

**SZÉCHENYI ISTVÁN
EGYETEM**



MŰSZAKI TUDOMÁNYI KAR

**MATEMATIKA ÉS
SZÁMÍTÁSTUDOMÁNY
TANSZÉK**

TMDK Dolgozat



AZ ARCHICAD ÉS AZ IFC SZABVÁNY

*Konzulens oktató:
Dr. Kallós Gábor
egyetemi docens*

*Készítette:
Baldauf Richárd (CLY9XW)
Mérnök informatikus hallgató*

2012/2013. I. félév

TARTALOMJEGYZÉK

1. BEVEZETÉS.....	3
1.1. A dolgozat bemutatása	3
1.2. A kitűzött célok bemutatása	3
2. AZ IFC SZABVÁNY BEMUTATÁSA	5
2.1. Az IFC szabvány „felhasználói”: programok és szakemberek	5
2.2. Az IFC kapcsolódó részeinek bemutatása.....	10
2.3. GSA szervezet bemutatása és követelményei	11
3. AZ IFC2X3 COORDINATION VIEW 2.0.....	14
3.1. Milyen problémákra kínál megoldást a CV 2.0 almodell?	15
3.2. A modell miatt ArchiCAD-ben módosított elemek ismertetése	16
4. AZ ARCHICAD ÉS A CV 2.0.....	24
4.1. A csapat segítségével általam megvalósított új funkciók bevezetése	24
4.2. A CV 2.0 miatt megjelent új módosítások az ArchiCAD-ben.....	25
4.2.1. Az egyszerű fal speciális reprezentációs eseteinek újításai	25
4.2.2. Ferde oszlopok megfelelő vágása.....	28
4.2.3. Íves és egyenes gerendák tükrözésének problémája.....	29
4.2.4. Tető exportálása és importálása	31
4.2.5. Opening vágások megfelelő exportálása	38
4.2.6. A födém vágásainak geometriánkénti mentése	39
5. AZ ARCHICAD TESZTELÉSE	41
5.1. Graphisoft Autotester-ről röviden.....	41
5.2. Whitebox tesztelés	41
5.3. Blackbox tesztelés.....	41
5.4. Tesztek kiértékelésének különféle típusai.....	42
6. FEJLESZTŐI ESZKÖZÖK	43
6.1. Felhasznált szoftverek.....	43
6.2. A belső intranet részei.....	47
7. A FEJLESZTÉS MENETE	50
7.1. Szerepkörök a fejlesztésben	50
7.2. Projekt életciklusának fázisai.....	51
8. ÖSSZEFOGLALÁS.....	53
9. IRODALOMJEGYZÉK.....	54

1. BEVEZETÉS

1.1. A dolgozat bemutatása

A dolgozat fontos központi témáját képezi egy nyitott szabvány, ami multi-diszciplináris területeket foglal magába. Épület gépészeti, építészeti, szerkezet építési szakterületekről van szó, ahol az Industry Foundation Classes (IFC) szabvány kiemelkedő szerepet tölt be, mivel egységesíti a tudományágak közötti kommunikációt az egyes tervező programok körében (például: ArchiCAD, Revit Autodesk, Tekla). Ez a szabvány lehetővé teszi a tudományágakban fellelhető elemek ábrázolását, vagyis környezeti, építészeti objektumok definiálását adja meg, majd a tervező programok az általuk megvalósított módon megjelenítik ezeket az objektumokat 2D és 3D formában.

A TDK munka felépítése a következő: a bevezető részek után a 2. és 3. fejezet bemutatja a szabvány részleteit és a Coordination View 2.0-ás almodellt (és az almodellnek megfelelő formai és tartalmi követelményeket), a 4. fejezetben a kapcsolódó fejlesztéseket írom le, a további részek pedig a tesztelő keretrendszerrel szólnak, illetve a fejlesztő eszközöket és a fejlesztés menetének a bemutatását tartalmazzák.

Bár mondhatjuk, hogy általában nem szerencsés TDK témaként céges team munkában végzett szoftverfejlesztést választani, de ez alól feltétlenül illik kivételt tenni olyan esetekben, amikor valami új ötlet, innováció megvalósítását, illetve valamilyen érdemleges probléma megoldását végezzük el. A dolgozat témaválasztásánál ez a cél motivált engem is. A fejlesztés jelentős innovatív értéket képvisel, a belső hasznosítás mellett cégünk belföldi és nemzetközi üzleti kapcsolatai révén sok partnerünk is profitál az újításokból és fejlesztésekből.

A vállalkozás közös jellege miatt külön is hangsúlyozom, hogy jelen dolgozat egyedül az én munkám, ebben más kolléga nem vett részt (bár a leírás – a kapcsolódási pontok miatt – szükségszerűen röviden tartalmazza a csapattársak javításainak egy részét is). A létrehozott új funkcionalitások az ArchiCAD 17-ben kerülnek majd kiadásra.

1.2. A kitűzött célok bemutatása

Már korábban több projekt is közreműködött ennek a nézetnek a támogatására, hiszen a Coordination View 1.0-ás nézetet már jelenleg is megvalósítja az ArchiCAD. Általánosságban kijelenthetjük, hogy a dolgozat célja ennek a szabványnak, illetve nézetnek a bemutatása és az ArchiCAD-ben létrehozott új funkcionalitások részletezése mérnöki, szoftverfejlesztői szemszögből. Mindez magában foglalja a lineáris algebrai ismereteket, feltételezzük ezeknek az ismereteknek valamilyen szintű ismeretét, de ezt részletesen nem tárgyaljuk. A konkrét programozási mélységben megvalósított kódra nem térünk ki üzleti titok, szellemi tulajdon védelme miatt, hiszen a piacon kiélezett verseny folyik a konkurens cégek között. Ennek ellenére fontos megemlíteni, hogy cégünk C++ nyelven valósítja meg mindezt, így az irodalomjegyzékben taglalt könyvek ([6] *C++ programming language* [7] *The Effective C++*) ismerete elengedhetetlen a

megvalósított funkcionalitások effektív implementálására. A kitűzött célok általánosságban a geometriai megjelenítést fogják javítani, amik az alábbi elemek fejlesztését veszik igénybe:

- Egy épülethez köthető hosszúsági, szélességi, magassági adatok kezelése.
- Gerenda megfelelő reprezentálása jobbsodrású koordinátarendszerben, illetve annak ellenőrzése, hogy abban van-e elhelyezve.
- Födém vágásainak szabványszerű exportálása.
- A fal egyszerűbb eseteinek kezelése, amikor nem döntött, nem íves geometriával rendelkezik, illetve a falban elhelyezhető vágások korrekt kezelése, pl. teljesen kivágja-e a falat vagy csak részben.
- Íves fal exportálása és annak importálása, a szabvány által szolgáltatott speciális geometriai forma kihasználásával.
- Íves falban elhelyezett részleges és teljes vágások megfelelő exportálása.
- Tető exportálása/importálása szükséges geometriai konverzió létrehozása az import folyamat során beolvasott eltérő geometriai reprezentálás miatt.
- Oszlopok importálása és azokon elhelyezett felső és alsó vágások megfelelő kezelése.
- Íves gerendák beolvasásának kezelése egy adott geometriai típusból.

A geometriai reprezentációk lehetséges eseteit „Az IFC kapcsolódó részeinek bemutatása” című fejezetben ismertetjük részletesen. Ezek alapvetően a 3D térben történő értelmezést határozzák meg.

Az eddigi célkitűzések leginkább a projekthez fűződtek és világszerte az ArchiCAD felhasználóinak érdekeit képviselik, azonban személyes céljaim közé tartozik a szabvány átható ismertetése és annak elsajátítása, hiszen 2012 márciusa óta a Graphisoft SE Zrt. munkavállalójaként – szoftvermérnöki pozícióban – folyamatosan a szabvány ArchiCAD-ben történő fejlesztésén dolgozom. Mivel még áthatóan nem ismerem a szabvány összes osztályát, ezért a megvalósítandó innovatív újítások mellett ezeket is jobban el tudom sajátítani.

2. AZ IFC SZABVÁNY BEMUTATÁSA

2.1. Az IFC szabvány „felhasználói”: programok és szakemberek

Az IFC szabvány – egy megfelelő tervező program segítségével – igen nagy segítséget nyújt az épülő házak tervezésében. Mivel a tervezési és építési folyamat különböző területein dolgozó szakemberek szoros és zökkenőmentes együttműködését határozza meg, ezért nagyon hasznos egy olyan adatséma, amely az információküldést rendszerezi és szabványosítja. Megfelelő formai és tartalmi követelmények mellett – amiket a tervező program biztosít – szabadon kommunikálhatunk más programok között, amik szintén implementálják az IFC szabványt. A nyitott szabvány további nagy előnye, hogy nem csak a szakemberek közötti kooperációt valósítja meg, hanem különböző programokat is nagyon sikeresen összehangolja. A teljesség igénye nélkül a szakirodalomban található egy külső hiperlink, amely a [10] IFC 2x3 szabványt teljes egészében részletezi angol nyelven, ahogyan azt a buildingSMART létrehozta.

A tudományágak közötti egységes kommunikációjának megvalósításához a fejlesztő cég (buildingSMART) létrehozott egy nézetet, ami jelenleg az IFC szabvány egyik legjobban megtervezett nézete és a 2.0 verziónál tart. A Coordination View 2.0 megköötéseket tartalmaz az IFC szabvány ábrázolásmódjára, ugyanis megszabja, hogy az egyes objektumokat milyen módon lehet ábrázolni geometriai és logikai kapcsolatokon keresztül (pl.: egy lyuk a falban, a fal geometriájában legyen reprezentálva, vagy külön elemként kapcsolódjon az adott falhoz).

A buildingSMART független, nemzetközi szervezet játszott közre a szabvány és a nézet megvalósításában, majd az IAI (International Alliance for Interoperability) által kidolgozott eszközöket fejlesztették tovább közösen, az IAI már 20 éve, a 80-as évek közepe óta készített ilyen szabványokat, és később belőle született meg a buildingSMART. Az egyik nevesebb közülük a STEP, amely szintén egy szabvány (ST – Standard), és a termékek adatainak (P – Product data) cseréjét, átváltását (E – Exchange) segítette, ami már lassan 20 éve sikeresen zajlik. Összefoglalóan a buildingSMART a következő fontos szempontokban segíti az építőipart:

- Segítséget nyújt „zöld” házak építésben.
- Az ezzel kapcsolatos üzleti folyamatokat fejleszti.
- Új üzleti modelleket fejleszt.
- Növeli a kereskedelmi értékláncot a cégek számára.

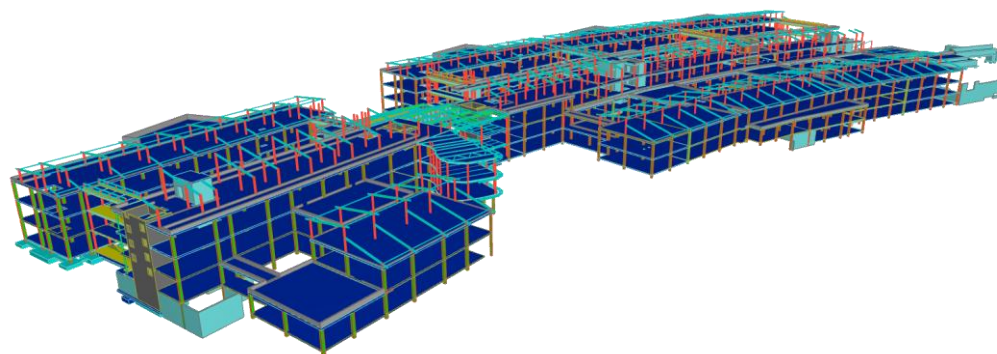
A szervezet fő profilja a nemzetközi szabványok fejlesztése és karbantartása, így ehhez szorosan kapcsolódó tevékenységek:

- A buildingSMART folyamatainak működtetése.
- buildingSMART adatszótár fejlesztése, amely a szakma idegen szavainak precíz jelentését részletesen taglalja.
- buildingSMART adatmodell karbantartása, amelynek részét képezi az IFC2x3 is.

Céljuk ezzel a fejlesztéssel is az, hogy megfelelő, minőségi épületeket építsenek, hiszen a korai tervezési fázisban egy ilyen tervező program kellően segíti a szakemberek problémamegoldó

készségét. Sok problémára felhívja a figyelmet és a költségeken át az energia szint minimalizálását is jól lehet vele előre becsülni. Választ kapunk például arra, hogy mennyi a legrövidebb idő, ami alatt fel lehet építeni az adott épületet, megfelel-e az épület az egyes kritériumoknak, mennyire környezetbarát. A buildingSMART az ipart abba az irányba vezeti, hogy alkalmazzák ezeket a szabványokat, hiszen sok előny származik belőle a cégeknek hosszú távon, továbbá biztosítja nekik a szakmai elismerést, az implementációt és a képzési programokat, külön kezelve az IAI technológia szabvány fejlesztését.

Az ArchiCAD épülettervező szoftver is sikeresen implementálja ezt a szabványt. A szoftver nemcsak az épületterv megjelenítését, hanem szerkesztését is lehetővé teszi, emellett környezet elemzési lehetőséggel, információ megosztással segíti a szakemberek együttműködését. Kliens-szerver vonatkozásban képes több ArchiCAD-del közösen együttműködni, ahol egy közös szerveren zajlik a munka.



1. ábra Az ArchiCAD-ben modellezett épület

Lássunk egy életszerű példát arra, hogy mire is jó ez a nyitott szabvány a tervezési fázisban. ArchiCAD-ben könnyedén össze lehet állítani egy egyszerű épületet, jelen esetben bőven elég, ha a belsejét nem dolgozzuk ki, csak a külsejét, mint az 1. ábrán is látható. A fájl/mentés másként menüpontban válasszuk ki, az *.ifc fájlformátumot és mentsük el a fájlt. Egyelőre az egyéb beállítási lehetőségeket most hagyjuk figyelmen kívül, később majd írunk róluk bővebben is. Ezt a mentett fájlt már más tervező programokkal is be tudjuk olvasni, amelyek szintén alkalmazzák az IFC import megjelenítést. Ilyen szoftver például a Solibri Model Viewer, ami ingyenesen elérhető¹ a gyártó oldaláról. Természetesen van még más szoftver is, amivel lehet IFC adatsémát megjeleníteni, ezekről említés szintjén később lesz majd szó. Amire felhívnam a figyelmet, az az export (ArchiCAD részéről) és az import (Solibri részéről) folyamat, hiszen ez a kulcsa annak, hogy jól kerüljenek megjelenítésre az adatok. Természetesen a legnagyobb előnye ennek a folyamatnak, hogy az épület egy reprezentációját mentjük fájlba, ami belül még nincs megtervezve, átadhatjuk egy belső építésznek és az befejezi a munkát.

Magának az IFC szabványnak többféle verziója is van. Jelenleg a 2.3-as verziót implementálja az ArchiCAD is, hiszen a negyedik még béta verzióban van és gyűléseket is tartanak a szoftver gyártókat bevonva, hogy milyen újításokat és módosításokat hajtsanak végre a szabványon. Ezen belül a Coordination View 2.0 egy al-modellje az IFC2x3-nak, ami még szigorúbb és javításokat is

¹ Az alábbi URL-en keresztül (<http://www.solibri.com/get-solibri-model-viewer.html>), némi adatmegadás után elérhető a Solibri Model Viewer, a TMDK mellékletében csatolva van egy épület IFC exportja a megtekintéshez.

Az IFC szabvány bemutatása

tartalmaz. Azon cégek amelyek szeretnék alkalmazni ezt a modellt, azoknak minőségi kritériumoknak kell megfelelnie. Erről később bővebben lesz majd szó, hogyan is zajlik a modell bővítése, ugyanis koránt sincsen még készen a teljes tesztlista.

Az ArchiCAD és az IFC szabvány verziókezelése			
<i>IFC verzió</i>	<i>Kiadás éve</i>	<i>ArchiCAD verzió</i>	<i>Kiadás éve</i>
IFC2x4 release	2012. szeptember 3.	---	---
ifcXML 2x3	2007. június	ArchiCAD 13	2009
IFC2x3	2006. február	ArchiCAD 12	2008
ifcXML2 for IFC2x2	2004	ArchiCAD 9	2004
IFC2x2	2004. július	ArchiCAD 9	2004

A már kiadott verziók nagy múltra tekintenek vissza, hiszen az első 1.5.1-es verziót 2000-ben bocsátották ki, aminek az ArchiCAD már sikeresen teljesítette minden kritériumát. A következő oldalon szereplő táblázat összefoglalja, hogy milyen verziók is jelentek meg és azok implementálásra kerültek-e az ArchiCAD adott verziójában.

Az IFC szabvány bemutatása

ifcXML1 for IFC2x	2004	ArchiCAD 7*	2004
IFC2x	2004. március	ArchiCAD 7*	2004
IFC 2.0	2001. május	ArchiCAD 7	2001
IFC 1.5.1	2000. május	ArchiCAD 6.5*	1999
IFC 1.5	1998	---	---

A fenti táblázat nem részletezi, melyik verzió mit valósított meg, de jól látható az egyes verziók kiadásai között eltelt idő és csak a fontosabb szabványokat említettük itt meg. A Wikipédián több verzió is található. Az egyes ArchiCAD verzióknál megjelent piros csillag jelöli azt, hogy későbbi Add-on formájában adták ki, nem pedig az adott verzió legelső kibocsátásakor. Így fordulhat elő az, hogy az ArchiCAD 6.5-ös verziót már korábban kiadták, mint az IFC 1.5.1-et és későbbi Add-on keretében került implementálásra. Az IFC Coordination View 2.0, amit a táblázat nem tartalmaz a következő verzióban kerül implementálásra, 2013-ra, de erről később is írunk sokkal részletesebben.

A szabványnak nem csak modellezéssel kapcsolatos céljai vannak, mivel energetikai számításokkal is foglalkoznak külső cégek (pl.: spórolni akarunk a fűtéssel). Az export folyamat során előállított szöveges fájl energetikai számításokhoz is fel szokták használni. Viszont az ArchiCAD rendelkezik ilyen beépített funkcióval, amit Eco Designer Add-onnak hívnak. Ezzel lehet készíteni egy PDF dokumentumot, amely különféle fogyasztási szempontok mérését és osztályozását tartalmazza. Ezeket a szempontokat csoportokra bonthatjuk az alábbiak szerint:

- Kulcs értékek: Hány szintes egy épület, mekkora tetővel rendelkezik, hőáramlási tényezők, energiafogyasztás, nettó hűtési és fűtési értékek, geometriai adatok, hány négyzetméteres az egész épület.
- Energia felhasználási források: Tipikusan idetartoznak a megújítható energiák (napenergia), de a megvásárolt szolgáltatás is energiaforrásnak minősül, mint a meleg víz és a világítás.
- Mekkora energia felhasználásra számítsunk: Idetartozik a meleg víz, hideg víz, fűtés, légkondicionálás, világítás (éves szintű adatokat kapunk).
- Havi energia megoszlás: A fenti csoportokon belüli értéket számítja ki havi szintre, mely hónapban mekkora hányaddal számoljunk velük. Két külön diagramon ábrázolhatók ezek az eloszlások. Az egyik a szolgáltatott, befelé áramló energia (víz, világítás, belső fűtés), a másik pedig a kibocsátott, kifelé áramló energia. Ide tartozik a felhasznált meleg víz, a környezetünkben felhasznált hűtés, a szellőzés.

Az imént felsoroltuk az ArchiCAD-ben használatos számításokat, ehhez hasonló jellemzők tartoznak az *.ifc fájl külső felhasználásaira is. Ezután már be tudjuk határolni, mely aktuális részterületeken vált fontossá ez a szabvány.

Tehát az IFC2x3 egy olyan szabvány, amely az építészethez kapcsolódik és definiál egy teljes objektum-orientált osztályokhoz hasonló hierarchiát. A legtöbb építészeti objektumnak megvan a megfelelő leképezése a szabvány különböző osztályaiban. Ezekre lehet alkalmazni különböző

szabályokat, megkötések. Az osztályok között – vagyis építészeti objektumok között – kapcsolatokat is definiálunk, amelyeket asszociációs típusokkal tudunk leírni (például, egy falhoz tartozik egy ablak). Ez a szabvány megkönnyíti az építőipar területén a különböző munkakörben dolgozó emberek együttműködését. Többféle program képes beolvasni *.ifc fájlokat, elkerülve az egy adott szoftverhez való ragaszkodást. Ilyen programok lehetnek²:

- Solibri Model Checker/Viewer
- DDS CAD Viewer
- Revit Structure
- FZK Viewer
- Tekla
- Graitec
- Scia
- MagiCAD
- EnergyPlus
- AutoCAD
- ArchiCAD

Ez úton szeretném felhívni minden kedves oktatók és hallgatók figyelmét arra, hogy kisebb adminisztráció után számukra ingyen elérhető egy évig az adott ArchiCAD licenz, amely most a 16-os verziót foglalja magában. Szóval bátran látogassanak el a cég hivatalos oldalára és szerezzenek belőle egyet. Ezt azért is célszerű megtenni, mert a TDK dolgozatban bemutatott dolgokat jobban el tudják képzelni, gyakorlatiasabbá tehetők saját maguk számára.

² A felsorolásnál a verziószámok részletezésétől eltekintünk, illetve több program is rendelkezik kiegészítő csomaggal, ezekre viszont nincs szükség, ha meg akarunk nyitni egy *.ifc fájlt.

2.2. Az IFC kapcsolódó részeinek bemutatása

Minden egyes háromdimenziós objektum, amit kezelni fogunk különféle módon írható le megjelenítés szempontjából. A közös nyilvánvalóan az, hogy pontokból és a poligonokból épül fel, ezeket használjuk fel ahhoz, hogy leírjuk az *.ifc fájlban szereplő egy adott objektumot. A pontokra és a poligonokra alkalmazott csoportosítás szempontját a módszer határozza meg. Ilyen kategorizálási csoportok lehetnek a következők:

- *Boundary Representation (BREP)*: A szabványban használatos neve *IfcFacetedBrep*, ahol egy test pontjait csoportokra bontjuk. Egy kocka esetén ez 6 poligont jelent, ahol a pontok indexei vannak feltüntetve egy poligonba, a redundancia elkerülése végett használjuk az indexeket.
- *Revolved Area Solid (Revolved Data)*: Ebben az esetben adott egy poligon, valahol a térben, illetve hozzá egy referencia vektor, amihez hozzá társítunk egy szöget és a megfelelő irányba elforgatva az adott síkot a térben kapunk egy 3D objektumot. Itt ugye bőven elég megadni annak az egy síknak a pontjait. A többit a szögnek megfelelően számítjuk ki és szegmensekre osztva azokat megkapjuk az ívet, feldarabolva, minél kisebb a szegmens mérete annál jobb közelítést kapunk. Az IFC szabványban az *IfcRevolvedAreaSolid* néven van nyilvántartva.
- *Surface Curve Swept Area Solid (Surface Curve Swept Data)*: Itt is egy síkban elhelyezkedő pontok az adottak, csak úgy nyújtjuk egy adott irányba (directx „útvonalon”), hogy akár egy csavar hengerformához hasonló elemet kapjunk, ritkán használt tárolási forma a szabvány *IfcSurfaceCurveSweptAreaSolid* néven említi.
- *Geometric Curve Set (Geometric Set Data)*: Itt 2D vagy 3D pontokról beszélhetünk, amelyek íveket írnak le. Ezzel ábrázoljuk azokat a formákat, amik formailag több ívet és pontot tartalmaznak. Ez nem feltétlenül határol be egy poligont, lehet akár egy egyszerű ív is, amihez vonalak társulnak a térben. *IfcGeometricCurveSet* a szabványban képviselt neve.
- *CSG Primitive 3D (CSG Primitive 3D Data)*: A CSG-nek (Constructive Solid Geometry) az értelmezése akkor nyer jelentőséget, ha adott kettő test (például két kocka), ekkor CSG segítségével meg lehet határozni azok unióját, metszeteit és különbségeit. Így egy szabványos fájlban értelmezett két test között CSG operátoroknak megfelelően jelenik meg az adott geometria, nevéből adódóan háromdimenziót használva. IFC2x3 TC1-es szabvány szerinti azonosítója: *IfcCsgPrimitive3D*.
- *Extruded Area Solid (Extrusion Data)*: Ez az ábrázolási forma lehetővé teszi, hogy egy adott síkban elhelyezkedő pontokat egy referencia iránnyal ellátva adott mélységgel, hosszúsággal kinyújtsuk. Így egy háromdimenziós testet kapunk. Szabvány szerinti neve *IfcExtrudedAreaSolid*.

Vannak még egyéb más csoportok is, amelyek lehetővé teszik egyszerűbb alakzatok leírását, de ezeket mi nem alkalmazzuk, viszont egy *.ifc fájl kulcsfontosságú összetevője az az elem, ami tartalmazza az összes olyan térbeli elem azonosítóját, ami megjelenítésre kerül az ArchiCAD-ben. Sok esetben használjuk ennek az elemnek a módosítását egy adott hiba javítása során. Így nem kell foglalkoznunk a többivel és az lényegesen leegyszerűsíti a debuggolási teendőinket. Ez az elem a *IfcRelContainedInSpatialStructure*, amely tartalmaz egy listát arról, hogy milyen térbeli elemeket tartalmaz a fájl. Ezek az elemek lehetnek az *IfcProduct* bármilyen leszármazottai, mint például *IfcSlab*, *IfcWallStandardCase*, *IfcWindow*. Ami pedig a listába bekerül, az egy sharp (#) jellel

ellátott szám, ami a konkrét elemet azonosítja a fájlban. Ezek az elemek pedig a már korábban felsorolt különféle reprezentációs formákként írhatók le geometriailag (pl.: Egy *IfcWallStandardCase IfcExtrudedAreaSolid* segítségével) egy szöveges fájlban.

2.3. GSA szervezet bemutatása és követelményei

A GSA (General Services Administration) egy független amerikai ügynökség (koordinátor szervezet), amely sok különböző (főként mérnöki) szakterületen egyfajta „nemzeti” vezetést biztosít (irányvonal, szabványok) az USA tagállamainak közreműködésével. Számunkra elsősorban az építészeti és építőmérnöki vonatkozások a fontosak. Az ezeken a területeken megvalósuló fejlesztések elsősorban a fenntartható dizájn, szépművészeti, történelmet reprezentáló építési szolgáltatásokat és a projektmenedzsmentet érintik. Azért, hogy biztosítsa a jó minőségű szabványok kiválóságát az amerikai nyilvánosság számára, a GSA a legjobb építészeket és mérnököket alkalmazza a privát szektorból kiválasztva, akik dizájn, renoválási/felújítási és építészeti feladatokat látnak el. A GSA szolgáltatásai közül cégünk, a Graphisoft számára a legfontosabb a Public Building Service (PBS), amely a szervezet saját tulajdonú ingatlanjainak vételét/eladását, menedzselését végzi, mint például:

- földterületek
- farmok
- családi rezidenciák
- postahivatalok
- repülőterek
- irodák
- raktárépületek

A GSA működése során az egyik fontos szempont, hogy olyan épületeket tervezzen, amelyek szem előtt tartják a környezetvédelmet (a közelmúltban 24 épületét is kitüntették a LEED díjjal – Leadership in Energy and Environmental Design). Az épületek többségét zöld tetővel, szintek alatti levegőelosztással és egyéb innovatív eszközökkel újították fel. A GSA emellett megkapta a „Honor Award díjat” a Nemzeti Épület Múzeumtól, szintén az innovációnak köszönhetően. Ezek a fejlesztések leginkább az amerikai társadalom érdekeit szolgálták, hiszen volt példa arra is, hogy teljesen ingyen és bérmentve felajánlott földterületeket a közérdek számára.

Az irodalomjegyzékben található könyvekben, hiperhivatkozásokban több információt is találunk arról, hogy milyen egyéb területeken rendelkezik befolyással a GSA, de ezekre most azért nem térünk ki, mert a dolgozat szempontjából érdektelenek, viszont a teljesség kedvéért megemlíjtük. A [2] Wolters Kruwer művében tájékozódhatunk a GSA által alkalmazott szabályokról, eljárásokról, milyen tennivalókkal jár, ha egy amerikai állampolgár házat akar építeni. A [9] GSA Wikipédia cikk felhasználásával készült ezen fejezet bevezetője, de több érdekes információ is olvasható angol nyelven a cikkben.

A GSA a világpiacon is fontos, nagy potenciállal bír. Mivel rengeteg épületet tartanak karban és folyamatosan fejlesztik azokat, innovációkkal szélesebbé teszik az építészeti modellek repertoárját, ezért célszerű a követelményeiket teljesíteni. A GSA 2007-től kezdve minden egyes építészeti szoftvergyártó cégtől vár tesztfájlokat, amivel össze tudják hasonlítani, melyik elégíti ki legjobban

az igényeiket. Így az ArchiCAD saját formátuma a *.pln, *.pla fájlok – továbbá kikötés még, hogy *.ifc fájlt is csatolni kell, amit az adott szoftver mentett ki.

Ebből is látszik, hogy mekkora jelentősége van ennek a fájlformátumnak. Másrészt pedig több európai szervezet is átvette a GSA követelményeit. Ilyen szervezet a Statsbygg (Norvégiából) és a Senate Properties (Finnországból). A GSA követelményei többféle típusra bonthatók, amelyek az alábbiak:

1. IFC hierarchia teljes megvalósítása – automatikusan teljesül
2. Projekt földrajzi helyzete: koordináták, hosszúsági szélességi fok adatok – felhasználói adatbevitellel
3. Mértékegység: Metric/Imperial (amerikai mértékegységekről európaira) – *translator* segítségével
4. BIM – Objektumok tulajdonosainak az adott objektumhoz történő rendelése – felhasználói adatbevitellel
5. Geometriai követelmények – *translator* segítségével
6. BoundingBox mentése – *translator* segítségével
7. Mennyiségek megfelelő kezelése (térfogat, hosszúság, terület) – *translator* segítségével
8. *IfcEntity* jellemzők: attribútumok, tulajdonságok, osztályozási referenciák – felhasználói adatbevitellel
9. IFC Assignments: az egyes projektelemek közötti kapcsolatot jelzi, csoportokba lehet vele az elemeket rendezni, tulajdonost lehet az *IfcSpace*-khez rendelni – felhasználói adatbevitellel

Automatikus teljesülés alatt azt értjük, hogy a teljes IFC hierarchia, amit használunk, az implementációnak köszönhetően nem igényel bármilyen beállítást a mentés során. (Mivel az egyes elemek közötti kapcsolatok automatikusan adódnak az import/export folyamat során.) **Felhasználó adatbevitel** során pedig egy dialógusablakon keresztül lehet megadni a megfelelő információt. A **translator segítségével** történő beállítás során is felhasználói beállításra van szükség, de ez az import és az export folyamatra vonatkozik. Ilyenkor nem direkt mi állítjuk be a konkrét értéket, – mint a felhasználói adatbevitel során – hanem beállítjuk az import/export folyamat során milyen formában kívánjuk például menteni egy *IfcSlab* geometriáját: ExtrusionData-ként vagy BREP-ként.

Már korábban említettük, hogy szükséges csatolni *.ifc fájlokat is ahhoz, hogy teljesítsük követeléseiket – ez több különféle tesztetből áll, amit a GSA írt elő és ezek típusait ebben a fejezetben kifejtettünk már. A tesztek elvégzése után kimentjük a fájlt IFC szerinti szükséges mentéssel a megfelelő translator beállításokkal és ezeket feltöltjük a GSA által üzemeltetett oldalra. Sajnos egy ilyen teszt elvégzése nem ingyenes, azonban az ArchiCAD 16 már megfelel a tesztek java részén, de akkor még nem teljesítettünk minden tesztet, illetve az adatok szemantikai helyességét nem vizsgálták ez esetben. Az eredményt, hogy melyik teszt volt sikeres és sikertelen, azt szöveges fájlba küldték vissza, a pontos adatokkal megjelölve. A következő tesztsorozat már vizsgálta az adatok helyességét, mivel célirányos modellezési teszteket futtattak le. Ezek a tesztek három típusra bonthatók:

- Térbeli program validáció: az elemeket a térben elhelyezkedő helyzetükhöz képest ellenőrzik
- Áramlás/biztonság analízis (nem a GSA-hoz kötődik, hanem a Georgia tech. vállalat végzi)

- Energia analízis (Lawrence Berkeley National Laboratory végzi a tesztelését)

A különféle szabályok megtartása miatt a Solibri Model Checker programmal szokták végezni a tesztelést. Ugyanis létre lehet benne hozni kritériumokat, például, hogy csak Extruded Area Solid típusú geometriákat tartalmazhat a fájl. Jelenleg ez a fejlesztés még nincsen kész az ArchiCAD-ben, majd egy másik projekt keretein belül kerül megvalósításra.

A térbeli program validáció alatt azt értjük, hogy ellenőrzésre kerül a kimentett objektumok egymáshoz viszonyított elhelyezkedése és azok geometria típusának vizsgálata (ahogy az előző bekezdésben szerepelt pl.: csak Extruded Area Solid lehet). A vizsgálat során ellenőrizhetik az objektumok összeszerkesztését is gerendák, födémek esetén, hiszen két födém vagy gerenda közös felületeit célszerű úgy ábrázolni, hogy ne szerepeljen kétszer a közös rész, mert egy építkezésen elég körülményes kivitelezni.

Aramlás és biztonság analízis esetén egy tűzriadó terv az ideális példa, de akár a hétköznapi átlagos forgalom vizsgálatát is ellenőrizhetik. Itt a „Mekkora folyosó szükséges az adott épületbe?” vagy a „Milyen alternatív útvonalak állnak rendelkezésre tűzriadó esetén?” kérdésre kaphatjuk meg a választ. Nyilván itt most nem az adott épületterve vonatkoznak a követelmények, hanem a lehetőségre, hogy ezt az analízist végre lehessen hajtani, különféle következtetéseket lehet belőle levonni az adott épületre. Az export során speciális paraméterekkel kell ellátni az *.ifc fájlban leírt objektumokat.

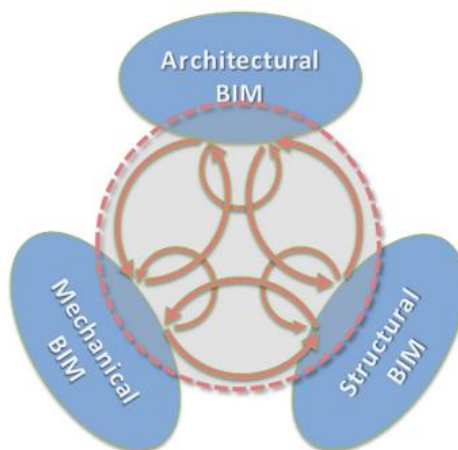
A harmadik vizsgálat tárgya pedig az energetikai vizsgálat, ahol az EnergyPlus segítségével mérik fel, hogy milyen káros anyag kibocsátást fog maga után vonni az épület, ha elkészül. Ehhez kapcsolódik az EcoDesigner Add-on, ami már bele van építve az ArchiCAD 16-ba és készíthetünk megfelelő riportokat az épülettervünkről, hogy az adott koncepció mellett mire számíthatunk. Egyelőre még maga az eljárás is változhat, ami a tesztet végzi, nincsen tökéletesen kiforrott módszer rá. Viszont az *IfcSpace* elemeket fogjuk hozzá felhasználni, amely az ArchiCAD-beli zóna megfelelője, az ezek között létrejött kapcsolatok segítségével lehet végezni a vizsgálatot.

3. AZ IFC2X3 COORDINATION VIEW 2.0

Az Industry Foundation Classes 2x3-as verziója a buildingSMART-nál lett kifejlesztve. Ez az általános adatséma, ami lehetővé teszi, hogy megtartsa és kicserélje az adatokat különböző cégek szoftver alkalmazásai között. Az adatséma információt tartalmaz, amely lefed különböző tudományágakat, amik mindvégig támogatják egy épület életciklusához fűződő feladatokat: a megépítésétől, a dizájn keresztül, a szerkesztés és a felújításon át akár a lebontási stádiumban is segítséget nyújtanak.

Ehhez kapcsolódik a Coordination View 2.0, ami egyéb megkötéseket tartalmaz a fájlban elhelyezhető tartalomhoz. Mivel egy ilyen fájlban többféle módon is meg lehet jeleníteni egy adott objektumot, ezért, hogy ne legyenek kompatibilitási kérdések, vannak különböző feltételek, amikhez tartani kell magunkat, ha egy szabványos *.ifc fájlt akarunk létrehozni. Ilyen lehet például egy fal, aminek a profilját³ egy alapsíkra elhelyezzük és azt egy adott irányba kinyújtjuk és a nyújtás végeztével kapjuk meg a falat (legegyszerűbb esetben egy téglatestet), amire értelmezhető kritériumok (pl.: merre lehet nyújtani a fal profilját). Ehhez hasonló problémák kiküszöbölésére nyújt egy egységes megoldást a CV 2.0. Várhatóan az ArchiCAD 17 már megfelelően fogja implementálni és teljesíteni fogunk minden szükséges kritériumot a certifikációhoz.

Általánosságban a CV 2.0-ról kijelenthetjük, hogy az épületgépészettel, építészettel és szerkezet építéssel kapcsolatban álló feladatokat koordinálja, segít a tervezési fázisban. A CV 2.0 hivatalos céljai alkalmazhatók az IFC2x3-as TC1-es szabványra. Továbbá a Coordination View létrehozása, kifejlesztése volt az első szempont a buildingSMART részéről és jelenleg ez a legjobban megtervezett nézete az IFC sémának. A 2. ábrán látható, hogy milyen szakágak is kapcsolódnak egymáshoz.



2. ábra A különböző építkezési területek együttműködése

³ A profil szó jelentése itt egy poligont takar, ami gyakorlatilag a falnak (ami egy téglatest is lehet) az egyik oldala és annak adott irányba történő eltolásával áll elő a fal. Téglatest fal esetén ez a legelső téglalapra merőleges irány segítségével toljuk el a poligont (ami a profil). Ennek megfelelően kapjuk meg a falat.

A fő célja a Coordination View-nek az, hogy egységessé tegye az épületmodellekről szóló információkat (Building Information Modelling – BIM) a fő építészeti tudományágak között, mint az épületgépészet, építészet és szerkezet építés. Definíciókat tartalmaz térbeli szerkezetekről, épületekről és épületben elhelyezkedő bármilyen szolgáltatást nyújtó elemekről, ahol ezeknek a tudományágaknak szükségük van megjelenési, dizájn információkra, a forma⁴ megjelenésére beleértve a parametrikus⁵ formákat az alapelemek szűk körében. Az adott elem saját tulajdonságainak halmaza, külső rétegek, mintázatok definíciói és egyéb leíró információk tehetők az elemre.

3.1. Milyen problémákra kínál megoldást a CV 2.0 almodell?

Az ArchiCAD 16 kiadásában már teljesítettük ennek a szabványnak közel 80 %-át, viszont a 17-es verzióban már szeretnénk teljesíteni a teljes követelmények listáját, ami várhatóan 2013-ra a következő ArchiCAD megjelenésre lesz készen. Bizonyos időközönként tartanak gyűléseket, ahol a legtöbb IFC szabványt alkalmazó cég megjelenik és az esetleges hiányosságokat felvetik, majd kifejlesztnek rá a buildingSMART-al közösen egy megoldást a szabványban. Viszont a CV 2.0 még a projekt kezdetekor nem volt teljesen készen, közben felvetődnek új kérések a gyártók, felhasználók részéről, ezért is a fejlesztés menetében az egyes fázisok ismétlődtek, mindig adottak az aktuális követelmények és ennek megfelelően elkészítjük, aztán leteszteljük az aktuális állapot megfelelő működését, majd addigra adottak lesznek az új követelmények. Amint készen lesz a szabvány teljes specifikációja utána készül el a teljes implementációja az ArchiCAD-ben is.

A problémák teljes tárházát, amelyre megoldást nyújt a CV 2.0 teljes egészében ezen TDK dolgozat keretein belül leírni nem lehet, viszont azt kijelenthetjük általánosságban, hogy a gyártók, akik alkalmazzák ezt a szabványt szigorúbb körülmények között hozhatnak létre modelleket, mint akik nem alkalmazzák. Ebből kifolyólag hatalmas előny származik belőle a felhasználók szemszögéből, hiszen a célszerűség miatt inkább azt a szoftvert fogják megvásárolni, amely implementálja a szabványt, mert ott ez egy közös interface az alkalmazások között és ennek következtében az exportált és importált fájlok biztosan megfelelően fognak megjelenni a felhasználó igényeinek megfelelően.

A legnagyobb előnye a CV2.0-nak az a GTDS-ben rejlik (Global Testing Documentation Server), ami egy központi oldal, ahova a gyártók feltöltik az *.ifc fájlokat végfelhasználók és a konkurens cégek számára. Ebből kifolyólag mindenki szabadon elérheti az egymás számára megosztott fájlokat és tesztelheti, javíthatja a hozzá kapcsolódó hibákat. Ráadásul ez kifejezetten elősegíti nem csak a CV 2.0 teljesítését, hanem a megfelelő implementáció a modell keretein túlnyúló problémákra. Természetesen ezeket a feltöltött fájlokat a buildingSMART-nál leellenőrzik az IFC szabványnak megfelelően.

⁴ Forma alatt egy háromdimenziós objektumot értünk, ami lehet egy piramisforma, téglatest vagy kocka is.

⁵ A parametrikus szó jelentése, az hogy van egy forma definiálva bizonyos paraméterekkel, ha a paraméterek közül egyet is módosítunk, akkor egy adott algoritmussal módosul minden más beállítása a formának, ami az adott paraméterre hivatkozik.

3.2. A modell miatt ArchiCAD-ben módosított elemek ismertetése

A projekt indulásakor még nem körvonalazódott ki teljesen melyek azok az elemek, amik nem felelnek meg a Coordination View 2.0-ás szabványnak, ez a GTDS segítségével folyamatosan körvonalazódik, mivel folyamatosan dokumentáljuk milyen eltérések vannak az ArchiCAD és a fájlhoz feltöltött *.pdf dokumentumban. Általánosságban kijelenthető, hogy a legtöbb újítás a geometriai megjelenést javította, illetve export során az *.ifc fájlban más típusú elemként írtuk le.

A munkahelyi vezetőm kérése alapján a csapat tagjait nem sorolom fel név szerint, de mindig egyértelműen jelzem, hogy mely eredményeket értem el egyedül, másokkal közösen, illetve mely eredmények kötődnek tisztán a többiekhez.

Így az alábbi elemek importján és/vagy exportján kellett módosításokat végezni:

- *IfcWallStandardCase* egyszerű falat lehet vele ábrázolni.
- *IfcWall* már bonyolultabb fal megjelenítésére is alkalmas, a különbsége a fenti változathoz képest később térünk ki.
- *IfcColumn* objektummal oszlopokat lehet ábrázolni.
- *IfcOpeningElement* a nyílásokat reprezentálja például egy falon.
- *IfcSite* pedig az épület földrajzi helyzetét határozza meg.
- *IfcSlab* lehet egy emelet padlózata, de akár egy irodaház lapos tetejének a szerepét is betöltheti.
- *IfcBeam* egy gerenda, ami egy családi házban a tetőszerkezetet is tartja.

Ezen elemekhez szorosan kapcsolódnak az következő felsorolásban kifejtett paraméterek, de nem mind értelmezett az összesre, külön kitérünk majd azokra, melyik paraméter melyik elemhez csatlakozhat. Most pedig jöjjenek részletesen kifejtve a teljesség igénye nélkül a paraméterek, tulajdonságok:

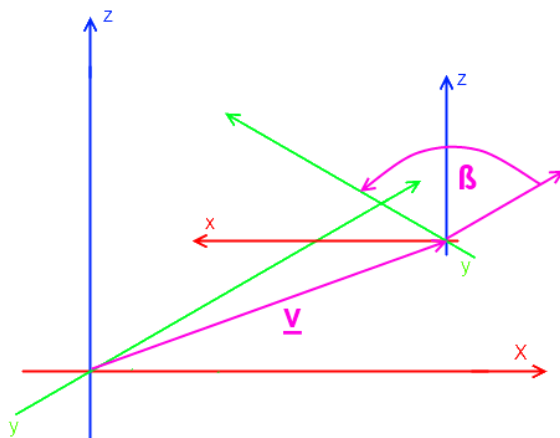
- *GrossFootPrintArea*: Ez egy bruttó területet jelöl, ami az alaprajzi kétdimenziós képen jelenik meg, logikusan azért bruttó, mert például egy fal nem tartalmaz semmilyen nyílást, ajtót, ablakot, ezek a kalkuláció során kívül esnek.
- *NetFootPrintArea*: Az előző fogalom alapján logikusan következik, hogy itt kivonjuk azokat a részeket, ahol ténylegesen megjelenik a kétdimenziós alaprajzon az ablak, ajtó. Így ténylegesen csak azt a területet kapjuk meg felülnézeti alaprajz alapján, ahol a fal látható.
- *GrossSideAreaLeft/Right*: Itt a test oldallapjainak a területét határozzuk meg és ezt hasonlóan a *GrossFootPrintArea* paraméterhez, vágások, openingek nélkül számítjuk ki külön a bal és a jobb oldalára. Az algoritmustól viszont erősen függ, melyik a bal és jobb oldal, amit számolunk.
- *NetSideAreaLeft/Right*: Számolható a *GrossSideArea* értékéből oda-vissza is egymásból, csak tudnunk kell a kapcsolódó geometriákat és azoknak a testtel érintkező felszínét kivonni a jobb vagy bal oldali oldallap területéből.
- *GrossVolume*: A teljes térfogat számítása szükséges, ahhoz hogy bővebben leírjunk bármilyen objektumot, amely használja ezt az értéket. Itt is – hasonlóan az előzőekhez – kikerülnek a számítás menetéből a testtel kapcsolatot létesített vágások, ajtók, ablakok.

- *NetVolume*: A nettó térfogat esetén is kivonjuk a csatlakoztatott elemek térfogatát az adott objektum térfogatából.
- *NominalLength*: A névleges hosszúsága az adott objektumnak, ami nem veszi számításba a levágásokat.
- *NominalArea*: Tipikusan opening elemeknél fordul elő vagy a vertikálisan elhelyezett területet számítják ki vagy pedig a kétdimenziós felülnézeti területet lehet vele leírni. Számítása nincs meghatározva, algoritmustól függ.
- *NominalVolume*: Az adott opening névleges térfogata. A számítás pontos menete az algoritmustól erősen függ.
- *NominalPerimeter*: *IfcSite* objektumnál használatos, amin elhelyezkedik minden épület és objektum, az adott *IfcSite* által lefoglalt terület kerületét határozza meg.
- *NominalWidth*: Névleges szélesség, amit általában *IfcSlab* esetén számolnak egy átlag értéként. Az *IfcSlab* egy beton réteg is lehet, amire felépítenek majd egy házat. Ebben az esetben a *NominalWidth* a vastagságát adja ennek a betonrétegnek.
- *CrossSectionArea*: Ez a paraméter tipikusan egy oszlop esetén a tetején és az alján elhelyezkedő oldalak területét jelenti, számítása szintén algoritmus függő.
- *OuterSurfaceArea*: Ha a *CrossSectionArea* az adott objektum tetejét és alját írta le, ez akkor a oldallapjának a felszínét fogja adni. Általában a kerület szorozva a magassággal (vágások esetén ez eltérhet).
- *TotalSurfaceArea*: Kétszer a *CrossSectionArea* plusz az *OuterSurfaceArea* összege adja a teljes felszín, szintén oszlopnál használatos.
- *GrossWeight*: Egy adott objektum bruttó súlya, nem tartalmazva a vágásokat, nyílásokat.
- *NetWeight*: Értelemszerűen itt már figyelembe vesszük a csatlakozó objektumok vágásait, így a *GrossWeight* értékénél kisebb számot kapunk.
- *Perimeter*: Tipikusan *IfcSlab* rendelkezik ezzel a paraméterrel, ami meghatározza a kerületét.

A fent felsorolt objektumokhoz kapcsolódnak más típusú paraméterek, amiknek a segítségével jelezni tudjuk, hogy milyen stádiumban van az adott oszlop, fal, gerenda, stb. Ezt az *IfcRelDefinesByProperties* asszociációs objektummal tehetjük meg, a konkrét értékeket pedig az *IfcPropertySet* halmazzal adhatjuk meg, ami lehet megépítés, dizájn, szerkesztés és a felújításon át akár a lebontási stádium is, amit már a bevezetésben is említettünk.

A következő ilyen paraméter a fent felsorolt objektum típusok esetén azt adja meg, mely anyagból épül fel az elem és mekkora eltolással rendelkezik a térben. Az *IfcMaterialLayerSetUsage* minden egyes anyaghoz adott mértékű vastagságot rendel fixen. Ebből fel lehet mérni az elemhez tartozó vastagságot annak függvényében, hogy milyen vastagság tartozik az adott anyaghoz. Tartalmaz még egy referencia vonalat is, ami mindenképpen pozitív értéket vehet csak fel és ez az eltolás adja meg, hogy honnét számítanak a hozzácsatolt anyagok és azok vastagságai.

Az *IfcWallStandardCase* önmagában egy falat definiál, amely elég sok megkötéssel rendelkezik mind a paramétereiben, amikkel le lehet írni, illetve a geometriai reprezentáció is megkötéseket tartalmaz. Csak olyan falat lehet vele ábrázolni, ami vertikálisan van extrudálva⁶. Továbbá a saját lokális és globális koordináta rendszerében is az extrúzió iránya a pozitív Z tengely mentén helyezkedhet el. A lokális koordinátarendszer megismeréséhez fűződik egy *IfcLocalPlacement* elem, ami minden geometriai elemhez hozzá csatolható és lényegesen leegyszerűsíti a lokális koordinátarendszerben történő módosításokat. Ezzel az elemmel elforgathatjuk, eltolhatjuk a közismert XYZ koordináta rendszert annak megfelelően, hogyan jelenik meg az elemünk.



3. ábra Példa az *IfcLocalPlacement* elforgatására és eltolására egy adott koordinátarendszerben

A 3. ábra egy *IfcLocalPlacement* vizuális leképezését is mutatja, ahol a kisebbik koordinátarendszer mutatja hova lett eltolva $\underline{v}$ vektorral illetve forgatva az XY sík Z tengely mentén β fokkal a koordinátarendszerünk. További megszorítás, ami az *IfcWallStandardCase*-re vonatkozik, hogy az oldalainak muszáj párhuzamosnak lennie egyenes fal esetén, illetve íves fal esetén is. Az alábbi paraméterek megadása szükséges ahhoz, hogy érvényes *.ifc fájlt állítsunk elő formailag:

- A fal magassága, amit az extrusion geometria nagyságából és annak irányvektorából számítunk, ebből már érthető miért csak Z tengely mentén pozitív irányba helyezkedhet el az extrúzió iránya.
- Vastagsága és az eltolása az *IfcMaterialLayerSetUsage* paraméterből jön (egy ilyen falra három réteget szoktak legalább hozzátenni), ennek a három rétegnek az anyagát lehet beállítani, ami automatikusan számítható, attól függ mi van erre a háromra beállítva. Így innét könnyedén meg lehet határozni a fal vastagságát. Az eltolását, pedig a – már említett – referencia vonalhoz képesti elhelyezkedésből lehet megadni.

⁶ Az extrudálás esetén adott egy poligon a háromdimenziós térben, amit egy adott irányvektorral és hozzá tartozó hosszúsággal eltolunk, így kapjuk meg a konkrét testet. A legegyszerűbb esetben a falhoz tartozó poligont egy téglalapként ábrázolhatjuk, amihez tartozik egy irányvektor és egy magasság, amivel eltolva megkapjuk a falat.

A szabványban külön van definiálva egy faltípus a bonyolultabb falak ábrázolására, érezhettük, hogy elég sok típusú fal van, amit az *IfcWallStandardCase* nem tud megfelelően leírni, erre az esetre hozták létre az *IfcWall* típust/objektumot a szabványban. Alapjában véve az eltérés a következőkben mutatkozott ki:

- Ez a típus képes ábrázolni az eltérő vastagságú falat, amely szűkül, vagy tágul, akár vertikálisan/horizontálisan.
- Tudja modellezni az általánosabb profilú falat, amely egy síkban elhelyezett poligon és ezt extrudáljuk felfelé, általában pozitív Z tengely irányába. Itt lehetőség van olyan falak ábrázolására is, amelyek „L” formájú profilt alkotnak, de bármilyen más profil is lehet ez a poligon.
- Lehetőség van Z tengelyi extrúziótól eltérni, bármilyen irányba mutathat az a vektor, ami meghatározza, merre történjen a fal kinyújtása.

Egy *IfcWallStandardCase* többféle geometriai típusként írható le, amik az alábbiak lehetnek:

- Curve2D az íves alaprajzok esetén használják, íveket lehet vele leírni, ha viszont egyenes falról van szó, akkor egy egyszerű poligonként kerül ábrázolásra, amiben nincsen semmi ív.
- A háromdimenziós ábrázolásra pedig *IfcExtrudedAreaSolid* geometriát használják, amit már korábban is taglaltunk, a hozzá tartozó vágásokat pedig az *IfcBooleanClippingResult* segítségével számítják. Amihez tartozik egy test, plusz a geometria, amin egy különbség műveletet végeznek, ami megmarad a falból, az kerül megjelenítésre.

Mivel az *IfcWall* az *IfcWallStandardCase* öse, ezért rá nem vonatkozik az összes megszorítás, amely az *IfcWallStandardCase*-re igen. Ezen túl még több információt is meg lehet róla adni, ahhoz hogy bővebben leírjunk egy ilyen típusú falat a fentebb már kifejtett paraméterek segítségével:

- *GrossFootPrintArea*
- *NetFootPrintArea*
- *GrossSideAreaLeft/Right*
- *NetSideAreaLeft/Right*
- *GrossVolume*
- *NetVolume*

Bármilyen oszlop ábrázolását az *IfcColumn* segítségével valósíthatjuk meg. A konkrét típushoz az *.ifc fájlban felhasználható egy *IfcColumnType* objektum, amelynek segítségével módosíthatjuk egy oszlop értelmezését az adott programban, alapértelmezett értéként mindent *column* típusként kell menteni. Az *IfcColumn* és a típusának a csatlakoztatását az *IfcRelDefinesByType* írja le, ami egy a többhöz kapcsolatot tesz lehetővé. A geometriai reprezentációja az *IfcExtrudedAreaSolid* segítségével történik, hasonlóan az *IfcWallStandardCase* reprezentációjához. Az *IfcColumn* esetén beállíthatók a következő paraméterek:

- *NominalLength*
- *CrossSectionArea*
- *OuterSurfaceArea*
- *TotalSurfaceArea*

- *GrossVolume*
- *NetVolume*
- *GrossWeight*
- *NetWeight*

Az *IfcOpeningElement* egy nyílást határoz meg egy falban, ami nem csak ajtó vagy ablak lehet, még csak nem is vágás, hanem egy olyan mélyedés, ami nem lyukasztja ki a falat. Így két esete lehetséges egy *opening*nek:

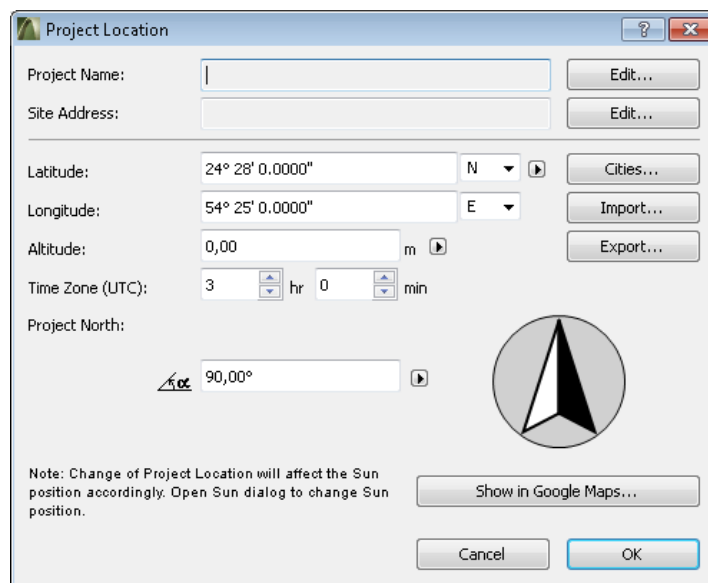
- Az *opening* vastagsága vagy megegyezik a fal vastagságával vagy annál nagyobb. Értelmszerűen, ha egy *opening* jelentősen túlnyúlik a falon és esetleg geometriailag vágást generálna egy másik testben is, akkor az csak akkor fog vizuálisan is megjelenni, ha ezt a vágótestet hozzákapcsoljuk az adott testhez is.
- A vastagság kisebb is lehet, mint a fal vastagsága, ebben az esetben csak részleges vágás keletkezik, ami nem lyukasztja ki a falat.

Az, hogy melyik változatról van szó azt *ObjectType* attribútuma határozza meg, ahol a „*recess*” jelöli a részleges vágást, az „*opening*”, pedig a teljes lyukasztást. Egy paramétere van, amit opcionálisan ki lehet vinni egy *.ifc fájlba, az pedig az *OpeningArea* nevet képviseli, jelentését pedig korábban tárgyaltuk már. A geometria típusa itt csak és kizárólag *IfcExtrudedAreaSolid* lehet és leginkább horizontális (állófalban lévő ablak pl.) és vertikális (járólapban elhelyezkedő ablak) iránnyal rendelkezhet.

A következő objektum megadja, hogy az adott épület, amit az ArchiCAD-ben létrehozunk a valós földrajzi viszonyok között hol helyezkedik el, mekkora területet fed le. Ezért is az *IfcSite* névvel írja a szabvány. ArchiCAD-ben alapvetően egy *IfcSite* megjelenítését tesszük lehetővé, ezeken több épület is szerepelhet, ebből csak egyet tudunk megjeleníteni egy fájlból, amit egy dialógus ablakon keresztül választhatunk ki és az tölti be a program. Az, hogy földrajzilag pontosan hol helyezkedik el az adott épület három paraméterrel adható meg:

- Hosszúsági fok (Longitude)
- Szélességi fok (Latitude)
- Magasság (Altitude, Elevation)

A magasság értékét egyszerűen méterben meg tudjuk adni ArchiCAD-en belül. A hosszúsági és szélességi fok adatokat is meg lehetne adni egy számmal, de ezt a szabvány nem így írja elő (mivel meg kell adni, hogy hányadik foknál, szögpercnél és szögmásodpercnél helyezkedik el az adott terület).



4. ábra Az *IfcSite* magasságára, hosszúsági- és szélességi fokára szolgáló dialógus ablak az ArchiCAD-ben

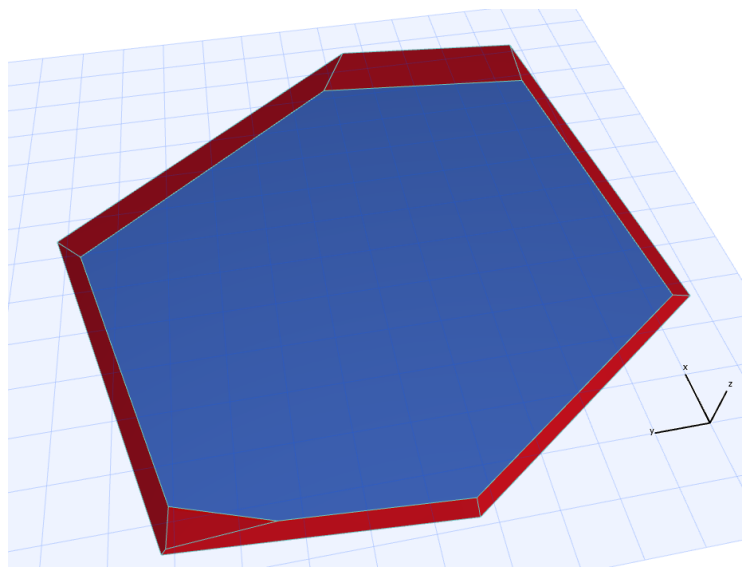
A 4. ábra jelöli, hogyan adható ez meg az ArchiCAD-ben. A fájlba kimentett *IfcSite* esetén ezek az értékek különféle típusként vannak kezelve. A magasságot a szabvány az *IfcLengthMeasure* valós típusként viszi ki általában milliméterként. A hosszúsági és szélességi értékeket pedig három vagy négy szám alapján lehet megadni. Ebből csak három áll rendelkezésre, amint a mellékelt ábrán is látszik (fok, szögperc, szögmásodperc) a negyedik az ezredmásodpercet jelöli, ezt már nem tudjuk kimenteni, de szerencsére megadása opcionális. A szabvány az *IfcCompoundPlaneAngleMeasure* típusként kezeli.

Az *IfcSlab* objektumnak a használata már nem olyan triviális, mint ahogy gondoljuk. Geometriailag több objektumhoz is hasonló lehet, ábrázolhatunk vele *IfcBeam* és *IfcColumn* példányokhoz hasonló testeket. ArchiCAD-ben két elem típusaként jelenik meg, ahol az egyik a tető, másik pedig egy vízszintesen elhelyezkedő födém. A kettő közötti különbségeket import során elsősorban az *IfcSlab* előre definiált típusaként tudjuk eldönteni milyen elemmel is van dolgunk. Többféle értéket is felvehet, amelyek az alábbiak:

- *Floor*: Egy épületben elhelyezkedő vízszintes emeletről van szó, ami nem a földszint. Ez az alapértelmezett értéke egy *IfcSlab* objektumnak.
- *Roof*: Itt már ritka a vízszintes geometria, de előfordulhat nulla fokos tető esetén.
- *Landing*: Épületen kívül elhelyezkedő földterület ábrázolására alkalmas.
- *Baseslab*: Épület földszintjén elhelyezkedő födémről van szó.
- *Notdefined*: Ha a tervező program nem foglalkozik a födém funkciójáról, akkor szokás így kimenteni.
- *Userdefined*: Ilyenkor a felhasználó határozza meg, milyen funkciót tölt be egy tervrajzon az adott *slab*.

Egy ilyen *slab*-et geometriailag ajánlott *IfcExtrudedAreaSolid* háromdimenziós leírással ellátni, mivel a legtöbb analizáló program ezekkel a geometriákkal dolgozik. A legegyszerűbb eseteken egyszerűen lehet ezzel ábrázolni, viszont a bonyolultabbakat, ahol már állítva van az dőlésszöge a

slab egyes oldalainak, arra sajnos nem elég. Erre viszont tökéletesen alkalmas az *IfcFacetedBrep*. A következő ábrán érezni fogjuk, miért jó ebben az esetben így kiexportálni egy födémeket.



5. ábra Egy *IfcSlab* 3D-s felülnézete oldalszögekkel beállítva

A fenti 5. ábra kék része jelzi a *slab* belső részét, amit ha oldalszögek nélkül ábrázolnánk, akkor a piros része a *slab*-nek jóval kisebb lenne, hiszen az jelöli a plusz oldalszögek miatt keletkezett részt. Ezeket a dőlésszög miatt *ExtrusionData*-ként elég nehéz lenne ábrázolni (vágásokat kellene elhelyezni rá, de azt a szabvány sem javasolja), így mindegyik oldalt külön poligonnal írjuk le *BREP* formájában.

Az *IfcSlab*-nek is több paraméterét meg lehet adni, amiknek az értéke erősen függ a használt algoritmustól, a paraméterek pedig:

- *NominalWidth*
- *Perimeter*
- *GrossFootprintArea*
- *NetFootprintArea*
- *GrossVolume*
- *NetVolume*
- *GrossWeight*
- *NetWeight*

Lehetőség van gerendákat is kivinni egy *.ifc fájlba, amely az *IfcBeam* fedőnevet képviseli. Egy ilyen gerendának a típusát az *IfcBeamType* segítségével lehet meghatározni, aminek az értékei az alábbiak lehetnek:

- *beam*: ez az alapértelmezett értéke, ha a tervező program nem foglalkozik a pontos kategorizálással, akkor így kerül mentésre.
- *joist*: az a típusú gerenda, ami a padlózatot tartja
- *lintel*: fejgerenda, amelyet tipikusan egy ajtókeretnek a legfelsőbb, vízszintes részét jelöli, de hidak építésénél is ennek hívják a tartó oszlopokon elhelyezett gerendákat.
- *t-beam*: építkezéseknél használt T alakú vasrudak geometriájával bíró gerendákat lehet ebbe a csoportba sorolni.

A geometria típusa a *slab*-hez hasonlóan *IfcExtrudedAreaSolid* lehet, más azonban nem. Egy fontos különbség viszont van, mivel ArchiCAD-ben *slab*-et nem lehet dőlésszöggel ellátni, csak tetőt, de az külön is van kezelve. Itt viszont mivel a geometria, amit extrudálunk általában függőleges, ezért az extrúzió iránya határozza meg, mekkora dőlésszöggel van ellátva az adott *beam*. Vágások szintén megengedettek az adott *beam*-re, erre szolgál segítségül az *IfcBooleanClippingResult*, amint azt már említettük. A *beam* esetén rendelkezésre álló egyéb adatok, paraméterek pedig:

- *NominalLength*
- *CrossSectionArea*
- *OuterSurfaceArea*
- *TotalSurfaceArea*
- *GrossVolume*
- *NetVolume*
- *GrossWeight*
- *NetWeight*

4. AZ ARCHICAD ÉS A CV 2.0

Minden ArchiCAD-beli elem rendelkezik az IFC szabványnak megfelelő reprezentációval, ami némelyik esetben nem egy az egyhez kapcsolatot jelöl. Például egy ArchiCAD-beli tetőt (roof) IFC-ben többféle módon is leírhatunk. Egy egyszerű irodaház teteje, ami vízszintes lapos tető is lehet azt *IfcSlab* formájában visszük ki, amelynek egyik paraméterében jelezzük, hogy ezt tetőként kell a későbbi importok során kezelni. Azonban, ha bonyolultabb komplex formája van, akkor lehetőség lenne *IfcRoof*-ként is kivinni, de ennek a megvalósítása még nincs tervbe véve. Ezt tartsuk szem előtt, hogy nem feltétlen direkt leképezésről van szó az ArchiCAD és az IFC elemek között, persze ez igaz más szoftverek exportjára is. A 4.2 pontban taglalt javításokról, fejlesztésekről elmondhatjuk, hogy terjedelmük arányos a befektetett munka nagyságával.

4.1. A csapat segítségével általam megvalósított új funkciók bevezetése

A folyamatos projektmegbeszélések során minden fontos kérdésre kitértünk és az optimális megoldás implementáltuk, fontos szempontként követve a felhasználók számára a megfelelő funkciók előállítását. Így az általam megvalósított funkciók röviden összefoglalva:

- *IfcWallStandardCase*: Megfelelő *opening* vágások kezelése, mivel teljes vágásokat hoztunk létre, akkor is amikor nincs rá szükség.
- *IfcColumn*: Megfelelő elhelyezése az alsó- és felső vágásoknak a geometria módosításával.
- *IfcBeam*: Jobbsodrású koordinátarendszerben történő input biztosítása ellenőrző függvények segítségével és íves geometria megfelelő exportálása.
- *IfcSlab*: Egy síkban elhelyezkedő *roof* beimportálása két különböző geometriai csoportból előállítva. Illetve egy ilyen *slab*-ben elhelyezett vágások mentése ügyelve az egy az egyhez kapcsolat megtartásához.
- *IfcOpeningElement*: Falban elhelyezett teljes és részleges vágások exportálása.
- *IfcSite*: Egy épülethez tartozó hosszúsági-, szélességi fok és a magasság megfelelő exportálása és importálása, illetve IFC Manageren való megjelenítése.

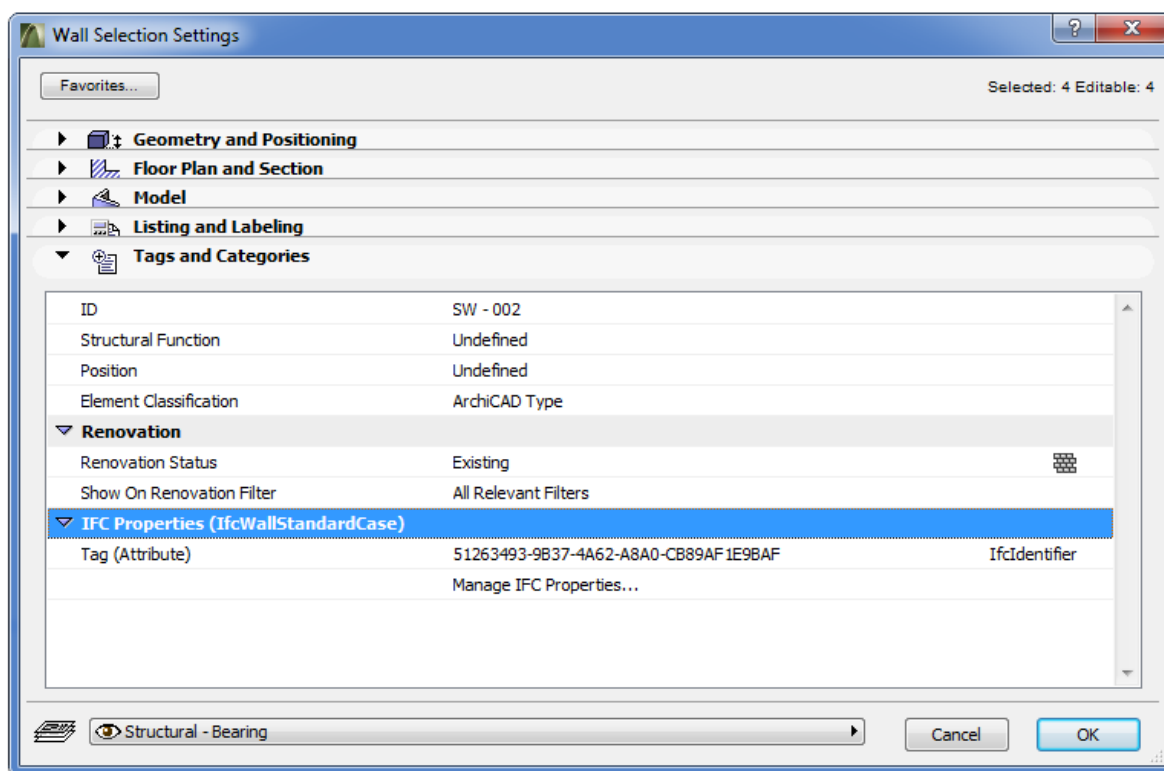
Ezek a javítások meg fognak jelenni a legújabb ArchiCAD kiadásakor, amelynek tervezett időpontja 2013 nyara. A csapattársakkal közösen együttműködve szeretnénk elérni az IFC2x3 Coordination View 2.0 teljesítését. A fejlesztés során eddig összesen 487 órát dolgoztam a projekten (ez 8 órás munkanapokban ~61 napot jelent), míg a projektre fordított teljes idő 1414 munkaórát foglal magába, ez felöli a csapattársak munkáját is. A saját munkám több kategóriákra bontható, mint:

- Implementáció tervezés
- Hibajavítás és implementálás
- Kód analízis, review-olás a csapattársakkal
- Konzultáció a csapattársakkal

4.2. A CV 2.0 miatt megjelent új módosítások az ArchiCAD-ben

4.2.1. Az egyszerű fal speciális reprezentációs eseteinek újításai

Az ArchiCAD-ben szereplő elemek IFC szerinti megjelenését az adott elem beállításainál tudjuk megtekinteni. A lenti ábrán látható, hogy ki van jelölve négy falra vonatkozó dialógus ablak, amelyen ez a négy fal szerkeszthető is. A könyv végén felhasznált szakirodalmak közül, [5] Simon Gilbert könyve részletesebben is taglalja ezeket a beállítási lehetőségeket az 5. fejezet első útmutatójában. Visszatérve a 6. ábrán kijelölt négy fal dialógusablakára, láthatjuk azt is mire van beállítva az *IFC Properties* mező értéke, ugyanis az lesz az *.ifc fájlban történő leképezése. Esetünkben az *IfcWallStandardCase*, ugyanis ahogy a neve is sugallja ez a forma csak a falnak az egyszerű eseteit képes ábrázolni, ezért is van szó speciális reprezentációs esetekről. Az egyéb bonyolultabb eseteket, ahol már az oldallapok nem párhuzamosak azt az *IfcWall* képes ábrázolni a szabvány szerint.



6. ábra Falak beállítására szolgáló dialógus ablak egy része

Korábban taglaltuk milyen értékeket lehet megadni egy *IfcWallStandardCase* esetén ezt a gyakorlatba átültetve nézzük meg, hogy a fájlban kimentve, hogyan jelenik meg:

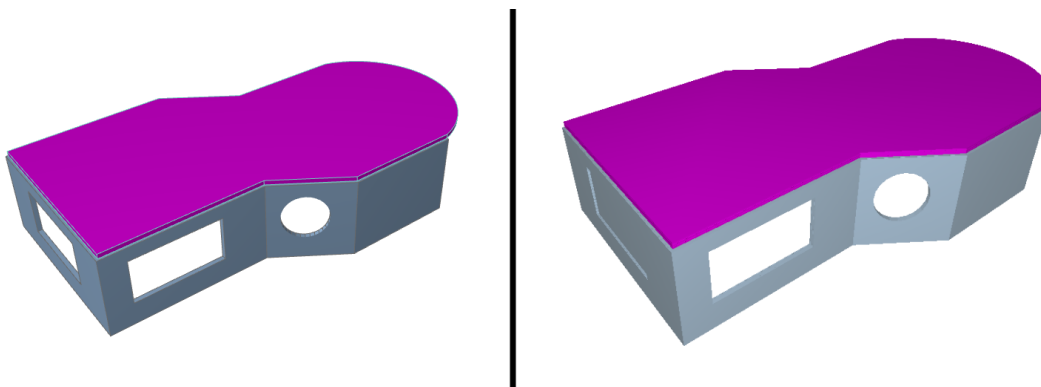
```
#267= IFCWALLSTANDARDCase('0nJHO7yYP32xU8i8usS1s8',#16,'SW - 001',  
'description','objecttype',#175,#263,  
'314D1607-F226-430B-B788-B08E36701D88');
```

A sharp jellel ellátott szám azonosítja be a fájlban az adott falat, ez minden objektum esetén így kerül megadásra, majd egyenlőségjel után következik az objektum típusa és hozzá a szükséges paraméterek, mint az egyedi azonosító, aminek segítségével az összes objektum közül pontosan meg lehet határozni melyik elemet kezeljük, szerkesztjük éppen IFC szerint. Látható, hogy az egyedi azonosító után, egy másik fájlbeli elem kerül hivatkozásra (#16), ami az *IfcOwnerHistory* objektumot jelöli. Ezzel adható meg minden fontos adat a fájlban a tulajdonosáról, milyen programmal lett elkészítve, mely cégnél állították elő. A harmadik paraméter, az adott fal neve, ezt általában a rendszer automatikusan generálja, a sorszám növelésével, ezeket követi a leírás és az objektum típusa, amit a felhasználó szabadon begépelhet. Aztán jön a geometriai kapcsolatot reprezentáló két hivatkozás az egyik az eltolás, elforgatás mértékét határozza meg ez a már korábban tárgyalt *IfcLocalPlacement*, a másik pedig az *IfcProductDefinitionShape*, ami a geometriai formát határozza meg (ez lehet BREP, Extrusion, illetve ami engedélyezett egy fal esetén). Az utolsó paraméter pedig hasonló az elsőhöz, azzal a különbséggel, hogy ez nem az IFC sémában határozza meg az elemet, hanem a tervezőprogramban, vagyis az ArchiCAD-ben.

Egy valamit viszont ne felejtsünk el, az előbb felsorolt paramétereken kívül is vannak még elemek, amiket már korábban bemutatunk, mint például az *IfcMaterialLayerSetUsage*, *IfcPropertySet*, *IfcMaterial*. Ezekhez még hozzátartoznak a szükséges asszociációs objektumok, amik létrehozzák a kapcsolatot a két elem között (pl.: egy *IfcWall* és egy *IfcMaterial* között). A későbbiek során tartsuk ezt szem előtt, mivel ez minden esetünkben tárgyalt objektumra vonatkozik.

A Coordination View 2.0-nak köszönhetően megjavult funkciók:

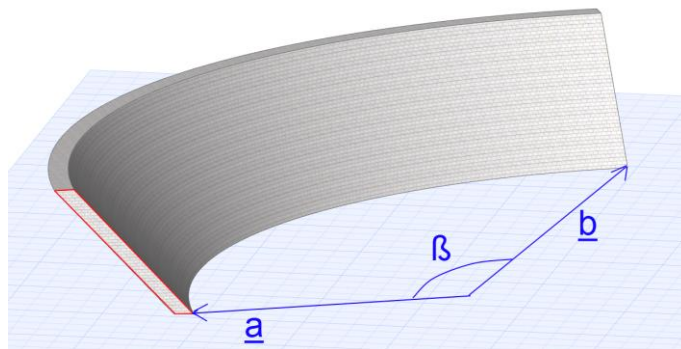
- Vágások megfelelő kezelése, amit általam került implementálásra
- Revolved geometria exportja és importja, ez a fejlesztés a csapattársak érdeme
- Íves fal esetén az ív irányának a megfelelő exportja szintén kollégák készítették el.



7. ábra A vágás hibajelenség a régi verzióban (bal oldalt) és a megfelelő megjelenítése (jobb oldalt)

A 7. ábrán látszik is az első fejlesztéssel kapcsolatos hiba (vágások megfelelő kezelése), hogy túl nagy a vágás, amit ráteszünk a falra, ilyenkor a hiba nem az ArchiCAD motorjában van, hanem az import folyamat során mi állítjuk be rosszul a vágásra vonatkozó paramétereket. Jelenleg az ArchiCAD 16-ban sajnos benne van ez a hiba, amit a fenti ábra is mutat. A megoldás módja már korábban megfelelően implementálták az ArchiCAD source-ban, csak rossz ágon futott végig az import során a lyukak geometriai beállítása. A vágás geometriája *IfcExtrudedAreaSolid*, mivel a vágás egy síkba esik a fal valamelyik oldallapjával, ezért az extrusion iránya és a fal vastagságának összehasonlításával kiszűrhetők azok az esetek, melyeknél alkalmazni kell a már létező módszert és megfelelően importálni a vágást a falba. A vágás geometriáját az *IfcExtrudedAreaSolid* profilja biztosítja, ami egy poligon azt nyújtjuk ki az *extrusion* irányának megfelelően. Ezt a fajta vágást *empty opening* is nevezzük és ablakként van kezelve, hiába nem keletkezik lyuk a falon. Ez a javítás a felsorolt három közül hozzám fűződik, a többi kettőt a csapattagok implementálták, mint azt már említettük.

Az íves (revolved) geometria export/importjánál sok fontos szempont van, ami befolyásolja a megfelelő megjelenítést. Alapvetően a ferde íves falaknál lehet ezt jól kihasználni, hiszen a 8. ábrán látható piros poligont kell az adott β szöggel elforgatni az **a** vektor segítségével. Ennek köszönhetően, már nem BREP-ként történik a mentés, hanem Revolved-ként, ami rengeteg feleslegesen szegmentált ponttól kíméli meg a megjelenítést, illetve az *.ifc fájlt. Az ábrán a legegyszerűbb eset látható, de előfordulhat olyan, hogy a fal több rétegből épül fel, ilyenkor vizsgálni kell mindig a poligont és a vektorhoz tartozó szögeket is, hogy képezhető-e belőle egy adott fal. Ezeket a részeket *buildingElementPart* névvel látták el, ha ablak, ajtó kerül elhelyezésre a falon, akkor pedig nem ezekhez a kis részekhez rendeljük hozzá, hanem globálisan a falhoz, amely tartalmazza a *buildingElementPart*-okat is. Viszont, ha nem lehet import során csoportosítani ezeket a részeket, akkor minden egyes *opening*-et külön alkalmazni kell mindegyik részhez is. Illetve, ezekhez a kis részekhez, a *buildingElementPart*-okhoz külön rendelünk profilt, ami gyakorlatilag a 8. ábrán a piros poligon sokszorozva annyiszor, ahány *buildingElementPart* létezik.



8. ábra Az *IfcRevolvedAreaSolid* geometria egyik lehetséges formája

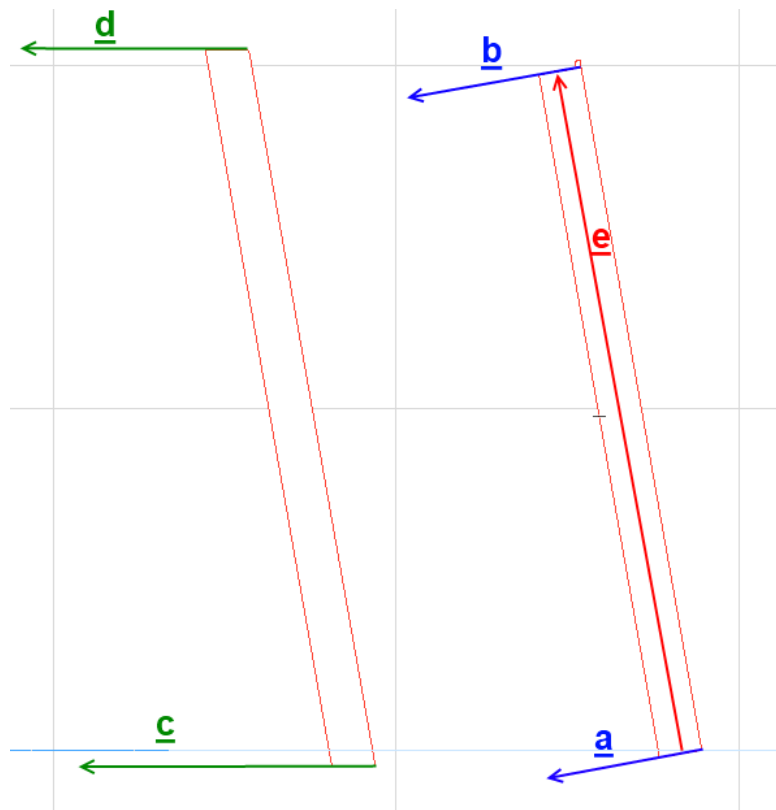
A felsorolás harmadik pontja az ív irányának korrekt mentésére vonatkozik, ugyanis jelenleg egy húr menti irányt exportálunk ki, ami némelyik másik program használatánál, a beimportálásnál problémát okozhat, és akár a geometriai megjelenítés rovására is mehet. A megfelelő export az pedig érintő szerinti irány mentését írja elő.

4.2.2. Ferde oszlopok megfelelő vágása

Lássuk, hogyan is néz ki egy ilyen *IfcColumn*, milyen paraméterei vannak és a vágások hogyan is jelennek meg formailag rajta. A fájl szerinti leíró adatai az alábbiak:

```
#511= IFCCOLUMN('3ngPet0y5CDhZeHUNArDjN',
#16,'Column_1-11',$,'L250X28',#175,#507,
'F1A99A37-03C1-4C36-B8E8-45E5CAD67B57');
```

Látható, hogy szinte ugyanazokat a paramétereket lehet itt is megadni, mint a fal esetében a vágásokat pedig *IfcBooleanClippingResult* formájában kezeljük. Mivel több vágás is elhelyezkedhet egy ilyen oszlopon, ezért minden vágás mellé opcionálisan meg lehet adni még egyet, ennek köszönhetően végtelenségig lehet csatolni vágásokat egymáshoz, mint egy láncolt lista. A lenti ábrán látható egy oszlop oldalnézete, aminek a dőlési szöge 80°. Pontosan ilyen oszlopot elég nehéz előállítani ArchiCAD-ben, mint a jobboldali oszlop (csak megfelelően elhelyezett vágásokkal lehetne) azonban baloldalon látható oszlopot bármikor könnyedén előállíthatunk. A jobboldalon látható példa is egy AutoCAD exportból származik.



9. ábra ArchiCAD-ben megjelenített oszlop oldalnézete AutoCAD-ből exportálva

A fenti ábrán látható két pirossal jelölt oszlop oldalnézete, illetve a hozzá tartozó vektorok. A bal oldali oszlopot ArchiCAD-ben hoztuk létre a jobb oldali viszont egy beimportált oszlop, amit körülményesen vágások segítségével lehetne előállítani eme tervező programban. Az importálás során a vágások kezelésének elhanyagolása miatt egy ArchiCAD által értelmezett – bal oldali – oszlop állt elő. Ez viszont számunkra nem jó, hiszen a jobb oldali oszlop formáját kell, hogy öltse.

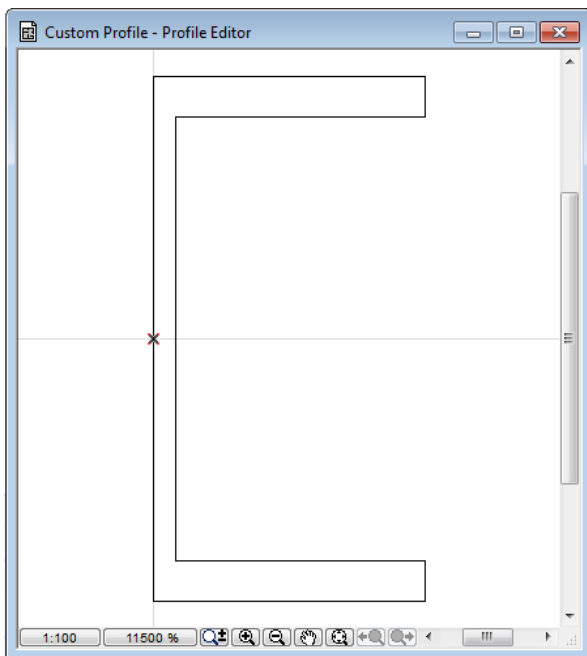
Nagy segítségünkre szolgálnak azonban a *gable*-k, amelyek vágásokat helyeznek el az import folyamat során az oszlopokra, így elő tudjuk állítani a számunkra megfelelő formát, amit a jobb oldalon is láthatunk. A konkrét érthetőség miatt, tekintsük meg a zöld színnel jelölt **c** és **d** vektort, amelyeknek iránya vízszintes, ellenkező esetben a kék **a** és **b** vektorhoz képest. Az utóbbi két vektor jelöli a vágásokat, amiket az oszlopra teszünk rá annak érdekében, hogy az alap poligon, ami felfelé ki van nyújtva⁷ – **e** vektor irányába – létrehozzon egy olyan oszlopot, amelynek a dőlés szöge nem csak az oldallapjaira, hanem az egész oszlopra vonatkozzon.

Meg kell említeni, hogy van egy nagyon hasonló elem az oszlophoz, ez pedig a gerenda. Az egyik általában merőleges a talajra, a másik pedig vízszintes. Viszont előfordulhat ferde oszlop is és gerenda is, itt kerül felhasználásra az IFC-beli típusmegnevezés *IfcBeam* vagy *IfcColumn*. Ebből kifolyólag, ha jön egy oszlop, ami teljesen vízszintes akkor abból ArchiCAD-ben gerenda lesz és nem oszlop. Természetesen ez a függőleges gerendára is levezethető, hogy oszlop lesz belőle. Továbbá a gerendára is ugyanúgy vonatkozik ez a hiba, ami javításra fog kerülni ebben az ArchiCAD verzióban, csak nem ennek a projektnek a kapcsán.

4.2.3. Íves és egyenes gerendák tükrözésének problémája

Egy gerenda létrehozásához szükséges paraméterek ugyanazt a szintaktikai és szemantikai jelentéssel bíró elemeket használnak, mint a korábbiakban. A teljesség kedvéért itt egy példa, amelyet a korábbiakban is bemutatunk:

```
#237= IFCBEAM('0y5J5QcmP2HPe3Bg83x_Sw',
#16,'BMR - 001',$,'200X200',#175,#233,
'3C15315A-9B06-4245-9A03-2EA203EFE73A');
```



10. ábra Egy gerenda profilja, ami háromdimenzióban kerül kinyújtásra

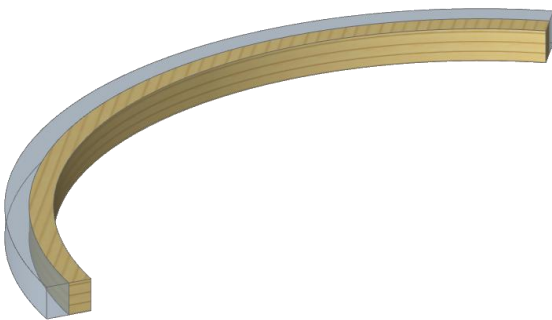
Két alapvető probléma is volt a tükrözéssel, az egyik geometriailag tükrözte meg a gerendát, lefelé, gyakorlatilag az alsó oldalára, de eltolásként is elképzelhetjük a műveletet, ahol a gerenda Z irányú magassága határozta meg a mértékét. A 10. ábrán a Z iránynak a leghosszabb él felel meg, amin az „x”-el jelölt origó is van. Ezen túl a másik tükrözés pedig a gerenda profiljában jelentkezett, amit a szomszédos ábrán is láthatunk. Ugyanis, ha az **.ifc* fájlban manuálisan kézzel megváltoztattuk bármely pont koordinátáit, egy kicsit lejjebb tettük, akkor az ellenkező oldalon történt a módosulás, míg a Solibri Model Checker tökéletesen megjelenítette, tükrözés nélkül. Ez a probléma azért történhetett meg, mert amikor implementálták az *IfcBeam* importálására szolgáló metódusokat, akkor nem számítottak arra az esetre, ha nem jobbsodrású koordináta rendszert adnak meg az *IfcLocalPlacement*-be. Viszont ez

⁷ Ez gyakorlatilag az **a** vektor mögötti piros részt jelölné az oldalnézet esetén, ami saját magára merőlegesen van kinyújtva és így áll elő az oszlop.

akkora módosítást igényel, ami túl nagy fejlesztés lett volna és ez nem képezi részét a CV 2.0-nak sem. Ezt a javítást is jómagam implementáltam az ArchiCAD-ben, de az ötlet egyik kollégámtól jött. Ennek köszönhetően a beimportálás során nem gerenda, hanem egy objektum/*object* készül belőle. Egy ilyen *object* bármit ábrázolhat, amire nincs megfelelő leképezés az ArchiCAD-ben történő ábrázolásra. Egy hátránya van, az a sebességbeli létrehozás, mivel GDL scriptet generál belőle az ArchiCAD. Ezzel a scripttel bármit lehet ábrázolni, egész könyvtárak találhatók meg az ArchiCAD-ben, amik foteleket, asztalokat, székeket és még sok mindent tartalmaznak. ArchiCAD-ben tegyük le egy ilyen *object*-et és a beállításainál (*Ctrl + T*) elő tudjuk hozni ezeket az elemeket és a rendelkezésre álló geometriai formák közül bármit beállíthatunk ezekre az *object*-ekre. A GDL scriptek elég erős matematikai tudást igényelnek, mit hogyan lehet benne megjeleníteni. Eltolások, forgatások történnek folyamatosan a térben és eltérő geometriákat lehet megadni különféle függvényhívásokkal. Az irodalomjegyzékben van egy könyv, ami ehhez szorosan kapcsolódik, [1] *GDL felhasználói kézikönyv* címszó alatt.

A problémát úgy oldottuk meg, hogy leellenőriztük a beolvasott *IfcLocalPlacement* mindhárom (X, Y, Z) tengelye derékszöget zár-e be és bármely két tengely vektoriális szorzata kiadja-e a maradék harmadikat. Ennek köszönhetően megfelelően megjelenik, némi lassúságért cserébe a gerendánk. A jövőben tervezzük kijavítani ezt az hibát, úgy hogy ne *object*, hanem *beam* jöjjön létre.

Az előző megoldás az egyenes gerendánál volt jelen, ami már ArchiCAD 16-ban is előidézhető, viszont az íves gerenda jelenleg nem képezi részét az ArchiCAD 16-nak, ez a 17-es verzióban került implementálásra és mivel íves geometriával rendelkezett, ezért *IfcRevolvedAreaSolid* formájában vittük ki az *.ifc fájlba. Ennek fejlesztését a csapattagok végezték, viszont egy szükséges javítást kellett bevezetni, amivel már én lettem megbízva. A probléma az volt, hogy a profil, amit nyújtottunk ívesen a térbe óramutató járásával megegyezően volt kezelve és abból indultunk ki, hogy mindig óramutató járásával ellentétesen történik ez a nyújtási/extrúziós⁸ folyamat.



11. ábra Az íves gerenda téves működésének illusztrálása

Az ArchiCAD-ben az *IfcRevolvedAreaSolid* reprezentálása eltér ahhoz képest, ami az *.ifc fájlban jelen van, ugyanis egy kezdő- és egy végpontra van szükségünk és a kettő közötti ívet egy radiánban megadott szöggel adhatjuk meg, vagyis ha egy félkört akarunk leírni két pont között, akkor π értéket megadva a két tetszőleges pont közé megkapjuk a félkörben nyújtott gerendánkat. A szomszédos 11. ábrán látható a javítás előtti állapotnak az illusztrálása. A barna színű gerenda az, amit kiexportáltunk és, ha ezt beimportáljuk, akkor kapjuk meg – tévesen – a jobb oldali, világoskék színű

gerendát. Az elvárt működés az lenne, hogy a két gerenda teljesen lefedje egymást.

⁸ Furcsa lehet, miért extrúziót emlegetünk itt is, hiszen az az *IfcExtrusionAreaSolid* geometriánál bír jelentéssel, viszont az *IfcRevolvedAreaSolid* is ugyanúgy kerül nyújtásra, mint az *extrusion* csak nem egyenesen, hanem ívesen.

ArchiCAD-ben ehhez a körívhez tartozik egy origó, illetve két pont (kezdő és végpont). Könnyedén el tudunk képzelni egy 2D koordináta rendszerben egy profilt (a 11. ábra esetén egy téglalapot) ami a bal felső negyedben helyezkedik el a koordináta rendszerben, a profil origótól vett x távolsága adná meg a sugarat. Amire mi számítunk, az az, hogy ez a profil mindig a pozitív x tengelyen belül esik (a jobb felső negyedbe) valahova. Ez viszont pont ellenkezőleg történik, mivel negatív tartományban exportáltuk ki ezt a profilt, ami az IFC szabvány szempontjából lényegtelen, de import során nem foglalkoztunk ezzel az ArchiCAD-ben. Ehhez a javítás annyiban merül ki, hogy felcseréljük a kezdő és a végpontot, majd a szöveget megneveljük, így a korrekt helyes reprezentációt kapjuk meg, ami ki lett exportálva.

Gondoljuk át még egyszer mi történik, akkor ha felcseréljük a kezdő és a végpontot, mivel ArchiCAD-ben óramutató járásával ellentétesen történik ennek a megadása (import során) így egy félkör esetén egy megtükrözött félkört kapnánk meg, ez még mindig nem a helyes forma számunkra, ezért van szükség a radiánban megadott szögnek a negálására, mivel így nem a külső íven fog elhelyezkedni a profil, hanem a belsőn. Ez a külső ív a 11. ábrán a két gerenda közös oldalának bal oldali területe, tehát ami a belső íven van ábrázolva az a barna gerenda. A negatív szög értékkel pedig pluszba még át is fordítjuk arra az oldalra, ahonnan ki lett mentve.

A hiba javítása során viszont észleltünk a Solibri működésében egy kisebb hibát, ami szintén hasonlít ennek a hibának a felépítésére. ArchiCAD-ben elő lehet állítani olyan íves *slab*-et is, aminek a nyújtási irány óramutató járásával megegyező, csak tükrözni kell egy adott elemet és máris megváltozik a körüljárási iránya. Egy ilyen *slab*-et kimentve szembesülhetünk az elem tükrözésével Solibri-be importálva. Munkatársam már jelezte felénk ezt az esetet, hogy ők is tudjanak róla és ki tudják javítani, még mielőtt mi is piacra bocsátanánk ezt a funkciót.

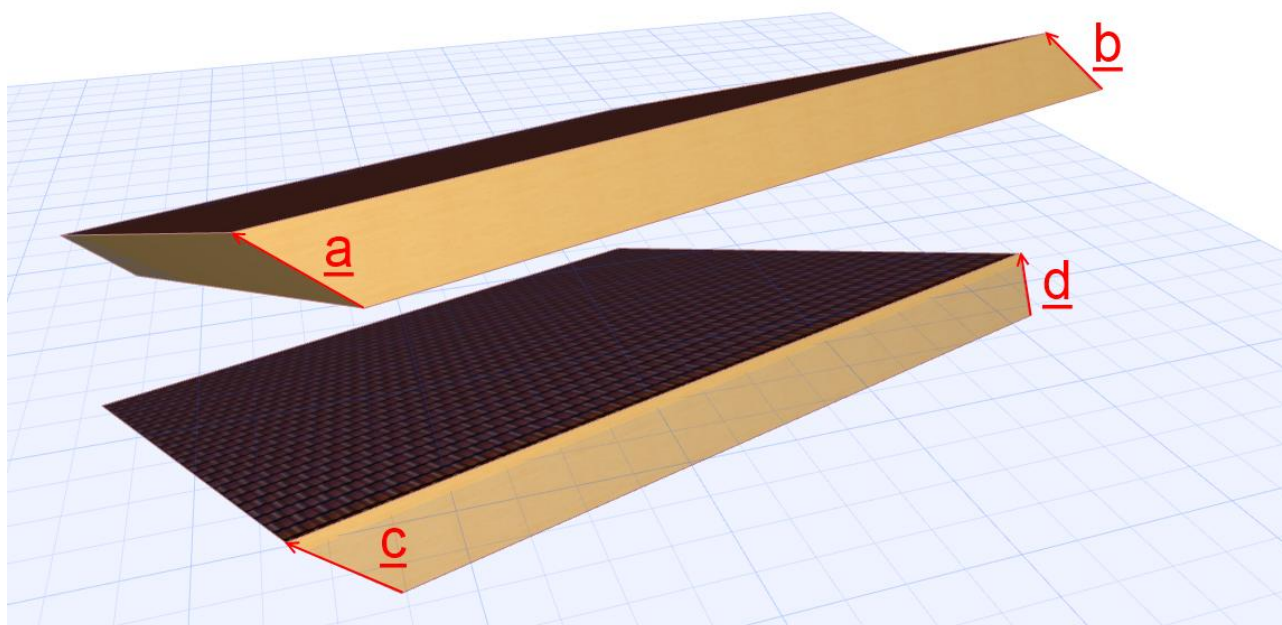
4.2.4. Tető exportálása és importálása

Ezt a feladatot ketten implementáltunk, csapattársam készítette az export folyamatot, én pedig az importálást végeztem. Az exportálás során kétféle módszerrel vihetünk ki tetőt, ebből az egyik az, amikor *roof*-ot teszünk le ArchiCAD-ben a másik pedig *slab* formájában reprezentálódik, a közös mindegyik esetben az, hogy *IfcSlab* objektumként fog megjelenni az **.ifc* fájlban, ami különbség előfordulhat – akár *roof*-nál akár *slab*-nél – az a geometriai reprezentáció. Ugyanis, ha megnézzük a szokásos fájlbeli reprezentációját, amivel eddig is kezdtünk:

```
#287= IFCSLAB('18XWUdc$JFCBj9l_MqJvQ6',#15,'RT - 001',  
$, $, #228, #284,  
'488607A7-9BFB-4F30-BB49-BFE5B44F9686',.ROOF.);
```

Észrevehetjük, hogy a végén van egy jelzés, az utolsó paraméter, ami azt jelzi, hogy ez a *slab* tetőként van értelmezve, ha ezt megváltoztatnánk, akkor ArchiCAD-ben nem *roof*-ként jönne létre. Viszont van egy másik paraméter, ami be van linkelve a *slab*-hez (esetünkben ez a #284-es paraméter) ez pedig az *IfcProductDefinitionShape* objektum, amely meghatározza bármely *IfcProduct* (és annak leszármazottainak) háromdimenziós formáját. Ez lehet *IfcExtrudedAreaSolid* vagy *IfcFacetedBrep* is. Az hogy melyik formában reprezentálódik több tényezőtől is függ, hiszen alapvetően mit tudunk megjeleníteni *extrusion* segítségével? Minden olyan geometriát, amivel egy irányvektor segítségével egy alap poligont kinyújtva megkapjuk a teljes testet. A következő 12. ábra

teljesen jó példa erre, hiszen a felső tetőhöz meghatározható egy *extrusion* irány, amit **a** és **b** vektor is jelöl. Ezzel ellentétesen a lenti tető már *BREP*-ként megy ki, mivel nincsen konkrét *extrusion* irány, hiszen **c** és **d** vektor nem párhuzamos.

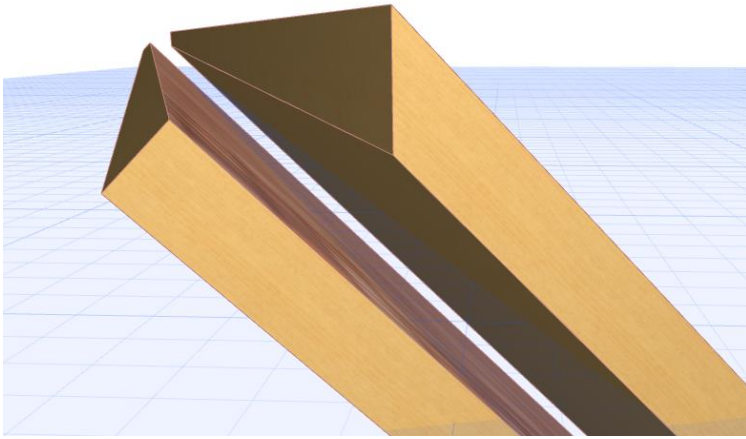


12. ábra Az *IfcExtrudedAreaSolid* és az *IfcFacetedBrep* közötti különbség

Az export folyamat során lekérjük a geometriáját az ArchiCAD-től és megvizsgáljuk, milyen geometriaként menjen ki. Majd ezt követően beállítjuk az egyéb paramétereket is, ami színezés, illetve a készült anyag szempontjából fontos. Külön van kezelve ArchiCAD-ben a *roof* aljának, közepének és tetejének a textúrája, sajnos a jelenlegi *slab* export szerint csak egyféle textúrát csatolhatunk hozzá a tetőhöz, jelenleg ez a legfelső textúra, mivel felhasználói szempontból ez a legfontosabb, hogy a ház tetején megfelelő mintázata legyen a tetőnek, a közepéhez és az aljához képest. Ennek az IFC szerinti megjelenítése az *IfcColourRgb*, amelynek nevéből könnyen rájöhethetünk, hogy a red-green-blue színskálának a keverésével állíthatjuk elő a tetőnek a textúráját. A másik, amit még ki kell menteni, az a tető anyaga, miből készül, ezt pedig a korábban említett *IfcMaterialLayerSetUsage* segítségével adhatjuk meg.

Az export folyamat során igyekeztünk egy szabályhoz ragaszkodni végig, ez a szabály pedig az *extrusion* geometriával történő mentés. Az ok egyszerű, hiszen extrúzióval jóval egyszerűbben meg lehet adni háromdimenziós testeket és a statikai, elemző programok is jobban használják ezt a fajta geometriát. Egyetlen egy poligonra van szükségünk, egy irányra, az eltolás konkrét értékére és meg is van a testünk. Így az export során az alábbi *roof*-okat mentjük extrúzióval:

- Az ArchiCAD-ben „ROOF.” előre definiált típussal (predefined type-ként használják a szakma szerint) ellátott *slab*-eket, ahol a korábban már részletezett módon, meg lehet határozni extrúziós irányt.
- Természetesen tetők (ArchiCAD szóhasználata szerint *roof*-ot) mentése is így történik, ha konkrét extrúziós irány is meghatározható.



13. ábra Az oldallapok oldalszögeinek bemutatása

Sajnos ezen kevés eset áll csak rendelkezésre ahhoz, hogy extrúzióval menthetőek legyenek a tetők. Ezek természetesen tartalmazhatnak vágásokat is, amik szintén extrúzióval kerülnek mentésre. Azonban minden más egyéb esetben *BREP* geometriával mennek ki a tetők, hiszen gondoljunk csak bele, ha a tetőnek a felső síkja nagyobb, mint az alsó, ami a tető oldallapjainak beállított dőlés szögnek köszönhető. A 13. ábrán két eset látható a bal oldali esetén mindegyik oldallap derékszöget zár be a tetőnek a felső

illetve alsó geometriájával is, míg a jobb oldali esetén ez a szög eltér (ezt a szög később *sideAngle*-ként fogjuk emlegetni). Illetve, ha visszaemlékszünk az első fejezetben bemutatott épületre, ott olyan tetők is előfordultak, amik íveket, illetve nem csak egy síkú tetőket ábrázoltak. Ezek már olyan bonyolult geometriával rendelkeznek, hogy nem is lehetne őket másként menteni. A *translator* beállításoknál, azonban szeretném felhívni arra a figyelmet, hogy nem mindegy milyen módszerrel mentünk ki *slab*-eket, *roof*-okat az ArchiCAD-ből. Ugyanis, nem megfelelő beállítások esetén az oldallapok felső, illetve alsó síkkal bezárt szögei nem mentődnek ki, ezért a projekt keretein belül végeztünk az alapértelmezett *translator* beállításain módosítást, ennek köszönhetően később az ArchiCAD 17-ben ez már zökkenőmentesen fog történni.

Azonban azoknak, akik szeretnék ezt az export folyamatot is maguk kipróbálni illetve jobban megérteni mi is történik a háttérben, leírjuk, hogyan lehetséges *sideAngle* mentése megfelelően. Amint elindítottuk az ArchiCAD-et és letettünk egy *roof*-ot *Ctrl + T* segítségével hozzuk be a beállításait. Ezt követve látni fogunk egy több részből felépülő dialógus ablakot, ahol az adott modellre vonatkozó információkat találhatunk. Nekünk egyetlen egy részre lesz szükségünk, az pedig a 'Model' résznél található 'Edge Angle' amit, ha átállítunk 'Vertical'-ra, akkor a tetőnek az alap poligonját, amiből indul az extrúzió (a fenti ábrán ez mindkét tető esetén a tető alját jelöli) a globális Z irányt fogja felvenni, és ennek megfelelően állítja be a *sideAngle*-t is. Ezt erősen befolyásolja, az hogy hány fokos tetővel van dolgunk. Ez után a pár kattintás után nyugodtan menjünk a fájl/mentés másként menüpontra, de még mielőtt mentenénk, válasszuk ki az *.ifc fájl formátumot és klikkeljünk a „Beállítások...” feliratú gombra. Itt fogjuk elvégezni azt a módosítást, hogy bármilyen *sideAngle* mentést sikeresen ki tudjunk vinni a megfelelő *translator* beállítás kihasználásával. Az export opciók között találni fogunk a „Komplex elemek”-re vonatkozó kritériumokat, nyissuk meg ezt a dialógus ablakot. Ezután a felsorolás legalján megtaláljuk a *slab* exportra vonatkozó opciót, jól látható kétféle export között választhatunk, amikről ennek a fejezetnek a keretein belül is szó esett: *BREP* és *extrusion*.

Tetők exportálása esetén nem csak magáról a tetőről kellett gondoskodni, hanem a hozzá csatolható ablakokról (*skylight*), viszont az import követelmények esetén nem került implementálásra a megfelelő *skylight*-ok beolvasása, azokból – a már korábban említett – *objectek* generálódtak. Visszatérve az exportra, *skylight* esetén figyelni kellett arra, hogy más egyéb tényezők

is befolyásolhatják, hogy mi kerül egy fájlba mentésre. A kétdimenziós nézet esetén, mindig az aktuálisan látható szinten szereplő összes elemet menteni fogjuk, ha kijelölünk valamit, akkor csak azt, de külön a mentésnél is beállíthatjuk ezeket a paramétereket. Továbbá, ha átváltanánk a nézetet háromdimenzióra, akkor több opció válik elérhetővé, amivel befolyásolhatjuk a mentést is. Mivel a *skylight* láthatóságát itt lehet befolyásolni⁹, ha kikapcsoljuk a megjelenítését, akkor egy üres vágás marad, lyukas lesz a tető. Ekkor logikusan háromdimenzióból mentve csak a lyukat kell kivinni a tetőt semmiféleképpen, azonban, ha már kijelöljük egy kattintással is a *roof*-ot, akkor – ha látszik, ha nem – a rajta lévő *skylight*-ot is exportáljuk. Továbbá, ha látható a *skylight* és ki van jelölve a *roof*, akkor is mentésre kerül a *skylight*, geometriával együtt.

Exportálás során lyukakat is ki kell vinni, ezeket – így mint a *skylight*-okat is – *IfcOpeningElement* formájában mentjük ki, ahol az asszociációs kapcsolatot az *IfcRelVoidsElement* biztosítja. Így egy *slab*-hez végtelen sok opening is tartozhat. Az egyszerű vágások a tetőben, bármilyen hihetetlen sokkal bonyolultabbak is lehetnek, mint egy *skylight*, hiszen ezekre a vágásokra is meghatározható *sideAngle*, amiknek az alkalmazása esetén már szinte biztos, hogy BREP geometriaként fognak mentődni megfelelően a beállításaink. Jó, ha észben tartjuk, hogy a *sideAngle* paramétere csak 0° és 180° fok közötti értékeket vehet fel a határokat nem beleértve.

Röviden összefoglalva az export folyamat megfelelő végrehajtásához szükségünk van a geometria pontos exportálására, ez az *IfcLocalPlacement*-ek összeállítását igényli, *IfcOpeningElement*-ek formájában visszükk ki a *skylight*-okat és a vágott *opening*-eket is. A *sideAngle*-k mentése pedig bele van égetve a geometriába, ami ugye vagy *extrusion*, vagy *BREP*. Illetve szó esett még az *IfcMaterialLayerSetUsage*-ról és a tető színezéséről, az *IfcColourRgb* kapcsán.

Az import folyamat bemutatása előtt fontosnak tartom bemutatni, hogy a beimportált *.ifc fájlban található összes objektum egy zárt rétegre kerül rá. Ezt természetesen feloldhatjuk¹⁰ és módosításokat is végezhetünk a tetővel. Viszont, ha szeretnénk elkerülni az import során ez az automatikus zárt rétegre történő beolvasást, akkor a *translator* beállításokat hívhatjuk segítségül. Régebben *translator*-ról már volt szó – a *sideAngle*-k kapcsán – hogyan is lehet behozni azokat, ezt tegyük meg most is. Viszont ne az export, hanem az import beállításokat vizsgáljuk meg, először duplikáljuk az *default translator*-t és állítsuk be alapértelmezettnek az importáláshoz. Ezt a jobb felső sarokban látható balra mutató ikonra/gombra klikkelve tehetjük meg, majd ezt követve oldjuk fel át az importálásra vonatkozó lakatot, így már nincs szükség manuálisan a rétegek feloldására, automatikusan történik.

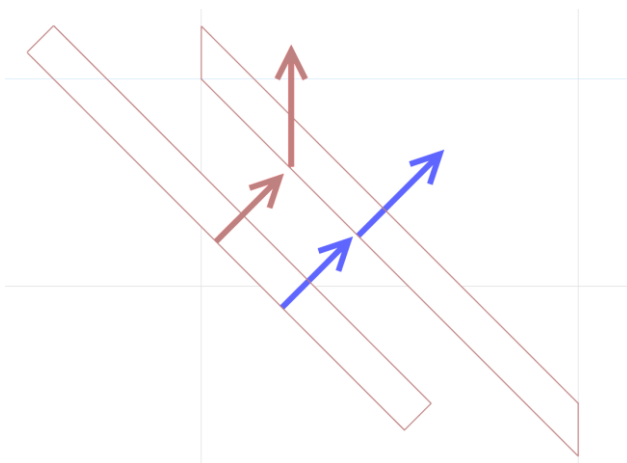
Mivel ArchiCAD-ben már *slab*-eket eddig is olvastunk, ezért az oda érkező *IfcSlab*-eket kellett kezelnünk a *slab* normálvektorának megfelelően, ugyanis, ha az párhuzamos a globális Z tengellyel, akkor nem *roof*, hanem *slab* lesz belőle. Ez alól egy kivétel van, mégpedig a – már korábban említett – *predefined type*, ami ha „ROOF.”, akkor tetőként kerül beolvasásra. Illetve, olyan tető

⁹ *Ctrl + Alt + A* segítségével pipálhatjuk ki és be mit kívánunk megjeleníteni a háromdimenziós nézetben.

¹⁰ *Ctrl + L* billentyű kombinációt lenyomva megjelenik egy dialógus ablak, amiben az importálás elvégezve találni fogunk egy piros lakattal ellátott réteget, ez „kilakatozva” módosíthatunk minden olyan elemet, ami az adott réteghez tartozik.

nem jöhet létre, aminek a normálvektora derékszöget zár be a globális Z tengellyel, ezekből *object* generálódik, mivel másként nem tudjuk kezelni.

Ahhoz, hogy bármit is tudjuk kezdeni a geometria megfelelő elhelyezésére, nyilvánvaló, hogy kell egy pontosan, precízen meghatározott normálvektor. Ehhez több paramétert is felhasználunk, a legfontosabbak közé tartoznak maga az alap poligon, amit extrudálunk. Logikusnak tűnhet, hogy miért nem az *extrusion* irányát használjuk fel normálvektornak? Az ok egyszerű, nem mindig értelmezhető ez az irány normálvektornak, erre tökéletes példa a 14. ábra, ahol a bal oldali *roof* esetén valóban normál vektor az *extrusion* iránya, míg a másikinál közel sem az, hiszen a globális Z irányról beszélünk, ami nem merőleges az alappoligonra, amely ki van nyújtva.



14. ábra Egy tető esetén az extrusion iránya és a normálvektor közötti különbség

A konkrét számításhoz szükségünk van egy *boundingBox* nevezetű háromdimenziós objektumra. Gyakorlatilag minden egyes objektumra meghatározható egy ilyen *boundingBox*. Képzeld el az Egyiptomban található Gizai piramisokat, közülük egyet, ha bele kéne helyezni egy hatalmas dobozba, akkor a piramis felső csúcsa érintené a doboz tetejét, míg az alja pont belepasszolna a doboz legaljára. Elméletben így lehet meghatározni egy ilyen *boundingBox*-ot. Már biztos mindenkit furdal a kíváncsiság mire is lesz ez jó nekünk? Hát ahhoz, hogy egy síknak meghatározzuk a normálvektorát¹¹, amelyben egy konkáv poligon is

elhelyezkedhet. Igaz, hogy egy síkban elhelyezkedő poligonról van szó, ahol a pontok nem egy piramist határolnak be, de ez nem probléma. Azért nem, mert egy piramisnak egy oldalára is meg tudjuk határozni a *boundingBox*-ot, ami már egy poligon és lényegesen egyszerűbb formát is ölt. A *boundingBox*-ot tipikusan kettő koordinátával lehet megadni (illetve többel is, de az már felesleges), a legkisebb koordinátája a minimuma és a legnagyobb koordinátája a maximuma. Ezt a poligon és a piramis esetén is a pontok végigjárásával kaphatjuk meg, ahol kisebb, vagy nagyobb x, y, z koordináták vannak, azokkal folyamatosan tágitjuk a *boundingBox*-ot, míg végül megkapjuk a piramis/poligon dobozát.

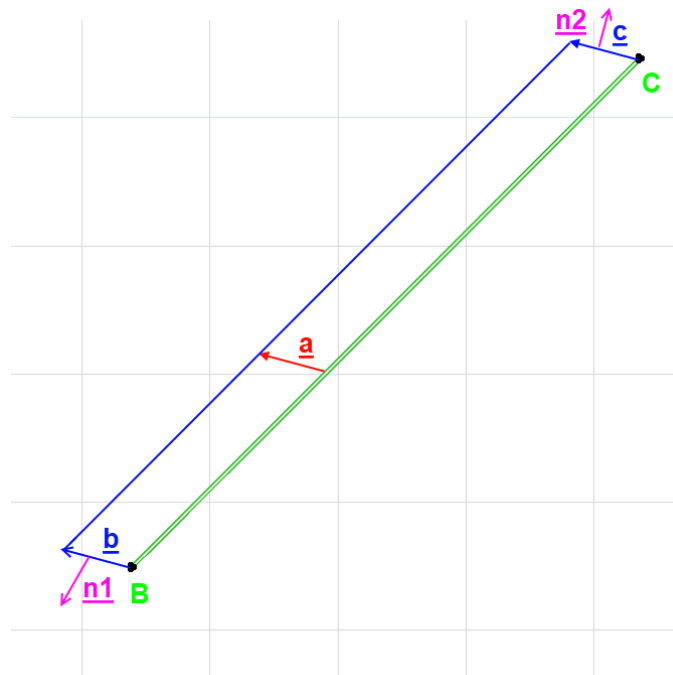
Ezek után végig iterálunk a pontokon, és ha közülük egy is ráesik a „doboz” bármelyik oldalára, akkor abból már meghatározhatunk egy normálvektort. Viszont amire itt nagyon figyelni kell, az az, hogy ebből a pontból, amit húzunk vektort az előtte lévő és utána lévő pontba, nem szabad, hogy kollineárisak, párhuzamosak legyenek, hiszen erre végtelen sok normálvektort lehet képezni. Ilyenkor keresni kell egy másik pontot, amire nem teljesül a kollinearitás és meg is kapjuk a megfelelő normálvektort.

¹¹ A normálvektor egy síkra van értelmezve, képzeljünk el egy négyzetet térben. Ezen a négyzeten minden pont egy síkra esik, és meghatározható egy rá merőleges vektor, ami derékszöget zár be azzal a síkkal. Ebből lehet számolni dőlésszöget, a négyzet oldalainak irányát meg lehet belőle határozni. A lényeg az, hogy bármit is számolni akarunk egy poligonnal szükségünk lesz annak normálvektorára.

A normál vektor meghatározása után egyszerűen ki tudjuk számítani a tetőnek a szögét, hiszen ha azt levetítjük az XY síkra, akkor fel tudjuk mérni a kettő között bezárt szöget, majd azt kivonva π -nek a feléből, megkapjuk a tényleges szöget. Az import során csak olyan tetőket fogunk beimportálni, amik 0° és 89° közé esnek, habár engedélyezve van -89° -ig is ArchiCAD-ben, de mivel a normálvektor mindig a koordináta rendszer felső két negyedébe fog esni, ezért nincs szükség negatív szögek előállítására.

ArchiCAD-ben minden tetőhöz tartozik egy pivot/referencia vonal, ami a tetővel kell, hogy egy síkba essen, ezt a vonalat a tető létrehozásakor le kell tenni. Ez a legelső lépés egy tető létrehozása esetén. Tartozik hozzá egy kezdő és egy végpont, ez határoz meg egy vektort. Ennek a számolása a normálvektor segítségével történik, ugyanis – a szögszámításhoz hasonlóan – levetítjük a normálvektor Z komponensét, tehát egyenlővé tesszük nullával, majd ezt követve elforgatjuk derékszöggel felfelé, hogy megkapjuk az irányát a referencia vonalnak. Ebből a vektorból egységvektort csinálunk, mivel a normál vektor is egységnyi hosszú. A következő lépésként, hogy ne egy ilyen rövid referencia vonalat kapjunk meg folyamatosan tágitani fogjuk ezt a vektort. Mivel a lokális koordinátarendszerben ez mindig az X tengelyen való mozgást jelenti, ezért megkeressük a legkisebb és legnagyobb X koordinátával rendelkező pontot, majd ennek megfelelően megnöveljük a referencia vonal kezdő és végpontját. Így jön létre a referencia vonal, ami meghatározza, hogy pontosan honnét számít a hajlásszög.

A referencia vonal létrehozása után foglalkozunk a *sideAngle* megfelelő beállításával, amire viszont felhívnam a figyelmet az a *BREP*-ről *Extrusion*-re történő konverzió. Ugyanis ez a konvertálás, már korábban lezajlik, de erről később bővebben is lesz majd szó. Esetünkben nyilván *Extrusion* geometriához állítjuk be a *sideAngle*-t az *extrusion* irányának megfelelően. Ahol az oldallapokat az *extrusion* irányának megfelelően előállítjuk a *roof* profiljából. Amint az a következő 15. ábra esetén is jól látszik.



15. ábra Az extrusion eltolása következtében létrehozott egyik oldallap tető esetén

A zöld vonal jelöli a *roof* profilját, ami oldalnézetben van ábrázolva, így látható az ábrán zölddel jelölt **B** és **C** csúcspont is. Az **a** vektor az extrusion iránya, a két kék színnel jelölt vektor – **b** és **c** – adja meg a pontok eltolását, így áll elő az egyik oldallap, ami az ábrán részben kékkel van rajzolva. A számítás során felhasználjuk a profil normálvektorát, ami merőleges a **BC** vektorra. Ez a normálvektor az **a** vektorral bezár egy szöget, amely meghatározza mekkora szöget kell beállítani az oldallapok esetén. Az ábrán látható kék vektorok bezárnak egy szöget a tetőnek a felső lapjával, ezt a szöget a **b** és **c** vektor mögött elhelyezkedő oldallap normálvektorával tudjuk kiszámítani. Pontosan mekkora szögről is van szó? Ezt az oldallapok normálvektorainak (**n1** és **n2** normálvektorokról van szó) és a profil normálvektorának skaláris szorzata adja, mivel negatív skalár szorzat esetén ki kell vonni az alapértelmezett 90°-ból, ellenkező esetben pedig hozzáadni.

Röviden tömören az alábbi adatok segítségével be lehet importálni és létre lehet hozni megfelelő kettő- és háromdimenziós *roof*-ot. Aminek a bemutatására ígéretet tettem az a *BREP* és *Extrusion* közötti konverzió. Ez a konverzió nem csak *roof* esetén fut le, hanem *slab* importálása esetén is, feltéve ha *BREP* geometriát importálunk. Ez esetben szinte bármilyen geometriai alak előfordulhat, ami lehet nem is ábrázolható se *slab*-ként, se *roof*-ként. A piramis például egy pontosan ilyen forma, – a fenti példánál maradva – azért mert nem határozható meg kettő olyan oldal, amelyek síkjának a normálvektora párhuzamos lenne egymással. Az ilyen objektumokból *object* lesz, amelyek viszont megfelelő geometriával rendelkeznek az vagy *slab* vagy *roof* formájában importálódhatnak. A megfelelő geometria alatt az alábbiértjük:

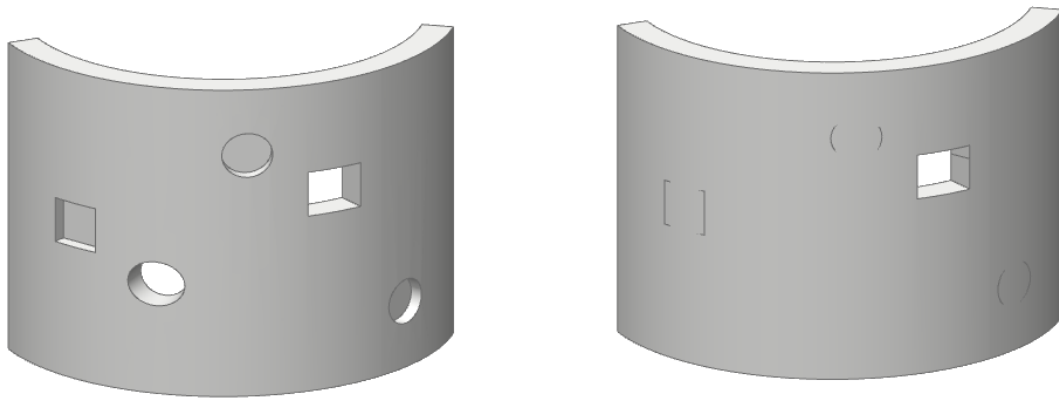
- Meghatározható két olyan oldal, amelyek esetén az oldalhoz tartozó síkok normálvektorai párhuzamosak.
- Ezen két oldal közül külön-külön az oldalak pontjainak számát nézve meg kell, hogy egyezzen az egész test pontjainak számának felével. Adott például egy téglatest alakú *roof* vagy *slab*. Ebben az esetben a téglatest legnagyobb kerülettel rendelkező oldala lesz a profil/alappolygon (esetünkben kettő ilyen oldal van, bármelyik lehet profil). Ennek az alappolygonnak a pontjainak a száma négy, míg a teljes téglatest pontjainak száma nyolc, így ez a forma tökéletesen reprezentálható *roof* vagy *slab* típusként ArchiCAD-ben.

A konverzió során szintén számolni kell a *sideAngle*-ket, az *extrusion* geometria esetén számított módon. A konverzió esetén szinte mindig negatív *Z* az *extrusion* iránya. Ez azért van mert egy *slab*-et mindig lefele extrudál az ArchiCAD, míg a tetőt felfelé, ezért meg kell fordítani az *extrusion* irányát, úgy, hogy a profilt lefelé eltoljuk és az *extrusion* irányát megfordítjuk.

Végül a *roof* export és import folyamatot lezárva kijelenthetjük, hogy az exportálás során teljesítettünk minden olyan fontos kritériumot, ami minden forma mentését lehetővé teszi, bármilyen nyílással/*opening*-gel ellátva. Itt ne csak az egy síkban elhelyezkedő tetőkre gondoljuk, hanem íves több oldalból álló tetőre is tökéletesen működik az exportálás. Az import folyamat során csak az egy síkban elhelyezkedő úgynevezett *planeRoof*-okat tudjuk beolvasni, mivel ez fért bele a fejlesztési ciklusba.

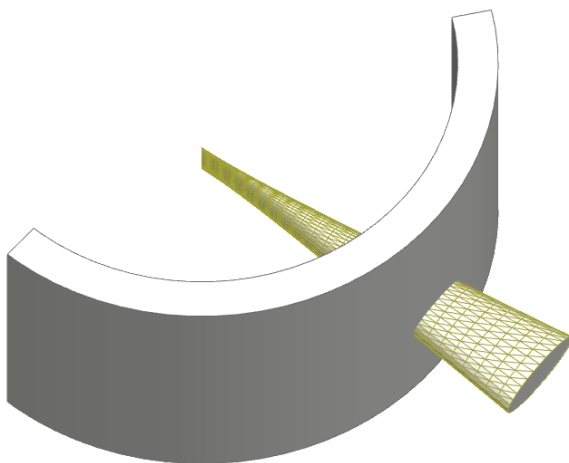
4.2.5. Opening vágások megfelelő exportálása

Ezeket a vágásokat íves falban helyezték el kör és téglalap alakot öltve, a következő 16. ábrán a bal oldali íves falon még jól láthatóak. Azonban, a jobb oldali már a beimportált vágásokat jelöli, amelyeket az IFC Add-on hozott létre. Az ArchiCAD export folyamat előtt ezek a vágások, nyílások még *window*/ablakként vannak jelen, a beimportálás után – mivel *empty opening*-ek – SEO (Solid Element Operations) műveletekként jelennek meg, amelyek rosszul vannak paraméterezve, ahogy a jobb oldali íves falon ez jól látszik is. Ezek a SEO műveletek az ArchiCAD-ben gyakorlatilag CSG műveletek, amelyek korábban már be lettek mutatva.



16. ábra Az *IfcOpeningElement* export hibájának bemutatása

Két alapvető problémát kell megoldani ennek a hibának a kapcsán. Látható, hogy egy kör alakú teljes vágás teljesen eltűnt az importálás során, illetve a mélyedések esetén egy vékony réteg elfedi a falba vágott kellően mély vágásokat. Ennek a vékony rétegnek a megszüntetését a csapattársam végezte, míg én a teljes mértékben eltűnt vágásra találtam egy megoldást. A következőkben elsősorban a saját megoldásomat mutatom be első sorban és a másik – nem egyszerű – problémára most nem térünk ki.



17. ábra A vágás geometriai formája

Tulajdonképpen, amikor beolvassuk ezeket a vágásokat, azok egy külön rétegre kerülnek elrejtésre az ArchiCAD-ben, így, ennek az egy elrejtett vágásnak a geometriáját is meg tudjuk tekinteni a *Ctrl + L*, *Ctrl + A* és „*Show all*” feliratú gombra klikkelve láthatóvá válik minden, ami el volt rejtve, így a vágások is. A következő 17. ábrán látható egy vágótest, ami számunkra fontos. Ez a test háromszögekből épül fel illetve egy körből. Az IFC szabványbeli reprezentálása *IfcFacetedBrep* mivel másféle módon lehetetlen ábrázolni ezt a testet. Az export, de leginkább az import folyamat során alkalmazunk egy algoritmust, ami kiszűri az

egymáshoz közeli pontokat. Erre az algoritmusra azért van szükség, hogy zárt testeket generáljunk, ugyanis a Revit ment olyan *brep* geometriát, ami nem alkot zárt testet, hanem két pont adott ϵ távolságra helyezkedik el egymástól. Ez a távolság nagyon kicsi, mikro és nanométer értékek közötti értékről van szó. Hiába vannak ilyen kicsi lyukak egy testben, akkor sem teljesíti a zártság kritériumát, ami a SEO művelet szempontjából rendkívül fontos. Ezeket a pontokat helyettesítjük meglévő pontokkal, majd a már nem indexelt pontokat megszüntetjük, töröljük. A vágás geometriájában a végénél az origónál lévő pontok között fordultak elő ilyen pontok leginkább – ráadásul mivel ez nem is Revitből lett exportálva, hanem ArchiCAD-ből itt a test még zárt is volt, csak közel voltak a pontok egymáshoz – ezeket összevontuk és így keletkezett egy olyan poligon, ami egy nulla területtel rendelkező logikailag, de a lebegőpontos számoknál ez csak megközelítette a nulla értékét. Ez a nulla területű poligon sem megengedett a SEO művelet végrehajtásához.

A pontok ugye már össze lettek vonva és előállt ezekből egy új poligon halmaz, ami már biztos zárt volt, de ez által létre jöhetnek olyan oldalak is, amik nulla területtel rendelkeztek – amint fent már említettük – és ezeknek a szűrésére lett létrehozva korábban egy másik algoritmus, ami az ilyen degenerált oldalakat szűrte ki, a pontok helyettesítése után. Viszont ez az algoritmus normálvektorokat számított ezekre a háromszögekre, poligonokra, ugyanis a normálvektor hosszúsága adja meg azt, hogy megfelelő-e az adott *brep* oldala. Előfordulhat olyan normálvektor – a kicsi ϵ távolság miatt, – aminek a hossza közel helyezkedik el a nullához. Az ilyen közelítéseket az ArchiCAD előre definiált konstansokkal végzi el, külön vannak kicsi, közepes és nagy ϵ értékek felvéve, amivel lényegesen gyorsabban tudjuk hasonlítani, hogy nulla közeli-e az érték. Másrészt pedig lebegőpontos értékek egyenlőséggel történő vizsgálat rendkívül rossz megoldás az esetleges kerekítési hibák miatt is, plusz a processzor terhelését is jobban növeli.

Visszatérve az oldalak normálvektorának ellenőrzésére, a probléma ezzel az ellenőrzéssel egy túl kicsi ϵ érték miatt történt. A 17. ábrán látható geometria esetén a vágás kiindulópontjánál, –ami egyben az íves falnak az origója is – keletkezett egy olyan oldal, ami ugyanazokat a pontokat tartalmazta duplikálva, ami egy nullaterületű poligont eredményezett. Az ϵ érték nagyobbra vételével viszont sikeresen észlelve lett a hiba így már az exportálás során el lett távolítva a degenerált oldal. Ennek köszönhetően már végre tudta hajtani az ArchiCAD motorja a vágást a testen.

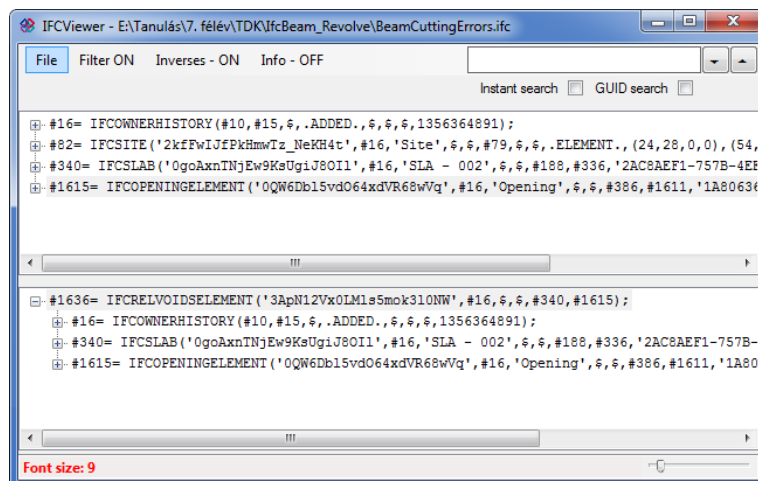
4.2.6. A fődém vágásainak geometriánkénti mentése

Erre a hibára egyik versenytársunk hívta fel a figyelmet, a Tekla, ugyanis Coordination View 1.0 mentési/exportálási opció beállításával rosszul mentjük ki az *IfcSlab*-re elhelyezett geometriákat, mivel – ahogy a 18. ábrán is látszódik – *IfcOpeningElement* mentésére kerül sor, viszont egy ilyen *openingelement*-nek egy geometriát kéne magába foglalnia és jelen esetben egy elem több geometriát ír le. CV 2.0 mentés esetén viszont már megfelelően működik ez a funkció, mert ott *IfcBooleanClippingResult* formájában kerül a fájlban leírásra, amik a végtelenségig láncolhatók egymásba, ahogy azt korábban említettük.

A hiba reprodukálása ArchiCAD 16-ban egy *slab* elhelyezésével és legalább kettő *beam* vagy valamilyen más test elhelyezésével könnyedén előidézhető. A *design* menüben belül találunk egy *Solid Element Operations* menüpontot, amivel vágásokat tudunk elhelyezni a *slab* geometriájában a *beam*-ek segítségével a *subtract downward* művelet kijelölésével. Ezt, ha CV 1.0 *translator*

beállítás mellett kiexportáljuk, akkor Solibri-vel könnyedén megtekinthetjük, ahogy egy *opening* kettő geometriát aggregált magába.

Jobban szemügyre véve az ábrát láthatjuk, hogy a felső részénél négy elem szerepel, ebből az alsó kettő fontos a számunka a *slab* és az *openingelement*, míg a kettő közötti kapcsolatot az ábra alsó felén elhelyezkedő *IfcRelVoidsElement* kapcsolja össze, ami egy az egyhez kapcsolatot képes leírni, esetünkben a földem és a vágás kapcsolatát. A Teklának az alkalmazása viszont egy analízis szempontjából fontos toolhoz használja ezt az elemet, ahol feltétel az, hogy egy *IfcOpeningElement*-hez tényleg csak egy geometria tartozzon. A javítás egy egyszerű ciklus létrehozása, ahol már külön elemként mentjük ki a geometriákat.



18. ábra Az *IfcSlab* és a csatlakozó *IfcOpeningElement* leírása a szabvány szerint

5. AZ ARCHICAD TESZTELÉSE

5.1. Graphisoft Autotester-ről röviden

Az ArchiCAD-nek a tesztelése egy belső fejlesztésű keretrendszerrel történik, amit teljes nevén Graphisoft Autotesternek hívunk, ez rövidítve GSAT. Céges fejlesztőgárda végzi az implementálását, ami Pearl nyelven zajlik és a fejlesztés OSX és Windows7 operációs rendszerre történik, mint az ArchiCAD is. A GSAT-ot magát a tesztmérnökök használják tesztek készítésére, írására, kiértékelésére, memória szivárgás detektálására. Kereskedelmi forgalomba nem kapható, licenz díjat nem igényel, mivel mi fejlesztjük saját magunknak. A tesztelés menete [8] Ian Sommerville gondolatmenetére építve két részre szedhető, Whitebox és Blackbox tesztelésre, ennek cégen belüli reprezentálása a következő fejezetekben kerül bemutatásra.

5.2. Whitebox tesztelés

A tesztelés során egyedül nekünk fejlesztőknek van lehetőség olyan tesztesetek létrehozására, ami kód közeli / whitebox tesztek futtatására ad lehetőséget. Egy ilyen tesztfájl létrehozása az implementáció elkészülte közben folyamatosan növekszik az új feature hibátlan működése végett. Korábban implementált *roof* import taszk esetén is kreáltam egy ilyen fájlt, ami később majd bekerül egy új teszt formájában a GSAT-ba. Ebben az esetben a fejlesztés gyakorlatilag megköveteli a precíz munkavégzés végett a megfelelő tesztfájlok létrehozását.

Szerencsére nem csak tesztfájlokat hozhatunk létre mi magunk, hanem kódból is programozhatunk egység/unit teszteket. Ezt úgy képzeljük el, hogy az ArchiCAD-ben rendelkezésre álló menüpontok közé felveszünk egy új menüelemet, és gyakorlatilag egy klikkelésre lefut a teszt, ami létrehozza az adott objektumokat és kiértékeli azokat, majd jelzi, hogy sikeresen lefutott-e a unit teszt. Ez a funkció a kiadott ArchiCAD-ben nem érhető el, teszt mérnökök használják.

Az adott taszk elkészülte után kettő tesztelési fázison kell átesnie a cégen belül, ami a verifikáláshoz (verify) szorosan kötődik. Az első nem is a konkrét teszteléshez kötődik, ez is inkább whitebox teszteléshez fűződik, amikor egy csapattárrsal, de leginkább a projekt menedzserrel egy közös kódanalízis (code review) során átnézzük a kódot és megvizsgálunk minden elméleti esetet, hol lehet probléma a kóddal kapcsolatban. Elvégezzük az utolsó módosításokat, közben folyamatosan ellenőrizve a teljes tesztkészlettel, amit mi magunk hoztunk létre.

5.3. Blackbox tesztelés

A következő lépésben, már csak a projekt menedzsere, illetve tesztmérnöke verifikálhatja az adott taszkot a kód megtekintését mellőzve, ekkor az implementáció már beletett másolva a projektbe és a projekt tesztmérnökei is, programozói is előtudják idézni annak működését. Ha a projekt menedzser/tesztmérnök úgy ítéli meg, hogy helyesen hajt végre minden műveletet az adott funkció, akkor a verifikálás sikeres, ellenkező esetben visszakerül a fejlesztőhöz javításra az adott taszk. A validálást pedig egy külső vállalat végzi (pl.: aki kérte a funkciót), esetünkben a GSA.

A tesztmérnökök munkája viszont nem csak a verifikálásokban merül ki, hanem a tesztek létrehozásában, megújításában a GSAT-on keresztül. Ez a tesztelő rendszer gyakorlatilag zökkenőmentesen biztosítja a tesztesetek létrehozását, mivel egérrel végigklikkelve létre lehet hozni az adott esetet, amit tesztelni akarunk. Ezt követően vagy az adott projekt esetén hozzuk létre a tesztet, vagy pedig a kibocsátásra készülő ArchiCAD mellé, ami a főág. A 6. fejezetben erre majd részletesebben is kitérünk mi is az a főág.

A blackbox tesztelés szerves részét képezi a projekt ágak és a napi automatikus tesztek futása. Ezen projekt keretein belül is többször időpontot kellett kérni a projekt tesztelésére, ekkor többféle teszt „csomag” is lefut a saját projektben fejlesztett ArchiCAD-re. Ezek a csomagok más-más funkciókat tesztelnek. Valamelyik csak egyszerű funkciókat, de sebesség mérés is történik, amit mi programozók állítunk össze, ugyanis a kódba mi helyezzük el a mérési pontokat, ez pedig folyamatosan naplózásra kerül. Ezek a sebesség mérések szorosan kötődnek a **Tinderbox**-hoz, amiről a 6. fejezetben olvashatunk részletesebben.

5.4. Tesztek kiértékelésének különféle típusai

A tesztek kiértékelése is komplex feladat lehet. A sebesség mérése a **Tinderbox**-nak köszönhetően egyszerű, hiszen nem várt lassulás esetén már lehet tudni, hogy hiba van. Ezzel ellentétben egy kiexportált *.ifc fájl tartalmában hibát találni, mert nem fogadja el a szabvány, az elég nehéz, de ez kevésbé tartozik a tesztmérnökök feladatkörébe. Amit szem előtt kell tartani, az export, import során bekövetkezett adatvesztés előfordulása, geometriailag rossz megjelenítés. Az automatikus tesztek lefutása után ki kell értékelni a keletkezett adatokat, esetünkben *.ifc fájlok megtekintéséről van szó. Export hibákat tipikusan a Solibri Model View-er segítségével vehetők észre, import hibák pedig csak ArchiCAD-be importálással ellenőrizhetők. Ennek érdekében a legtöbb fejlesztő és tesztelő rendelkezik több külső programmal is, amelynek segítségével tesztelni lehet más alkalmazás által generált *.ifc fájlokat.

Már történt említés az assertek-ről, amelyek a fejlesztők munkáját segítik, ezek az assertek tesztelőknél nem jelentkeznek felugró ablakok formájában, hanem csak a naplófájlokban találhatók meg, rendesen kinaplózva, a konkrét okkal, callstack-kel ellátva, hogy a fejlesztő már abból is leszűrje hol történt a hiba. A memóriaszivárgás hasonlóan ilyen probléma, ezt az ArchiCAD bezárásakor mérjük fel automatikusan, amelyik stackben nem történt memória felszabadítás az is naplózásra kerül, hogy hol, melyik metódusban történt a szivárgás mekkora adatmennyiségről van szó.

6. FEJLESZTŐI ESZKÖZÖK

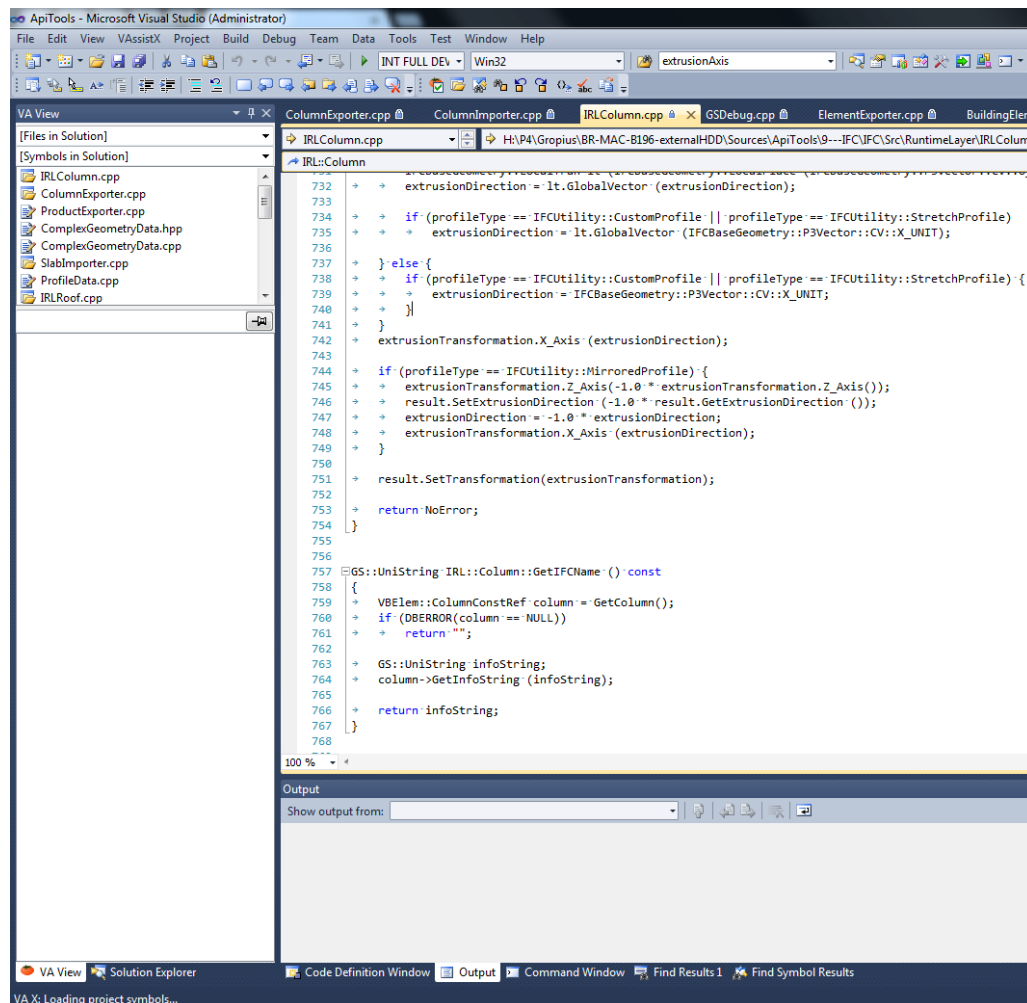
6.1. Felhasznált szoftverek

A munkahelyi gyakorlatom ideje alatt átfogóan megismertem a feladataim megoldásához használt szoftvereket. Illetve a feladatoknak a megoldása közben ismerkedtem meg a Graphisoft által használt új alkalmazásokkal és a belső intranettel. Az alkalmazások, amelyeket használunk az alábbiak:

- Visual Studio 2010, Visual Assist
- TestTrack
- Araxis Merge
- Perforce
- Solibri, IFCViewer
- Skype, Outlook, GoToMeeting

Röviden a felhasznált szoftverekről, Visual Studio 2010-ben (továbbiakban VS2010) programozunk Windows 7 alatt¹², ehhez kapcsolódik a Visual Assist, ami hasonló a VS2010-ben használt IntelliSense-hez. A különbség a kettő között az, hogy az ArchiCAD forráskód állománya, még részekre bontva is olyan hatalmas, hogy az IntelliSense számára túl nagy lenne, rengeteg időt elvenne, míg betölti az ArchiCAD adott forráskód részét és kevésbé illeszkedik rá a cégben használatos kódolási stílusára. Ezért Visual Assistot használunk a hatékony fejlesztéshez, melynél könnyedén beállítható, hogy milyen kódot generáljon például akkor, ha egy C++ metódust kifejtünk az osztály definíción kívül. A VS2010-hez fűződő fontos funkciók egy fájlban a módosításához szükséges gyorsbillentyűk, mivel a verziókezelő szoftver írásvédetten kezeli azokat a fájlokat, amik nincsenek kijelölve szerkesztésre. Az Visual Assist segítségével megkereshetjük az adott fájlt, amit szerkeszteni akarunk és a beállított gyorsbillentyűvel szerkeszthetjük is a fájlkunkat. Az Assist további nagy lehetősége még, hogy bármilyen névteret, osztályt, metódust – azaz egyszóval szimbólumot – képes megkeresni a benyitott projektben. A 19. ábrán látható a VS2010 és – beleintegrálva – a Visual Assist, a bal oldali részén helyezkedik el a Visual Assist a VS2010-ben, a korábban felsorolt funkciók közül a szimbólum keresés és a fájlok tartalmának azonnali megjelenítésére vonatkozó szöveges mezők láthatóak, míg globálisan a VS2010-et tekintve elérhető funkció még az adott forrásfájl és fejfájlja közötti váltogatás lehetősége, azaz *.cpp és *.hpp fájlok közötti váltás.

¹² iMAC-en OSX-ben is van lehetőség fejleszteni, de ott xCode-ot használnánk, kevésbé használható hatékonyan VS2010-hez képest. OSX-et általában a Teszt mérnökök (Quality Assurance Engineer) szoktak használni.



19. ábra A Visual Studio 2010 és a Visual Assist

TestTrack-ben vannak nyilvántartva az ArchiCAD-hez kapcsolódó teendők, amelyek két részre bonthatók. Hibákra (Defects, Errors) és feladatokra (Tasks), ami közös bennük, hogy tökéletesen végig lehet követni melyik verzióban mikor lett kész az adott funkció, kinél volt az adott hiba, ki javította, ki ellenőrizte a tesztelésnél. A 20. ábrán látható a TestTrackról egy képrészlet, az alul szürkével és bal oldalt pipával ellátott hibák/taszkok le lettek ellenőrizve, és átmentek a tesztelésen, míg amelyek nincsenek kiszűrítve, azok még javításra várnak. Amikor egy javítás, funkció készen van, akkor másolható be az adott ArchiCAD verzióba¹³. Ezeknek a javításoknak, funkcióknak a helyes működését általában a projekt menedzsere ellenőrzi le egy közös code review során. A TestTrack segítségével könnyedén szűrhetők a benne szereplő hibák, ezért is hoztak létre több szűrőt, aminek előnyeit pár klikkeléssel könnyedén élvezhetjük. Ilyen szűrők segítségével megtalálhatjuk az adott verzióra vonatkozó kritikus hibákat, projekt ági hibákat, taszkokat, mely hiba/taszk vár tesztelésre, verifikálásra.

¹³ Jelenleg az ArchiCAD 17 fejlesztése zajlik, viszont ha egy hibát valamelyik előző verzióban kijavítottunk, akkor az utána következő összes verzióba is bele kell tenni a javítást.

Number	Summary	Type	Target
143936	B-196-M2: IfcOpeningElement export in case of recess, curved wall and slab with CSG geometry	Task	
143003	B-196: Source review/testing feladatok	Task	
143004	B-196: Implementációs taszkhoz nem köthető feladatok	Task	
143005	B-196: Integrálási feladatok	Task	
142126	B-207: Source review/testing feladatok	Task	
142127	B-207: Implementációs taszkhoz nem köthető feladatok	Task	
142128	B-207: Integrálási feladatok	Task	
125940	IFC: Identical geometry should be handled as mapped	Bug - Functionality	
130300	MEP: a letett MEP elemek ID-ja nem növekszik	Bug - Functionality	
130343	IFC: (0,0,0) coordinate and unit direction vectors aren't reused	Bug - Request	
132926	IFC: Unittest Asserts while creating fake IOInfo	Bug - Assert / Memory Leak	
138766	IFC: Better Project Info derivation to IFC	Bug - Design Issue	Gropius (ArchiCAD 17)
139636	IFC: Window geometry is not mapped	Bug - Functionality	
139655	IFC: Dump IFC data with Dump2XML	Bug - Request	Gropius (ArchiCAD 17)
140730	IFC: ArchiCAD opens any file which has an IFC extension even though it is not a real IFC	Bug - Design Issue	Gropius (ArchiCAD 17)
140743	IFC: IfcLogical and Boolean type parameter value's are all uppercase characters even if Common Localiz...	Bug - Request	Gropius (ArchiCAD 17)
141809	B-196: InfoStrings' 15 char limitation has to be removed from ArchiCAD	Task	
144361	IFC: warning when deleting IFC parameter	Bug - Design Issue	Gropius (ArchiCAD 17)
144377	IFC grid lines from revit not showing on all stories	Bug - Functionality	Gropius (ArchiCAD 17)
147260	IFC: Beam with Gables doesn't imported correctly	Bug - Functionality	
148196	IFC: B-196-M2: Beam not in the right place	Bug - Functionality	Gropius (ArchiCAD 17)
148197	IFC Manager: Name of elements should be editable here	Bug - Request	
125942	B-196-M2: Brep slanted IFCslabs are imported as brep objects	Bug - Functionality	Gropius (ArchiCAD 17)
130215	IFC: Slanted extruded slabs imported as objects not as roof	Bug - Functionality	Gropius (ArchiCAD 17)
139403	CodeReview: Merge from B-164 Correction	Bug - Cosmetic	Wright (ArchiCAD 16)
144834	B-196-M2: Roof import	Task	
135652	ODB - Assert while opening file	Bug - Assert / Memory Leak	Wright (ArchiCAD 16)
146501	B-196-M1: WallStandardCase offset differentiation	Bug - Cosmetic	Gropius (ArchiCAD 17)
130001	IFC: IFC Manager becomes invalid (outdated) after every Save operation	Bug - Functionality	Wright (ArchiCAD 16)
137319	IFC: Cylindric column profile is very segmented by importing attached IFC file	Bug - Functionality	Gropius (ArchiCAD 17)
137413	IFC: IFC unit test crashes on second run	Bug - Crash	Wright (ArchiCAD 16)
138551	IFC: Sorting translators by Project or Partner data	Bug - Functionality	Wright (ArchiCAD 16)
139785	IFC: duplicate IFC Translator does not work for an item of its	Bug - Functionality	Wright (ArchiCAD 16)
140935	Unnecessary folder icon in Add button	Bug - Functionality	Wright (ArchiCAD 16)
141685	IFC - Wrong representation of Windows/Doors after export	Bug - Functionality	Wright (ArchiCAD 16)

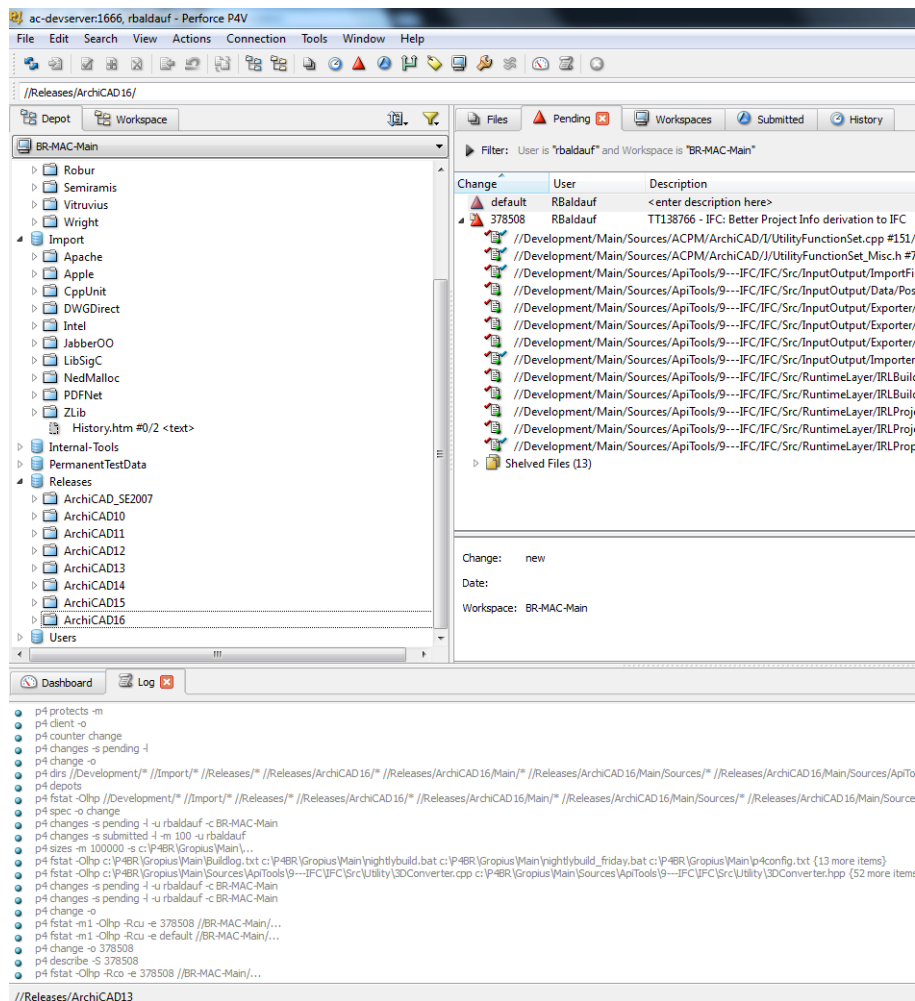
20. ábra A TestTrack alkalmazás, ahol a feladatokat adminisztráljuk

Az előbb említett közös code review során kerül használatba az Araxis Merge alkalmazás, ahol tökéletesen látható az általunk elvégzett módosítás a fájlba az előző (szerveren lévő) verzióhoz képest. A program maga képes összehasonlítani kettő, maximum három fájlt karakteresen, így nem kell keresni a fájlban belüli eltéréseket, azok azonnal láthatóak és módosíthatók is. Segíti a fejlesztő munkáját, ha több verzióba is be kell ugyanazt a javítást másolni.

Az ArchiCAD verziókövetése a Perforce használatával történik. A főágban zajlik a mostani ArchiCAD 17-es verzió fejlesztése (ide másolják be a már kész projekteket), minden egyes projekt, ami adott fejlesztési scope-ban¹⁴ található külön ágot kap, ezek az ágak gyakorlatilag külön leválasztott ArchiCAD-ek a főágról. A főágon zajlanak a hibajavítások, ahonnan lemásolnak egy adott jól letesztelt ArchiCAD verziót, amiből lesz az adott projekt ága és a fejlesztés során hetente egyszer integrálunk a főággal, hogy kövessük az ArchiCAD főágán végzett módosításokat. Az integrálás során előfordulhat az, hogy a már elkészült taszk, ami benne van az adott ágba és a főágról érkező forráskód ütközést mutat. Két ember ugyanazt a fájlt módosította, amit a Perforce nem tud automatikusan összeilleszteni, ekkor a fejlesztő maga végez rajta módosítást, úgy hogy mind a kettő funkció elérje a célját. Ha ezeket az ütközéseket hetente feloldjuk megfelelően, akkor

¹⁴ A scope az, amiben felméri, hogy mennyi fejlesztést leszünk képesek bevinni az adott fejlesztési fázisban, mennyi ember dolgozhat az adott projekten, ha ez kevés, akkor a projektet áteszik a következő verzióra

amint elkészült a projekt és azt bemásoljuk a főágba kevés ütközés lesz, mivel minden héten az integrálás során azokat megfelelően kezeltük. A Perforce használatához szorosan kötődik a TestTrack is, mivel TestTrack-ben található meg a hibák ezért logikus lenne úgy végezni a módosításokat, hogy azok csoportosítva legyenek, ne legyen keveredés. Erre használjuk a Perforce-ban használatos changelisteket, amely tartalmazza, hogy melyik TestTrack-en belüli hibáról, taszkról van szó. A 21. ábrán is beazonosítható egy ilyen changelist, amit a piros háromszög jelöl és ott látható egy lista a módosítandó fájlokról, illetve amire felhívnam még a figyelmet, az a changelist elnevezése, ami „TT138766 – IFC: Better...” nevet kapta. Itt az elején az azonosító az a TestTrack-en lévő hibát azonosítja, vagyis amíg egy hiba nincs kész, addig köthető hozzá legalább egy ilyen changelist.



21. ábra A Perforce, amivel az ArchiCAD verziókezelése zajlik

Az ArchiCAD-en belül a csapatunk az IFC (Industry Foundation Classes) részét fejleszti az programnak. Ez egy fájlformátum is egyben, amely sok neves építészeti szoftver is használ, az egyik ilyen szoftver a Solibri. Mivel ők kifejezetten az IFC fájlformátumot használják, ezért kitűnően alkalmas tesztelésre is. Kifejezetten hasznos import, export tesztek esetén. Solibri mellett IFCView-ert is használunk, ez a szoftver szövegesen jeleníti meg egy IFC fájl tartalmát, lényegesen leegyszerűsíti a hiba javítását, hiszen nem kell nekünk az egész fájlt betölteni, elég csak azt az

oszlopot, gerendát tesztelni, ami rossz. Ezt a megfelelő módosításokkal könnyedén meg tudjuk tenni.

A közös, megbeszélést igénylő eseteket általában Skype-on szoktuk megbeszélni, az előre betervezett gyűléseket/meetingeket pedig MS Outlook-ban egyeztetjük. A GoToMeeting pedig a saját gépünk felhasználói felületét tudjuk megosztani.

6.2. A belső intranet részei

A belső intraneten érhetőek el az egyéb dokumentációk, fejlesztői oldalak, minden, ami a céggel kapcsolat belső információt tartalmaz, ilyen fontos oldalak¹⁵:

- Info és Development site
- CruiseControl monitor
- Tinderbox Speedtest, ConvertLog
- TimeTrack, MS Exchange oldal (Outlook)
- Developer Wiki site

Az Info site-on helyezkednek el a jövőbeli és aktuális hírei a cégnek a dolgozók számára, például itt jelenik meg az újonnan felvett emberekről rövid információ, ki van szabadságon. Tájékozódhatunk a csapat tagjainak szabadságairól. Az Info site körül öleli az egész vállalati struktúrát, külön részekre van szedve, mint például emberi erőforrás, marketing és fejlesztői részlegekre. Ezért is a development site része az Info sitenak. Itt tartják nyilván az utoljára kiadott ArchiCAD verzióknak a státuszait, ki másolhat be adott nap a forráskód állományba (azaz ki kapott engedélyt bemásolásra), milyen projekt bemásolások futnak. Ráadásul ezek a legfontosabb információk a fejlesztők számára.

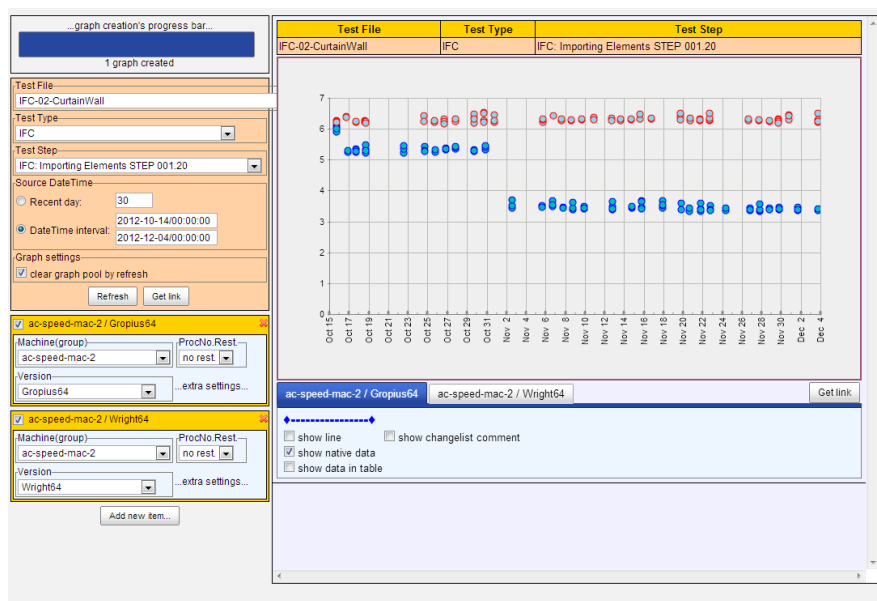
	ac-cc-win32	ac-cc-win64	ac-cc-win-dev32	ac-cc-win-dev64	ac-cc-mac-fin	ac-cc-mac-dev	ac-cc-lint-fin	ac-cc-lint-dev	ac-cc-odbc-fin	ac-cc-odbc64fin
P4@382924 12/21 13:56 by gleher "DACH build failed javítások"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382889 12/21 10:43 by ppopovics "ERR150442 - Crash #3967 - Morph"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382859 12/20 22:01 by minarik "Groplus <bnum>139</bnum> on <ti>"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382734 12/20 11:18 by vschnedarek "TT#150367 - IFC: Missing icon o"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382658 12/19 22:01 by minarik "Groplus <bnum>138</bnum> on <ti>"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382611 12/19 15:56 by jkiss "Integration error fix"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382591 12/19 15:03 by csmondics "R150355 - C4DRendererPlugin: Ex"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382276 12/19 09:45 by gleher "LOC build failed javítások"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382178 12/19 01:14 by csmondics "Fix Cinema4DRendererPlugin build"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382173 12/18 23:54 by csmondics "Fix LINT"	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded
P4@382172 12/18 23:34 by csmondics "Fix build on mac after merge A"	Failed	Failed	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded	Failed	Failed	Failed

22. ábra A CruiseControl monitor, ahol fordítják az ArchiCAD

Cruise Control Monitoron lehet nyomon követni a verziók fordítását (ArchiCAD 15, 16, 17), a fenti ábrán a 17-es verzió fordítása látható. Például, ha valaki bemásol egy javítást a főágba, akkor azt 32, 64 bites verziókra fordítják eltérő futtatási környezetekbe. Ilyen környezet lehet Mac, Windows. Fontos megemlíteni, hogy még 32 és 64 bites fordításon is belül van különbség. Elkészítik fejlesztői (dev-es) és végső, kiadandó (final) fordítást is. A kettő között az a különbség,

¹⁵ Itt csak a fejlesztői szemszögből fontos oldalak kerülnek megemlítésre, a könyveléshez, emberi erőforrással foglalkozó részlegek oldalai nem kerülnek részletezésre

hogy a fejlesztői tartalmaz asserteket¹⁶ (belefordítja a fordító), amik segítik a fejlesztők munkáját. Ezeket az asserteket, mi fejlesztők tesszük bele, arra az esetre, ha olyan értékevítés történik egy függvénybe, ami nem megengedett. A végső fordítás ezeket az asserteket már nem tartalmazza, optimalizált, speed test méréseket is ezen verzión végzik. CruiseControl-on történik az éjszakai fordítás is, általában a fejlesztői gépek is be vannak állítva a hétköznapi esti fordításra (miután megtörtént a Build szám növelés), hogy másnapra naprakész verzióon folytathassuk a munkát. Minden egyes sor a 22. ábrán egy bemásolás **Perforce**-on keresztül, röviden olvasható a changelist neve is, illetve megtekinthető, mely fordítással vannak problémák, azokat sürgősen javítani kell, hiszen blokkolhatja a következő GSAT napi teszt futását, sőt akár a következő ArchiCAD verzió kiadását.



23. ábra TinderBox sebességmérés webes felülete

TinderBox segítségével követhetjük az egyes verziókon mért lassulásokat és gyorsulásokat különféle műveleteket lemérve (pl.: import, export, programon belüli konvertálás). Diagramok segítségével össze is hasonlítható az egyes verziók közötti különbség, ami a fenti 23. ábrán is látható, ugyanis az ArchiCAD 17-es verzióban történt egy gyorsulás (míg ArchiCAD 16-ba nem), amit ki kellett nyomozni miért is történt, hiszen lehet hibához köthető. Ugyanis, ha eltérés mérés történt egy korábbi buildhez képest, amiről nem tudtuk, hogy előre be fog következni, azt a ConvertLog programmal tudjuk a javítás során ellenőrizni. Ez egy olyan program, ami az ArchiCAD működése közben készített *.log fájlból készít egy számunkra jobban értelmezhető szövegfájlt (megfelelő parancssori paraméterekkel megadható, mire vagyunk kíváncsiak) ahol ellenőrizhetjük történt-e gyorsulás. A 20. ábrán látható a teszt fájl neve és alatta a teszt lépése, ennek segítségével szűrhetjük ki az ArchiCAD által készített *.log fájlt a ConverLog programmal.

Minden olyan információ, amivel fel tudjuk mérni mennyi erőforrást igényelt egy adott hiba, task, projekt azt a TimeTrack-be könyveljük. Itt gyakorlatilag a fejlesztők írják be minden napi tevékenységüket, melyik hibával foglalkoztak adott nap, milyen projekt megbeszéléseken voltak. Nyilvánvaló, hogy általában nyolc vagy több óra kerül be egy napra. Maga a TimeTrack egy

¹⁶ Ezek hibaüzenetek, melyek tájékoztatják a többi fejlesztőt az ArchiCAD hibás működéséről

munkanap alatt nyolc órát értelméz. Ezen az oldalon keresztül ellenőrizhető, ki mit csinált, mennyi munka ment bele egy projektbe és a felsővezetők ebből is tudják írni a heti jelentéseket a dolgozók számára. Ezek a jelentéseket egy levelező listához hasonló módon publikálják. A jelentéseket webes felületen is elérhetjük. Outlookon keresztül nézzük meg a saját céges e-mail címünkön. MS Outlookban is van erre lehetőség természetesen.

Fejlesztői Wiki oldalon van fent minden projektnek a specifikációja, modellje (Overall Model, melyet a fejlesztő készít el), amin keresztül láthatók a fejlesztő(k) jövőbeli tervei az adott projekttel kapcsolatban, ami ha jóváhagyásra kerül, nekiállhatunk a fejlesztésnek. Minden fejlesztő monogramja itt található, amit az általa elkészített saját osztályoknál kötelező feltüntetni. Fontos jövőbeli döntések is ezen az oldalon kerülnek publikálásra. A már elkészült projektek is hasznos információt tartalmaznak a jövő számára, nyomon követhető, hogy az egyes fejlesztési fázisokban mennyit töltött az adott projekt. Szükséges külső linkek is itt kerülnek elhelyezésre.

7. A FEJLESZTÉS MENETE

A fejlesztés belső kiépítése ezen projekt esetén eltérő a szokásoshoz képest. Általánosságban minden új ArchiCAD-ben készülő feature esetén új al-ágat (branch) hoznak létre, ahol a fejlesztés zajlik. Ez általában akkor valósul meg így, ha előre tisztáztak a követelmények, viszont a CV 2.0 szabvány teszt fájljait a projekt haladása során változtatták és nem lehetett tudni előre melyik az, amelyik meg fog felelni és melyik az, amelyik nem. Ebből következően a változó követelmények miatt két ágat hoztunk létre, az egyiket folyamatosan ment a fejlesztés a másik ágon pedig a tesztelést valósítottuk meg.

Ezeket az ágakat egy erre speciálisan kifejlesztett alkalmazással – amelynek neve **Perforce** – működtetjük megfelelően. Lehetőségünk van minden egyes projektet külön kezelni, fejleszteni. Viszont van egy kitüntetett ág a fő-ág, ebből fog készülni majd a végleges verzió. Ide lesz előbb utóbb minden projekt bemásolva, majd kiadás előtt itt javítjuk a felmerülő hibákat. **Perforce** alkalmazásáról bővebben volt szó a Felhasznált szoftverek című fejezetben. Egy projekt fejlesztési ciklusa a következő fázisokból tevődik össze:

1. Értékelési (Evaluate)
2. Specifikáció (Specification)
3. Dizájn és fejlesztés (Design and Development)
4. Tesztelési (Released for Testing)
5. Készenléti (Completed)
6. Felfüggesztett (Suspended)
7. Leállított (Cancelled)

A fejlesztési fázisok közül, jelenleg is a specifikációs fázisban vagyunk, mivel amíg nincsenek kész a konkrét követelmények addig nem léphetünk át a dizájn és fejlesztési fázisba, de ennek ellenére kénytelenek vagyunk az eddig létrehozott követelményeket folyamatosan megvalósítani. A projekt indulásakor az értékelésnél mindig felméri, hogy hány embert fog felölelni a projekt jelenleg három ember dolgozik a készítésén, de ez nem öleli fel a tesztelésre vonatkozó létszámot, illetve a projekt menedzsert, dizájnert, kontroller és a termék menedzser sem foglalja magába.

7.1. Szerepkörök a fejlesztésben

A projekt implementációját pedig a szoftverfejlesztők valósítják meg, ők kezelik a hibákat és a taszkokat a **TestTrack**-ben, illetve végzik az integrálásokat a **Perforce** segítségével. A fontos eseteket a project menedzserrel és fejlesztő társakkal együtt beszéljük meg, hogy későbbiek során ne legyen a megoldással probléma. Az overall modell elkészítésével pedig fel tudjuk becsülni a szükséges erőforrásokat, mint emberszámba és időben egyaránt. Amint elérünk egy mérföldkövet azt tesztelésre bocsájtjuk és az ott keletkező hibákat azon nyomban javítjuk, addig a következő mérföldkö scopejában elhelyezkedő implementálásra váró taszkokat szüneteltetjük. Ez igen rizikós része az időbecslésnek, mert nem lehet tudni előre a hibajelenségek mértékét, csak becsülni tudjuk.

A projekt menedzsere az, aki felügyeli a folyamatos fejlesztés menetét, ellenőrzi melyik taszk mennyi erőforrásba és a szoftver fejlesztőknek előre jelezni kell, ha valamelyik taszk csúszik, mert ez befolyásolhatja a projekt ArchiCAD-beli kiadását. Heti gyűléseket (meetingeket) is ő szervezi, ahol közérdekű információkat közzétesznek a projekttel kapcsolatban, az itt elhangzott döntéseket dokumentáljuk és későbbiek során ezekhez tartjuk magunkat. Továbbá lehetőség van a projekt támogatása érdekében egyéb hardver/szoftver erőforrásokat igényelni a fejlesztőknek, melyeket adott időközönként lehet csak bejelenteni és a költségvonzatának, prioritásának megfelelően teljesítik azokat.

A kontroller az a személy, aki a legfontosabb döntések meghozásában kell, hogy szerepet vállaljon, egyáltalán elindulhat-e a projekt, csak adott számú szoftverfejlesztő vehet részt a fejlesztésben. Hosszú távú döntések tipikusan ide tartoznak. Ennek a dolgozatnak a létrejöttéhez is az ő engedélyére volt szükség, mely esetben meg kellett vitatni milyen információkat lehet közzé tenni. Projekt bemásoláshoz a fő-ágba tőle kell engedélyt kérni, mivel ez teljes mértékben érint minden projektet a cégnél, ugyanis az adott nap nem lehet semmi mást hozzáadni a főághoz, csakis az adott projektet, még egyéb súlyos hibákat sem lehet ilyenkor bemásolni. Ő az, aki ellenőrzi az overall modellt, a többi projekt vonatkozásában, lehetséges ütközések problémák észlelése esetén biztosítani a megfelelő közreműködést a projektek között.

A termék menedzsere (product manager) segíti létrehozni a specifikációt, ezáltal kerülnek meghatározásra a projekt keretei. A követelmények létrehozásában is aktív szerepet játszik, mivel ő a közvetítő a kívülről érkező igényeknek, amik a felhasználóktól, nagy cégektől (mint pl. a GSA) érkeznek. Ezért is kritikus, hogy ellenőrizze az elkészült funkciók helyességét, hiszen ő helyezkedik el legközelebb a potenciális vásárlókhoz. Közvetítő szerepe miatt a követelmények változása esetén sürgősen tájékoztatja a projekt résztvevőit.

A dizájnér feladatai közé tartozik a specifikáció és a követelmények létrehozása a termék menedzserrel közösen közreműködve. Minden egyes funkciót, amit létre kell hozni az ArchiCAD-ben el kell látni egy felhasználói prioritással (user experience) ami az implementálási sorrendben számít fontos szerepet. A szoftver fejlesztőket segítve közösen létrehozzák az overall modellt, amely alapján majd elkészíti a specifikációt. Illetve ő is ellenőrzi, hogy az elkészült implementáció megfelel-e a követelményeknek.

A tesztmérnökök (Quality Assurance Engineer) létrehozzák az új funkciók számára a szükséges teszteseteket. Általában már a szoftverfejlesztők rendelkeznek a szükséges fájlokkal a helyes funkciók tesztelésére, így azokat a **TestTrask**-en keresztül el tudják érni (mivel feltöltik oda a fájlokat) és ezek a tesztesetek a GSAT-nak köszönhetően bekerülnek a napi tesztek közé. Esetleges kérdések felmerülése esetén felkeresi a projekt résztvevőit.

7.2. Projekt életciklusának fázisai

Most, hogy tisztáztuk kinek milyen feladatköre van részletesen be tudjuk mutatni a projekt különböző fázisait, ahol a legelső az értékelési fázis, ilyenkor a felmerülő ötleteket kezeljük, mi az ami megvalósítható, kevésbé időigényes és hasznos feature, kiszűrjük a kockázatos ötleteket és elkészítünk egy elérhető listát a szükséges új funkciókról. Ezt a folyamatot a projekt menedzser vezeti közreműködve a termék menedzserrel és a dizájnérrel. Létrehoznak egy magas szintű overall modellt, egy korai időbecslést állapítanak meg.

Ezt követően a specifikációs fázisban a cél az, hogy befejezzék az overall modellt, amit a szoftverfejlesztők vesznek át és készítenek el. Esetleg opcionálisan egy implementációs tervet készítenek el. A dizájn elkészíti a használati use-case diagramokat, milyen esetekben kerülnek használatra az adott funkciók. A fejlesztők minden egyes felvett taszknak beállítják az időbecslést és azt folyamatosan frissítik, míg el nem készül. A konkrét követelményeknek megfelelően beállítjuk a projekt befejezési határidejét, akár mérföldkövekre lebontva. Itt kerül bemutatásra az overall modell a controllernek és kerül elbírálásra, hogy átléphet-e a dizájn és fejlesztési fázisba.

A dizájn és fejlesztési fázisban az eltervezett funkcionalitást időben megvalósítjuk a rendelkezésre álló erőforrások között a megfelelő minőségben. Ennek a folyamatnak a vezetője is a projekt menedzser. Ezért az ide tartozó feladatok: lebontani az implementációs tervet fejlesztési és dizájn feladatokra. Részletes felhasználói interfész és végleges specifikációs dokumentum elkészítése, ahol a kezelő felület is már bemutatásra kerül. A QA mérnök számára itt kerülnek kifejlesztésre a már korábban említett teszt fájlok és elkészül egy szógyűjtemény is az újonnan bevezetett fogalmakból. Az elkészült taszkokat a QA mérnök közösen a projekt menedzserrel verifikálják és akkor lesz tesztelési állapotba helyezve a projekt, ha a verifikáláson minden egyes feladat átesett és le lett zárva.

Tesztelési fázisban a GSAT segítségével minden létező már megírt tesztet lefuttatnak a projekten, hogy a lehető legtöbb hibát feltárják. Itt már azok a tesztesetek is futnak természetesen, melyek az új taszkokhoz kötődnek. A lefutott teszteket a QA mérnökök kiértékelik. Az export/import folyamatot igénylő tesztesetekre **Tinderbox** méréseket alkalmaznak, hol történt lassulás, illetve arra lehetett számítani. Ezeket a méréseket természetesen nem a fejlesztői fordításon mérik, hanem a végső (final) fordításon, ami kiadásra kerül. Ebben a fázisban kerül sor a fő-ágba történő bemásolásra és innét már részét képezi a kiadásra tervezett ArchiCAD verzióknak. A következő készenléti fázis (completed) automatikusan bekövetkezik amint megtörtént a fő-ágba történő bemásolás.

Az utolsó két fázis közül a felfüggesztett állapotba ritkán kerülnek projektet, de általában akkor szokott ez bekövetkezni, ha nincs elég ember az adott projektre, ilyenkor elhalasztják a következő verzióknak fejlesztési idejére. A leállított fázis esetén a projekt el lett indítva, de valamilyen oknál fogva meg lett szakítva, ilyenkor minden szükséges fájlt le kell menteni és majd egy új projekt keretén kerül folytatásra ahol végig haladunk majd az összes projekt fázison¹⁷.

¹⁷ Természetesen itt a felsorolás első 5 pontjáról van szó, hiszen az utolsó 2 fázis egy normál projekt esetén be sem következik.

8. ÖSSZEFOGLALÁS

A dolgozat, amelynek célja az IFC szabvány Coordination View 2.0 nézetének ArchiCAD 17-ben történő implementálása és bemutatása a korábbi fejezetekben taglalt megoldásoknak köszönhetően megfelelően lezajlott. Így ennek köszönhetően több sikeres újítást vittünk be az új verzióba, amit 2013 nyarán fog kiadni a Graphisoft SE Zrt. világszerte. A csapattagok, akikkel mindezt végigvittük hatalmas segítséget nyújtottak munka közben, lehetséges ötletekkel és megoldásokkal, amivel végül a problémákat megoldottuk és sikeresen befejeztük a projektet. Segítségüket ez úton is köszönöm.

A módszerek, amikkel a projekt és a dolgozat is elkészült, az a rendszeres konzultációk a projekt tagjaival és a belső konzulenssel nélkülözhetetlen volt a dolgozat létrejöttének, illetve a cégen belüli felsővezetői engedély megadásával jöhetett létre ez a szellemi alkotás is. Az ArchiCAD 17 így már rendelkezni fog az alábbi funkciókkal:

- Íves, egyenes fal geometriai reprezentációja a vágások megfelelő kezelésével.
- Födém vágásainak megjelenítése a szabvány által elvárt módon.
- Íves, egyenes gerenda jobbsodrású koordináta-rendszerben történő ábrázolása és annak íveinek megfelelő kezelése.
- Egysíkú tető exportálása és importálása, illetve több síkból álló tető exportjának megvalósítása.
- Oszlopok vágásainak megfelelő elhelyezése.

Jelenleg a Coordination View 2.0 certifikációját még nem nyertük el, mert a buildingSMART-nak le kell tesztelni a szükséges fájlokkal az új programot, ebből néhány export teszt végrehajtása már megtörtént, de a teljes import/export teszthalmazból még maradtak hátra funkciók, amik kiértékelésre várnak. Ezen tesztek sikeres végrehajtása után nyerhetjük el a CV 2.0 certifikációt.

Személyes céljaim közé tartozott, hogy áthatóan megismerjem a szabvány szükséges elemeit és azoknak ArchiCAD-beli megjelenését, továbbá az IFC forráskód részét teljes mértékben feltérképezsem, mivel általában IFC oldalát fejlesztjük az ArchiCAD-nek. Így a későbbiek során jobban, gyorsabban és a megfelelő helyen tudom megoldani a szükséges fejlesztéseket. Úgy gondolom ezzel a dolgozattal közelebb kerültem a kitűzött célomhoz, így amint elkészül az IFC2x4-es szabvány, a megjelenő innovatív megoldásokat könnyedén el tudom helyezni az ArchiCAD IFC forráskódjában.

9. IRODALOMJEGYZÉK

- [1] Graphisoft, *Graphisoft GDL felhasználói kézikönyv*, Graphisoft 2001 Budapest, 262 pages
- [2] Wolters Kruwer, *General Services Administration Acquisition Manual*, 2012 Riverwoods, 469 pages
- [3] Jesse Russell, Ronald Cohn, *Industry Foundation Classes*, Book on Demand Ltd. 2012 Stoughton, 180 pages
- [4] Tulio A. Sulbaran, Mohd Shiratuddin, Shane Germany, *Intruduction to ArchiCAD: A BIM Application*, Delmar Cengage Learning First edition 2010 New York, 240 pages
- [5] Simon Gilbert, *ArchiCAD for AutoCAD Users v15-International*, Graphisoft R&D 2011 Budapest, 269 pages
- [6] Stroustrup Bjarne, *The C++ programing language*, AT&T, 1997 New Jersey, 962 pages
- [7] Scott Meyers, *The Effective C++ Third Edition 55 Specific Ways to Improve Your Programs and Designs* , Addison Wesley Professional 2005 Boston, 320 pages
- [8] Ian Sommerville, *Szoftver rendszerek fejlesztése*, Panem kiadó 2002 Budapest, 752 pages
- [9] General Services Administration:
http://en.wikipedia.org/wiki/General_Services_Administration
- [10] Industry Foundation Classes 2x3 Standard:
<http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm>