametre. Ved å koble disse opp som digitale tvillinger vil en ha mulighet til å optimalisere de videre parametrene. For å oppnå dette er det viktig at det er en klar sammenheng mellom styrende dimensjoner og beregninger.

Multiconsult har benyttet seg av denne metoden i ulike sammenhenger, men framgangsmåten er ikke satt i system. Derfor er det ønske om å utvikle en strategi slik at de effektivt kan sette sammen parametriserte modeller og på sikt vil kunne effektivisere ressursbruken i prosjekter.

Ved å utforske teknikker innen parametrisering av modeller er det ønskelig å finne fram til en effektiv metode. Ideelt sett skal modellen kunne gjenbrukes av andre ingeniører med kjennskap til programvaren. Dette stiller altså krav til brukervennligheten og designfleksibilitet.

Inventor benyttes som modelleringsverktøy, men vurderingene av modelleringsstrategi skal gjøres mest mulig generell og om mulig også vurderes mot andre modelleringsverktøy.

Nøkkelord

Parametrisk modellering
Digitale tvillinger
Modelleringsstrategi
Designfleksibilitet
Brukervennlighet

1.3 Oppgave

I dette delkapittelet konkretiseres det hvilke oppgaver som skal utføres i dette prosjektet. Dette gjøres gjennom en konsis problemformulering og gjennom flere resultatmål. I tillegg er det beskrevet hvorfor denne oppgaven ble valgt.

1.3.1 Problemformulering

Prosjektet skal dokumentere og teste ut ulike strategier for å bygge opp en parametrisert 3D-modell integrert med digitale beregnings- og analyseverktøy. Disse vurderes opp mot hverandre og om mulig skal en beste strategi etableres.

1.3.2 Valg av oppgave

Ved å velge denne oppgaven har det vært en mulighet til å tilegne seg ferdigheter det er behov for i de fleste bedrifter som driver med produktutvikling eller konstruksjon. Denne måten å kombinere maskinteknikk, modellering og koding er framtidsrettet og en smart måte å jobbe på.

Selv om de fleste bedrifter ville oppnå fordeler ved å bruke parametriserte modeller, er det heller få ingeniører som jobber med dette. Ved å tilegne seg denne kunnskapen håper prosjektdeltakerene å oppnå en konkurransefordel i jobbsøkerfasen og senere i arbeidslivet.

1.3.3 Resultatmål

I dette avsnittet konkretiseres det hvilke mål en ønsker å nå i løpet av prosjektperioden.

Hovedmål

Hovedmålene beskriver hovedhensikten med hva som skal oppnås i prosjektet.

1	Anbefale en god strategi på hvordan bygge opp en parametrisert modell i et DAK program.
2	Kartlegge hvordan en mest hensiktsmessig bygger opp digitale tvillinger med 3D DAK modeller.

Delmål

Delmålene er arbeidsoppgaver som skal beskrive veien fram til hovedmålene.

1	A	Beskrive parametrisert modell og digitale tvillinger. Beskrive hva som tidligere er dokumentert rundt dette og se om det er noen strategier som utpeker seg.
	B	Gjennomføre tester av det som beskrives i teorien og beskrive hvilke resultater en da finner.
	C	Utarbeide en hensiktsmessig strategi for hvordan bygge en parametrisert modell.
2	A	Se på hvilke beregningsprogrammer som kan kobles opp mot modelleringsprogrammer og gi en anbefaling om hvilket som bør nyttes.
	B	Se på hvilke analyseprogrammer som kan kobles opp mot modelleringsprogrammer og gi en anbefaling om hvilket som bør nyttes.
	C	Definere i hvilket program en bør legge inn de drivende parametrene for å styre modellen.
	D	Identifisere eventuelle krav til 3D-modellen for at en slik metode kan benyttes.

1.4 Veiledning

Prosjektet har i all hovedsak blitt fulgt opp av to veiledere, en fra universitetet og en fra Multiconsult. Vider følger det en kort beskrivelse av disse to.

Bjarne Børresen	Multiconsult
Bjarne har 25 års erfaring fra design, analyse og prosjektering av mekanisk utstyr til vannkraftanlegg. Fra 1995 til 2010 arbeidet han i Kværner Energy, GE Hydro og Rainpower med ulike oppgaver og i ulike roller knyttet til turbindesign. GE Hydro og Rainpower med ulike oppgaver og i ulike roller knyttet til turbindesign. Som rådgiver i Energi Norge (2010-2014) var han ansvarlig for kurs, kompetanseutvikling og forskningsprosjekt for norsk kraftproduksjon. Redaktør for ulike publikasjoner bla. Veileder for Maskinforskriften for vannkraftutstyr, KOLEMO 3.0 og revisjon av Malingsspesifikasjonen. Siden 2014 har han arbeidet i Multiconsult med prosjektering, oppfølging og problemløsing for vannkraftprosjekt i Norge og internasjonalt. Bjarne Børresen er utdannet siv.ing maskin fra NTNU og Docteur Ès Sciences Techniques (dr.ing) fra EPFL, Sveits.	

Tor Anders Nygaard	Norges miljø- og biovitenskapelige unviversitet (NMBU) og Institute for Energy Technology (IFE)
Master- og doktorgrad, maskin NTH/NTNU. Har arbeidet med vindkraft siden 1986 ved IFE, bortsett fra 6 år ved NASA Ames Reserach Center (helikopter aerodynamikk). IFE, bortsett fra 6 år ved NASA Ames Reserach Center (helikopter aerodynamikk). Jobber for tiden på IFE med design og analyse av flytende vindturbiner. Har 20% prof stilling ved NMBU.	

1.5 Informasjonskilder

I dette delkapittelet er det en kort beskrivelse av hvilke typer informasjonskilder som det ble planlegges med å benytte i dette prosjektet. Teorien baseres seg på bøker, tidsskrifter og websider. I tillegg er ulike diskusjonsforum og Youtube brukt for å tilegne seg ferdigheter nødvendig for å løse oppgaven .

Det er også gjennomført intervjuer med ansatte i Multiconsult som allerede har laget parametriserte modeller. Dette for å tilegne seg kunnskap fra deres erfaringer og bygge videre på disse. Det er også hentet erfaringer fra en bacheloroppgave som Multiconsult tidligere har hatt på dette temaet.

1.6 Avgrensning

For å sette rammer rundt prosjektet og dens problemstilling er det nødvendig med avgrensinger. De som er satt for denne oppgaven beskrives videre i dette delkapittelet.

Det er kun Inventor som er vurdert i denne oppgaven som modelleringsprogram. Det er dette programmet Multiconsult bruker i dag og for å begrense ulike kombinasjonmuligheter er det kun denne programvaren som det er brukt tid på. Videre for beregningsprogrammer er det de som Multiconsult har tilgjengelig i dag eller som er tilgjengelig gratis som ble tatt med i oppgaven og det samme gjelder for analyseprogrammer med unntak av SolidWorks. Dette fordi SolidWorks kanskje er den mest naturlige konkurrenten til Multiconsult sine analyseprogrammer.

Videre er det ikke brukt ressurser på å løse beregningene. Dette fordi det ikke er relevant for målene for prosjektet. Det er kun brukt tid på å vurdere hvordan 3D-modellen og beregningene påvirker hverandre. De ulike modellen er kun satt opp godt nok til å kunne vurdere de ulike strategiene.

Det er ikke tatt hensyn til hvilken lagringsplass eller datakapasitet løsningen vil kreve.

1.7 Oppbygging av rapporten

Videre følger en beskrivelse av innholdet i hvert kapittel. Ved å ta med en slik beskrivelse vil det være lettere for leseren å finne nettopp den informasjonen den søker.

Kapittel 1	**Innledning** I dette kapittelet presenteres oppgaven og oppdragsgiver. Det er beskrevet valg av oppgave, hva prosjektet omhandler, avgrensinger og lignende, slik at leseren kan forberede seg på rapportens videre innhold.
Kapittel 2	**Teori** I dette kapittelet dekkes noe av den generelle teorien som omhandler parametrisering av modeller og oppbygging av digitale tvillinger. De ulike begrepene beskrives, ulike strategier diskuteres og de aktuelle programvarene er omtalt. Dette kapittelet er i all hovedsak basert på internasjonal forsking og produsentenes egen omtale av programvarene.
Kapittel 3	**Ståstedsanalyse** I dette kapittelet beskrives det hva som er vanlig praksis i Norge med hensyn på å nytte ulike strategier i 3D-modellering. Det er sett på hvordan dette implementeres i bedrifter, hvilken masteroppgaver som er gjort og hvilke kurs som tilbys innenfor dette. Internasjonal forsking på dette området er redegjort for i kapittel 2.
Kapittel 4	**Metode** I dette kapittelet er det beskrevet framgangsmåten benyttet i forbindelse med dette prosjektet. Hensikten er å bygge opp troverdigheten rundt rapporten ved å forklare hvordan konklusjonen er nådd, samt gjøre det lettere å gjenskape de forsøkene som er gjort.
Kapittel 5	**Resultater** I dette kapittelet presenteres de observasjoner som er blitt gjort under testingen av de ulike strategiene og programvarene.
Kapittel 6	**Drøfting** I drøftingen er begrunnelsen for enkelte valg presentert i tillegg til at styrker og svakheter ved rapporten er beskrevet. Dette kapittelet ansees som nødvendig for at leseren skal kunne danne seg en mening om rapportens validitet.
Kapittel 7	**Konklusjon** I dette kapittelet beskrives det hvilke konklusjoner som er utarbeidet av denne rapporten. Dette gjøres ved å svare på hvert enkelt resultatmål satt for dette arbeidet.

1.8 Oppbygging av tillegg

Videre følger en beskrivelse av oppbyggingen til tileggene. Hensikten her er den samme som for delkapittelet over.

Tillegg A	**Design Intent** Resultatet av arbeidet som er gjort i dette prosjektet. *Design Intent* er ment som et verktøy i planleggingsprosessen til å lage en parametrisert modell.
Tillegg B	**Ståstedsanalyse** I denne delen av vedleggene ligger alt det som er med på å beskrive situasjonen slik den er i dag.
Tillegg C	**Forsøksrapporter** Hvert enkelt forsøk som er gjort i løpet av prosjektet er beskrevet i minirapporter og ligger vedlagt i denne delen.

2. Teori

I dette kapittelet dekkes noe av den generelle teorien som omhandler parametrisering av modeller og oppbygging av digitale tvillinger. De ulike begrepene beskrives, ulike strategier diskuteres og de aktuelle programvarene er omtalt. Dette kapittelet er i all hovedsak basert på internasjonal forsking og produsentenes egen omtale av programvarene.

2.1 Om modellering

Før en går videre i rapporten er det viktig å etablere hva de ulike begrepene innebærer. I dette delkapittelet beskrives det hva en parametrisert modell er, hvilke aspekter av dette det skal sees på videre i prosjektet og andre relevante uttrykk som nyttes i denne sammenhengen.

2.1.1 Hva er en parametrisert modell?

Parameter defineres som en størrelse som kan ha ulike verdier, men som i hvert tilfelle utdeles en bestemt verdi, som igjen vil påvirke resultatet av det en studerer [3].

At en 3D-modell er parametrisert betyr at det er en geometrisk figur hvor endring i utformingen kan styres ved å kun endre på noen få parametre. På komponentnivå forsøker en å relatere dimensjonene til hverandre slik at ved å oppdatere en av parametrene i modellen så vil de andre følge etter. For en parametrisert sammenstilling må denne settes opp slik at også antall komponenter, og plassering av disse, oppdateres riktig ved endring av parametre.

Det er gjerne flere konstruksjoner som gjenbrukes fra prosjekt til prosjekt, men hvor størrelsen vil variere. Et eksempel kan være en ventil på et vannrør. En bestemt type ventil vil ha samme komponenter og være satt likt sammen, men størrelsen vil avhenge av dimensjonene på røret og vanntrykket. Hensikten med å bygge parametriserte 3D-modeller er at en ved å endre på færrest mulig parametre skal kunne tilpasse ventilen til det prosjektet en jobber med for øyeblikket. I dette tilfellet vil eksempler på relevante parametre være nettopp rørdimensjon og vanntrykk.

2.1.2 Avhengigheter

Ingeniører kurses gjerne i hvordan bruke ulike DAK programmer, men ikke i hvordan de kan bygge robuste modeller. Med en robust modell menes det en 3D-modell som tåler relativt store endringer i parametre uten at modellen kræsjer. Avhengigheter definerer hvilke forhold som påvirkes av hverandre. Hvor robust en modell blir avgjøres av hvor godt disse avhengighetene er analysert og hvordan disse er satt opp.

Eksempelvis kan en lage en rektangel som styres av to parametre: høyde og bredde. Endres en av disse vil den andre stå uberørt. Dersom vi har en kvadrat hvor høyden alltid skal være dobbelt så stor som bredden, kan vi sette opp den samme modellen som styres av bare en parameter. Altså vil bredden være den styrende parameteren. Ved å endre på denne, vil høyden automatisk endres til det dobbelte. Vi sier da at høyden avhenger av bredden. Det skal relativt lite til før en modell blir vesentlig mer kompleks, og da vil det bli vesentlig mer utfordrende å se hvilke avhengigheter som gir robuste modeller. For å øke brukervennligheten er det ønskelig med modeller som styres av færrest mulig parametre.

2.1.3 Kompleksitet

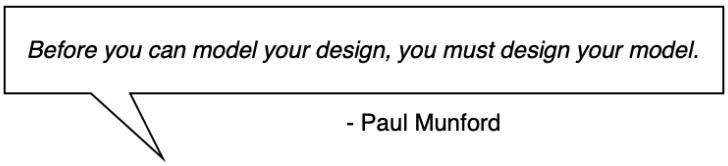
Hvor enkelt det er å definere de ulike avhengighetene, og deretter parametrisere en modell, vil i stor grad avhenge av modellens kompleksitet.

Det er funnet to definisjoner på hva som beskriver kompleksitet. Den første er at det defineres ut fra produksjonsmetoder. Altså om den kan framstilles i en dreiebenk eller etter lignende maskiner

regnes den for å ha primitive former og dermed ha lav kompleksitet [4]. En annen definisjon baserer seg på en undersøkelse blant DAK-brukere hvor de konkluderer med at en komponent regnes for å være kompleks dersom den består av mer enn 230 overflater [5]. Begge disse definere det videre grunnlaget for hva som menes med komplekse modeller i denne rapporten.

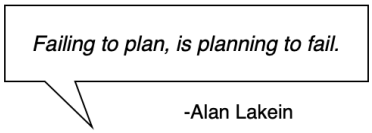
2.1.4 Design intent

Design Intent defineres som hvordan modellen oppfører seg når dimensjoner endres på [6]. Begrepet omhandler arbeidet i forkant av selve modelleringen. Altså planleggingen som er nødvendig for å få en robust og brukervennlig modell. *Design Intent* gir oss mulighet til å beskrive hvordan en modell burde oppføre seg ved oppdatering av en parameter [7].



Figur 2.1: Sitat om *Design intent*

Dette er et begrep som brukes spesielt i forbindelse med parametriserte modeller og det hevdes at tiden som brukes på planlegging i forkant spares inn fordi det er uunngåelige at endringen av modellen vil inntreffe [8]. Ved å ha en forståelse av hvilke endringer som med høyest sannsynlighet vil inntreffe, kan en bygge opp modellen etter dette.



Figur 2.2: Sitat om planlegging

2.2 Strategi for oppbygging av komponenter

I denne rapporten er det valgt å skille på strategier for oppbygging av komponenter og oppbygging av sammenstillinger. Dette delkapittelet tar for seg forskjellige måter å bygge opp komponenter, mens sammenstillinger er beskrevet i delkapittel 2.4.

Først er det et avsnitt som beskriver hvorfor det er viktig å modellere etter en strategi før de neste avsnittene tar for seg ulike strategier. I det siste avsnittet sammenlignes de ulike strategiene slik at det blir lettere å danne seg et overblikk.

2.2.1 Generelt

Ved å ha et bevisst forhold til hvilken metode som brukes til oppbygging av komponenter vil det påvirke gjenbrukbarheten. Dersom en bedrift etablerer en standard for hvordan bygge opp 3D-modeller i DAK vil dette kunne øke bedriftens effektivitet ved at det er lettere for flere å arbeide på samme modell [5]. Det finnes svært mange strategier og metoder for parametrisering som ulike bedrifter har etablert for å effektivisere eget arbeid. På grunn av konkurransefordelen dette kan skape holdes disse ofte hemmelig, men noen har valgt å dele kunnskapen slik at andre kan dra nytte av dette [1].

Ifølge Camba, Comtero og Company [1] kan strategivalg være avgjørende for hvor godt modellen fungerer til gjenbruk og hvor robust modellen er. Likevel påstås det at erfaringsgrunnlaget til den enkelte ingeniør vil ha større betydning for modellens robusthet enn hvilken strategi som velges

[5]. Dette fordi en må klare å forutse hva som skal endres senere og forstå hvordan modellen skal nyttes.

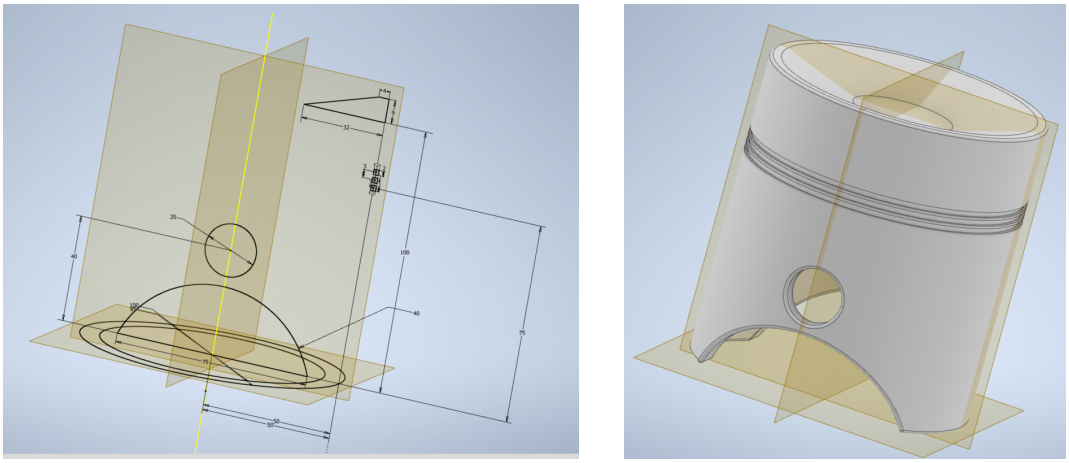
Studien til Y.Bodein et al. [5] tar for seg et forsøk hvor de bruker masterstudenter til å modellere. De har ingen kunnskaper utover at de har lært om hvordan DAK-programmets funksjoner fungerer. Det ble gitt noen timer opplæring underveis i eksperimentet, slik at de kunne modellere etter en bestemt strategi.

Den eneste måleparameteren nyttet i eksperimentet var tiden studentene brukte på å endre komponentene. Det viste seg at de brukte dobbelt så lang tid på å endre sin egen modell, som de brukte på å endre en modell bygget etter en strategi. Det viste seg også at det gikk mindre tid for studentene å bygge en ny modell, enn tiden de brukte på å endre på en modell en annen student bygget uten å følge en strategi.

2.2.2 Delphi´s Horisontal Modeling

Denne strategien for modellering av komponenter kalles *horisontal modeling* på grunn av den flate strukturen man bygger modellen etter. Dette er den første av tre strategier for komponenter som beskrives i denne rapporten.

En beskriver strukturen som flat på grunn av at hver *feature* tar utgangspunkt i et referanseplan, et referansepunkt eller en referanselinje. Det vil si at en aldri tar utgangspunkt i eksisterende geometri for å bygge neste *feature* [9]. Hensikten med en slik oppbygging er at hver *feature* skal kunne endres uavhengig av andre og på denne måten reduseres risiko for at modellen kræsjer ved endringer.



(a) Viser hvordan komponenten er bygget opp med utgangspunkt i plan, akser og origo

(b) Komponent designet med *Horizontal modeling strategy*

Figur 2.3: Horizontal modeling

På tross av at det å sette opp plan, linjer og punkter i forkant krever ekstra planlegging, skal *design tree* oppfattes som enkel, oversiktlig og intuitivt å overta av en annen ingeniør [1]. Derimot vil slik planlegging stille krav til ingeniøren sitt erfaringsnivå og innsikt.

Det er nettopp denne flate strukturen metoden får mest kritikk for: Det er stor enighet om at avhengigheter mellom geometri i en modell vil være svært nyttig så lenge disse brukes på riktig måte. Det kan også sies at denne avhengigheten er selve grunnpilaren for parametrisering [1]. I tillegg er dette i USA en patentert metode som ikke kan benyttes uten lisens fra *Delphi Technologies, Inc*, dog ser det ikke ut til at det er patentert i Europa da en ikke finner noe om denne patenten på hjemmesidene til det Europeiske patentkontoret sine hjemmesider [10].

2.2.3 Explicit Reference Modeling

Explicit Reference Modeling er den neste strategien som beskrives i denne rapporten. Denne har noen av de samme prinsippene som *Horizontal*, men den gir litt større rom for å vurdere hva en vil referere til.

Når en modellerer i et DAK-program kan en velge mellom to metoder for å sette fastholdninger til sine neste *features*. Enten defineres fastholdningen ut fra eksisterende geometri eller så defineres det ved hjelp av plan, linjer, punkter og lignende. Disse to forskjellene er vist i tabell 2.1.

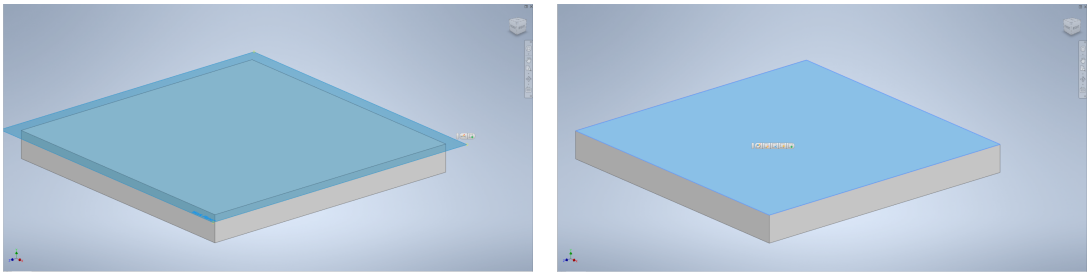
Tabell 2.1: Inndeling av fastholdninger i *Explicit reference modeling*

Klasse	**Fastholdninger**	**Eksempler**
1	Fastholdninger som **ikke må** defineres av eksisterende geometri	*Extrude*, *cut*, *revolve*
2	Fastholdninger som **må** defineres ut fra eksisterende geometri	*Chamfer*, *fillets*,

Ut fra denne klassifiseringen ble *Explicit Reference Modeling* strategien skapt av Bodein, Rose og Caillaud [5]. Målet med denne er å minimere bruken av fastholdninger som hører til i klasse to, men som nevnt er det ikke like strengt som i *Horizontal modeling*. Forfatterene mener at det meste av geometri både kan og bør defineres ut fra punkter, plan eller linjer. Selv om en i de fleste tilfeller kunne definert det ut fra eksisterende geometri, burde en begrense det til å kun utføres for lokale modifiseringer som *chamfer*, *fillets* og lignende.

Ved å utføre dette på denne måten, i tillegg til å holde skissene enklest mulig, vil en i følge forfatterene få mer robuste modeller som tåler å endres på. Ved å gjøre skissene mer komplekse vil det i større grad kreve topologiske endringer som er vanskelig å parametrisere.

I figur 2.4 er det vist forskjellen mellom klasse 1 og klasse 2 referering. I figur 2.4a er det markert et plan som ligger oppå flaten, mens i figur 2.4b er overflaten til modellen markert. Ved å hele tiden strebe etter å bruke klasse 1 referering kan en holde en flatest mulig struktur. Ved å ha en flat struktur vil en kunne gjøre endringer på en *feature* uten at dette skal påvirke andre *features*. Er det flere *features* som er bygget oppå hverandre, som eksempelvis klasse 2 bygget på en klasse 2, vil de øvre kunne kræsje dersom det utføres endringer på noen av de i bunnen. I tillegg skal alle klasse 2 refereringer holdes nærmest mulig mor-geometrien.



(a) Klasse 1 (b) Klasse 2

Figur 2.4: Forskjellen på klasse 1 og klasse 2 referering

2.2.4 Resilient Modeling

Resilient modeling strategy, RMS, er den siste av de tre strategiene som beskrives. Her deles alle *features* inn i seks kategorier. Målet med dette er å få en nøytral løsning på ustabile modeller ved å strukturere designtreet inn i sekvenser. Disse sekvensene er organisert etter hvor viktig de er, funksjonen og hvor sannsynlig det er at de påvirker andre features. Dette skal maksimere fleksibilitet og robusthet ved DAK-modellen og minimere uoverensstemmelser [1].

I *Resilient modeling strategy* beskrives et sett med regler som former et tankesett, og det hevdes at dersom en mestrer å følge disse, vil en klare å bygge robuste modeller. Videre i denne rapporten vil det være RMS sin *best practice* som beskrives. Det er dette som er rettet mest mot den praktiske gjennomføringen. For en som virkelig ønsker å beherske dette anbefales det å lese mer på hjemmesiden [4].

Målet med å dele opp alle *features* i grupper er å definere tydelig hvor det kan etableres linker og i hvilken rekkefølge *features* skal bygges. I tillegg skapes det et designtre som er lett gjenkjennelig for andre som kan RMS og på denne måten blir det enklere å overta en modell. Et annet viktig moment er at det anbefales at hver *feature* navngis etter hva de er designer for, og ikke etter hvordan de er laget.

I gruppe 1 er det referanseplan, -punkt, -linjer eller lignende som skal være tilgjengelig gjennom hele modelleringsprosessen. Dersom det er nødvendig med konstruksjonsgeometri slikt som kurver, *paths* eller overflater, hører disse hjemme i gruppe 2. I gruppe 3 er hovedstrukturen til modellen som definerer den grove utformingen av modellen. Altså *features* som bygger materiale, slik som *extrude* eller *sweep*. Drastiske endringer på hovedgeometrien vil kreve endringer innenfor denne gruppen. Geometriske features som fjerner materiale hører til under gruppe 4. *Features* som ikke krever noen *child nodes* hører til i gruppe 5 og 6. Gruppe 5 inneholder slikt som *pattern* og symmetriske elementer. Denne gruppen er dog valgfri å bruke. Kosmetiske og avsluttende *features* som *chamfer* tilhører gruppe 6 og lages til sist. Alle gruppene er vist i tabell 2.2.

Tabell 2.2: *Features*-grupper i *resilient modeling* strategien [1]

Gruppe	**Beskrivelse**	**Eksempler**	**Merk**	**Linker**
1 - Ref	Alle referanser som lages først og gjøres tilgjengelig/synlig for alle andre *features*	Ref *bodies*, skisser, Ref plan, koordinatsyst., bilder	Ingen Solid	Om den er synlig, er det akseptabelt å linke til denne
2 - Construction	Konstruksjon*features* som overflater eller 3D-kurver som nyttes til å definere komplekse solide *features*	Overflater, *Project*, *Extend*, 3D-kurver, *Trim, Split*	Ingen solid	
3 - Core	En «*Super Bases Feature*» som styrer modellens utforming og orientering	*Extrude, Aweep, Thin Wall, Revolve, Loft, Shell*	Legg til material	
4 - Detail	Detalj*features* som utgjør den endelige utformingen ved å kun linke til *core*gruppen	*Extrude, Sweep, Hole Revolve, Loft, Thread*	Fjern materiale	Linker til andre grupper er akseptert, men ikke til internt i *detail*-gruppen
5 - Modify	Til *faces* eller gjenskape *features* for så å legge til *final features*	*Draft, Pattern, Mirror, Final Features*		Om den er synlig, er det akseptabelt å linke til denne
6 - Quaratine	Flyktige *features* som ikke burde være *parent*	*Chamfer, Blend, Round*	Størst først	

Det er viktig at slike detalj-*features* som er i gruppe 5 og 6 ikke brukes som *parent node* med mindre det er absolutt nødvendig. Disse detaljene burde alltid styres av en *parent node* som tilhører gruppe 1, 2 eller 3. For å kunne kvalitetsikre modellen ytteligere gjennom denne metoden har RMS laget en sjekkliste for å sikre riktig oppbygging av modellene. Denne sjekklisten kan også nyttes til å dele informasjon innad i DAK-gruppen.

2.2.5 Sammenligning av strategier

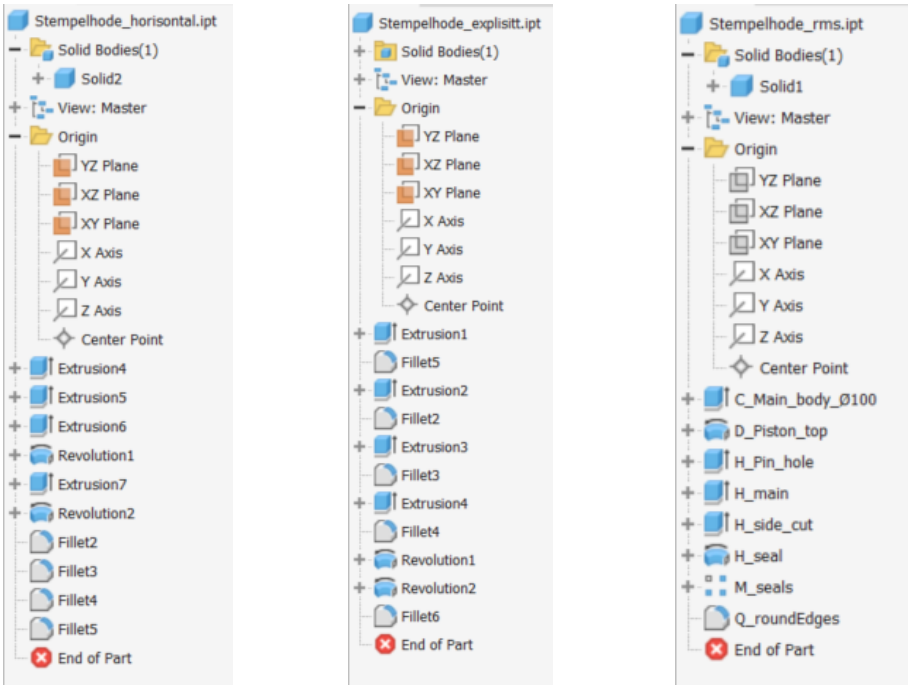
Sammenligningen baserer seg på resultatene i artikkelen til Camda, Contero og Company [1]. Tiden det tar å endre en modell er ulik avhengig av hvilken strategi modellen er bygget etter. Det brukes lengst tid på *Horizontal* og minst tid på RMS. I tillegg var det flest som endte opp med feil modell da de skulle endre på *Horizontal*, noen hadde feil med *Explicit*, men alle leverte korrekt modell etter å ha endret på modellen bygget etter RMS.

En annen ting de oppdaget var at selv om det ikke kom opp noen feilmelding etter endring i en modell bygget i *Horizontal*, kunne det allikevel være store feil i modellen. Dette kan derfor medføre større risiko å nytte denne strategien for en parametrisert komponent.

Tabell 2.3: Sammenlikning av strategier for oppbygging av komponenter

Strategi	**Styrker**	**Svakheter**
Delphis horisontal modeling	Flat struktur Enkelt og intuitivt *design tree*	Kan ikke utnytte avhengigheter Patentert/lisensert
Explicit reference modeling	Enkel klassifisering	Begrenset mulighet til å utnytte avhengigheter
Resilient modeling	Gir robuste modeller Gir brukervennlige modeller	Omfattende å sette seg inn i

Tabell 2.3 oppsummerer fordeler og ulemper med de ulike strategiene og i figur 2.5 er det illustrert hvordan *design tree* vil se ut for de tre metodene. Det er stempelhodet vist i figur 2.3 som er bygget på etter de tre ulike strategiene.



(a) *Horizontal modeling* (b) *Explicit reference modeling* (c) *Resilient modeling*

Figur 2.5: Sammenligning av *Design tree* til komponent

Design tree kalles også *History tree* fordi her kan en se hvordan en komponent er modellert og det påstås en skal kunne hente ut *Design Intent* fra denne [5]. Det er dette stedet neste ingeniør må navigere for å kunne endre på parametre til en komponent og derfor vil organiseringen av *Design tree* ha stor betydning for hvor intuitivt det er å overta en modell.

Resilient er den eneste strategien hvor du skal navngi hver *feature* etter hva den er og ikke hvordan den er laget. Dette gjør at det er mye lettere for neste person å navigere i *Design tree*. I tillegg vil rekkefølgen til hver *feature* i *Design tree* bidra til oversiktlighet ved at en vet hvilke *features* som henger sammen. Derfor er det slik i *Explicit* at de ønsker å holde klasse 2 referering nærmest mulig mor-geometrien, som er en muligheten man mister i *Horizontal* siden der skal alt slikt som *fillets* og *chamfer* lages til sist.

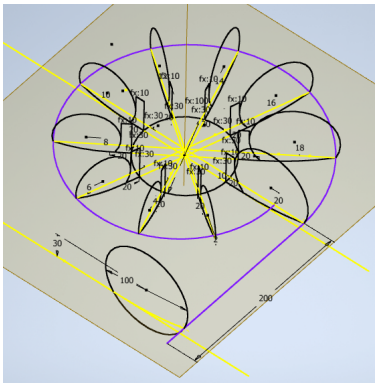
2.3 Organisk formede komponenter

I dette delkapittelet beskrives ulike teknikker for hvordan modellere komponenter med organsike former. Fellesnevneren for alle disse er at de har få rette linjer og kan derfor oppleves som vanskelig å parametrisere og dermed må en kanskje tenke noe annerledes enn med andre komponenter. Et eksempel er geometrier med samme utforming som sneglehus, er det ofte radier en referere til for å parametrisere modellene. Strategiene i delkapittel 2.2 er fortsatt gjeldene for disse, men det må nyttes andre modelleringsverktøy for å lykkes.

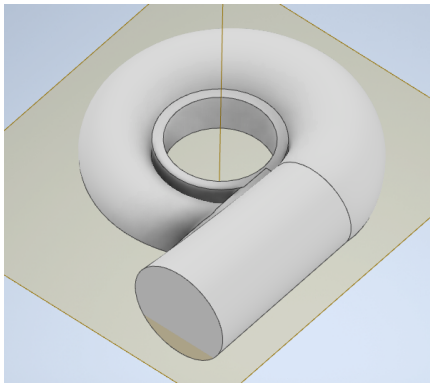
Modelleringsprogrammene har ulike verktøy for å lage organiske former. I denne rapporten er to av de som er regner for å være mest aktuelle omtalt: *Sweep* og *Loft*. I tillegg er Programvaren CAESES beskrevet i vedlegg B.2. Denne er ikke tatt med inn i hovedrapporten siden det ikke er vurdert til å være generell teori som kan brukes i andre modelleringsprogrammer, men den er et svært godt verktøy og er derfor valgt å nevnes.

2.3.1 Loft

Denne funksjonen er laget for å modellere en komponent med et varierende tversnitt [11]. Verktøyet bruker flere skisser for å definere tverrsnitt og deretter lage en solid eller surface. Hvordan dette ser ut er vist i figur 2.6. Dess flere skisser en lager dess bedre kontroll vil en få på komponentens utforming og programmet vil selv generere overgangen i tversnittet mellom hver skisse. For å ytterligere øke kontrollen kan man også definere noen hjelpelinjer for å utforme overgangene selv.



(a) Mange skisser for å definere tversnitt

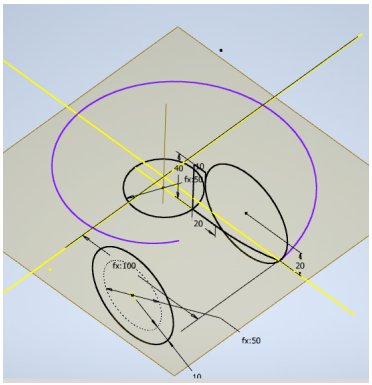


(b) Selve komponenten

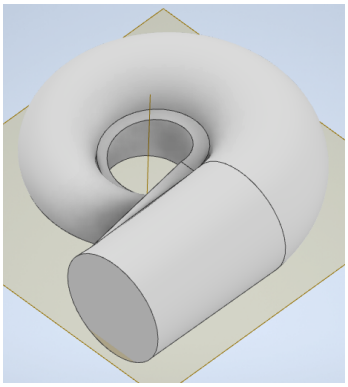
Figur 2.6: Organisk utformet komponent med *Loft*

2.3.2 Sweep

Denne er laget slik at en kan få et gitt tverrsnitt til å følge en linje i rommet som vist i figur 2.7 [11]. Dette verktøyet gir mindre kontroll over tversnittet som betyr at den er lite egnet for komponenter med brå overganger, men siden en bare behøver å definere ett tversnitt er dette er hurtigere og enklere modelleringsteknikk. Det er også muligheter for å lage en hjelpelinje slik at hele geometrien skalerer seg.



(a) Skissene som definerer hele komponenten



(b) Selve komponenten

Figur 2.7: Organisk utformet komponent med *Sweep*

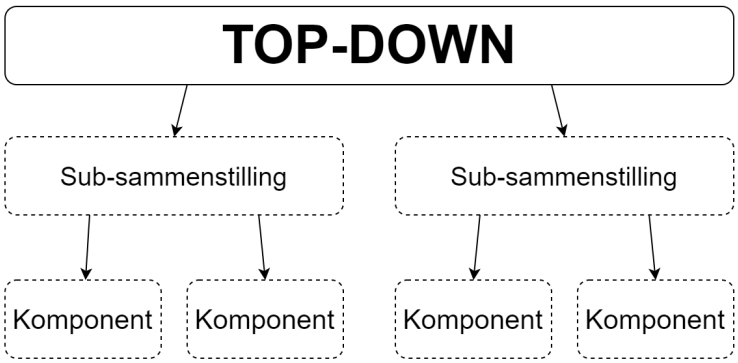
2.4 Strategi for oppbygging av sammenstillinger

I avsnitt 2.2 er det beskrevet ulike strategier for hvordan hver enkelt komponent kan bygges opp for å få robuste modeller. I dette avsnittet vil det beskrives ulike strategier for å parametrisere sammenstillingsmodeller. For å bygge robuste sammenstillinger må det legges en strategi for hvordan de ulike komponentene skal samhandle.

Det beskrives tre ulike mulige hovedmåter å tenke på for å bygge opp en parametrisert sammenstilling [12]. Disse er *Top-Down*, *Bottom-Up* eller *Middle-Out*, hvor den sistnevnte er en kombinasjon av de to første. Dette er uttrykk som er vanlig å bruke om strategier for å prosessere informasjon. Den generelle betydningen av disse er at *Top-Down* betyr å starte med det store bildet for så å bryte dette ned til detaljnivå [13], *Bottom-Up* er å sette sammen enkeltdeler for å skape et større og mer komplekst system [14]. *Middle-Out* er når det er en del som er førende for hvordan resten blir [12].

2.4.1 Top-Down Design

Det finnes ulike strategier for modellering innenfor *top-down design*, TDD. Felles for disse er at de bruker noe overordnet (et skjelett) for å dele informasjon mellom alle komponenter [12]. Akkurat hvordan dette gjøres er løst på ulik vis. Noen av disse ulike metodene vil beskrives nedenfor. Hensikten med å bygge opp en modell på denne måten, er at en skal kunne endre på alle komponenter ved å kun endre på det overordnede skjelettet. Dette prinsippet er vist i figur 2.8.



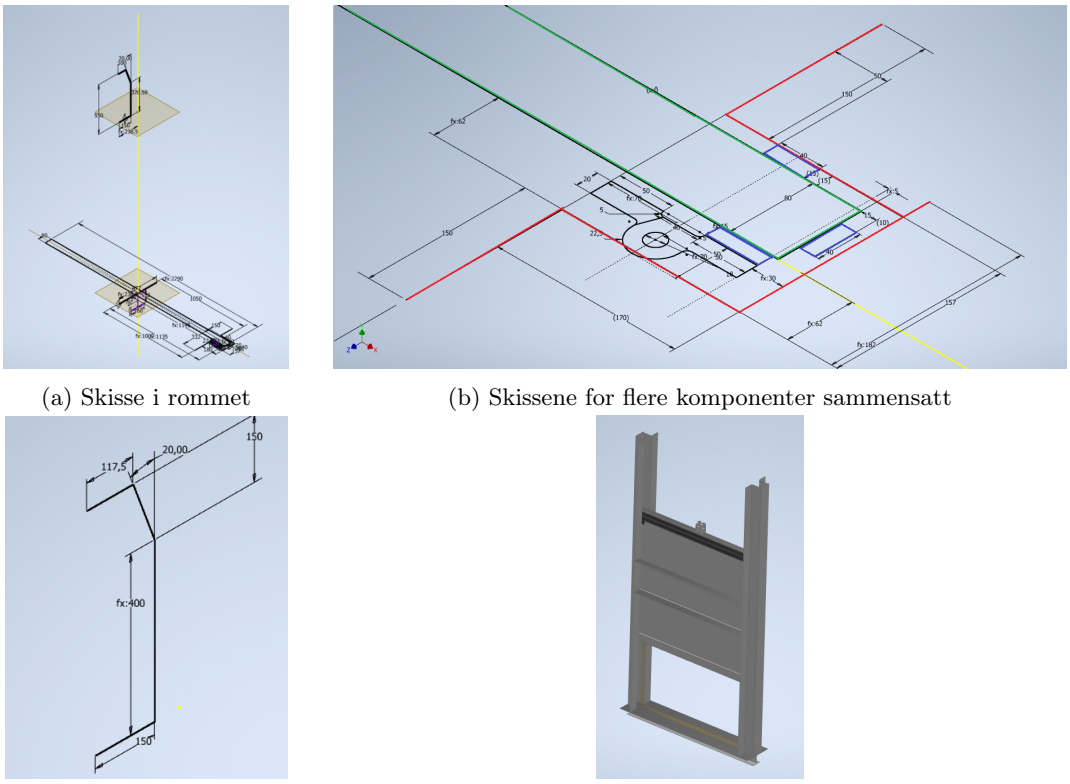
Figur 2.8: Prinsippet i et *Top-Down* design

Dette er en metode som gjerne nyttes der hvor det er spesifikasjoner som overstyrer alt det andre. Dette er i utgangspunktet en unik konstruksjon hvor det er forventet at alt må spesiallages [12]. Dersom det likevel er noen deler i denne som er hyllevarer, slikt som eksempelvis bolter, kan disse lages som adaptive komponenter. Ved å gjøre disse adaptive vil delene rundt tilpasse seg dem, eksempelvis at et hull tilpasser seg bolten som skal plasseres i den.

I denne rapporten er det tre metoder innenfor *Top-Down* som skal beskrives ytterligere. Dette er *Single Skeleton*, *Multi Skeleton* og *Multi-body part*. Det som skiller de to skjelett-metodene er om all informasjonen er i et skjelett eller om du har delt det opp i ulike skjelett avhengig av hvilken type informasjon de inneholder. Den siste metoden er en metode utviklet for sammenstillinger hvor det ellers er vanskelig å få komponentene til å passe sammen. Dette er beskrevet i detalj i et avsnitt lengre ned.

Single skeleton

Denne metoden kalles *Single Skeleton* fordi informasjon lagres innenfor et skjelett. Det lages flere skisser innenfor en *part*-fil, altså det overordnete skjelettet, og videre lages hver komponent fra skissen [15]. Dette er vist i figur 2.9b. Noen av parametrene som skaper et toppnivå som de andre parametrene referer til. Dette betyr at før en starter å modellere må en bestemme seg for hvilke parametre som skal være førende for hele modellen.



(a) Skisse i rommet

(b) Skissene for flere komponenter sammensatt

(c) Skisse toppstokk

(d) Resulterende modell

Figur 2.9: Skeleton modeling

Det kan fortsatt lages flere nivåer av sammenstillinger innenfor *Single Skeleton*. For å holde det oversiktlig anbefales det å holde seg til maks tre nivåer: sammenstilling, sub-sammenstilling og komponent som vist i figur 2.8 [16]. Her vil skjelettet være hovedsammenstillingen, hver skisse vil være en sub-sammenstilling og hver blokk som drives ut av skissen vil være en komponent.

Dette er en metode som egner seg ved enkle og/eller enkeltstående konstruksjoner. Glideluker er et godt eksempel på dette.

Multi-skeleton

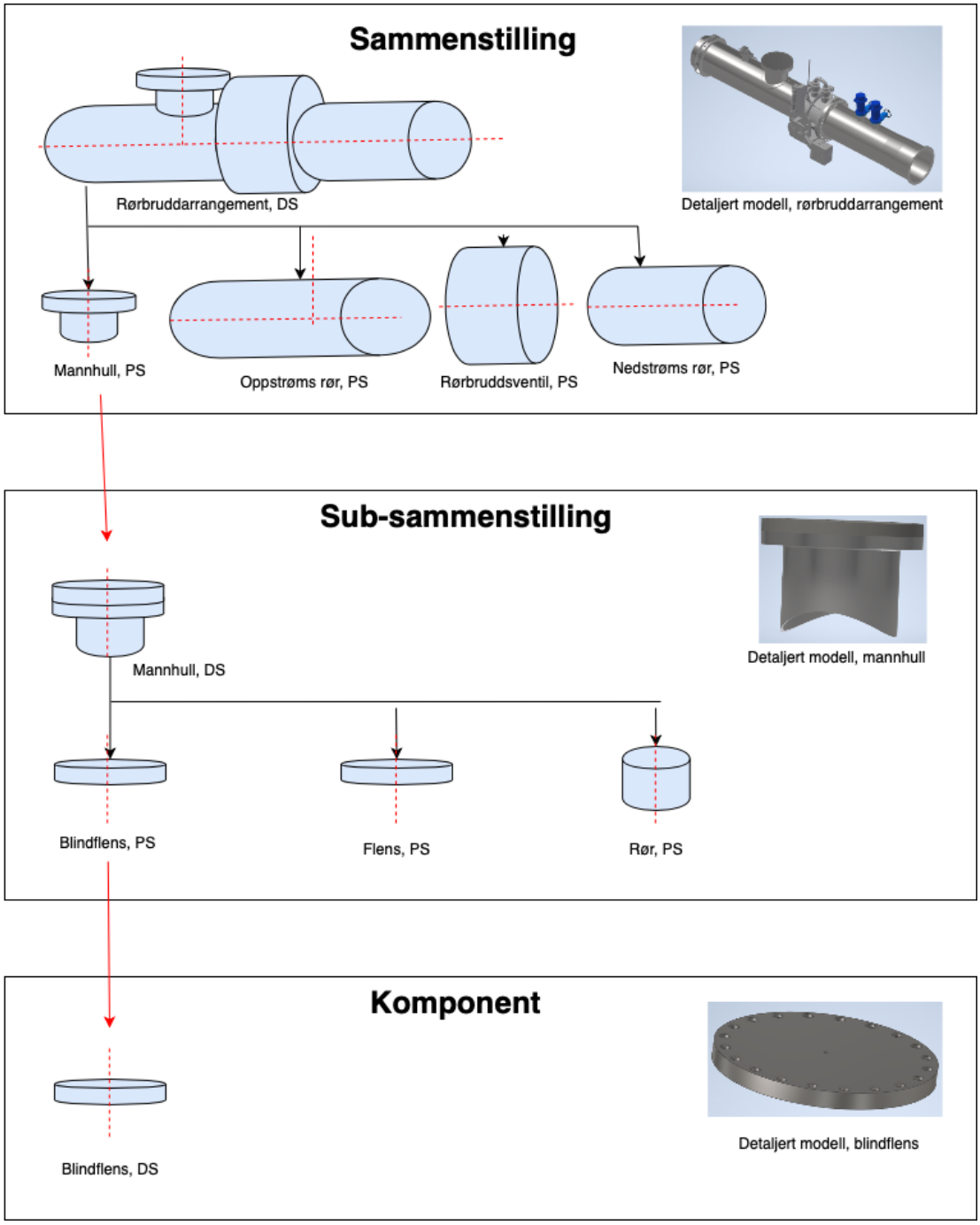
I større modeller kan det bli store mengder informasjon som det er vanskelig å definere innenfor et skjelett [13]. Eksempler på dette kan være en rørgate med flenser og ventiler, eller en turbin med de omkringliggende komponentene. Her vil du ha større behov for å tydelig få fram informasjon om hvordan de ulike komponentene skal plasseres i forhold til hverandre.

Dexin, Guolin, Xuening og Jing foreslår en metode hvor en kan strukturere all informasjonen, *multi-skeleton* [16]. Målet med denne strategien er at det kun er nødvendig informasjon som skal deles med de ulike delene av modellen. For å få en kompleks sammenstilling robust er det nødvendig å ha god oversikt over hvilke parametre som hører hjemme på de ulike stedene. Dette har de løst ved å dele de ulike typene informasjon inn i tre grupper: LS, PS og DS. Disse er forklart i tabell 2.4 og i figur 2.10.

Tabell 2.4: Skjelettinndeling i *Multi-Skeleton*

LS	**Location Skeleton Model**	**Beskrivelse** Dette er parametrene som forteller hvor ting skal plasseres i forhold til hverandre, altså det som orienterer de ulike komponentene i rommet. Den skaper et kart som sørger for at alle designenhetene passer sammen. **Eksempel** Posisjonsreferanser, grensesnitt og *feature* posisjoner.
DS	**Design Skeleton Model**	**Beskrivelse** En DS består av flere PS satt sammen og viser hvordan disse plasseres i forhold til hverandre. Formålet til DS er å vise hvilke komponenter eller sub-sammenstillinger PSen på dette nivået består av. På øverste nivå er den svært overordnet og grovt utformet, men vil bli mer og mer detaljert på lavere nivå. Det er ut fra det laveste nivå av DS at detaljmodellen utformes. **Eksempel** Grove figurer av sammenstillingen uten detaljer
PS	**Published Skeleton Model**	**Beskrivelse** Hver PS genereres ut fra sin overordnede DS. Formålet med en PS er å ta med informasjon om hvordan den skal kobles sammen med de andre PSene på samme nivå. **Eksempel** Sammenstilling/komponent uten detaljer.

Figur 2.10 viser et eksempel på sammenhengen mellom LS, DS og PS. Den viser at en DS på det øverste nivået er veldig overordnet og viser hele rørbruddsventilen i grove trekk. Denne innholder kun den informasjonen som er nødvendig for å knytte sammen de ulike PSene som hører til på dette nivået. I figuren vises det hvordan DS for mannhullet ville sett ut og videre brytes dette ned til PSer som tilhører dette nivået. Slik fortsetter det ved å kun ta med den informasjonen som er nødvendig for neste steg, fram til en er på komponentnivå. For hver DS finnes det en detaljert modell som viser det endelige produktet. Se tabell 2.4 for ytterligere beskrivelse.



Figur 2.10: Sammenheng mellom PS og DS i *multi skeleton modelling*

Før en starter med denne metoden er det nødvendig å planlegge hvor mange nivåer som er nødvendig i sammenstillingen, hvilke type skjeletter som skal hvor, hvor de ulike parametrene hører hjemme og hvordan overføre parametre mellom nivåer og innad i nivået. For at dette systemet med flere skjelett skal fungere må det være noen regler for hvordan parametre deles mellom de ulike nivåene.

Regel 1: Hvis det er avhengighet mellom designenheter på to forskjellige nivåer, så må den på lavest nivå la seg styre av den på høyere nivå.

Regel 2: Hvis det er avhengighet mellom to designeneheter på samme nivå, så må man gjøre om slik at de begge kan hente parametre fra et felles skjelett på et høyere nivå.

Også her ønsker en å holde seg innenfor tre nivåer. Dersom det blir et for stort hierarki av sub-sammenstillinger og komponenter risikerer en at det dannes relasjoner/avhengigheter til andre komponenter på samme nivå [16], eller at en komponent blir direkte avhengig av en annen komponent. Målet er en flatest mulig struktur for å unngå at modellen blir unødvendig høy.

I tillegg til at det deles inn i tre typer skjelett, deles også parametre inn i tre grupper etter type [16]. Noen beskriver posisjoner i rommet, noen beskriver dimensjoner og andre beskriver detaljer på en komponent. Det er ikke alle parametre som er relevant alle steder i modellen. Ved å dele de inn i grupper er det lettere å vite hvilket skjelett de hører hjemme i. Gruppene er beskrevet i tabell 2.5.

Tabell 2.5: Parameterinndeling i *multi-skeleton*

LP	**Interface and Location Design Parameter**	**Beskrivelse** Dette er de parametrene som styrer hvordan komponenter plasseres og beveger seg i forhold til hverander. Dette skal arves fra nivået over. **Eksempel** Fastholdninger, lokalisering
KP	**Key Product Design Parameter**	**Beskrivelse** Dette er de viktige parametrene som definere selve komponenten. Altså det som omhandler utformingen av hovedgeometrien. Også dette er parametrene som må arves fra nivået over. **Eksempel** *Extrude, revolve*
GP	**General Product Design Parameter**	**Beskrivelse** Detaljer som kun er viktig for den ene enheten. **Eksempel** *Chamfer*

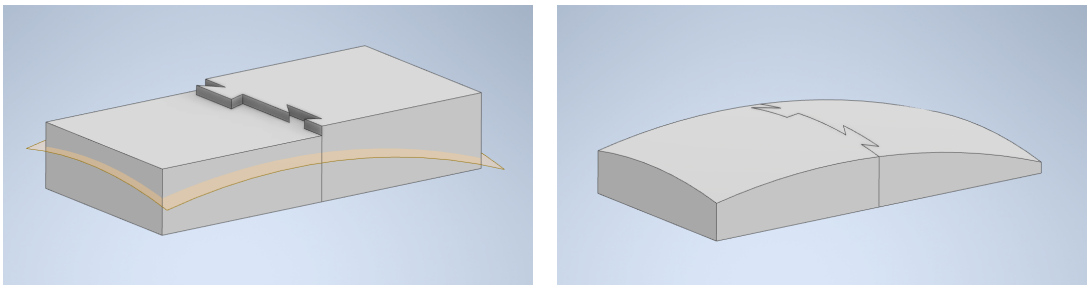
For å oppsummere hvordan informasjonsflyten er i *Multi-Skeleton* kan en si at skjelett LS vil holde styr på hvor alle delene er i sammenstillingen og hvordan de samhandler med hverandre. Sammenstillingen og hver sub-sammenstilling vil derfor ha en LS med informasjon om seg selv og alle underenheter. I skjelettet PS vil en kun finne informasjon som skal videreformidles mellom to designenheter. Det deles altså aldri informasjon direkte mellom LS og DS, det er kun gjennom en PS det formidles parametere som LP og KP. Dette bidrar til å sikre at en ikke får flere koblinger enn det som er nødvendig. Det er i det siste skjelettet, DS, hvor all informasjon for den aktuelle enhet vil lagres.

Hvert skjelett inneholder dermed følgende informasjon:

LS - LP
PS - LP og KP
DS - LP, PS og GP

Multi-body part

For å forenkle utfordringen med å modellere sammenstillinger hvor det er kurver eller linjer som skal bevege seg sømløst over flere komponenter kan en ta i bruk et metode som kalles *Multi-body part*. Her starter en med den ytre geometriske utformingen av en sammenstilling for deretter å legge til detaljer og dele det opp i flere komponenter [17].



(a) To komponenter ekstrudert og plan er definert (b) Ferdig sammenstilling

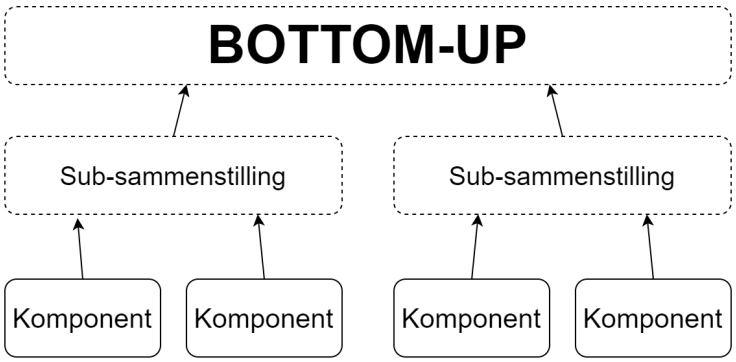
Figur 2.11: Eksempel på en sammenstilling hvor du typisk nytter *Multi-body part*

Et eksempel på en sammenstilling som typisk designes ved hjelp av denne metoden er vist i figur 2.11. Her ble sammenstillingen bygget ved at det først ble laget en skisse for grunnflaten. Deretter ble komponentene ekstrudert i to omganger ut fra skissen som vist i figur 2.11a. I den figuren er det også vist planet som ble laget for å definere krumningen. Det ble kuttet etter planet og fikk på denne måten en jevn overgang i overflaten til begge komponentene. På denne måten har en sikret at komponentene passer i hop der de kobles sammen og der er en sømløs overgang i overflaten.

På denne måten behøver en heller ikke definere fastholdninger mellom komponentene da de defineres ut fra skissen. Dette betyr at denne metoden ikke egner seg for sammenstillinger med bevegelige deler [18]. Videre lagres hele sammenstillingen i en fil, men i tillegg genereres det en egen *part*-fil for hver komponent.

2.4.2 Bottom-Up Design

Dette er den tradisjonelle måten å bygge en sammenstilling på. Denne strategien baserer seg på å plassere eksisterende deler i en sammenstilling for så bruke fastholdninger på vanlig måte for å posisjonere hver enkelt komponent [12]. Dette prinsippet er vist i figur 2.12 Det anbefales at dersom det er mulig, burde delene plasseres i samme rekkefølge som de ville blitt montert på et verksted.



Figur 2.12: Prinsippet i et *Bottom-Up* design

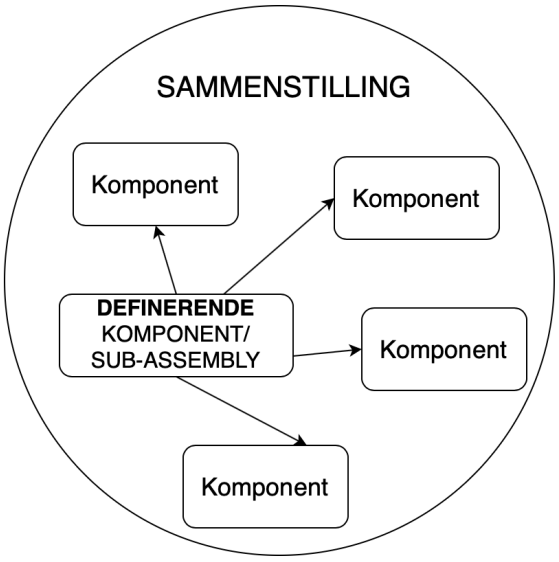
For at sammenstillingen skal fungere som en parametrisert modell er det viktig at hver komponent er designet med adaptive *features* i sin *part*-fil [12]. Det er også muligheter for å plasserene en komponent i sammenstilling for så å gjøre den adaptiv i sammenstillingen. Hele modellen linkes sammen ved å linke sammen *featurene* til de enkeltstående komponentene med hverandre.

Dersom det er ønskelig at alle underdefinerte *features* skal tilpasse seg så snart den plasseres i sammenstillingen, kan en lage en sub-sammenstilling som er adaptiv. Så snart sub-sammenstillingen linkes til en *fixed geometry* vil komponenten endre størrelse etter behov.

Siden en etterhvert vil bygge opp et bibliotek av komponenter, vil dette være en god metode dersom en ofte nytter standardiserte deler. En kan hente ut de komponentene det er behov for og bygge unike sammenstillinger av dette.

2.4.3 Middle-Out Design

Som allerede nevnt er dette en strategi som kombinerer *Bottom-Up* og *Top-Down* design. Denne strategien er nyttig dersom en har en komponent som styrer utformingen på resten av sammenstillingen [**?**]. Dette prinsippet er illustrert i figur 2.13. Et godt eksempel på dette kan være skovlehjul i en turbin. Det er størrelsen på denne som definerer hvordan resten av kraftstasjonen skal utformes. Dermed settes denne først inn i sammenstillingen, avhengighetene analyseres og de resterende komponentene designes etter denne.



Figur 2.13: Prinsippet i et *Middle-Out* design

Dersom det er noen av de andre komponentene som skal brukes i flere sammenstillinger kan en sette de som adaptive. For å endre størrelse eller posisjon på noen *features*, kan disse stå som ikke dimensjonert. På denne måten kan størrelsen og geometrien endres etter resten av sammenstillingen. Her er det viktig å tenke på at endringer av en komponent vil lagres i *part*-filen, derfor er det viktig å ha mer enn en versjon av denne filen.

En viktig funksjon for å skape adaptive komponenter i en *Middle-Out* er *project geometry*. Denne gjør at en kan projisere allerede eksisterende geometri inn i en annen skisse. Disse skissene kan også eksistere i andre plan enn geometrien.

2.4.4 Sammenligning av strategier

Ved hjelp av tabell 2.6 kan en raskt danne seg et overblikk over hovedtrekkene ved de ulike strategiene for å se hvilke fordelere og ulemper de har. Dette verktøyet kan dermed hjelpe til med å velge hvilken strategi som vil være mest hensiktsmessig for hvert tilfelle.

Tabell 2.6: Sammenlikning av strategier for oppbygging av sammenstillinger

Strategi	**Styrker**	**Svakheter**
Top-Down	Enklere å styre komplekse systemer. Ingen *constraints* (Multi-body)	Hver komponent må designes for hvert nytt system.
Bottom-Up	Kan gjenbruke komponenter i andre konstruksjoner og på sikt bygge et bibliotek av standardiserte deler. Hurtig å sette sammen et system dersom det består av standardkomponenter.	Ingen avhengigheter mellom komponentene
Middle-Out	Gjenbruk av standardkomponenter Kan tilpasse spesialkomponenter Fungerer godt dersom det er en komponent som styrer utformingen på resten av konstruksjonen	

2.5 Digitale tvillinger

Begrepet «Digitale tvillinger» defineres som å skape en digital tvilling av et fysisk objekt [19]. Dette i seg selv er ikke interessant med mindre denne modellen knyttes sammen med annen type informasjon. Det kan være varierende hva en konkret ønsker å oppnå med nettopp denne modellen, men overordnet dreier det seg om å effektivisere bedriftens ressursbruk [20]. Slike tvillinger gjør det også enklere å lage kompliserte modeller og en kan eksempelvis nytte den til feilsøking eller til å predikere mulige utfall.

Når det er snakk om en digital tvilling i denne rapporten menes det å koble 3D-modellen sammen med beregnings- og/eller analyseprogrammer. Ved at disse programmene snakker sammen for å utvikle modellen innenfor de parameterne som er satt, vil dette gå raskere enn å gjøre alle operasjonene separat. Ved at det kun er ett sted en setter inn parametrene vil en lette på mange manuelle operasjoner og en slipper at det er et sted en har glemt å oppdatere dersom det utføres endringer.

I tillegg vil det å koble sammen matematisk modell, 3D modell og analyseobjektet tilrettelegge for å drive optimalisering av et objekt. Det er svært mange av programvarene som har innebygde funksjoner for å drive optimalisering og det finnes ulike tilnærminger til dette. Dette beskrives nærmere i avsnittene som følger.

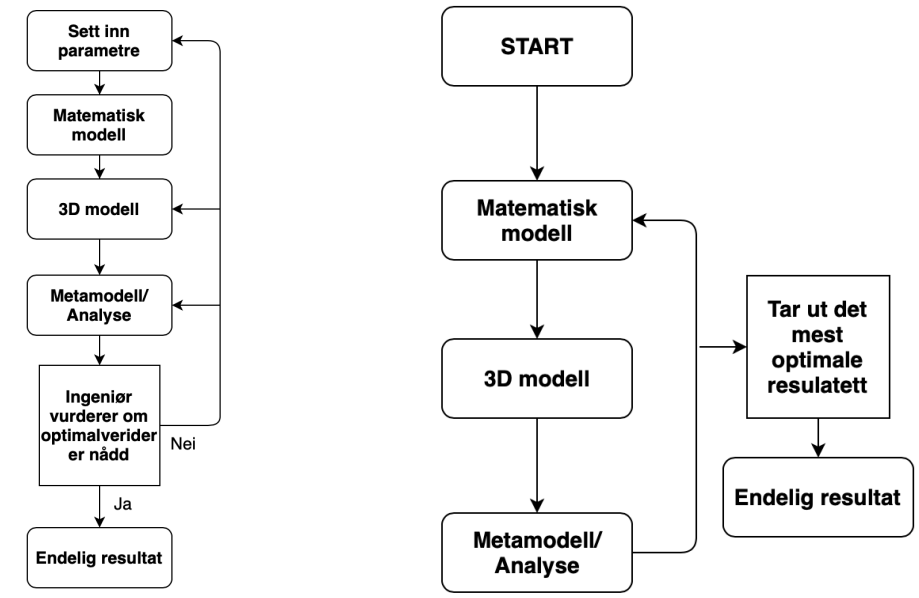
Det er også en mulighet å lage egne programmer som styrer datadelingen. Dette krever at en behersker koding som er en ferdighet som blir stadig viktigere fordi den i større og større grad implementeres i samfunnet vårt.

2.5.1 Optimalisering av modell

I en optimaliseringsprosess ønsker vi gjerne å minimere en kostnadsfunksjon ved å endre et sett designvariabler og samtidig tilfredstille et sett av begrensinger [21]. Et forenklet eksempel kan være en damluke, hvor kostnadsfunksjonen er stålmassen, og designvariablene er platetykkelsene, høyde og bredde. Begrensingene er min- og maksverdier for hver designvariabel, og et krav om at spenningene holdes under en gitt verdi.

For å få et godt utbytte av optimaliseringen er det viktig med god helhetsoversikt for å avgjøre hva som faktisk burde optimaliseres, og hvilke deler som bør styres av det som optimaliseres [21]. Eksempelvis i en vannvei: Er det luken, støping av kanalen eller effektapet som vil generere størst kostnader i det store bildet?

Etter at denne analysen er gjennomført kan en skape en digital tvilling av det som skal optimaliseres. På denne måten vil en kunne prosessere mer informasjon om modellen enn om informasjonen er fordelt på ulike programmer. I denne delen av prosessen må en ta et valg om optimaliseringen skal utføres automatisk, eller om en ingeniør skal inn å gjøre vurderinger mellom hver prosessloop. Et eksempel på disse to forskjellene er vist skjematisk i figur 2.14.



(a) Semi-automatisert (b) Automatisert

Figur 2.14: Alternativer til hvordan bygge opp en optimaliseringsprosess

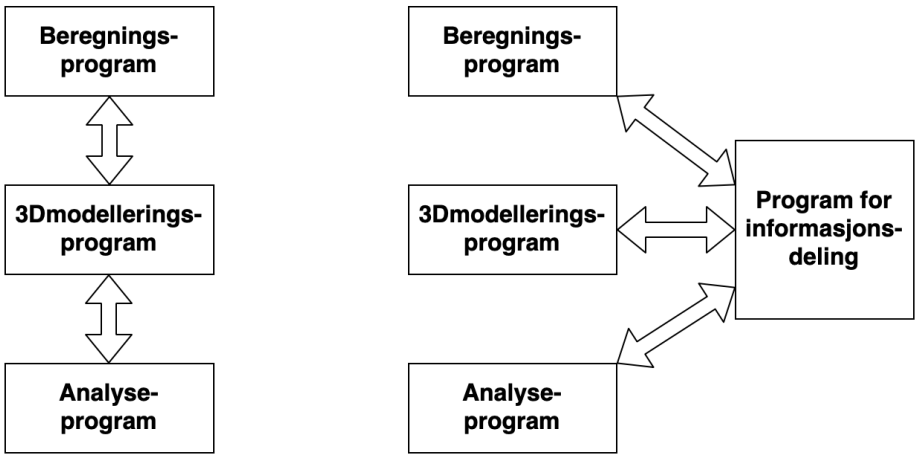
Ved å sette opp en modell som iterere mellom de ulike programmene for å finne de optimale parametrene skapes det en automatisert optimaliseringsprosess. Er denne satt opp korrekt skal en kunne trykke på start og etter litt tid få et ferdig resultat. For enkle modeller hvor det er satt opp gode begrensninger på parametre, vil dette kunne fungere godt. Problemet som kan oppstå er at programmene kommer fram til en løsning som vanskelig lar seg realisere i virkeligheten. Det kan være at den ikke lar seg produsere, er svært dyr å produsere, den kan være utfordrene å frakte eller at den ikke lar seg installeres.

Derfor kan det ofte lønne seg at det er en ingeniør som ser på resultatet mellom hver loop og setter inn nye parametere til neste gjennomføring. Et dataprogram vil eksempelvis kunne henge seg opp i en spenningskonsentrasjon og øke mengden gods for å ta opp dette, mens en ingeniør vil kunne gjøre vurderinger på om dette er noe som egentlig må tas hensyn til. Ved å ha en semi-automatisert prosess vil en kunne få det beste fra begge verdener. En ingeniør som gjør vurderinger en maskin ikke klarer, men en maskin som tar seg av de trivielle prosessene som ellers kan være svært tidkrevende.

2.5.2 Datadeling mellom programvarer

Uansett hvilket formål en ønsker å oppnå med sin digitale tvilling vil en ha behov for å dele informasjon mellom ulike typer programvarer. I noen programvarer er det allerede integrerte verktøy for å dele informasjon med enkelte andre programmer. Dette er ofte der hvor programmene har samme leverandør, eller hvor de har funnet en annen stor fordel med å integrere dette.

Dersom det er to eller flere programmer en ønsker å få til å snakke sammen, hvor det ikke eksisterer integrerte funksjoner, må en ha et eget program som står for informasjonsdelingen. Også her finnes det noen ferdige programmer på nettet, men ofte vil en måtte sette opp dette programmet selv. Den prinsipielle forskjellen mellom disse to metodene for å dele informasjon er vist i figur 2.15.



(a) Integrerte systemer for datadeling (b) Eksternt program for datadeling

Figur 2.15: Prinsipper for deling av data mellom ulike programvarer

For å sette opp et slikt informasjonsdelingsprogram må en se på hvert av de andre programmenes programmeringsgrensesnitt. Dette er kanskje bedre kjent som programmets API, *Aplication Programming Interface*. API kan beskrives som hvilke krav som stilles til andre programmer for å kunne kommumisere med et program [22]. Det er altså gjennom API en kan ta kontroll over et program sine innebygde elementer.

2.6 Inventor

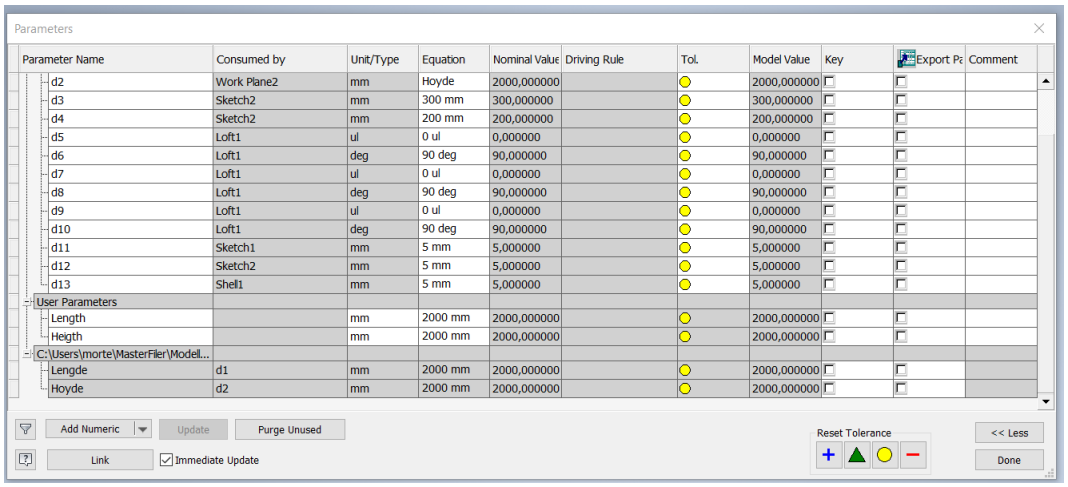
DAK er en forkortelse for Dataassistert konstruksjon [23]. De tradisjonelle tegningene på papir er flyttet inn et dataprogram som tillatter deg å tegne i 3D og med et 1:1 forhold. Ved å ha muligheten til å sette sammen alle 3D komponentene til en sammenstilling i et dataprogram, vil en enkelt kunne se hvordan enkeltdelene samvirker med hverandre [23]. Disse DAK programvarene oppdateres stadig og dette gir økte muligheter for å lage gode modeller. I denne oppgaven er det modelleringsprogrammet Inventor det fokuseres på.

Inventor er et profesjonelt verktøy for 3D mekanisk design. En av funksjonene som gjør Inventor spesielt attraktiv når en skal bygge paramatriserte modeller er at den støtter implementering av *Visual Basic* slik at en har mulighet til å automatisere de fleste prosesser [24]. *Visual Basic* er et programmeringsspråk utgitt av Microsoft [25].

2.6.1 Modelleringsfunskjoner

Det er noen innbygde funksjoner i Inventor som ytterligere tilrettelegger for parametrisering. En kan fint bygge parametriserte modeller uten bruk av disse, men ved å lære seg disse verktøyene vil en gjøre arbeidet lettere for seg selv på sikt.

I figur 2.16 vises parametervinduet i Inventor hvor en kan styre alle parametrene i en 3D-modell eller endre navn på parametrene. I dette verktøyet kan man lage avhengigheter mellom parametre og på denne måten redusere til noen få styrende parametere. Disse kan også styres med formler som legges inn i kolonnen merket *Equations*.



Figur 2.16: Parametervindu i Inventor

I tillegg til dette har Inventor en API som åpner for at en kan bruke de samme objektene og funksjonene en nytter til programmering, uavhengig av hvilket språk disse er skrevet i [26]. På denne måten kan en bruke både *AddIns* og selvstendige applikasjoner som utvider funksjonaliteten til Inventor. Ved å søke på nettet vil en finne flere slike programmer som ligger åpent.

iLogic

Dette verktøyet gjør det mulig å skape et regelbasert design i den hensikt å kunne gjenbruke arbeidet og automatisere denne prosessen [27]. Dette betyr at en lager korte kodesnutter for å si noe om hvordan en del skal settes sammen eller designes. Disse kodene er basert på *Visual Basic*.

Denne funksjonen kan blant annet nyttes til å lese eller skrive til et Excel dokument, automatisk oppdatere materiallisten i tegninger eller aktivere komponenter eller sammenstillings *features* ut fra gitte regler. I tillegg vil denne funksjonen tillate deg å kode slik at en får hentet ut de parametrene en trenger slik at det ikke blir sirkelavhengigheter i parametriseringen. Det er ennå flere funksjonaliteter og disse kan en lese mer om på Inventor sine hjemmesider.

iPart

Ved å bruke *iPart* har du mulighet til å lage komponentfamilier hvor en skiller på «medlemmene» ved å endre på størrelser, materiale eller andre variabler [28]. Etter at en komponent er designet settes alle variablene opp i en tabell. Deretter kan en kan legge til nytt medlem i familien og sette de ønskede variablene på denne.

Dette er en god metode for komponenter som av ulike årsaker er standardiserte og det ikke er en glidene endring i dimensjoner. Eksempler på dette kan være flenser og bolter hvor dimensjonene økes i trinn. Det samme vil være på veggtykkelse i rør. Det vil som regel være ønskelig å bruke

de standardiserte veggtykkelsene i stedet for å ende opp med å spesialbestille rør fordi modellen kun la på én millimeter for å få spenninger innenfor de tillatte verdiene.

iAssembly

Sammenstillingen fungerer etter samme prinsipp som *iPart*. Det dannes familier av sammenstillinger med ulike medlemmer. Medlemmene settes opp i liste med ulike designvariabler som antall komponenter eller størrelse. Her kan en hente ut komponenter fra en tabell i nettopp *iPart* [29]. Denne sammenstillingen kan styres med koder fra *Visual Basic*.

2.6.2 Optimaliseringsfunksjoner

I modelleringsprogrammet til Inventor er det en form for innebygget optimaliseringsfunskjon. Denne kalles *AutoLimits*. Her har en mulighet til å sette øvre og nedre grense på noen designvariabler, og det vil komme opp et varsel dersom disse grensene overskrides [30]. Denne vil i større grad virke som en hjelper til å huske på begrensinger i parametre dersom disse eksisterer, og kanskje ikke bidra i svært stor grad til optimalisering.

2.6.3 Datadelingsfunksjoner

Det finnes en integrert funksjon i Inventor som tillater import og eksport av parametre mellom Inventor og Excel [31]. I tillegg vil den åpne APIen til Inventor, som ble beskrevet tidligere, gjøre det mulig å skrive egne programmer for datadeling med andre programmer enn Excel [26].

Applikasjoner som kan dele data fra Inventor kan programmeres i blant annet: Visual Basic, C++, C#, Python og Java takket være Microsoft sin *Automation* teknologi[32]. Automation er utviklet for at programmer skal kunne manipulere objekter eksponert av andre applikasjoner[33]. Dette gjelder også for Excel og mange flere[34].

2.7 Beregningsprogrammer

Det finnes flere ulike typer programmer som kan nyttes til utføre beregninger. Det vil være ulike måter å sette opp regnestykkene ettersom hvilket program en nytter, men felles for de programmene tatt med i denne oppgaven er at formlene er dynamiske. Dette betyr at du trenger bare endre der variablene er definert, og så vil alle beregninger oppdatere seg selv. På den måten er det lettere å gjenbruke oppsettet til neste gang en liknende beregning skal utføres.

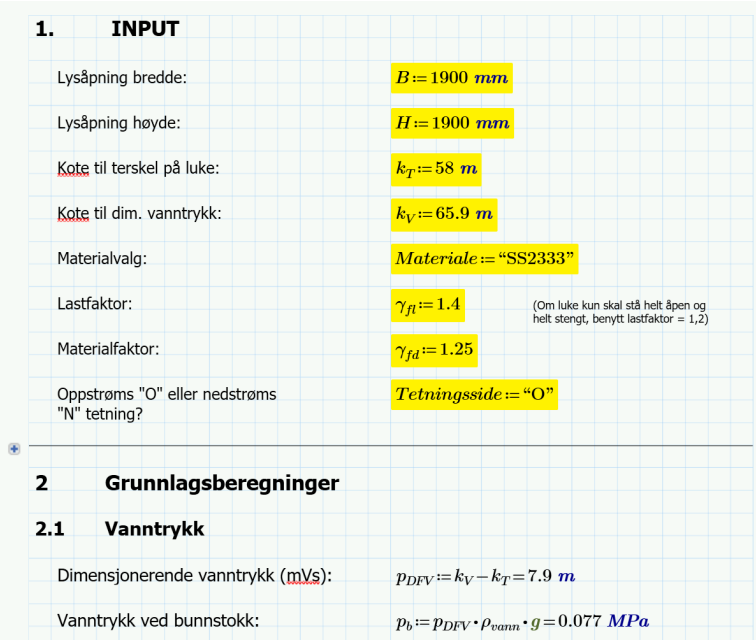
Det er denne mulighet til gjenbruk som gjør slike programmer godt egnet til å koble seg sammen med 3D-modellen. Ved å skape en link for datadeling mellom modellerings- og beregningsprogrammet vil en bare trenge å endre på parametrene ett sted. Denne linken vil også skape mulighet for automatisert optimalisering. Ved at data kan flyte fram og tilbake mellom beregninger og 3D-modell, vil den kunne iterere seg fram til den best mulige løsningen.

I tillegg til å sette inn formler for de ulike beregningene som skal utføres, er det viktig å definere en variasjonsbredde på parameterene. Ved å sette en maksimum- og en minimumsverdi vil en i større grad unngå å få noen urealistiske resultater. Eksempelvis vil det ikke være akseptabelt med 1 mm tykke vegger på et vannrør selv om de etter styrkeberegninger skulle tåle påkjenningene.

Videre følger en beskrivelse av de beregningsprogrammene som det skal sees nærmere på i denne rapporten. De punktene som dras fram er hvordan de egner seg til å koble seg sammen med andre programmer, hvilke optimaliseringsfunskjoner som er innebygget og i hvor stor grad de vil oppleves som brukervennlig for en eventuell ingeniør som skal overta en ferdig modell.

2.7.1 Mathcad

Dette programmet har et oppsett hvor du skriver inn formler på samme måte som en ville gjort dersom en regnet for hånd. Her har en i tillegg mulighet til å sette opp hver variabel og definere dem bestemte steder i dokumentet. Dette er vist i figur 2.17. Altså så lenge et regneark er satt opp på en ryddig måte, vil den være svært intuitivt å overta det for en annen ingeniør.



Figur 2.17: Eksempel på et regneark i Mathcad

En har muligheten til å bruke regnearket som en notatblokk for å dokumentere egne beregninger, men også for å vise modellens *Design Intent*. Det krever ingen forkunnskaper om programmet for å sette seg inn i en annen sitt regneark [35], men det krever kjennskap til programvaren for å kunne utnytte alle mulighetene som Mathcad byr på.

Optimaliseringsfunksjoner

Mathcad byr på en innebygget funksjon kalt *Solve Blocks*. Denne nyttes til å løse lineære, ikke-lineære og differentielle ligningssytemer. Funksjonen kan også brukes til optimalisering ved at den finner topp- eller bunnpunkt for en gitt funksjon med gitte begrensninger [36]. Dette løser den ved å ta utgangspunktet i *guess values* for deretter å iterer seg fram til en løsning.

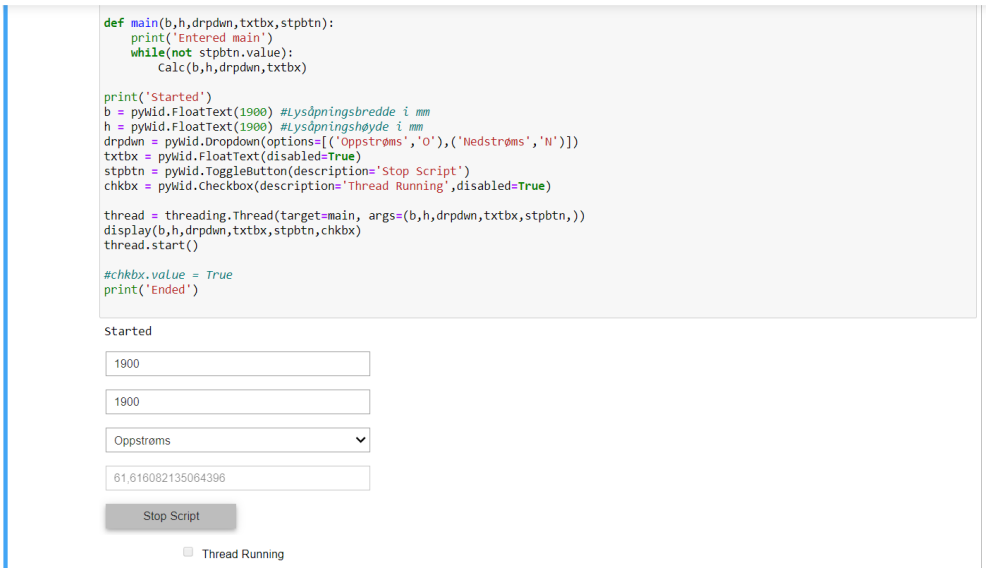
Det er også muligheter for enkel programmering i regnearket [37]. På denne måten kan en definere variasjonsbredden på de parametrene hvor det måtte være behov. Det er også muligheter til programmere slik at resultatet havner innen gitte verdier eller på bestemte intervaller. Eksempelvis at den alltid rundes opp til nærmeste hele tall.

Datadelingsfunksjoner

Dersom en bygger sin 3D-modell i Creo finnes det integrerte funksjoner i disse to programmene for at de kan dele data til hverande [38]. Det finnes også en integrert funksjon for datadeling til Excel [39]. Videre er Mathcad sin API åpen for at en kan skrive egne programmer for datadeling med andre programvarer [40].

2.7.2 Jupyter notebook

Dette er et programmeringsbasert beregningsverktøy som hvem som helst kan laste ned gratis fra nettet [41]. At det er programmerbart betyr det at en kan endre utseende og layout ettersom en ønsker. Det kan benyttes Python i dette verktøyet, men det støtter også andre programmeringspråk. En fordel med Python som språk er at det finnes masse eksempler på internett og flere universitet har dette som en del av grunnutdanningen til ingeniører [42]. Noe av grunnen til dette er at det er et veldig anvendelig og et språk som er enkelt å lære [43].



Figur 2.18: Eksempel på regneark laget med Jupyter Widgets

I eksemelet i figur 2.18 er det brukt Jupyter Widgets for å lage et brukergrensesnitt. Dette er python objekter med flere bruksområder [44], hvor det i dette tilfellet er utnyttet til å lage tekstbokser en kan skirve i og *drop-down* menyer.

Man kan også velge om en ønsker å laste ned programvaren til sin PC eller om det skal jobbes online via en server på nettet [41].

Optimaliseringsfunksjoner

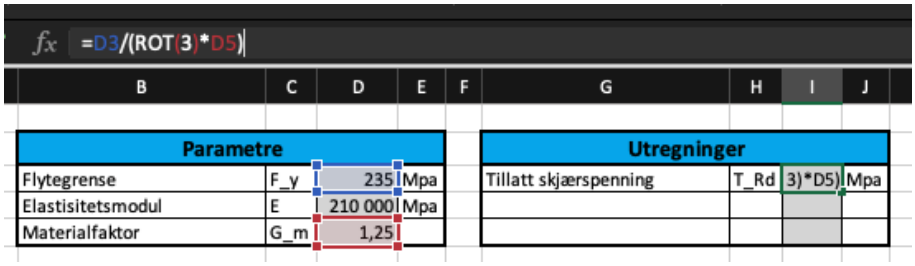
Styrken til python ligger i alle de utallige tilleggspakkene som har blitt laget hvor en av disse heter *scipy.optimize*. Denne inneholder ferdige metoder for å optimalisere funksjoner. Blant disse finnes: minimering, *least-squares* og funksjoner for å finne røtter til en funksjon [45].

Datadelingsfunksjoner

Python har en *win32com* som betyr at en kan nytte denne til å styre alle programmer som har en COM API [46]. Dette betyr eksempelvis at denne kan kobles sammen med både Inventor og Excel.

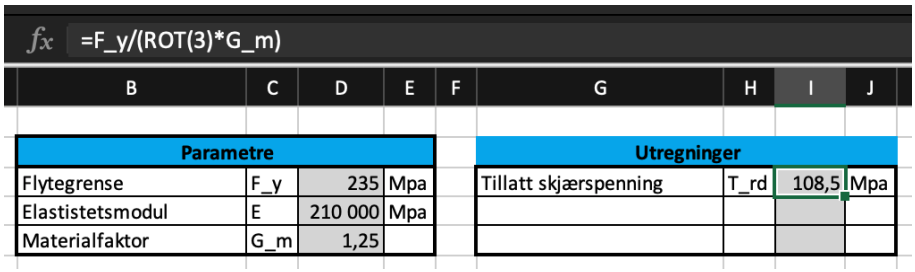
2.7.3 Excel

De aller fleste har et eller annet forhold til Excel fra tidligere og vil derfor ha lav terskel for å starte å jobbe med dette programmet. Excel byr også på svært mange innebyggede funksjoner som gjør det lettvindt å jobbe med mye data [47]. En kan legge inn formler og visualisere data i grafer. I tillegg kan en kode celler slik at det blir lett å se dersom noen utregninger ikke kommer innenfor sine tillatte verdier. Det er muligheter for å lage *drop down* lister slik at en kun kan velge mellom bestemte verdier. Ved å opparbeide seg kompetanse på Excel vil en kunne lage svært gode regneark.



Figur 2.19: Eksempel på oppsett av formel i Excel

I figur 2.19 er det vist hvordan en formel vanligvis settes opp. Den store ulempen med dette er at det er fort gjort å sette inn feil celle som referanse når en lager en formel, og det kan være knotete for en ingeniør å kontrollere om et regneark er satt opp riktig. Formelen er definert i den cellen som er grønn. For å gjøre det lettere å kontrollere er hver av cellene benyttet i formelen vil de tildeles hver sin fargekode. På det lille arket i figuren er det relativt lett å finne igjen cellene, men dette blir stadig mer krevende ettersom hvor stort regnearket blir.



Figur 2.20: Eksempel på bruk av *Excel name range*

En ganske ukjent, men også et svært nyttig verktøy i Excel er *Excel name range* [48]. Denne gir deg mulighet til å navngi celler, områder, konstanter eller formler hvor du siden kan kalle opp disse kun ved hjelp av navnet. I figur 2.20 kan vi se at dette bidrar til å gjøre det lettere å kontrollere regneraket. Derimot er formlene er fortsatt skrevet på en mindre intuitiv enn andre beregningsprgrammer, siden de i Excel listes bortover og ikke slik som de ville vært skrevet for hånd.

Optimaliseringsfunksjoner

Excel har en innebygget funksjon kalt *Solver*. Denne skal finne den optimale maksimum- eller minimumsverdien for en formel i en gitt celle [49]. Dette løser den ved at man definere begrensninger i bestemt celle og så hvilken celle som er variabler. Deretter sier en hvilken celle *Solver* skal minimere eller maksimere ved å endre på variablene.

Datadelingsfunksjoner

Gjennom en Microsoft teknologi kalt Automation kan man styre mer eller mindre alle sider ved Excel. Man lager da egne applikasjoner eller *add-ins* ved å kode i C#. Det finnes mange eksempler og alt er dokumentert på Microsoft sine egne sider [34].

Tidligere i rapporten er det vist at det er flere programmer med integrerte funksjoner for å dele data med Excel. Det kreves lite forkunnskaper for å kunne bruke disse integrerte funksjonene og de vil derfor være et godt alternativ for de med begrensende ferdigheter innen koding.

At Excel kan dele data med flere programvarer byr også på muligheten til å bruke Excel som en plattform til datadeling mellom andre programvarer. Eksempelvis ha det som en mellomstasjon for å dele data fra Mathcad til Inventor.

2.7.4 Sammenligning av beregningsprogrammer

I tabell 2.7 er beregningsprogrammene sammenlignet mot hverandre. Her er det viktig å merke seg at de blant annet er sammenlignet opp mot hverandre på grunnlag av hvordan de er å koble opp mot Inventor. Nyttes det et annet modelleringsprogram vil resultatet kunne se noe annerledes ut.

Tabell 2.7: Sammenligning av beregningsprogrammer

Prgramvare	**Styrker**	**Svakheter**
Mathcad	Polert Minimalt med opplæring Oversiktlig	Lisensert Krever et tredjeprogram for å dele data med Inventor
Jupyter notebook	Gratis Kan jobbe online Uendelig med muligheter	Krever kjennskap til python e.l.
Excel	Lett å visualisere data og gjøre regnestykker med dem Håndterer store mengder data	Lite oversiktlige formler Vanskelig å sette seg inn i andres ark

2.8 Analyseprogrammer

Med analyseprogrammer menes det programmer med funksjonalitet innen styrkeberegninger (FEM-analyse). Dette er programmer som skal hjelpe ingeniøren i å gjøre de riktige beslutningene med tanke på størrelser og former på komponenter. For dynamiske modeller er det også viktig å kunne gjennomføre en simulering for å kontrollere at alle komponenter kan bevege seg slik som tenkt. I tillegg til styrkeberegning og dynamisk analyse kan ofte slike programmer håndtere strømningssimuleringer, termisk analyse, frekvensanalyse og de fleste typer påkjenninger en konstruksjon kan utsettes for.

For å finne et program som er egnet til å nytte i en digital tvilling er det viktig med et program det er mulig å kode til å dele informasjon med andre programmer. I tillegg må de kunne utføre analyser for de påkjenningene en konstruksjon utsettes for.

2.8.1 Inventor

I tillegg til å være et modelleringsprogram kan Inventor analysere komplette mekaniske systemer. Dette vil da være både FEM-analyser og dynamisk simulering. Programmet kan også kjøre analyser for sammenstillinger og sveisede deler [50].

Resultatet presenteres i såkalte plot hvor det tydelig markeres hvor det er størst og minst påkjenninger. En kan selv bestemme hvilke typer simuleringer som skal gjennomføres og får opp ulike plot for hvert tilfelle.

Inventor har også et tilleggsprogram kalt *Inventor Nastran*. Dette programmet er spesiallaget for å utføre avanserte FEM-analyser og har innebygde funksjoner for modelloptimalisering. Denne tilleggspakken vil være nødvendig for å kunne gjennomføre eksempelvis utmatting- og termiske analyser [51].

Optimaliseringsfunksjoner

Inventor byr på et verktøy kalt *Shape Generator*. Her starter du med et overordnet design for deretter å definere lastene det utsettes for. Deretter vil *Shape Generator* kutte bort alt overflødig materiale i den hensikt å maksimere komponentens stivhet [52]. Etter dette er gjennomført vil det trolig være nødvendig å justere noe på modellen, men dette kan være et nyttig verktøy for å analysere designet.

Analyseprogrammet Nastran har en funksjon hvor en kan optimalisere basert på topologi. Enkelt forklart betyr dette å optimalisere materialbruken. En starter helt uten et design og deretter defineres laster og eventuelle parametriske begrensninger før programvaren generer et optimalt design. [53]. Dette er et verktøy som er godt egnet for å genere ideer om en står fast i designet, men det er ingen garanti for at dette «optimale designet» vil kunne produseres. En risikerer også at det blir svært dyrt å produsere.

Datadelingsfunksjoner

Analysemiljøet til Inventor er ikke eksponert i APIen som betyr at det ikke kan styres av et eksternt program eller bli en del av en automatisert loop [54].

2.8.2 SolidWorks Simulation

Simuleringsdelen til SolidWorks er delt inn i tre pakker slik at en kan selv velge hvor avanserte pakker en har behov for å kjøpe. Med den mest avanserte pakken skal en kunne gjennomføre alle typer analyser på sin modell [55]. Disse tre pakkene er Standard, Professional og Premium.

Utenom dette vil SolidWorks Simulation i stor grad fungere likt som Inventor sitt analyseprogram ved at resultatene presenteres som plot. De vil ha ulikt brukergrensesnitt, men hvilket som er foretrukket vil trolig avhenge mest av personlige preferanser.

SolidWorks har også laget et tilleggsprogram kalt DriveWorks som er spesialisert for designautomatisering [56]. Den har som mål å redusere repetitive oppgaver, fjerne feil og korte ned tiden det tar å utvikle et produkt. Programvaren beskrives ikke i detalj her da den ikke er vurdert til å være generell teori, men nevnes likevel som et alternativ som leseren eventuelt kan utforske videre om ønskelig.

Optimaliseringsfunksjoner

I pakken *Professional* vil en få tilgang til et verktøy kalt *Topology Studies*. Dette er et optimaliseringsverktøy som hjelper brukeren til å se hvordan en kan minimere bruken av materiale under lineær elastisk statisk last, men hvor komponenten fortsatt møter kravene til stivhet og kan ta opp de spenningene den utsettes for [57].

Studien settes i stor grad opp på samme måte som for Inventor. En starter med å definere den maksimale størrelsen komponenten kan ha, for deretter å definere laster, fastholdninger og begrensinger med hensyn på produksjonsmetoder. Deretter vil vektøyet søke etter den geometrien som minimerer bruken av materialet, men fortsatt møter alle krav som stilles til den.

Datadelingsfunksjoner

Solidworks har en API som dekker alle aspekter ved programmet inkludert *Simulation*[58]. Dette betyr at endring av modellparametre, starte simuleringer og hente resultater fra simuleringer kan styres og automatiseres av eksterne applikasjoner. De eksterne applikasjonene kan enten kjøpes eller lages selv.

2.8.3 ANSYS

ANSYS tilbyr et program for alle analyser som er vanlig å gjennomføre [59]. Denne kan løse komplekse ingeniørproblemstillinger og bidra til bedre og raskere designbeslutninger. Det er innebygde muligheter for automatisering av designprosessen som kan parametriseres for å analysere flere ulike design alternativer. ANSYS skal passe ditt bruk uansett erfaringsnivå eller hvor komplekse problemstillinger en måtte ha.

Også ANSYS har utviklet et program med designautomtisering som formål, optiSlang [60]. De har lagt fokus på et program hvor du kan automatisere simulering- og optimaliseringsprosesser. Denne programvaren vil heller ikke beskrives nærmere, kun nevnes som et alternativ for leseren.

Optimaliseringsfunksjoner

Det er et innebygget verktøy kalt *Optimization*. Dette er en funksjon som gir modellen din flere ulike designparametre, ulike laster og endringer i miljøpåvirkninger for å finne de optimale parametrene. Dette vil være en mye hurtigere løsning enn å endre på selve DAK-modellen [59].

I likhet med de andre analyseprogrammene har ANSYS også et verktøy for *Topology Optimization*. Deres funksjon fungerer i stor grad likt som de to overnevnte.

Datadelingsfunksjoner

ANSYS Workbench har en egen komponent for å importere og eksportere data fra Excel[61]. Denne komponenten lagrer en kopi av det linkede arket i prosjektfilene for full kontroll.

Ansys har en egen konfigurator slik at man kan importere 3D-modeller direkte gjennom CAD configuration manager [62]. ANSYS støtter å åpne Inventor/Solidworks sine filformater. Dette verktøyet er ikke tilgjengelig med studentlisens.

2.8.4 Sammenligning av analyseprogrammer

I tabell 2.8 er de ulike analyseprogrammene målt opp mot hverandre. De viktigste styrkene og svakehetene er plukket ut. Også her er det viktig å merke seg at i denne sammenligningen er det sett på hvordan de er å koble sammen med Inventor som modelleringsprogram, og at denne sammenligningen ville sett noe annerledes ut om det var et annet modelleringsprogram som ble nyttet.

Tabell 2.8: Sammenligning av analyseprogrammer

Prgramvare	**Styrker**	**Svakheter**
Inventor	Integrert i modelleringsprogram	Færre analysemuligheter
SolidWorks	Delt opp i pakker etter behov Mange analysemuligheter	Ekstra ledd å overføre modell til
ANSYS	Mange analysemuligheter Flere optimaliseringsfunksjoner	Ekstra ledd å overføre modell til

Det er også verdt å nevne at både SolidWorks og ANSYS har egne modelleringsprogrammer som på samme måte som Inventor er integrert med sitt analysemiljø. Siden rapporten kun tar for seg Inventor som modelleringsprogram er disse kun vurdert som analyseprogrammer.

3. Ståstedsanalyse

I dette kapittelet beskrives det hva som er vanlig praksis i Norge med hensyn på å nytte ulike strategier i 3D-modellering. Det er sett på hvordan dette implementeres i bedrifter, hvilken masteroppgaver som er gjort og hvilke kurs som tilbys innenfor dette. Internasjonal forsking på dette området er redegjort for i kapittel 2.

3.1 Tidligere masteroppgaver

Det er gjennomført søk i Brage for å finne tidligere masteroppgaver innen dette temaet. Alle de engelske søkeordene i tabell 4.1 ble nyttet i dette søket i Brage ved NTNU og NMBU. I tillegg ble det gjort noen mindre søk i Brage ved UiS, UiT, USN og UiB. Det ble gjennomført søkeord på engelsk siden alle sammendrag uansett er oversatt til engelsk og det er forventet at nøkkelordene vil være benyttet der.

Det lykkes ikke å finne noen relevante hovedoppgaver.

3.2 Kurs om strategi 3D-modellering

Det lykkes ikke finne noen kurs som lå ute på nettsider til norske leverandører av ulike modelleringsprogramvarer. Det ble også tatt kontakt per telefon til Symetric[63], PLM Group [64] og EDR Medeso [65]. Der bekrefte de at slike kurs ikke tilbys per dags dato.

Det har heller ikke lykkes å finne emner ved NTNU eller NMBU som underviser i strategier ved oppbygging av 3D-modeller [66][67].

3.3 Patentregisteret

Det er kun søkt i det norske patentregisteret, NIPO, men det lykkes ikke med å finne noen søknader innen strategier for oppbygging av 3D modeller. Alle de norske søkeordene i tabell 4.1 ble nyttet. I dette registeret legges det ut straks det søkes om en patent, så en eventuell søknad ville dukket opp selv om den ikke er blitt godkjent [68].

Det antas at det hittil ikke er søkt om patenter i Norge på strategi(er) for parametrisering av 3D-modeller.

3.4 Oppsummert

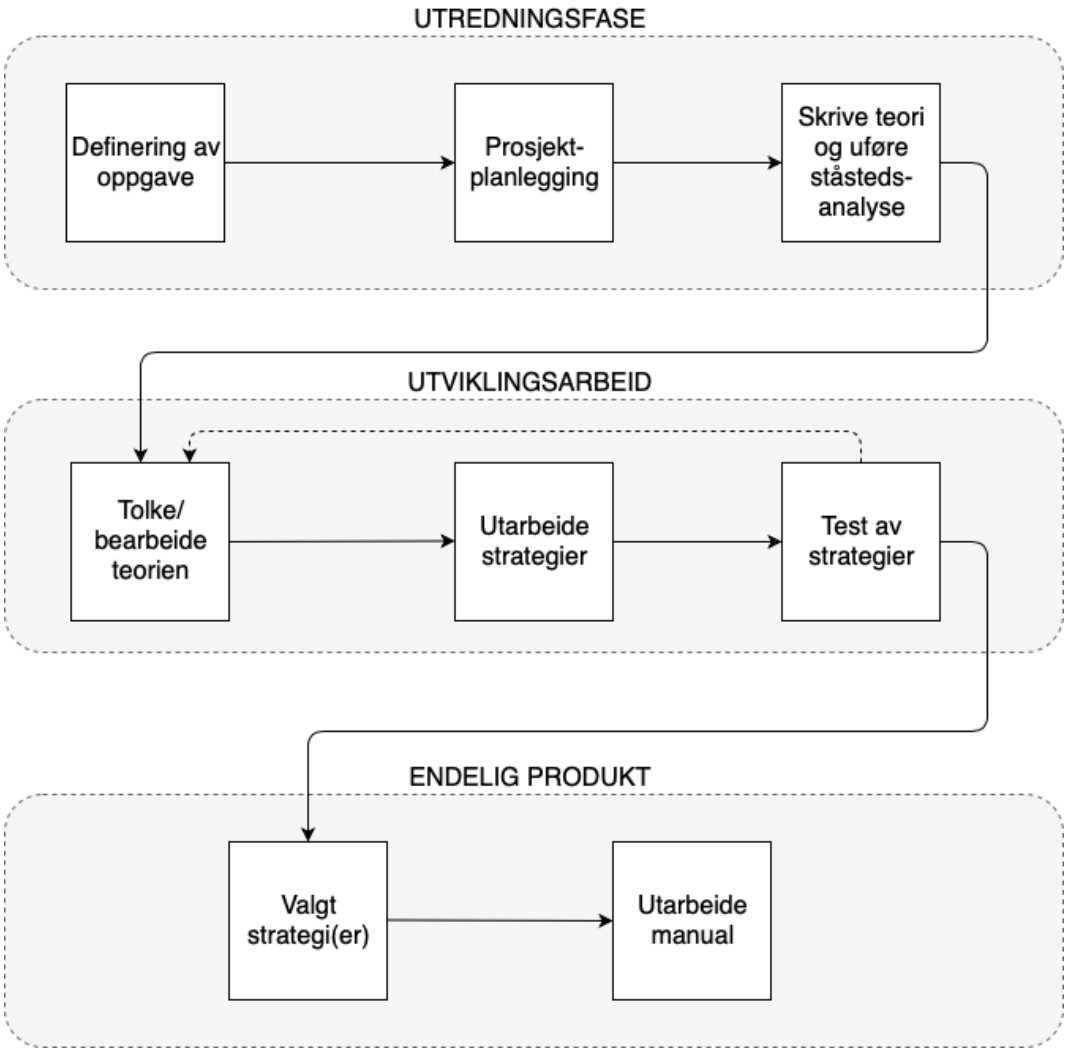
Ut fra undersøkelsen som er gjennomført kan det virke som om hver person, eller hver bedrift, finner sin egen framgangsmåte i modellering ved hjelp av egne erfaringer. Det har ikke lyktes med å finne noen organiserte kurs eller studier som er gjennomført på dette i norsk sammenheng. Det er selvsagt en mulighet for at bedrifter velger å skjule denne informasjonen for å oppnå markedsfordeler.

4. Metode

I dette kapittelet er det beskrevet framgangsmåten benyttet i forbindelse med dette prosjektet. Hensikten er å bygge opp troverdigheten rundt rapporten ved å forklare hvordan konklusjonen er nådd, samt gjøre det lettere å gjenskape de forsøkene som er gjort.

4.1 Prosesstrinn

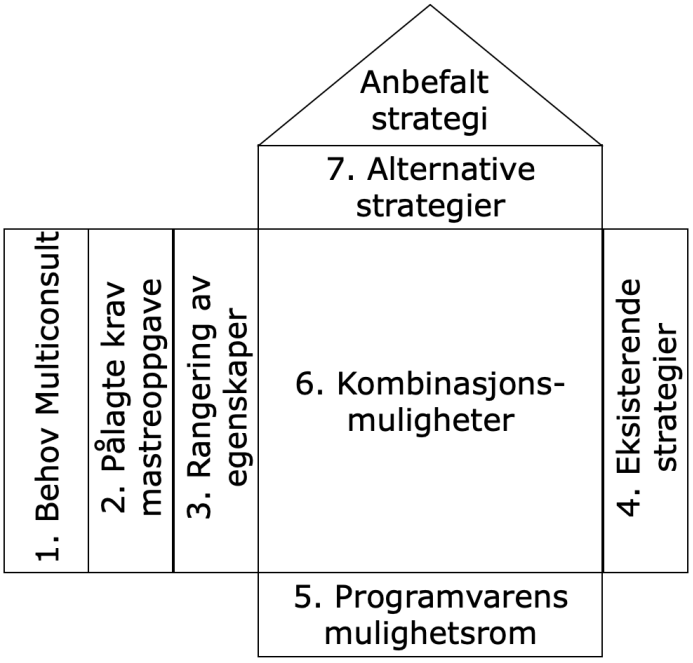
I starten av prosjektet ble det samlet inn relevant informasjon både for å skape en større forståelse for problemstillingen, men også for å ha et utgangspunkt å jobbe ut fra. Deretter ble det gjort forsøk på å bygge ulike type modeller med ulike strategier. Disse modellene ble testet opp mot evnen til å holde mål ut fra robusthet, brukervennlighet og gjenbrukbarhet. Etter noen runder med testing ble teorien bearbeidet på nytt for deretter å gjennomført nye runder med tester. Denne metoden ble brukt til å jobbe seg fram til en kunne konkludere med en god løsning. Disse trinnene er illustrert i figur 4.1



Figur 4.1: Prosessdiagram for prosjektarbeidet

4.2 Kvalitetsikring

Metoden som ble benyttet for å sikre kvaliteten på rapporten har tatt inspirasjon fra *House og Quality*, HOQ [69]. Målet var å skape en strukturerte måter å implementere ønsker, behov og krav i produktutviklingen. Hvordan denne ble benyttet i dette prosjektet er vist i figur 4.2.



Figur 4.2: Kvalitetsikring av rapporten ved bruk av HOQ

For å sikre at rapporten holder sin relevans ut fra Multiconsult sin problemstilling var det et mål å oppretthold en god kommunikasjon gjennom hele prosjektperioden. Veileder ved NMBU bidro med tilbakemeldinger på det faglige innholdet, samt at rapporten holder det akademiske nivået som er påkrevd for bestått masteroppgave. I praksis ble dette gjennomført med ukentlige veiledningsmøter hvor rapporten sendes til gjennomlesing god tid i forveien.

For å sikre at innholdet i rapporten ble forståelig for flere enn de som er involvert i prosjektet, ble det brukt personer med maskinteknisk ingeniørbakgrunn til å lese gjennom rapporten. Deres tilbakemeldinger var svært nyttige for å kontrollere at innholdet i rapporten er beskrevet på god måte.

4.3 Innsamling av data

I rapportens teoridel er det beskrevet det som allerede er kjent gjennom tilgjengelig litteratur. Her var det naturlig å bruke bøker, rapporter, artikler og lignende som kilder. I tillegg ble websøk brukt for å verifisere informasjon eller for å skape en bedre forståelse av det som ble beskrevet i vitenskapelig litteratur.

Intervju ble i all hovedsak benyttet for å heve forståelsen og er ikke direkte gjengitt i rapporten.

Tabell 4.1: Søkeord benyttet i litteratursøk

Norske ord	**Engelske ord**
Parametrisk modellering	Top down design CAD
Parametrisk design	Parametric CAD
Modelleringsmetoder	Design automation
Digital tvilling	
Dataassistert konstruksjon	
3D modellering	

4.4 Databehandling

Informasjon som er hentet inn fra utenfor prosjektgruppen er samlet i kapittel 2 og 3. Det ble brukt som et grunnlag for å utarbeide hensiktsmessige metoder for hvordan bygge opp parametriserte modeller. Denne informasjonen var altså ikke direkte førende for resultatet, men er brukt som et utgangspunkt og en pekepinne gjennom arbeidet.

For å verifisere validitet til informasjonen som ble hentet ut av rapporter og artikler ble primærkildene undersøkt. Alt som er gjengitt fra fagbøker har blitt ansett som troverdig da det ikke er satt av ressurser til å etterprøve dette.

5. Resultater

I dette kapittelet presenteres de observasjoner som er blitt gjort under testingen av de ulike strategiene og programvarene. Rekkefølgen på dette har samme struktur som teorikapittelet slik at det skal være enklere å orientere seg i rapporten.

Hvert enkelt forsøk som er gjort er beskrevet i korte rapporter i vedlegg C. Der det er naturlig å sammenligne strategier eller programvarer er resultatene satt opp i tabeller for å forenkle sammenligningsprosessen. De områdene som er vurdert til å være relevante for dette er satt opp som punkter i venstre kolonne, hvor det så er skrevet kort om hvert av strategiene eller programvarene i sine respektive kolonner. Der hvor det er tomme ruter har det ikke blitt oppdaget noe konkret om gitt strategi eller programvare, og leseren kan da lese teorien i kapittel 2.

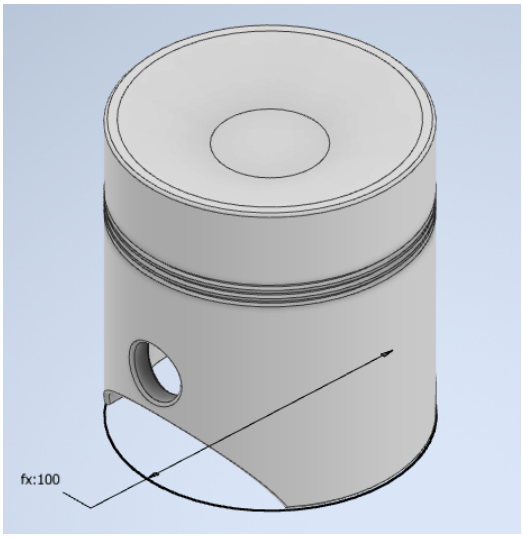
5.1 Oppbygging av komponent

I dette delkapittelet presenteres de resultatene som er gjort i forbindelse med testing av de ulike strategiene for å modellere en komponent. Resultatene er oppsummert i tabell 5.1. Innholdet i tabellen baserer seg på forsøkene beskrevet i vedlegg C.1, C.2 og C.3. Generelle resultater er beskrevet som tekst nederst i dette delkapittelet.

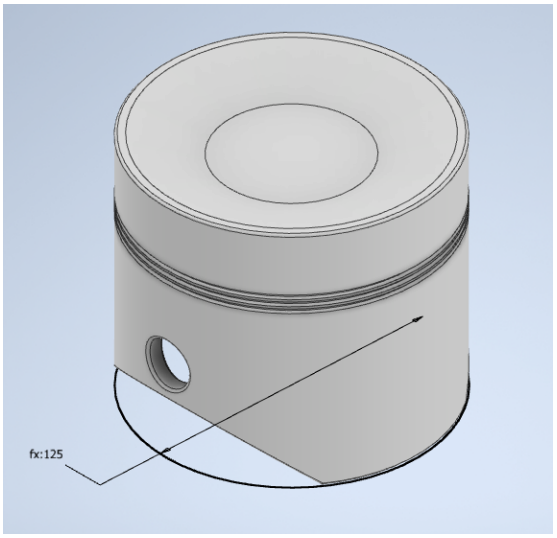
Tabell 5.1: Resultater oppbygging av komponent

	Delphis' Horizontal Modeling	**Explicit Reference Modeling**	**Resilient Modeling**
Tilgjengelig informasjon om metoden	Hele metoden er beskrevet i patenten.	Noe begrenset, men nok til å kunne sette seg inn i strategien.	Det er svært mye tilgjengelig informasjon om dette på nettet. RMS har en egen hjemmeside hvor alt beskrives.
Hvor enkelt er det å forstå metoden	Teorien oppleves som enkelt å forstå, derimot oppleves det som krevende å utføre dette i praksis.	Enkel å forstå, relativt enkel å ta i bruk.	Relativt komplekst da denne metoden er omfattende.
Endring av parametre	Enkelt for den som har laget modellen, men kan være lite intuitivt for en som overtar.	Dette vil i stor grad avhenge av hvordan den som lager modellen legger opp til dette.	Intuitivt både for den som har laget og en som overtar en modell. Dette skyldes i all hovedsak navngivingen av alle *features*.
Annet	Oppleves som unødvendig vanskelig at en ikke kan relatere *features* til hverandre.	Oppleves som en *ligth* versjon av *Horizontal*.	Pga. avhengigheter mellom *features* ga denne metoden 60% reduksjon i antall parametre som måtte endres for å oppdatere hele modellen.

Forsøkene viser at alle strategiene gir robuste komponenter. I figur 5.1 kan vi se at alle *features* på stempelhodet oppdateres slikt det skal da diameteren endres. Dette er stempelhodet som ble brukt i alle forsøk. Akkurat denne modellen er bygget etter RMS, men alle strategier ga til slutt like robuste modeller.



(a) Stempelhode med diameter 100mm



(b) Stempelhode med diameter 125mm

Figur 5.1: Parametrisert komponent

En ser tydelig gjennom forsøkene at det kreves relativt høy kompleksitet før en kan skille de ulike metoden fra hverande på hvor robust modellen blir. De modellen som er nyttet i disse forsøkene er trolig ikke komplekse nok. Uansett vil valg av metode skille på hvor intuitiv modellen er å overta for en annen ingeniør.

Typiske *features* som skapte problemer for endring av modellene er når en skal gjøre drastiske endringer på modellens utseende, slikt som å flytte på hull eller lage større utvekster. I tilfeller hvor en bare skal skalere på ulike parametre ser det ikke ut til å skape noen problemer på komponentnivå uansett framgangsmåte. Det er ikke valgt å skrive noe om robustheten til de ulike strategiene da det i tilfelle burde vært gjort flere tester og at en trolig burde hatt en mer komleks modell for dette.

5.2 Organisk utformet geometri

Fokuset har vært å teste metoder som er generelle og kan nyttes i alle modelleringsprogrammer. Det skal også nevnes at *Multi-body parts* er et alternativ for organisk utformet geometri, men i denne rapporten er det valgt å sette den med gjennomgang av sammenstillinger. Slik det er tolket her vil den kun være hensiktsmessig for organiske former dersom den er bygget opp av flere komponenter, og derfor satt sammen med sammenstillinger.

Videre har det blitt sett på et program kalt CAESES som er beskrevet i vedlegg B.2. Dersom en skal jobbe svært mye med organiske former vil dette programmet trolig lette mye på arbeidsmengden. Derfor er det valgt å nevne dette i rapporten, men at det vektlegges ikke da denne programvaren er antatt å ikke være tilgjengelig for ingeniører flest. Opplæringsdokumentene til dette programmet var lett å følge, men det vil nok kreve en del tid for å mestre dette fullt ut. Grunnen er at modelleringsteknikkene i dette programmet er satt opp på en helt annen måte enn i Inventor og SoldiWorks. Testen av programmet beskrives i sin helhet i vedlegg C.6.

For de to generelle teknikkene oppdager en fort at det er svært mange tilgjengelige ressurser på nettet. Om en søker på Youtube finner en nesten alltid en video av en som har bygget en tilsvarende modell som en selv skal. Resultatene fra testene som er gjort er oppsummert i de to påfølgende avsnittene.

5.2.1 Loft

I utgangspunktet er dette en svært enkelt metode, men det ble raskt tydelig at dette er tidkrevende arbeid om en ønsker fin oppløsning på modellen. Det er nødvendig med svært mange skisser med avhengigheter mellom seg og det kan bli nødvendig med noen hjelpelinjer om Inventor ikke gir tilfredstillende resultater automatisk. Det krever trolig et visst erfaringsnivå for å bygge gode modeller da en hele tiden må se for seg hvor komponenten skal ende og vite hvilke veier som fører dit.

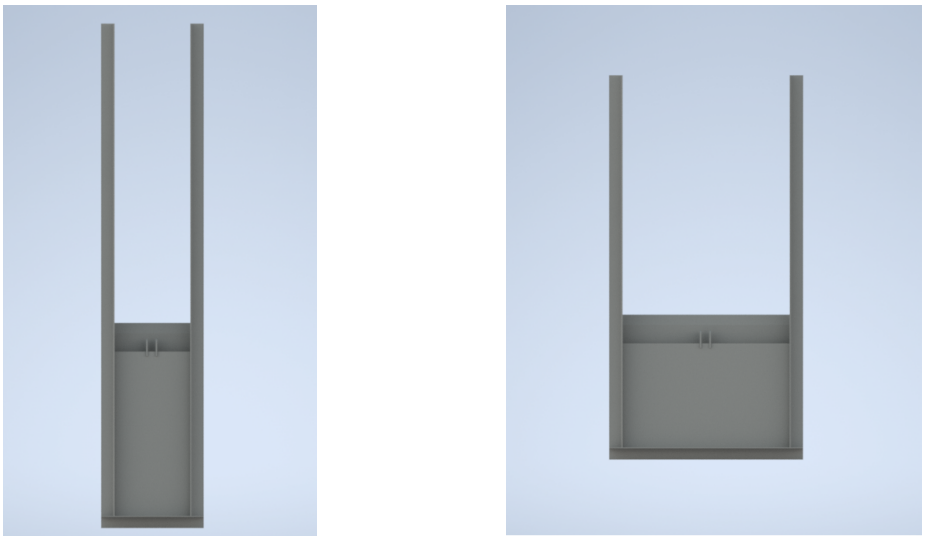
Denne metoden egner seg godt for kompliserte organiske former hvor en hele tiden må ha kontroll på tversnittet, men kan også brukes til noe så enkelt som vannveier som bare skal endre tversnitt. Test av denne metoden er lagt til vedlegg C.4.

5.2.2 Sweep

Dette er den enkleste metoden for organisk utformet geometri og fungerer godt så sant ikke modellen endrer tverrsnitt brått. Hele testen av denne metoden er beskrevet i vedlegg C.5. En kan bruke matematiske formler for å styre retning på *path* som tverrsnittet sin skisse følger. Man kan også endre tverrsnittet underveis, men da vil den kun gjøre dette gradvis og det blir ikke like god kontroll som ved å sette opp mange skisser og benytter *loft*.

5.3 Oppbygging av sammenstilling

I kapittel 2 er klart definert i hvilken av de tre overordnede strategiene for sammenstillinger en en skal velge i de ulike tilfellene og derfor ble ansett som unaturlig å sammenligne *Top-Down*, *Bottom-Up* og *Middle-Out*. Derimot er det beskrevet ulike metoder for *Top-Down* som er sammenlignet i det første avsnittet før det er beskrevet hvilken erfaringer det er gjort med de resterende i asvnittene som følger.



(a) 1000x2000mm (BxH) (b) 2000mmx1200mm (BxH)

Figur 5.2: Parametrisert glideluke

I figur 5.2 er det vist den glideluken som er bygget under alle tester av strategier for sammenstillinger. Alle modeller ble til slutt robuste og tålte endring av de styrende parametre, uavhengig av valgt strategier. Derimot varierte det mye hvor robust modellen ble i forste forsøk innen de ulike strategiene.

5.3.1 Top-Down Design

Resultatene fra de ulike strategiene for TDD er sammenlignet i tabell 5.2 og disse baserer seg på testene beskrevet i vedlegg C.7, C.8 og C.9. Konseptet om TDD kan oppleves som en krevende metode i begynnelsen og det er tydelig at hver modell blir bedre og bedre for hver gang det lages.

Videre er det svært enkelt å endre på geometrien ved å kun gjøre endringer i en skisse. Siden det ikke er noen fastholdning vil ikke modellen krasje. Modelleringen går relativt raskt da man hele tiden jobber i samme fil.

Tabell 5.2: Resultater test av ulike strategier for TTD

	Single Skeleton	**Multi Skeleton**	**Multi-Body part**
Tilgjengelig informasjon om metoden	Lett tilgjengelig	Kun funnet den ene artikkelen som er referert til i teorien.	Lett tilgjengelig
Hvor enkelt er det å forstå metoden	Prinsippet er lett forståelig, men krever erfaring for å bygge gode modeller.	Krevende.	For den som foretrekker *Bottom-Up* kan denne framstå som mer intuitiv enn de andre.
Strategi komponenter	Irrelevant	Irrelevant	Kan påvirke stabiliteten.
Endringer i modellen	Kan fort bli rotete morskjelett. Er det for mange parametre som er viktige kan det være vanskelig å orientere i dette etterpå. Spesielt om de bygges etter *Horizontal* og en ikke definere nok hjelpegeometri.		Tilsynelatende ikke mulig å endre materialet etter en komponent er laget.
Egnet bruksområde		Svært store sammenstillinger.	Sammenstillinger med kurver eller linjer som går over flere komponenter, men var også en bra metode for eksempelvis lukebladet. Kan bare ikke ha noen bevegelige deler i denne.
Annet	Oppdaget at ved å ta utgangspunkt i skissene til beregningsgrunnlaget (de i standarder) ble det straks mer oversiktlig.	I Inventor vil verktøyet *Make Part* fungere som en PS.	Den av strategiene som det trolig er lettest å gjøre feil. På grunn av at geometrien ble laget ut fra solider, opplevdes dette som en mer intuitiv måte å jobbe på sammenlignet med skjelettmetoder hvor en kun jobber med plan og skisser.

5.3.2 Bottom-Up Design

Hele forsøket er beskrevet i vedlegg C.10. For denne metoden var det nødvendig å sette opp *iLogic*-regler for å unngå sirkelrefereringer fra enkeltkomponenter og sammenstillingen. Dette er en løsning som er spesifikk for Inventor. På denne måten kan en også linke hver parameter direkte til Excel, men dette blir noe uoversiktlig da en får mange komponenter fordi en må skrive en kode for hver komponent.

Speiling av komponenter i sammenstillingen ville ikke kopiere fastlåsingene. Dette skapte noe ekstraarbeid under oppsettet, men opplevde ikke problemer med dette i etterkant.

Det var noe mer krevende å analysere alle avhengigheter i denne metoden sammenlignet med *Top-Down*, men så snart fastholdningene var korrekt definert ga også dette en stabil modell.

5.3.3 Middle-Out Design

I praksis kan metoden bli veldig lik en *Bottom-Up* eller en *Top-Down* alt ettersom hvordan en velger å lage delene rundt hovedkomponenten. Dette og resten av resultatet fra testen er beskrevet i vedlegg C.11.

Dersom en trives med å modellere etter *Bottom-Up* kan en ganske enkelt gjøre de andre komponentene adaptive. Er det ønskelig med *Top-Down* skjelett rundt må en projisere den styrende geometrien over på skissen til de komponentene som drives. For å lykkes med dette er det viktig at skissen er fullt definert. Parametre som skal endres må være definert som en drivende dimensjon. Dog er den per definisjon ikke fullt definert dersom noe er drevet.

5.3.4 Oppsummert

Top-Down er den framgangsmåten som tilsynelatende gir de mest stabile modellene ved endring, men å nytte denne metoden i en sammenstilling med svært mange standard komponenter ansees som merarbeid. Det kan også hende *Bottom-Up* modellene ville blitt mer stabile dersom prosjektgruppen hadde hatt høyere erfaringsnivå og gjort bedre analyser av avhengigheter.

5.4 Inventor

I dette avsnittet er det beskrevet hvorden de ulike verktøyene til Inventor var å ta i bruk. Testene dette delkapittelet baserer seg på ligger i vedlegg C.12, C.13 og C.14.

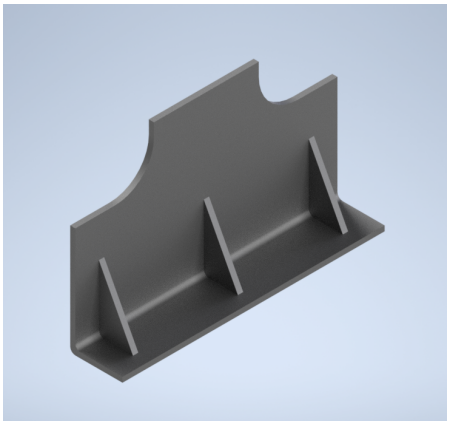
Etterhvert som en modellerte i Inventor ble det tydelig at det hadde betydning hvilken rekkefølge en modellerte de ulike *features*. En kan flytte de opp eller ned i *Design tree* senere ved behov, men det er viktig å ikke basere en ny *feature* på en annen som en senere oppdager en gjerne skulle ha flyttet oppover i *Design tree*.

I en automatisert prosess kan en lage et program hvor en ber det stoppe dersom Inventor sender en feilmelding. På denne måten vil en kunne identifisere grensene til parametrene i en modell. Dette er beskrevet mer inngående i vedlegg C.14.

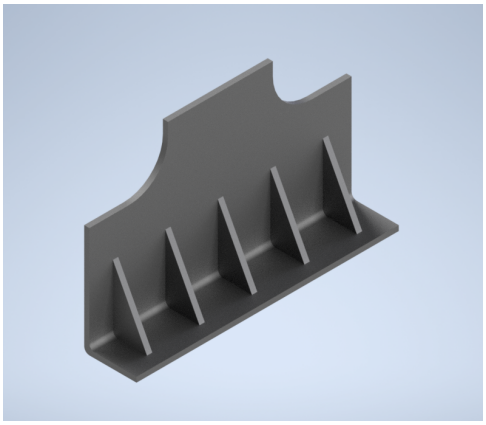
iPart/iAssembly

Dette opplevedes som et enkelt og intuitivt verktøy å nytte. Det er lett tilgjengelige ressurser på nett for å lære seg verktøyene, og siden vil en hurtig kunne produsere svært mange standardiserte komponenter til sitt bibliotek. I figur 5.3 er det vist vinkelen som ble bygget i testen. Etter alt var definert kunne man endre fra tre- til fem ribber kun ved hjelp av et dobbeltklikk.

Derimot viste det seg at denne ikke kunne nyttes til å drive en *Middle-Out* modell. Hele testen av disse verktøyene ligger i vedlegg C.12



(a) Vinkel med tre ribber



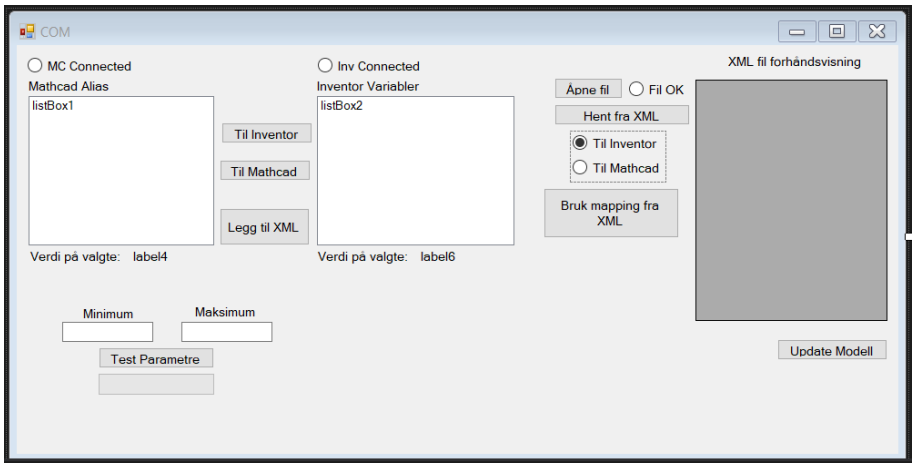
(b) Vinkel med fem ribber

Figur 5.3: Vinkel avstivet av ribber

Deling av data

Testingen av datadelingsfunskjonene til Inventor er beskrevet i vedlegg C.13. Det integrerte verktøyet til Inventor for datadeling med Excel var svært enkel og intuitiv å nytte. Et enkelt søk på Google var nok for å løse dette. Altså er det ikke behov for noen spesielle ferdigheter i koding for å lykkes.

Videre er det meste av APIen til Inventor tilgjengelig og det er svært mange ressurser på nett for hvordan personer tidligere har laget små programmer for å tilpasse Inventor til sitt bruk. Både med hensyn på datadeling, men også lage verktøy for automatisering av prosesser. Et eksempel på en selvlaget applikasjon for kommunikasjon mellom Inventor og Mathcad er vist i figur 5.4. Det vurderes til at det er noe mer behov for ferdigheter innen koding for å lykkes. En burde minimum beherske klasser.



(a) Brukergrensesnittet i testapplikasjonen

```
2 references
public Params[] GetParameters()
{

    storeHere = new Params[_mcOuts.Count];

    for(int i = 0; i<_mcOuts.Count; i++)
    {
        storeHere[i] = new Params();
        storeHere[i].varName = _mcOuts.GetAliasByIndex(i);
        storeHere[i].varValue = _mcNS.OutputGetRealValueAs(storeHere[i].varName,"mm").RealResult;
    }

    return storeHere;
}
```

(b) Koden benyttet for å lese parametre fra Mathcad

```
4 references
public void SendParameter(Params lb_mc, Params lb_inv)
{
    _invParameter = _invParameters[lb_inv.varName];
    _invParameter.Expression = lb_mc.varValue.ToString() + " mm";
}
```

(c) Koden benyttet for å skrive parametre i Inventor

Figur 5.4: Applikasjon for å kommunisere mellom Mathcad og Inventor

5.5 Beregningsprogrammer

Sammenligning av resultatene for de ulike beregningsprogrammene er oppsummert i tabell 5.3. Generelle resultater er beskrevet som tekst nederst i dette avsnittet. I forsøkene har det ikke blitt identifisert noen spesifikke krav som stilles til en 3D-modell for å kunne koble den sammen med et regneark. Derfor vil det være brukervennligheten til programvaren samt hvordan den deler informasjon med Inventor som er avgjørende for hvilken en velger.

De ulike forsøkene som er gjennomført for å kartlegge beregningsprogrammene er beskrevet i vedlegg C.15, C.16, C.17 og C.18. I tillegg er det gjort forsøk på hvordan de er å koble sammen med Inventor og disse testene er beskrevet i vedlegg C.21, C.22 og C.23.

Tabell 5.3: Resultater beregningsprogrammer

	Mathcad	**Jupyter Notebook**	**Excel**
Tid til opplæring i programvaren	Kort tid for å forstå det grunnleggende. Krever erfaring for å utnytte alle verktøy fullt ut.	Krever tid for å lære seg programmeringsspråk. Deretter mye erfaring for å lage gode koder.	Kort tid for å forstå det grunnleggende. Krever erfaring for å utnytte alle verktøy fullt ut.
Gjenbruk	Meget godt egnet.	Meget godt egnet. En kan selv velge layout, så det krever ikke kodekunnskaper for å overta en slik modell.	Meget godt egnet.
Kontroll av utregninger	Svært intuitivt.	Krever tid å sette seg inn i kodingen	Krever tid å sette seg inn i regnearket.
Dokumentasjon	Meget godt egnet.	Meget godt egnet.	Egnet.
Sammenkobling med 3D-modell i Inventor	Lykkes med å sette opp et program for både semi-automatisk og helt automatisk datadeling.	Kort kodesnutt i selve regnarket løste dette. Krever forståelse for objekt orentert programmering for å lykkes.	Integrert- Løses ved en *iLogic*-kode eller en plug-in vha. VBA/C#/C++.
Annet		Dersom en behersker koding vil en ha ubegrenset med muligheter. Andre programmer vil en begrenses av verktøyene i programvaren.	Begrenset kontroll på avrunding. Kun til heltall eller hvert tiende heltall. Kan bruke Mathcad til dokumentasjon da data fra Excel kan overføres til Matchcad.

Det som raskt identifiseres som utfordrende er å sette opp beregninger på en slik måte at en kan bruke færrest mulig parametre for å styre utregningene. Det er enkelte valg som må tas for å komme seg videre, slik som eksempelvis platetykkelse for å få en gjeldende flytegrense for sitt materiale. Derimot kan det settes opp slik at beregningen iterere seg fram til korrekt platetykkelse og gjeldende flytegrense da flytegrensen i dette tilfellet også var avhengig av tykkelsen.

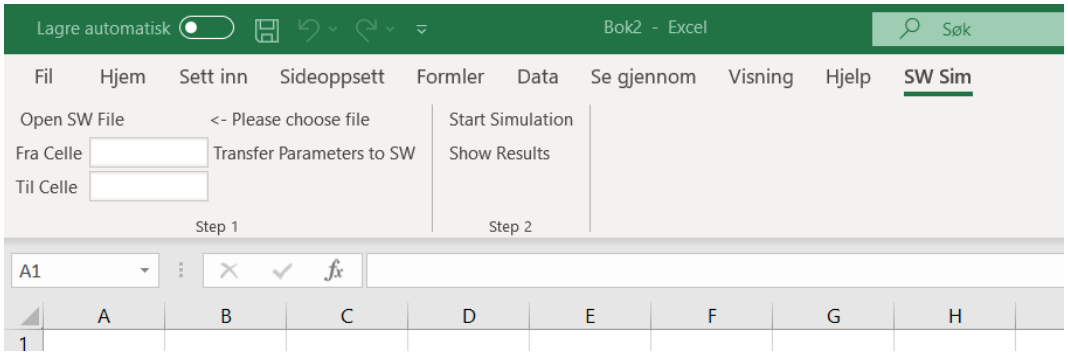
5.6 Analyseprogrammer

For å enklere sammenligne resultatene for de ulike beregningsprogrammene er de oppsummert i tabell 5.4. Denne sammenligningen baserer seg på testene som er beskrevet i vedlegg C.19, C.20 og C.24. Generelle resultater er beskrevet som tekst nederst i dette avsnittet.

Tabell 5.4: Resultater analyseprogrammer

	Inventor	**SolidWorks Simulation**	**ANSYS**
Analyse-verktøy	Få muligheter, men den kan nyttes til enkle kontroller.	Mange muligheter og god kontroll.	Mange muligheter og god kontroll.
Automatisk overføring av 3D-modell	Begrense muligheter da analysemiljøet ikke er tilgjengelig i APIen.	Åpen API som gjør det mulig å automatisere.	Integrert verktøy for import av DAK-modeller, men framstår som ikke-mulig å automatisere dette
Annet		Lykkes med å sette dette opp et program hvor modellering og analyse styres fra Excel.	Testingen har blitt noe begrenset av studentlisens.

Fra forsøkene som er gjennomført var det ikke mulig å automatisere prosessen med analyse i ANSYS. Tilsvarende prosess i SolidWorks var tidkrevende å løse første gang, men ellers opplevdes det som uproblematisk. Resultatet av dette er vist i figur 5.5. Det er satt opp en egen fane i Excel hvor du kan styre eksport av data til SolidWorks.



Figur 5.5: Egen fane i Excel som styre kommunikasjon med SolidWorks

6. Drøfting

I drøftingen er begrunnelsen for enkelte valg presentert i tillegg til at styrker og svakheter ved rapporten er diskutert. Dette kapittelet ansees som nødvendig for at leseren skal kunne danne seg en mening om rapportens validitet.

Kapittelet er bygget opp med utgangspunkt i prosessdiagrammet og de tre hovedfasene som er definert der: Utredningsfasen, utviklingsarbeid og endelig produkt. Disse har fått hvert sitt delkapittel og deles så videre opp etter det som er hensiktsmessig.

6.1 Utredningsfasen

Utredningsfasen deles videre inn i to hoveddeler som drøftes. Dette er selve planleggingen og teoridelen som ligger til grunn for resten av rapporten. Drøftingen av disse er beskrevet i to ulike avsnitt.

6.1.1 Plan

I planleggingen av et prosjekt settes rammene for prosjektet og dette kan påvirke resultatet. Derfor er det viktig å vurdere om det er noe i denne fasen som kunne vært løst på en bedre måte. Det er valgt å se på tre områder innenfor planlegging som skal gjennomgås: Problemformuleringen, valg av metode og kvalitetssikringstiltak. Disse momentene er å finne i kapittel 1 og 4 i rapporten.

Problemformulering

Det er valgt å ha en kort tekst for å holde oppgaven mest mulig åpen. Allikevel er hovedpunktene som er viktig for rapporten godt definert innenfor teksten. På denne måten blir det tydelig hva som er hensikten med rapporten, men oppgaveteksten setter ingen begrensinger for hvordan resultatet blir.

Valg av metode

Avsnittet tar for seg metoden brukt under utviklingsarbeidet i prosessdiagrammet vist i figur 4.1. Hver strategi som er testet er basert på prosjektdeltakernes tolkning av det som er beskrevet i kapittel 2 og resultatet begrenses derfor av Maja og Morten sine erfaringer. Akkurat denne faktoren gir liten mulighet for å gjenskape resultatene. Allikevel baserer refleksjonene seg på teori etablert av forskere og vitenskapelige tekster, så resultatene bør ikke avvike i stor grad ved gjentakelse av forsøkene.

Kvalitetsikringstiltak

Kvalitetsikringstiltakene beskrevet i kapittel 4 er i svært stor grad fulgt. Tett dialog med veildere fra både NMBU og Multiconsult har ført til at feil er blitt oppdaget før det tar for mye tid å rette dem opp. Samtidig har det bidratt til at en raskt ser hvilken retning som er riktig å følge og dermed bidratt til et resultat som dekker behovet beskrevet i kapittel 1. Tilbakemeldinger fra personer helt utenfor prosjektet har bidratt til at teorien er presentert på en god og forståelig måte for de med en maskinteknisk bakgrunn.

Likevel har størsteparten av kvalitetsikringen vært internt i prosjektgruppen mellom Maja og Morten. Deltakerne har fulgt hverandres arbeid tett og det har vært stor takhøyde for kritiske tilbakemeldinger. Dette tiltaket ble ikke beskrevet i kapittel 4 da det i forkant ikke ble tenkt gjennom hvor viktig dette kunne være. Likevel har det vært det viktigste verktøyet for effektivt og målrettet arbeid.

Totalt sett er det vurdert til at kvalitetsikringen har vært tilstrekkelig for å levere en rapport med et høyt faglig nivå.

6.1.2 Teori

En av vanskelighetene med litteratursøket til denne oppgaven er at det er mange bedrifter som velger å skjule sine kunnskap og erfaring innefor dette fagfeltet. I tillegg er det som er gjort av masteroppgaver er svært spesifikke og tar gjerne for seg koblingen mellom to bestemte programvarer eller hvordan parametrisere en bestemt modell.

Kilder

Nedenfor er en vurdering av de hovedkildene som er nyttet i rapporten:

Metode	Styrker	Svakheter
Bøker, rapporter, artikler etc.	Pålitelige	Mye informasjon
Intervjuer	Mye erfaring som er vanskelig å lese seg opp på	Vanskelige å etterprøve Kan oppstå misforståelser
Websider	Raskt å finne informasjon Enkelt å sjekke opp flere sider	Endrer seg raskt, kan derfor være vanskelig å etterprøve Kan være vanskelig å vurdere validiteten

Det er blitt brukt betydelig flere websider enn hva som var tenkt i planleggingsfasen. Dette fordi det er skrevet mye om programvarer og disse oppdateres gjerne hvert år. Det publiseres ikke bøker som beskriver verktøyene til programvarene siden bøkene nødvendigvis vil være utdatert etter kun et år. Websider er derimot oppdaterte og er blitt brukt som hovedkilder til å beskrive det som omhandler programvarer.

Presentasjon av teorien

For å begrense hva som skrives innenfor for hvert tema presentert i kapittel 2 er det valgt å holde det helt grunnleggende. Det er forsøkt å plukke ut det som er relevant for problemstillingen. Videre er det beskrevet på en slik måte at en med grunnleggende erfaring innenfor DAK skal kunne forstå innholdet. Det har vært viktig å skape en rapport som kan nyttes til opplæring for de som skal starte å jobbe med DAK fordi det er ønskelig at rapporten skal brukes mest mulig

6.2 Utviklingsarbeid

Metoden som er nyttet er drøftet i avsnitt 6.1.1 under «Valg av metode». I dette delkapittelet vil en kun gå inn på to konkrete oppgaver som har hatt stor betydning for der endelige resultatet: Bearbeiding av teorien og testingen.

6.2.1 Bearbeiding av teorien

Den største usikkerheten ved bearbeid av teori er om prosjektdeltakerene har forstått strategiene godt nok til å kunne vurdere disse skikkelig. Ved å kun ha grunnleggende kunnskap innenfor modellering vil en mangle erfaring til å forutse hvilke problemer som ofte oppstår ved endring av 3D-modeller. Det er nettopp robusthet ved endring av modeller som strategiene baserer seg på.

Det har vist seg at resultatene til prosjektgruppen i stor grad samsvarer med resultatene i de vitenskapelige artiklene og derfor er det grunn til å anta god validitet ved resultatene i denne rapporten. Allikevel vil det anbefales at en ingeniør med mye erfaring innenfor DAK også tester ut de ulike strategiene.

6.2.2 Testing

Det har blitt testet ulike strategier for oppbygging av komponenter og ulike strategier for sammenstilling, i tillegg har beregningsprogrammer og analyseprogrammer blitt testet og sammenkobling av ulike programvarer. Dette betyr mange ulike kombinasjonsmuligheter og mange tester bare for å gjennomføre alle kombinasjoner en gang. Derfor vil mange av resultatene belager seg på kun én test og det anbefales å gjennomføre flere tester før en endelig konklusjon kan trekkes.

I tillegg vil prosjektdeltakerne sin begrensede erfaring med modellering kunne føre til lavere forståelse av 3D-modellenes bruksområde og at det dermed er noen aspekter som det ikke er tatt hensyn til. Derfor burde *Design Intent* som er utviklet i dette prosjektet testes og godkjennes av en ingeniør med mye erfaring innenfor DAK og de type modellene bedriften ofte designer.

Det har heller ikke blitt vurdert til å være hensiktsmessig å bruke ressurser på å finne store grupper som kan teste hvordan det er å overta de ulike modellene da det er funnet forskningsartikler hvor slike forsøk er gjort. Dermed er noe av konklusjonen i kapittel 7 basere seg direkte på teorien i kapittel 2.

6.3 Endelig produkt

I denne delen av drøftingen skal det endelige produktet vurderes. Dette er delt inn i to deler hvor «Resultater» tar for seg det som er skrevet i kapittel 5 og «*Design Intent*» tar for seg det som er presentert i vedlegg A.

6.3.1 Resultater

Resultatene er presentert på en todelt måte. Det er valgt å ta inn i selve rapporten hvordan opplevelsen av bruk er. Det er disse tingene som er viktige for å kunne anbefale en gitt strategi som er målet med dette. I vedlegg C er det skrevet grundigere og mer teknisk om hvert enkelt forsøk slik at det er mulig å gjenskape disse testene. Ved å dele det opp på denne måten er det tenkt at det skal være lettere for leseren å finne fram til det en søker. Selve rapporten blir mer konsis, men det er allikevel mulig å gå i vedleggene for å etterprøve resultatene.

Usikkerheten rundt resultatene er diskutert i avsnitt 6.2.2.

6.3.2 Design Intent

Det er relativt mange kriterier som ligger til grunn for å vurdere om det er blitt et godt resultat i denne rapporten. Det er likevel brukervennlighet, robusthet og designfleksibilitet som er de viktigste. Dette basert på oppgaveformulering og mål med oppgaven. For å beslutte hvilken løsning som anbefales må en nødvendigvis inngå et kompromiss mellom de ulike kriteriene og vekte hvilket som skal være viktigst. Listen er bygget opp slik at den er til hjelp i planprosessen og er til støtte for den som overtar en modell.

Løsningen som er valgt kan oppfattes som noe avansert da det inngår en del programmering i den. Dette ble allikevel anbefalt fordi det er vurdert at programmering stadig vil bli mer aktuelt. Ved å ikke gå inn for en slik løsning fryktes det at en velger en løsning som er utdatert allerede før den er implementert i bedriften.

Prosjektdeltakerene ikke har forutsetninger for å kunne gjøre gode vurderinger på hvilke komplikasjoner som en kan forvente ved endringer av modeller. Det har blitt gjennomført intervju med ansatte ved Multiconsult for å øke forståelse på nettopp dette. Likevel er det ikke forventet at alle problemstillinger er blitt oppfattet og det er viktig at denne biten gjennomgås kritisk og strategiene prøves ut før denne distribueres i bedriften.

7. Konklusjon

I dette kapittelet beskrives det hvilke konklusjoner som er utarbeidet av denne rapporten. Dette gjøres ved å svare på hvert enkelt resultatmål satt for dette arbeidet. I første avsnitt gjengis målene fra kapittel 1 og i neste avsnitt besvares hvert mål helt konkret. Disse konklusjonene er bygget på teorien fra kapittel 2 og resultatene beskrevet i kapittel 5.

Flere av målene krever konkrete anbefalinger på strategier og dermed er det gitt konkrete svar. Likevel blir det tydelig at det som er aller viktigst for en bedrift er å ha en enhetlig strategi for oppbygging av 3D-modeller. For at valg av strategi skal ha stor betydning for modellens robusthet, må det være snakk om komplekse modeller.

7.1 Resultatmål

Målene nedenfor er en nøyaktig gjengivelse av målene beskrevet i kapittel 1.

Hovedmål

1	Anbefale en god strategi på hvordan bygge opp en parametrisert modell i et DAK program.
2	Kartlegge hvordan en mest hensiktsmessig bygger opp digitale tvillinger med 3D DAK modeller.

Delmål

1	A	Beskrive parametrisert modell og digitale tvillinger. Beskrive hva som tidligere er dokumentert rundt dette og se om det er noen strategier som utpeker seg.
	B	Gjennomføre tester av det som beskrives i teorien og beskrive hvilke resultater en da finner.
	C	Utarbeide en hensiktsmessig strategi for hvordan bygge en parametrisert modell.
2	A	Se på hvilke beregningsprogrammer som kan kobles opp mot modelleringsprogrammer og gi en anbefaling om hvilket som bør nyttes.
	B	Se på hvilke analyseprogrammer som kan kobles opp mot modelleringsprogrammer og gi en anbefaling om hvilket som bør nyttes.
	C	Definere i hvilket program en bør legge inn de drivende parametrene for å styre modellen.
	D	Identifisere eventuelle krav til 3D-modellen for at en slik metode kan benyttes.

7.2 Måloppnåelse

I dette avsnittet vil det gis et svar på hvert av resultatmålene. Disse svarene er korte, konsise og med henvisninger der dette er naturlig uten at det i stor grad drøftes rundt hver konklusjon. Dette er for at leseren raskt skal kunne danne seg et overblikk. For å oppnå en dypere forståelse henvises det til de tidligere kapitlene i rapporten.

7.2.1 Hovedmål 1

Bedriften anbefales å etablere en felles mal i sitt DAK-team for hvordan bygge opp en parametrisert modell og *Design Intent* bør innføres som et naturlig steg i hver modelleringsprosess. Planlegging i forkant, og at hver DAK-ingeniør kjenner til oppbyggingen, er i denne rapporten identifisert som de viktigste faktorene for effektiv modellering internt i en bedrift.

Det anbefales at bedriften innarbeider RMS som strategi for hvordan bygge 3D modeller. Dette er en helhetlig strategi og den strategien som kommer best ut i tidligere forskningsartikler. Den er tilgjengelig gratis og har en hjemmeside som stadig oppdateres.

Videre anbefales det at det er ingeniører med erfaring som bygger parametriserte modeller da de har best forutsetninger til å forutse endringsbehovet.

Delmål 1A

Dette er beskrevet i kapittel 2 og kapittel 3.

Delmål 1B

Alle resultater er gjengitt i kapittel 5.

Delmål 1C

Det er utarbeidet et forslag til *Design Intent* som ligger i vedlegg A. En slik sjekkliste vil fungere med to formål. Først vil den kunne viderebringe informasjon til den som overtar en modell og så vil den begrense antall muligheter for hvordan designe en 3D-modell. Ved å redusere antall muligheter bidrar den til at bedriften har en enhetlig strategi.

7.2.2 Hovedmål 2

Det anbefales å skreddersy et program som står for datadeling mellom de ulike programvarene. På denne måten vil den kunne tilpasses etter behov da ikke alle digitale tvillinger har samme formål. I tillegg står en friere til å selv velge hvilke programvarer en ønsker å bruke til modellering, beregninger og analyser.

Programmering er antatt å bli svært viktig i framtiden og bedrifter som ikke behersker dette risikerer å tape konkurransen i næringslivet. Implementering av programmering i arbeidsoppgaver kan gjøres gradvis og tiden som vil brukes på opplæring vil trolig tjenes inn på sikt.

Delmål 2A

På grunn av Inventor sin åpne API kan alle beregningsprogrammer dele data med dette modelleringsprogrammet. Excel er det eneste beregningsprogrammet som er integrert i Inventor for datadeling og hvor det ikke er behov for et eksternt program.

Jupyter Notebook er det beregningsprogrammet som anbefales å benytte. Her kan en lage alle funksjoner en har behov for og begrenses ikke av funksjonalitet eller verktøy i selve programvaren.

Delmål 2B

For å kunne automatisere en prosess anbefales det å bruke SolidWorks som analyseprogram. Der vil en får tilgang på alle analysemiljøer og den åpne APIen gir tilgang til å styre den utenfra.

Delmål 2C

I utgangspunktet kan en legge inn de styrende parametrene hvor en vil da dette ikke vil påvirke robusteheten av den digitale tvillingen. Det må bare defineres hvor parametrene skal hentes i det eksterne programmet som kobler sammen alle modellene.

Allikevel anbefales det å legge i beregningsprogrammet da dette vil være det som er mest intuitivt å navigere for en som overtar en modell. I tillegg kan en hurtig kontrollere utregningene før de resterende modellene oppdateres.

Delmål 2D

Der er ikke funnet at det er noen krav som stilles til en 3D modell for at den skal kunne nyttes i en digital tvilling. Bruk av strategi ses på som et verktøy for å bedre modellens lesbarhet for nye ingeniører.

7.3 Videre arbeid

De påfølgende avsnittene inneholder anbefalinger til videre arbeid for å ytterligere kvalitetsikre eller optimalisere det arbeide som er utført i denne rapporten.

7.3.1 Design Intent

Se vedlegg A for lese forslaget til *Design Intent*.

Kvalitetsikring

Sjekklisten burde kvalitetsikres og viderutvikles av en erfaren DAK-ingeniør før den eventuelt tas i bruk. Både med hensyn på at gode strategier er valgt og med hensyn på struktur i malen. Det er viktig at den er bygget opp på en hensiktsmessig måte med tanke på hvordan den skal brukes senere. Dette for å unngå å bruke tid på implementering før en har forvisset seg om at den er hensiktsmessig.

Videreutvikling

Det er mulig å lage et enkelt program som velger ut spørsmål basert på tidligere svar og som deretter generere en rapport som følger modellen. Dette vil kunne gi ryddigere rapporter da det kun medfølger nødvendig informasjon for nettopp denne modellen.

7.3.2 Testing

Både strategier som er tatt med videre og strategier som er utelukket kan testes flere ganger på ulike typer modeller for å kontrollere at det er de beste strategiene som er valgt ut. Det krever et visst antall tester for å kunne fastslå at anbefalt løsning er den mest hensiktsmessige.

7.3.3 Programmering

Utvikle et eget program som er klar til å koble sammen beregninger, 3D-modeller og analyse. Ideelt sett kunne det være programmert med ulike valgmuligheter for hvilke deler den skal koble sammen og om det skal nyttes til optimalisering.

7.3.4 Optimalisering

Det ble ikke gjort noen forsøk med optimalisering. Det ble gjennomført tester på hvordan kommunisere mellom de ulike programvarene som vil være å tilrettelegge for optimalisering, men det gjenstår ennå å teste hvordan dette best løses i praksis.

Verktøy i programvarene

I kapittel 2 er optimaliseringsverktøyene til de ulike programvarene beskrevet. Disse må testes for å avdekke hva som dekker bedriftens behov og hvor brukervennlige de er.

Optimaliseringsloop

Videre burde det settes sammen en loop fra beregninger, til 3D-modell og til analyse hvor det først testes hvordan dette kan gjennomføres som en semi-automatisk prosess for deretter å gjøre den heltautomatisk. Siden må det avgjøres hvilken av disse som er å foretrekke i ulike sammenhenger.

Bibliografi

[1] Jorge D. Camba, Manuel Contero, and Pedro Company. Parametric CAD modeling: An analysis of strategies for design reusability, 2016. ISSN 0010-4485. URL http://www.sciencedirect.com/science/article/pii/S0010448516000051.

[2] Kort om Multiconsult. URL https://www.multiconsult.no/om-oss/kort-om-multiconsult/. (Hentet: 08.01.2020).

[3] Definisjon av parameter. URL https://snl.no/parameter. (Hentet: 08.01.2020).

[4] Resilientmodeling.com. URL http://resilientmodeling.com. (Hentet: 28.02.2020).

[5] Rose Bertrand Caillaud Emmanuel Bodein, Yannick. Explicit reference modeling methodology in parametric CAD systems. *Computers in industry*, 65(1), 2014. doi: 10.1016/j.compind.2013.08.004. URL https://doi.org/10.1177/1687814016651210.

[6] SolidWorks Design Intent, . URL http://help.solidworks.com/2012/English/SolidWorks/acadhelp/Design_Intent.htm. (Hentet: 13.04.2020).

[7] Paul Munford. Reliable Modelling Techniques for Comples Assembly Design in Autodesk Inventor. URL https://www.autodesk.com/autodesk-university/class/Reliable-Techniques-Complex-Assembly-Design-Autodesk-Inventor-2019. (Hentet: 24.04.2020).

[8] PTC Design Intent, . URL https://www.ptc.com/en/cad-software-blog/design-intent-explained. (Hentet: 13.04.2020).

[9] *Horizontally-structured CAD/CAM modeling for virtual concurrent product and process design*, 2004.

[10] Søk i EPO. URL https://worldwide.espacenet.com/patent/search?called_by=epo.org&q=Horizontal-structured%20CAD%2FCAM%20. (Hentet: 12.05.2020).

[11] Ron K. C. Cheng. Autodesk Inventor Tutorial 2: Introduction to Part Modeling. URL http://acad-atc.ic.polyu.edu.hk/inventor/02part.pdf. (Hentet: 24.04.2020).

[12] Autodesk knowledge network: Top-down, bottom-up, middel-out design. URL https://knowledge.autodesk.com/support/inventor/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Inventor-Help/files/GUID-63FA128E-63E2-4176-8653-327BD80D8A43-htm.html. (Hentet: 15.01.2020).

[13] Xiang Chen Youdong Yang Shuming Gao, Shuting Zhang. A framework for collaborative top-down assembly design. URL https://www.sciencedirect.com/science/article/pii/S0166361513001139. (Hentet: 09.05.2020).

[14] Top-Down og Bottom-Up. URL https://en.wikipedia.org/wiki/Top-down_and_bottom-up_design. (Hentet: 30.04.2020).

[15] Use skeletal modeling, Autodesk. URL https://knowledge.autodesk.com/support/inventor/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/Inventor-Help/files/GUID-0435FE9D-6283-4CE0-813B-BD9B0BE5778C-htm.html. (Hentet: 30.04.2020).

[16] Dexin Chu, Guolin Lyu, Xuening Chu, and Jing Shen. Multi-skeleton model for top-down design of complex products. *Advances in Mechanical Engineering*, 28(6), 2016. doi: 10.1177/1687814016651210. URL https://doi.org/10.1177/1687814016651210.

[17] Multi-body parts, Autodesk, . URL https://knowledge.autodesk.com/support/inventor-products/learn-explore/caas/CloudHelp/cloudhelp/2014/ENU/Inventor/files/GUID-8E641EB9-90E7-40AB-ACC6-222CB613E712-htm.html. (Hentet: 30.04.2020).

[18] Multi-body parts, SolidWorks, . URL http://help.solidworks.com/2017/english/solidworks/sldworks/c_multibody_parts_vs_assemblies.htm. (Hentet: 01.05.2020).

[19] Hva er digitale tvillinger?, . URL https://www.obforum.com/lederblikk/hege-skryseth-kan-virksomheten-din-dra-nytte-av-digitale-tvillinger-og-digitale-trader. (Hentet: 09.01.2020).

[20] Digitale tvillinger, Deloitte, . URL https://www2.deloitte.com/no/no/pages/technology/articles/digitale-tvillinger-praksis.html. (Hentet: 30.04.2020).

[21] Jasbir Singh Arora. *Introduction to optimum design, 4th edition*. Academic Press is an imprint of Elsevier, 2016. ISBN 0128008067.

[22] Programmeringsgrensensitt, Store norske leksikon. URL https://snl.no/programmeringsgrensesnitt. (Hentet: 30.04.2020).

[23] DAK, Store norske leksikon. URL https://snl.no/DAK_-_IT. (Hentet: 01.05.2020).

[24] Brian Ekins. Unleashing Hidden Powers of Inventor with the API, Part1. URL http://download.autodesk.com/us/community/mfg/Part_1.pdf. (Hentet: 01.05.2020).

[25] Visual Basic Wikipedia. URL https://no.wikipedia.org/wiki/Visual_Basic. (Hentet: 07.04.2020).

[26] Inventor Developer Center, . URL https://www.autodesk.com/developer-network/platform-technologies/inventor/overview. (Hentet: 01.05.2020).

[27] iLogic Autodesk. URL https://knowledge.autodesk.com/support/inventor/learn-explore/caas/CloudHelp/cloudhelp/2020/ENU/Inventor-iLogic/files/GUID-AB9EE660-299E-408F-BBE1-AFE44C723F59-htm.html. (Hentet: 08.04.2020).

[28] iPart Autodesk. URL https://knowledge.autodesk.com/support/inventor/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/Inventor-Help/files/GUID-9D7FF4CB-6045-4E2A-AC88-40A2F4DDF392-htm.html. (Hentet: 08.04.2020).

[29] iAssembly Autodesk. URL https://knowledge.autodesk.com/support/inventor/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/Inventor-Help/files/GUID-6E529299-CAB9-4F5C-B100-7901D877B83F-htm.html. (Hentet: 08.04.2020).

[30] AutoLimits Autodesk. URL https://knowledge.autodesk.com/support/inventor/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Inventor-Help/files/GUID-F9592987-AEC6-45D7-BD46-7261758671AD-htm.html. (Hentet: 08.04.2020).

[31] Excel Data Link functions in iLogic reference, . URL https://knowledge.autodesk.com/support/inventor-products/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Inventor-Help/files/GUID-00F9D915-B53A-46BF-AAAC-3535F9FD9970-htm.html. (Hentet: 01.05.2020).

[32] Getting Started with Inventor's API. URL http://help.autodesk.com/view/INVNTOR/2020/ENU/?guid=GUID-4939ABD1-A15E-473E-9376-D8208EC029EB. (Hentet: 02.05.2020).

[33] Microsoft: Automation. URL https://docs.microsoft.com/en-us/cpp/mfc/automation?view=vs-2019. (Hentet: 02.05.2020).

[34] Microsoft: Excel Automation. URL https://docs.microsoft.com/en-us/previous-versions/office/troubleshoot/office-developer/automate-excel-from-visual-c. (Hentet: 02.05.2020).

[35] Mathcad.com, . URL https://www.mathcad.com/en/. (Hentet: 01.05.2020).

[36] Solve Blocks Mathcad, . URL http://support.ptc.com/help/mathcad/en/index.html#page/PTC_Mathcad_Help%2Fabout_solve_blocks.html%23. (Hentet: 09.04.2020).

[37] Mathcad support, . URL http://support.ptc.com/help/mathcad/en/index.html#page/PTC_Mathcad_Help. (Hentet: 22.01.2020).

[38] Mathcad integrering med Creo. URL http://support.ptc.com/help/mathcad/en/index.html#page/PTC_Mathcad_Help%2Fabout_mathcad_and_proe_integration.html%23. (Hentet: 09.04.2020).

[39] Mathcad integrering med Excel, . URL http://support.ptc.com/help/mathcad/en/index.html#page/PTC_Mathcad_Help%2Fto_import_data_from_excel.html%23wwIDOEYTUY. (Hentet: 09.04.2020).

[40] Mathcad API, . URL http://support.ptc.com/help/mathcad/en/index.html#page/PTC_Mathcad_Help%2Fabout_mathcad_and_automation_api.html%23. (Hentet: 09.04.2020).

[41] jupyter.org. URL https://jupyter.org. (Hentet: 01.05.2020).

[42] Eksempelplan Maskin. URL https://www.nmbu.no/studier/studietilbud/master-femarig/maskin_prosess_og_produktutvikling/eksempelplaner. (Hentet: 01.05.2020).

[43] Python Home About. URL https://www.python.org/about/. (Hentet: 01.05.2020).

[44] ipywidgets - Jupyter Widgets. URL https://ipywidgets.readthedocs.io/en/latest/. (Hentet: 08.05.2020).

[45] scipy.org. URL https://www.scipy.org. (Hentet: 01.05.2020).

[46] The win32com package. URL http://timgolden.me.uk/pywin32-docs/html/com/win32com/HTML/package.html. (Hentet: 01.05.2020).

[47] Excel resurser, microsoft.com. URL https://www.microsoft.com/nb-no/microsoft-365/excel?rtc=1. (Hentet: 01.05.2020).

[48] Excel name range, . URL https://www.ablebits.com/office-addins-blog/2017/07/11/excel-name-named-range-define-use/. (Hentet: 12.05.2020).

[49] Excel Solver, . URL https://support.office.com/en-us/article/define-and-solve-a-problem-by-using-solver-5d1a388f-079d-43ac-a7eb-f63e45925040. (Hentet: 09.04.2020).

[50] Om analyser i Inventor, . URL https://knowledge.autodesk.com/support/inventor-products/learn-explore/caas/CloudHelp/cloudhelp/2020/ENU/Inventor-Help/files/GUID-61F01A5D-7E54-45A1-9698-7BB11F0AEE94-htm.html. (Hentet: 10.04.2020).

[51] Om Inventor Nastran, . URL https://www.autodesk.com/products/inventor-nastran/overview?plc=PDCOLL&term=1-YEAR&support=ADVANCED&quantity=1#internal-link-features. (Hentet: 10.04.2020).

[52] Om Inventor *Shape Generator*, . URL http://help.autodesk.com/view/INVNTOR/2018/ENU/?guid=GUID-D74F47F3-FE22-44EF-85BE-7C6B1F56DCF9. (Hentet: 10.04.2020).

[53] Om Nastran topologi optimalisering. URL https://www.autodesk.com/autodesk-university/class/Topology-Optimization-Autodesk-Nastran-CAD-2017. (Hentet: 10.04.2020).

[54] Full object model, . URL https://help.autodesk.com/cloudhelp/2020/ENU/Inventor-API/images/FullObjectModel.png. (Hentet: 12.05.2020).

[55] SolidWorks Simulation, . URL https://www.solidworks.com/product/solidworks-simulation. (Hentet: 11.04.2020).

[56] Hjemmesiden til DriveWorks. URL https://www.driveworks.co.uk. (Hentet: 12.05.2020).

[57] SolidWorks Topology Studies, . URL http://help.solidworks.com/2018/english/WhatsNew/c_generative_design_study.htm. (Hentet: 11.04.2020).

[58] SOLIDWORKS API Help. URL https://help.solidworks.com/2020/English/api/sldworksapiprogguide/Welcome.htm. (Hentet: 12.05.2020).

[59] ANSYS. URL https://www.ansys.com/products/structures. (Hentet: 11.04.2020).

[60] Om optiSlang. URL https://www.ansys.com/products/platform/ansys-optislang. (Hentet: 12.05.2020).

[61] Workbench and Excel, Part 1: Using the Excel Component System, . URL https://www.padtinc.com/blog/workbench-and-excel-part-1-using-the-excel-component-system/. (Hentet: 12.05.2020).

[62] CAD Configuration Manager and Solidworks with Ansys Student ver, . URL https://studentcommunity.ansys.com/thread/cad-configuration-manager-and-solidworks-with-ansys-student-ver/. (Hentet: 12.05.2020).

[63] Kurs fra Symetri, . URL https://www.symetri.no/kurs/. (Hentet: 24.04.2020).

[64] Kurs fra PLM Group, . URL https://plmgroup.no/kurs/#training-calendar. (Hentet: 24.04.2020).

[65] Events EDR Medeso, . URL https://digitallabs.edrmedeso.com/events-blog. (Hentet: 24.04.2020).

[66] Emner ved Fakultet for ingeniørvitenskap ved NTNU, . URL https://www.ntnu.no/studier/emnesok#semester=2019&gjovik=false&trondheim=false&alesund=false&faculty=834&institute=1218&multimedia=false&english=false&phd=false&courseAutumn=false&courseSpring=false&courseSummer=false&pageNo=1&season=spring&sortOrder=ascTitle. (Hentet: 24.04.2020).

[67] Emner ved ved NMBU, . URL https://www.nmbu.no/emnesok. (Hentet: 24.04.2020).

[68] Informasjon om søketjenesten til NIPO. URL https://search.patentstyret.no/advanced/#/about. (Hentet: 24.04.2020).

[69] L Dieter, G. E og Schmidt. *Engineering design, 4.opplag.* McGraw-Hill International Edition, 2009. ISBN 9780072837032.

Tillegg

A. Design Intent

A.1

Prosjekt nr.:
Modell nr.:
Dato:

Design.:
Kontr.:

Design Intent

3D-modell

Modell:
Beskrivelse av hva du designer

Bruksområde:
Illustrasjon, konstruksjon etc.

Hva er de viktige parameterne? *(tips: de som nyttes i beregninger er ofte de viktige)*	
Variable:	**Konstante:**
Hvilken av disse er som er styrende:	

Strategi for sammenstilling:		
Top-down: *Benyttes dersom den i hovedsak består av unike deler.*	**Middel-out:** *Benyttes dersom det er en komponent/sub-sammenstilling som er styrende for resten av sammenstillingen*	**Bottom-up:** *Benyttes dersom den skal bygges opp av mange standardiserte komponenter.*
Beskrivelse av hvilken Top-down strategi som er nyttet	*Beskriv hvilken komponent som er styrende*	

Beskrivelse av skjelettstrukturen:
Dersom dette er nyttet.

Digital tvilling

Skal modellen kobles sammen med et beregningsark?	
Ja:	**Nei:**
Beskriv hvilket program og hvor en skal endre på de styrende parametrene	

Skal modellen kobles sammen med et analyseprogram?	
Ja:	**Nei:**
Hvilket program	

Beskrivelse av programmet for datadeling:

B. Ståstedsanalyse

B.1 Erfaringsskriv Multiconsult

Parametriserte modeller – erfaringer fra Multiconsult

Denne rapporten er skrevet på oppdrag fra Multiconsult, nærmere bestemt maskinsektoren i enheten for fornybar energi. De har allerede jobbet noe med parametrisering av 3D-modeller tidligere og de erfaringene de har gjort seg i den forbindelsen er samlet i dette skrivet. Alt innholdet er basert på intervju med to av de ansatte i maskinsektoren, samt innholdet i en bacheloroppgave skrevet for Multiconsult.

Bacheloroppgave

I 2016 ble det gjennomført en bacheloroppgave av fire studenter ved Høgskolen i Oslo og Akrashus: Parametrisk desig av rulleluke (A. Sørensen, Ø. Fredriksen, K.O.A. Smedsgård og O.J.D. Ulveseth). Her designet de en parametrisk rulleluke integrert med beregninger i PTC Mathcad.

De nyttet en *Top-Down Skeleton*-metode med et mor-skjelett og flere datter-skjeletter som skulle forhindre at mor-skjelettet ble uoversiktlig. Selv mente de god planlegging var avgjørende for at de lykkes med sin parametriserte modell. Styringen av parametere foregikk gjennom Inventor sin funksjon kalt *Derive.* Den sikrer at hver komponent kun bruker de parameterne den trenger fra mor-skjelettet. Kommunikasjonen mellom Inventor og Mathcad gikk via Excel.

Multiconsult sin erfaring med bruk av modellen er at *skeleton*-metoden ga en robust modell, men at komminkasjonene mellom regneark og 3D-modellen ikke var en like god løsning. Problemet var at kommunikasjonen var satt opp med en kode som ble for avansert, slik at Multiconsult klarte ikke selv å rette opp i eventuelle feil som oppstår.

Bruk av strategier

Med tanke på at *Top-Down Skeleton* viste seg å være en god metode for å modellere luker, har denne praksisen blitt videreført for luker.

Ved design av elementer i tilknytning til turbinen, har det vist seg å være mer hensiktsmessig med en *Middel-Out* strategi. Dette fordi turbinens løpehjul er det som er dimensjonere for alt det omkringliggende.

Det har også blitt bygget luker med en *Middel-Out* strategi. Her har rammens luke blitt brukt som den førende komponenten. Også dette har resultert i stabile modeller.

Positive erfaringer

Ved å ha origo i sentrum av luken har en kunne bygge hele sammenstillingen ved å plassere komponenter ut fra origo. Dette har vist seg å gi mer stabile modeller sammenlignet med om en bruker fastholdninger for å plassere komponenter i forhold til andre komponenter.

Det er svært viktig å navngi sketsjer, plan, linjer og komponenter på en intuitiv måte. For å sikre at komponenter grupperes på en hensiktsmessig måte, er det utviklet et nummersystem for navngivning av komponenter.

Negative erfaringer

Så langt har de ikke klart å bygge noen modeller som oppfører seg helt korrekt ved endring av parametere. Dette gjelder for de små detaljene som eksempelvis bolteforbindelser. Dermed har de måtte legge til slike detaljer manuelt før det kan lages produksjonstegninger av lukene.

I noen tilfeller har gamle modeller (ikke parametriserte modeller) blitt skalert for å gjenbruke denne i et annet prosjekt. Dette gjelder spesielt for turbiner hvor det er behov for 3D-modeller på et tidlig stadium i et prosjekt. Ved en slik tilnærming vil hele modellen skaleres etter forholdet som definere. Dette samsvarer ikke nødvendigvis med det reelle behovet for tilpasning. Noen ganger ønsker en eksempelvis kun å justere høyden og ikke bredden.

Bedriftens nytteverdi av parametriserte modeller

Et viktig bruksområde er å hurtig kunne generere en 3D-modell på tidlig-stadiet i et prosjekt. Denne modellen kan også nyttes som et utgangspunkt for konstruksjonsfasen, men i utgangspunktet er modellens behov for detaljer lavt. Noe av bakgrunnen for dette ambisjonsnivået er at det har vist seg vanskelig å få til flere av detaljene.

Videre er det behov for noen solide modeller som kun skal vise geometri. Dette gjelder for de konstruksjonene som Multiconsult ikke designer selv. Eksempelvis turbiner. Her har utfordringen vært å parametrisere modeller med organisk utforming.

B.2 Modelleringsprogram for organisk utformet geometri

CAESES – CAD for automated shape optimization

CAESES er et fleksibelt DAK basert modelleringsprogram for et raskt og robust design studie sammen med analyseverktøy. Integrerte muligheter for prosessautomatisering og geometrioptimalisering i et designverktøy[1)].

Om CAESES

Dette er altså et modelleringsprogram som har spesialisert seg for automatisk design, hvor en til og med skal kunne bygge modeller basert på scripting alene om det er ønskelig[2)]. Dette vil spare prosessorkraft i en optimaliseringsprosess. Disse faktorene gjør at CAESES egner seg svært godt til å integreres i en automatisk optimaliserings-*loop* og kan brukes i kombinasjon med eksisterende automatiseringsverktøy som eksempelvis Dakota eller OptiSlang. Da vil CAESES kjøre i bakgrunnen for å genere geometri.

Programvaren har spesialisert seg for å designe komponenter som er utsatt for strømninger, da både utvendige-innvendige strømninger[2)]. Dette kan eksempelvis være utsiden av en bil, bauen på en båt, eller turbiner.

Organisk geometri

CAESES sin modellering baserer seg heller ikke ene og alene på parametere slik tradisjonelle DAK programmer gjør[2)]. Det er noe av det som gjør den fleksibel og godt egnet for å designe organisk utformet geometri. I tillegg har den et kraftig verktøy for å manipulere overflater som de kaller *meta surface*[3)]. Generelt har de strukket seg lengre for å lage verktøy for å effektivt designe komplekse flater.

Meta surface lages i utgangspunktet som en *sweep*, men deretter kan en legge til polynomiske likninger for å ytterligere definere overflaten. Vider kan disse likningene kobles sammen med parametere som styrer likningen og utformingen.

Resultater

Programvaren ble testet i bruk gjennom å utføre noen grunnleggende torturials. Det opplevdes som noe krevende da dette er en helt ny måte å modellere på, men der er mye tilgjengelig informasjon og derfor vurderes det til at dette er overkommelig. Rapporten fra denne testen ligger i sin helet i vedlegg C, «CAESES og metasurface».

1) https://www.caeses.com
2) https://www.caeses.com/products/caeses/
3) https://www.caeses.com/products/caeses/geometry-modeling-for-variation/

C. Forsøksrapporter

C.1 Strategi for komponent: Horizontal

Strategi for komponenter

Delphis´ Horizontal Modeling

Mål:

Prøve strategien i praksis og teste robusteheten til en komponent bygget etter denne strategien

Beskrivelse:

Dette er strategi som baserer seg på en flat struktur slik at sannsynligheten for at *features* ikke klarer å generere er liten. I denne strategien er det ikke lov å referere til noe annet en plan, linjer og punkter. Avrundinger *(fillets)* og skråkanter *(chamfer)* som er avhengig av linjer på modellen skal samles til slutt. Det skal altså ikke være noen avhengigheter på tvers av strukturen til komponenten.

Framgangsmåte:

Det ble designet et stempelhode etter denne strategien. Ønsket en komponent med noe kompleks geometri for å teste strategien best mulig. Høyden er satt til å være konstant så det som ble parametrisert var alle dimensjoner relatert til omkretsen. Så snart alle avhengigheter var analysert fungerte det fint å oppdatere modellen. Det var et lite problem med en bue, denne skulle styre ett *cut* fra sida. Denne slo seg vrang da diameteren ble økt. Det fordi at den sto fast på 40mm. Hvis man gjorde denne *driven* gikk bare kuttet opp i hullet til pinnen. Letteste løsning ville vært å tvinge en fast avstand til hullet, men det er ikke lov i horizontal. Klarte ikke finne noen skalering for bua som fungerte tilfredsstillende. Derimot det som fungerte var å lage en strek, parallell med den ene aksen, som kunne definere en tilfredsstillende høyde på bua. Det har blitt ganske klart i arbeidet med denne modellen at basert på hva man vil oppnå så vil man lage skissene på forskjellig vis.

Fordeler:	**Ulemper:**
Flat struktur Enkelt og intuitivt *design tree* Enkel metode	Lett for at features ikke henger med i oppdatering av parametre da de er uavhengige av hverandre*
Annet:	
Oppleves som unøvendig rigid med hensyn på referring til eksisterende geometri. Dermed stilles det ekstra krav til å avdekke avhengigheter slik at modellen oppdatere seg i riktig rekkefølge. Siden ingenting er navngitt er det heller ingen lett sak å ta opp igjen modeller for videre endringer uten at den er styrt fra et sett med parametre.	

**Hvis jobben gjøres grundig mtp parametrisering er det ikke opplevd noen problemer rundt dette.*

C.2 Strategi for komponent: Explicit

Strategi komponenter

Explicit Reference Modeling

Mål:

Prøve strategien i praksis og teste robusteheten til en komponent bygget etter denne strategien

Beskrivelse:

Explicit Reference Modeling baserer seg på å å klassifisere alle *features* i to klasser. Klasse 1 og klasse 2. Hvor klasse 1 defineres ut ifra referanse geometri mens klasse 2 defineres ut i fra allerede eksisterende deler på modellen. Essensen i strategien er å begrense klasse 2, men hvis den må brukes skal det tilstrebes å referere til noe som ligger nærmest mulig i Design tree.

Framgangsmåte:

Det ble designet et stempelhode etter denne strategien. Ønsket en komponent med noe kompleks geometri for å teste strategien best mulig. Høyden er satt til å være konstant så det som ble parametrisert var alle dimensjoner relatert til omkretsen. Så snart alle avhengigheter var analysert fungerte det fint å oppdatere modellen. I denne strategien er det litt større rom for avhengigheter mellom *features* enn i *Horizontal.* Derfor satte jeg denne gangen buen i det horisontale kuttet til en gitt avstand fra senter på hullet over. Nå fikk jeg ingen problemer med at disse to *features* gikk over hverandre.

Fordeler:	**Ulemper:**
Enkel å forholde seg til	Begrensede muligheter for referering til geometri
Annet:	
Bare denne lille lettelsen i å kunne referere til eksisterende geometri gjorde det en del lettere å bygge parametriserte modeller.	

C.3 Strategi for komponent: Resilient

Strategi komponenter

Resilient Modeling Strategy

Mål:

Prøve strategien i praksis og teste robusteheten til en komponent bygget etter denne strategien

Beskrivelse:

Denne strategien blir streng navngivning og gruppering av *features* implementeret. Man deler opp slik at det kategoriseres etter viktighetsgrad. Det som er viktigs lages først og det kosmetiske lages til slutt. På grunn av hierarkiet som blir laget av grupperingen så trenger man ingen regler for hvordan ting skal modelleres så lenge grupperingen blir overholdt.

Framgangsmåte:

Det ble designet et stempelhode etter denne strategien. Ønsket en komponent med noe kompleks geometri for å teste strategien best mulig. Høyden er satt til å være konstant så det som ble parametrisert var alle dimensjoner relatert til omkretsen. Så snart alle avhengigheter var analysert fungerte det fint å oppdatere modellen. Selv om det er tillatt å referere til eksisterende geometri er det regler for hvilken type *features* som kan referere til hverandre. Det er ikke lov å refere internt i de gruppene med lavt tall. Derfor måtte det lages en hjelpelinje definert ut fra aksen for å klare å definere buen som også er omtalt i de to andre strategiene. Kunne eventuelt gått for å projisere bunnen på stempelet, men den og aksen er samme sted så derfor falt valget på aksen pga høyere opp i kategoristrukturen.

Fordeler:	**Ulemper:**
Veldig oversiktlig. Godt verktøy for identifisering av viktige *features.*	Krever noe organisering på forhånd.
Annet:	
Finner all informasjon en trenger for å kunne bruke metoden på RMS sin hjemmeside (learnrms.com). På grunn av at en kan refere til eksisterende geometri var det denne metoden som ga færrest variabler som måtte defineres for å kunne parametrisere modellen.	

C.4 Organisk utformet geometri: Loft

Organisk Geometri

Loft

Mål:

Test av hvordan bruke *loft* for å modellere organisk utformet geometri

Beskrivelse:

Loft-funksjonen tar inn en eller flere skisser sammen med hjelpelinjer og lager en geometri ut ifra det. Denne metoden gjør at man kan lage modeller med varierende tverrsnitt og flytende overganger mellom disse tverrsnittene. .

Framgangsmåte:

En form for vannvei ble modellert. Det ble laget to skisser. Et rektangulært tverrsnitt og et sirkulært tverrsnitt. Det ble definert ei linje mellom disse også. Ved å ha to parametre, en for høydeforskjellen og en for lengde mellom skissene, kunne man variere og teste hvordan modellen reagerte på endringer. Hjelpelinja var en spline generert av Inventor basert på funker man setter. Man kunne også brukt en liknings kurve hvis man har noe matematisk definert. Eller man kunne droppet hjelpelinja og latt Inventor ta korteste rute mellom de to skissene.

Fordeler:	**Ulemper:**
Strømlinjeformede modeller Svært god kontroll på tverrsnittet i hvert punkt	Komplisert med mange skisser
Annet:	

C.5 Organisk utformet geometri: Sweep

Organisk geometri

Sweep

Mål:

Test av hvordan man kan bruke *sweep* for å modellere sneglehus

Beskrivelse:

For å lage en *sweep* vil en vanligvis bruk en tverrsnittskisse og en føringslinje for å styre retningen skissen skal føres langs. I tillegg kan en bruke en hjelpelinje for å skalere tverrsnittet. Her kan en velge om skal skalere i en eller to retninger.

Framgangsmåte:

Tegnet noe som kunne minne om et inntak(les: sirkulært tversnitt med litt ekstra). Opprettet ei føringslinje som dette tversnittet skulle følge. I funksjonen “liknings linjer” slik at jeg kunne styre retning på linjen ved hjelp av likninger for å få kontroll på hvilken retning linjen tar.

Fordeler:	**Ulemper:**
Slipper mange skisser for tverrsnitt	Kan være krevende å parametrisere Mindre kontroll på tversnittet på en gitt plassering
Annet:	
En fin metode å bruke så lenge tverrsnittet ikke skal ha noen brå endringer.	

C.6 Organisk utformet geometri: CAESES og metasurface

Organisk geometri

CAESES og meta surfaces

Mål:

Finne ut om meta surfaces kan bidra til å lage organiske former og hvordan programvarer CAESES er å bruke

Beskrivelse:

CAESES er et program som ligner på Inventor/SolidWorks men som i større grad er baser på former i stedet for faste parametre. Kan minne om programmvarene Maya og Blender. Det får et mer kunstnerisk preg enn den mekaniske måten Inventor modellere på. CAESES har spesielt en type som heter *meta surface*. Dette er en type overflate som kan manipulerers på en spesiell måte. Den lages i utgangspunktet som en *sweep*. Men man kan definere hjelpe funksjoner(polynomiske likninger) i form av linjer i planet. Disse funksjonene kobler man med parametre som definerer overflaten. Variabelen til denne funksjonen er gjerne avstand eller grad(hvis noe går om en akse). Dette gjør at man får en sweep som endrer fasong langs med hjelpelinja.

Framgangsmåte:

Utførte grunnleggende tutorials.

Fordeler:	**Ulemper:**
Gode parametriske modeller klare for ANSYS Spesialisert for optimaliseringsproblemer	Mye nytt å sette seg inn i pga ganske annet enn inventor Vanskelig å få noe annet enn dummy modeller inn i Inventor
Annet:	
Siden det er en annen programvare kreves en del timer for å komme inn i bruken og enda flere for å utnytte måten man modellerer på i dette programmet.	

C.7 Strategi for sammenstilling: Top Down: Single Skeleton

Parametrisering av sammenstilling

Top Down Design: Singel Skjelett

Mål:

Test av robustheten til en sammenstilling bygget etter *Single Skeleton* prinsippet

Beskrivelse:

Singel skjeletter er en måte å organisere du ulike skissene som til sammen definere en sammenstilling. En starter med den overordnede skissen og deler videre denne opp i mindre og mindre skisser. Dermed vil de underordnede skissene bli drevet av den overordnede. Alt dette gjøres i en part-fil. Inventor har en funksjon som heter *make part*, denne gjør at skissene kan skilles inn i egne filer og automatisk være en del av sammenstillingen. Nedsiden er at alle delene som blir pushet fra et skjelett blir satt til *grounded* i sammenstillingen. Noe som gjør det vanskelig å lage dynamiske deler.

Framgangsmåte:

Benyttet luke modellen. Satte opp et skjelett hvor alle komponentene er tegnet inn som 2D skisser. Brukte *make part* til å lage separate filer til delene og en sammenstilling til dem. Tester for endringer i etterkant viste at modellen oppførte seg som den skulle allerede ved første forsøk.

Fordeler:	**Ulemper:**
Robust Enkelt å gjøre endringer i etterkant	Må ha oversikt over alle komponenter på forhånd. Uten struktur -> vanskelig
Annet:	

C.8 Strategi for sammenstilling: Top Down; Multi Skeleton

Parametrisering av sammenstilling

Top Down Design: Multi Skeleton

Mål:

Test av robustheten til en sammenstilling bygget etter *Multi Skeleton* prinsippet

Beskrivelse:

Multi Skeleton er en videreutvikling av skjelett-metoden ved at en definere flere ulike skjelett for hvert lag som inneholder hver sin type informasjon. Inventor har en egen funksjon hvor en kan velge hvilke parametre en tar med seg til neste nivå. *Make Part* funksjonen lar en velge fra ei liste hvilke parametre, plan og linjer man vil overføre videre. Deretter velger man navn på det nye skjelettet og om det skal lages en ny sammenstilling med dette skjelettet i. Det er denne funksjonen som lager PS for neste nivå. En slik funksjon er ment for store sammenstillinger med store mengder informasjon. Modellen brukt i dette forsøket er egentlig litt lite til at det er nødvendig med *multi-skeleton*, men dette har bare vært for å teste hvordan det oppleves å bruke denne.

Framgangsmåte:

Jeg bygget et overordnet skjelett for hele glideluka hvor jeg la inn all informasjon fra beregningsgrunnlaget. Her ble det også ble laget *surface-features* av lukebladet og føringene. Deretter brukte jeg *Make Part*-kommandoen til å ta med meg de nødvendige parametrene og geometrien til neste nivå. Eksempelvis var det ikke behov for at skjelettet til luken inneholdt noe informasjon om parametrene til føringene. Siden denne informasjonen allerede ligger i mor-skjelettet. Mor-skjelettet ble nok litt for detaljert. Det kunne vært spart litt på detaljene, da dette skal representere LS og DS. De overflatene(surface) kunne vært mer som firkanter. Så lager man heller detaljene på neste nivå. Men siden dette var en enkel sammenstilling var det mer naturlig å lage detaljert på et høyere nivå, for å holde på oversiktligheten.

Fordeler:	**Ulemper:**
Oversiktlig Alt på et sted	Krever modning Alt på et sted(hvis det ikke lages med omhu)
Annet:	
Vanskeligste er å se for seg hvordan man kan legge ut hele sammenstillinga i 2D. Krever god forståelse for hva som skal modelleres siden en må ha en klar tanke på hvor alle komponenters skal og hvilken informasjon som er nødvednig hvor. Bevegelige deler kan være noe mer krevende å lykkes med.	

C.9 Strategi for sammenstilling: Top Down: Multibody

Parametrisering av sammenstilling

Top Down Design: Multibody

Mål:

Test av robusthet til en sammenstilling bygget etter *multibody*

Beskrivelse:

Det som skiller en *Multibody* fra andre sammenstillinger er at den lages i en *part*-fil som inneholder flere solider. På denne måten behøver en ikke definere fastholdninger mellom de ulike komponentene siden dette defineres ut fra hvordan solidene er plassert i forhold til hverandre. Dette betyr også at en sammenstilling av *Multibody* ikke kan inneholde noen bevegelige deler.

Framgangsmåte:

For å teste denne strategien ble det valgt å bygge opp lukebladet med sine lister. Jeg startet med å lage grove dimensjoner for *multibodypart*-filen, for deretter å gjøre finarbeidet i hver enkeltfil som ble generert. Alle delene ble konstruert i mor-fila, for så å bli overført til hver sin respektive datter-fil. Dette bidrar til å holde en kobling mellom mor-datter. Sammenstillinga som denne luka skulle inn i var en føring laget med skjellet. Dimensjoner som ble overført fra master skjelettet var; høyde og bredde plan, tykkelses plan, glidelistkontakt plan, paknings kontakt plan. Disse parametrene kommer fra dokumentasjons beregningene.

Fordeler:	**Ulemper:**
Hurtig, man jobber bare i en fil til å begynne med Intuitivt	Kan gå i fella om å referere til eksisterende geometri på steder det ikke er lurt
Annet:	
Av en eller annen grunn var det vanskelig å endre materialer i etterkant. Det er ikke noe krav at delene må være i samme plan eller være i nærheten av hverandre. Fikk problemer med å utvide høyden, viste seg at problemet var en skisse som ikke var fult definert. For de som trives best med *Botton-up* vil dette kunne oppleves som en intuitiv måte å modellere etter *Top-down.*	

C.10 Strategi for sammenstilling: Bottom-Up

Parametrisering av sammenstilling

Bottom Up

Mål:

Test av robustheten til en sammenstilling bygget etter *Bottom Up* prinsippet

Beskrivelse:

I en *Bottom Up* er hver komponent bygget for seg selv, i dette tilfellet etter horisontal modelling, for deretter å settes inn i en sammenstilling som skal parametriseres. Dette betyr at i utgangspunktet er det ingen avhengigheter mellom hver komponent. Når sammenstillingen skal parametriseres må man legge inn iLogic slik at styringsparametre fra sammenstillingen blir ført tilbake til hver enkelt komponent. Disse parametrene kan også hentes fra Excel. Det finnes egentlig en funksjon for å linke komponenter til sammenstillinger, men dersom en liker det til en komponent i gjeldende sammenstilling blir det en sirkelavhengighet. Hele modellen skal styres av to parametre: Høyde og bredde.

Framgansmåte:

I denne sammenstillingen ble plasseringen av hver komponent definert ut fra origo. Det viste seg at hver komponent måtte ha minst tre fastholdninger for å få en fullt definert modell. Dette fordi at en fastholdning låser to frihetsgrader og hver komponent har seks frihetsgrader.

Fordeler:	**Ulemper:**
Robust (når korrekt utført) Lettvint å lage delene Kan bruke allerede eksisterende deler	Blir fort mye å sette sammen i sammenstilling Fort å gjøre feil med fastholdningene
Annet:	
Parameteroverføringene må programmeres Speiling av komponenter kopierte ikke fastholdningene. Utgjør ingen klar forskjell at komponentene er bygget med Horizontal	

C.11 Strategi for sammenstilling: Middle-Out

Parametrisering av sammenstilling

Middle-Out

Mål:

Test av robusthet til sammenstilling bygget etter *Middel-Out* prinsippet

Beskrivelse:

Middel-Out brukes når du har en komponent eller en sub-sammenstilling som er definerende for resten av sammenstillingen. Inventor støtter at man kan projisere eksisterende geometri på andre plan, og på denne måten kan omliggende komponenter drives av den du projisere fra.

Framgangsmåte:

Det ble laget en aksel med et nav. Akselen blir satt som fullt fastlåst, siden det er den som er den drivende komponenten her. Navet ble laget i en egen fil og deretter lastet inn i en sammenstilling sammen med akselen. Kan settes sammen som om det var bottom up, unntatt eventuelle fastlåsninger som hindrer ønsket endringsmønster. En annen mulighet er å lage en top down skjelett som styres av projisering. For å drive navstørrelsen etter akselen måtte man passe på at skissen til navet var fullt definert som betyr ingen manglende dimensjoner og skissefastlåsninger. Hvis skissen ikke er fullt definert vil ikke projiseringsfunksjonen gi en dynamisk projeksjon. Dvs. projeksjonen vil ikke endre seg når komponenten endrer seg. Når projiseringa var lagt inn i skissa var det bare å definere at indre diameter i navet ble lik projeksjonen. Slik at navet kunne følge akseldiameteren.

Fordeler:	**Ulemper:**
Adaptivitet	Fastlåsninger
Annet:	

C.12 Inventor: iPart/iAssembly

Inventor

iPart og iAssembly

Mål:

Kartlegge brukervennligheten til *iPart* og *iAssembly*

Beskrivelse:

Disse to verktøyene tillater deg å lage en komponent/sammenstilling, for deretter å lage et bibliotek av tilsvarende komponent/sammenstilling bare ved å oppdatere parametrene i en tabell. Under *Manage -> Author -> Create Ipart* får man opp et vindu hvor man definerer delen ut fra en tabell. Hvor radene representerer forskjellige iterasjoner av deler, mens kolonnene bortover spesifiserer hva som skal være forskjellene i parametre.

Framgangsmåte:

Jeg laget en vinkel som ble stivet av ved hjelp av ribber. Det ble tenkt et scenario hvor man har flere ribber, slik at den kan tåle større laster. For å teste ble det laget fem forskjellige versjoner. Disse hadde alle samme romlige dimensjoner men det ble variert hvor mange instanser det var av ribben. Dette er da en iAssembly, men det ville vært tilsvarende prosess for å bygge opp biblioteket med ulike varianter av komponent/sammenstilling.

Fordeler:	**Ulemper:**
Hurtig metode for å lage standardiserte deler (deler fra kataloger). Intuitivt	Kan ikke brukes til å drive parametrene til en sammenstilling.
Annet:	
Man må ha et bevisst forhold til hvilken rekkefølge man modellerer i. Dette fordi at Inventor organiserer rekkefølgen på features kronologisk. En kan ikke bruke en del laget i iPart eller iAssembly for å drive en Middle-Out. (https://forums.autodesk.com/t5/inventor-forum/multibody-ipart-generating-files-for-use-in-assembly/td-p/9034006(28.04.2020)).	

C.13 Inventor: API

Inventor

API

Mål:

Kartlegge mulighetsrommet i Inventor sin API

Beskrivelse:

Inventor har en åpen API hvor en kan nå alle dens funksjoner, med unntak av analyse miljøet. Dette betyr faktisk at en kan både lage tilleggsapplikasjoner for å styre Inventor, automatisere enkelte prosesser eller lage rapporter for dokumentasjonsarbeid. Blant språkene en kan bruke er Visual Basic, C# eller Python.

Framgangsmåte:

Det ble laget en applikasjon for å teste parameteroverføring. Programmert i C#. Det ble testet ut en manuell og en automatisert måte å gjøre det på. Den manuelle måten gikk ut på at man valgte hvilke parametere man skulle linke. Den andre brukte et xml-dokument til å gjøre denne jobben.

Fordeler:	**Ulemper:**
Full kontroll	Avhengig av gode programmeringskunnskaper(objektorientert)
Annet:	
Autodesk har lagt ut dokumentasjon for alle funksjoner gratis på nett. I tillegg er det et stort miljø av ingeniører som lager egne applikasjoner. Dette betyr at det er mye ressurser på nettet og en har høy sannsynlighet for finne akkurat det en trenger ved å gjennomføre et internettsøk.	

C.14 Inventor: Program for test av parametre

Inventor

Program for test av parametre

Mål:

Lage et program som kan teste ved hvilke parametre modellen bryter sammen

Beskrivelse:

Når en bygger en parametrisert modell er det nyttig å ha en oversikt over hvilke parametre som er gyldig for nettopp denne modellen. Altså vite ved hvilke kombinasjoner av parametre modellen vil bryte sammen.

Framgangsmåte:

Jeg laget en egen applikasjon som kunne teste en hel serie av parametre. Det ble utnyttet at et kall på oppdateringsfunksjonen returnerte verdien SANN eller USANN, hvor USANN er der hvor modellen bryter sammen. Applikasjon testet alle verdier mellom høyde=1400-1500mm og bredde=1400-1500mm, og alle ulike kombinasjoner av disse to. For hvert steg økte verdien med 5mm. Applikasjonen ble laget i Visual Studio og programmert i C#..

Fordeler:	**Ulemper:**
Ingen problemer Kan brukes uavhengig av modell	Krever kunnskaper om objekorientertprogrammering
Annet:	

C.15 Beregningsprogram: Mathcad

Beregningsprogram

Mathcad

Mål:

Kartlegge brukervennligheten og funksjonalitet til Mathcad

Beskrivelse:

Mathcad er en blanding av excel og håndskrift. Derfor er det veldig oversiktlig og kraftfullt regneverktøy.

Framgangsmåte:

Det settes opp et regneark med utgangspunkt i tilsendte beregninger fra Multiconsult Med utgangspunkt i de tilsendte beregningene, snus dette om på beregningene i den hensikt å bruke færrest mulig inngangsparametre. Her blir HVIS-funksjoner nyttige for å kunne hente ut parametre basert på valg. Eksempelvis at en får riktig flytegrense og E-modul ut fra materialvalget som gjøres. Det blir også gjort forsøk på å bruke SOLVE-boks for å gjør små optimaliseringsprosesser underveis, men dette lykkes ikke.

Fordeler:	**Ulemper:**
Velegnet for dokumentasjon Kan brukes til å vise Design Intent	
Annet:	
Det viser seg at det krever svært liten tid for å håndtere de grunnleggende funksjonene i Mathcad. Allikevel er det mange nyttige verktøy som krever noe mer erfaring for å beherske.	

C.16 Beregningsprogram: Mathcad kommunikasjon

Mathcad

Import og eksport av parametre

Mål:

Utforske muligheter for kommunikasjon mellom Mathcad og andre programmer

Beskrivelse:

Mathcad har innebygd funksjon for å håndtere excel import. Det man gjør da er at man imbedder et excel ark i mathcad arket. Det finnes forøvrig ingen innebygd måte å eksportere til excel. Mathcad har også en egen API som kan brukes for kontroll av mathcad regneark. Bruk av API krever programmering i C#.

Framgangsmåte:

For å kunne gjøre bruken av mathcad mest mulig automatisert eller å få overført til andre programmer er det nødvendig med lettvint import/eksport. Hvor test av en egenkomponert tredjeparts applikasjon ble testet for import og eksport fra mathcad. Samt testet de innebygde excel funksjonene.

Fordeler:	**Ulemper:**
Fungerer bra	Krever kunnskap om objektorientert programmering(ikke Excel biten). Excel kan bare overføre til Mathcad og ikke hente fra Mathcad.
Annet:	
Mathcad har ingen gratis dokumentasjon og kodesnutter. Siden det eksisterer en mathcad API er det mulig å lage en plugin til excel hvis dette skulle vært avgjørende. Hvis man skulle hatt lyst til å benytte mathcad for å lettere vise fram beregninger så fungerer det fint å eksportere fra excel.	

C.17 Beregningsprogram: Jupyter Notebook

Beregningsprogram

Jupyter Notebook (JpN)

Mål:

Kartlegge brukervennligheten og funksjonalitet til JpN

Beskrivelse:

En programmeringsbasert notatbok. Man kan benytte flere ulike språk for å programmere i JpN og det kan utføres online eller programvaren kan lastes ned på PC.

Framgangsmåte:

Det settes opp et regneark med utgangspunkt i tilsendte beregninger fra Multiconsult I dette forsøket er det valgt å bruke Python da dette er et av de mer vanlige språkene som en gjerne starter med å lære. Ved å ta utgangspunkt i de tilsendte beregningene fra Multiconsult ble det laget et regneark i JpN. Dette tok en del tid i starten, men siden det er mye lett tilgjengelige ressurser på nett for å skrive i Python, var det godt mulig å lage et regneark selv for en som er uerfaren.

Fordeler:	**Ulemper:**
Store tilpasning muligheter	Må ha programmeringskunnskaper
Annet:	
Med de riktige tilleggene og fin programmering kan det bli et bra verktøy for å vise Design Intent og dokumentere beregninger. Det må lastes ned et eget script for å kunne bruke enheter.	

C.18 Beregningsprogram: Excel

Beregningsprogrammer

Excel

Mål:

Kartlegge brukervennligheten og funksjonalitet til Excel

Beskrivelse:

Framgangsmåte:

Det settes opp et regneark med utgangspunkt i tilsendte beregninger fra Multiconsult. Med utgangspunkt i de tilsendte beregningene snus dette om på for å få færrest mulig inngangsparametre. Her blir HVIS-funksjoner nyttige for å kunne hente ut parametre basert på valg. Eksempelvis at en får riktig flytegrense og E-modul ut fra materialvalget som gjøres. Dette regnearket ble satt opp etter det vi laget i Mathcad. Da en kunne ta utgangspunkt i denne var det relativt enkelt å sette opp formler.

Fordeler:	**Ulemper:**
Lettvint å sette opp store mengder tall Symbolet til verdiene er i en egen celle. Kan definerer navn til celleområder for lettere å navigere i alle tallene.	Lite egnet for dokumentasjon av beregninger Lite egnet til å vise Design Intent Kronglete å finne årsak til problemer med utregning
Annet:	
En fin funksjon å benytte i Excel når det kommer til beregninger er: *Definer Navn(Define Range)*. Denne lar deg opprette variabelnavn som kan brukes i formler. Dette bidrar stort til lesbarheten til formler i excel.	

Kilde Definer navn:
https://support.office.com/nb-no/article/definere-og-bruke-navn-i-formler-4d0f13ac-53b7-422e-afd2-abd7ff379c64

C.19 Analyseprogram: Inventor

Analyseprogram

Inventor

Mål:

Teste funksjonaliteten til Inventor sin FEM-analyse

Beskrivelse:

Inventor har sitt eget innebygde FEM-miljø. Denne funksjonen kjøres i samme fil som modelleringen, som betyr at slipper import eller eksport av filen. Derimot har den kun noen få laster en kan bruke og færre muligheter til justere meshing sammenlignet med eksempelvis ANSYS.

Framgangsmåte:

Laget en vinkelbrakett med noen ribber på. Kunne raskt se at det var vanskelig å definere en last på kun en liten del av flaten. Endte opp med å lage en kloss i den størrelsen jeg ville ha. Dermed ble vinkelen holdt fast i den horisontale flaten så ble lasten satt på den vertikale flaten.

Fordeler:	**Ulemper:**
Simpelt, fort gjort å sette opp en enkel analyse	Støtter ikke interne spenninger. Tverrsnitt Ikke tilgjengelig via API
Annet:	
Inventor støtter ikke intern spenningsanalyse. Autodesk har ikke gjort analyse miljøet tilgjengelig gjennom API. Dette hindrer i stor grad å automatisere analyser.	

C.20 Analyseprogram: Solidworks

Analyseprogram

SolidWorks og Excel integrasjon

Mål:

Lage en add-in til excel som kommunisere med solidworks modellering og simulering

Beskrivelse:

Både SolidWorks og Excel har en API som kommuniserer via COM. Dette gjør at programmene kan prate sammen gjennom kode. Hvis man bruker excel som beregning og dokumentasjon har en all data der. Her skal det sees på muligheten for å la Excel styre modellering og analyse ved å kun bruke en knapp i Excel.

Framgangsmåte:

Add-in laget her er av typen VSTO COM og programmert i C# ved hjelp av Visual Studio sin Excel Add-in template. Det viste seg at ikke alle distribusjoner av Office støtter dette lenger. Av office 365 må man ha “Klikk og bruk” og ikke “Microsoft Store”: Forskjellen er at du må gå på office.com og laste ned, ikke gjennom microsoft store. De utgivelsene av Office 20xx er allerede “Klikk og bruk”. Gjennom et par kjappe videoer på youtube ble det satt opp en ny fane i excel. Fra denne fanen ble det lagt til knapper slik at åpning av fil, endring av parametre, starting av simulering og prosessering av resultater kunne gjøres stegvis. Siden dette er programmert så er det ingen problemer med å gjøre dette mer automatisert eller generelt. Siden både Excel og Solidworks sine API’er er veldokumenterte og lett tilgjengelig, så er det bare snakk om tid før man får til den funksjonaliteten man ønsker. Gitt at man har kunnskap om objektorientert programmering i et språk støttet av API’ene. Det ble ikke satt opp noen optimalisering, da dette ville krevet enda mer tid i en travel sluttfase. Men alle kravene i funksjonalitet for å få det til er testet ut. Så igjen, kun et spørsmål om tid.

Fordeler:	**Ulemper:**
Man får det slik man vil To programmer ordner beregning, modellering og simulering.	Krever mye tålmodighet å lære Tidkrevende første gang
Annet:	
Hvis man legger ned tid i starten for å lage et bra oppsett på bruk av API’ene, så vil man kunne legge til mer og mer funksjonalitet etterhvert som andre prosjekter krever mer av programmet.	

https://office365.uservoice.com/forums/264636-general/suggestions/35176501-office-365-store-version-does-not-recognize-excel

C.21 Datadeling: Mathcad og Inventor

Overføring av parametre

Kommunikasjon mellom Mathcad og Inventor

Mål:

Overføre parametre mellom Mathcad og Inventor for å kunne skape en digital tvilling

Beskrivelse:

Det finnes ingen integrerte funksjoner for kommunikasjon mellom Inventor og Mathcad. Antakeligvis på grunn av at utgiveren av Mathcad har en konkurrerende programvare. Derfor må det lages en egen applikasjon for kommunikasjon mellom disse. Både Inventor og Mathcad har en API som er åpen for at det kan programmeres i C#.

Framgangsmåte:

Det ble laget to ulike løsninger for å overføre parametre. Begge løsningene ble programmert i C#. I den første løsningen plukket en hvilke parametere en ville linke sammen for hver gang en parameter skulle overføres. En kunne også velge hvilken vei denne parameteren skulle overføres. Dette var en svært manuell metode som tok mye tid for hver gang. Derfor forsøkte en på nytt med en ny metode. Denne gangen ble en fil(XML) benyttet til å definere hvilke parametere som er knyttet sammen i de ulike programmene, slik at da en ønsket å oppdatere ble alle parametrene automatisk oppdatert. Også her kunne overføringen gå begge veier.

Fordeler:	**Ulemper:**
Full kontroll	Krever kunnskap om objektorientert programmering
Annet:	
En annen løsning er å lage alle parameternavnene ut fra aliasene i mathcad. For deretter å opprette nye navn i inventor. og de neste gange oppdatere man bare dem. Da vil man slippe det manuelle mellomleddet. I alle fall etter at de nye parametrene er satt inn i modellen. Ved hjelp av xml-filer kan man lagre forskjellige mappinger for overføring av parametre. Slik at man redigerer xml dokumentet og lar tredjepartsprogrammet ta seg av å overføre parametre i henhold til det som er spesifisert.	

C.22 Datadeling: Jupyter Notebook og Inventor

Overføring av parametre

Kommunikasjon mellom Jupyter Notebook (JpN) og Inventor

Mål:

Overføre parametre fra Jupyter Notebook til Inventor for å kunne skape en digital tvilling

Beskrivelse:

Inventor sin API er åpen for å kunne programmere i Python, som er samme språk JpN nytter. Syntaksen vil være lik som den for Visual Basic.

Framgangsmåte:

Det ble laget en test script for å se om et Pythonscript klarte å endre på parametrene i en 3D modell. Testen ble laget som en selvstendig python script gjennom Visual Studio. Ved hjelp av en kode på ca 10 linjer kunne en åpne et bestemt dokument, lese av en verdi, endre verdi og oppdatere dokumentet med den nye verdien.

Fordeler:	**Ulemper:**
Relativt lett Python og JpN er gratis Gjøre det så vanskelig eller lett man vil	Trenger kunnskaper om objektorientert programmering Lite ferdig kodeeksempler skrevet i Python
Annet:	
Syntaksen til Inventor API for Python er lik den for VBA. Python oppleves som noe mer utfordrende grunnet mindre dokumentasjon knyttet til Inventor sin API.	

C.23 Datadeling: Excel og Inventor

Overføring av parametre

Kommunikasjon mellom Excel og Inventor

Mål:

Overføre parametre mellom excel og inventor for å kunne skape en digital tvilling

Beskrivelse:

Det er et integrert verktøy i Inventor som står for kommunikasjon med Excel. “Link” knappen inne i parametervinduet eksportere fra Excel til Inventor, mens en iLogic regel leser informasjon fra Inventor og overfører til Excel.

Framgangsmåte:

Det ble laget en loft-feature i Inventor med to tverrsnittskisser og guide linje. Dimensjonene som styrer hvor langt unna hverandre disse skissene er ble parametrisert med tall fra Excel. Jeg gikk inn i parameter vinduet og valgte hvilken fil som skal leses. Deretter måtte jeg spesifisere hvilken kolonne den skulle lese fra. Det er da viktig at alle parametrene som skal overføres er satt opp etter hverandre som en liste nedover. Inventor ber om en start celle og leser deretter alle nedover i den kolonna. Den leser da variabelnavnet f.eks i A1 og tilhørende verdi i B1. Enheter legges inn i kolonne C. Disse må også ligge i det første arbeidsarket i arbeidsboka. For å eksportere parametre til Excel ble det nytett en kode i iLogic: “GoExcel.CellValue("C:\Users\morte\MasterFiler\Modeller\M_M_Originaler\Organisk\Parametre.xlsx", "Ark1","B1") = Length” Den er avhengig av at excel arket eksisterer men ikke er åpnet. Hvis den er åpnet vil man få problemer med at excel vil lagre kopier pga skrivebeskyttelse.

Fordeler:	**Ulemper:**
Sømløst, må bare oppdatere modell i Inventor etter endring i Excel Veldig rett fram for overføring til Excel	Kan ikke velge hvilket ark man leser fra arbeidsboka
Annet:	
Det er den kolonnen i det første arket i boka som leses da en laster fra Excel til Inventor.	

C.24 Datadeling: Overføring til ANSYS

ANSYS

Kommunikasjon

Mål:

Lage en applikasjon for import av parametre, modell og laster inn til ANSYS

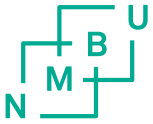
Beskrivelse:

Workflow extensions som kan programmeres i både python og C#. Videre har ANSYS en egen API for ACT. Dette er tillegg man kan lage for å øke funksjonaliteten til ANSYS. ANSYS har en CAD Configuration Manager som gjør at CAD modeller kan importeres direkte inn til ANSYS. Av innebygde funksjoner har ANSYS en Excel komponent som man kan koble opp mot parametersettet man bruker i et prosjekt. Dette gjør at informasjon kan flyte mellom programmene slik at de automatisk oppdateres etter å ha endret parametrene kun et sted.

Framgangsmåte:

Etter noen timer søk på internett ble det klart at å koble andre programmer sammen med ANSYS faller utenfor kunnskapene mine. Dersom det i det hele tatt er mulig å koble ANSYS med andre programmer. Det var ikke mye informasjon eller hjelp å oppdrive og det som ble funnet var utdatert informasjon. Derimot viste det seg at det var enkelt å lage tilleggsfunksjoner i ANSYS. Gjennom ACT kunne man enkelt legge til ekstra funksjonalitet. Siden et slikt ACT-tillegg kan programmeres i C# er det muligheter for at det kan lages knapper i WB for overføring av info til inventor/Mathcad. Dette er ikke testet grunnet mangel på tid. Importering av .ipt til ANSYS ble stoppet av at jeg ikke har korrekt lisens, men jeg fikk det til å fungere med step-fil. Fra man oppdaterer CAD modellen til ansys har et nytt resultat (gitt at ikke geometrien er så endret slik at ANSYS ikke finner at flatene) er det to knapper som må trykkes på. Den ene oppdatere geometri-fila, den andre starter analysene på nytt. Dette ble testet med en flow analyse og en statisk struktur analyse i samme prosjekt. Excel komponenten fungerer ved at man linker inn en fil. ANSYS lager da en kopi av fila og lagrer den ved resten av prosjekt filene. For at ANSYS skal kjenne igjen variabler er man nødt til å definere navn i Excel. Dette er Excel sin egne måte å lagre variabler på. Når disse er lagt inn i ANSYS velger man om det skal være input eller output. Input er fra Excel til ANSYS Og output motsatt.

Annet:
Tips og triks rundt excel komponent: https://www.padtinc.com/blog/workbench-and-excel-part-1-using-the-excel-component-system/



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway