



WS 2018/19

Skript - Kap. 5

Autodesk Revit 2019 SDK (3. Hörsaalübung)

Prof. Dr.-Ing. Uwe Rüppel
Meiling Shi, M.Sc.

- Revit SDK Grundlage
- Revit Erweiterung
 - Events
 - Parameter ändern
 - Parameter anlegen
 - Objekte platzieren
- Ausgabe 2. Hausübung



Autodesk Revit 2019 SDK



TECHNISCHE
UNIVERSITÄT
DARMSTADT



AUTODESK® REVIT® 2019



AUTODESK.

BIM mit Autodesk Revit 2019

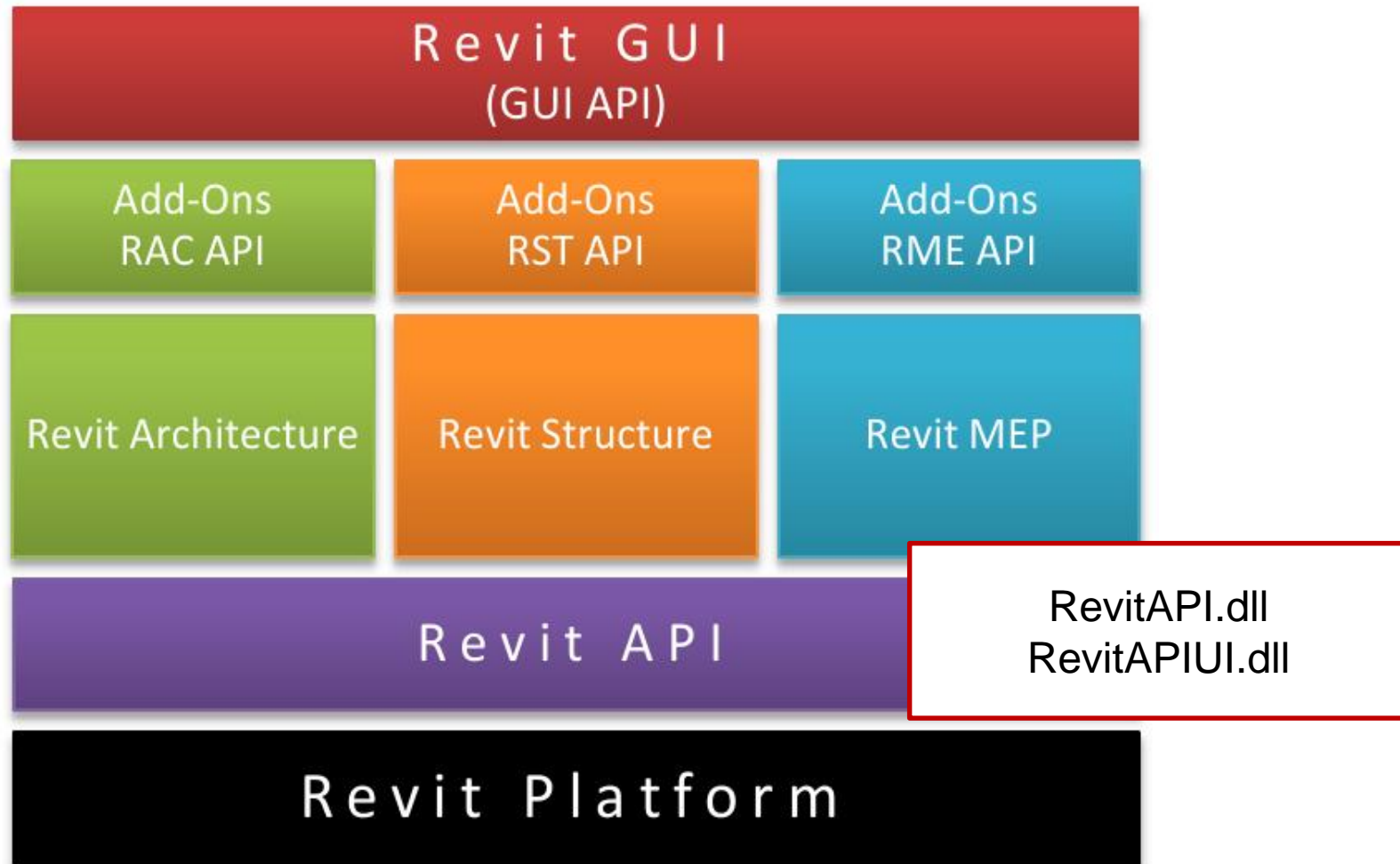
- Software für Building Information Modeling (BIM)
 - Parametrische 3D-Modellierung
 - Bauteilorientiertes Gebäudemodell
(z.B. Wände und Türen anstatt Quader mit Löchern)
 - **Das Gebäudemodell ist eine Datenbank!** (*.rvt-Datei)
 - Parametrische Objekte als „Familien“ (*.rfa) in der Datenbank abgelegt
 - Änderungen wirken sich auf das gesamte Gebäudemodell aus!
-
- **Autodesk Revit Architecture:** BIM für (architektonische) Planung
 - **Autodesk Revit MEP:** BIM in der Gebäudetechnik
 - **Autodesk Revit Structure:** BIM für konstruktiven Ingenieurbau



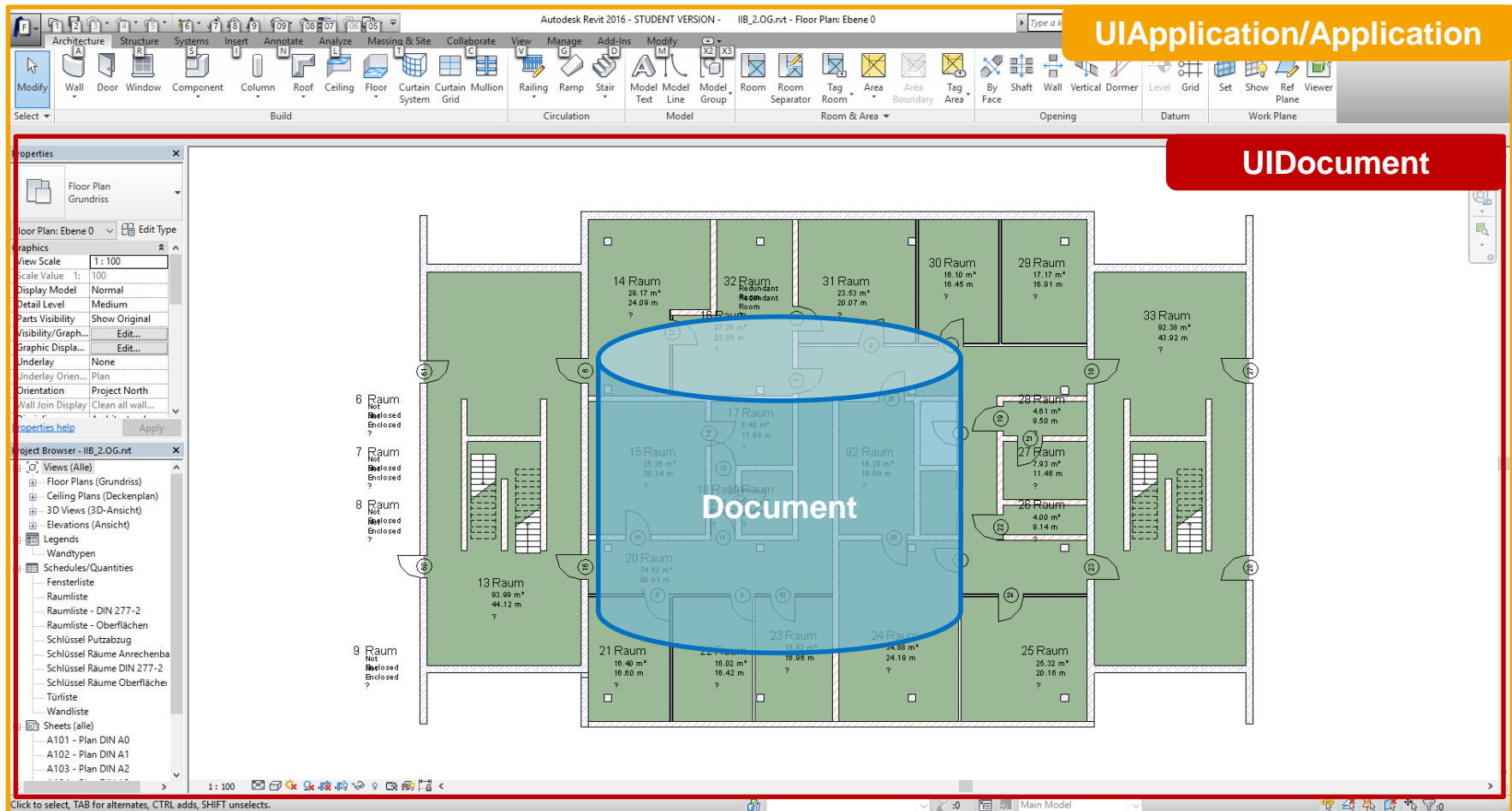
- SDK = Software Development Kit
- Revit-SDK / Revit-API → .NET-Schnittstelle (VB.NET und C#)
- Schnittstelle stellt einen Satz Bibliotheken zur Verfügung, der in das eigene Programm eingebunden werden kann
- Es wird eine DLL (dynamic link library) erzeugt, d.h. kein eigenständiges Programm (.exe), sondern eine Programmbibliothek, welche als Zusatzmodul (Add-In) in Revit geladen und genutzt werden kann.

- Benötigte Software:
 - Autodesk Revit 2019 → <http://www.autodesk.com/education/home>
(Bei Anmeldung: Verifizierung durch TU-Mailadresse)
 - Revit 2019 SDK
 - <https://www.autodesk.com/developer-network/platform-technologies/revit>
 - Microsoft Visual Studio 2017 → MSDNAA

Erweiterung von Autodesk Revit



Revit API Grundlagen – Application & Document



Revit API Grundlagen – Application & Document

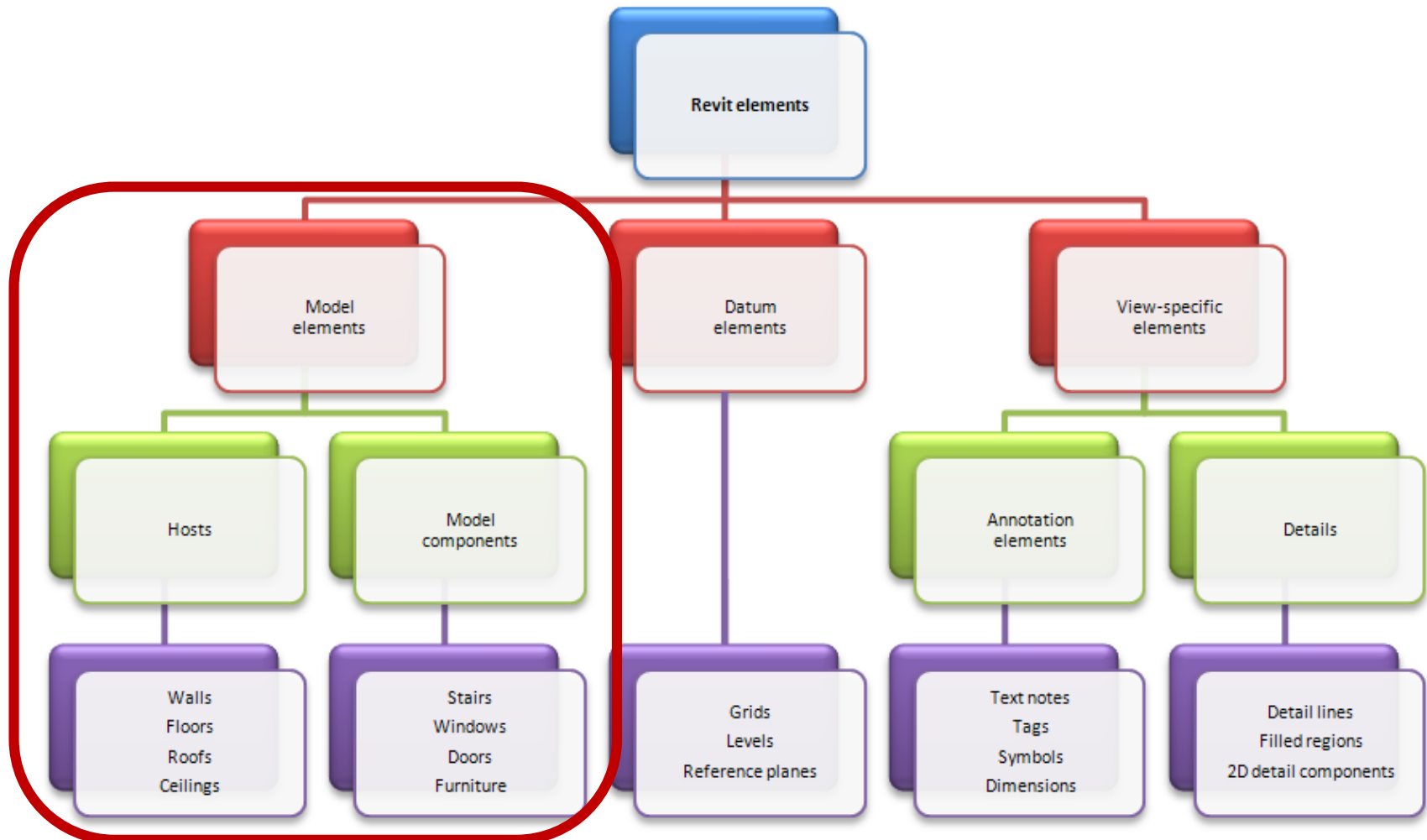
Top-Level Objekte der API

- `Autodesk.Revit.UI.UIApplication`: Zugang zur Benutzerschnittstelle der Anwendung, Zugang zum aktiven Dokument
- `Autodesk.Revit.ApplicationServices.Application`: alle weiteren Eigenschaften auf Anwendungsebene
- `Autodesk.Revit.UI.UIDocument`: Zugang zu allen Schnittstellen auf UI-Ebene, wie bspw. aktuelle Auswahl oder Interaktion mit Benutzer
- `Autodesk.Revit.DB.Document`: alle weiteren Eigenschaften auf Dokumentenebene



- Element kann eine Gebäudekomponente sein (z.B. Tür, Wand), aber auch ein Typ, eine Ansicht oder eine Materialdefinition etc.
- Revit unterscheidet 6 Klassen von Elementen:
 - **Model Elements**
 - **View Elements**
 - **Group Elements**
 - **Annotation / Datum Elements**
 - **Sketch Elements**
 - **Information Elements**

Revit API Grundlagen – Elements

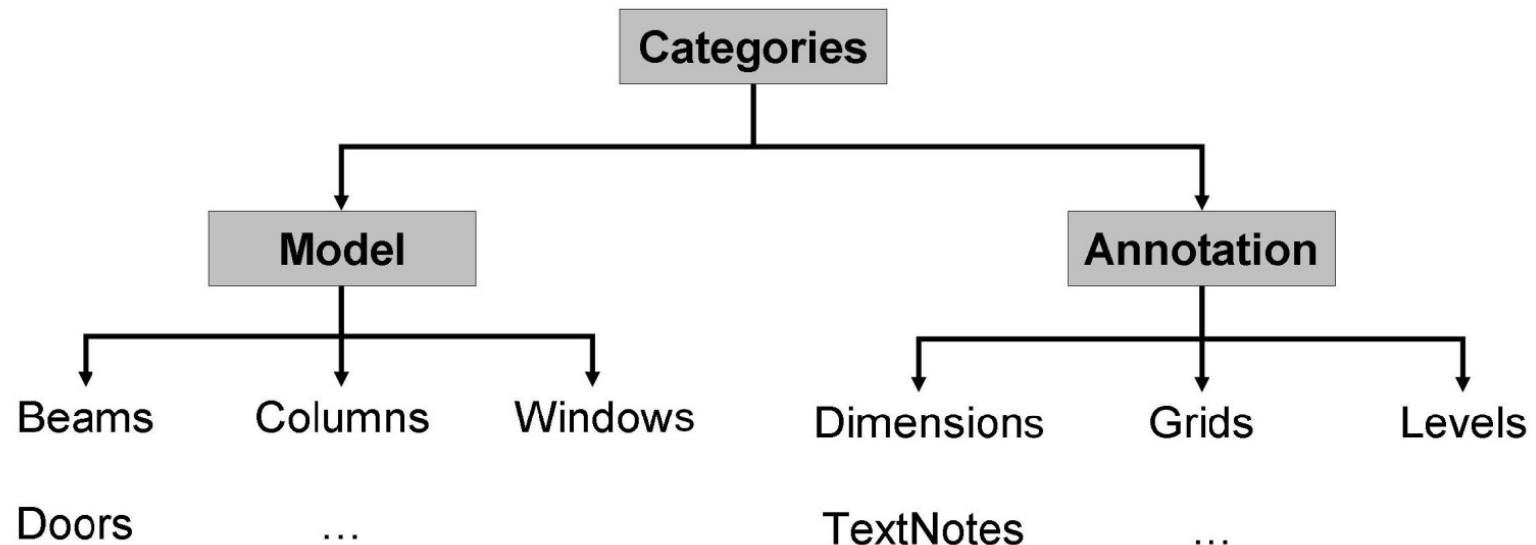


Model Elements = Physikalische Objekte eines Projekts

- Host Elements:
 - Systemfamilien (System Family Objects) die auch andere Modellelemente enthalten können
 - z.B. Wände, Decken, Dächer, etc.
- Model Components:
 - Gebäudeelemente/Bauteile, die normalerweise angeliefert und vor Ort montiert werden (z.B. Türen, Fenster, Möbel usw.)
 - Bauteile sind Exemplare ladbarer Familien (Family Instances) und von anderen Elementen (System Family Objects) abhängig (z.B. Tür abhängig von Wand)

Revit API Grundlagen – Element Categories

- Elemente/Element-Typen können auch nach Category, Family, Symbol oder Instance unterschieden werden (Nicht alle Elemente haben eine Category!)
- Nach der Category eines Elements kann gefiltert werden.



- Eigenschaften von Objekten
 - Property „Parameters“: alle vorhandenen Parameter des Objektes
 - Suche nach speziellen Parameter über Methode
„GetParameters(String parameterName)“: `IList<Parameter>`
 - Wert eines Parameters je nach Typ über:
 - `AsString()`
 - `AsDouble()`
 - ...
 - `AsValueString()`
- ➔ Revit LookUp !

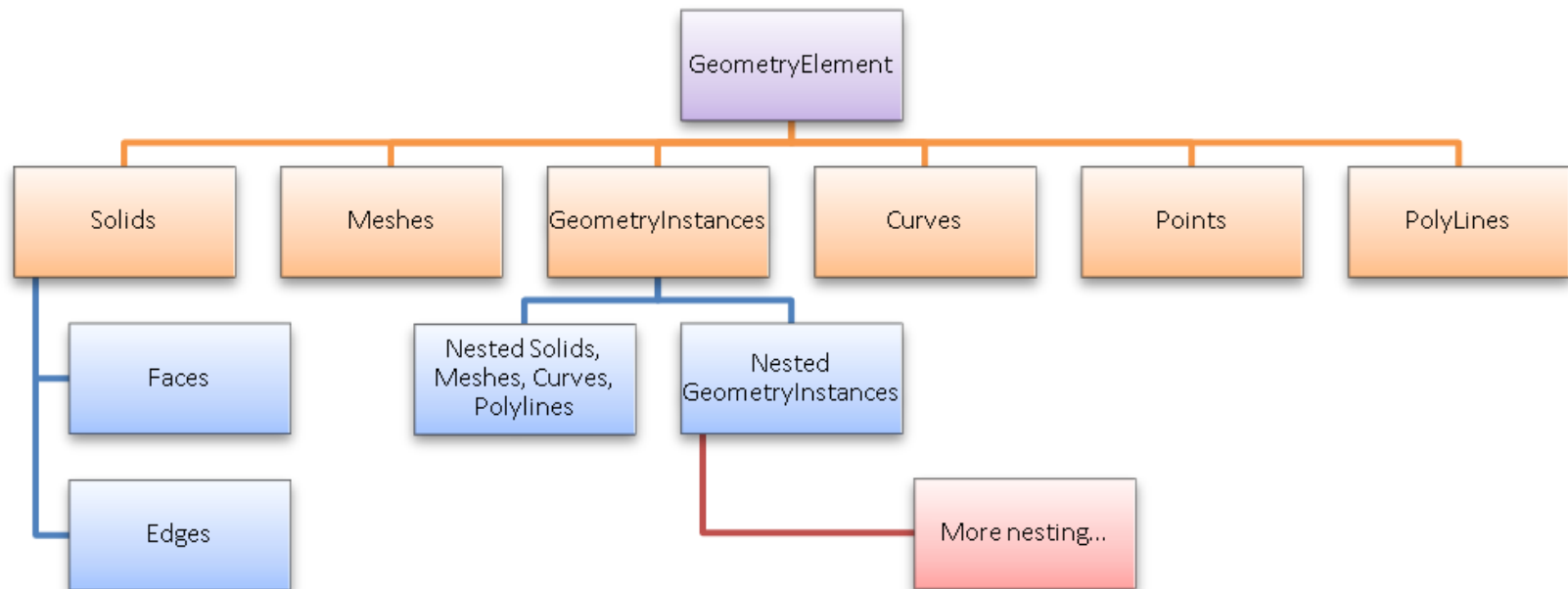
Methode parseRaum(Room room)



```
public static Raum parseRaum(Room room)
{
    List<FamilyInstance> revitFensterListe = findeAlleRaumFenster(room);
    BindingList<Fenster> fensterListe = parseFenster(revitFensterListe);
    double flaeche = squarefeetToQuadratmeter(room.Area);
    string raumtyp = room.GetParameters("Nutzungsgruppe DIN 277-2")[0].AsString();
    if (raumtyp == "2-Büroarbeit")
    {
        Buero buero = new Buero(flaeche, room.Number, fensterListe);
        return buero;
    } else if (raumtyp == "1-Wohnen und Aufenthalt")
    {
        Wohnen flur = new Wohnen(flaeche, room.Number, fensterListe);
        return flur;
    }
    return null;
}
```

Revit API Grundlagen – Geometrie von Model Elements

- Jedes Model Element (3D Objekt) hat eine Geometrie, welche durch ein Autodesk.Revit.DB.[GeometryElement](#) repräsentiert wird.



Revit API Grundlagen – Geometry Flavors

Autodesk.Revit.DB.Element

GeometryElement

GetEnumerator()

IEnumerator<GeometryObject>

Curve

Edge

...

Solid

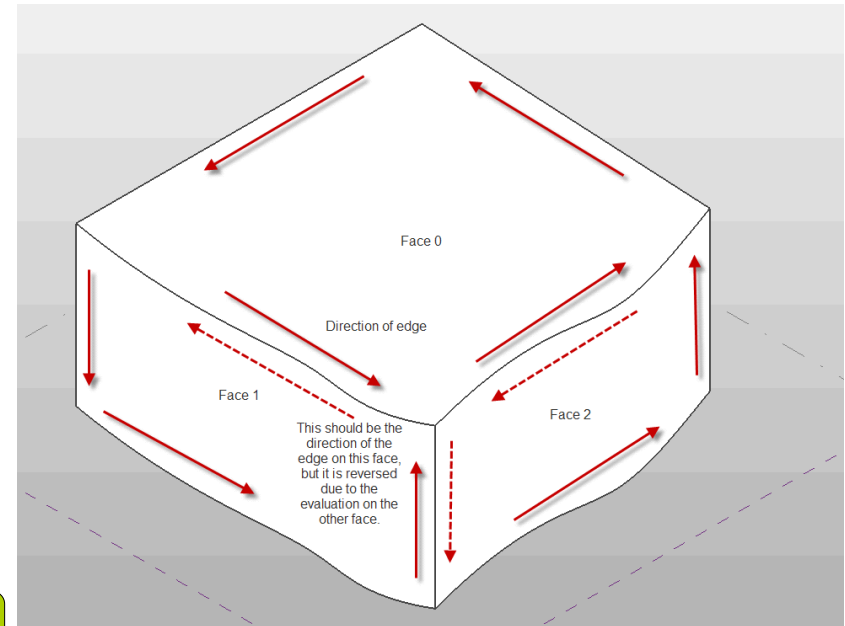
...

Faces

Tessellate()

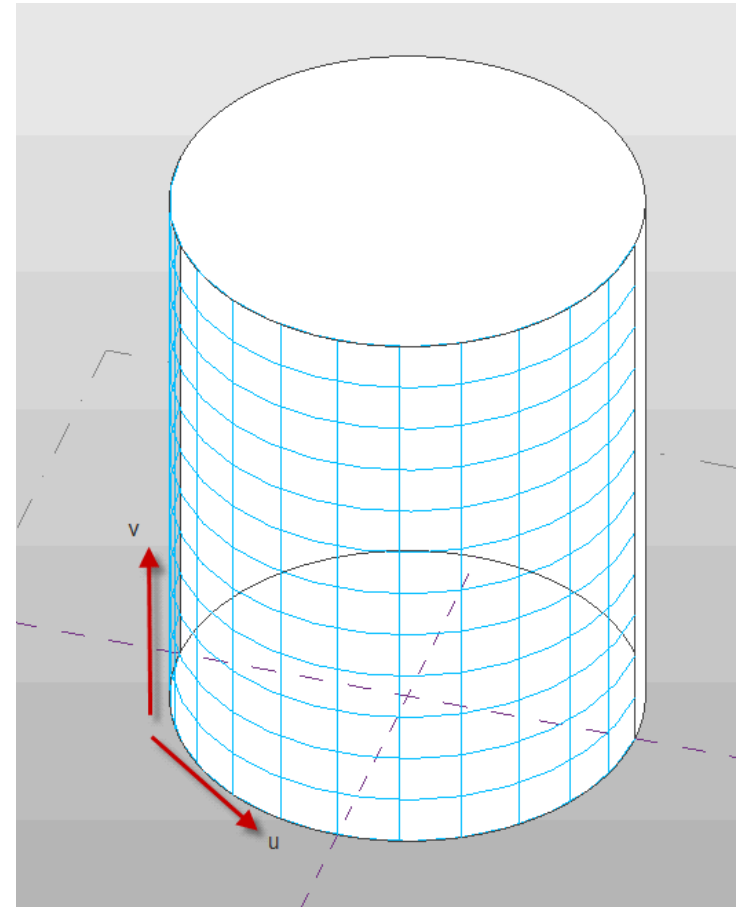
Edges

XYZ



Revit API Grundlagen – Faces

- Faces sind Oberflächen
- Jeder Punkt eines Face im XYZ-Raum wird durch eine Funktion der Parameter U und V abgebildet
- Die Richtungen von U und V werden automatisch in Abhängigkeit der Form des Face bestimmt
- Arten von Faces:
 - PlanarFace, CylindricalFace, ConicalFace, RevolvedFace, RuledFace, HermiteFace



<http://thebuildingcoder.typepad.com/blog/2010/01/faces.html>

Revit API Grundlagen – Eigenschaften und Methoden von Faces (1)

- Area: Flächeninhalt des Face
- EdgeLoops: Array von EdgeArrays, repräsentiert Begrenzung eines Face durch Kanten (Edges)
- IsInside(): Methode, die bestimmt, ob ein UV-Punkt innerhalb des Face liegt
- Evaluate(): liefert die XYZ-Koordinaten eines UV-Punkts
- Intersect(): berechnet die Verschneidung des Face mit einer Kurve
- Project(): projiziert einen Punkt auf das Face
- Triangulate(): liefert ein Mesh des Face aus Dreiecksflächen

Revit API Grundlagen – Eigenschaften und Methoden von Faces (2)

- `ComputeDerivatives()`: liefert ein Transform-Objekt zurück, welches das Face an einem bestimmten UV-Punkt beschreibt
 - `Transform.Origin`: XYZ-Koordinaten des UV-Punkts
 - `Transform.BasisX`: Tangentialvektor in U-Richtung
 - `Transform.BasisY`: Tangentialvektor in V-Richtung
 - `Transform.BasisZ`: Normalenvektor
- Besonderheit `PlanarFace`: `Normal` → Normalenvektor des Face

Methode groessteFensterflaeche(...)



```
private static double groessteFensterflaeche(FamilyInstance fenster)
{
    double re = 0;
    Autodesk.Revit.DB.Options opt = new Options();
    GeometryElement geomFenster = fenster.Symbol.get_Geometry(opt);
    foreach(GeometryObject geomObj in geomFenster)
    {
        Solid geomSolid = geomObj as Solid;
        if (geomSolid != null)
        {
            foreach(Face geomFace in geomSolid.Faces)
            {
                if (geomFace.Area > re)
                {
                    re = geomFace.Area;
                }
            }
        }
    }
    return re;
}
```

- Repräsentiert den Ortsvektor eines Punktes im dreidimensionalen Raum
- Ermöglicht verschiedene Vektoroperationen:
 - `Add()`, `Subtract()`, `Multiply()`, `Divide()`, `Negate()`
 - `DistanceTo()`, `AngleTo()`, `AngleOnPlaneTo()`
 - `DotProduct()`, `CrossProduct()`, `TripleProduct()`
 - `Normalize()`
 - `GetLength()`
 - `XYZ.BasisX`, `XYZ.BasisY`, `XYZ.BasisZ`, `XYZ.Zero`
(Basisvektoren und Nullvektor)

Revit API Grundlagen – Transform Klasse



TECHNISCHE
UNIVERSITÄT
DARMSTADT

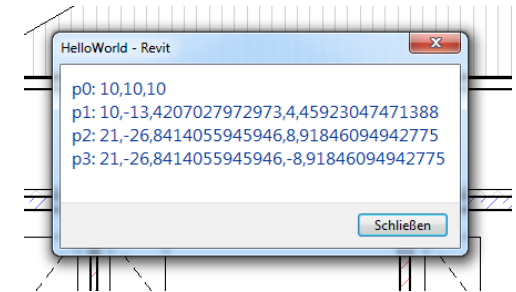
- Oft sind Koordinatentransformationen nötig, bspw. um Koordinaten aus einem lokalen in das globale Koordinatensystem zu überführen etc.
- Die Klasse Transform bietet hierfür verschiedene Möglichkeiten

```
public void transformExample()
{
    Transform trans1, trans2, trans3;
    XYZ ptOrigin = XYZ.Zero;
    XYZ ptXAxis = XYZ.BasisX;
    XYZ ptYAxis = XYZ.BasisY;

    // rotation
    trans1 = Transform.CreateRotation(ptXAxis, 90);
    // translation & scaling
    trans2 = Transform.CreateTranslation(ptXAxis).ScaleBasis(2.0);
    // mirror
    trans3 = Transform.CreateReflection(
        new Plane(ptXAxis, ptYAxis, ptOrigin));

    XYZ p0 = new XYZ(10, 10, 10);
    XYZ p1 = trans1.OfPoint(p0);
    XYZ p2 = trans2.OfPoint(p1);
    XYZ p3 = trans3.OfPoint(p2);

    TaskDialog.Show("Revit",
        "p0: " + p0.X + "," + p0.Y + "," + p0.Z
        + "\np1: " + p1.X + "," + p1.Y + "," + p1.Z
        + "\np2: " + p2.X + "," + p2.Y + "," + p2.Z
        + "\np3: " + p3.X + "," + p3.Y + "," + p3.Z);
}
```



Entwicklung einer Revit-Erweiterung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Vorbereitung



CommitStrip.com

1. Autodesk Revit 2019

- <http://www.autodesk.com/education/home>
(Bei Anmeldung: Verifizierung durch TU-Mailadresse)

2. Revit SDK 2019

- <https://www.autodesk.com/developer-network/platform-technologies/revit>

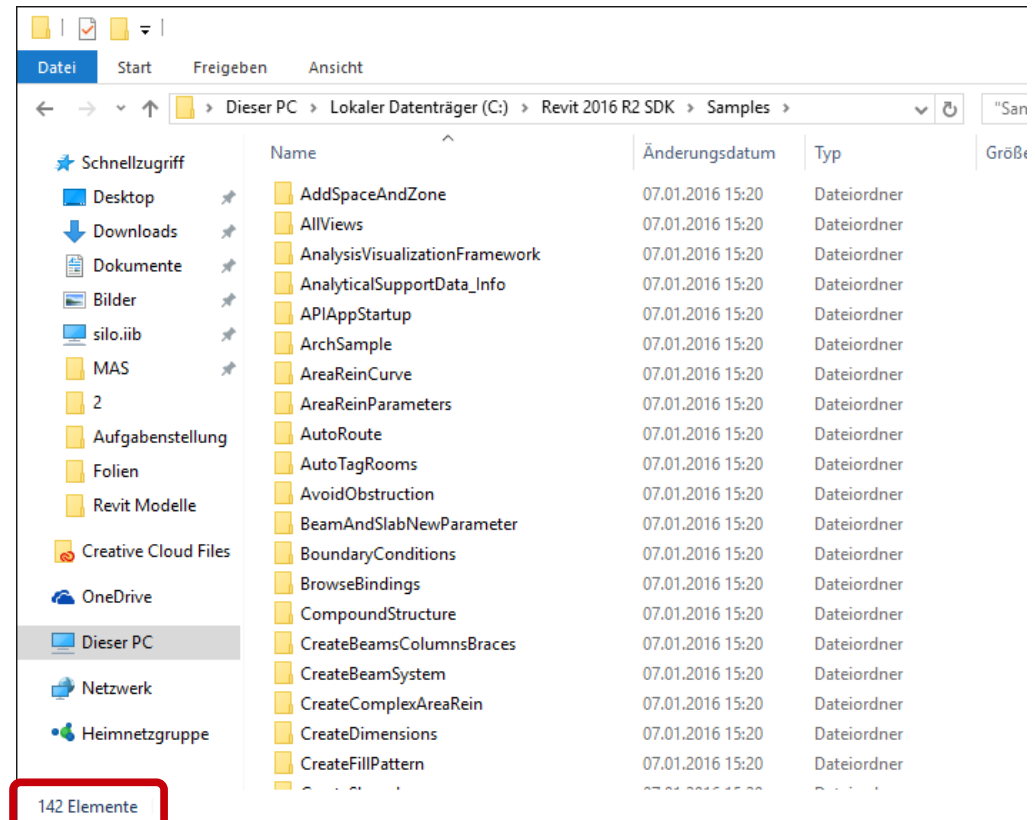
3. Revit Addin Manager (in Revit SDK Ordner)

4. Revit AddIn Lookup

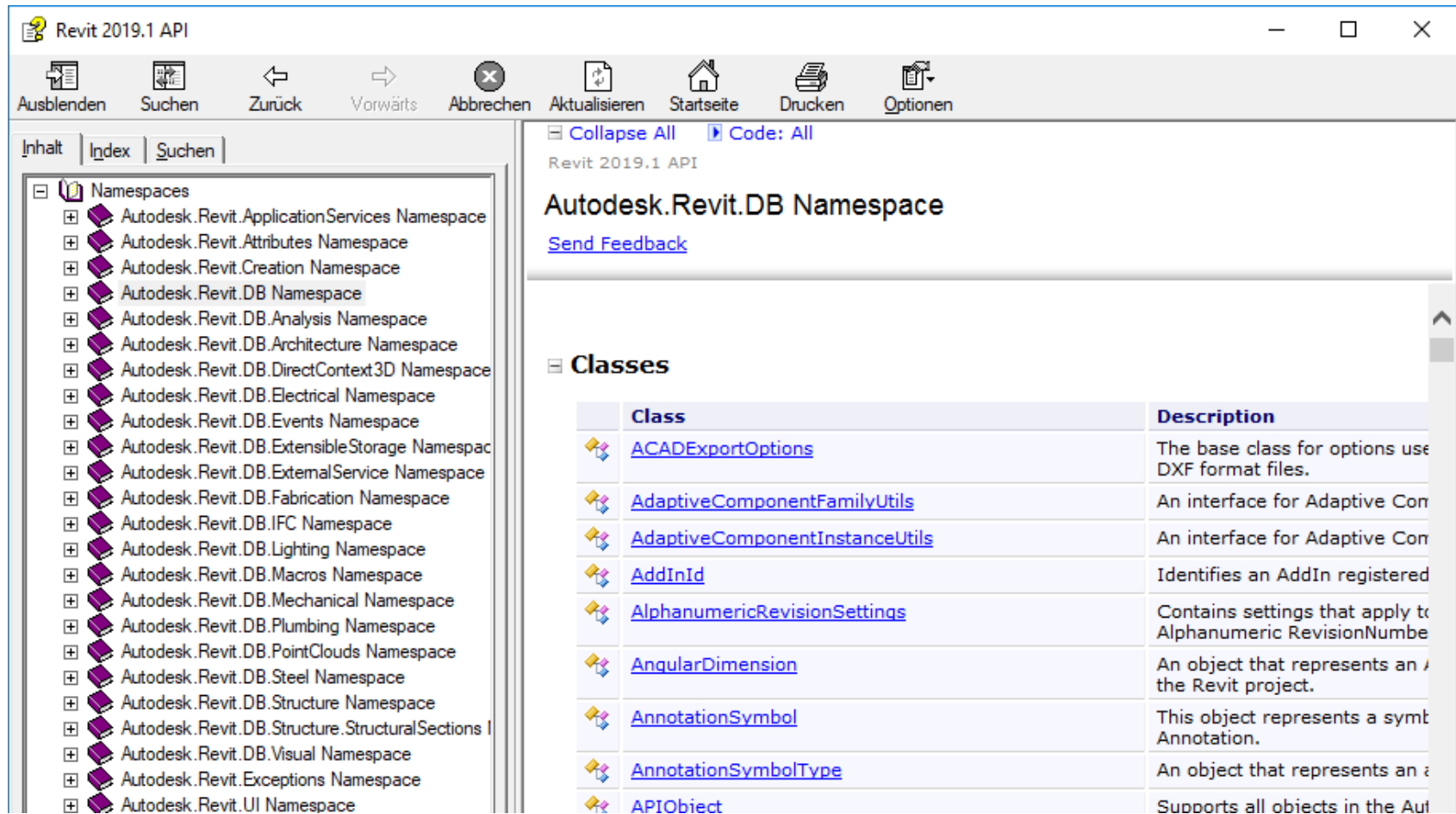
(In Revit 2019 ist Revit AddIn Lookup bei Default bereits installiert)

Revit SDK - Samples

1. In Visual Studio öffnen
2. Verweise aktualisieren
3. Kompilieren (Projektmappe Erstellen)
4. In Revit als Zusatzmodul laden



Revit SDK - Revit 2019 API.chm

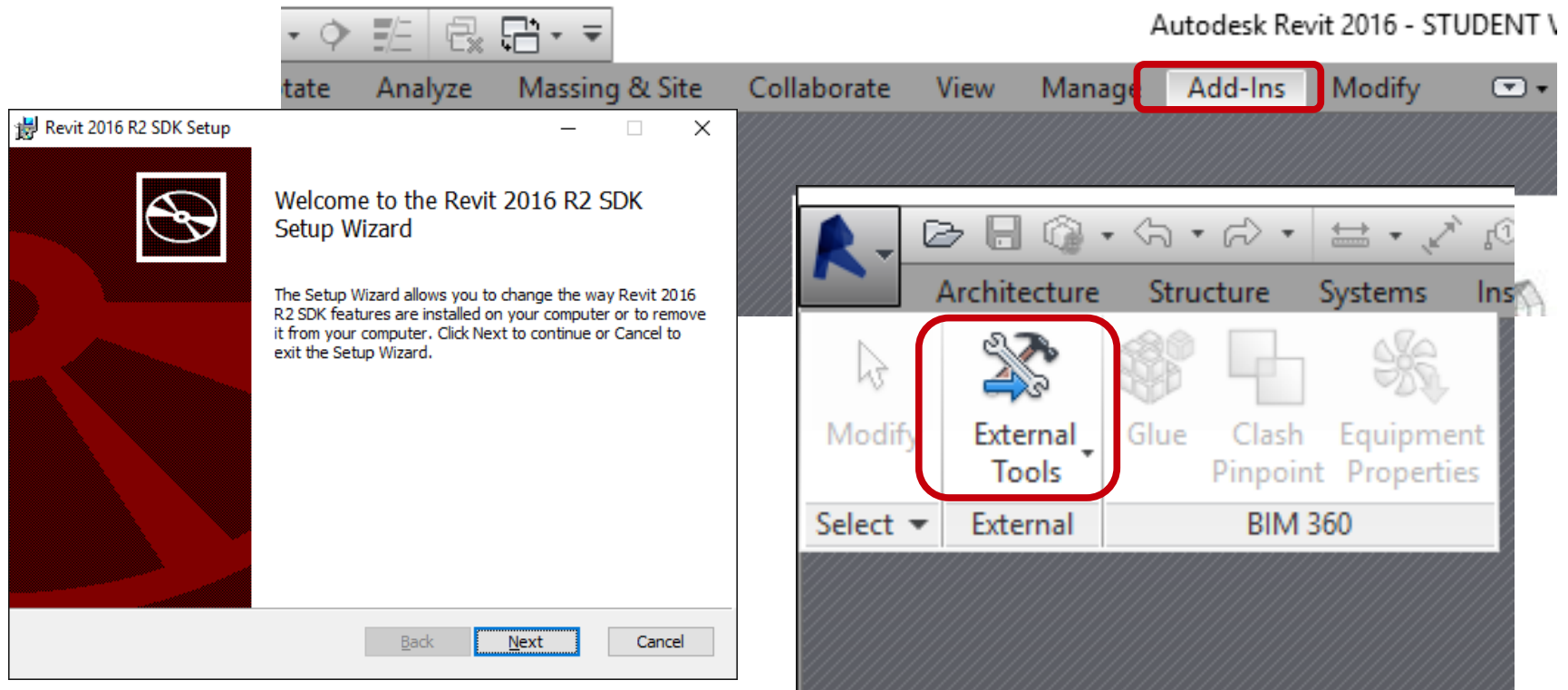


The screenshot shows the Revit 2019.1 API documentation window. The left sidebar contains a tree view of namespaces, with 'Autodesk.Revit.DB Namespace' selected. The main content area displays the 'Autodesk.Revit.DB Namespace' and a list of classes. The classes are listed in a table with columns for Class and Description.

Class	Description
ACADEExportOptions	The base class for options used to export DXF format files.
AdaptiveComponentFamilyUtils	An interface for Adaptive Component Family Utils.
AdaptiveComponentInstanceUtils	An interface for Adaptive Component Instance Utils.
AddInId	Identifies an AddIn registered with Revit.
AlphanumericRevisionSettings	Contains settings that apply to Alphanumeric RevisionNumbers.
AngularDimension	An object that represents an angular dimension in the Revit project.
AnnotationSymbol	This object represents a symbol for an annotation.
AnnotationSymbolType	An object that represents an annotation symbol type.
APIObject	Supports all objects in the Autodesk.Revit.DB namespace.

Revit Add-In Manager

enthalten im Revit 2019 SDK



Autodesk Add-In Manager installieren



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Windows (C:) > Autodesk > Revit_2019_G1_Win_64bit_Trial_wi_en-US > Revit 2019.1 SDK > Add-In Manager

Name	Änderungsdatum
AddInManager.dll	03.08.2018 19:
AddInManager.dll.config	03.08.2018 19:
AddInManagerHelpENU	03.08.2018 19:
Autodesk.AddInManager	16.01.2019 05:
SplitButton_License(BSD)	03.08.2018 19:

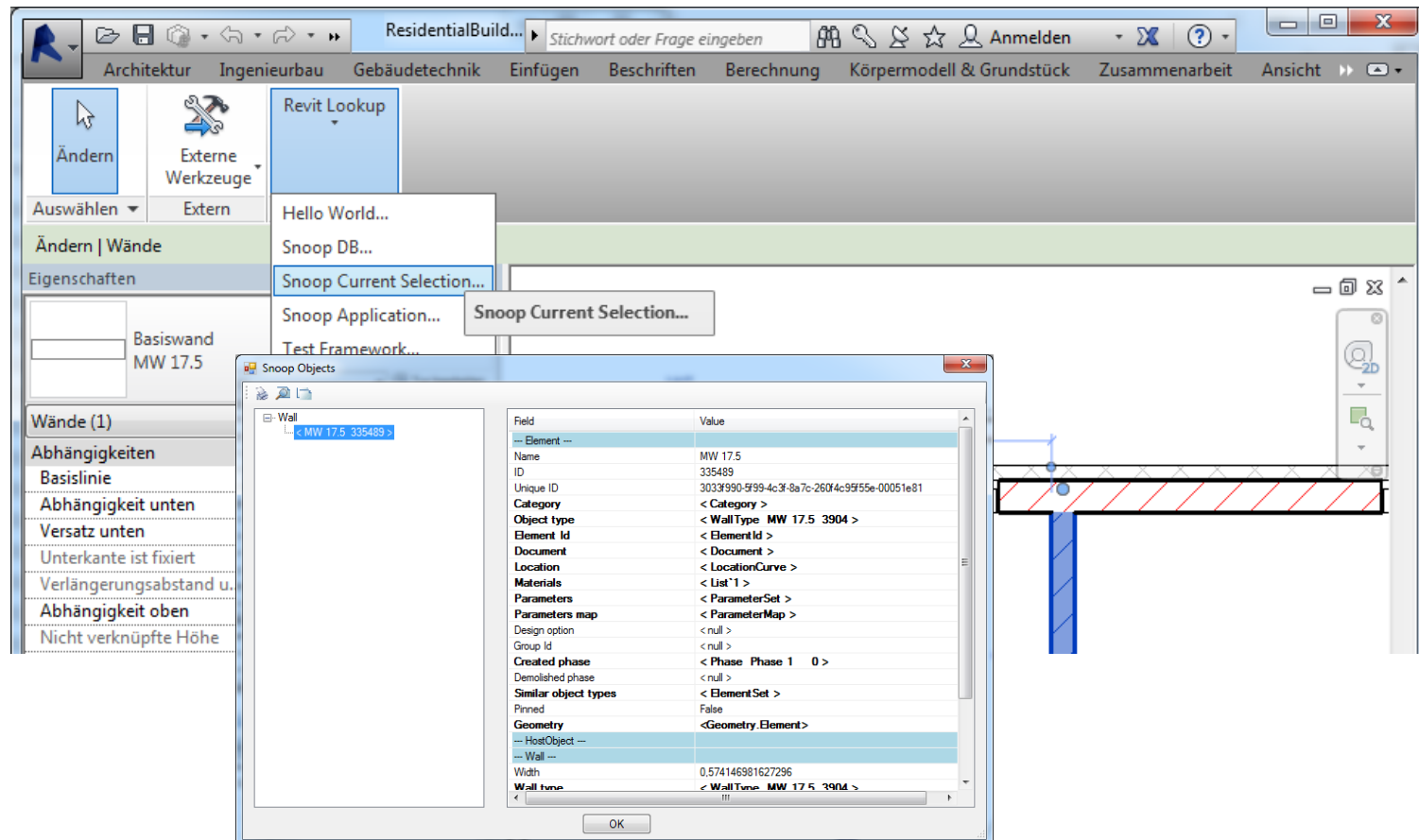
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RevitAddIns>
3   <AddIn Type="Command">
4     <Assembly>[TARGETDIR]AddInManager.dll</Assembly>
5     <ClientId>8C0A9E25-B7C5-421c-A1AB-702F73FA551F</ClientId>
6     <FullClassName>AddInManager.CAddInManager</FullClassName>
7     <Text>Add-In Manager (Manual Mode)</Text>
8     <VisibilityMode>AlwaysVisible</VisibilityMode>
9     <LanguageType>Unknown</LanguageType>
10    <VendorId>ADSK</VendorId>
11    <VendorDescription>Autodesk, www.autodesk.com</VendorDescription>
12  </AddIn>
13  <AddIn Type="Command">
14    <Assembly>[TARGETDIR]AddInManager.dll</Assembly>
15    <ClientId>6FDB8EC7-CCD3-4fc0-ADB7-B459D298FB93</ClientId>
16    <FullClassName>AddInManager.CAddInManagerFaceless</FullClassName>
17    <Text>Add-In Manager (Manual Mode, Faceless)</Text>
18    <VisibilityMode>AlwaysVisible</VisibilityMode>
19    <LanguageType>Unknown</LanguageType>
20    <VendorId>ADSK</VendorId>
21    <VendorDescription>Autodesk, www.autodesk.com</VendorDescription>
22  </AddIn>
23  <AddIn Type="Command">
24    <Assembly>[TARGETDIR]AddInManager.dll</Assembly>
25    <ClientId>91A2419C-5FCA-491A-BAA3-29A497EC07C7</ClientId>
26    <FullClassName>AddInManager.CAddInManagerReadOnly</FullClassName>
27    <Text>Add-In Manager (ReadOnly Mode)</Text>
28    <VisibilityMode>AlwaysVisible</VisibilityMode>
29    <LanguageType>Unknown</LanguageType>
30    <VendorId>ADSK</VendorId>
31    <VendorDescription>Autodesk, www.autodesk.com</VendorDescription>
32  </AddIn>
33 </RevitAddIns>
```

- Autodesk.AddInManager.addin
- Autodesk.AddInManager-Automatic.addin (wenn exists)

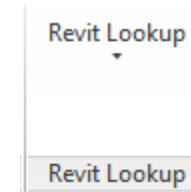
1. Bearbeiten:
[TARGETDIR] = Pfad zu Speicherort der AddInManager.dll
2. Beide Dateien kopieren nach:
C:\ProgramData\Autodesk\Revit\Addins\2019\

Informationen über Elemente erhalten

- Add-In „RevitLookup“ (<https://github.com/jeremytammik/RevitLookup>)



Revit Lookup

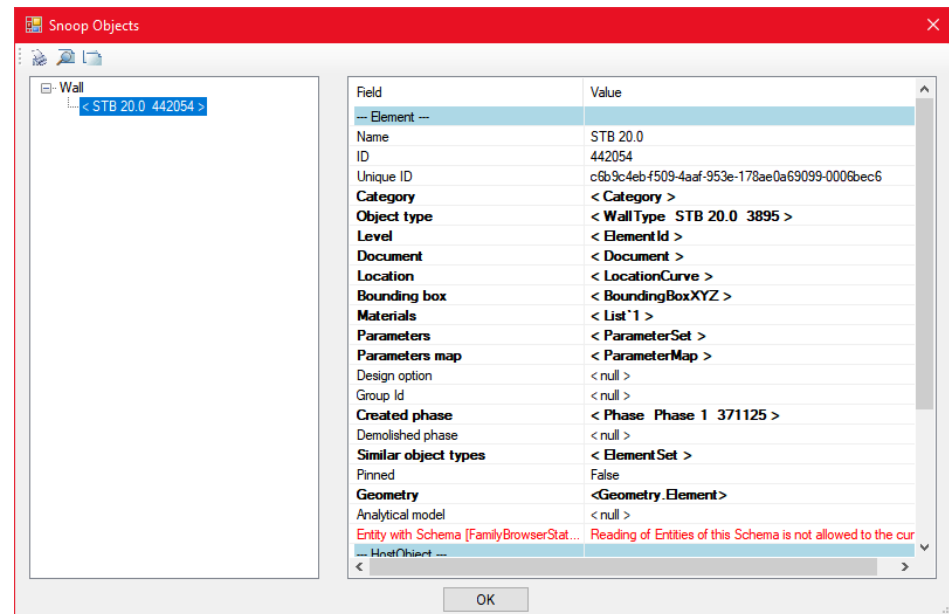


TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Teil des Revit SDK
 - AddIn-Manifest anpassen (Dateipfad zur DLL → bin → Debug)
 - **AddIn-Manifest in C:/ProgramData/Autodesk/Revit/Addins/2019 kopieren**

- Verschiedene Modi:
 - Snoop Current Selection
 - Snoop DB
 - Snoop Active View
 - ...

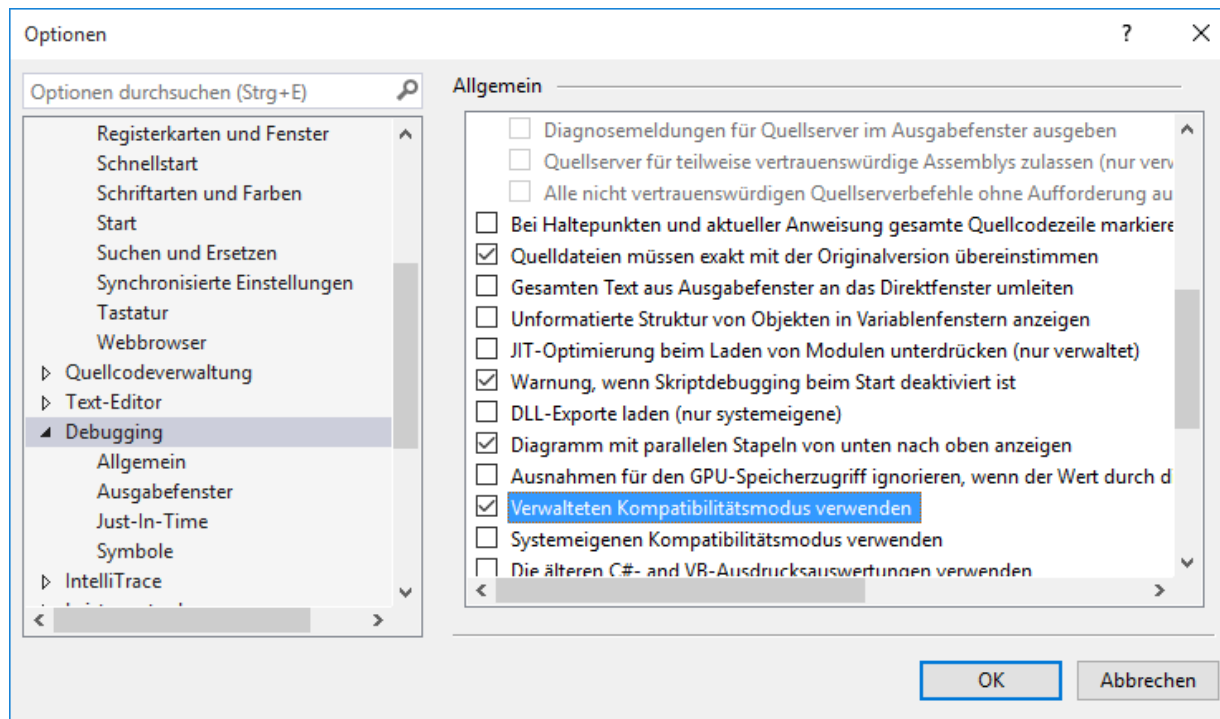
- Fett gedruckte Zeilen öffnen neues Fenster mit Feldern des ausgewählten Objekts bei Doppelklick



Hinweis bei Fehlermeldung (0xe06d7363)

Tritt die Fehlermeldung (0xe06d7363) auf, wie folgt vorgehen:

- Visual Studio → Extras → Optionen → Debugging → „Verwalteten Kompatibilitätsmodus verwenden“ aktivieren



Entwicklung einer Revit-Erweiterung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

2. Prozesse



99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.

127 little bugs in the code...

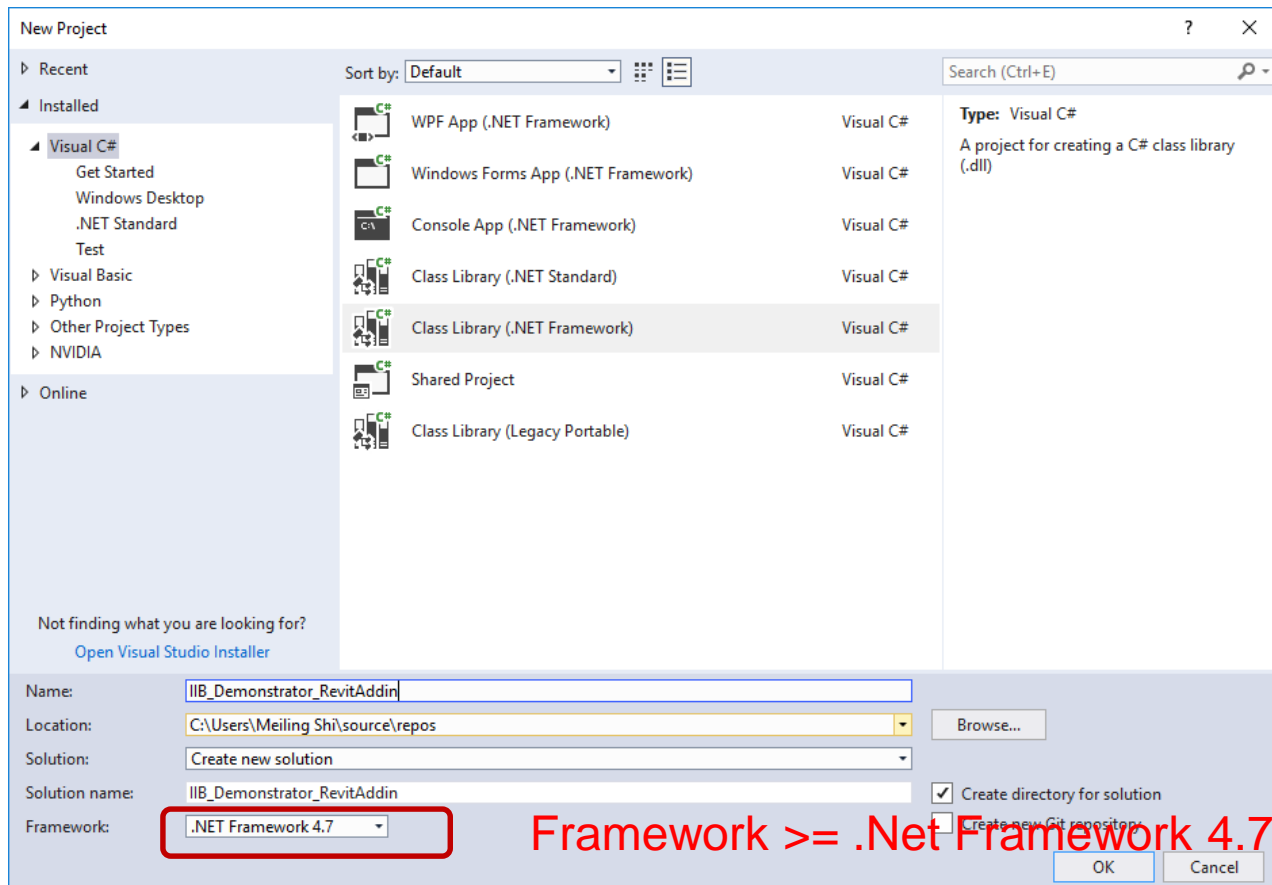


Anbindung Windows-Form Anwendung an Revit AddIn

- GUI Anwendung fertigstellen
- neue Klassenbibliothek erstellen
- Revit Verweise einfügen
- Projekteinstellungen anpassen
- File → Add → Existing Project
 - Projekt der GUI Anwendung auswählen
- Neue Referenz im AddIn Projekt anlegen
 - Projekte → Projekt der GUI Anwendung auswählen

Revit Add-In mit Visual Studio erstellen

- Neues C# Projekt als Klassenbibliothek erstellen



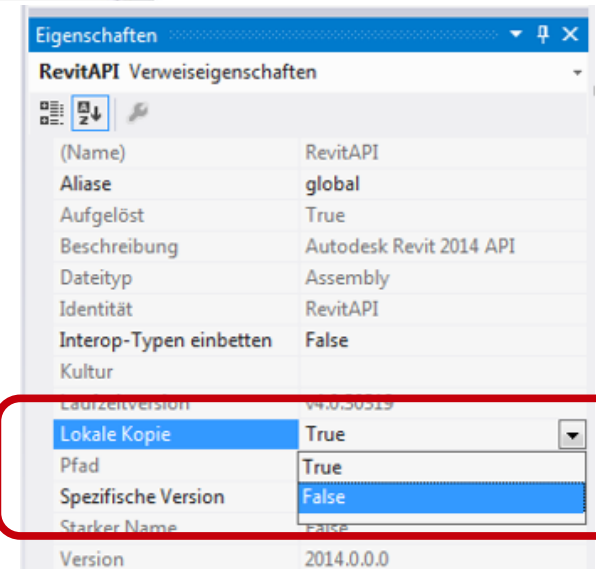
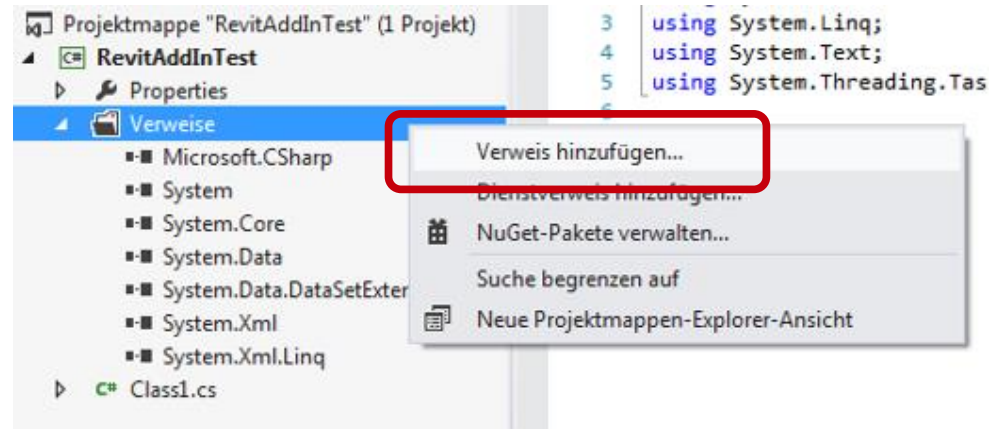
Revit Add-In mit Visual Studio erstellen

- Verweise hinzufügen

Im Stammverzeichnis von Revit:

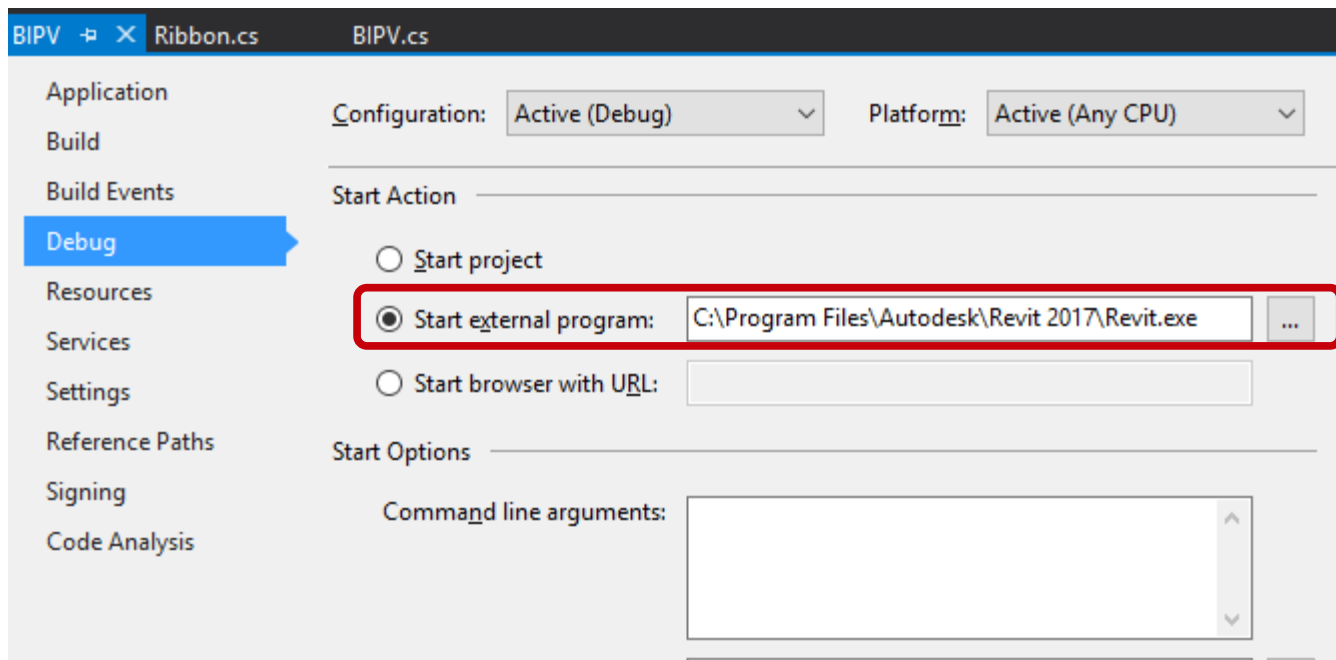
- **RevitAPI.dll**
- **RevitAPIUI.dll**

- Nach dem Hinzufügen, die Eigenschaft „**Lokale Kopie**“ auf „**False**“ setzen!
(für beide Bibliotheken)
- Aufgrund von Sicherheitseinstellungen können evtl. aus dem Internet bezogene DLLs nicht gestartet werden!



Revit Add-In mit Visual Studio erstellen

- In den Projekteigenschaften unter „Debuggen“ die Revit.exe als externes Programm zum Starten wählen



- **Möglichkeit 1:**

- Projekt-Eigenschaften: Debuggen → Externes Programm → Revit.exe
- Visual Studio: Klick auf → Debug → Revit startet
- In Revit Projekt öffnen + Add-In starten

- **Möglichkeit 2:**

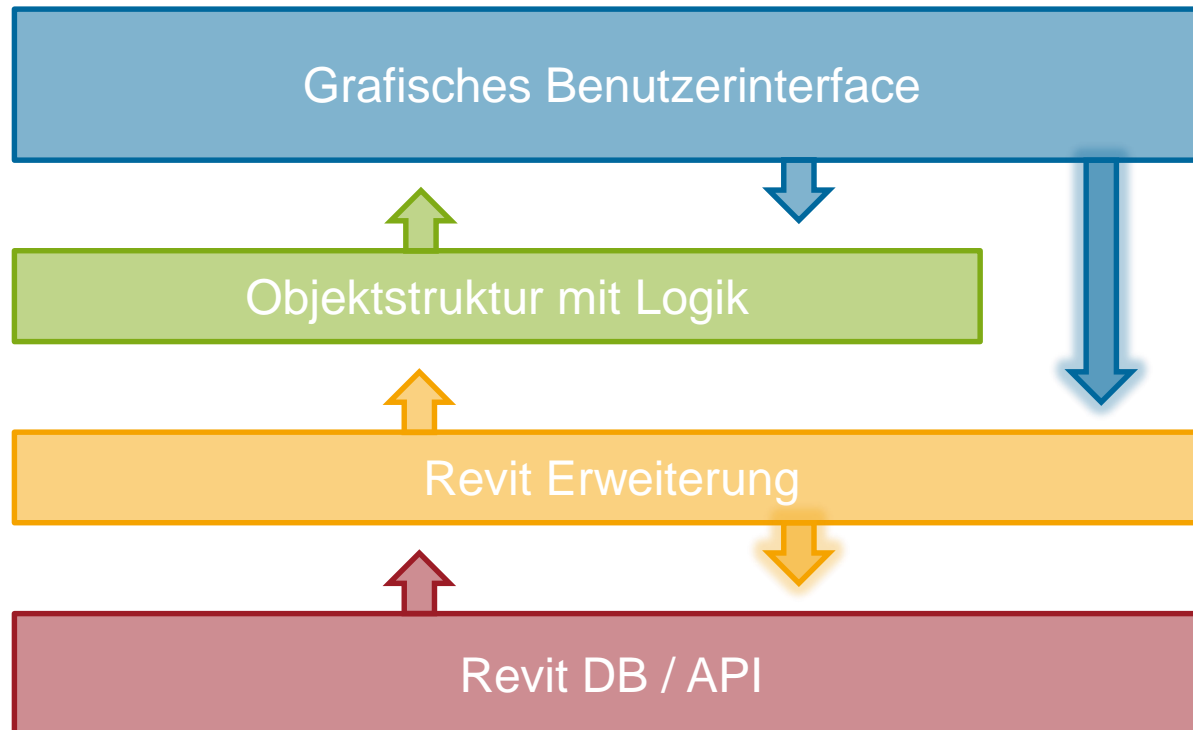
- Code kompilieren
- Revit starten
- ggf. .addin-Datei konfigurieren / .dll laden
- Visual Studio: Debug → Attach to process → Revit.exe
- Add-In starten

Revit API



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Information aus Revit lesen und Änderungen am Revit Document durchführen



- Übergabe der Änderungen in die Revit Erweiterung

- Veränderung der Datenbasis von Revit auf Basis der Nutzereingabe

- Ribbon erstellen
- Daten aus Revit Dokument auslesen
- Daten zurück in Revit schreiben
 - Änderungen in aus Revit gelesenen Objekten
- Neue Informationen in Revit übertragen
 - Ergebnisse von Berechnungen
- Objekte in Revit platzieren
 - Im Programm neu erstellte/zugewiesene Objekte
- Funktionserweiterung der Programme

Ribbon/Button erstellen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhalte:

- Typen von Revit-Erweiterungen
- AddIn Manifest Datei
- Ribbon und Button
- Auszuführende Commands

External Command:

- Können über den „External Tools“ Button aufgerufen werden
- Command wird bei Benutzereingabe ausgeführt (Execute()-Methode)

External Application:

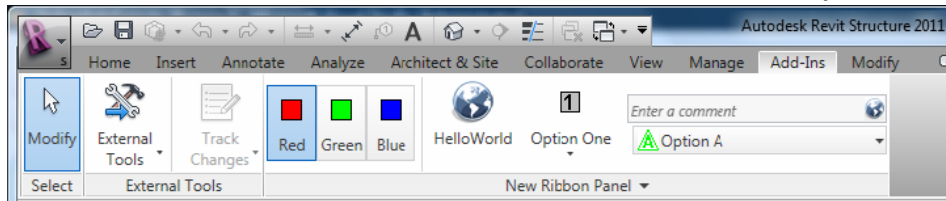
- Können mehrere External Commands verwalten
- Können eigene Tabs, Ribbons und Buttons in der Revit GUI haben
- Application wird beim Start von Revit geladen
 - u.a. Erzeugung von Tabs, Ribbons und Buttons
- Bei Betätigung eines Buttons wird in Application bestimmter Command ausgeführt

- **Autodesk.Revit.UI:**
 - Revits Grafisches Interface
- **Autodesk.Revit.DB:**
 - Zugriff auf Revit-interne Datenbank
- **Autodesk.Revit.DB.Element:**
 - Vorhandene Elemente in Revit
- **Autodesk.Revit.ApplicationServices:**
 - Zugang zu Applikations-Eigenschaften (Dokumente, Optionen, applikationsübergreifende Daten und Einstellungen)
- **Autodesk.Revit.Attributes:**
 - Bietet verschiedene Attribute zur Einstellung der Schnittstellen `IExternalCommand` (z.B. Transaction-Mode) und `IExternalApplication`

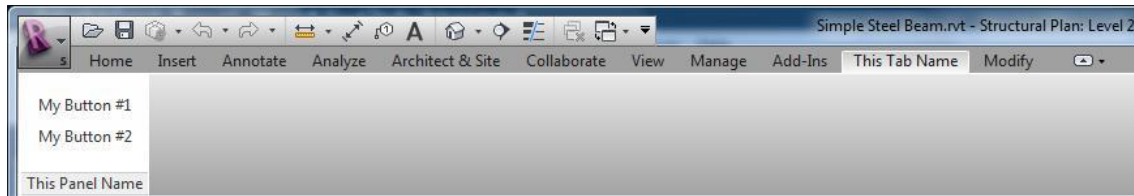
Ribbon Panels und Controls

Es ist möglich eigene UI-Elemente (Buttons, Panels, Tabs, etc.) zur Revit-Oberfläche hinzuzufügen

1. Variante: Controls zum Add-In Tab („Zusatzmodule“) hinzufügen



2. Variante: Eigenständiges Tab erstellen (max. 20 Tabs möglich)



Hierfür muss ein .addin Datei erstellt werden!

Für Abgabe notwendig!

Anleitungen:

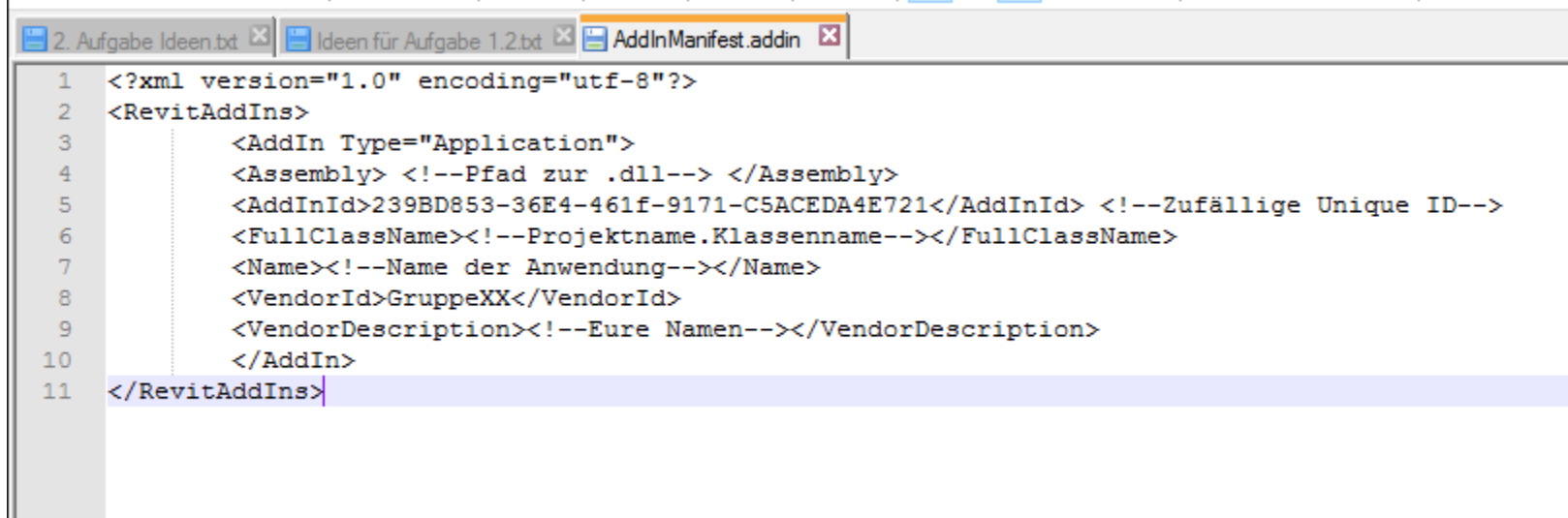
<http://help.autodesk.com/view/RVT/2014/DEU/?guid=GUID-1547E521-59BD-4819-A989-F5A238B9F2B3>

<http://thebuildingcoder.typepad.com/blog/2009/12/custom-ribbon-tab.html>

Aufbau der .addin Datei

- Muss in Ordner **C:\ProgramData\Autodesk\Revit\Addins\2019** abgelegt werden
- Alle in den .addin-Dateien enthaltenen Verweise werden beim Programmstart von Revit geladen

Aufbau einer .addin-Datei:



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RevitAddIns>
3     <AddIn Type="Application">
4         <Assembly> <!--Pfad zur .dll--> </Assembly>
5         <AddInId>239BD853-36E4-461f-9171-C5ACEDA4E721</AddInId> <!--Zufällige Unique ID-->
6         <FullClassName><!--Projektname.Klassenname--></FullClassName>
7         <Name><!--Name der Anwendung--></Name>
8         <VendorId>GruppeXX</VendorId>
9         <VendorDescription><!--Eure Namen--></VendorDescription>
10        </AddIn>
11 </RevitAddIns>
```

AddIn Manifest (1)

- Definiert eine Erweiterung
 - Dateipfad zur DLL und voller Klassenname (mit Namespace)
 - Bezeichnung der Anwendung (**Namenskonvention !**)
 - Urheber / Hersteller
 - **Eindeutige** ID für das AddIn (→ The Building Coder:
<http://thebuildingcoder.typepad.com/blog/2010/04/addin-manifest-and-guidize.html>
(s.o. plus Anleitung zur Erstellung einer neuen GUID)
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\Tools)
 - Es kann nicht mehrere Erweiterungen mit der selben GUID geben
- Wird bei Start von Revit ausgelesen und weiterverarbeitet
 - Commands werden „External Tools“-Button hinzugefügt
 - Applications werden ausgeführt

weitere Hilfe: <https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2017/ENU/Revit-API/files/GUID-7577712B-B09F-4585-BE0C-FF16A5078D29-htm.html> (Aufbau und weitere Infos zum Thema Manifest)

AddIn Manifest (2)

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RevitAddIns>
```

```
<AddIn Type="Application">
```

Erweiterungsart

```
<Name>IIB 1 Demonstrator</Name>
```

```
<Assembly>[DATEIPFAD]\IIB1_Demonstrator_AddIn.dll</Assembly>
```

```
<AddInId>c47e9a59-b5dd-40f5-a443-0bde9b0957cc</AddInId>
```

Muss unique sein

```
<FullClassName>IIB1_Demonstrator_AddIn.RevitAddIn</FullClassName>
```

```
<VendorId>Anna Wagner</VendorId>
```

Namespace.Klassenname

```
<VendorDescription>Informatik im Bauwesen 1</VendorDescription>
```

```
</AddIn>
```

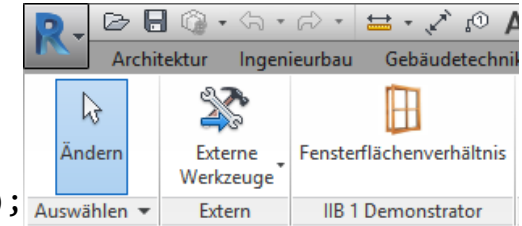
```
</RevitAddIns>
```

Ribbon erstellen

Erstellen eines neuen Ribbons:

```
String panelName = "IIB 1 Demonstrator";
```

```
RibbonPanel ribbonDemoPanel = application.CreateRibbonPanel(panelName);
```



Hinzufügen eines Buttons mit Bild:

```
PushButton myButton = (PushButton)ribbonDemoPanel.AddItem(new PushButtonData("IIB1Demo",  
    "Fensterflächenverhältnis", AddInPath, "IIB1_Demonstrator_AddIn.StartApp"));
```

Text des Buttons

namespace und Bezeichnung der Klasse, die aufgerufen werden soll

```
myButton.LargeImage = new BitmapImage(new Uri(Path.Combine(ButtonIconsFolder, "Fenster.png"),  
    UriKind.Absolute));
```

Hinzufügen eines ToolTips:

Text, der angezeigt wird

```
myButton.ToolTip = "Öffnet ein Tool, in dem das Fensterflächenverhältnis bestimmt wird.";
```

```
myButton.ToolTipImage = new BitmapImage(new Uri(Path.Combine(ButtonIconsFolder,  
    "panoramafenster.jpg"), UriKind.Absolute));
```

Bild, das angezeigt wird

Event für den Button hinzufügen

Neue Klasse vom Typ “IExternalCommand”

```
[Transaction(TransactionMode.Manual)]
public class StartApp : IExternalCommand
{
    public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
    {
        ...
        RevitAddIn.thisApp.ShowForm(meineRaeume);
        return Autodesk.Revit.UI.Result.Succeeded;
    }
}
```

Gleicher Name wie der beim
Button angegebene

Beim Ausführen werden zunächst alle
Räume & Fenster ausgelesen (...)

Anschließend wird in der ShowForm()-
Methode der RevitAddIn-Klasse das Form
gestartet

Zugriff auf Dokument



TECHNISCHE
UNIVERSITÄT
DARMSTADT

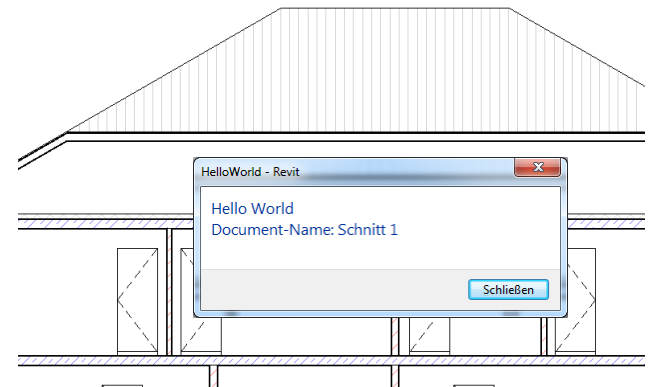
Inhalte:

- Zugriff auf Dokument Daten
- Elemente nach Category Filtern

Zugriff auf Document Daten

```
namespace RevitAddInTest
{
    [Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)]
    public class HelloWorld : IExternalCommand
    {
        public Autodesk.Revit.UI.Result Execute(ExternalCommandData revit,
            ref string message, ElementSet elements)
        {
            UIApplication uiApp = revit.Application;
            Document doc = uiApp.ActiveUIDocument.Document;

            TaskDialog.Show("Revit", "Hello World\nDocument-Name: " + doc.ActiveView.Name);
            return Autodesk.Revit.UI.Result.Succeeded;
        }
    }
}
```



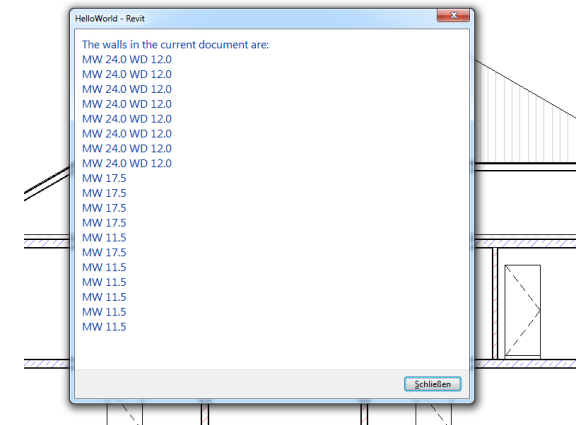
Elemente nach Category Filtern (Beispiel)

```
[Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)]
public class HelloWorld : IExternalCommand {
    public Autodesk.Revit.UI.Result Execute(ExternalCommandData revit,
        ref string message, ElementSet elements)
    {
        UIApplication uiApp = revit.Application;
        Document doc = uiApp.ActiveUIDocument.Document;

        // Find all Wall instances in the document by using category filter
        ElementCategoryFilter filter = new ElementCategoryFilter(BuiltInCategory.OST_Walls);
        FilteredElementCollector collector = new FilteredElementCollector(doc);
        IList<Element> walls = collector.WherePasses(filter).WhereElementIsNotElementType().ToElements();
        String prompt = "The walls in the current document are:\n";
        foreach (Element e in walls) { prompt += e.Name + "\n"; }

        TaskDialog.Show("Revit", prompt);

        return Autodesk.Revit.UI.Result.Succeeded;
    }
}
```



Filter:

- `ElementCategoryFilter(BuiltInCategory gesuchteKategorie)`
- `BoundingBoxIntersectsFilter(BoundingBoxXYZ geschnitteneBoundingBox)`
- `ElementIntersectsElementFilter(Element geschnittenesElement)`

Verknüpfung von Filtern:

- `LogicalAndFilter(Filter filter1, Filter filter2)`
- `LogicalOrFilter(Filter filter1, Filter filter2)`

Collector:

- `FilteredElementCollector(doc)`
 - `WhereElementIsNotElementType()`
 - `WhereElementIsViewIndependent()`
 - `WherePasses(Filter angewendeterFilter)`
 - `OfClass(Type gesuchteKlasse)`
 - `OfCategory(BuiltInCategory gesuchteKategorie)`

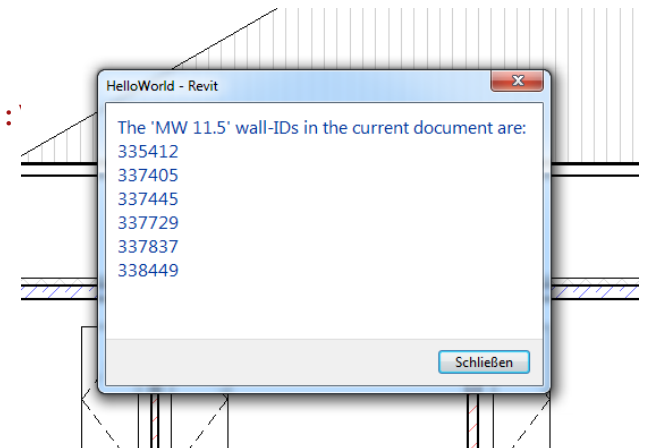
Language Integrated Query (LINQ)



```
public void linqExample(Document doc) {  
    // Find all Wall elements  
    ElementCategoryFilter filter = new ElementCategoryFilter(BuiltInCategory.OST_Walls);  
    // Apply the filter to the elements in the active document  
    FilteredElementCollector collector = new FilteredElementCollector(doc);  
    IList<Element> walls = collector.WherePasses(filter).WhereElementIsNotElementType().ToElements();
```

```
    // Use LINQ query to find walls with specific name  
    var query = from wall in walls  
                where wall.Name == "MW 11.5"  
                select wall;
```

```
    List<Wall> selectedWalls = query.Cast<Wall>().ToList<Wall>();  
    String prompt = "The 'MW 11.5' wall-IDs in the current document are:"  
    foreach (Wall e in selectedWalls)  
    {  
        prompt += e.Id + "\n";  
    }  
    TaskDialog.Show("Revit", prompt);  
}
```



Filtern aller Räume (Auszug Methode Execute(...))

```
UIApplication uiApp = commandData.Application;
UIDocument mdoc = uiApp.ActiveUIDocument;
Util.Doc = mdoc.Document;

List<Element> Rooms = new FilteredElementCollector(mdoc.Document).
    OfClass(typeof(SpatialElement)).WhereElementIsNotElementType().
    Where(room => room.GetType() == typeof(Room)).ToList();
Util.holeAlleFenster();
BindingList<Raum> raeume = new BindingList<Raum>();
foreach (Element e in Rooms)
{
    Raum r = Util.parseRaum((Room)e);
    if (r != null)
        meineRaeume.Add(r);
}
FormMain m = new FormMain(meineRaeume);
m.ShowDialog();
```

Auslesen aller Räume

Auslesen aller Fenster

Parsen des
betrachteten Raums

Parameter in Revit ändern



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhalte:

- Rechte in Revit
- Externe Events
- Transactions
- Parameter

Revit aus einem externen Programm bedienen

Problem:

Revit API erlaubt nur bedingt Zugriffe von externen Threads

- Transactions können nur aus einem validen Revit Kontext aufgerufen werden

Lösung:

- ExternalEvents – von einer externen Anwendung getriggertes Event
- Idling Event – Frequenz der Abrufe nicht kontrollierbar, kann die CPU auslasten

Thread: kleinster sequentieller Abarbeitungsstrang innerhalb eines Prozesses



- Von der RevitAPI gestellte Events
- Verwendung ähnlich normaler Events in Forms

Schritte:

1. Erstellen einer geschachtelten Klasse vom Typ `IExternalEventHandler` in `RevitAddIn.cs`
 - Vom Interface erforderte Methoden: `Execute(UIApplication app)` und `GetName()`
 - `Execute` führt gewünschte Funktionen aus
2. Erzeugen eines Events unter Verwendung des erstellten `EventHandlers`
3. Übergabe des Events an die aufgerufene Form
4. In der Form an gewünschter Stelle das Event mit der `Raise()` Funktion triggern

Weitere Infos:

<http://thebuildingcoder.typepad.com/blog/2015/12/external-event-and-10-year-forum-anniversary.html#3>

<http://thebuildingcoder.typepad.com/blog/2012/04/idling-enhancements-and-external-events.html>

SDK Samples - ModelessDialog

Externe Events (2) – Implementierung

Implementierung der Klasse in RevitAddIn.cs (geschachtelte Klasse):

```
public class RaumdatenUpdater : IExternalEventHandler
{
    public void Execute(UIApplication app)
    {
        Util.updateRaumDaten(demoForm.Raeume);
    }
    public string GetName()
    {
        return "RaumdatenUpdater";
    }
}
```

Ruft Util-Methode, die Änderungen
in Revit überträgt, auf

Externe Events (3) – Erzeugung Event(Handler)

Übergabe des Events:

```
public void ShowForm(BindingList<Raum> revitRaeume)
{
    if (demoForm == null || demoForm.IsDisposed)
    {
        RaumdatenUpdater updateHandler = new RaumdatenUpdater();
        ExternalEvent updateEvent = ExternalEvent.Create(updateHandler);
        demoForm = new FormMain(updateEvent, revitRaeume);
        demoForm.Show();
    }
}
```

Erzeugt Event und übergibt dieses dem Form im Konstruktor

Externe Events (4) – Aufrufen Event

In FormMain:

```
public FormMain(ExternalEvent update, BindingList<Raum> _raeume)
{
    this.ex_updateEvent = update;
    InitializeComponent();
    this.raeume = _raeume;
    fuelleListe();
}
```

Speichern des Events als globale
Variable

```
private void buttonUpdate_Click(object sender, EventArgs e)
{
    ex_updateEvent.Raise();
}
```

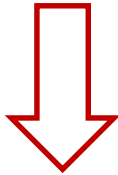
Triggert das Event in der
RevitAddIn-Klasse

- Veränderungen am aktiven Revit-Dokument können über Transactions vorgenommen werden (Zur Erinnerung: Revit-Modell = Datenbank)

- Eigenschaften / Methoden:
 - `Start()` → Startet Transaction-Kontext
 - `Commit()` → Beendet Transaction und übergibt alle Änderungen an das Dokument
 - `Rollback()` → Beendet Transaction und ignoriert alle Änderungen des Dokuments
 - `GetStatus()` → liefert aktuellen Status der Transaction (Uninitialized, Started, RolledBack, Committed, Pending)

```
Transaction trans = new Transaction(doc);  
trans.Start(„MyTransaction“);  
// Mache Veränderungen am Document  
trans.Commit();
```

besser



```
using (Transaction trans = new Transaction(doc))  
{  
    if (trans.Start("MyTransaction") == TransactionStatus.Started)  
    {  
        // Mache Veränderungen am Document  
        trans.Commit();  
    }  
}
```

ggf.



```
if (TransactionStatus.Committed != trans.Commit())  
{  
    TaskDialog.Show("Failure", "Transaction could not be committed");  
}
```

- Transactions immer so nah wie möglich an der Stelle im Code platzieren, wo die Veränderungen am Document vorgenommen werden!
- Transactions immer in **using**-Blöcken verwenden!

Properties in Revit ändern

Verwendete Methode: `Parameter.Set("Wert")`

```
room.Number = r.RaumNummer;
```

```
if (zugehörigeNutzungsart(r) != "")
```

```
    room.GetParameters(nutzungsart).First().Set(zugehörigeNutzungsart(r));
```

mit

Room room: betrachteter Raum – Durch RevitId von r aus Revit
Dokument auslesen

zugehörigeNutzungsart: Methode, die Raumklasse entsprechende
Nutzungsart nach DIN 277-2 zurückgibt

nutzungsart: Globale, finale Variable mit Bezeichnung des Parameters

→ *Für alle Räume des Gebäudes durchführen!*

- Nicht alle Parameter sind editierbar
- Fehler: „The parameter is read-only“
- Ursachen
 - Parameter setzt sich aus anderen Parametern zusammen
 - z.B. Raumname = Raumschlüssel + Raumnummer
 - Parameter wurde als nicht-editierbar bestimmt
- Einsehen über LookUp:
 - Parameterliste auswählen
 - Gewünschten Parameter auswählen

Field	Value
-- APIObject --	
Is read-only	True
-- Parameter --	

Platzieren von Objekten in Revit



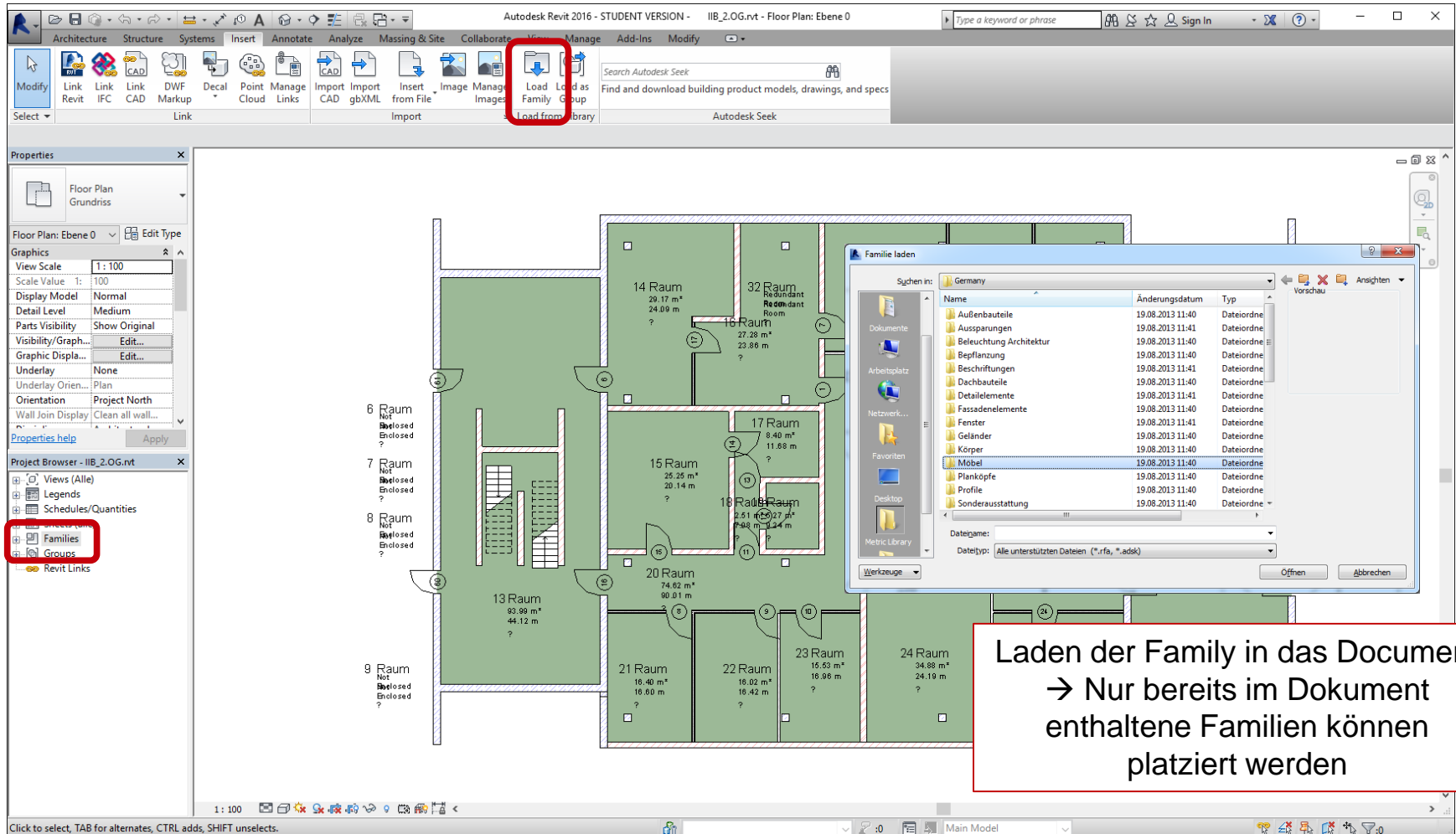
TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhalte:

- Revit Families
- Laden von Families (UI & API)
- Platzieren von Families

- Eine **Family** ist eine Gruppe/Kategorie von Elementen
 - Gemeinsame Eigenschaften (Parameter) und ähnliche grafische Repräsentation
- Verschiedene Elemente einer Familie können verschiedene Werte für ihre Parameter haben, man spricht deshalb von Family Types
- Family Types werden in der API durch die Klasse **FamilySymbol** repräsentiert
- Fügt man ein Element einer bestimmten Family und eines bestimmten Family Types (d.h. ein FamilySymbol) zum Document hinzu, so spricht man von einer **FamilyInstance**
 - Jede Family Instance hat ein Set von Eigenschaften, welche unabhängig von den Eigenschaften des Family Types geändert werden können. Derartige Änderungen betreffen ausschließlich dieses eine Element.
 - Änderungen an Eigenschaften des Family Types hingegen haben Einfluss auf alle Elemente dieses Typs im gesamten Projekt.

Load Family (UI)



Autodesk Revit 2016 - STUDENT VERSION - IIB_2.OG.rvt - Floor Plan: Ebene 0

Architecture Structure Systems Insert Annotate Analyze Massing & Site Collaborate Manage Add-Ins Modify

Search Autodesk Seek
Find and download building product models, drawings, and specs
Autodesk Seek

Properties

Floor Plan Grundriss

Floor Plan: Ebene 0 Edit Type

Graphics

View Scale 1:100

Scale Value 1:100

Display Model Normal

Detail Level Medium

Parts Visibility Show Original

Visibility/Graph... Edit...

Graphic Displa... Edit...

Underlay None

Underlay Orien... Plan

Orientation Project North

Wall Join Display Clean all wall...

Properties help Apply

Project Browser - IIB_2.OG.rvt

Views (Alle)

Legends

Schedules/Quantities

Families

Groups

Revit Links

14 Raum 29.17 m² 24.09 m

32 Raum Redundant Room

16 Raum 27.28 m² 23.86 m

17 Raum 8.40 m² 11.68 m

15 Raum 25.25 m² 20.14 m

18 Raum 2.51 m² 2.27 m

20 Raum 74.62 m² 90.01 m

13 Raum 93.99 m² 44.12 m

21 Raum 16.40 m² 16.60 m

22 Raum 16.02 m² 16.42 m

23 Raum 15.53 m² 16.96 m

24 Raum 34.88 m² 24.19 m

8 Raum Not Enclosed

7 Raum Not Enclosed

9 Raum Not Enclosed

Familie laden

Suchen in: Germany

Name	Änderungsdatum	Typ
Außenbauteile	19.08.2013 11:40	Dateiordne
Aussparungen	19.08.2013 11:41	Dateiordne
Beleuchtung Architektur	19.08.2013 11:40	Dateiordne
Bepflanzung	19.08.2013 11:40	Dateiordne
Beschriftungen	19.08.2013 11:41	Dateiordne
Dachbauteile	19.08.2013 11:41	Dateiordne
Detailelemente	19.08.2013 11:40	Dateiordne
Fassadenelemente	19.08.2013 11:40	Dateiordne
Fenster	19.08.2013 11:40	Dateiordne
Geländer	19.08.2013 11:40	Dateiordne
Körper	19.08.2013 11:40	Dateiordne
Möbel	19.08.2013 11:40	Dateiordne
Planköpfe	19.08.2013 11:40	Dateiordne
Profile	19.08.2013 11:40	Dateiordne
Sonderausstattung	19.08.2013 11:40	Dateiordne

Dateiname:

Dateityp: Alle unterstützten Dateien (*.rfa, *.adsk)

Offnen Abbrechen

1:100

Click to select, TAB for alternates, CTRL adds, SHIFT unselects.

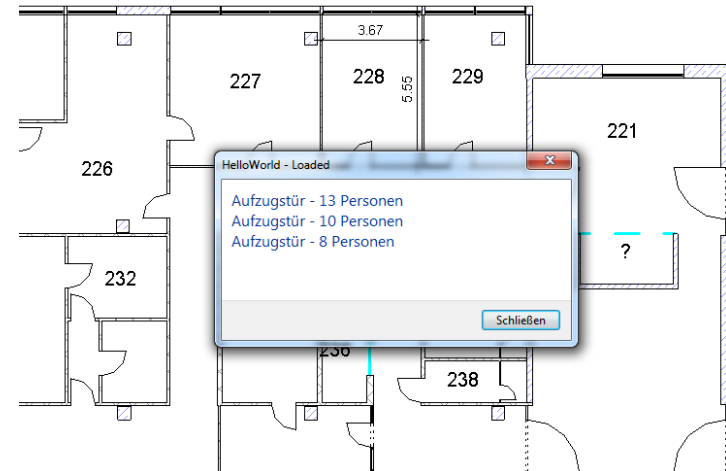
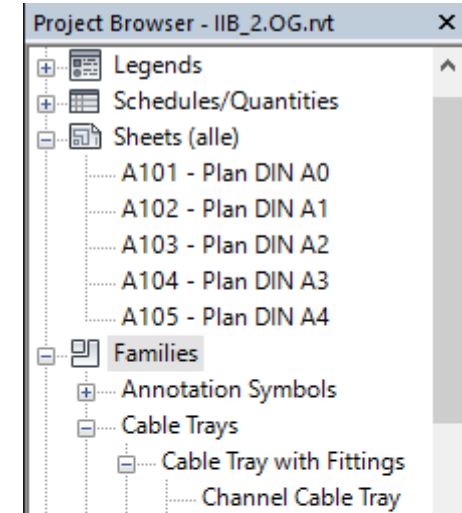
Main Model

Laden der Family in das Document
→ Nur bereits im Dokument
enthaltene Familien können
platziert werden

Revit API Grundlagen – Load Family (API)

```
public void loadFamilyExample(Document doc){  
    string fileName = @"C:\ProgramData\Autodesk\RVT 2016\Libraries\" +  
        "Germany\Sonderausstattung\Rolltreppen und Aufzüge\Aufzugstür.rfa";  
    Family family = null;  
    using (Transaction t = new Transaction(doc)) {  
        if (t.Start("LoadFamily") == TransactionStatus.Started) {  
            // try to load family  
            if (!doc.LoadFamily(fileName, out family)) {  
                throw new Exception("Unable to load " + fileName);  
            }  
            t.Commit();  
        }  
    }  
    // loop through family symbols  
    FamilySymbolSet familySymbolsId= family.GetFamilySymbolIds();  
    string symbolNames = "";  
    foreach (ElementId symbolId in familySymbolsId) {  
        symbolNames += family.Name + " - " + ((FamilySymbol)  
            family.Document.GetElement(symbolId)).Name + "\n";  
    }  
    TaskDialog.Show("Loaded", symbolNames);  
}
```

Laden der Family in das Document
via API & Dateipfad zur .rfa-Datei



FamilySymbol auswählen

Filterung aller Elemente des Dokumentes nach

- BuiltInCategory
- Klasse(FamilySymbol)
- Name

```
private static FamilySymbol GetFamilySymbolByName(BuiltInCategory bic, string name)
{
    return new FilteredElementCollector(doc).OfCategory(bic)
        .OfClass(typeof(FamilySymbol)).FirstOrDefault<Element>
        (e => e.Name.Equals(name)) as FamilySymbol;
}
```

Im Demonstrator – Lampe:

BuiltInCategory: OST_LightingFixtures

Name: " F - 1850 x 450 "

Platzierung der Lampen ermitteln

Abhängig von Platzierung der Raumbeschriftung

```
Room rr = doc.GetElement(r.RevitId) as Room;  
XYZ locR = ((LocationPoint) rr.Location).Point;
```

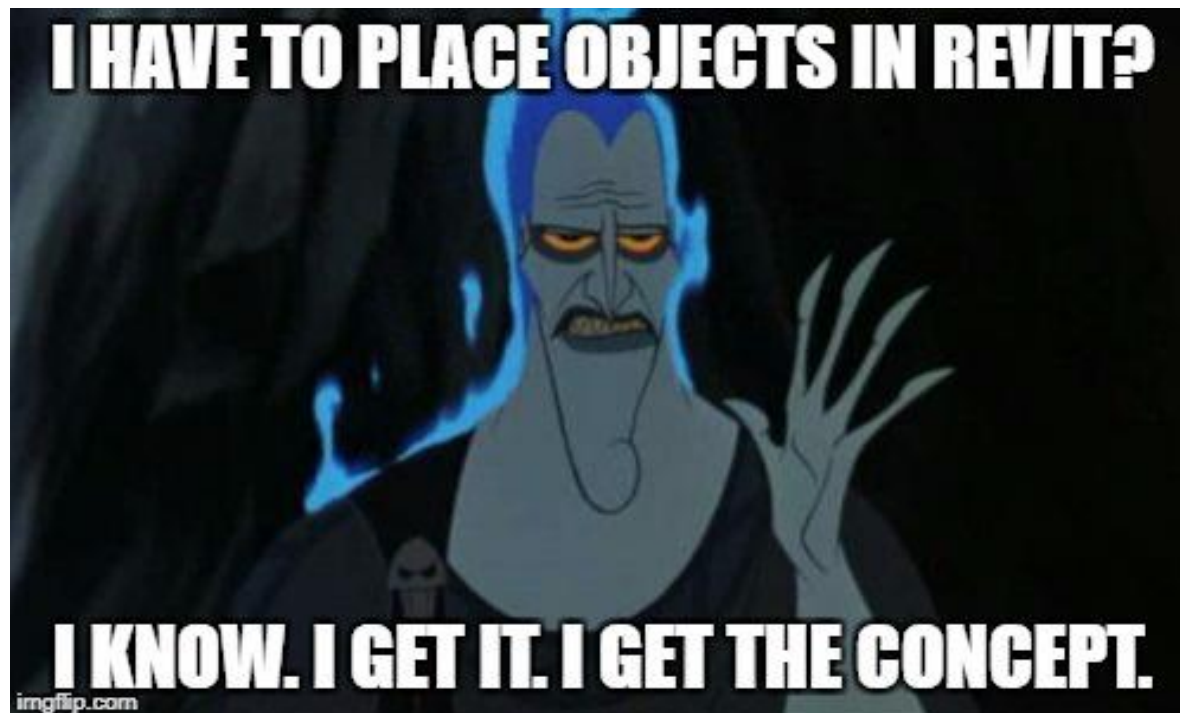
Erzeugung & Platzierung neue FamilyInstance

```
FamilyInstance fi = doc.Create.NewFamilyInstance(locR,  
    GetFamilySymbolByName(BuiltInCategory.OST_LightingFixtures,  
        "F - 1850 x 450") , StructuralType.NonStructural);
```

Weitere nützliche Funktionen

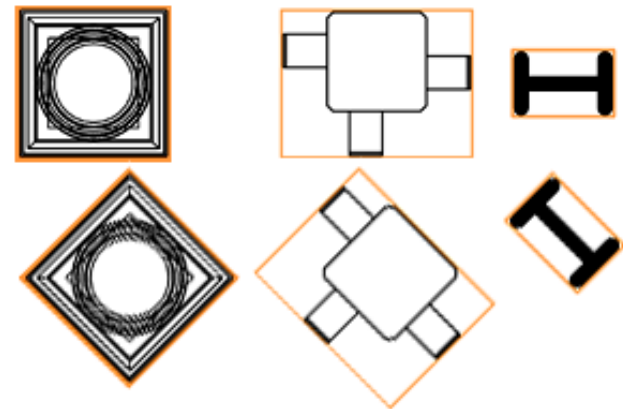
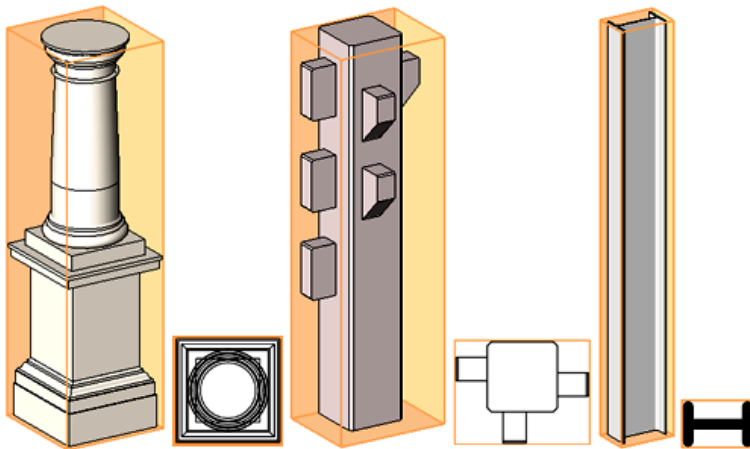


TECHNISCHE
UNIVERSITÄT
DARMSTADT



- Unsichtbarer 3D-Hüllkörper, der die minimalen und maximalen Ausdehnungen eines Körpers beschreibt

```
BoundingBoxXYZ b = element.get_BoundingBox(doc.ActiveView);  
XYZ max = b.Max;  
XYZ min = b.Min;
```



- Dient bei Benutzerauswahl als Filter
 - Erlaubt Auswahl der definierten Elemente

```
public class RaumFilter : ISelectionFilter
{
    Document doc = null;
    public RaumFilter(Document _doc)
    {
        doc = _doc;
    }

    bool ISelectionFilter.AllowElement(Element elem)
    {
        return elem is Room;
    }

    bool ISelectionFilter.AllowReference(Reference reference, XYZ position)
    {
        return true;
    }
}
```

Neue Parameter erstellen

- Neue Parameter in .txt Datei definieren
- Zum Start von Revit einlesen und anlegen
- Ausführlicher Blogeintrag und öffentliches Beispielprojekt von Jeremy Tammik (The Building Coder):
 - Building Coder Projekt:
<https://github.com/jeremytammik/PopulateMaterialProperty/blob/master/PopulateMaterialProperty/ExportParameters.cs>
 - Building Coder Blog-Eintrag:
<http://thebuildingcoder.typepad.com/blog/2009/06/model-group-shared-parameter.html>

CopyElement

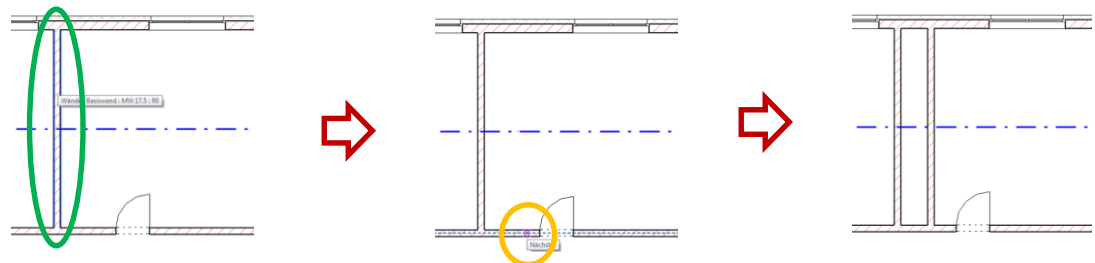
```
public void copyWallExample(ExternalCommandData revit)
{
    UIDocument uidoc = revit.Application.ActiveUIDocument;
    Document doc = uidoc.Document;
    Selection sel = uidoc.Selection;
```

Beispiel: Kopieren einer
ausgewählten Wand an
einen gewünschten Punkt

```
1 Reference picked = sel.PickObject(ObjectType.Element, new ElementSelectionFilter<Wall>(), "Pick a wall.");
   Wall wall = doc.GetElement(picked) as Wall;
```

```
2 XYZ point = revit.Application.ActiveUIDocument.Selection.PickPoint("Pick a point to place wall.");
```

```
using (Transaction trans = new Transaction(doc))
{
    if (trans.Start("CopyWall") == TransactionStatus.Started)
    {
        ElementTransformUtils.CopyElement(doc, wall.Id, point);
        trans.Commit();
    }
}
```

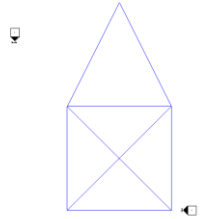


Auf eine Ebene zeichnen



```
public void drawOnPlane(Document doc)
{
    using (Transaction trans = new Transaction(doc))
    {
        if (trans.Start(„Draw“) == TransactionStatus.Started)
        {
            Plane planeXY = doc.Application.Create.NewPlane(new XYZ(100, 0, 0), new XYZ(0, 100, 0), XYZ.Zero);
            SketchPlane sketchPlane = SketchPlane.Create(doc, planeXY);

            ModelCurveArray lines = new ModelCurveArray();
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 0, 0), new XYZ(25, 50, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 50, 0), new XYZ(75, 50, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(75, 50, 0), new XYZ(75, 0, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(75, 0, 0), new XYZ(25, 0, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 0, 0), new XYZ(75, 50, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 50, 0), new XYZ(75, 0, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 50, 0), new XYZ(50, 100, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(75, 50, 0), new XYZ(50, 100, 0)), sketchPlane));
            trans.Commit();
        }
    }
}
```



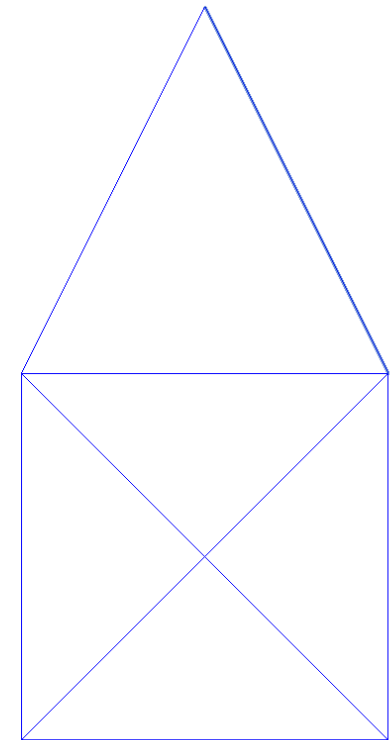
```
public void textNoteExample(Document doc, string text)
{
    using (Transaction trans = new Transaction(doc))
    {
        if (trans.Start(„Texting“) == TransactionStatus.Started)
        {
            TextNote textNote = doc.Create.NewTextNote(
                doc.ActiveView,
                new XYZ(-10,0,0),
                new XYZ(0,0,0),
                new XYZ(0,0,1),
                2.0,
                TextAlignFlags.TEF_ALIGN_CENTER,
                text
            );
            trans.Commit();
        }
    }
}

/* ... */
textNoteExample(doc, "Das ist das Haus vom Nikolaus :P");
```

Bei Änderungen am aktuellen Dokument sind immer Transactions erforderlich!



Das ist das Haus vom Nikolaus :P



Hilfreiche Foren / Internetadressen

- MSDN Library ([https://msdn.microsoft.com/de-de/library/w0x726c2\(v=vs.110\).aspx](https://msdn.microsoft.com/de-de/library/w0x726c2(v=vs.110).aspx))
- Stackoverflow (<http://stackoverflow.com/>)
 - Beachtet den grünen Haken!
- Jeremy Tammik's Blog (<http://thebuildingcoder.typepad.com/>)



October 27, 2016

AI, Edit and Continue



I am still in Munich supporting the one-week Forge accelerator workshop, returning back to Switzerland by train tonight. For ecological reasons, I prefer to avoid flying whenever I

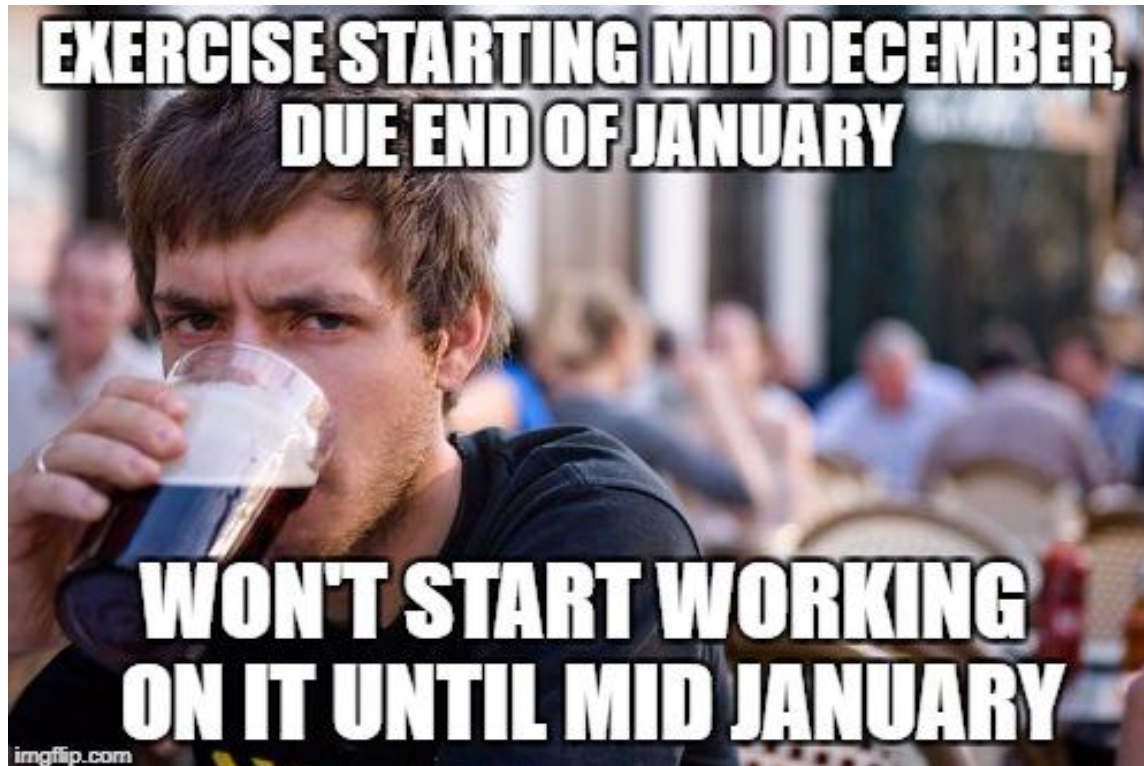


Hausübung 2



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorstellung der Aufgabenstellungen



Achtung!!! Namenskonvention



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Wieso?

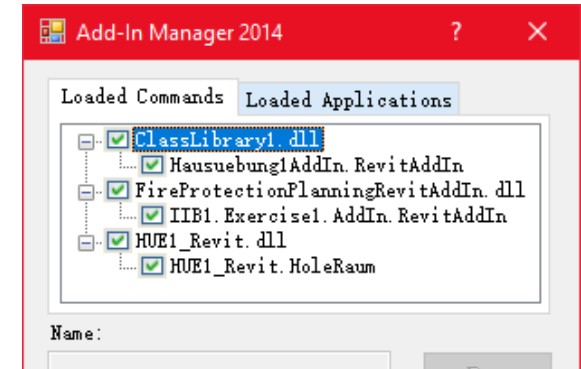
Einfachere Verwaltung mehrerer Abgaben

Projekte:

- IIB1_GruppeXX_Klassen
- IIB1_GruppeXX_GUI
- IIB1_GruppeXX_Revit

Command-Klasse / Ribbon:

- IIB1_GruppeXX



```
[Autodesk.Revit.Attributes.Transaction(Autodesk.I
public class IIB1_GruppeXX : IExternalCommand
{
    private static BindingList<Raum> meineRaeume

    public Result Execute(ExternalCommandData coi
    {
        try {

            UIApplication uiApp = commandData.Api
            UIDocument mdoc = uiApp.ActiveUIDocu
            Util.Doc = mdoc.Document;

            IList<Element> Rooms = new FilteredFl
```

Aufgabe 1- 3

Aufbauend auf Programm der 1. Blockübung

Übertragung der Änderungen in das Revit Dokument

- Auslesen Gebäudeinformation aus Revit-Modell
- Ändern von Parametern
- Hinzufügen neuer Parameter
- Platzieren von Familien

Aufgabe 4

Fehlerbuch Dokumentation (Bsp. Struktur)

- Wann entstanden / entdeckt
- Ursache
- Gab es schon ähnliche Fehler?
- War eine frühere Gegenmaßnahme erfolgreich?
 - Gegenmaßnahme ...
- Ausführliche Fehlerbeschreibung

Fehlerbuch

Lfd. Nr.: _____

Wann entstanden (Datum): _____

Wann entdeckt (Datum): _____

Programmname: _____

Ursache (Verhaltensmechanismus): _____

Aufgabe 5

Korrektur / Bewertung von UML Diagrammen

- Diagramme einer anderen Aufgabenvariante
- Verwendung von Formular (ab 16.10.2019 in Moodle zur Verfügung gestellt)
- Positive wie auch Negative Kritik

Viel Erfolg !

- Abgabe: 03.02.2018, 23:55 Uhr, Moodle

- Betreuungstermine (L5|01 – 222) :
 - Mo., 21.01.2019 10:00 – 17:00 Uhr
 - Fr., 25.01.2019 12:00 – 17:00 Uhr
 - Do., 31.01.2019 10:00 – 17:00 Uhr
 - Fr., 01.02.2019 12:00 – 17:00 Uhr

- Kolloquien:
 - KW 6 - 7

