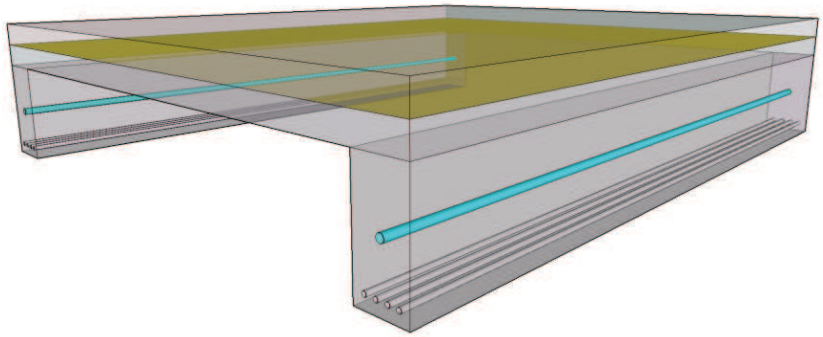
Förfinad metod för användning av BIM-modeller för strukturanalys och dimensionering av betongkonstruktioner



Robin Nerborg och Johan Olsson

Avdelningen för Konstruktionsteknik
Lunds Tekniska Högskola
Lunds Universitet, 2014

Rapport TVBK – 5232

Avdelningen för Konstruktionsteknik
Lunds Tekniska Högskola
Box 118
221 00 LUND

Division of Structural Engineering
Faculty of Engineering, LTH
P.O. Box 118
S-221 00 LUND
Sweden

Förfinad metod för användning av BIM-modeller för strukturanalys och dimensionering av betongkonstruktioner

Refined method for use of BIM models for structural analysis and design of concrete

Robin Nerborg och Johan Olsson

2014

Abstract

IFC is key enabler for data interoperability of BIM models. This enables reusing the geometry for structural analysis, thus saving modeling time. BIM models and the IFC format consists of solid elements, therefor the imported geometry will generate solid elements when meshed. When performing structural analysis with the finite element method on concrete structures the use of shell and beam elements is preferred. A method has been developed to import BIM models into BRIGADE/Plus generating shell and beam elements when useful. The method works quite well importing most elements in an efficient way, given that definitions in the BIM model are sufficient. The result is that the user will have to spend less time rebuilding the geometry. Often the design engineer has not given parts definitions suitable for the structural analysis. The import will then not suffice and the geometry in the structural analysis has to be reconstructed. When importing a BIM model for structural analysis, a feature that would be beneficial is the opportunity to choose simplification for geometrical objects.

Keywords: finite element method, reinforced concrete, BIM, IFC, structural analysis, solid, shell, beam, geometry

Rapport TVBK-5232
ISSN 0349-4969
ISRN: LUTVDG/TVBK-14/5232(69)

Examensarbete
Handledare: Johan Kölfors och Anders Vennerstrand, Scanscot Technology AB
Juni 2014

Sammanfattning

Det ökande användandet av BIM leder naturligt till att programvaror behöver utvecklade funktioner och nya möjligheter. Fördelen med BIM är att en konstruktions alla egenskaper kan samlas i en och samma modell. Rimligtvis bör modellens geometriska information även enkelt kunna användas vid en strukturanalys. Däremot innehåller en beräkningsmodell i regel en rad förenklingar av geometrin, för att på ett rimligt sätt spegla den verkliga strukturens beteende.

Många anläggningskonstruktioner, exempelvis broar, innehåller oftast en betydande del armerad betong, för vilken dimensioneringen utgår från snittkrafter. Då anläggningskonstruktioner ofta är stora och komplexa krävs det att beräkningsmodellen anpassas efter tillgänglig datorkraft och behov av resultat. För att beräkna snittkrafter ur en beräkningsmodell finns det klara fördelar att modellera med skal- och balkelement, vilka ger önskvärda resultat samtidigt som erforderlig datorkraft kraftigt minskas.

Scanscot Technologys strukturanalysprogram BRIGADE/Plus kan importera BIM-modeller sparade i det standardiserade och öppna filformatet IFC. Beräkningsmodellen kommer då att bestå av tredimensionella solida objekt. En förenkling av beräkningsmodellen till att bestå av balkar och skal innebär att den måste rekonstrueras från grunden. Inom ramen för detta arbete har en metod tagits fram där en BIM-modell som importeras via IFC-fil till stor del automatiskt genereras till en modell bestående av skal- och balkelement.

Metoden bygger vidare på befintlig funktion för import av IFC-filer i BRIGADE/Plus. När en IFC-fil är inläst kontrolleras byggnadselementens objektdefinitioner. Definierade balkar och pelare importeras som balkelement, medan plattor och bjälklag importeras som skalelement. Resterande element och de element som inte uppfyller vissa villkor importeras som solidelement precis som tidigare. Generering av balkelement görs efter given information om längd, orientering och tvärsnittsprofil lagrad i IFC-filen. Förenkling till skalelement sker efter två villkor; om de två största ytorna är lika stora och parallella. Resultatet är skalelement i tyngdpunktsplanet mellan de två största ytorna.

Den framtagna metodens villkor för bedömning av vilka objekt som kan importeras som skalelement har vid tester visat sig ge goda resultat. Vissa balkprofiler definieras inte på samma sätt i IFC som i BRIGADE/Plus, vilket gör att information kan gå förlorad. De element som importeras som skal- och balkelement representerar inte alltid den riktiga geometrin exakt, dock tillräckligt bra. En del avancerade geometrier genererar dock skalelement som inte är korrekta.

Ofta används BIM för att få en bra visuell byggnadsprototyp, där objektdefintionen för byggnadselementen är av mindre vikt. Denna definition är dock viktig för att ge rätt förutsättningar för metodens importering i BRIGADE/Plus. En modell med lämpliga definitioner kan spara beräkningsingenjören mycket tid medan en modell med blandad kvalitet kan ta längre tid att bygga om än en modell som importerats med enbart solidelement.

Summary

With increased use of BIM, the demand of features in utilised software will increase. The advantage of BIM is the easiness of visualizing a structural prototype. Reasonably, this geometric information should be simple to utilize for structural analysis as well. However, such model analysis contains simplifications of the geometry in order to be manageable and to reflect the structure's actual behaviour.

Structures such as bridges are commonly constructed with reinforced concrete, which design derives from calculated section forces. Bridges are large and complex constructions and a structural analysis model commonly requires simplification in order to be reasonable due to computing capability and results. This can be done by using shell and beam elements instead of solid elements for the structural analysis.

Scanscot Technology develops the structural analysis software BRIGADE/Plus, which can import BIM-models from the interoperable open IFC format. The result of such an import is solid element analysis models. In order to simplify the model to shell and beam elements, the engineer has to reconstruct the model. This led to the development of a feature in BRIGADE/Plus that allows an automatic simplification to shell and beam elements when importing a BIM-model.

This feature is based on the existing import of IFC-files in BRIGADE/Plus. When loading an IFC-file the object definition of all parts are scanned. Defined beams and columns will be imported as beam elements in BRIGADE/Plus, while slabs and plates will be imported as shell elements. Remaining parts and parts that does not fulfil certain criteria to be simplified are imported as solid elements. The simplification to a beam element is based on the given information length, orientation and profile written in the IFC file. For shell elements, the simplification occurs when two conditions are fulfilled, namely if the two largest surfaces are equal and parallel. The result is then shell elements in the midsurface of the two largest surfaces.

For the method, the assessment conditions for which objects that shall be imported as shell elements have been tested to give good results. Regarding beam elements, some profiles are not defined equally in IFC as in BRIGADE/Plus, which can lead to an information gap. Some parts that are transformed to shell and beam elements are not always an exact representation of the geometry, but the difference is noticeable small. There are also some cases when advanced geometry fulfil given conditions for shell elements, when they are not entirely suitable for the simplification.

Often BIM is utilised for viewing a building prototype, and for this the definition of objects are not of great importance. However, correct definitions are vital for the import in BRIGADE/Plus with this method. If a model has suitable definitions, it can save the structural analysis engineer a fair amount of time constructing the analysis model.

Förord

Rapporten täcker vårt examensarbete motsvarande 30 högskolepoäng och är det sista vi gör i civilingenjörsutbildningen i väg- och vattenbyggnad vid Lunds Tekniska Högskola. Arbetet har genomförts under våren 2014 på Scanscot Technologys kontor i Lund och i samarbete med avdelningen för konstruktionsteknik på LTH.

Idén till examensarbetet kommer från Johan Kölfors och Anders Vennerstrand på Scanscot Technology. Båda två har varit handledare under arbetets gång och vi vill tacka dem för den tid de lagt ner på att hjälpa oss. Vi vill också tacka samtliga anställda på Scanscot Technology för den hjälp vi fått med de problem vi stött på och för den trevliga arbetsplats vi fått vara del av under våren. Vi vill även tacka Oskar Larsson på avdelningen för konstruktionsteknik som har varit examinator och som hjälpte oss att hitta rätt examensarbete.

Robin Nerborg och Johan Olsson
Lund, maj 2014

Innehållsförteckning

Inledning

Bakgrund

Det blir allt vanligare att projektering inom byggbranschen sker med hjälp av datorbaserade 3D-verktyg. Ofta är det samma geometriska information som används både i projekteringsverktyget och i beräkningsmodellen i ett strukturanalysprogram. Informationsflödet mellan dessa verktyg är inte alltid det bästa, vilket resulterar i att beräkningsingenjörer ofta får rita om den geometriska information som redan finns.

I Scanscot Technologys strukturanalysprogram BRIGADE/Plus är det idag möjligt att importera modeller från olika ritverktyg. Alla byggnadselement importeras idag som solidelement, något som inte alltid är önskvärt. På grund av dimensioneringsmetoden så är det framförallt vid dimensionering av armerade betongkonstruktioner som solidelement inte kan användas. En möjlighet att importera objekt som skal- och balkelement i strukturanalysprogram skulle leda till betydande effektiviseringsvinster för beräkningsingenjören.

Syfte

För att underlätta arbetet för beräkningsingenjörer ska en metod tas fram där importen av BIM-modeller i BRIGADE/Plus genererar beräkningsmodeller bestående av skal- och balkelement.

Avgränsningar

Det finns oändligt många geometrier som skulle kunna vara med i byggnadsmodeller. För att hålla arbetet på en rimlig nivå har hänsyn tagits till de vanligaste elementen och de vanligast tillvägagångssätten inom BIM.

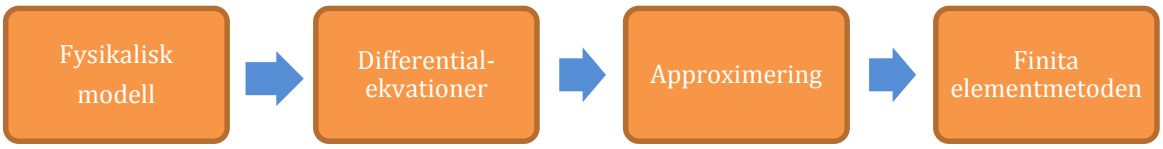
Det finns mycket som går att förenkla för användaren av BRIGADE/Plus. Delar av detta beskrivs i avsnittet ”Förslag på utveckling av BRIGADE/Plus”. För att hålla arbetet inom vårt primära kunskapsområde (konstruktionsteknik) har ingen större vikt lagts vid den datatekniska användarvänligheten av programmet.

Grundläggande begrepp

Finita elementmetoden (FEM)

Finita elementmetoden är ett kraftfullt matematiskt verktyg som används inom många ingenjörsområden. De flesta fysikaliska modeller och strukturers beteende definieras med hjälp av differentialekvationer. Dessa kan bli väldigt svåra och tidskrävande att lösa med analytiska metoder. För komplicerade strukturer som exempelvis broar blir detta i princip omöjligt.

Finita elementmetoden bygger på att istället för att söka en lösning på differentialekvationerna som beskriver problemet över hela regionen, så delas denna region in i mindre delar, så kallade finita element. Vidare tilldelas dessa element en enkel approximativ lösning som är nära den verkliga lösningen. Fördelen med denna uppdelning är att även om en variabel varierar på ett komplicerat och olinjärt sätt över hela regionen så kan variabeln antas variera linjärt eller med högre polynom över ett element. Beskrivningen av finita elementmetoden i detta avsnitt bygger på Ottosen Saabye, Petersson (1992).



Figur 1 - Förenklingssteg för finita elementmetoden.

När approximationen för varje element är bestämd kan de kopplas samman efter olika regler, för att tillsammans ge en lösning för hela den beskrivna regionen. Det sammanhängande området av element kallas för elementnät eller ”*mesh*”. Där elementen kopplas samman formas knutpunkter som kallas för noder, vilka även kan existera i ändelementens ränder. För varje nod existerar ett visst antal frihetsgrader som tillskrivs vissa egenskaper, till exempel hur noden roterar eller förskjuts i olika riktningar.

Finita elementmetoden kan alltså användas för att lösa generella differentialekvationer, och den är tillämpningsbar för många olika fysikaliska modeller. Inom strukturmekaniken används den ofta för att lösa randvärdesproblem, där variabler så som upplagsvillkor och laster för modellen är kända.

Generellt gäller att hela regionens approximativa lösning blir bättre med fler element och högre polynomrang. Beräkningarna får då en högre noggrannhet med konsekvensen att ekvationssystemet växer i storlek vilket gör att beräkningarna blir mer omfattande. Lösning av stora ekvationssystem kräver datorkraft, där adekvat noggrannhet bör väljas med hänsyn till konvergens i resultat och tillgänglig datorkraft.

Beräkningsgång med FEM

För konstruktionsberäkningar kan FEM-ekvationen beskrivas på matrisform (Ottosen Saabye, Petersson 1992):

$$\boldsymbol{Ku} = \boldsymbol{f}$$

K styvhetsmatris (nxn)
u förskjutningsvektor (nx1)
f kraftvektor (nx1)

Där **K** är strukturens styvhet. När strukturen belastas med kraften **f**, så uppstår deformationen **u**. Styvhetsmatrisen är en beskrivning av strukturens geometri och innefattar längder, tvärsnittsarea, tröghetsmoment samt materialegenskaper som tvärkontraktionstal, elasticitetsmodul och längdutvidgningskoefficient. Förskjutningsvektorn **u** innehåller information om nodernas translation och rotation. Här tilldelas systemet även upplagsvillkor där kända förskjutningar föreskrivs när exempelvis ett upplag är förhindrat en viss rörelse.
Kraftvektorn **f** tillskrivs de kända laster som påverkar strukturen.

Stegvis kan beräkningsgången för finita elementmetoden beskrivas (Gustafsson 2012):

1. **Modellering**, strukturens topologi förenklas för att skapa en beräkningsmodell. Detta görs med hänsyn till geometri, laster och randvillkor. Till elementen görs antaganden om materialbeteenden, där det oftast antas ett linjärelastiskt beteende.

2. **Elementsamband**, här skapas först lokala ekvationssystem. Eftersom riktningar är angivna i lokala system för varje element så transformeras dessa till ett globalt system.

3. **Assemblering**, elementekvationerna kombineras till ett gemensamt ekvationssystem. Hela systemet får en global styvhetsmatris, förskjutningsvektor samt kraftvektor.

4. **Lösning**, ekvationssystemet löses och resultat erhålls som värden på förskjutningar för varje nod, exempelvis translation och rotation.

5. **Elementkrafter**, med hjälp av förskjutningarna kan reaktionskrafter beräknas.

6. **Spänningar**, med hjälp av förskjutningarna kan även spänningar beräknas.

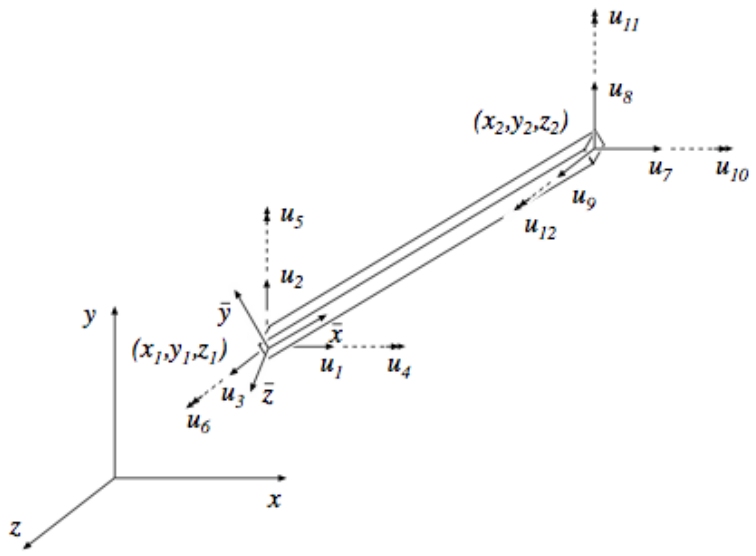
Grundläggande strukturelement i finita elementmetoden

Strukturanalysen ligger till grund för att ta fram snittkrafter såsom normalkraft, tvärkraft och moment, då dessa storheter används vid dimensionering av strukturens delar.
När finita elementmetoden tillämpas kan användaren justera modellen på flertalet sätt för att uppnå resultat som skall spegla den verklighet som söks. För att skapa en beräkningsmodell med finita elementmetoden idealiseras geometriska delar till olika elementtyper, där det används olika basfunktioner för att approximativt söka en lösning för den tänkta variabeln. Nedan följer några exempel på element som används (Davidson 2003).

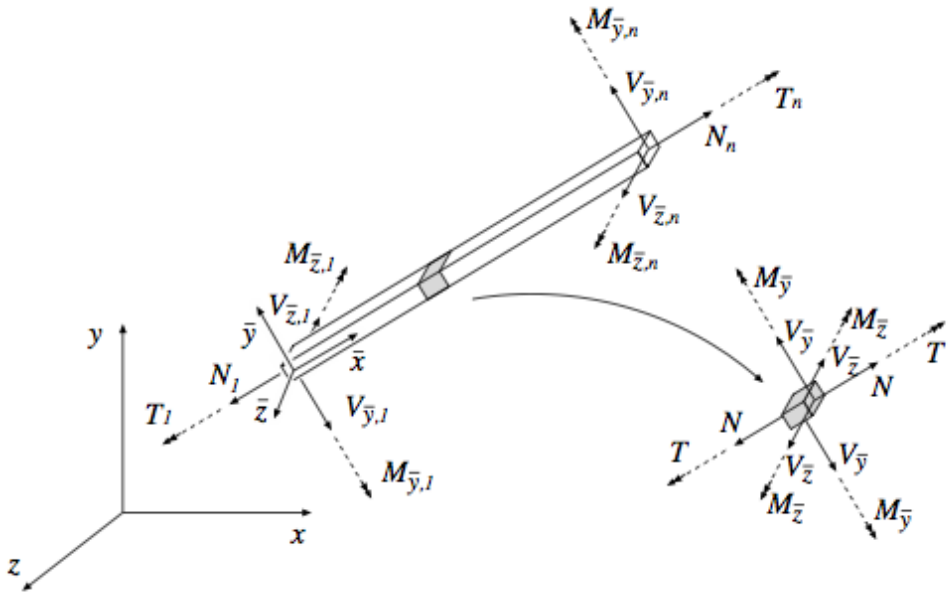
1 – dimension	**2-dimensioner**	**3-dimensioner**
Linjeelement	2D balk	3D balk
	Skiv-, plattelement	Skalelement

Balkelement 3-dimensioner

Principen med balkelement är att antaganden görs så att den okända variabeln endast varierar längs balkens längdaxel. Antaganden bygger på flera olika teorier och strukturanalysprogram använder olika teorier beroende på vilken elementtyp som tillämpas. Den simplaste teorin är Euler-Bernoulli som försummar skjuvdeformation (Simulia 2014a). Elementet innehåller två noder med sex frihetsgrader vardera, se Figur 2. För balkelementet kan då resultaten normalkraft, tvärkraft och moment presenteras i elementets mitt samt i dess noder, se Figur 3.



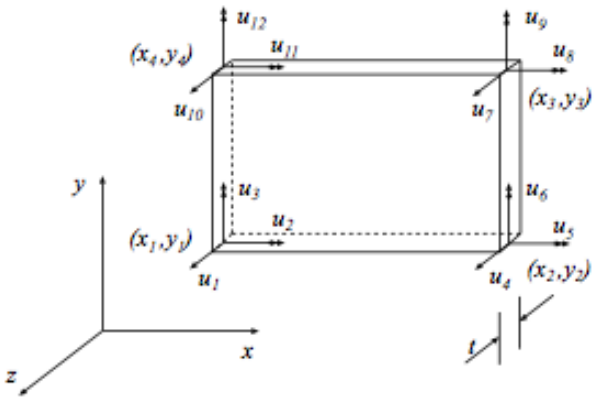
Figur 2 - Balkelement i tre dimensioner med frihetsgrader (Austrell m.fl. 2004).



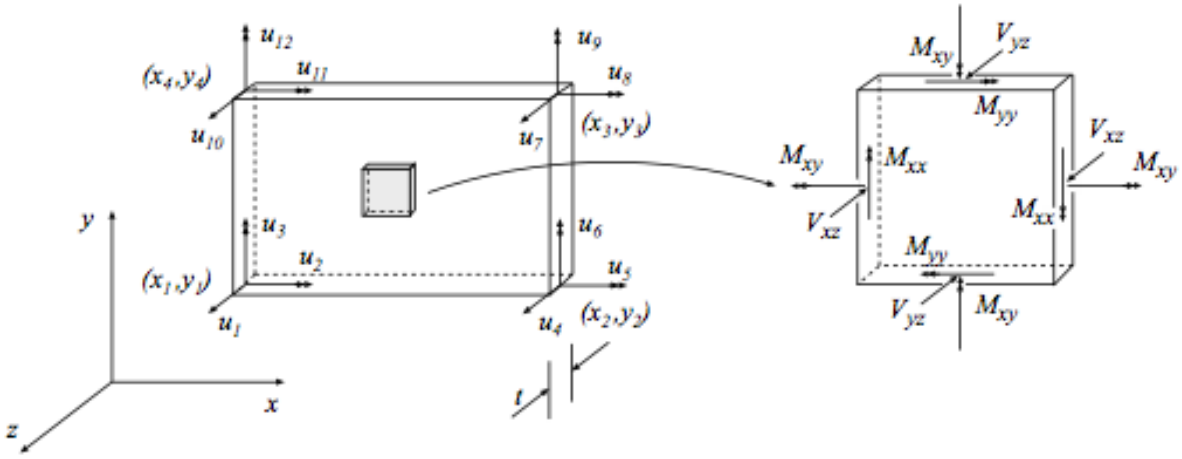
Figur 3 - Balkelement i tre dimensioner med verkande krafter och moment (Austrell m.fl. 2004).

Plattelement

För att förenkla en tredimensionell solid reduceras problemet till en enklare beräkning i två dimensioner där elementet antas vara symmetriskt ovan och under tyngdpunktsplanet i XY-planet, med tjockleken *t (*Ottosen Saabye, Petersson 1992). Ett plattelement innehåller tolv frihetsgrader, se Figur 4. Resultat för elementet ges som tvärkraft och moment, se Figur 5.



Figur 4 - Plattelement med frihetsgrader (Austrell m.fl. 2004).



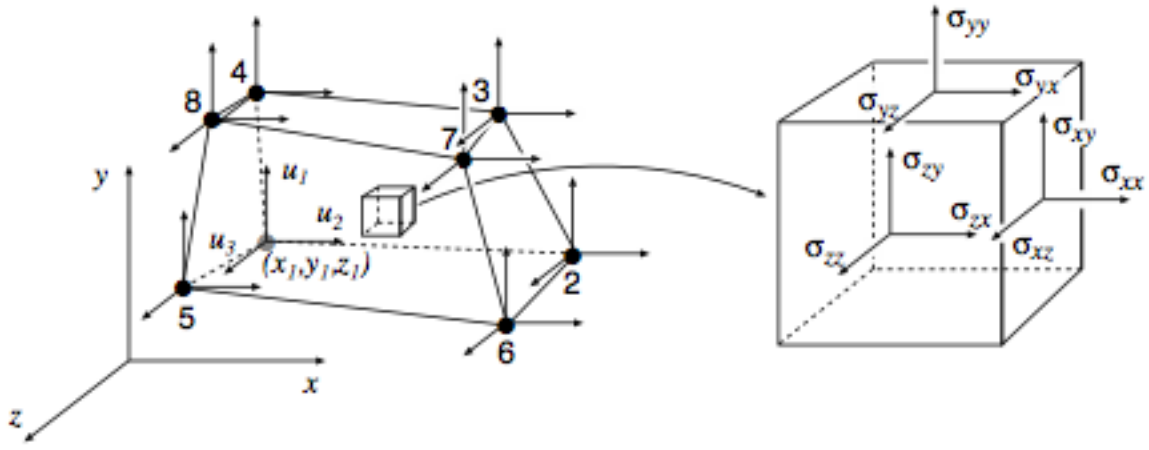
Figur 5 - Plattelement med verkande krafter och moment (Austrell m.fl. 2004).

Skalelement

Skalelement innehåller fler frihetsgrader än plattelement och kan även beräkna normalkraft, då elementet tar hänsyn till spänningar inom planet (Simulia 2014b). Detta leder till att antalet frihetsgrader ökar till 24 stycken. De fyra noderna innehåller sex frihetsgrader, tre rotationer och tre translationer.

Solidelement

Solidelement blir nödvändiga när antingen geometrin eller lasterna är för komplicerade, eller när mer detaljerade resultat efterfrågas. Noderna innehåller vardera tre frihetsgrader i form av translationer, varpå resultat ges i form av spänningar, se Figur 6.

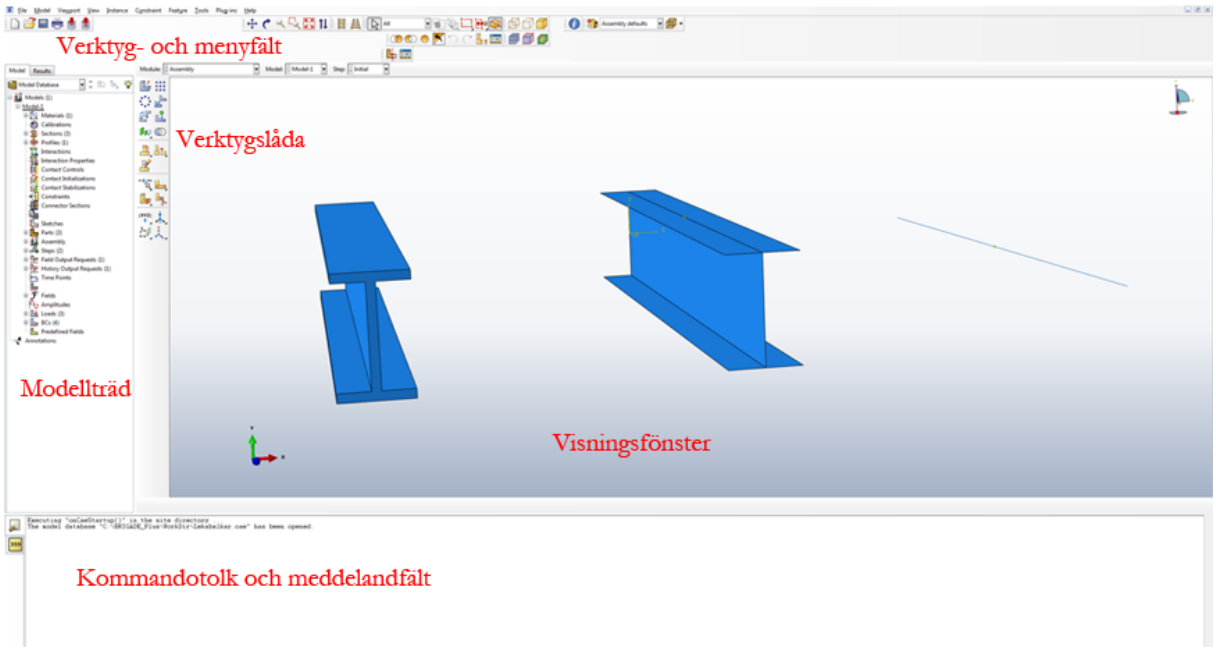


Figur 6 - Solidelement med frihetsgrader samt verkande spänningar (Austrell m.fl. 2004).

BRIGADE

För detta arbete används Scanscot Technologys program BRIGADE vilket används för strukturmekaniska beräkningar främst för brokonstruktioner med hjälp av finita elementmetoden. Programmet tillämpas för att göra strukturanalyser och beräkna samspelet mellan reaktionskrafter och yttre laster som påverkar en bromodell. Detta för att kunna dimensionera konstruktionsdelar med avseende på hållfasthet. Broar är oftast komplicerade och måste dimensioneras för ett mycket stort antal lastställningar och kombinationer av laster.

BRIGADE finns i två versioner, BRIGADE/Standard och BRIGADE/Plus. Gemensamt för de två versionerna är att båda använder samma beräkningsplattform från Dassualt Systems program Abaqus. BRIGADE innehåller utökade funktioner som är praktiska för bromodellering, som till exempel lastkombinering och trafiklast.



Figur 7 - Gränssnittet i BRIGADE/Plus

BRIGADE/Plus kan hantera både rutinproblem och avancerade beräkningar. Programmet kan förenklat delas in i tre olika huvuddelar; *preprocessor, solver* och *postprocessor*.

Inom *preprocessor* skapar användaren en beräkningsmodell innehållande geometri, randvillkor, laster och elementindelning. Detta kan göras på olika sätt, vanligtvis med hjälp av funktionerna i gränssnittet. Det finns möjlighet att importera geometrier från en rad olika CAD-filformat, dock kan problemet med ett vanligt ritningsformat som till exempel *DXF* vara hur information är sparad. Formatet är vektorbaserat och användaren måste lägga en hel del tid på att bygga om modellen till en beräkningsbar modell.

Fördelen med att rita en 3D-modell i ett BIM-baserat ritningsprogram är att information är knuten till objekt, och modellen kan oftast exporteras till filformatet IFC som BRIGADE/Plus idag kan importera. BIM och IFC är utförligt förklarat i egna avsnitt längre fram i rapporten.

En annan funktion i BRIGADE/Plus är att användaren kan köra *script* skrivna med programmeringsspråket Python, detta går att skriva direkt i gränssnittet eller så kan externa *script* matas in i programmet. En användare som utnyttjar detta kan undvika repetitivt arbete.

Beräknings- och simulationsfasen kallas *solver*. Detta är en kärnfunktion som användaren inte själv kan påverka eller gå in och styra. Efter att beräkningarna genomförts kan resultat presenteras och denna del kallas för *postprocessor*.

Beräkningsmodell

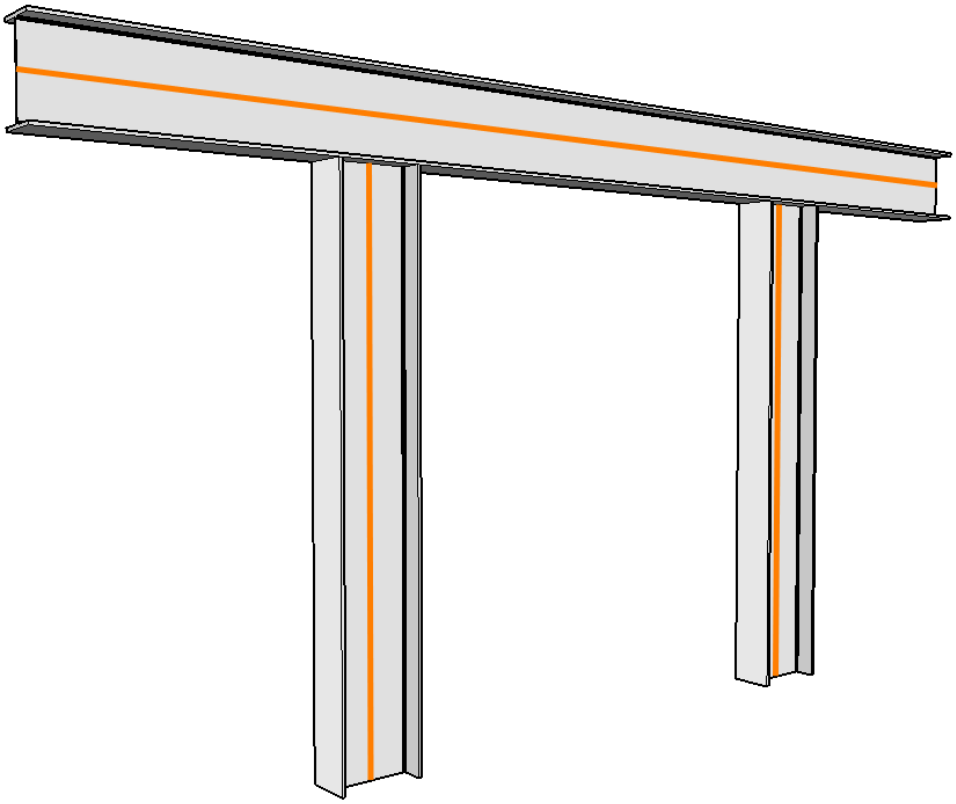
Valet av element kommer ha betydelse för vilka resultat som kan presenteras. Valet av elementtyp styrs av konstruktionsdelens geometri och mekaniska verkningssätt. Elementen bör väljas med hänsyn till detta.

När användaren i BRIGADE/Plus väljer vilka resultat som skall presenteras i resultatlistan går det inte att för solida element välja snittkrafter, en begränsning som gäller generellt för solider. Däremot kan användaren i *postprocessorn* med hjälp av verktyget *Free Body Cut* få fram snittkrafter för ett tvärsnitt bestående av solida element. Exempelvis sker beräkningen av moment i ett valt tvärsnitt med hjälp av nodkrafter i de noder som är belägna i tvärsnittet. Detta kräver att användaren har producerat ett elementnät som gör att verktyget kan användas.

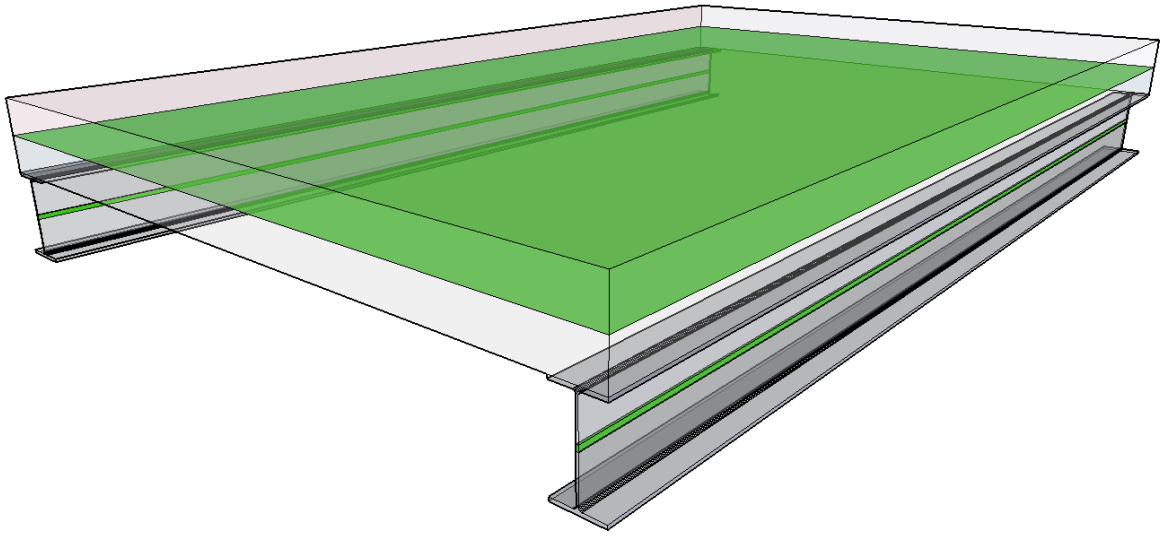
Eftersom en solidmodell jämfört med en skal- och balkmodell innehåller fler element vid elementuppdelning kommer en sådan modell kräva mer datorkraft och tid för beräkning. En brokonstruktion modellerad med solidelement blir nästintill orimlig med tanke på tidsåtgången att genomföra analysen. Det viktigaste för en användare är att ha god kännedom om hur vald beräkningsmodell presenterar resultaten, och hur de kan tolkas. Eftersom en linjärelastisk analys oftast antas, kommer det med säkerhet vara en diskrepans mellan det verkliga beteendet och de beräknade resultaten för armerad betong. Detta eftersom materialet beter sig olinjärt på grund av uppsprickning, olinjär respons i tryckzon samt plasticering av armeringen (Davidson 2003). Detta skapar en omfördelning av spänningarna vilket inte syns i resultatmodellen.

När beräkningsmodellen idealiseras till tredimensionella skal- och balkelement, se Figur 8 samt Figur 9, så kommer det uppstå en geometrisk skillnad i anslutningarna mellan elementen. Detta följs av att representationen för linjer och ytor placeras längs tyngdpunktslinjer och tyngdpunktsplan.

BRIGADE/Plus kan hantera detta genom att till exempel införa stela kopplingar mellan regioner som har olika egenskaper, elementnät och ligger med visst avstånd ifrån varandra.



Figur 8 - Modell av balkar med inritade balkelement

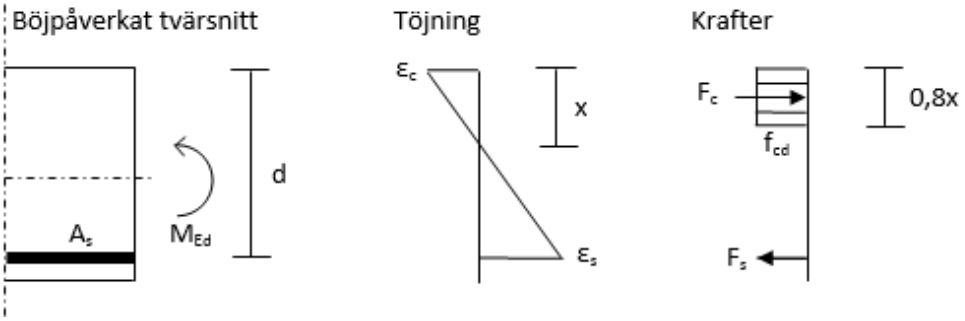


Figur 9 - Modell av balkar och bjälklag med inritade balk- och skalelement

Betong i konstruktioner

Anläggningskonstruktioner, bland annat broar, består ofta av armerad betong. Egenskaper och verkningssätt hos armerade betongkonstruktioner skiljer sig mycket mot stål- och träkonstruktioner. Armerad betong har inte en linjärelastisk spänningsfördelning och modeller med solidelement ger därför inte en korrekt bild av materialet.

Draghållfastheten i betong är mycket låg, runt tiondelen av tryckhållfastheten (Isaksson m.fl. 2010). Den låga draghållfastheten medför att betongen spricker i dragzoner vilket påverkar den globala hållfastheten. Vid dimensionering av betongkonstruktioner bortser man från draghållfastheten och dimensionerar att armeringen ska klara av all dragbelastning. Detta angreppssätt medför att dimensioneringen blir mer komplex än för stål och trä. För dimensionering med hänsyn till böjbelastade tvärsnitt används det moment som påverkar tvärsnittet för att ta fram dragkraft i armeringen och tryckkraft i betongen enligt en väl utvecklad metodik, se Figur 10. En liknande metod används för dubbelarmerade betongkonstruktioner.



Figur 10 - Dimensioneringsmetod för enkelarmerade betongtvärsnitt (Isaksson m.fl. 2010).

Kraftjämvikt: $F_s = F_c \rightarrow A_s \cdot \sigma_s = f_{cd} \cdot 0{,}8x \cdot b$

Momentjämvikt runt F_s: $M_{Ed} = F_c(d - 0{,}4x) = f_{cd} \cdot 0{,}8x \cdot b(d - 0{,}4x)$

Momentjämvikt runt F_c: $M_{Ed} = F_s(d - 0{,}4x)$

För dimensioneringen av armerad betong så är det alltså moment, tvärkraft och normalkraft som är intressant. En konstruktion som modelleras med solidelement får vid beräkningar resultat i form av spänningar, vilka ej är optimala för dimensionering av armerade betongkonstruktioner. För att få önskade resultat (krafter och moment) ur beräkningsmodellen måste den därför bestå av skal- och balkelement vilka ger resultat i krafter och moment.

Ur BIM-modeller erhålls som regel solidelement eftersom modellerna är till för att geometriskt visa utseende, utbredning, mängd med mera av elementen. Alla element av armerad betong som kommer från BIM-modeller måste därför göras om till skal- och balkelement för att kunna ge önskvärda resultat i en beräkningsmodell.

BIM

Building Information Modelling (eller *Building Information Model*, BIM) har sedan 90-talet utvecklats som framtidens 3D-modellering inom byggteknik utan någon exakt definition. I början handlade det om att förbättra informationen i 3D-CAD och göra den plattformsoberoende, på senare tid har det handlat om att förbättra standarder och anpassa dessa efter marknadens behov (JBIM 2007).

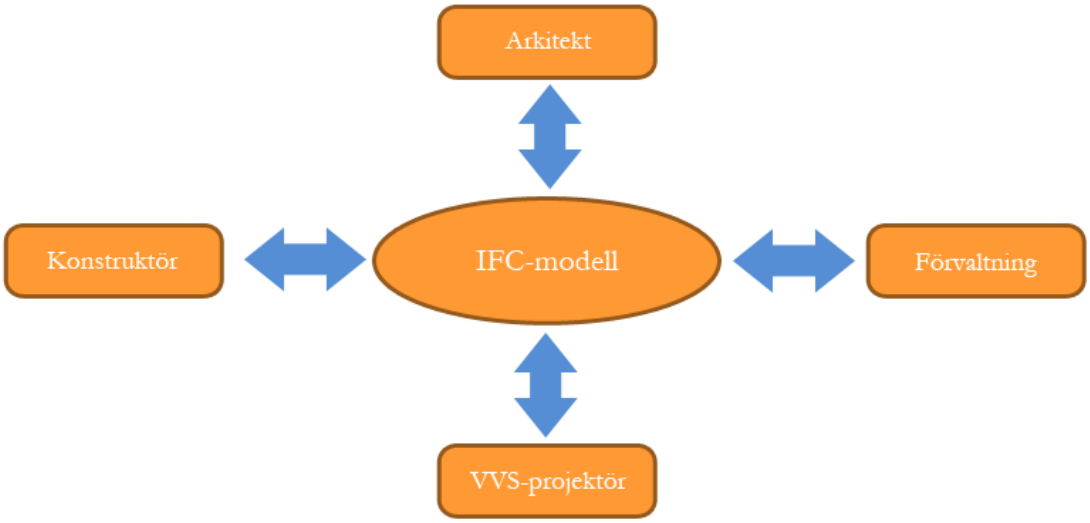
Idag är BIM ett verktyg för att behandla information i alla led av ett byggprojekt, från idé, design och dimensionering till byggnation, förvaltning och rivning. BIM växer snabbt och det blir vanligare att Trafikverket såväl som privata aktörer ställer krav på användning av BIM (Gelder m.fl.2013). BIM-modeller kan idag även användas till strukturanalyser genom import i program som exempelvis BRIGADE/Plus.

BIM innehåller objektsorienterad geometrisk, fysisk och funktionell information om en byggnad och dess beståndsdelar. Det går därför att bygga upp en virtuell byggnad under projekteringen och genomföra problemlösning så som kollisionskontroll på denna (JBIM 2007). BIM kan även utökas med information om när i tiden saker ska ske, vad olika saker kostar samt förvaltning (4D-BIM, 5D-BIM samt 6D-BIM.)

IFC

År 1994 skapades ett konsortium av tolv företag verksamma inom AEC/FM (*Architecture Engineering Construction / Facilities Management*) med syftet att koppla samman den tidens befintliga programvara. De första testen visade sig lyckosamma, vilket ledde till bildandet av organisationen IAI - *International Alliance for Interoperability* (Ekholm m.fl. 2000). Namnet på organisationen är numera buildingSMART och består av cirka 550 företag från 24 olika länder. En viktig aspekt för att trovärdigförklara organisationens arbete är att den är ideell och icke vinstdrivande, men den är då beroende av bidrag samt volontärarbete.

BuildingSMART har som fokus att utveckla filformatet IFC - *Industry Foundation Classes* som är ett öppet filformat för BIM. Formatet bidrar till smidigare informationsdelning mellan de olika program som används inom AEC/FM. Principen bygger på att information knuten till en byggnadsmodell är centralt samlad i ett filformat, vilket leder till att endast ett filformat behövs för importering och exportering av modellen.



Figur 11 - Exempel på instanser som alla jobbar mot en och samma IFC-modell

Eftersom IFC är ett öppet filformat är det tillgängligt för alla och kan inte heller begränsas av en viss programägare. Bredden av formatet utvecklas kontinuerligt med nya funktioner samtidigt som innehåll stabiliseras för att uppfylla standardisering enligt ISO.

Den version av IFC som används idag heter IFC2x3. Uppföljaren IFC2x4 har lanserats, men det är fortfarande IFC 2x3 som generellt används (buildingSMART 2014a). För att begränsa en version kan den ha olika ”*views*”, som bestäms innehålla en delmängd av hela versionen. En *view* är låst till sitt innehåll, vilket gör att programvaror som använder IFC efter en viss *view* inte behöver kontinuerlig uppdatering. För att en programvara skall kunna visa på tillförlitlighet kan den certifieras för en viss *view* av buildingSMART. Den *view* som primärt används idag är IFC2x3 *Coordination View* V2.0 (buildingSMART 2010). Andra delar av standarden bland annat IFC2x3 *Structural Analysis View* som är intressant ur beräkningsteknisk synvinkel, är fortfarande är under utveckling.

Exportering och importering av IFC-filer

IFC är strukturerat att vara flexibelt. Det finns till exempel flera varianter för att geometriskt representera samma konstruktionselement eller assemblera element i en modell. Den översättning till IFC som görs vid exportering är alltså nödvändigtvis inte entydig. En identisk modell i olika ritningsverktyg kan alltså generera en IFC-fil med samma betydelse, men som är konstruerad med olika innehåll.

För att ett program skall exportera en modell till IFC krävs det att programutvecklaren har konstruerat denna funktion. Det samma gäller för importering. BuildingSMART utfärdar certifiering för import och export till IFC (buildingSMART 2014b). I bilaga 1 presenteras de i nuläget kommersiella programvaror som är certifierade. Certifiering kan anses vara viktigt för mjukvaruägare eftersom det innebär att programmet lämnar en stabil IFC-modell till en annan part.

Certifiering behöver inte betyda att programvarans export eller import är felfri. IFC har en komplicerad struktur och det krävs tid för att skapa sig en förståelse för inbördes samspel. En rimlig bedömning är att en certifieringsprocess inte kan innehålla samtliga scenarion, och en certifiering kan då sägas innebära att ett programs exportering eller importering av IFC är stabil, men nödvändigtvis inte felfri.

De modeller som använts inom detta arbete är exporterade av *Tekla Structures* eller *Autodesk Revit Structure* som båda är certifierade för export för IFC 2x3 *Coordination View* V2.0 (buildingSMART 2014c).

BRIAGDE/Plus och IFC

BRIGADE/Plus har idag möjligheten att importera IFC-filer för att generera en modell av en geometri producerad med ett BIM-ritningsverktyg. Den modell som då erhålls har geometrisk beskrivning som definieras av konstruktionens omslutningsytor. Finita elementmetoden kommer då bestå av någon typ av solidelement. Det som användaren senare måste komplettera med är information som inte finns i IFC-modellen såsom laster, randvillkor och materialegenskaper. BRIGADE/Plus importering är uppbyggd efter IFC2x3 *Coordination View* V2.0.

Visualisera IFC-modeller

Det finns en rad program som kan läsa av en IFC-fil och visualisera geometrin, vilket kan vara ett bra hjälpmedel för att analysera topologin. Det som har använts inom detta arbete heter Solibri Modelviewer, där det även går att enkelt söka efter enskilda strukturelement.

Strukturen för IFC

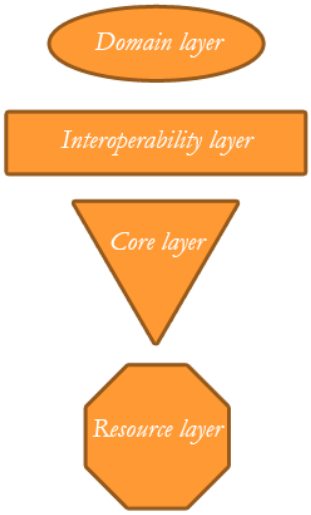
IFC har en komplex struktur som inte är enkel att förstå, och det råder brist på pedagogiska förklaringar till många delar. De enskilda objektens beskrivningar finns tillgängliga och är bra, däremot är det inte helt lätt att få en övergripande bild om hur allt interagerar. Det är viktigt att känna till att den aktiva delen kring IFC sker vid exportering och importering till formatet. Själva IFC är ”passivt” och är endast ett sätt strukturera information.

Styrkan med BIM- och IFC-modeller är att de är objektbaserade, där information lagras knutet till objekt. Denna skillnad blir avgörande mot vanliga CAD-format som sparar information som till exempel linjer och ytor i olika lager.

För detta arbete är det inte relevant att gå igenom hela IFC-strukturen, däremot är det bra att förklara det grundläggande för att sedan detaljerat förklara de viktigaste objekten för detta arbete, vilka är *IfcExtrudedAreaSolid* och *IfcFacetedBreb*. Dessa två är centrala objekt som definierar geometrier och är de som stöds av BRIGADE/Plus för import av IFC.

IFC är uppbyggt av cirka 770 objekt som är uppdelade hierarkiskt i fyra olika nivåer, se Figur 12. Den hierarkiska uppdelningen innebär att objekt kan ärva information från objekt som ligger ovanför eller på samma nivå i hierarkin. Samtidigt kan ett objekt endast referera till andra objekt inom samma nivå i hierarkin eller nedanför. Denna begränsning är också viktigt eftersom de lägre nivåerna är gemensamma för alla discipliner som använder en IFC-modell, men inte den högsta. Detta hindrar ett objekt i en gemensam nivå från att kräva information i en högre nivå som kanske till exempel tillhör en annan disciplin.

Nedan beskrivs de nivåer som är definierade i IFC (Khemlani 2007). Det bör påpekas för läsaren att ett objekt syftar till en samling information som kan vara abstrakt, fysisk eller så kan till exempel ett objekt beskriva en relation mellan andra objekt.



Figur 12 - Nivåer i IFC

Resource layer – Detta är den lägsta nivån i hierarkin och innehåller den grundläggande och generella informationen i IFC-modellen. Detta innefattar egenskaper såsom geometri, material, materialkostnader med mera. Det är till exempel i detta lager som geometriska punkter definieras vilka sedan används för att rita och placera geometriska former. Ett objekt behöver endast beskrivas en gång här, även om det används flera gånger i modellen.

Core layer – Kärnlagret definierar den grundläggande uppbyggnaden av modellen. Lagret innehåller vilka aktörer som är inblandade, delar i byggnaden och hur dessa är placerade och vilka relationer objekt har sinsemellan. Enkelt uttryckt är det här innehållet i projektet definieras, och det är denna information som de högre lagren använder för att knyta samman modellen.

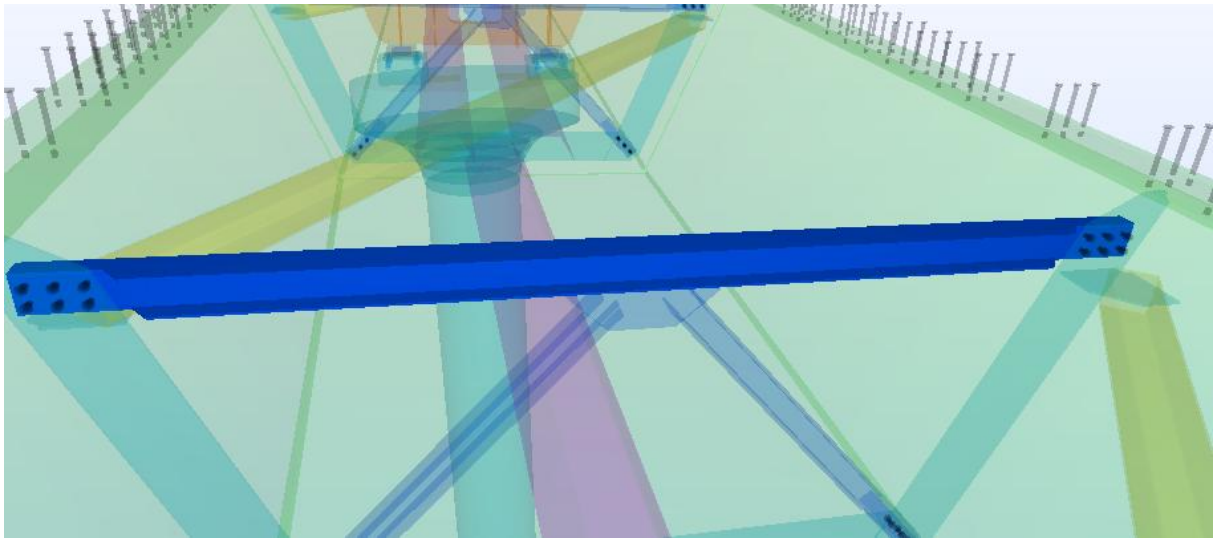
Interoperability layer – Den näst högsta nivån i hierarkin, och även den sista nivån där information delas av de som arbetar med modellen. Här definieras till exempel byggnadselement såsom balkar, väggar, plattor med mera. Här finner vi då de viktiga IFC-objekten för detta arbete, *IfcPlate*, *IfcSlab*, *IfcBeam* och *IfcColumn*.

Domain layer – Detta är den högsta nivån, där innehållet är disciplinspecifikt. Här definieras objekt som är specifika för förgreningar, exempelvis arkitekt, struktur, VVS och fastighetsskötsel.

Det finns även en semantisk gruppering av objekten inom varje nivå, denna uppdelning delar in objekten i avdelningar och lagrar dem som enheter. En enhet i en lägre nivå blir kallad subenhet, om den används av en enhet i högre lager som då kallas för supraenhet. Syftet med denna gruppering har ingen betydelse för hierarkin, utan den gör det enklare att överskåda objekt inom olika avdelningar så som vilka objekt som primärt används av en viss disciplin.

IFC-syntax

En IFC-modell är i grund och botten en textfil, som är uppbygg enligt det ISO-certifierade *STEP-file* formatet. Detta används främst för informationsutbyte och är praktiskt när en textrad i taget skall läsas av (Loffredo 2014). Filen inleds med en huvuddel som kort beskriver vilket program som exporterat, IFC-version, datum och namn med mera. Större delen av filen är datasektionen som innehåller alla IFC-objekt som används i modellen. Varje rad innehåller endast ett objekt och enligt *STEP*-formatet har varje rad ett id-nummer. Det går inte att se någon hierarkisk uppdelning av raderna utan detta kan endast uppfattas när man studerar varje objekt för sig.



Figur 13 - Exempel på en balk ur en IFC-modell

Balken som visas i Figur 13 används för att överskådligt förklara IFC:s syntax, i textfilen är balken utskriven på följande sätt:

#64473=IFCBEAM('1DE3Ib0002C34oEJKrCp8q',#5,'BALK','HEA160','HEA160',#64458,#64472,'ID4D3834A5-0000-0230-3132-393535333234');

Till vänster om likhetstecknet finner vi ett id-nummer för raden. Direkt till höger om likhetstecknet står IFC-objektet skrivet, som inom parentes innehåller en rad olika attribut. För att definiera detta objekt, *IfcBeam*, så krävs det åtta attribut som alla anges i Tabell 1.

Tabell 1 - Attributen i IfcBeam

Textsträng	Attribut
1DE3Ib0002C34oEJKrCp8q	*GlobalId*
#5	*OwnerHistory*
BALK	*Name*
HEA160	*Description*
HEA160	*ObjectType*
#64458	*ObjectPlacement*
#64472	*Representation*
ID4D3834A5-0000-0230-3132-393535333234	*Tag*

De flesta attribut är självförklarande, med undantag från *Representation*. Detta attribut syftar till hur det fysiska objektet är geometrisk representerat. Det finns flera olika varianter beroende på vilket fysiskt objekt som skall beskrivas. Hur geometrier byggs upp är viktigt för detta arbete och förklaras i detalj i nästa avsnitt.

När ett attribut är skrivet som ett id-nummer av en annan rad innebär detta att information refererar till detta objekt, till exempel är *OwnerHistory* skrivet som #5, vilket innebär att den informationen finns i ett objekt på angiven rad. Ett refererat objekt kan i sin tur innehålla andra refererade objekt, på detta sätt som den hierarkiska uppdelningen kan spåras.

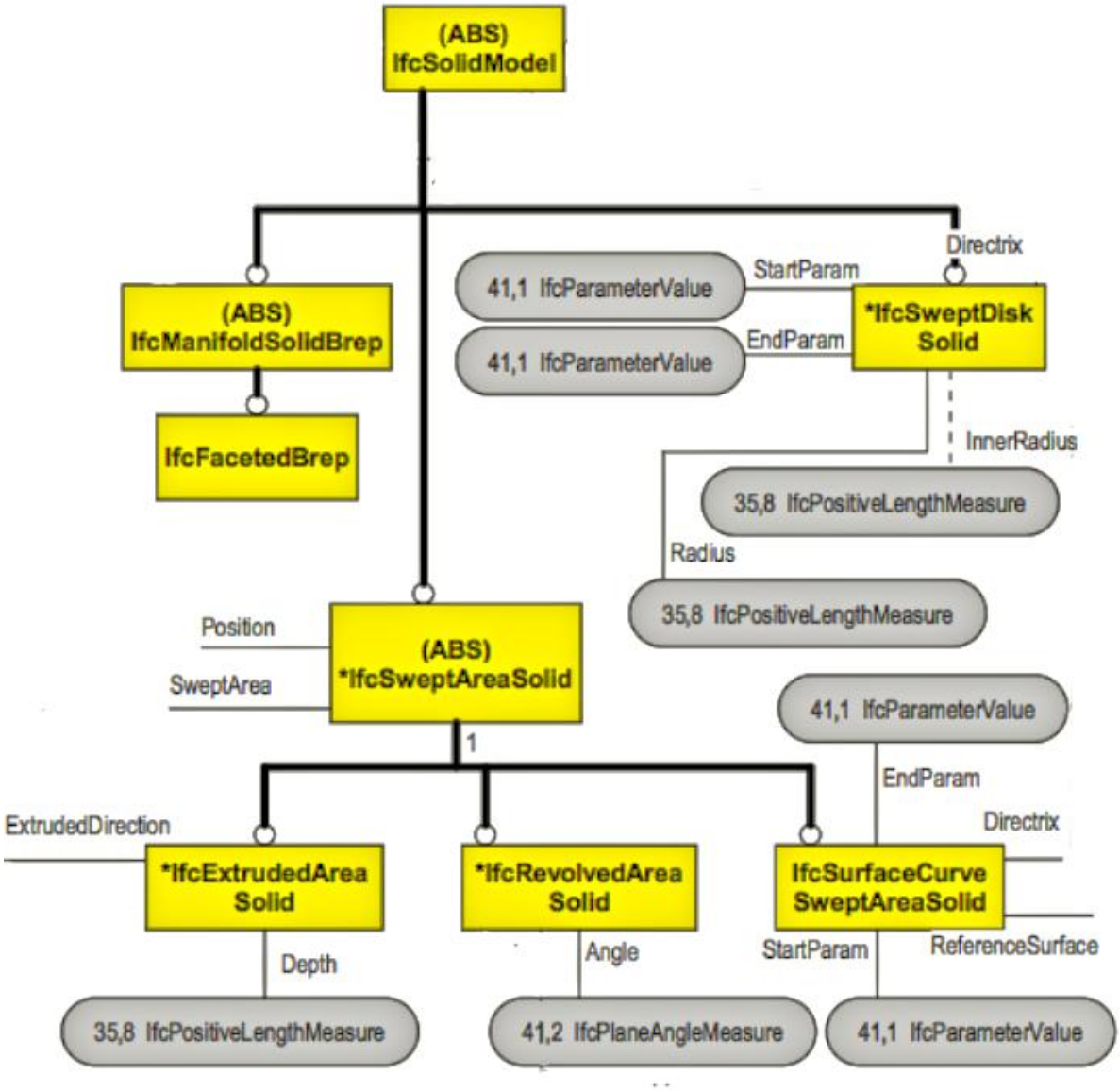
Hur IFC definierar geometrier

Det finns flertalet metoder att geometriskt definiera fysiska objekt i IFC, många av dessa har tillkommit i efterhand varpå IFC2x3 *Coordination View* V2.0 innehåller endast ett fåtal varianter. Beskrivning av samtliga IFC-objekt finns på webbsidan buildingSMART (2007).

De två primära metoderna för att definiera en geometri i IFC2x3 *Coordination View* V2.0 är *IfcExtrudedAreaSolid* och *IfcFacetedBreb*. Det första av dessa hanterar oftast balkar, och principen är att ett tvärsnitt dras en viss längd längs en linje. Det andra sättet kan hantera komplicerade geometrier och innebär en uppbyggnad med omslutningsytor.

Det finns andra sätt att definiera ett solitt objekt vilket är illustrerat i Figur 14, där förgrening efter lagringsobjektet *IfcSolidModel* även innehåller *IfcRevolvedAreaSolid, IfcSurfaceCurveSweptAreaSolid* och *IfcSweptDiskSolid*.

Hur IFC-modellens geometriska former hanteras vid exportering kan dels bero på hur CAD-operatören har valt att rita, samt hur använd mjukvara är programmerad att hantera den geometriska informationen vid exportering till IFC.



Figur 14 - Definitioner av solida objekt i IFC2x3 Coordination View V2.0 (buildingSMART 2010)

Tomrum

När till exempel ett rum i en byggnad definieras så kommer IFC sätta samman väggar, tak, golv men även det tomma rummet. Detta gör att program som importerar en IFC-modell inte behöver tolka tomrum utefter befintlig geometri. Denna princip följer även för byggnadselement som innehåller till exempel hål, där först balken definieras i sin helhet och sedan placeras ett definierat tomrum i hålets position. Detta underlättar då till exempel en balk med hål i kan extruderas med konstant tvärsnitt längs en linje, sedan kan ett hål som geometriskt objekt placeras i balken. Detta istället för att göra en komplicerad lösning med extruderingen där tvärsnittet ändras.

Geometriska representationer

För att hjälpa läsaren på traven lägger vi till en förklarning för användandet av ”definiera ett solitt objekt” och ”representera ett solitt objekt”. Ett definierat objekt är ett konkret färdigt geometriskt objekt. Denna information finns i de primära IFC-objekten *IfcExtrudedAreaSolid* och *IfcFacetedBreb* som förklaras i respektive avsnitt om dessa. Definierade objekt används av geometriska representationer som till exempel *Breb* och *SweptSolid*, vilka är placerade högre upp i hierarkin. Geometriska representationer kan använda sig av en eller flera geometriska objekt när de bildar ett byggnadselement.

För ett byggnadselement finns det flertalet representationer som kan användas. Vilken som används beror på hur geometrin är ritad och hur mjukvaran som exporterar är konstruerad. De primära representationerna är *SweptSolid* och *boundary representation*, *brep* numera kallat. Dessa är de som uteslutande har existerat inom de studerade IFC-modellerna.

Brep

Denna representation innebär en uppbyggnad med ränder, som måste vara ytor. Detta tillåter komplexa geometrier, med eller utan tomrum.

SweptSolid

Detta är en enklare representation som avser att en yta, till exempel ett tvärsnitt, extruderas längs en angiven riktning.

AdvancedSweptSolid

Denna är lik den tidigare men innebär att tvärsnittsytan är extruderad längs en ledlinje, *directrix*.

Clipping

Den geometriska representationen använder sig av flera geometriska objekt, till exempel kan ett redan använt tredimensionellt objekt kapas med hjälp av ett plan.

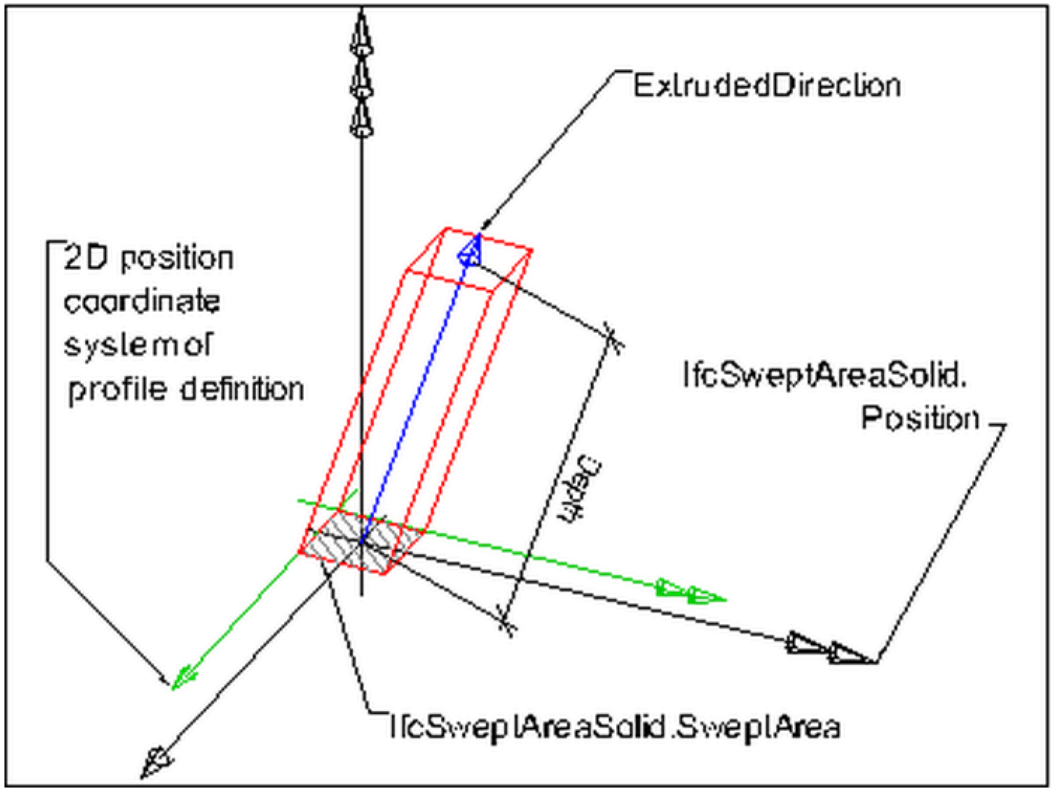
MappedRepresentation
Denna metod innebär att man refererar till ett annat objekts representation. Detta blir lämpligt med flertalet likadana byggnadselement, och istället för att bygga upp alla var för sig så kan ett ritas upp och resten referera till detta.

BoundingBox
Den simplaste geometriska representationen, vilken används för att skapa en box med hjälp av längder på tre sidor.

SurfaceModel
Denna representation bygger på att en eller flera skalgeometrier kombineras.

IfcExtrudedAreaSolid

Inom de studerade modellerna är *IfcExtrudedAreaSolid* ett av de vanligaste sätt att definierade geometriska objekt, principen är att extrudera ett tvärsnitt längs en linje, se Figur 15.



Figur 15 - IfcExtrudedAreaSolid enligt IFC2x3 TC1 (buildingSMART 2007)

Objektet *IfcExtrudedAreaSolid* kräver fyra attribut, som beskrivs nedan.

SweptArea
Detta attribut innehåller *IfcProfileDef* som anger vilket tvärsnitt elementet har. IFC har en rad olika tvärsnitt definierade, såsom I-profil, Z-profil och rörprofil med flera. Se *IfcSweptAreaSolid.SweptArea* i Figur 15, här visas tvärsnittets egna koordinatsystem *2D position coordinate system of profile definition*.

Position
Detta attribut anger det lokala koordinatsystemet för elementet, och det objekt som relateras är *IfcAxis2Placement3D*. Detta objekt innehåller i sin tur riktningen för Z-axeln, X-axeln och en punkt för origo. Riktningen för Y-axeln bestäms som normalen till XZ-planet. Se *IfcSweptAreaSolid.Position* i Figur 15.

ExtrudedDirection
Här anges i vilken riktning som extruderingen skall ske, se *ExtrudedDirection* i Figur 15. IFC-objektet *IfcDirection* tillämpas för att ange en vektor.

Depth
Här anges ett positivt längdmått för extruderingen, se *depth* i Figur 15. Värdet är en *IfcPostiveLengthMeaseure* som kan anges med siffror direkt i objektet eller vara relaterat till *IfcPostiveLengthMeaseure*.

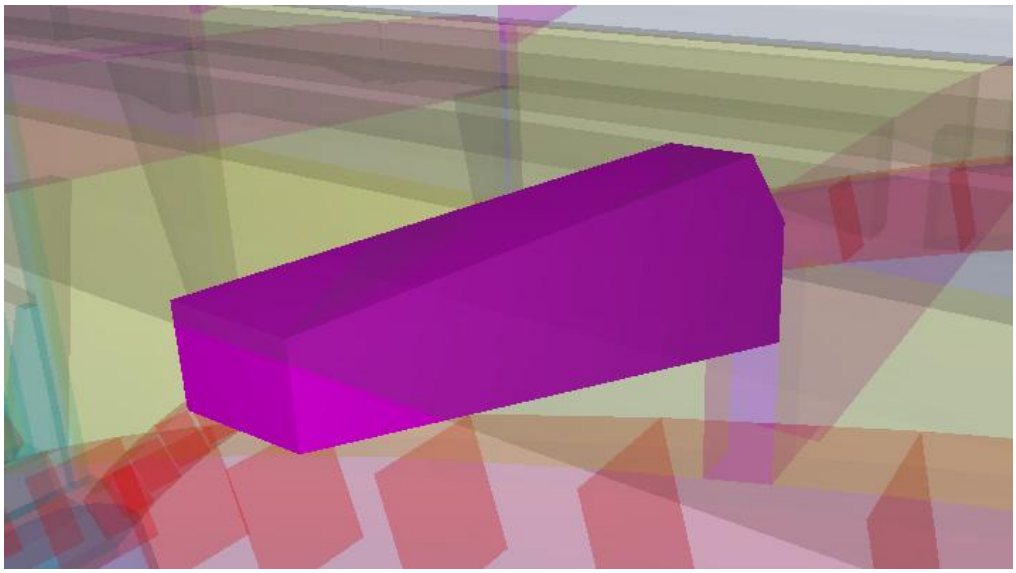
Objektet *IfcExtrudedAreaSolid* är underställt ett annat objekt som heter *IfcSweptAreaSolid*, som samlar geometrier som konstrueras med hjälp av extrudering, se Figur 14. Under detta objekt finns två andra objekt, nämligen *IfcRevolvedAreaSolid* och *IfcSurfaceCurveSweptAreaSolid*. Dessa två är inte implementerade i BRIGADE/Plus och de har inte heller existerat i studerade IFC-modeller.

IfcFacetedBrep

Det andra sättet som tillämpas för att definiera geometriska objekt i IFC tillåter både enkla och komplicerade former. Detta tillsammans med *IfcExtrudedAreaSolid* är de som stöds av BRIGADE/Plus i nuläget, och det är dessa som existerar i de IFC-modeller vi har studerat. Objektet *IfcFacetedBreb* är underställt *IfcManifoldSolidBreb*, som definierar förutsättningarna för att skapa en geometri med *IfcFacetedBreb*, se Figur 14.

För att använda objektet krävs bara ett attribut, vilket i sin tur relaterar till en rad andra objekt. Själva namnet på objektet kanske inte är lika självklart som *IfcExtrudedAreaSolid* varför en kort semantisk förklaring är i sin ordning. ”*Faceted”* betyder ”*Having faces*”, och ”*Brep”* är sen tidigare förklarat en som ”*boundary representation”*. Alltså förklarar objektnamnet ett solitt objekt som representeras av ränder, i form av ytor.

Hur det solida objektet byggs upp blir lättare att förklara med ett exempel. Figur 16 visar ett exempel på ett geometriskt objekt som är representerat som en *breb,* och definierat med *IfcFacetedBreb.*



Figur 16 - Exempel på en kantbalk ur en IFC-modell

IFC-syntax

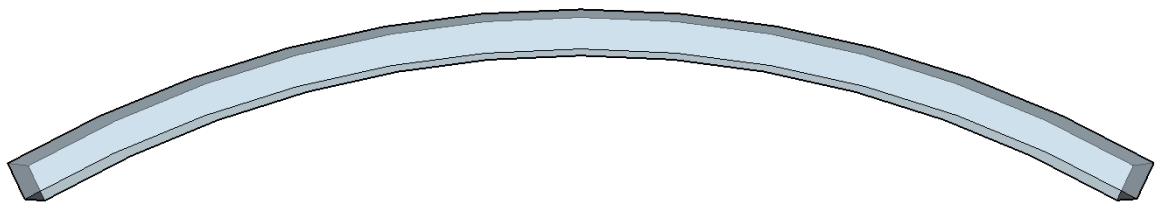
#6214= IFCCARTESIANPOINT((6.96433071425417E-005, 124.999999999956, 238.765482818679));
#6228= IFCPOLYLOOP((#6129,#6128,#6215,#6214));
#6229= IFCFACEOUTERBOUND(#6228,.T.);
#6230= IFCFACE((#6229));
#6231= IFCCLOSEDSHELL((#6193,#6198,#6203,#6208,#6213,§#6218,#6134,#6221,#6224,#6227,#6230));
#6232= IFCFACETEDBREP(#6231);
#6234= IFCSHAPEREPRESENTATION(#12,'Body','Brep',(#6232));
#6235= IFCPRODUCTDEFINITIONSHAPE($,$,(#6234));
#6236= IFCBEAM('1HKKoq006mCp4pDZGpC3Os',#5,'BALK','900*250','900*250',#6186,#6235,'ID51514CB4-0001);

Alla nödvändiga rader inte är med då alla dubbletter är utelämnade, även
de IFC-objekt som inte hänger samman med de understrukna delarna är utelämnade.

Högst upp i hierarkin (*Interoperability Layer*) i det här exemplet är IFC-objektet *IfcBeam,* vilket läses av först. Sedan läses samtliga attribut av, där #6235 hänvisar till *IfcProductDefinitonShape* som identifierar ett element och all relevant information för dess geometriska form.
Objektet innehåller attributet *IfcShapeRepresentation,* #6234. Här återfinns det attribut som beskriver den geometriska representation och i detta fall är detta en *brep*.
Här följer då attributet #6232 som är *IfcFacetedBrep* vilket konstruerar formen. För att detta objekt skall fungera krävs det endast ett attribut, en volym definierad av ett visst antal omslutningsytor. Det attribut som används är *IfcClosedShell,* #6231, vilket samlar en mängd ytor.
En yta, exempelvis #6230 definieras av *IfcFace.* Från start är en yta härled från en rad koordinater, exempelvis #6214. Detta görs genom att först dra en sammanhängande linje med *IfcPolyLoop* genom koordinaterna, #6228. När en linje är dragen kan den sammanhängande linjen definieras som yttre rand för en yta, vilket hanteras av objektet *IfcFaceOuterBound,* #6229.

Detta sätt att beskriva geometrin har dock en begränsning. Objektet *IfcFacetedBreb* kräver att alla ytor är plana och att alla linjer är raka. Vilket leder till att krökta former inte kan hanteras utan en viss partitionering ochmedför att diskretisering av geometrin måste göras.

Som exempel visas en balk med kvadratiskt tvärsnitt som är krökt i en dimension, se Figur 17. För detta fall kommer ändytorna kunna definieras i planet följt av ursprungsformen. Ovan- och underkant kommer däremot segmenteras i mindre ytor placerade längs med den krökta formen, ränderna till dessa ytor måste även följa sidoytornas ränder för att geometrin skall hänga samman. Sidoytorna kommer dock kunna hanteras i planet, däremot har de många dellinjer som bildar dess yttre rand.

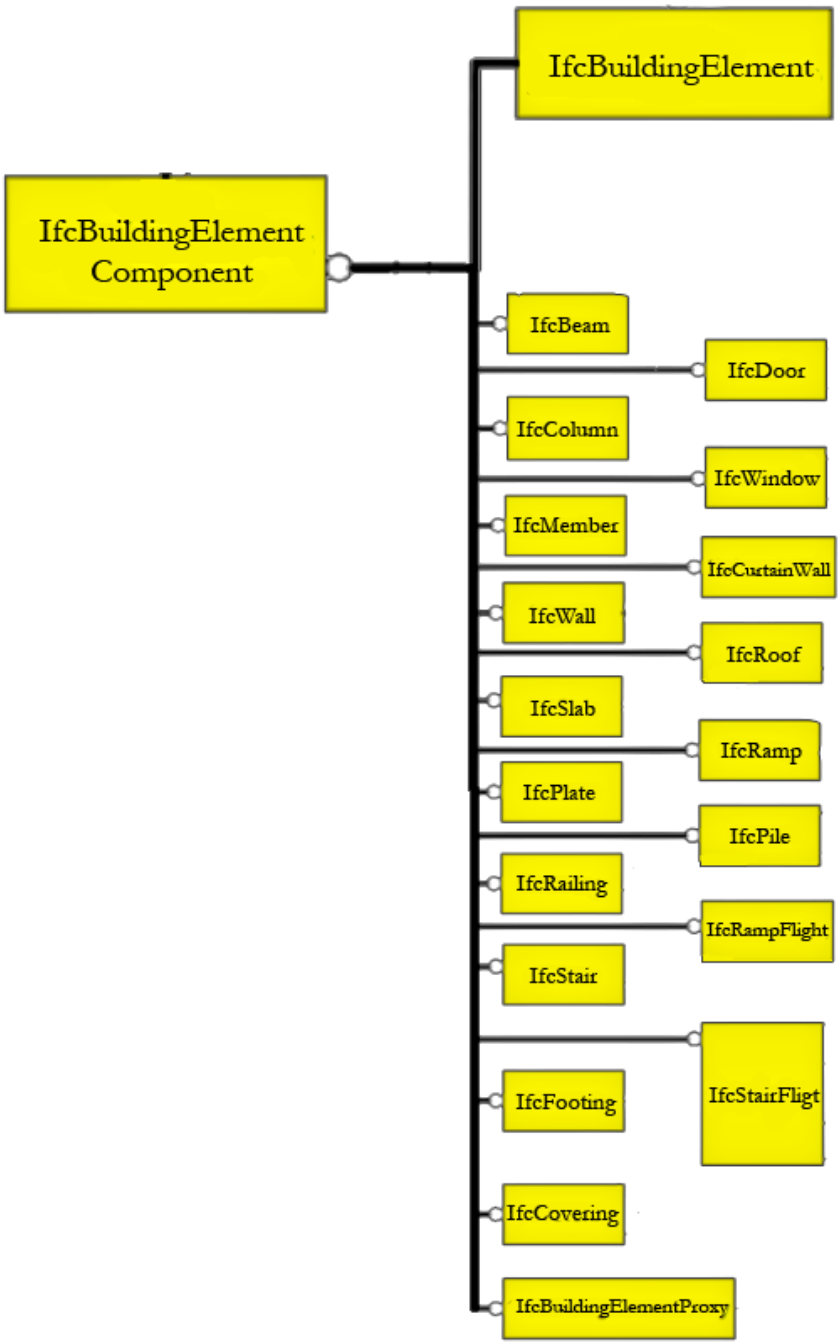


Figur 17 - Balk krökt i en dimension

En liknande geometri i BRIGADE/Plus kommer också partitioneras efter samma princip. Skillnaden för beräkningsresultat är antagligen av mindre betydelse beroende på om diskretisering är gjord i BRIGADE/Plus eller av ett ritningsverktyg som exporterar till IFC.
Det är bra att känna till denna partitionering eftersom det för ett objekt kommer det skapas väldigt många ytor som skall ritas upp, vilket kommer kräva påtagligt mer tid vid en importering i BRIGADE/Plus.

Byggnadselement i IFC

Inom *IfcElement* finns alla komponenter lagrade som ingår i modellen, och under detta objekt är *IfcBuildingElement* placerat, se Figur 18. Här finns objekt för fysiska byggandselement definierade.



Figur 18 – Definitioner av byggnadselement i IFC2x3 Coordination View V2.0 (buildingSMART 2010)

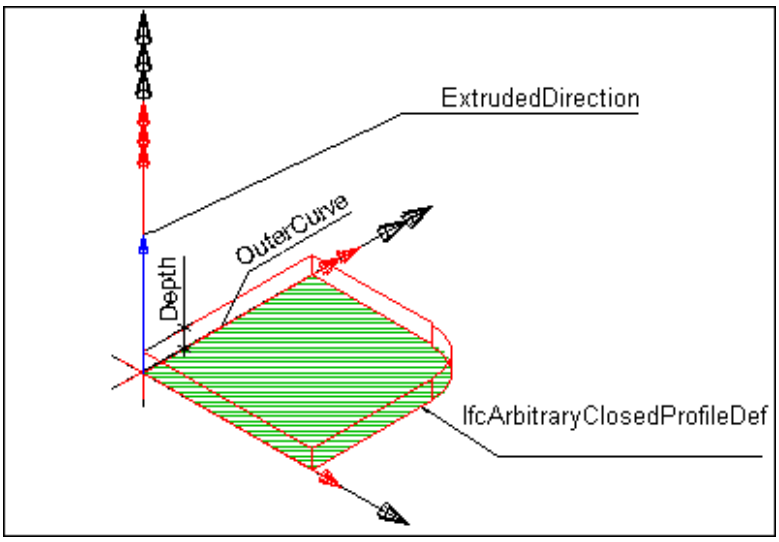
Endast några få av dessa byggnadselement kommer att användas i detta arbete, eftersom det är ett begränsat antal som är relevanta för strukturanalys och som är typiska för BIM-modeller av olika brokonstruktioner.

De byggnadselement som är intressanta för detta arbete och ingår i studerade modeller presenteras nedan.

Platta – IfcSlab

De bärande plattorna i en konstruktion definieras som *IfcSlab* och dessa är normalt horisontella. För en vanlig byggnad används andra objekt för att definiera golv, tak med mera för att särskilja att *IfcSlab* syftar främst till bärande del. Objektet kräver ett särskilt attribut *SlabType* som beskriver hur plattan är lokaliserad; horisontellt, vertikalt eller lutande.

Det finns olika sätt för hur *IfcSlab* är uppbyggd, vilket styrs av den geometriska representationen. För de bromodeller vi har studerat används *SweptSolid* och *Brep*, och de geometriska objekten definieras med *IfcExtrudedAreaSolid* och *IfcFacetedBreb*. Figur 19 visar en extrudering av *IfcSlab*.



Figur 19 - IfcSlab enligt IFC2x3 TC1 (buildingSMART 2007)

IfcArbitraryClosedProfileDef – Tvärsnitt som extruderas
Depth – Längden för extrudering
ExtrudedDirection – Riktning för extrudering

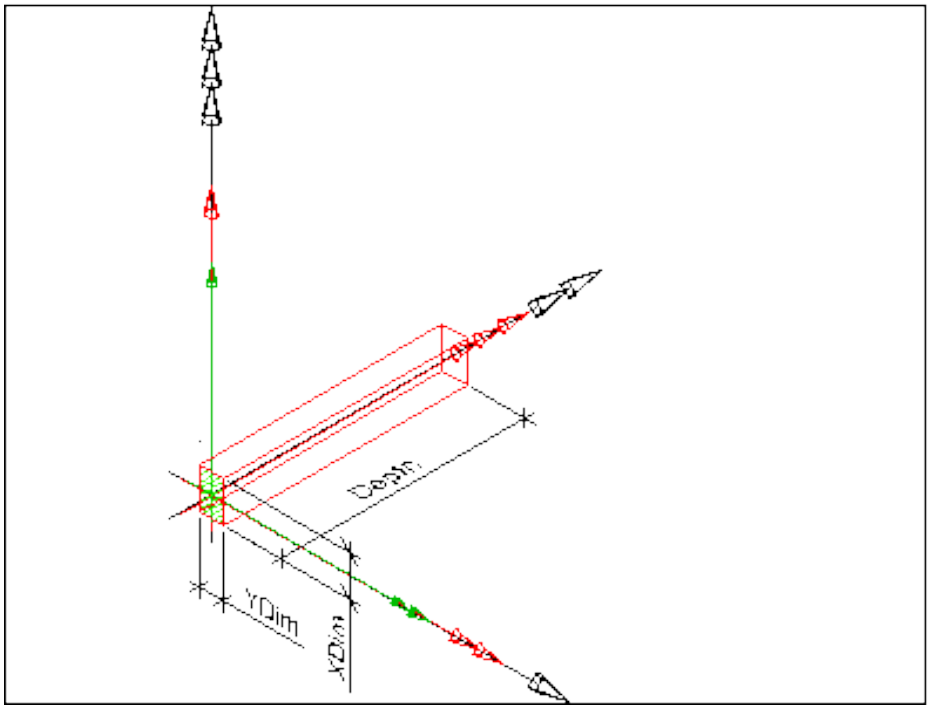
Platta – IfcPlate

Detta objekt är avsett för plattor som oftast, men inte nödvändigt, är mindre och har en konstant tjocklek. Det har oftast en bärande funktion men detta är inte nödvändigt. Objektet är väldigt likt *IfcSlab* till attributen, förutom att detta objekt inte kräver ett attribut som liknar *SlabType*.

De geometriska representationer som används för *IfcPlate* studerade bromodeller är *SweptSolid* och *brep,* och definitionen av de geomtriska objekten har gjorts med *IfcExtrudedAreaSolid* och *IfcFacetedBreb*.

Balk – IfcBeam

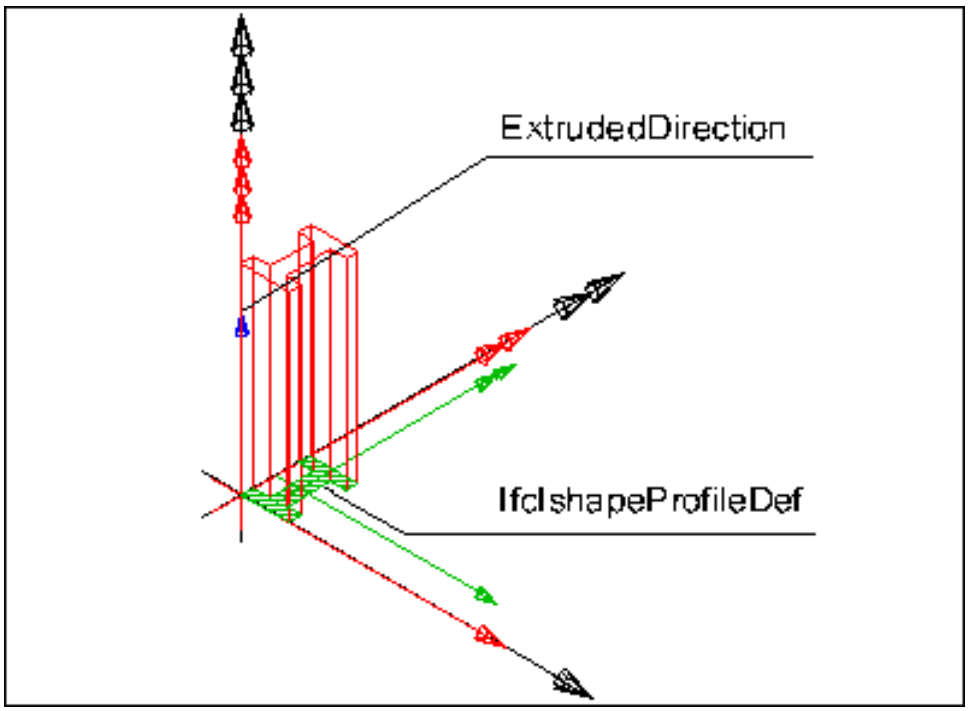
Detta objekt avser ett konstruktionselement som oftast har små tvärsnittsdimensioner i förhållande till sin längd, och är ett bärande element i konstruktionen. Först och främst används det för horisontella element, men detta har egentligen ingen direkt betydelse. Det finns olika sätt att geometriskt representera en balk inom IFC, men även här är det *SweptSolid* och *brep* som används i studerade bromodeller. Principen för extrudering av en balk kan ses i Figur 20. Detta exempel visar ett tvärsnitt definierat med måtten *XDim* och *YDim*, men istället kan tvärsnittet vara en definerad känd profil.



Figur 20 - IfcBeam enligt IFC2x3 TC1 (buildingSMART 2007)

Pelare – IfcColumn

Information och beskrivning för pelare skiljer sig endast semantiskt från den för balkar. Skillnaden är att det strukturella elementet först och främst syftas till att vara vertikala, men det finns inget attribut som kräver detta. I Figur 21 visas en pelare som är har ett definierat tvärsnitt med *IfcIshapeProfileDef* vilket är extruderat längs en riktning *ExtrudedDirection*. Det finns även här en längd som inte syns i figuren.



Figur 21 - IfcColumn enligt IFC2x3 TC1 (buildingSMART 2007)

I Tabell 2 visas vilka attribut som ingår i de vanligaste byggnadselementen samt från vilka IFC-objekt de ärver attributen ifrån.

Tabell 2 - Attribut för byggnadselement och vilka objekt de ärvs från

IFC-objekt för attribut	Attribut	IFC-objekt för byggnadsdel			
		IfcSlab	IfcPlate	IfcBeam	IfcColum
IfcRoot	GlobalId	X	X	X	X
	OwnerHistory	X	X	X	X
	Name	X	X	X	X
	Description	X	X	X	X
IfcObject	ObjectType	X	X	X	X
IfcProduct	ObjectPlacement	X	X	X	X
	Representation	X	X	X	X
IfcElement	Tag	X	X	X	X
IfcSlab	PredefinedType	X			

Metod

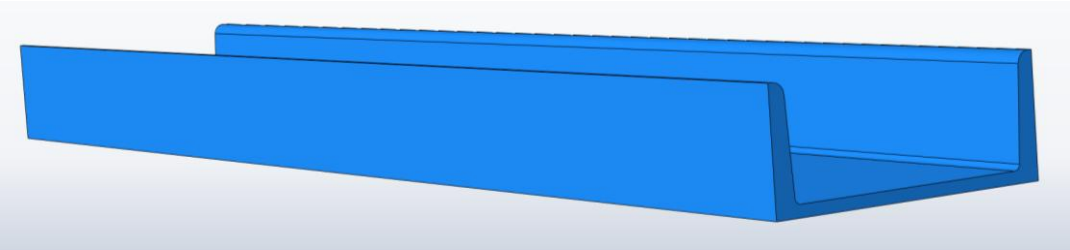
Sedan tidigare finns det stöd i BRIGADE/Plus för import av IFC-filer. De delar som är intressant för det här arbetet är import av geometri och material. Importen av material kommer att ske på samma sätt som tidigare med undantag av hur det implementeras i *Section* i BRIGADE/Plus. Importen av geometri fokuserar på att så många geometrier som möjligt ska importeras som skal- eller balkelement. Begränsning ligger i att importerade skal- och balkelement måste avspegla den ursprungliga geometrin väl och importen ska bara ske då det är relevant med skal- och balkelement.

Geometri – IfcExtrudedAreaSolid

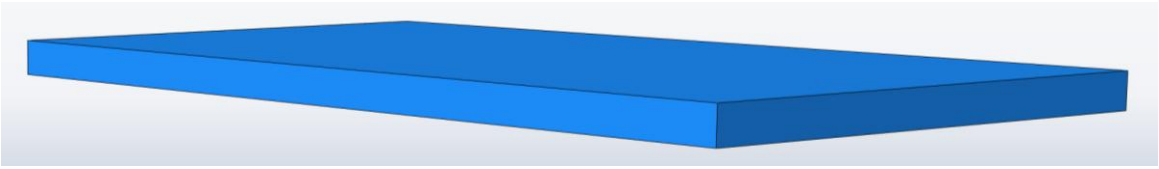
Geometriska objekt definierade med *IfcExtrudedAreaSolid* importeras idag som solida objekt genom att den tillhörande IFC-profilen ritas upp i BRIGADE/Plus verktyg *sketch*. Tvärsnittet extruderas sedan till ett solitt objekt med hjälp av angiven extruderingslängd i *IfcExtrudedAreaSolid*. Geometrin i IFC-filen är redan diskretiserad vid exportering, vilket gör att geometrin i BRIGADE/Plus blir exakt samma som den i IFC-filen.

Eftersom *IfcExtrudedAreaSolid* består av en profil och en extruderingslängd tolkas det lätt som en balk, så är dock inte alltid fallet. Ett exempel är en rektangulär form där långsidans yta är definierad som tvärsnitt och extruderingslängden är kortsidans längd. Den geometri som bildas kan då beskrivas bäst som en platta eller ett block.

Figur 22 visar exempel på en geometri som bör förenklas till ett balkelement och Figur 23 när förenkling till skalelement är att föredra.



Figur 22 - Geometri som bör förenklas till ett balkelement



Figur 23 - Geometri som bör förenklas till ett skalelement

Geometri – IfcFacetedBrep

Geometriska objekt som definieras av *IfcFacetedBrep* importeras som solida objekt genom att alla dess tillhörande IFC-ytor (*IfcFace*) skapas i BRIGADE/Plus. Ytorna sätts samman och volymen mellan dem skapar en solid. Denna import medför att geometrin i BRIGADE/Plus i princip blir samma som i IFC-filen. Det kan skilja något mellan geometrierna då BRIGADE/Plus anpassar ytorna efter behov för att kunna skapa en solid. Exporten från ett BIM-program till IFC är troligtvis mycket bra då varje yta i modellen får en geometrisk definition. Exporten kan dock skilja från program till program beroende på hur modellen exporteras. Även avancerade geometrier som diskretiseras vid exporten kan skilja lite i IFC-filen. Precis som med *IfcExtrudedAreaSolid* är det inte givet med vilket geometriskt objekt som elementet bör representeras.

Kontroll av elementtyp

Det framgår inte av *IfcExtrudedAreaSolid* och *IfcFacetedBrep* vilken typ av geometri (balk, skal eller solid) de representerar och därför behövs någon form av geometrisk kontroll. Det är troligtvis tekniskt möjligt att skapa en mängd variabler och kontroller för att göra en bedömning av lämplig förenkling för ett byggnadselement. En sådan kontroll blir mycket avancerad där både utveckling och testning skulle ta mycket lång tid. Användaren förlorar även möjligheten att påverka förloppet och det bakomliggande hos förenklingen. En lösning är att utgå från definitionen av byggnadselementet (*IfcBeam*, *IfcPlate*, *IfcFooting* osv). Har CAD-operatören i BIM-programmet modellerat på ett korrekt sätt så bör detta ge en bra indikation på vilka element som ska få vilken geometri. Användaren får då direkt kontroll över förenklingen av byggnadselement.

Genom att utgå från definitionen av byggnadselementen blir processen enkel och förståelig samtidigt som det öppnar för en framtida utvecklig av BRIGADE/Plus, där användaren själv får möjlighet att välja hur byggnadselementen importeras. Genom att skapa en separat lista av alla byggnadselementen i IFC-filen blir det vid skapandet av varje byggnadselement i BRIGADE/Plus enkelt att hantera geometrin vid import.

Målet är att element med definitionen *IfcPlate* eller *IfcSlab* skall importeras som skalelement och element med definitionen *IfcBeam* eller *IfcColumn* skall importeras som balkelement.

Import av solidelement

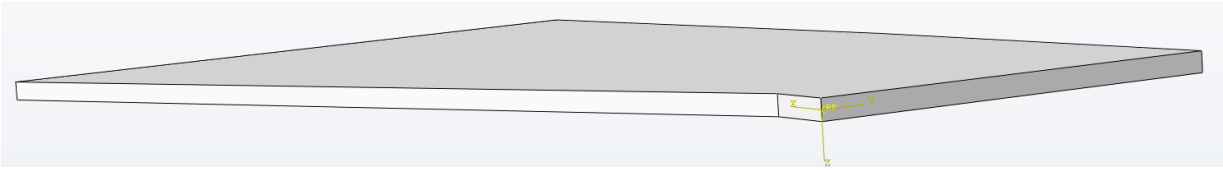
För import av byggnadselement som inte skall förenklas till skal- eller balkelement behöver ingenting ändras för importen. Ett element förblir solitt i fall det inte uttryckligen skall bli ett skal- eller balkelement. Om ett element ska importeras som skal- eller balkelement men misslyckas med detta eller inte uppfyller kraven för att bli skal/balk så görs ett försök att importera elementet som solidelement istället. Detta försök sker med samma metod för solidimport som tidigare.

Import av skalelement

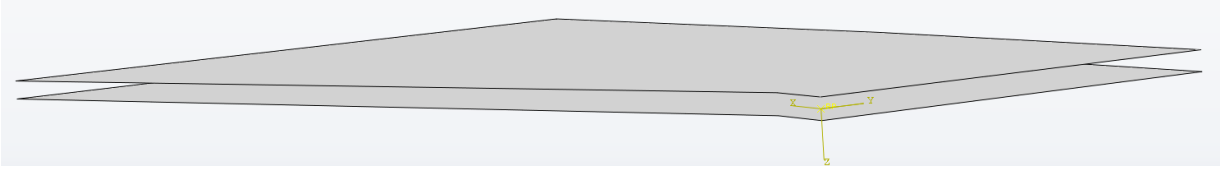
För att importen till skalelement ska bli tillfredställande så måste skalet kunna representera geometrin väl. Skalet ska ha samma tjocklek som den ursprungliga geometrin (tjockleken behandlas mer i detalj i avsnittet om *Section*). Skalet ska ha samma utbredning som den ursprungliga geometrin och det ska vara placerat i tyngdpunktsplanet av den ursprungliga geometrin.

IfcExtrudedAreaSolid som ska bli skal är oftast ett rektangulärt tvärsnitt och därmed ett ganska enkelt geometriskt objekt. Dock skulle exempelvis ihåliga bjälklag kunna representeras av *IfcExtrudedAreaSolid* och därmed bli svåra att modellera med skal. Vad gäller *IfcFacetedBrep* så kan dessa i princip se ut hur som helst, vilket gör att elementen är svåra att förutse och handskas med. Värt att notera är att en CAD-operatören har suttit och ritat all geometri i ett BIM-program. Såvida denna person inte har haft mycket stränga krav på sig är troligtvis stora delar av modellen ritad på ett så enkelt sätt som möjligt. De flesta element i modeller är därför ganska enkla och de som ska representeras av skal ser oftast ut som just skal. En perfekt lösning av importen för alla element med definition *IfcPlate* eller *IfcSlab* finns troligtvis inte. Istället har fokus lagts på att ta fram en realistisk lösning som fungerar i de flesta fall.

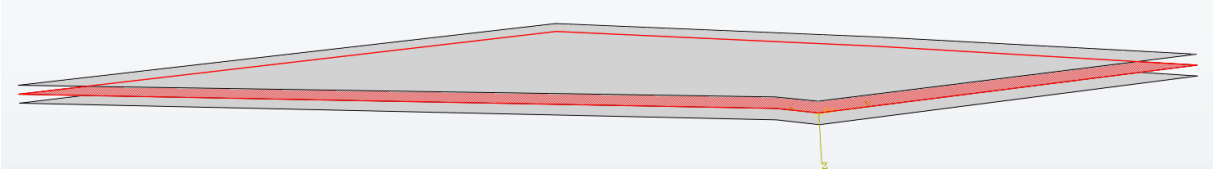
Ett skalelement har en dimension som är väsentligt mindre än de övriga två. Detta gör att de flesta geometrier som skall förenklas till skal består av flera mindre ytor och två stora ytor, se Figur 24. De stora ytorna används till att skapa skalets yta och den mindre dimensionen används till att skapa skalets tjocklek i *Section* (behandlas i avsnittet om *Section*). Med samma metod som tidigare importeras elementen som solid, därefter kontrolleras om de två största ytorna på soliden är lika stora samt parallella. Är dessa villkor uppfyllda är det stor chans att geometrin kan förenklas till ett skalelement. De två största ytorna sparas medan allt annat tas bort, se Figur 25. Med hjälp av ytorna skapas sedan med hjälp av Abaqus-kommandot *OffsetFaces*, en tredje yta mitt emellan dem, se Figur 26. När de två ytorna sedan tas bort blir det enda som är kvar en enda yta, den ligger i geometrins tyngdpunktsplan och har samma utbredning som de ursprungliga största ytorna. Ytan bildar sedan ett skal (behandlas i avsnittet om *Section*).



Figur 24 - Element som ska förenklas till ett skalelement

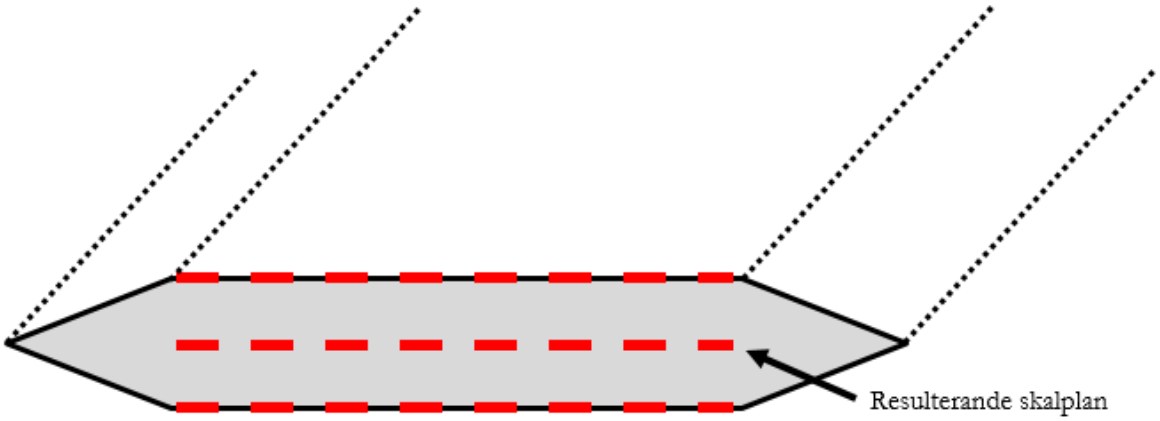


Figur 25 - Elementet med endast de största ytorna kvar



Figur 26 - Elementet med ny yta skapad i tyngdpunktsplanet

Metoden skapar i de flesta fall ett korrekt skal. De geometrier som inte är lämpade för den här metoden kommer i de flesta fall att inte uppfylla antingen storleken på de två största ytorna eller vinkeln mellan dem. Dessa geometrier kommer i så fall fortsätta vara solidelement. Det finns dock geometrier som uppfyller de två kriterierna men ej är lämpliga som skal. De kommer i så fall antingen att misslyckas med importen eller så skapas skal med felaktig utbredning. Ett exempel på ett element som får felaktig utbredning finns i Figur 27. De största sidorna är markerade längs kanterna och det resulterande nya skalet är markerat i mitten.

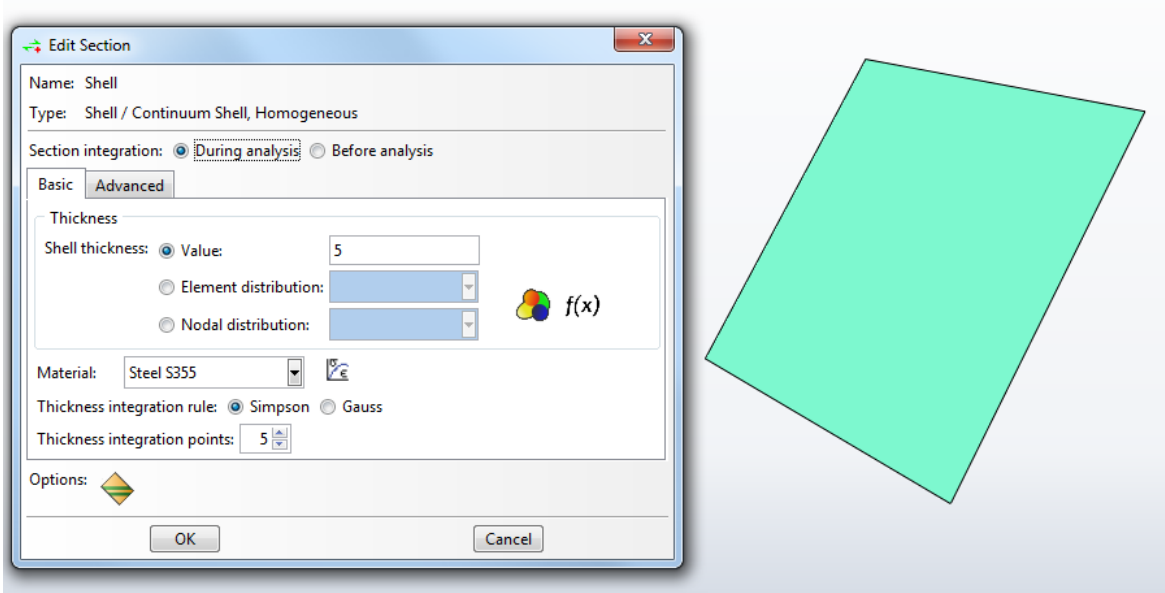


Figur 27 - Exempel på hur elementets största ytor inte representerar dess utbredning

Section

Ett skal blir inte komplett i BRIGADE/Plus utan en *Section*. I den befintliga importfunktionen med solidelement så tilldelas geometrin en *Section* (*Section Assignment* av en *Section*), vilken innehåller information om materialet, något som importeras från IFC.

För skal- och balkelement finns en ytterligare anledning till att ha en *Section* utöver att elementen tilldelas materialegenskaper. När en förenkling av den ursprungliga tredimensionella geometrin görs till tvådimensionella skal och endimensionella balkar så förloras dimensioner. För att kompensera för den förlorade dimensionen hos skal så innehåller dess *Section*, utöver material, även tjockleken på skalet. Denna information kan lätt hämtas då ytan till skalet skapas, nämligen avståndet mellan de två ytorna som används till *OffsetFaces*. Informationen kan sparas i en separat lista liknande den för definitionen av byggnadselementen. I Figur 28 finns ett exempel på ett skal och dess *Section*, skalet har fått materialet *Steel S355* och tjockleken 5 millimeter. Enheten för längdmått i IFC-filen är i det här fallet millimeter vilket gör att tjockleken i BRIGADE/Plus får samma enhet.



Figur 28 - Ett skal och dess Section

Import av balkelement

Importen av balkelement ser helt annorlunda ut jämfört med importen av skalelement. Geometrier som bygger på *IfcFacetedBrep* bortses från helt då det är alltför tekniskt komplicerat att ta fram en balk från dessa objekt. Bland annat anges inte ett tvärsnitt för dessa objekt. De flesta byggnadselement som är intressanta att importera som balkar är beskrivna med *IfcExtrudedAreaSolid*. En annan sak som är speciellt med balkar är att de måste ha en profil, det vill säga en tvärsnittsform. Vid import av balkar så ska alla sorters profiler kunna importeras (se vidare om profiler i avsnittet om *Beam Section*).

Definition med *IfcExtrudedAreaSolid* av byggnadselementen *IfcBeam* eller *IfcColumn* är i princip alltid tillfredställande att importera som balk. Det finns dock undantag, exempelvis om CAD-operatören i BIM-programmet använt ett balkelement och extruderat det en kort bit för att få en platta utan ändra objektnamnet. För importen av balkar så är det svårt att ha några kontroller likt de för skal. Har exporteringen från BIM-programmet skett på rätt sätt så är referenspunkten i *IfcExtrudedAreaSolid* samma som tyngdpunkten i profilen. Genom att i BRIGADE/Plus med Abaqus-kommandot *BaseWire* skapa en *Wire* (det som senare ska bli balken) från referenspunkten i *IfcExtrudedAreaSolid* med samma riktning och längd som *IfcExtrudedAreaSolid* så skapas geometrin till balken, både med rätt längd och i rätt position.

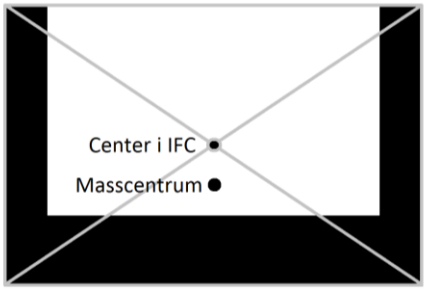
Beam Section

För balkar så ritas den förenklade geomtrin ut i BRIGADE/Plus som en linje, kallad *Wire*. Denna linje tilldelas ett tvärsnitt kallat *Beam Section* som utöver material även innefattar en tvådimensionell profil. Profilens tvärsnittsegenskaper väger upp för de två dimensioner som förlorats vid förenkling till balkelement.

Objekt definierade av *IfcExtrudedAreaSolid* som ska bli balkar har attributet *IfcProfileDef* som i de flesta fall är en standardprofil, kan vara exempelvis *IfcIShapeProfileDef*, *IfcUShapeProfileDef*, *IfcCircleProfile*. Dessa definieras av parametrar som bredd, vinkel och andra nödvändiga mått. Det finns även specialprofiler som *IfcArbitraryClosedProfileDef*, vilka bestäms av linjer av *IfcPolyLine*.

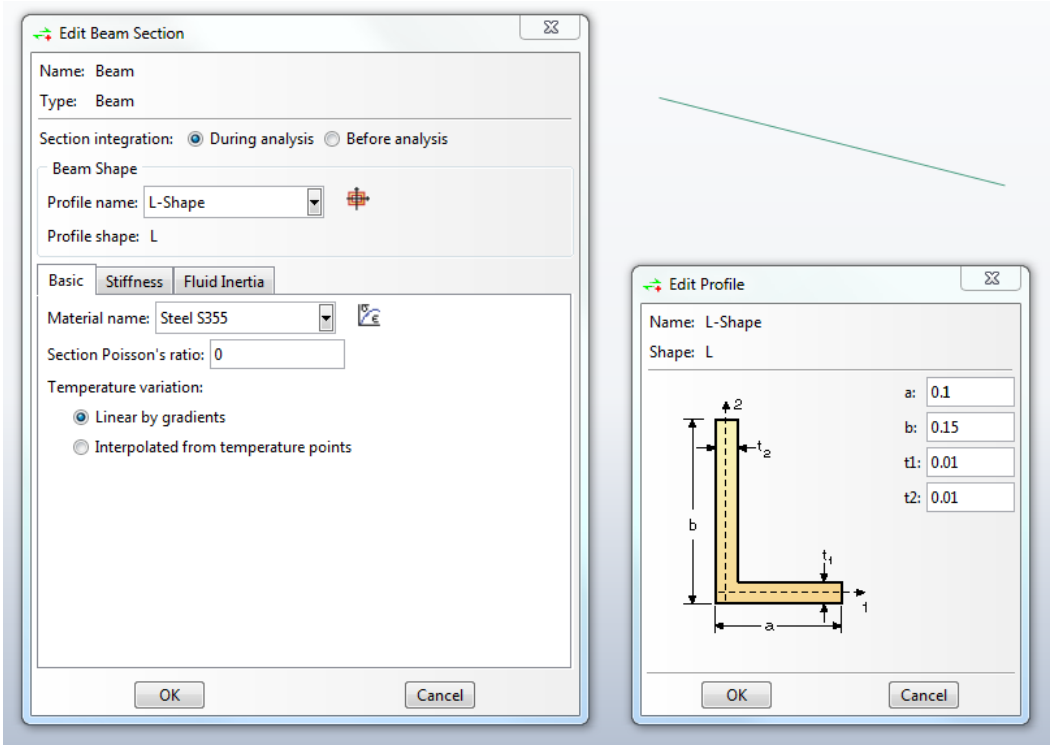
Profiler i BRIGADE/Plus kan ganska enkelt skapas för de flesta standardprofilerna i IFC då de har samma eller liknande parametrar. En profil kan även i skapas som en *Arbitrary Profile* i BRIGADE/Plus, vilket är en form av polygontåg där varje del har sin egen längd, tjocklek och riktning. Det enda undantaget för parametriserade profiler är *IfcEllipseProfileDef* som inte har någon motsvarighet i BRIGADE/Plus. För IFC-profilen *IfcArbitraryClosedProfileDef* är det svårare att skapa en korrekt motsvarande profil i BRIGADE/Plus, då de är uppbyggda på ett sådant sätt att det blir tekniskt komplicerat. En lösning för dessa profiler är att importera dem som *Generalized Profile* i BRIGADE/Plus. Detta är en lösning där inga parametrar för profilens geometri behövs utan istället ska area, tröghetsmoment med mera anges. Det främsta problemet här är att inga värden för detta går att få ut ur IFC-filen, lösningen blir då att importera på detta vis och att användaren själv får beräkna och mata in dessa värden.

I IFC så utgår alla standardprofilers värden och placering från mitten av ”omslutningslådan”, se Figur 29, och inte masscentrum. För många profiler spelar detta inte någon roll eftersom de är symmetriska och de två centra då sammanfaller, eller så kan de anpassas vid skapandet av profilen. De profiler där centra inte sammanfaller är för är L-, T- och asymmetriska I-profiler.



Figur 29 - Omslutningslådan i IFC

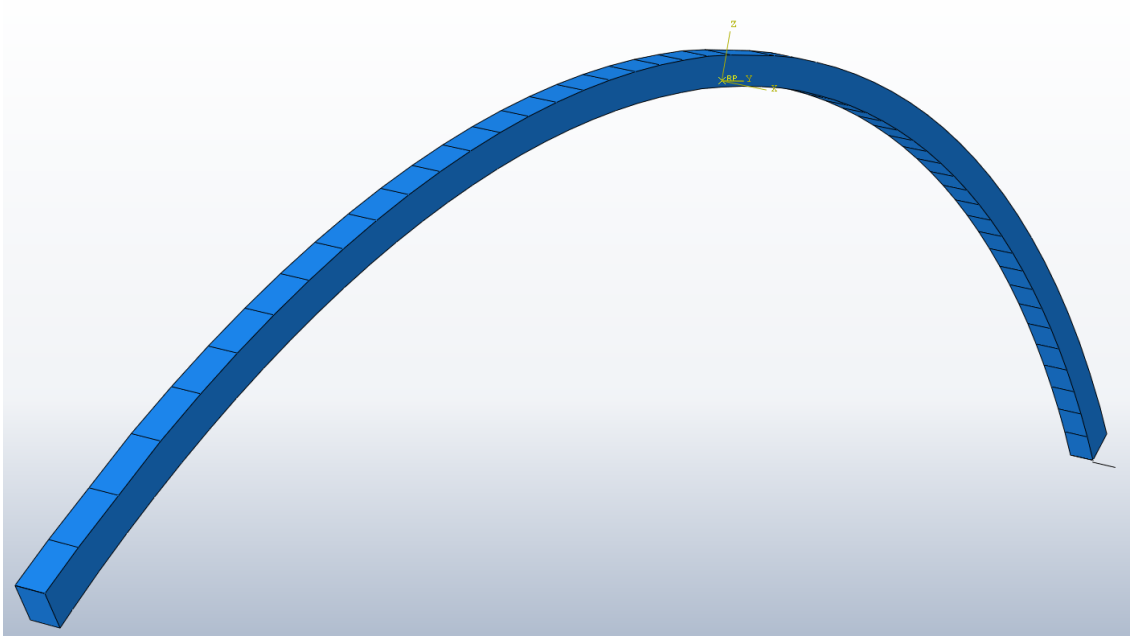
Genom att på samma sätt som tjockleken för skal sparas i en lista så kan det för balkar sparas namn på den tillhörande profilen. När geometrin får sin *Section/Beam Section* senare i importen kan respektive material, tjocklek och profil enkelt hämtas för varje byggnadselement. I Figur 30 finns ett exempel på en balk och dess *Beam Section*. Balken har fått materialet *Steel S355* och profilen *L-shape* vars egenskaper är importerade och syns till höger.



Figur 30 - En balk, dess Beam Section och dess L-profile

Import av avancerad geometri (bågar)

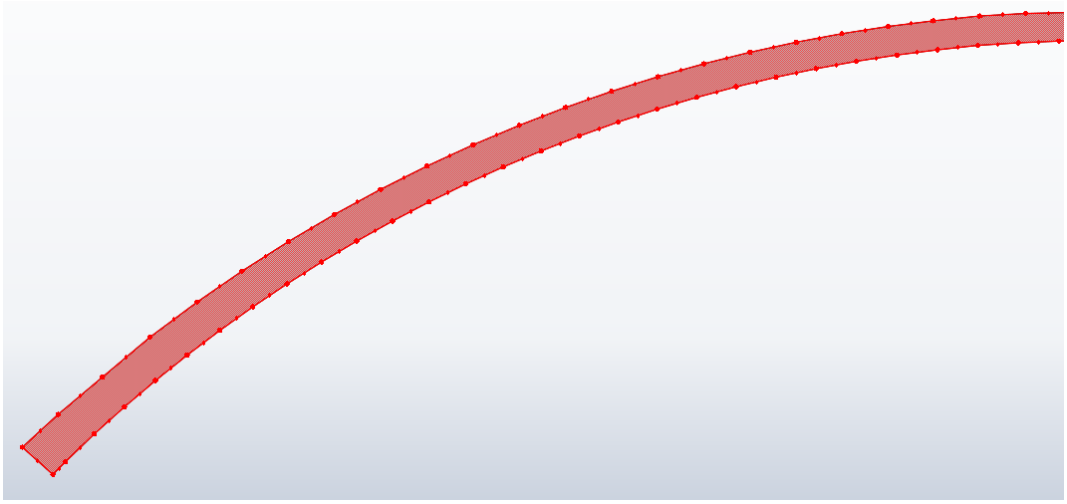
Utöver de element som ska importeras som skal- och balkelement enligt tidigare angiven metod så är nästan alla andra geometrier för avancerade eller för ovanliga för att det ska vara intressant att möjliggöra skal-/balkimport av dessa. Det är däremot förhållandevis vanligt att bågar ingår i brokonstruktioner, det hade därför varit till hjälp om dessa importerades som krökta balkelement. De bågar som studerats har uteslutande definierats med *IfcFacetedBrep*. Dessa består av en större mängd mindre ytor, se Figur 31. Detta är diskretisering av geometrin som görs vid exportering till IFC.



Figur 31 - En båge uppbyggd av många små ytor

Det enklaste sättet att handskas med bågformer är att definiera krav som kan särskilja bågar från övriga geometrier. Ett krav för att förenkla till ett balkelement är att byggnadselementet är *IfcBeam*. Diskretisering har även skapat en stor mängd ytor, rimligtvis fler än många andra byggnadselements diskretiseringar. Krav har då ställts på att objektet skall innehålla ett högt antal ytor, inom detta arbete 30 stycken. Uppfylls båda delar så är det stor sannolikhet att den geometriska formen är en båge.

När kontrollen är godkänd så skapas först en mittenyta mellan de två största ytorna, enligt samma metod som vid importering av skalelement. De två sidorna på bågar är de största ytorna och om de är parallella så kommer en yta att skapas mellan dessa, se Figur 32. För mittytan kontrolleras det vilka kanter som är längst, här definitionen av en kant ”de linjer som sammanbinds med ingen eller liten vinkel mellan normlariktningarna”. Vinkeln mellan de korta linjernas normaler i ovan- respektive undersidan är maximalt några få grader, se Figur 32.



Figur 32 - En båge omgjord till en yta

Mitt emellan punkter på ovansidan och närmsta punkt på undersidan skapas referenspunkter, det vill säga längs mitten av ytan, för vilka koordinaterna sparas i en lista. När hela ovansidan är genomförd skapas ändkoordinater mitt på bågens ändar för att göra en så bra anpassning som möjligt i de fall då ändarna på bågen är avancerade. När listan med koordinater är komplett raderas mittenytan och ett balkelement skapas som ett polygontåg genom alla koordinater i listan. Detta gör att det slutgiltiga balkelementet inte är böjt exakt likadant som den ursprungliga bågen utan består av flera små raka balkelement.

På grund av den avancerade geometri som bågarna har så är det för avancerat att ta fram en profil för bågarna utifrån IFC-filen. Det behövs även en kontroll för om bågen är krökt i fler än en riktning.

Resultat

För att på ett enkelt sätt hantera vilka byggnadselement som ska bli solid-, skal- respektive balkelement så sparas ID-nummer och typ för dessa i en global *Python-dictionary* innan geometrin importeras. En *Python-dictionary* lämpar sig för detta då den innefattar nyckelord med tillhörande värden. Nedan följer ett exempel på hur informationen för tre olika element sparas.

IFC-fil:
#94= IFCBEAM('Test1',#5,'BALK1','PL300','PL300',#87,#93,'5125');
#699= IFCPLATE('Test2',#5,'PLATTA1','PL10','PL10',#577,#698,'5126');
#3464= IFCFOOTING('Test3',#5,'GRUND1','PL20','PL20',#2808,#3463,'5127');

Python-dictionary:
{'94': 'IfcBeam', '699': 'IfcPlate', '3464': 'IfcFooting'}

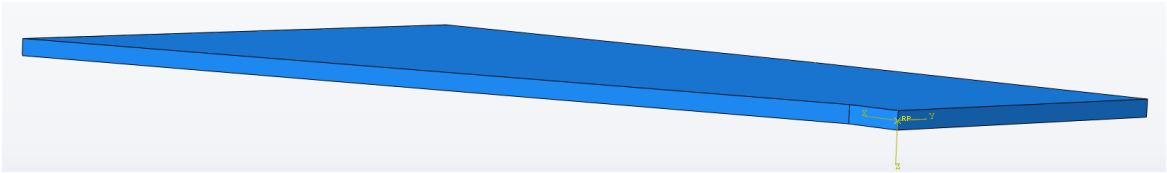
En bedömning av hur byggnadselement normalt sätt bör importeras finns i Tabell 3.

Tabell 3 - Importsätt för byggnadselement

IFC-element	Import	Kommentar
IfcBeam	**Balk**	
IfcDoor	Solid	Ej bärande
IfcColumn	**Balk**	
IfcWindow	Solid	Ej bärande
IfcMember	Solid	För avancerad, upp till användaren att avgöra
IfcCurtainWall	Solid	Ej bärande
IfcWall	Solid	För avancerad, upp till användaren att avgöra
IfcRoof	Solid	För avancerad, upp till användaren att avgöra
IfcSlab	**Skal**	
IfcRamp	Solid	För avancerad, upp till användaren att avgöra
IfcPlate	**Skal**	
IfcPile	Solid	För avancerad, upp till användaren att avgöra
IfcRailing	Solid	Ej bärande
IfcRampFlight	Solid	För avancerad, upp till användaren att avgöra
IfcStair	Solid	Ej bärande
IfcStairFlight	Solid	Ej bärande
IfcFooting	Solid	Bör vara solid
IfcCovering	Solid	Ej bärande
IfcBuiildingElementProxy	Solid	För avancerad, upp till användaren att avgöra
IfcBuildingElementComponent	Solid	För avancerad, upp till användaren att avgöra

Import av skal – IfcExtrudedAreaSolid

Då *IfcExtrudedAreaSolid* skapar solidelement genom extrudering längs en rak linje skapas i de flesta fall en ganska enkel geometri. Detta gör att det förhållandevis ofta går lätt att göra om geometrin till skal och skalen som skapas är exakta kopior av den största ytan på soliden. Då det är bekräftat att denna yta är lika stor som och parallell mot en annan yta i soliden så är det nästan uteslutande korrekta plattor som blir skalelement. Tjockleken på skalen tas från avståndet mellan de två största ytorna. Om skalet fått en korrekt geometri så stämmer tjockleken bra överens med verkligheten. I Figur 33 finns ett exempel på en platta som är *IfcExtrudedAreaSolid* vilken i Figur 34 har importerats som skalelement.



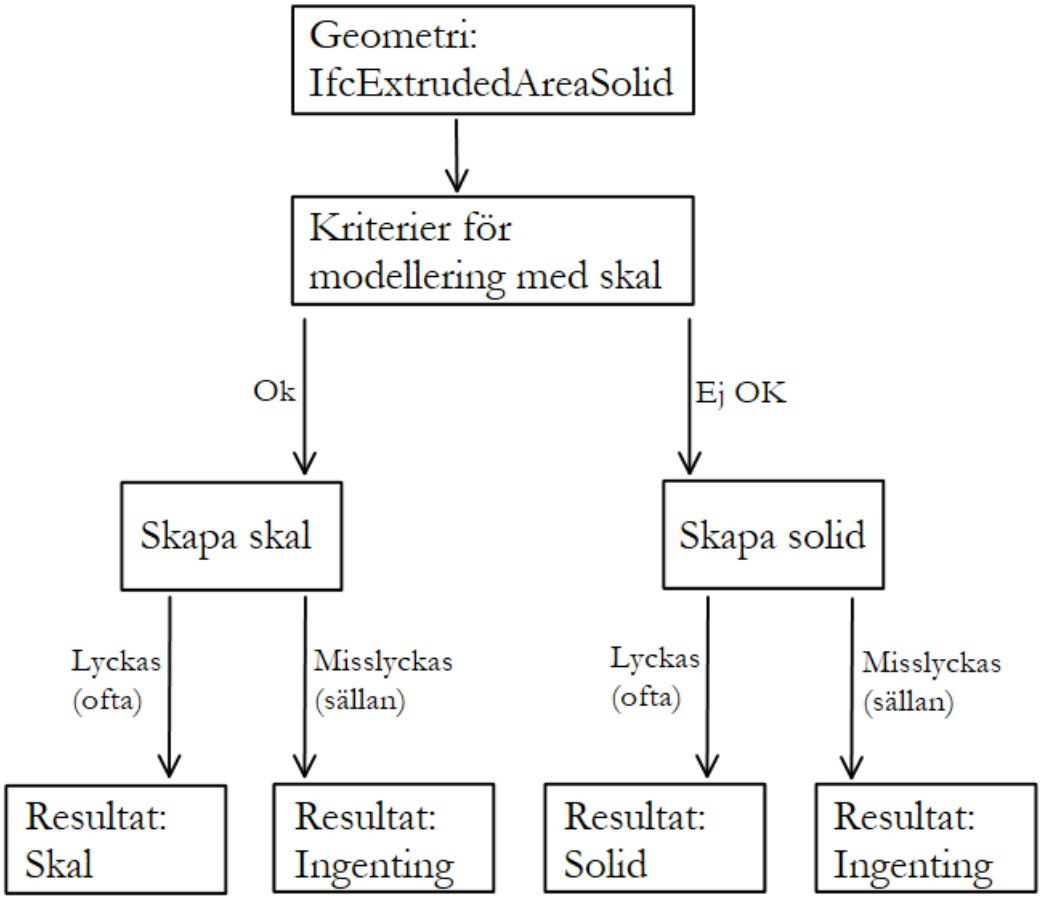
Figur 33 - Element konstruerat som IfcExtrudedAreaSolid importerat som solidelement

Figur 34 - Element konstruerat som IfcExtrudedAreaSolid importerat som skalelement

I de fall då processen misslyckas med att skapa ett skal finns det två möjliga utgångar. Den första är att de två största ytorna inte är lika stora eller att de inte är parallella. I detta fall så behålls soliden vilket gör att användaren tydligt kan se vad som misslyckats med att bli skal samt att elementet fortfarande finns med i modellen och går att använda. Den andra möjligheten är att processen misslyckas med själva skapandet av skalytan, exempelvis på grund av att en av ytorna inte är platt. Det senare misslyckandet kan antingen resultera i att det som tidigare förblir en solid eller att geometrin blir oanvändbar och inte används i modellen. Det senare misslyckandet är dock ovanligt.

Det finns fall då användaren behöver vara uppmärksam på resultatet av importen. Exemplet från metodkapitlet med ett ihåligt bjälklag som kan vara problematiskt skulle kunna lyckas att bli skal då de största ytorna kan vara lika stora och parallella. Resultatet av detta skulle i så fall skapa ett skal med överskattad tjocklek. Det kan också finnas situationer där geometrin underskattas om de största ytorna inte motsvarar plattans utbredning.

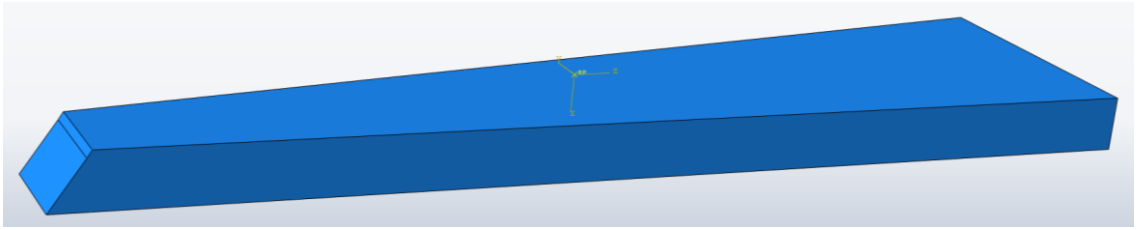
I Figur 35 finns ett schema över processen för skapandet av geometrin då *IfcPlate* eller *IfcSlab* har geometrin *IfcExtrudedAreaSolid*.



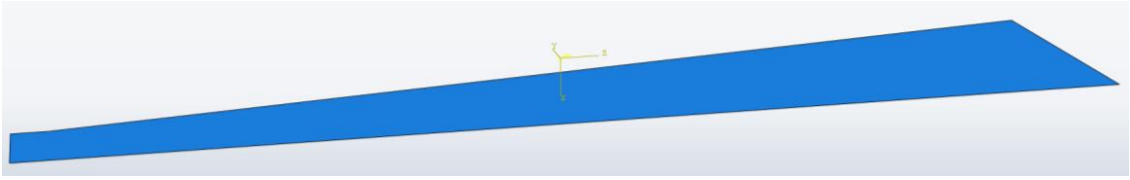
Figur 35 - Process för att skapa skal av IfcExtrudedAreaSolid

Import av skalelement – IfcFacetedBrep

Den mer komplexa geometrin i *IfcFacetedBrep* än i *IfcExtrudedAreaSolid* påverkar inte resultatet i någon större utsträckning. Det är fortfarande en stor andel av geometrierna som är tillräckligt enkla och anpassade för att bli skal. Samma kontroller görs för skalen här som i föregående fall och samma goda resultat ges vad gäller tjockleken och storlek/placering av skalet. I Figur 36 finns ett exempel på en platta som är *IfcFacetedBrep*, vilken i Figur 37 har importerats som skalelement med kravet på att de två största ytorna ska vara lika stora och att de ska vara parallella har godkänts med en viss avvikelse.



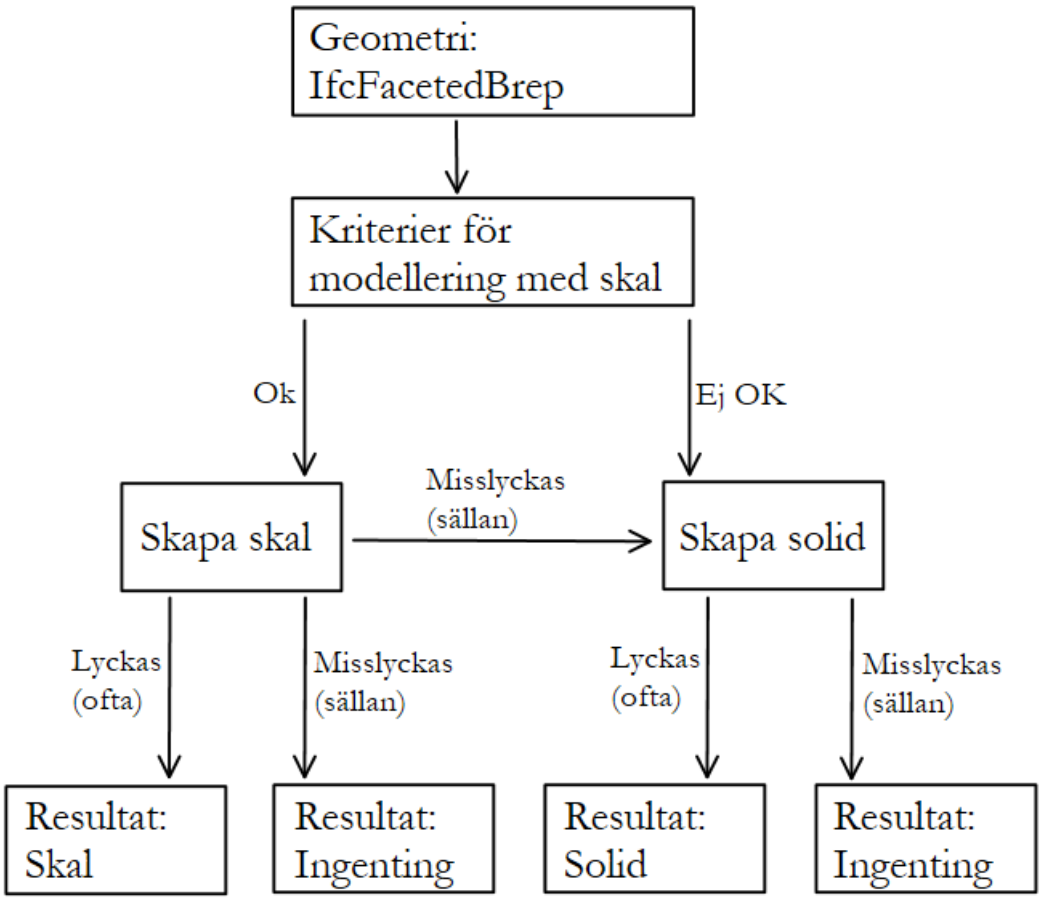
Figur 36 - Element konstruerat som IfcFacetedBrep importerat som solidelement



Figur 37 - Element konstruerat som IfcFacetedBrep importerat som skalelement

Misslyckande av att skapa skal kan ske även med *IfcFacetedBrep*, dessa skiljer sig dock något från de med *IfcExtrudedAreaSolid*. Soliden bibehålls i de fall då de största ytorna inte är lika stora eller parallella. Om processen däremot misslyckas med själva skapandet av skalytan så kommer objektet att misslyckas med att bli både skal och solid och därmed, precis som med *IfcExtrudedAreaSolid*, bli oanvändbar och inte används i modellen.

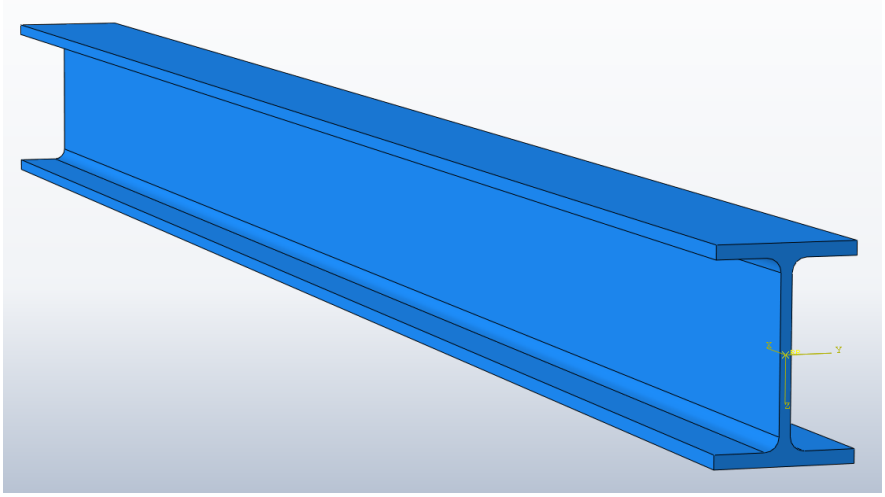
I Figur 38 finns ett schema över processen för skapandet av geometrin då *IfcPlate* eller *IfcSlab* har geometrin *IfcFacetedBrep*.



Figur 38 - Process för att skapa skal av IfcFacetedBrep

Import av balkelement

IfcExtrudedAreaSolid är mycket lämpad för att skapa balkar och resultatet av tillämpningen är bra. Utgångspunkt, riktning och längd för balken överförs mycket enkelt och blir alltid helt korrekt. I Figur 39 finns ett exempel på en balk som är *IfcExtrudedAreaSolid*, vilken i Figur 40 har importerats som balkelement.

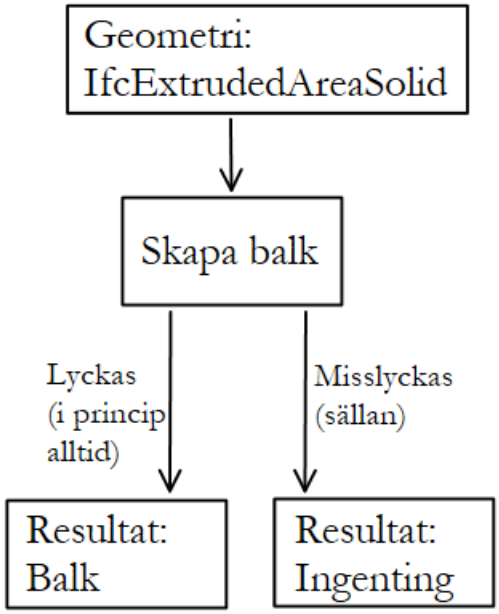


Figur 39 - Element konstruerat som IfcExtrudedAreaSolid importerat som solidelement

Figur 40 - Element konstruerat som IfcExtrudedAreaSolid importerat som balkelement

Det finns en begränsning som relaterar till extruderingsriktningen. I *IfcExtrudedAreaSolid* måste inte extruderingsriktningen vara vinkelrät mot profilen. I importen så antas den vara det, vilket gör att för alla fall då den inte är vinkelrät så blir importen felaktig och profilen överskattas. Om en CAD-operatör skapar en balk med speciella ändar genom att rita en vanlig balk och sen plocka bort bitar (*Clipping* i IFC), så kan det bli problem genom att den importeras som hel balk och sedan misslyckas processen med *Clipping*. Även detta medför att balken överskattas.

I Figur 41 finns ett schema över processen för skapandet av geometrin då *IfcBeam* eller *IfcColumn* har geometrin *IfcExtrudedAreaSolid*.

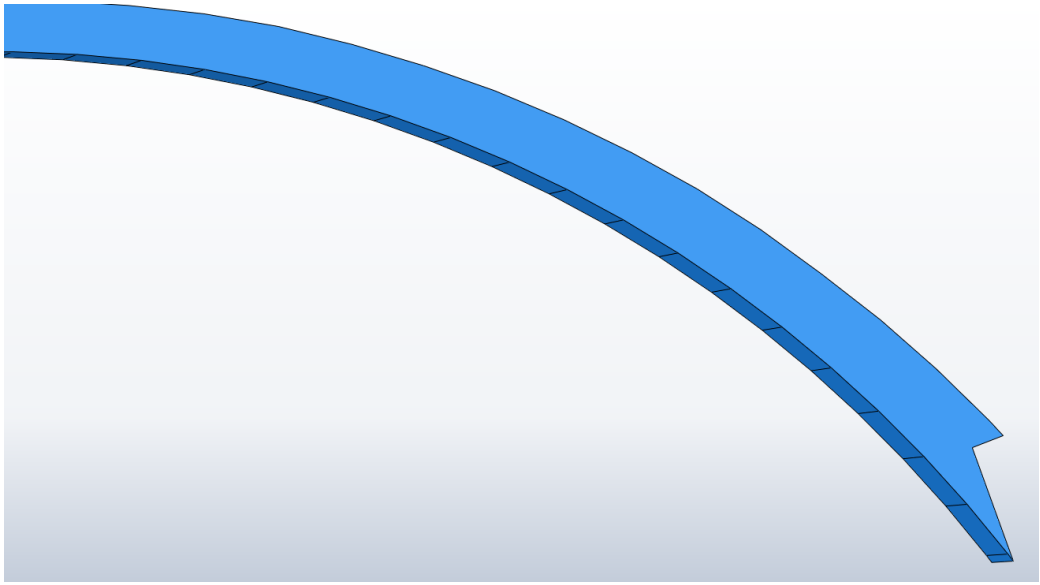


Figur 41 - Process för att skapa balk av IfcExtrudedAreaSolid

Import av avancerad geometri (bågar)

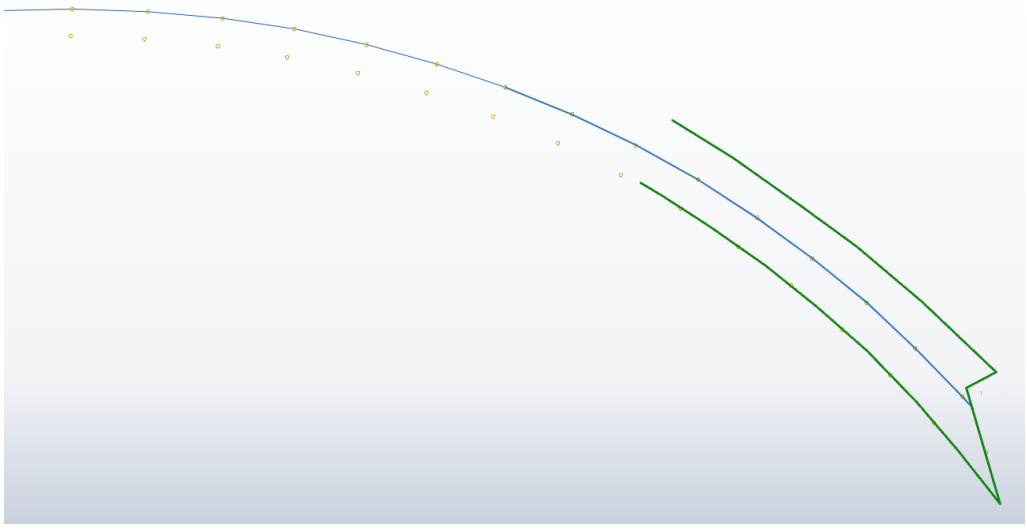
Importen av bågar är den mest avancerade biten av det här arbetet vilket innebär att det också är den del som är mest osäker. Eftersom en båge kan ritas på en mängd olika sätt så är det är svårt att sätta något värde på hur ofta importen kommer att lyckas vid praktisk användning, hur ofta den misslyckas och hur ofta den försöker göra balkimport av något som inte ens är en båge. Av de studerade modellerna har det visat sig att det endast är ett fall där det funnits element som är *IfcBeam* och har många ytor men som inte borde vara en båge, medan det funnits flera element som borde vara bågar.

Importen bygger på att elementet först importeras likt skalelement vilket innebär att alla problem som kan uppstå vid skalimport även kan uppstå här. Om en yta skapas så finns det troligtvis en del som kan gå fel med skapandet av balkelement. På grund av otillräcklig testning finns det inget som garanterar att importen är bra, dock är resultatet bra på de geometrier som testats. De problem som uppstått gör att inte ens skalimporten lyckas, detta medför att elementen blir oanvändbara och inte används i modellen. I Figur 42 finns ett exempel på en del av en båge som är *IfcFacetedBrep* vilken i Figur 43 visas med skalimport och i Figur 44 har importerats som balkelement. I Figur 44 visas även konturerna från skalelementet för att det ska vara möjligt att se hur resultatet blir.



Figur 42 - Del av båge importerad som solidelement

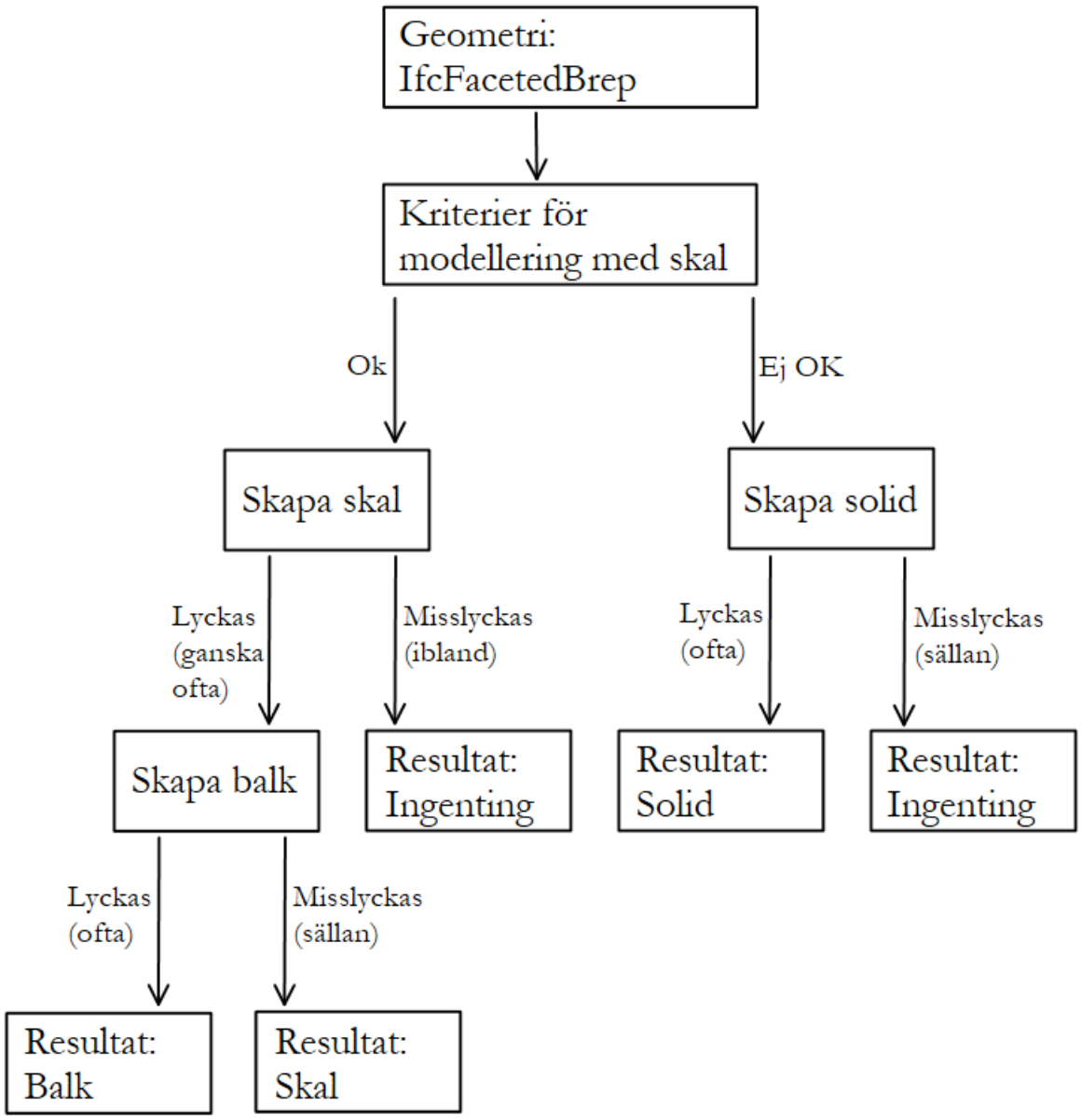
Figur 43 - Del av båge importerad som skalelement



Figur 44 - Del av båge importerad som balkelement (med konturer från skalelementet för visualisering)

Eftersom ingen profil skapas för dessa element så kan inte heller någon *Beam Section* skapas.

I Figur 45 finns ett schema över processen för skapandet av geometrin då *IfcBeam* har geometrin *IfcFacetedBrep* och många ytor.



Figur 45 - Process för att skapa balkelement av båge

Section

Information om vad för typ av *Section* som varje element ska ha sparas i en global *Python-dictionary*. Till att börja med sätts alla värden till ”*Solid*” för att sedan ändras om geometrin lyckas importeras som en yta (skalelement) eller en *Wire* (balkelement). Blir geometrin för ett element en yta så ändras värdet för elementet till skalets tjocklek och om geometrin blir en *Wire* så ändras värdet till namnet på balkens profil. När varje element sedan ska tillsättas en *Section* så skapas dessa utefter vad för information som finns för respektive element. Material, vilket ingår i alla *Section*, är separat relaterat till varje element och importeras på samma sätt som tidigare. Om värdet för ett element är ”*Solid*” så skapas en *Solid Section* med rätt material. Om värdet är ett tal så skapas en *Shell Section* med rätt material och tjocklek motsvarande talet. Om värdet varken är ”*Solid*” eller ett tal så skapas en *Beam Section* med rätt material och den profil som heter samma sak som värdet. Nedan följer ett exempel på hur informationen för tre olika element sparas.

IFC-fil:
#94= IFCBEAM(’Test1’,#5,'BALK1','PL300','PL300',#87,#93,'5125');
#699= IFCPLATE(’Test2’,#5,'PLATTA1','PL10','PL10',#577,#698,'5126’);
#3464= IFCFOOTING('Test3',#5,’GRUND1’,'PL20','PL20',#2808,#3463,'5127‘);

Python-dictionary:
{'94': ‘ID_47-PL300‘, '699': ’10.0’, '3464': ‘Solid’}

För att göra det så enkelt som möjligt för användaren att förstå importen så får respektive *Section* namn efter vad för information som finns för elementet och vad för material det har. Exempel på detta finns i Tabell 4.

Tabell 4 - Exempel på Section för olika element

Element	Värde i dictionary	Materialnamn	Section-namn
IfcBeam	ID_47-PL300	Steel/S355	Beam-ID_47-Steel/S355
IfcPlate	10.0	Steel/S355	Plate-10-0-Steel/S355
IfcFooting	Solid	C25/35	Footing-Solid-C25/35

Beam Section

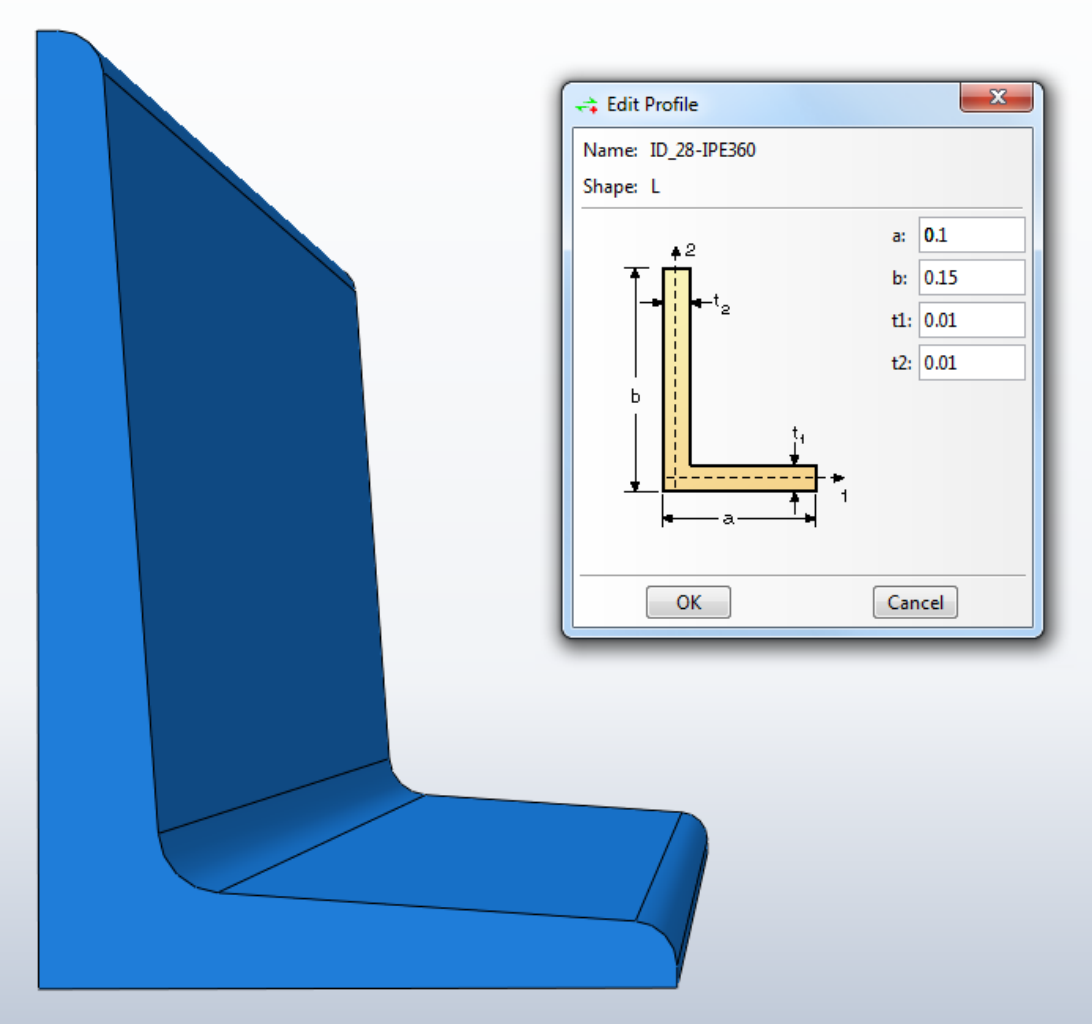
Profilen för respektive balk skapas samtidigt som dess *Wire*. I många fall är det bredd, höjd och tjocklek på profilens delar som överförs till BRIGADE/Plus. Standard-profilerna i BRIGADE/Plus är ofta mindre detaljerade än IFC-profilerna och viss data går därför förlorad. Resultatet är varierande beroende på profil och detaljnivå i IFC-filen, i Tabell 5 finns resultatet av profilimporten.

Tabell 5 - Resultat av profilimport

IFC-profil	BRIGADE/Plus-profil	Kvalité	Kommentar
IfcArbitraryClosedProfileDef	Generalized Profile	Dålig	Inga egenskaper
IfcArbitraryProfileDefWithVoids	Generalized Profile	Dålig	Inga egenskaper
IfcAsymmetricIShapeProfileDef	I Profile	Bra	Ignorerar rundade hörn Masscentrum ≠ IFC-center
IfcCircleHollowProfileDef	Pipe Profile	Perfekt	
IfcCircleProfileDef	Circular Profile	Perfekt	
IfcCShapeProfileDef	Arbitrary Profile	Mycket bra	Ignorerar rundade hörn
IfcEllipseProfileDef	Generalized Profile	Dålig	Inga egenskaper
IfcIShapeProfileDef	I Profile	Mycket bra	Ignorerar rundade hörn
IfcLShapeProfileDef	L Profile	Ok	Ignorerar rundade hörn Ignorerar vinklar på ändar Masscentrum ≠ IFC-center
IfcRectangleHollowProfileDef	Box Profile	Mycket bra	Ignorerar rundade hörn
IfcRectangleProfileDef	Rectangular Profile	Perfekt	
IfcRoundedRectangleProfileDef	Rectangular Profile	Mycket bra	Ignorerar rundade hörn
IfcTShapeProfileDef	T Profile	Ok	Ignorerar rundade hörn Ignorerar vinklar på ändar Masscentrum ≠ IFC-center
IfcUShapeProfileDef	Arbitrary Profile	Bra	Ignorerar rundade hörn Ignorerar vinklar på ändar
IfcZShapeProfileDef	Arbitrary Profile	Mycket bra	Ignorerar rundade hörn

L-, T- och assymetriska I-profiler importeras med standardprofiler i BRIGADE/Plus vilket gör att deras masscentrum hamnar i balkelementets punkter. Balkelementet i sin tur är importerat efter dess placering i IFC-filen vilket är mitten på ”omslutningslådan” av IFC-profilen. Detta gör att profilerna hamnar något fel. De profiler som inte passar för någon av de vanliga BRIGADE/Plus-profilerna och därmed måste bli *Generalized Profile* importeras som detta men utan egenskaper (för att kunna skapa profilen så sätts dess egenskaper till 0 eller 1). Det är då viktigt att användaren inser att denne själv måste ändra egenskaperna, för närvarande skrivs ett varningsmeddelande ut i *Message Area*.

För de IFC-profiler där det finns en vinkel på ändarna (exempelvis båda delar i en L-profil) så skapas en profil i BRIGADE/Plus utan denna vinkel, dock är area för de båda profilerna densamma. Då denna vinkel oftast är mycket begränsade blir effekten liten. För profiler med rundade hörn så har hörnen helt bortsetts från, det vill säga skarpa hörn har använts. Detta medför att inre hörn (exempelvis svetsar i I-balkar) bortses från, man blir då ”på den säkra sidan”. Det medför också att yttre hörn (exempelvis ändar på en L-balk) räknas med, man räknar alltså med material som inte finns. Den area som gör att man hamnar ”på den osäkra sidan” är mycket liten, men kan vara värd för användaren att ha vetskap om. En balk med L-profil har importerats som både solid och balk. I Figur 46 syns profilen vid solidimport samt i rutan värden för profilen vid balk-import. Notera att värden för balken som har importerats är överdrivna för att tydliggöra utseendet.



Figur 46 - Balk importerad som solidelement till vänster och dess profil vid balkimport till höger

Utöver profilerna i tabellen så finns det ett antal andra profiler i IFC vilka är mer eller mindre oväsentliga och har därför ej implementerats.

För att göra det så enkelt som möjligt för användaren att förstå importen så får respektive Profil namn efter dess IFC-ID och namnet på profilen i IFC-filen (*IfcLabel*). Detta gör att om flera olika element har samma profil så skapas endast en version av profilen och endast en version av *Section* (om de också har samma material vill säga). Nedan följer exempel på IFC-profiler och motsvarande profilnamn i BRIGADE/Plus.

IFC-profil:
#3764= IFCUSHAPEPROFILEDEF(.AREA.,'UPE200',#43,200.5,80.5,6.5,11.5,13.5,4.5,2.5,$);
#3806= IFCRECTANGLEPROFILEDEF(.AREA.,'PL12*142.5',#43,12.,142.5);

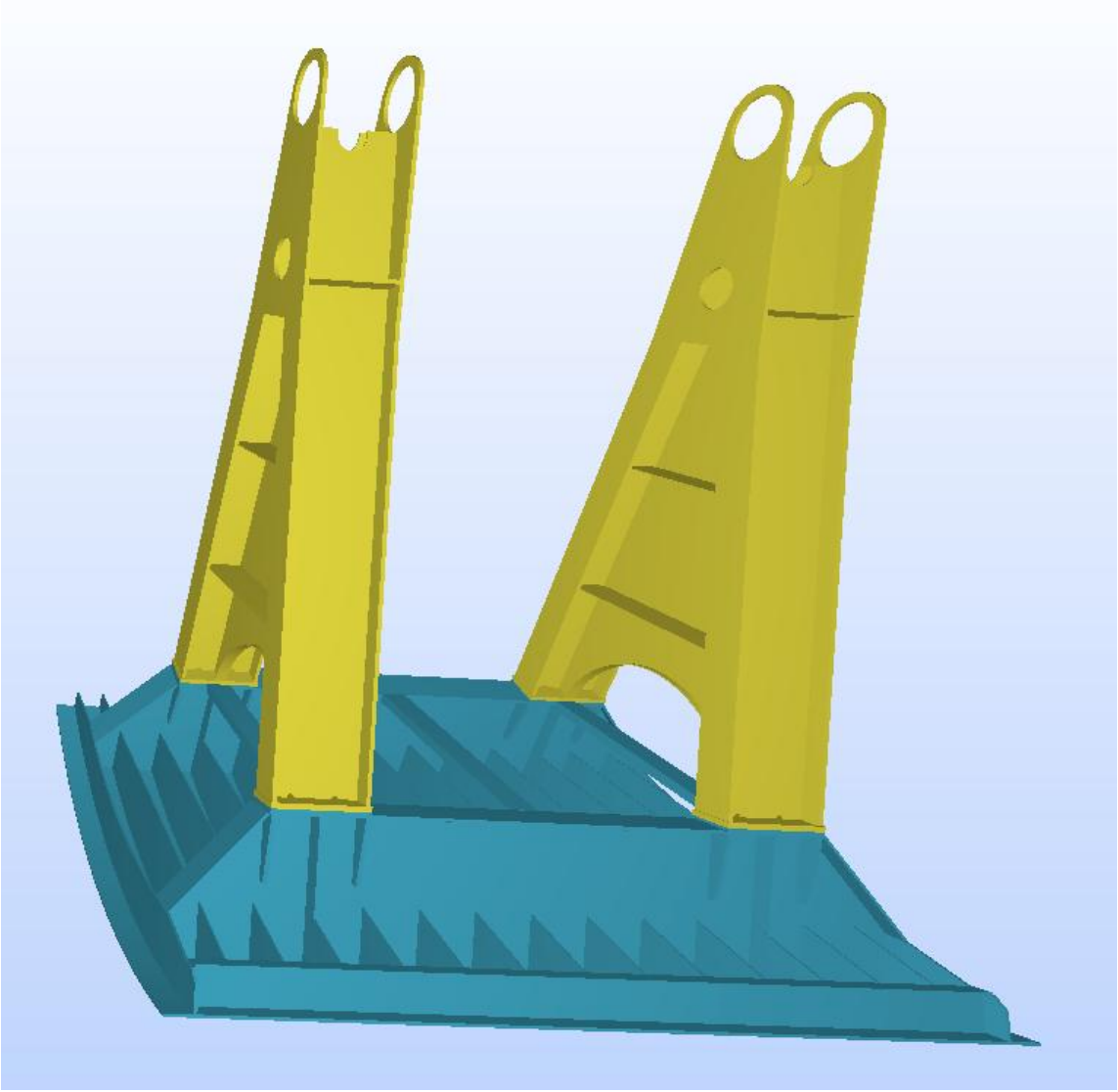
Profilnamn i BRIGADE/Plus:
ID_3764-UPE200
ID_3806-PL12*142.5

Tidsåtgång

Idag tar import av IFC-filer i BRIGADE/Plus förhållandevis lång tid, självklart beroende på storlek, komplexitet och datorkraft. En fil med en komplett bro kan ta timmar att importera och dammluckan i nästa avsnitt tar med en inte alltför långsam dator cirka en halvtimma att importera. En uppdatering av Abaqus är på väg vilket gör att importen kommer att bli snabbare. Det som tar tid i importen är framförallt ritandet av alla ytor i *IfcFacetedBrep*, ju fler ytor ett element består av desto längre tid tar det. De förändringar som gjorts för att *IfcFacetedBrep* ska importeras som skal bygger på att alla ytor först ritas upp vilket gör att importen fortfarande tar lika lång tid. Visserligen har det tillkommit delar i importen men det nya tar mycket liten tid i förhållande till ritandet av ytor. Även om en import tar flera timmar så är den extra tidsåtgången för att göra om element till skal och balkar minimal. Resultatet är att tidsåtgången är i princip samma som tidigare.

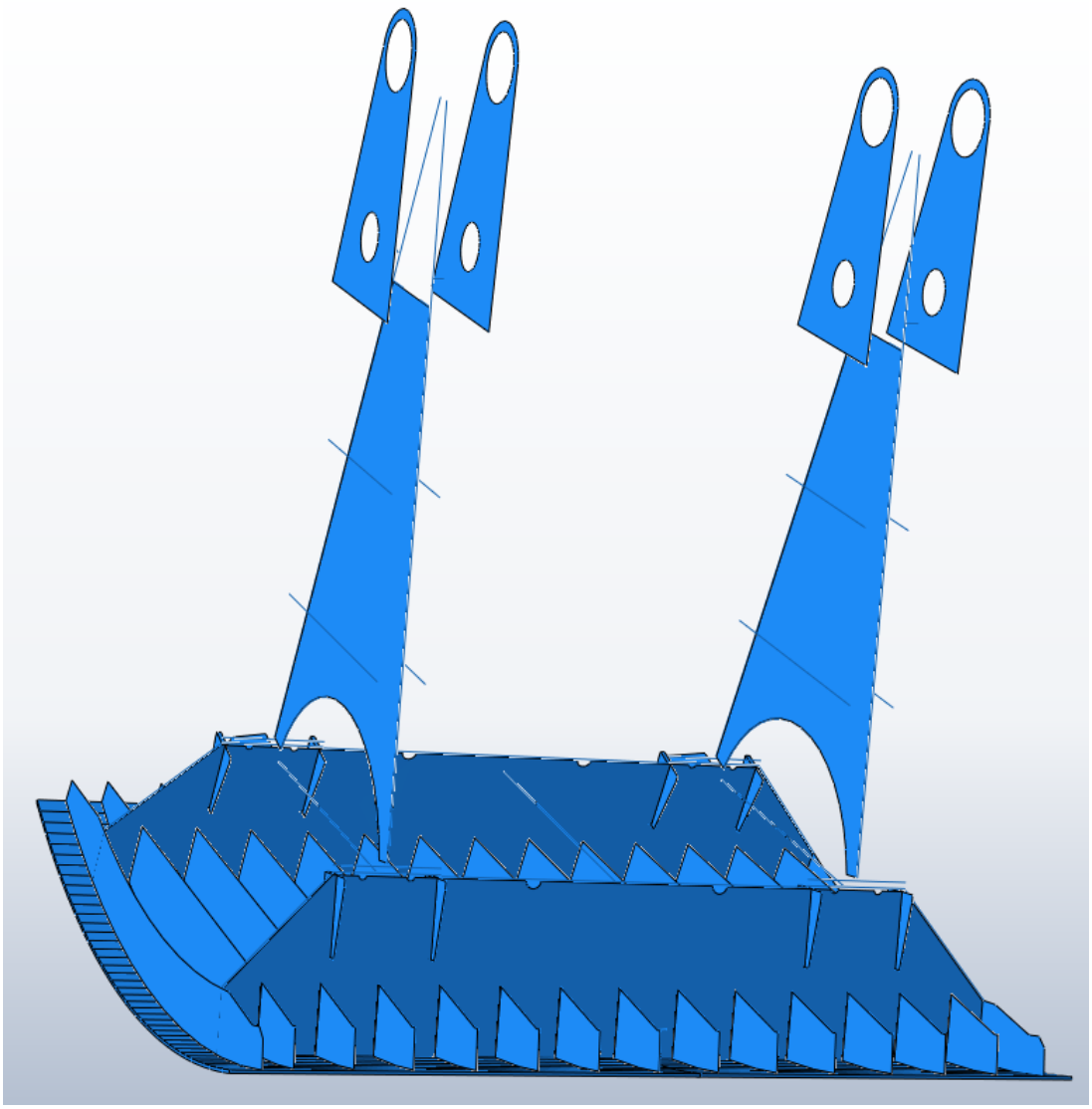
Helhetsresultat

Det finns fortfarande många element som inte importeras på ett korrekt sätt. Den övervägande delen av alla element kan dock importeras tillräckligt bra för att importen ska ses som lyckad vad gäller enskilda element. När det gäller hela modeller så finns det modeller som importeras helt perfekt. Många av de modeller som använts till testning består dock bland annat av plattor som är definierade som *IfcBeam* och därmed importeras som balkar samt balkar som är definierade som *IfcSlab* och därmed importeras som plattor. Efter flera tester är det tydligt att IFC-filen och dess element beror mycket på hur CAD-operatören har ritat. CAD-operatören kan ha gjort helt rätt ur dennes perspektiv men definitioner av element blir konstiga ur BRIGADE/Plus-perspektiv. En av WSP Vattenbyggnads modeller av en dammlucka används för att visa exempel på import, den visas som BIM-modell i Figur 47.



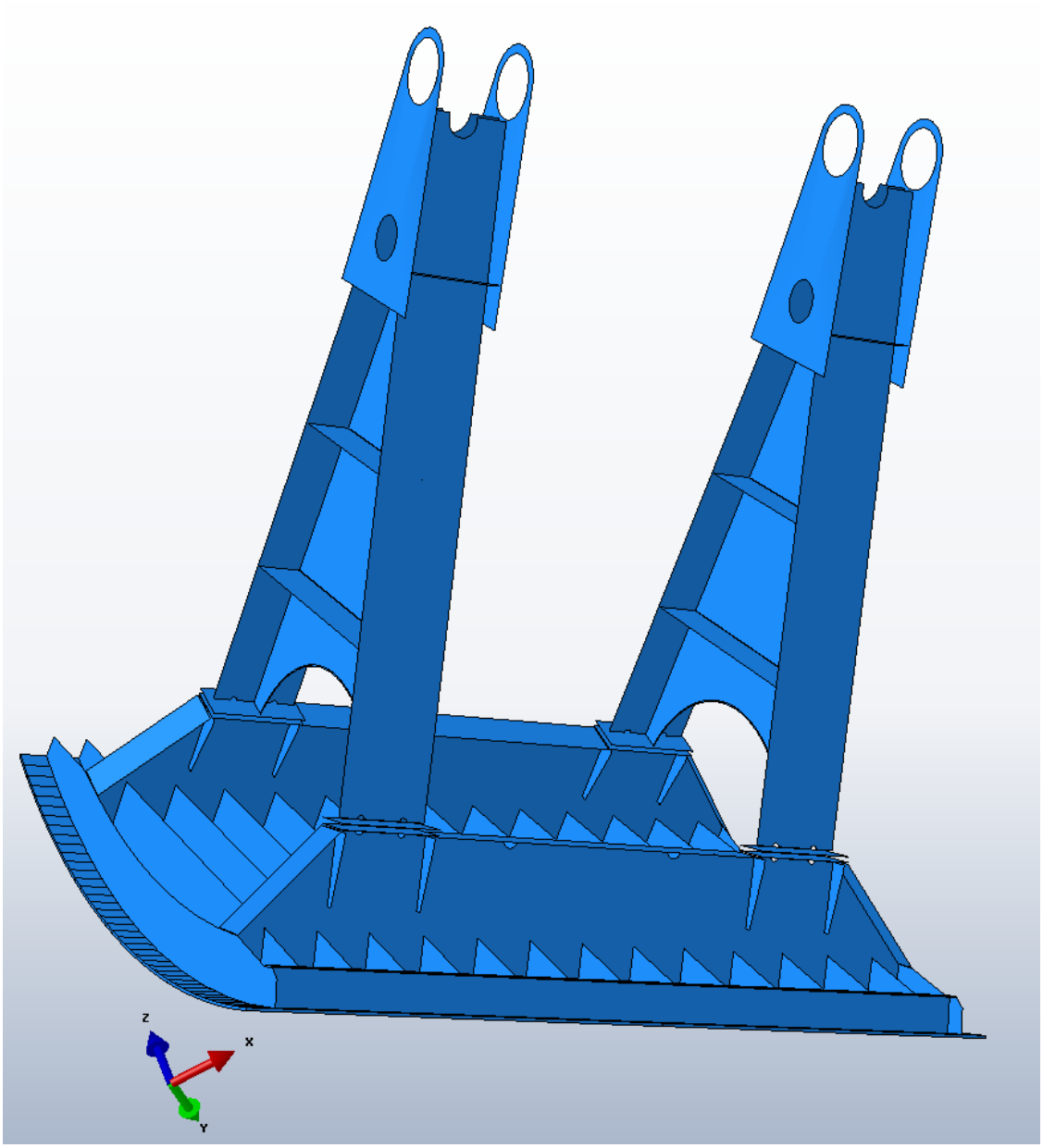
Figur 47 - Visualiserad BIM-modell

Så länge användaren inte har möjlighet att välja vilka element som ska importeras som vad, så blir importen helt baserad på CAD-operatörens ritteknik, något som inte alltid blir optimalt, se Figur 48. Många av elementen i detta exempel är definierade som balkelement och importeras som detta, det hade dock varit bättre med nästan uteslutande skalelement.



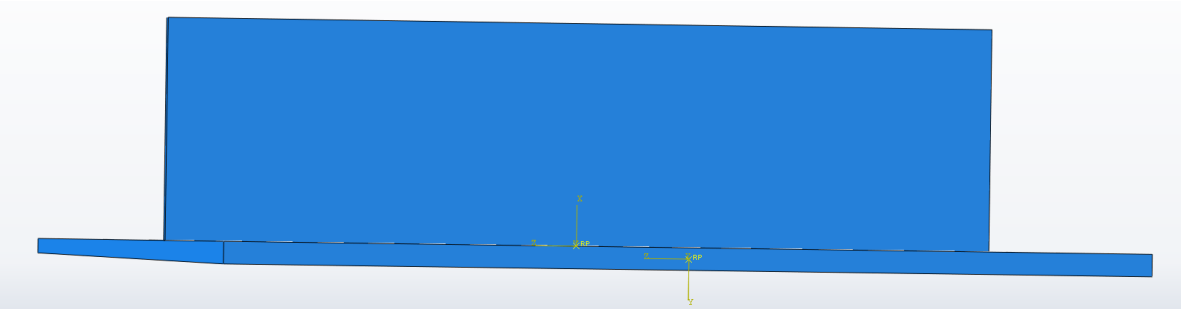
Figur 48 - Modell importerad med skal- och balkelement

Genom att gå in och simulera ett aktivt val av användaren (genom att ändra elementdefinition i IFC-filen), skapas en ny import med enbart skalelement, se Figur 49. Resultatet visar på goda möjligheter till bra resultat vid import om användaren får möjligheten till ett val. Det är fortfarande vissa element som inte inkluderas i importen då de består av IFC-objekt som ännu inte är implementerade i BRIGADE/Plus.

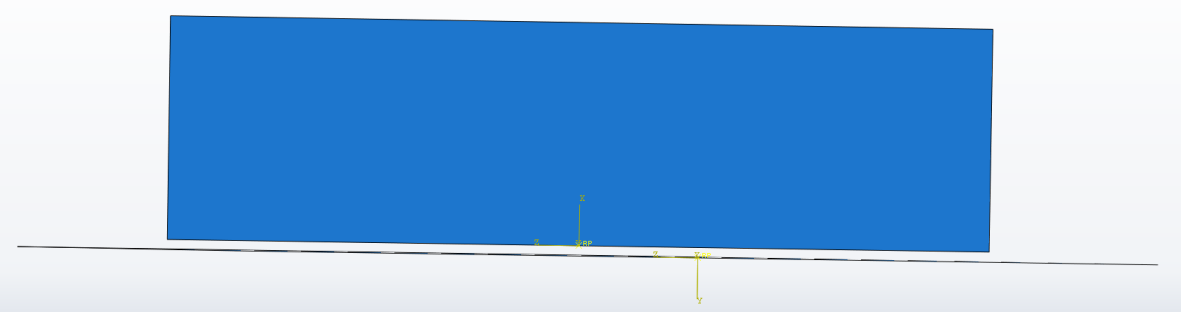


Figur 49 - Modell importerad med endast skalelement

När ett skal skapas istället för en solid så bildas ett tomrum på båda sidor om skalet. För det enskilda elementet så gör detta ingen skillnad eftersom det har en definierad tjocklek, men för modellen i sin helhet så innebär detta att närliggande element får ett avstånd om halva skalets tjocklek till skalet, se solidelement i Figur 50 och skalelement i Figur 51. På liknande sätt så får allting ett visst avstånd även till balkar. Dessa avstånd gör att användaren få anpassa sitt sätt att hantera interaktionen mellan olika delar och ta hjälp av den riktiga BIM-modellen för att förstå hur delarna sitter samman samt även vilka element som faktiskt sitter samman.



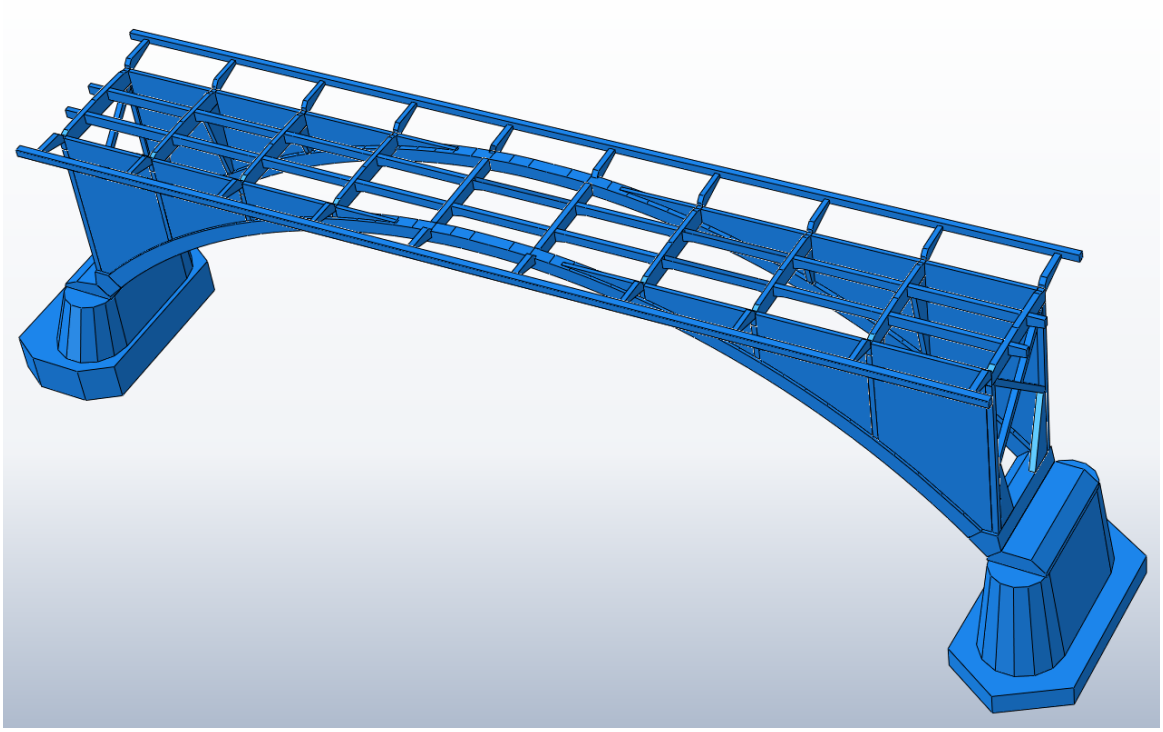
Figur 50 - Två element importerade med solidimport



Figur 51 - Två element importerade med skalimport

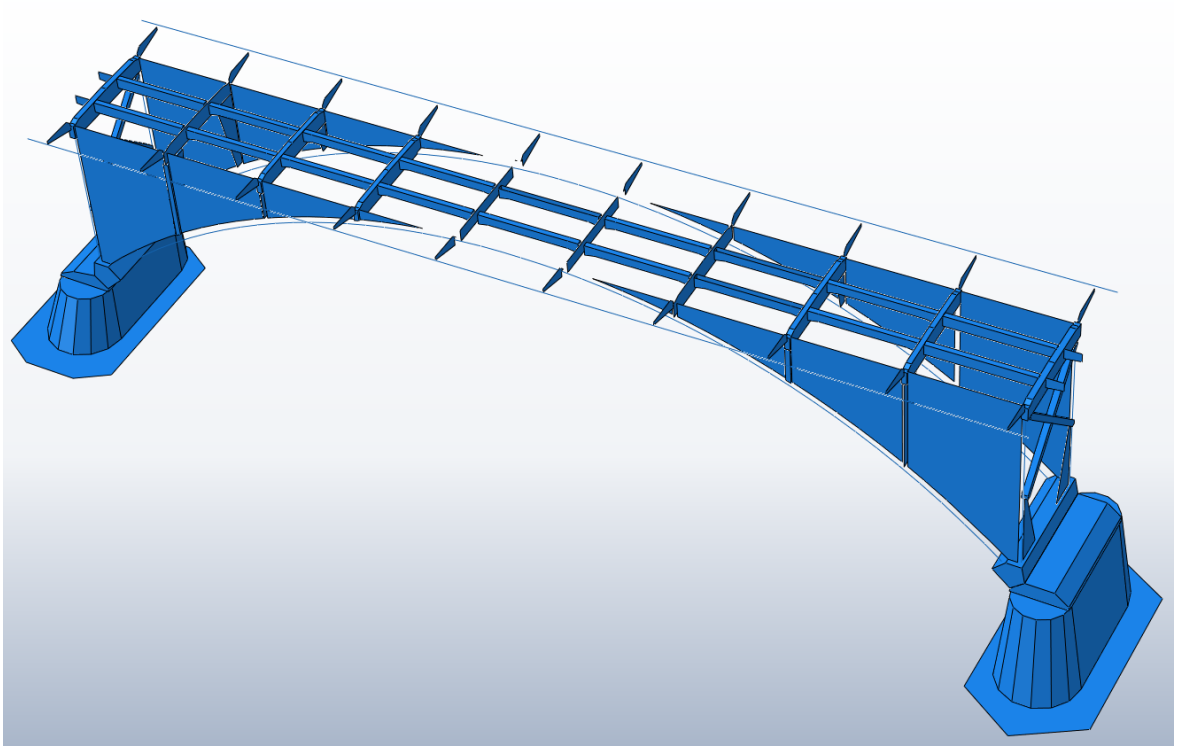
Import av hel modell

Nedan följer ett exempel på import av en hel brosektion. I Figur 52 visas bron importerad med de gamla metoderna för solidimport. Vägbanan har redigerats bort för att tydliggöra modellen. Det är lätt att i figuren identifiera många element som passar som skalelement respektive balkelement.



Figur 52 - Brosektion importerad som solidmodell

I Figur 53 visas bron importerad med den nya metoden, även här utan vägbana. Det går tydligt att se att kantbalkarna och bågarna har blivit balkelement, det finns även en vertikal balk i varje hörn av bron som blivit balkelement. Resten av balkarna, både tvärgående och längsgående, i modellen är av avancerad geometri och därmed *IfcFacetedBrep*. Detta gör att de inte kan importeras som balkelement. Något man kan se (dock lite svårt att urskilja i figuren) är att en del av tvärbalkarna importerats som skalelement och en del som solidelement. Anledningen till skillnaden är att en del av dem är *IfcBeam* (vilka inte kan bli balkelement och därmed blir solidelement) och en del av dem är *IfcPlate* (vilka lyckas med att bli skal).

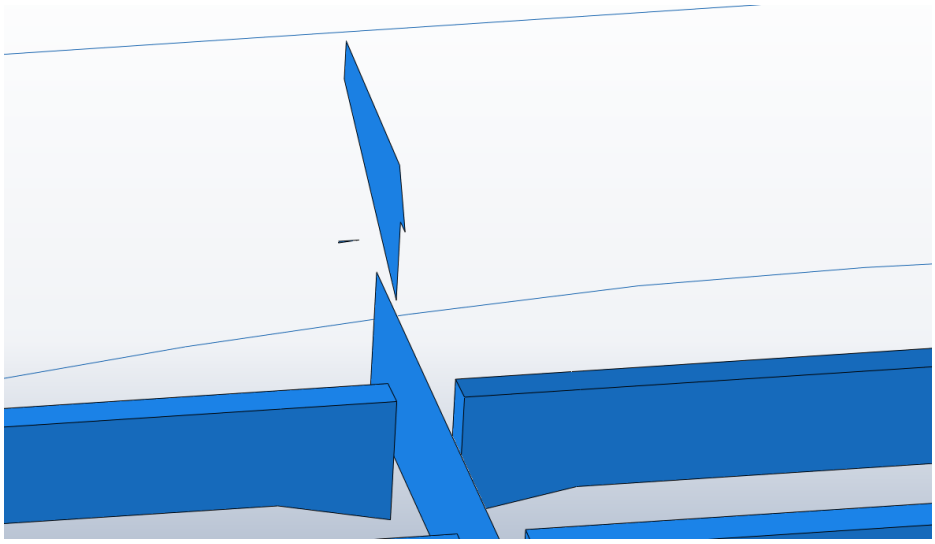


Figur 53 - Brosektion importerad som skal- och balkmodell

En hel del av plattorna importeras som skalelement på ett korrekt sätt och de mer avancerade elementen i botten importeras som solidelement. Till höger i Figur 53, på den bortre sidan, kan man se ett vertikalt hål mellan två plattor där ett element har försvunnit. Ett antal plattor har misslyckats med att bli skalelement, de ligger tvärs och ovanpå bågen vilket gör att deras två största sidor slutar på olika höjd.

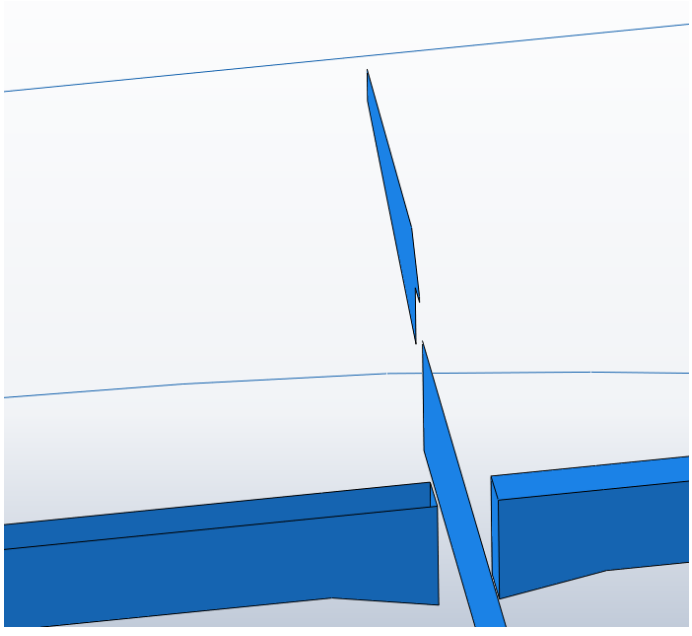
Modellen består av 103 element varav 81 har lyckats få *Section*. Att 22 element inte lyckats få *Section* beror inte på misslyckande med att importera själva elementen. Element som importerats som solidelement har fått en *Solid Section*, skalelement har fått en *Shell Section* och balkelement har fått en *Beam Section*, oavsett vad de från början var tänkt att importeras som. Det som gör att 22 element inte fått en *Section* är den ovanliga definitionen av materialet för dessa element i IFC-filen, *IfcMaterialLayerSetUsage*, som inte är implementerat i BRIGADE/Plus. Den nya metoden lyckas alltså för alla element.

Ett element, ändan på en av tvärbalkarna, som syns i mitten av den bortre sidan i Figur 53, har misslyckats vid import. Resultat visas tydligare och i en annan vinkel i Figur 54. Det har inte skapats ett ytelement i mitten av elementet utan istället har en liten bit blivit kvar och en av de två stora sidorna är kvar.



Figur 54 - Element som misslyckats med importen till skalelement

Hade ett korrekt skalelement skapats så hade det legat i linje med tvärbalken. Misslyckandet med att skapa detta skal beror på att elementet är *IfcFacetedBrep* med oexakta ytor. Importen lyckas vid solidimport eftersom Abaqus-kommandot *Stitch* anpassar ytorna för att passa ihop. Vid skalimport däremot så används de oexakta ytorna varpå något går fel med att skapa den nya ytan och processen misslyckas, kvar blir två andra ytor. I Figur 55 visas den närliggande tvärbalken och alltså hur det hade sett ut om importen lyckats.



Figur 55 - Element som lyckats med importen till skalelement

Diskussion och slutsatser

Detta arbete ligger till grund för att förenkla strukturmodelleringen av brokonstruktioner med armerad betong där lämplig beräkningsmodell innehåller skal- och balkelement istället för solida element. Det har visat sig att förenkling av både skal- och balkar fungerar bra i många fall. De fall då det uppstår problem är framförallt vid avancerade geometrier. Betongelement har oftast en relativt enkel geometri vilket gör att en större andel betongelement fungerar att transformera till skal- och balkelement vid importering av en IFC-modell i BRIGADE/Plus.

IFC och import/export i allmänhet

IFC-standarden är, trots många års användning, fortfarande förhållandevis ny. Standarden är brett definierad men dokumentationen är bristfällig, framtiden är oklar och anpassningen långsam. Bristerna med IFC kan vara en stor anledning till att användandet av BIM inte kommit längre. Utvecklingsföretagen vill inte lägga tid på något som inte fullt fungerar och användarna har fortfarande svårt att förstå hur allt hänger ihop. IFC2x3 *Coordination View* V2.0 är den del av IFC som används och IFC4 har nyligen kommit ut, men vad som hände med IFC2x3 *Structural Analysis View* är dåligt dokumenterat. Den senare hade troligtvis passat bra till ett samarbete mellan modelleringsverktyg och beräkningsverktyg. Kanske är det så att delar av detta finns inbakat i IFC4?

Det finns, trots mycket problem, många program som är anpassade och certifierade för IFC, import, export eller båda delar. Det verkar dock som att programmen importerar/exporterar på olika sätt, kanske framförallt för att dess export ska bli enkel att importera och se bra ut i de egna programmet. Ett exempel som vi stött på är en helt vanlig IPE300-balk i Autodesk Revit som exporteras med profilen *IfcArbitraryClosedProfileDef*. Varför det görs på detta vis när det finns en *IfcIShapeProfileDef* med alla nödvändiga parametrar är oklart. Det kan vara så att *IfcArbitraryClosedProfileDef* ser bättre ut när den importeras tillbaka in i Revit eller att det passar bra med någon funktion i tillverkaren av Revits, Autodesks, andra program. Det kan också vara så att Autodesk anpassat sin exportfunktion för att fungera bra med något annat program eller att profilen i fråga, IPE300 i det här fallet, inte är en standardprofil i Autodesks sortiment och därmed ritas som en godtycklig profil.

Det vi vill visa med ovanstående exempel är att import och export av IFC kan ske på många olika sätt, med många olika aktörer, många olika behov och många olika synvinklar. Detta gör att det inte går att förvänta sig något från en IFC-fil utan man får anpassa sig efter alla möjliga scenarier som kan finnas. Det gör även att importer till beräkningsprogram inte alltid kan bli på bästa sätt utan användaren kommer alltid att få anpassa sig till rådande förhållanden så länge IFC inte blir mer standardiserat och framförallt mer förståeligt.

Det finns ytterligare en faktor som påverkar hur IFC-filen är uppbyggd, nämligen CAD-operatören. Hur CAD-operatören valt att rita olika element påverkar hur de ser ut i IFC-filen. För importen i BRIGADE/Plus är det den geometriska representationen och elementdefinitionen som är intressant medan det ur CAD-operatörens syn kanske bara är intressant hur geometrin ser ut på pappret. Detta medför exempelvis att en CAD-operatör kan välja att rita en balk med rektangulärt tvärsnitt där bredden är mycket kortare än höjden. Geometriskt ser detta ut som en platta i BIM-programmet och fungerar väl för CAD-operatören, importen i BRIGADE/Plus blir dock helt fel med en balk.

Import i BRIGADE/Plus

Att vid import av elementen skapa en *Python-dictionary* med definitionen på varje element fungerar bra för den efterföljande importen av rätt sorts geometri. Den valda metoden att spara definitionerna (*IfcBeam*, *IfcSlab* osv) är dock inte nödvändigtvis den bästa. Ett bättre alternativ är troligtvis att lägga in ”*Beam*” istället för *IfcBeam* och *IfcColumn*, ”*Shell*” istället för *IfcPlate* och *IfcSlab* samt ”*Solid*” för alla andra. På detta sätt blir listan enklare att använda om man i framtiden vill ändra vilka element som ska importeras med vilken geometri.

Allting som har med importen att göra är kodat i Python, ett språk som ingen av oss hade tidigare erfarenhet av. Det medför troligtvis att flera av våra lösningar går att göra på ett bättre och/eller effektivare sätt. Dokumenteringen av Abaqus-funktioner är inte alltid helt klar, vilket även det gör att tillvägagångssättet inte alltid är det mest lämpade.

Import av skalelement

Import som skal görs om de två största ytorna är lika stora och om de är parallella. Här är det svårt att säga vad som är ”lika stora” och vad som är ”parallella”. Det gå inte att sätta dessa värden helt strikt eftersom det är omöjligt att säga att alla element har exakt geometri, därför finns det en tillåten felmarginal. Det är svårt att sätta ett exakt värde på denna felmarginal. För de två största ytorna så kan det vara bra om importen fungerar om sidorna är parallella men ytorna skiljer ett par procent i storlek. Vad gäller ytornas parallellitet så finns det säkert många fall där det går att godkänna en vinkelskillnad på 1 grad för exempelvis betongplattor. En vinkelskillnad på 1 grad i en stålplatta kan dock aldrig ge en rättvis import av skal. Dessa felmarginaler är något som bör testas och utvecklas för att hitta ultimata värden. Ett alternativ kan vara att sätta dem något högt för att få igenom så mycket som möjligt om man anser att användaren har godkänt importen av skal.

Import av balkelement

Import som balkar är relativt entydigt och förenat med få komplikationer. Det är framförallt import av specialprofiler som kan skapa problem om användaren inte inser att den själv måste ändra profilens egenskaper. Även om det finns ett varningsmeddelande för detta så är det inte säkert att användaren läser detta och meddelandet försvinner i den stora mängden information som skrivs ut, information som i dagsläget inte går att förhindra. Det finns även en möjlighet att inte ge dessa balkar någon profil alls. Detta tvingar användaren att skapa profilen själv och därmed omöjligt kan använda den felaktiga balken. Finns det ingen profil så kan det heller inte finnas en *Section*, vilket i så fall skulle innebära att det material som balken ska ha inte är kopplat till den. Användaren måste alltså utöver att ta fram egenskaper för profilen även ta reda på vilket material som balken ska ha. Det senare är troligtvis lika enkelt om inte enklare än att ta fram profilens egenskaper. Huruvida det är möjligt att importera dessa specialprofiler på ett sätt som inkluderar profilens egenskaper är värt att titta närmare på i framtiden.

IfcPropertySingleValue är ett IFC-objekt som innefattar en egenskaps namn, värde och enhet. Det är ett speciellt objekt i IFC som gör det möjligt för exporterande program att exportera egenskaper som inte ryms i övriga IFC. I dessa objekt kan det finnas värdefull information för importen men som kräver extra försiktighet och kontroll eftersom det inte finns regler för dessa på samma sätt som övriga IFC-objekt. Intressanta egenskaper som vi stött på i *IfcPropertySingleValue* är framförallt profilparametrar, vilka skulle kunna förbättra importen av specialprofiler.

Importen av standardprofiler innebär en del mindre komplikationer. Många av profilerna importeras med förenklingar enligt Tabell 5. Vad som är mer intressant är det faktum att L-, T- och asymmetriska I-profiler importeras med fel placering. Skillnaden mellan IFC-center och masscentrum är visserligen oftast liten, men skillnaden gör att laster som appliceras på balkelementet kan bli excentriska, likaså upplagskrafter. Detta är något som användaren måste ha koll på men som är svårt att förmedla. Det ser ut som att importen blivit bra då en korrekt profil finns. L-profiler skulle, precis som C- och U-profiler, kunna importeras som *Arbitrary Profile* och därmed anpassas profilen för att placeringen ska stämma. Det är svårt att avgöra om den något bättre placerade *Arbitrary Profile* eller den mer tydliga *L Profile* är det bästa valet.

Ett sätt att få korrekt placering av L-, T- och asymmetriska I-profiler är att helt enkelt flytta hela balkelementet. Det skulle troligtvis vara möjligt att ta fram värden och ge balkelementet en annan placering men det är oklart hur bra det hade fungerat. En sak som gör omplacering av balkelement krångligt är det faktum att värden på var masscentrum ligger oftast inte finns med i IFC-filen. Avståndet mellan masscentrum och mitten på ”omslutningslådan” är ett av attributen i IFC-profilerna, men det är dock sällan som det exporterande BIM-programmet har skickat med denna information. Återigen är det intressant att undersöka om *IfcPropertySingleValue* kan användas för att förbättra importen, i vissa exporter finns nämligen exakt placering av masscentrum (för hela elementet) bland dessa värden.

Import av avancerad geometri

Funktionen för import av bågar som balkelement är inte färdigutvecklad och det är inte överskådligt hur väl importen fungerar på olika varianter av bågar. Det är också svårt att, utan stort underlag, avgöra om det finns många geometrier som skulle kunna bli felaktigt importerade för att de har många ytor och är *IfcBeam*. Ett argument för att använda denna import är om användaren i framtiden får chansen att välja vad elementen ska importeras som. Bara för att CAD-operatören valt att rita ett element som balk så är det inte säkert att det passar som balkelement i BRIGADE/Plus.

Metoden för import av bågar kan utvecklas vidare och bli bättre. Med en vidareutveckling skapas troligtvis en möjlighet att använda metoden, eller en liknande metod, för att importera alla *IfcBeam* som är *IfcFacetedBrep* som balkar. Idag ser vi inget behov av ytterligare möjligheter att inkludera avancerade geometrier. Finns detta behov i framtiden så bör samma tillvägagångssätt, att elementet måste klara vissa krav, fungera bra för att sortera ut vilka element som ska använda en viss metod för importen.

Tidsåtgång

Tidsvinsten med den nya importen är något som bara den enskilde användaren kan avgöra, det beror framförallt hur korrekt importen blir i förhållande till användarens behov. En import till endast solid-element när användaren vill göra en skal- och balkmodell medför en stor mängd extra arbete med att göra om alla element till skal och balkar. Om importen istället sker med skal- och balkelement så sparar detta troligtvis mycket redigeringstid för användaren, även om inte alla elementen lyckas. Import med skal- och balkelement kan medföra visst extra arbete med interaktion mellan element. Användaren måste dock sköta interaktionen även med solidelement varför den nya importen inte bör innebära någon betydande tidsförlust i detta avseende.

Helhetsresultat

Metoderna som används fungerar bra och de flesta element går att importera som de ”ska” importeras. Det stora problemet är just hur det ”ska” importeras och hur CAD-operatören har valt att rita elementen. Så länge användaren inte ges en möjlighet att välja vilka element som ska importeras på vilket sätt så blir importerna inte optimala. En möjlighet att försöka förbättra helheten av importen är att skapa vissa parametrar och kontroller för varje element som gör att de kan importeras på ett annorlunda sätt. Ett exempel är att en *IfcBeam* med rektangulär profil där en sida är mycket kortare än den andra importeras som skalelement istället för balkelement. En metod av det här slaget skulle troligtvis kunna förbättra importerna en del men samtidigt gör det att användaren får sämre koll på val som sker under importen. Även om vi inte är så insatta i det datatekniska bakom BRIGADE/Plus, så tror vi ändå att det bästa och minst tidsödande är att utveckla ett grafiskt val där användaren får välja vilka element som ska importeras på vilket sätt.

Förslag på utveckling av BRIGADE/Plus

Vi har endast begränsad erfarenhet av att använda BRIGADE/Plus som beräkningsverktyg och kan därför inte uttala oss nämnvärt om användarnas behov i framtiden. Det vi kan ge är ett antal förslag på hur BRIGADE/Plus kan utvecklas baserat på den vidareutveckling som vi har stått för inom ramen för detta arbete.

Ett förslag rör valet av metod som ska användas vid importen av respektive objekt. Det kanske enklaste tillvägagångssättet skulle vara att läsa in IFC-filerna utan att exekvera något i BRIGADE/Plus och ge användaren en grafisk lista på alla byggnadselement med föreslagen geometri (solid, skal eller balk) förifylld. I detta grafiska gränssnitt kan användaren ges möjlighet att ändra den föreslagna geometrin. Ett enkelt sätt att importera de nya valen är att skapa en kopia av IFC-filen där valen har ändrats och sedan importera denna fil precis som vanligt.

Ett annat förslag är att den *Python-dictionary* som skapas med respektive elementdefinition istället för att innehålla *IfcBeam*, *IfcPlate* osv, skulle kunna innehålla ”*Beam*”, ”*Shell*” eller ”*Solid*” och att det är denna lista som användaren får ändra i det grafiska gränssnittet. Det senare skulle troligtvis bli mer avancerat att göra, fördelen är att man hela tiden använder samma IFC-fil och utan att ändra i denna.

I många IFC-filer finns det mycket information i *IfcPropertySingleValue* som kan vara användbart. Importen av dessa och vad de kan användas till bör ses över. I *IfcPropertySingleValue* kan det finnas information om placering av masscentrum, profilegenskaper som area och tröghetsmoment samt vikt och volym för elementet vilket skulle kunna användas för att importera materialdensitet. Eventuellt måste implementering av *IFCPropertySingleValue* anpassas efter olika CAD-verktyg.

Det finns fortfarande en del IFC-objekt som inte är implementerade i BRIGADE/Plus, men som varit med i de modeller som använts för testning och därmed kan vara tillräckligt vanliga för att det ska vara intressant att implementera dessa. *IfcMaterialLayerSetUsage* har nämnts tidigare, vilken har med materialimporten att göra. En implementering av denna skulle göra att fler element kan få *Section*. *IfcBooleanClippingResult* är ett annat exempel på IFC-objekt som återfunnits i testmodeller, men som inte finns implementerade. En inventering över vilka IFC-objekt som inte är implementerade och om de bör implementeras hade underlättat för framtida arbete.

Förslag på framtida studier

IFC är, trots många år av användning och ISO-standardisering, fortfarande under utveckling och har liten mängd informationsmaterial. Det kan därför vara svårt att ta sig an uppgifter relaterande IFC. Vågar man detta så finns det en del frågeställningar som vi har.

- Hur ser IFC ut om 5 år? 10 år? Kan man använda nya implementeringsmetoder? Kan exporter och importer bli bättre och mer standardiserade? Namnet IFC *Bridge* är något som dykt upp men som vi aldrig sett någon information om, kan det användas?
- Hur ser importen av IFC ut i andra beräkningsprogram? Går den att effektivisera?

Exporten från BIM-program till IFC följer visserligen standard men hur det ser ut i IFC-filen beror mycket på vilket det exporterande programmet är och vilken CAD-operatör som har ritat.

- Kan man med något program importera IFC-filer från olika program och exportera ut mer standardiserade IFC-filer?
- Vad kan man göra för att förenkla för CAD-operatörer och för att få olika CAD-operatörer att arbeta på ett mer standardiserat sätt? Vilka krav kan man ställa på CAD-operatörer?
- För att uppnå bra integration mellan projektering och strukturanalys måste båda parter ha tillräcklig kunskap om exportering och importering av IFC. Det krävs också god kännedom om den andra partens behov av BIM-modellen. Hur uppnås detta?

Mängden projekt som idag utnyttjar BIM är stor när det gäller husbyggnad. På anläggningssidan och framförallt brobyggnad så är det inte alls fallet.

- Varför används inte BIM mer inom anläggning? Vad kan man göra för att implementera BIM mer? Vilka hinder är det som stoppar utvecklingen?

Referenser

Austrell P-E. Dahlblom, O. Lindemann, J. Olsson, A. Olsson, K-G. Persson, K. Petersson, H. Ristinmaa, M. Sandberg, G. Wernberg, P-A (2004), *CALFEM A finite element toolbox Version 3.4*, Lund, KFS AB. Structural Mechanics, LTH Sweden.

buildingSMART (2007), *IFC 2x Edition 3 Technical Corrigendum 1* (IFC2x3 TC1) (www). Hämtat från http://buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm, maj 2014.

buildingSMART (2010), *Coordination View Version 2.0 EXPRESS-G Wall Paper Version 1* (pdf-dokument). Hämtat från http://www.buildingsmart-tech.org/downloads/view-definitions/coordination-view/sub-schema/CoordinationView_V2-0_WallPaper_IFC2X3_Version-1.pdf/view. Publicerat den 30 januari 2010. Hämtat februari 2014.

buildingSMART (2014a), *Industry Foundation Classes (IFC) data model* (www). Hämtat från http://www.buildingsmart.org/standards/ifc, februari 2014.

buildingSMART (2014b), *Open BIM* (www). Hämtat från http://www.buildingsmart.org/openbim, februari 2014.

buildingSMART (2014c), *Participants of the official buildingSMART IFC2x3 Coordination View V2.0 certification process* (www). Hämtat från http://www.buildingsmart-tech.org/certification/ifc-certification-2.0/ifc2x3-cv-v2.0-certification/participants, maj 2014.

Davidson, M (2003), *Strukturanalys av brokonstruktioner med finita elementmetoden – Fördelning av krafter och moment* (pdf-dokument). Hämtat från Brosamverkan Väst, www.brovast.nu/getfile.ashx?cid=32446&cc=3&refid=3, maj 2014.

Ekholm, A. Tarandi, V. Thåström, O (2000), *Tillämpning av IFC i Sverige – etapp 2 Slutrapport* (pdf-dokument), Stockholm, AB svensk Byggtjänst. Hämtat från http://www.caad.lth.se/fileadmin/projekteringsmetodik/research/Construction_informatics/slutrap98309.pdf, mars 2014.

Gelder, J. Tebbit, J. Wiggett, D. Mordue, S (2013), *BIM for the terrified a guide for manufacturers* (pdf-dokument). NBS, Construction Products Association. Hämtat från http://www.thenbs.com/pdfs/BIM-for-the-terrified.pdf, februari 2014.

Gustafsson, P-J (2012), *Balkteori* (Föreläsningslitteratur), Lund, KFS AB. LTH, kurskod: VSMN35.

Isaksson, T. Mårtensson, A. Thelandersson, S (2010), *Byggkonstruktion*, Lund, Studentlitteratur AB.

JBIM (2007), *Journal of Building Information Modeling* (pdf-tidning), NBIMS/NIBS. Hämtat från http://www.wbdg.org/pdfs/jbim_fall07.pdf. Publicerat hösten 2007. Hämtat februari 2014. Hämtat från artikel An Introduction to Building Information Modeling (BIM).

Khemlani, L (2004), *The IFC Building Model: A Look Under the Hood* (www). Hämtat från AECbytes, http://www.aecbytes.com/feature/2004/IFCmodel.html. Publicerat den 30 mars 2004. Hämtat mars 2014.

Loffredo, D (2014), *Fundamentals of STEP Implementation* (pdf-dokument). Hämtat från http://www.steptools.com/library/fundimpl.pdf, maj 2014.

Ottosen Saabye, N. Petersson, H (1992), *Introduction to the Finite Element Method*, Lund, Prentice Hall

Simulia (2014a), *Abaqus 6.12 Theory Manual* (www). Hämtat från http://abaqus.me.chalmers.se/v6.12/books/stm/default.htm. Hämtat från avsnitt: 3.5.1 Beam element overview, maj 2014.

Simulia (2014b), *Abaqus 6.12 Theory Manual* (www). Hämtat från http://abaqus.me.chalmers.se/v6.12/books/stm/default.htm. Hämtat från avsnitt: 3.6.1 Shell element overview, maj 2014.

Bilaga 1

Tabell över deltagande program i buildingSMARTs certifieringsprocess (buildingSMART 2014c).

Software Developer	Software Application	Exchange Req.	Export/Import	Status
4Projects Ltd.	4Projects	- (*)	Import	in progress
Archimen	Active3D	- (*)	Import	in progress
Autodesk	AutoCAD Architecture	Architecture	Import & Export	in progress
Autodesk	AutoCAD MEP	BuildingServices	Export	in progress
Autodesk	Revit Architecture	Architecture	Import & Export	**Export: certified** Import: in progress
Autodesk	Revit MEP	BuildingService	Import & Export	**Export: certified** Import: in progress
Autodesk	Revit Structure	Structural	Import & Export	**Export: certified** Import: in progress
Autodesk	Revit LT	Architecture	Import & Export	in progress
Bentley Systems	AECOsim Building Designer	Architecture, BuildingService, Structural	Import & Export	in progress
Cad-Quality	CADiE Sähäkkä	- (*)	Import	in progress
DICAD Systeme GmbH	STRAKON	- (*)	Import	in progress
Data Design System	DDS-CAD MEP	BuildingService	Export	in progress
Design Data	SDS/2	Structural	Import & Export	in progress
Dlubal Software GmbH	RFEM/RSTAB	- (*)	Import	in progress
ETU Software GmbH	HottCAD 4	- (*)	Import	in progress
FirstInVision	CasCADos / P3cad	Architecture	Import & Export	in progress
Gehry Technologies	Digital Project	Architecture	Import & Export	in progress
Graphisoft	ArchiCAD	Architecture	Import & Export	**Export: certified** **Import: certified**
International Training Institute ITI	Benchmark	BuildingService	Export	in progress
Kymdata Oy	CADS Planner	BuildingService	Export	in progress
NEMETSCHEK Allplan	Allplan	Architecture	Import & Export	**Export: certified** Import: in progress
NEMETSCHEK BIM+	BIM+	- (*)	Import	in progress
NEMETSCHEK Vectorworks, Inc.	Vectorworks	Architecture	Import & Export	**Export: certified** **Import: certified**
NEMETSCHEK Scia	Scia Engineer	Structural	Import & Export	**Export: certified** **Import: certified**
Plancal S&E GmbH	nova	BuildingService	Import & Export	in progress
Progman	MagiCad	BuildingService	Export	in progress
RIB	Arriba CA3D	Architecture	Export	in progress
RIB	iTWO	- (*)	Import	**Import: certified**
Seokyoung Systems	NaviTouch	- (*)	Import	**Import: certified**
Solibri	Solibri Model Checker	- (*)	Import	**Import: certified**
Solideo Systems	ArchiBIM Server	- (*)	Import	**Import: certified**
Tekla	Tekla Structures	Structural	Import & Export	**Export: certified** **Import: certified**
VIZELIA	Facility on line	- (*)	Import	in progress