Lecture Notes

Building Information Modeling with Revit Architecture

by Simon Greenwold March 2004; 2005–2007 version edits by David Driver

Contents

Preface

This document comprises a series of units (lecture notes with associated exercises in the Revit® Architecture building information modeler) that teach principles of the creation of a building information model (BIM) design. The actual training in Revit Architecture topics associated with the lectures must be done using other materials. This document is more conceptual than technical.

There are 20 units, which could easily be spread over a two-semester course or an intense one-semester course. Not all of them are equal in length or importance. The following table indicates how long each one might take to teach. Inevitably, it's the technical training that takes the most time, so that must be accounted for, but the conceptual material is presented first for each unit to give structure to the training work.

sem	wk	**Theory**	**Revit Architecture**
I.	1	CAD Versus BIM (long)	Introduction, Interface, and Sketching
I.	2	Objects (long)	Walls, Floors, and Ceilings
I.	3	Families and Nested Families (short)	Editing Types
I.	4	Parameters (long)	Dimensions, Doors, and Windows
I.	5	Representations (medium)	Views, Visibility, and Sheets
I.	6	Design Constraints (short)	Levels, Reference Planes, and Grids
I.	7	Design Information Organization (medium)	Components, Categories, Subcategories
I.	8	Domain-Specific Knowledge (medium)	Roofs
I.	9	Delaying Specificity (short)	Massing
I.	10	Component Design (medium)	Family Editor
I.	11	Propagation of Constraints (medium)	Alignment, Locking, and Constraints
II.	12	Interdependencies (short)	Site
II.	13	Unique Form (short)	In-Place Families
II.	14	Detail (short)	Drafting and Linework
II.	15	Sequence (short)	Design as You Would Build
II.	16	Is Architecture Engineering? (short)	Formulas
II.	17	Databases (short)	Databases
II.	18	Things You Won't Have to Do in School (short)	Tags, Schedules and Legends
II.	19	Time (short)	Walkthroughs and Phasing
II.	20	Variation (short)	Options

Organization

In general, each unit is broken down into the following parts:

- Theory
- Revit Architecture
    - The Connection
    - Features to Learn
    - Notes
    - Hands-on Topics
    - Questions

At the end of each unit is a list of topics for hands-on technical training; in some cases these reference the companion Student Workbook. In other cases these are just areas of functionality the instructor may use to illustrate the concepts and theory discussed in the lecture notes.

These units do not themselves constitute a syllabus, but they could become the framework for one. You can use them to supplement an existing curriculum, or as the seed for a new one. Thinking of them as lecture notes is helpful. What the lectures comprise is up to you.

Introduction

Building information modeling is a process that fundamentally changes the role of computation in architectural design. It means that rather than using a computer to help produce a series of drawings and schedules that together describe a building; you use the computer to produce a single, unified representation of the building so complete that it can generate all necessary documentation. The primitives from which you compose these models are not the same ones used in CAD (points, lines, curves). Instead you model with building components such as walls, doors, windows, ceilings, and roofs. The software you use to do this recognizes the form and behavior of these components, so it can ease much of the tedium of their manipulation. Walls, for instance, join and miter automatically, connecting structure layers to structure layers, and finish layers to finish layers.

Many of the advantages are obvious—for instance, changes made in elevation propagate automatically to every plan, section, callout, and rendering of the project. Other advantages are subtler and take some investigation to discover. The manipulation of parametric relationships to model coarsely and then refine is a technique that has more than one career’s worth of depth to plumb.

BIM design marks a fundamental advance in computer-aided design. As the tools improve, ideas spread, and students become versed in the principles, it is inevitable that just as traditional CAD has secured a deserved place in every office, so will BIM design.

Unit 1

This unit discusses the differences between CAD and building information modeling, and introduces the Revit Architecture interface as well as sketching in Revit Architecture.

Theory: CAD Versus Building Information Modeling

Modeling Is Not CAD

BIM (building information modeling) is entirely unlike the CAD (computer-aided drafting) tools that emerged over the past 50 years and are still in wide use in today's architectural profession. BIM methodologies, motivations, and principles represent a shift away from the kind of assisted drafting systems that CAD offers.

To arrive at a working definition of building information modeling first requires an examination of the basic principles and assumptions behind this type of tool.

Why Draw Anything Twice?

You draw things multiple times for a variety of reasons. In the process of design refinement, you may want to use an old design as a template for a new one. You are always required to draw things multiple times to see them in different representations. Drawing a door in plan does not automatically place a door in your section. So the traditional CAD program requires that you draw the same door several times.

Why Not Draw Anything Twice?

There is more to the idea of not drawing anything than just saving time in your initial work of design representation. Suppose you have drawn a door in plan and have added that same door in two sections and one elevation. Now should you decide to move that door, you suddenly need to find every other representation of that door and change its location too. In a complicated set of drawings, the likelihood that you can find all the instances of that door the first time is slim unless you are using a good reference tracking system.

Reference and Representation

But doesn't reference tracking sound like something a computer ought to be good at? In fact, that's one of about three fundamental things a computer does at all. And it is exceedingly good at it. The basic principle of BIM design is that you are designing not by describing the forms of objects in a specific representation, but by placing a reference to an object into a holistic model. When you place a door into your plan, it automatically appears in any section or elevation or other plan in which it ought to be visible. When you move it, all these views update because there is only one instance of that door in the model. But there may be many representations of it.

You Are Not Making Drawings

That means that as you create a model in modeling software, you are not making a drawing. You are asked to specify only as much geometry as is necessary to locate and describe the building components you place into the model. For instance, once a wall exists, to place a door into it, you need to specify only the type of door it is (which automatically determines all its internal geometry) and how far along the wall it is to be placed. That same information places a door in as many drawings as there are. No further specification work is required.

That means that the act of placing a door into a model is not at all like drawing a door in plan or elevation, or even modeling it in 3D. You make no solids, draw no lines. You simply choose the type of door from a list and select the location in a wall. You don't draw it. A drawing is an artifact that can be automatically generated from the superior description you are making.

Even More Than a 3D Model

You are making a building information model—a full description of a building. This should not be confused with making a full 3D model of a building. A 3D model is just another representation of a building model with the same incompleteness as a plan or section. A full 3D model can be cut to reveal the basic outlines for sections and plans, but there are drawing conventions associated with these representations that cannot be captured this way. How will the swing line for a door be encoded into a 3D model? For a system to intelligently place a door swing into a plan but not into a 3D model, you need a high-level description of the door building model elements separate from the 3D description of their form. This is the model in BIM design.

Encoding of Design Intent

This model encodes more than form; it encodes high-level design intent. Within the model, walls and roofs are modeled not as a series of 3D solids, but as walls and roofs. That way if a level changes height, both of the objects automatically adjust to the new criteria. If the wall moves, any roof that has a relationship to that wall adjusts automatically.

Specification of Relationships and Behavior

As the design changes, the modeling software attempts to maintain design intent. The model implicitly encodes the behavior necessary to keep all relationships relative as the design evolves. Therefore the modeler is required to specify enough information that the system can apply the best changes to maintain design intent. When you move an object, it is placed at a location relative to specific data (often a floor level). When this data moves, the object moves with it. This kind of relativity information is not necessary to add to CAD models, which are brittle to change.

Objects and Parameters

You may be troubled by the idea that the only doors you are allowed to place into a wall are the ones that appear on a predefined list. Doesn't this limit the range of possible doors? To allow variability in objects, they are created with a set of parameters that can take on arbitrary values. If you want to create a door that is 9 feet high, it is only necessary to modify the height parameter of an existing door. Every object has parameters—doors, windows, walls, ceilings, roofs, floors, even drawings themselves. Some have fixed values, and some are modifiable. In advanced modeling you also learn how to create custom object types with parameters of your choosing.

How Do BIM Tools Differ from CAD Tools?

Clearly, because modeling is different from CAD, you are obliged to learn and use different tools.

Modeling tools don't offer such low-level geometry options. As a general rule modeling deals with higher-level operations than CAD does. You are placing and modifying entire objects rather than drawing and modifying sets of lines and points. Occasionally, you must do this in BIM, but not frequently. Consequently, the geometry is generated from the model and is therefore not open to direct manipulation.

Modeling tools are frustrating to people who really need CAD tools. For users who are not skilled modelers, modeling can feel like a loss of control. This is much the same argument stick-shift car drivers make about control and feel for the road. But automatic transmission lets you eat a sandwich and drive, so the choice is yours. There are also ways to layer on low-level geometric control as a postmodeling operation, so you can regain control without destroying all the benefits of a full building model.

Modeling entails a great deal of domain-specific knowledge. Many of the operations in the creation of a building information model have semantic content that comes directly from the architectural domain. The list of default door types is taken from a survey of the field. Whereas CAD gets its power from being entirely syntactical and agnostic to design intent, BIM design is the opposite. When you place a component in a model, you must tell the model what it is, not what it looks like.

Or else requires you to build it in yourself: Adding custom features and components to a BIM design is possible but requires more effort to specify than it does in CAD. Not only must geometry be specified, but also the meanings and relationships inherent in the geometry.

Is Modeling Always Better Than CAD?

As in anything, there are trade-offs.

A model requires much more information: A model comprises a great deal more information than CAD drafting. This information must come from somewhere. As you work in a modeling tool, it makes a huge number of simplifying assumptions to add all the necessary model information. For instance, as you lay down walls, they are all initially the same height. You can change these default values before object creation or later, but there is a near infinitude of parameters and possible values, so the program makes a great many assumptions as you work. The same thing happens whenever you read a sketch, in fact. That sketch does not contain enough information to fully determine a building. The viewer fills in the rest according to tacit assumptions.

Flouting of convention makes for tough modeling. This method works well when the building being modeled accords reasonably well with the assumptions the modeler is making. For instance, if the modeler assumes that walls do not cant in or out but instead go straight up and down, that means that vertical angle does not need to be specified at the time of modeling. But if the designer wants tilted walls, it’s going to require more work—potentially more work than it would to create these forms in a CAD program. It is even possible that the internal model that the software maintains does not have the flexibility to represent what the designer has in mind. Tilted walls may not even be representable in a given piece of software. Then a workaround is required that is a compromise at best. Therefore unique designs are difficult to model.

Whereas CAD doesn’t care. In CAD, geometry is geometry. CAD doesn’t care what is or isn’t a wall. You are still bound by the geometric limitations of the software (some CAD software supports non-uniform rational b-splines [NURBS] curves and surfaces, and others do not, for instance), but for the most part there is always a way to construct arbitrary forms if you want.

Modeling can help project coordination. Having a single unified description of a building can help coordinate a project. Because drawings cannot ever get out of sync, there is no need for concern that updates have not reached a certain party.

A single model could drive the whole building lifecycle. Increasingly, there is interest in the architectural community for designing the entire lifecycle of a building, which lasts much longer than the design and construction phases. A full building description is invaluable for such design. Energy use can be calculated from the building model, or security practices can be prototyped, for instance. Building systems can be made aware of their context in the whole structure.

BIM potentially expands the role of the designer. Clearly, this has implications for the role of architects. They may become the designers of more than the building form, but also specifiers of use patterns and building services.

Modeling may not save time as it is being learned. It is likely that while designers are learning to model rather than to draft, the technique will not save time in an office. That is to be expected. The same is true of CAD. Switching offices from hand drafting to CAD occurred only as students became trained in CAD and did not therefore have to learn the techniques on the job. The same is likely to be true of BIM design. As more students familiar with BIM design enter the job market, offices are more willing to switch to this new work method as they have larger pool of resources to draw from.

Potential hazards exist in BIM that do not exist in CAD. Because design intelligence is embedded into a model, it is equally possible to embed design stupidity.

Improperly structured models that look fine can be unusable. It is possible to make a model that looks fine but is created in such a way that it is essentially unusable. For instance, it may be possible to create something that looks like a window out of a collection of extremely tiny walls. But then the program's rules for the behavior of windows would be wasted. Further, its rules for the behavior of walls would cause it to do the wrong things when the design changed.

What Is an Engineering Technique Doing in Architecture?

BIM design comes from engineering techniques that have been refined for many years. Many forces are acting together to bring engineering methodologies like BIM into architecture. First, the computing power and the basic ability to use computers have become commonly available. Second, efficiencies of time and money are increasingly part of an architect's concern. BIM offers a possible edge in efficiency of design and construction.

Revit Architecture: Introduction, Interface, and Sketching

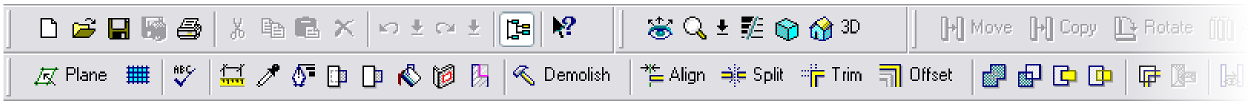
The Connection

This topic introduces the basics of BIM design as implemented in Revit Architecture. You will be asked to place a few components and modify their properties. It helps explain why the environment can seem foreign to a CAD user.

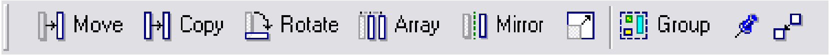
Revit Architecture Features and Concepts to Learn

Basic interface

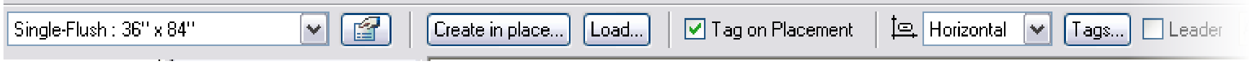
- Toolbars



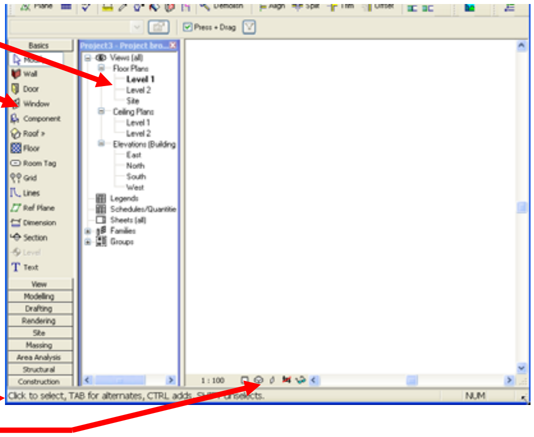
- Edit tools



- Options bar



Single-Flush : 36" x 84"
Create in place...
Load...
☑ Tag on Placement
Horizontal
Tags...
□ Leader



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

- Project browser
- Design bar
    - Basics
    - View
    - Modeling
    - Drafting
    - Rendering
    - Site
    - Massing
    - Room and area
    - Structural
    - Construction
- Status Bar
- View Toolbar

Definitions

- Elements
    - Model (host versus hosted)
    - View
    - Annotation (annotation versus data)
- Project
    - Template
    - Component
    - Category
    - Sheet
    - Sketch

Working in the environment

- Switching views and 3D view navigation
- Placing walls, windows, and doors
- Changing a model in different views
- Numerical editing of dimensions
- Wall joins
- Context-sensitive menus
- Keyboard shortcuts

Notes

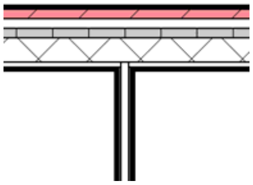
- In general in Revit Architecture, things that show up in a view in blue are editable. When you select an object, a variety of blue controls appear. Use these to drag wall ends, flip inserts, manipulate text notes, and so forth.
- The chain option is often useful for placing walls.
- You do not start by making a "drawing" in Revit Architecture. You start with a project—usually from a template. This project contains an empty model with several predefined views, along with some initial families and types of objects to work with.
- To add different kinds of objects you load different family files into the project.

Hands-on Topics

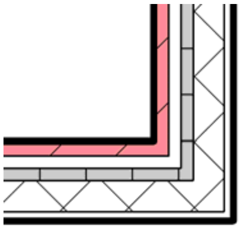
- Make a new project, and then experiment with it. Tile a plan view along with an elevation and 3D views. Establish walls. Place windows and doors. Add a floor and a ceiling. Now make several different modified versions of your design. How long did all this take you? How long would it have taken you in a CAD program?
- Use the File menu to start different projects with the different provided templates. What are the major differences between the templates?
- Workbook Unit 1, Exercise 1A: Interface and Terminology.
- Workbook Unit 1, Exercises 1B: Developing the Design.

Questions

- Where are all the drawing tools?
- What forms does the wall placement not support? Why do you suppose not?
- What sort of domain-specific rules are encoded in how Revit Architecture joins walls?
- Can you determine exactly what its rules for wall joins are (both corners and T's)?
- How tall are the walls you're placing? Can you change the height?
- What is the difference between the options bar and the toolbars? Why are some parameters available but not others from the options bar?
- What parameters are available as you draw the objects and how do these differ from the options bar as you select an object already in the project?
- How do you access more tool palettes (design bar)?
- What functions does the project browser perform?
- Can you find a way to make a wall that slopes from top to bottom?



These images show automatic wall joins as seen at a high level of detail. What assumptions is Revit Architecture making?



Unit 2

This unit introduces the concept of objects and gives you ideas for hands-on practice creating walls, floors, and ceilings in Revit Architecture.

Theory: Objects

BIM design basically proceeds by "placing" objects into a model and then adjusting their parameters. These objects are full-fledged building components like walls, doors, and windows. Although their parameters may vary, the placed objects retain their fundamental identities. A door that you make wider remains a door no matter what.

A Line Object, by Contrast

The objects that make up a drawing in CAD take their identities from their roles in constituting a drawing. A line is a line, and a point is a point. That you have the option of selecting the line and changing its properties such as thickness or layer indicates that this line is an object with parameters. It is an instance of a line class that has parameters for thickness and layer. But the program manipulates objects only at the level of drawing. The higher-level object you are depicting with your line objects gets no explicit support from CAD. CAD is solely concerned with the production of representation, whereas BIM design is concerned with constructing a building model complete enough that representations can be automatically generated from it. The objects used in BIM design encode much more than pure geometry.

Elements, Families, Types, and Instances

Each object in the Revit Architecture project belongs to a hierarchy that helps organize the objects in the building model. The terms used to describe this hierarchical classification from broad to specific are elements, families, types, and instances. This is the fundamental organization of the building model database. Most of the aspects of the building model, including the views, have this organizational structure. This concept is important because each of the objects has parametric control at these different levels of organization.

Elements or Categories: All objects in the building model are assigned a category. All doors in a project belong to the category Doors. This broad category is further broken down into families.

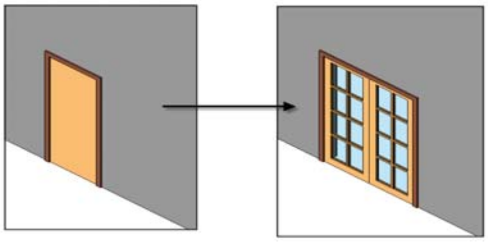
Families: Families are groupings of like geometry. Continuing with the door example, a single flush door belongs to a different family than a double door with glass in it because the geometry of the two types of doors is different.

Types: All design objects have a type. (A type is the same as a class.) The type defines what properties an object has, how it interacts with other objects, and how it draws itself into each different kind of representation.

Instances: An instance is simply a single object of a type in the building model.

Objects Are Created from a Type

When you place an object into a model, you are making an instance of a family. Most families have multiple types. A type specifies default values for a family's parameters. A type of a door has a specific width and height. Some types can be swapped for one another once an object has been created. For instance, a door can be reassigned to a type with a different width and height. A single door can even



A door can easily be swapped for a different kind of door, or later swapped back.

be swapped on the fly for a door of a different family, such as a double door. The software handles all necessary low-level geometric changes.

You could equally easily make a wall and change its structure later. Its thickness may change as a result. This kind of change is assimilated by the system. You cannot, however, change one object type to a totally unrelated type. There is no mapping of properties and functionality for the program to follow. You cannot, for example, make a wall and then change it into a window.

Instances

Instances of families are the substance of a model. Whereas a family is an abstract description of what parameters a class of objects must have and how they relate, an instance is a concrete exemplum of that type. It is an embodiment of the parameters and geometry that a type establishes. Types are unique, but there can be many identical instances of any type. These instances can be located at different places in the design. Some may have parameters set to different values, but fundamentally, they share a type and class.

Nothing New Here

Architects already deal with types and instances, although not always explicitly. A schedule of doors, for example, must list every door instance grouped by its type. Windows in a schedule may be listed by type. CAD has no notion of type versus instance, so it cannot help with this. CAD also cannot differentiate between the low-level linework that represents a door from the linework representing a window. However, with a full object-oriented building model, each object uses and retains the category, family, type, and instance information. The software uses this information to generate door and window schedules based on either the instance or type.

Encoding Design Intention

If you know something is a wall, you can say something about how it works. You know it has to join other walls at corners or Ts (structure to structure, finish to finish). When you move one endpoint of a wall, you'd like both sides of the wall to move. You might also like things embedded in the wall like windows and doors to move with the wall as you edit it. That is exactly what BIM design software does.

For the software to do this, it needs to know what the drawn objects actually represent. In CAD you may have drawn a wall as a long, skinny rectangle. You might also have drawn a long, straight walkway that way. How would the software recognize that one was a wall and one was a walkway? It would require sophisticated artificial intelligence. In fact, there is no machine on earth today that can do even a marginal job of deciphering design intention from a traditional CAD drawing. This goes right to the broad failure of the mid-twentieth century's promises for artificial intelligence. The problems are much harder than researchers thought they were. In fact, even humans often have a hard time deciphering others' drawings and need to ask for clarification. In short, the computer cannot read your mind. Given this understanding gap, how can BIM design not be a doomed project?

Specify What, Then Where

Workflow in BIM design proceeds as follows:

1. Specify what type of object you are about to place into the model.
2. Specify all the necessary information about it so that it can be placed properly.

What that information is depends in large part on what kind of object you are placing. For a wall you must at least specify its start and end points. You will also have chosen ahead of time what kind of wall it is (partition, exterior, and so forth) and its height, but you can change these later. This way there is no ambiguity. Anytime you are creating the building model, you are really specifying some parameter of an object that you are placing into the model. A path would be a path, and a wall would be a wall from its inception. Notice that this methodology frees you from unnecessary labor. You don't need to draw four lines to complete a wall; picking two points for a straight wall or three points for an arced wall now usually suffices. You can draw an arced wall if you want to, but you must first specify what you want to do. You can change parameters such as the arc's center or diameter afterward.

Everything in Revit Architecture Is an Object

Models are made of design objects—walls, doors, windows, stairs, and so forth. But it isn't just physical building objects that are "objects" in Revit Architecture. Anything with properties that can vary is a Revit Architecture object, and that includes things such as views. A view has properties that specify what is visible. You can choose to show furniture, for instance, or not. A section specifies the location of the cut and direction of viewing.

Summary

Object orientation is a powerful methodology that has made its way from software engineering into architectural design in the form of BIM design. BIM design enables architects to specify their design intention at a higher level. They are free to compose their models from premade building components and types and then vary the parameters.

Revit Architecture: Walls, Floors, and Ceilings

The Connection

Clearly, the walls added to a project are model objects. When manipulated in a single view, they update in all views, indicating that they are present in an underlying abstract building information model. Now that you understand the concept of objects, let's start populating a model with a new kind of object with interdependencies.

Revit Architecture Features and Concepts to Learn

- Floors:
  - Floor by raw sketching
  - Floor by wall outline
  - Moving walls with attached floors versus floor edges by raw sketching
  - Changing which wall edge the sketch references
  - Floor with cutout
- Ceilings
  - Ceilings by walls and rooms
  - Compound ceilings
- Walls
  - Changing a wall's parameters
    - Type
    - Orientation
    - Location line
    - Height
    - Base and top attachment conditions, including offset
  - Curtain walls, stacked walls
  - Wall sweeps and reveals
  - Sweeps and reveals interacting with doors and windows
- Making walls, floors, and ceilings at different levels
- Work planes
- Changing a view's parameters
  - Display mode (model graphics style)
  - Wireframe
  - Hidden line
  - Shading
  - Shading with edges
  - Level of detail
  - View range (cut plane)
  - Underlay
- External family libraries
  - Browsing available types installed, Autodesk website, other websites
  - Loading and unloading
- Selecting all of a type (project browser), or RMC with object selected in drawing window

Notes

A sketch in Revit Architecture is an informational framework that a complicated component like a floor uses to control its shape. The best way to think about a sketch is as a graphical parameter used to control the shape of a component. It can be edited later, and the component to which it is attached updates accordingly. There is no such thing as a sketch without an associated component. The sketch may be made first to define a default shape for the component, but the sketch is not the component, only a property of it.

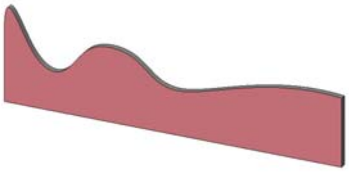
The sketch interface is modal, meaning that when you're in it, you're in it. You can do things in sketch mode that you can't do outside it, and vice versa. In fact, there are lots of different sketch modes, each one tailored to the component for which it is a sketch. Some of them are volumetric sketches, some 2D shapes, some closed shapes, some open. But in all cases to leave sketch mode you have to choose either Finish Sketch to accept the sketch or Quit Sketch to discard it. Another way to think about sketch mode is a set of geometric parameters for an object. A planar orthogonal wall needs only two selection points and a height to determine its extents. A floor slab, however, needs you to tell it its boundaries. You do so by establishing its sketch and then passing this geometry back to the object as its edge parameter.

Hands-on Topics

- Experiment with sketching. Determine what each tool does. Make a multistory building with different cutouts in each floor.
- Use the File menu and choose Load From Library>Load Family. Browse for interesting-sounding components. Load your building with them. Are there any that you can't identify? Swap some for others of the same category.
- Modify your design so that not all walls are the same height. Some may span several stories and some only one. Tie the base and top conditions to the levels directly. Now change the levels. Did the walls adapt appropriately?
- Workbook Unit 2: Walls, Floors, and Ceilings

Questions

- Why is the sketch interface modal?
- What are the rules for determining if a sketch is a closed loop?
- Are two disjoint closed loops valid?
- Clearly a closed loop inside a closed loop is interpreted as a hole. What is a closed loop inside a hole interpreted as?
- Is there more than coincidence to thank for the similarity of the tools available for editing floor sketches and for placing walls?
- In sketch mode what is the difference between the properties on the options bar and the properties in the design bar?
- Do all objects have editable properties?
- What properties in the wall's Properties dialog box don't you understand?
- Why are there dimmed properties at the top of the dialog box that you can't change?
- How did I do this?



- In the workbook Unit 2, after you loaded the *Curtain Wall Dbl Glass.rfa,* can you use this in a wall? Why or why not? Explain in terms of category, family, type, instance.

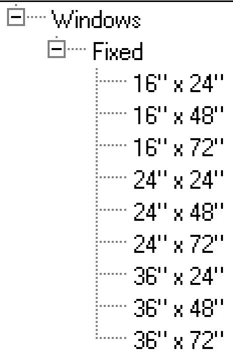
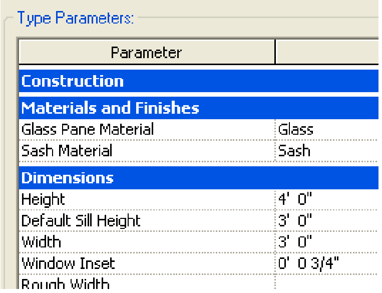
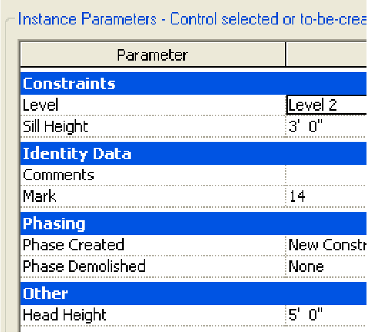
Unit 3

This unit introduces the concept of families and nested families, and gives you ideas for hands-on practice editing types in Revit Architecture.

Theory: Families and Nested Families

Families: Groups of Types

As discussed in the previous unit, types are descriptions of objects' parameters, and objects are instantiations that hold specific parameter values. Immediately above *type* in the Revit Architecture hierarchy is the *family*. Families are related collections of types. It is common for certain sets of parameter values to be meaningful as a group. A certain kind of window, for example, may be available only in a specific set of sizes. These sets of parameter values can be given names and grouped together into a family. Succinctly, families are sets of types that vary only by their "type" parameter values. In addition, the family controls how an object behaves. Much of the encoded design intent noted in Unit 1 is encoded at the family level. There are two main types of families: system families and hosted or external families. External families are those families stored in the libraries such as doors and windows. System families are families of objects such as walls, roofs, and floors. System families cannot exist outside the Revit Architecture project file as a separate family file. All windows of one type have the same dimensions internally.

		
The fixed window family has the types shown here.	All windows of type *Fixed: 16" x 24"* have the same material, height, and width.	Each instantiation, however, can have its own head or sill height.

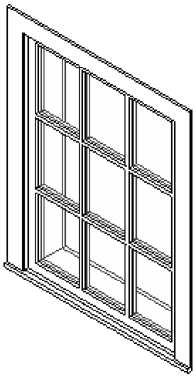
Windows
Fixed
16" x 24"
16" x 48"
16" x 72"
24" x 24"
24" x 48"
24" x 72"
36" x 24"
36" x 48"
36" x 72"

Type Parameters:

Parameter	
Construction	
Materials and Finishes	
Glass Pane Material	Glass
Sash Material	Sash
Dimensions	
Height	4' 0"
Default Sill Height	3' 0"
Width	3' 0"
Window Inset	0' 0 3/4"
Rough Width	

Instance Parameters - Control selected or to-be-crea

Parameter	
Constraints	
Level	Level 2
Sill Height	3' 0"
Identity Data	
Comments	
Mark	14
Phasing	
Phase Created	New Constr
Phase Demolished	None
Other	
Head Height	5' 0"



Although you have control over some basic size parameters, in general you cannot adjust the internal construction of the window using basic parameters. For the fixed window family there is no parameter that shows the muntins of the window.

The software supports the ability to create complex families that drive geometry as well, but that topic is beyond the scope of this document. Again, as a general rule the geometry or construction is held in the family.

This image shows a different window family: Casement 3 x 3 with trim. This family holds the geometry for muntins and all types of this family also have the same 3 x 3 divisions.

Editing Types

You can create types of your own inside most families. Types can be created either within the project, or within the external *family.rfa* file. These types are parametric variations on the family's theme. You could create a new type in the Fixed window family and call it anything you choose. The names that have been established already indicate the parameter choices they represent (36" x 48", for example), but that is merely a convention. You could call yours anything you want.

Type Parameters

When editing the parameters of an object such as a wall, many properties remain uneditable. They appear in a gray box called *type parameters*. Type parameters are the values that must be the same for any instance of this type. The family provides the template for what these parameters are, and the types themselves are the templates that fill in those values. Finally, instances fill in the values of the instance parameters.

As a result, all objects of the same type share the same type parameters. Any change to their values affects all these objects. Changing these values is something you should do with care. That is why they are not immediately editable from the same page that exposes the instance parameters to editing.

Duplicating and Editing

To edit the type parameters, you must either explicitly choose to edit them, or duplicate a type and edit the parameters of that one. Duplication and modification is the best way to work with canonical types so that you don't destroy them as you modify them. Further, if you want to export families with their associated types for use in other projects, you need to give them unique names that you will remember.

Choosing Type Versus Instance

Why are some parameters type parameters and some instance parameters? Type parameters are the ones that define the identity of a class of objects, and instance parameters are the ones that vary across specific individuals. Instance parameters tend to deal with context (extrinsic relationships, such as placement in a host), where type parameters tend to be intrinsic (self dimensions, for example).

The Role of the Family

If type values are stored in a type and instance values are stored in an object, what exactly is the role of the family? The family defines what parameters exist, whether they are type or instance parameters. The family defines how the object renders itself into various views depending on its parameters. Exactly how all this is specified is covered in the discussion on custom types later in this document.

Nested Families

A family may contain one or more other families as part of its definition. These are called nested families, and they are a powerful concept. Nested families allow aggregates of components to be treated as a single component. For example, a set of door hardware can be created as its own family and then used in many different door families. If a change is made to the door hardware, the change can be easily propagated into all the applicable door families by reloading the hardware family.

Revit Architecture: Editing Types

The Connection

Editing types is a fundamental exercise in parametric variation. The ability to modify and store named sets of parameter values enables you to build custom component libraries suited specifically to your design needs.

Revit Architecture Features and Concepts to Learn

- Sweeps and reveals
  - Creating, editing, and using profiles
  - Wall sweeps/reveals
    - Horizontal wall type
    - Vertically applied
  - Roof and slab edges
  - Railings
- Types
- Editing types
  - Creating new types from loaded families
    - From properties page
    - From project browser
  - Creating new types from system families
  - Different ways to access parameters
- Families
  - New families from the family library
  - Introduction to the Family Editor
  - Save Family from Project (Save to Library)
  - Edit in place in the project

Notes

- Functionally, families are different from one another. There are families that can exist outside the building model as separate files, and there are system families that can exist only with the project building model file.
- System files can be copied from one project to another.
- There are two ways to edit a family: you can open the external family to modify it, or you can select the family and edit it in the project. If you select a hosted family in the project you can edit it directly within the project. Once you are finished making the edits, you can chose to either load into any open project (save back into the original project, overwriting the existing definition); save as an *.rfa* file, overwriting the original; or save as new family.

Hands-on Topics

- Load a few families from disk. Pick several types and modify them radically. What is the difference between modifying the type and the instance parameters? Is there a pattern to which parameters are which?
- Create two different wall types: gypsum wallboard on metal stud and gypsum board on wood stud. Add several of each to a project so that they intersect forming L's and T's. View these in a plan view set to medium detail. Should these walls clean up the components? Why or why not?
- With the walls created in the previous topic, use the **Edit Wall Joins** tool to modify how the walls clean up.
- Workbook Unit 3.

Questions

- Do you see a pattern in the list of system families? Why are they fixed? (There is a specific logic to it, although it can be argued that the decision is detrimental.)
- What types can you swap in place for one another? What is the pattern? Are there exceptions? What would be the consequence of allowing more general switching, say from door to window? What kinds of things would the software have to know?
- Can you add a window to a curtain wall? Why or why not? What assumptions does the software make?
- Load the door openings from the installed family library. Why do you have to add this as a component rather than a door?

Unit 4

This unit introduces the concept of parameters and gives you ideas for hands-on practice creating dimensions, doors, and windows in Revit Architecture.

Theory: Parameters

Objects are differentiated by two things:

Behavior: How an object behaves—how is it added, modified, displayed—as well as how it relates to other objects is primarily determined by the category. This is hard coded in the software. Even seemingly disparate objects of the same category such as a curtain wall and a brick wall share some common behavior. Both are added in a similar manner. They can be swapped for one another, and both can establish room boundaries. Some behavior is also influenced by family settings. When you create a component family, you can choose a template that makes the component anchor to a wall, ceiling, or floor.

Properties and Parameters: Every object in a building information model has properties that affect both its behavior and description. These properties are the parameters held either by instance or type. In every case, you modify an object by changing the parameters that describe that object, be it length, material, color, or textual description. Every object is also encoded with specific rules that either permit or disallow specific modifications. For example, a window must have a host wall to exist. A window cannot be placed if there is nothing to place it in.

Property Types

You may have noticed in your experimentation with object properties that the values of the parameters, too, have types that indicate what kind of values they can contain. Some parameter values are numbers, some are positions in space, some are strings of characters, and some are even other objects. For example, a wall's "unconnected height" is a distance parameter, whereas its "base constraint" refers to any already-instantiated level object. So the "base constraint" parameter is really a parameter of type "level." The ability of parameters to refer to other model objects introduces a fundamental split between two types of object parameters.

Primitive Versus Complex Types

Primitive types of parametric values comprise things like numbers, lengths, and raw locations. These types are not object types that can be instantiated directly into a model. There is no way to place down a "2" into the second floor of a house. You can certainly annotate views with text, but these annotations are annotation objects, and their text is simply one of their properties. Primitive types are easy to specify: you can type them in or read them off a model by means of dimensions. Complex-typed parameters take values of full object types. These parameters usually reference a specific object in a model, such as a view or level. Therefore the objects of reference must exist before you set the parameter. You cannot attach a wall's base constraint to level 4 before you instantiate level 4. And then once you have attached a complex property to a model object, that relationship remains invariant. Should level 4 change height, the wall changes height with it. If level 4 is deleted, then the property bound to it becomes invalid. There is no standard of correct behavior for what to do when a property becomes invalid. It usually results in a warning message or error.

Default Values

When you create an object, some of its parameters are specified by the action you take to add it to the model. For example, a wall has its start and end points specified this way. Other parameters are not explicitly specified at the time of object placement and are given default values. What value a property gets by default is part of the description of the object's family or type, and it is a mark of good design to make these defaults reasonable. There is no point making a wall's default height 1 foot. It just makes work for users because they have to change the height of every wall they make. Instead, a wall takes as its default height the height of the level to which it is being added. That way it extends from floor to floor by default. If you want to change it afterward, you can.

Instance Versus Type Parameters

The distinction between instance and type parameters is subtle but important. Every object has a type and is an instance of that type. Type parameters provide the leverage to change many individual instances at once. For instance, if you were to place a series of doors of the same type and that type specified the door width as a type parameter, all it would take to affect all doors of that type would be to change that parameter's value. This is indeed the default characteristic of most of the doors provided with the software

In contrast, if the width of the door were an instance parameter, each door object would maintain its own copy of that parameter and could be changed separately. Although Revit Architecture does provide the functionality to do this, in general this level of autonomy is not desired so your building can use standard sizes for doors, windows, and other objects. More emphasis is therefore placed on type parameters, and many subtly different types are defined for flexibility. Some parameters, however, make sense only as instance parameters. The sill height of a window could be specified as a type parameter. That way all windows of a certain type could be moved up and down at once, but it would allow no flexibility: you could not place this type of window at different heights even in different walls. This is exactly the kind of parameter that makes sense as an instance parameter. In general, you can think of the distinction this way: Type parameters define all that is common between individual instances of the type, and instance parameters define all that can vary from instance to instance (such as placement).

Implicit, Explicit, and Mixed Parameters

There are many different ways to modify parameters. Different types of parameters require different modes of manipulation. One of them is by typing in new values from a dialog box, another may be positioning a point with the mouse, and even a single parameter may offer itself to modification in multiple ways.

Explicit parameter change: Let's take the height of a wall as an example parameter. This property appears in a list of properties for a wall. This is its unconnected height. You can type a new value into this field and the wall changes height. This is an explicit parameter change.

Implicit parameter change: But you can also tie this wall's top constraint to another value—the height of a "level"—and change that level by dragging it up or down in elevation with the mouse. If a wall's top constraint has been tied to "Up to Level 2." That means that however high Level 2 becomes, this wall stops exactly there. And although "unconnected height" no longer represents the height of this wall (since it is connected), notice its value changes when you change the height of Level 2. The wall height in this case has become an implicit parameter. It changes to reflect what value it has to take to satisfy the system as you modify it. Implicit parameters often go unnoticed. A single mouse-click may affect hundreds of them at once.

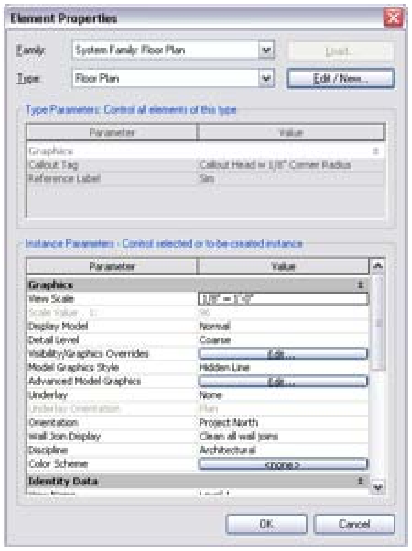
You can operate on a wall height either explicitly or implicitly, depending on how you set things up. But some parameters are purely one or the other. Take the material parameter, for example. A *material* is a definition that contains its own set of parameters. These parameters control the display of hatch, shade, and rendering mode. Because a material is a collection of parameters, there is no more convenient way to specify what material an object is made of than by selecting it from a list. So this parameter is bound to show up exclusively in property lists. By contrast, there are purely implicit parameters that never show up in property lists because they can't.

A good example of this kind of purely implicit parameter is a *sketch*. Revit Architecture introduces the sketch concept to represent any free-form, shape-based parameter that an object may have. A *floor* is defined by several parameters that determine its composition, thickness, and level. But it also needs to conform to a shape. Revit Architecture provides the sketch interface to specify such shapes. The tools available here (especially when you select Lines) are quite familiar to CAD users. These are the shape-drawing tools that a raw geometric specification requires. The difference here is that in a CAD tool, this geometric specification mode is the only mode of operation. Geometry is about all there is to specify. In a BIM tool, however, notice the two options Finish Sketch and Quit Sketch. These indicate that in Revit Architecture, sketching is a minor mode that is entered into occasionally to determine a geometric parameter of an object, but no more. (Finish Sketch and Quit Sketch both leave sketch mode, but Quit discards the work.) This is a subtle but important point. A sketch *is not* a floor. A floor *has* a sketch parameter, which defines its outline. As you can imagine, there is no convenient way to describe a sketch in a property list, so it never appears in them. But it is a parameter, just like wall height. This is reinforced by access to the floor's properties from the Sketch design bar. As you are working on the sketch parameter, you also have access to the construction and type of the floor.

Multiobject parameter interaction: One of the most intricate ways to work in parametric design is by specifying relationships between parameters that a design must maintain. You have already seen a simple example of this in the tying of the wall's top to the height of Level 2. You can tie many other objects' parameters to the same data, and then when that changes, they all remain valid. You expect a floor and the top of a stair to be similarly tied to a level. Alignment like this is a simple equivalence relationship, but you are free to specify far more complicated relationships. You could mandate that one wall opening is twice the width of another. You could specify that a roof angle is to be a specific multiple of a sun angle that you don't yet know. When you plug in the sun angle you are working with, the design changes to accommodate it.

It is likely that the parameters you are tying together are geometric and may therefore not appear on a parameter list. But they respond to each other automatically when either is manipulated.

Most Objects Have Parameters



Element Properties

Family: System Family: Floor Plan

Type: Floor Plan

Type Parameters: Control all elements of this type

Parameter	Value
Graphics	
Callout Tag	Callout Head w 1/8" Corner Radius
Reference Label	Sim

Instance Parameters - Control selected or to-be-created instance

Parameter	Value
Graphics	
View Scale	1/8" = 1'-0"
Scale Value 1:	96
Display Model	Normal
Detail Level	Coarse
Visibility/Graphics Overrides	Edit...
Model Graphics Style	Hidden Line
Advanced Model Graphics	Edit...
Underlay	None
Underlay Orientation	Plan
Orientation	Project North
Wall Join Display	Clean all wall joins
Discipline	Architectural
Color Scheme	<none>
Identity Data	

Many of Revit Architecture software's elements are objects even if they are not building components, and they have properties. Views, for example, have lots of parameters (see image at right). Among other things, you can specify whether furniture is visible in a specific view. Because there is a single, unified building model, you don't make one version of the model with furniture, and another without. You place the furniture should you ever want to see it, and choose whether it is visible in each specific view of the model.

Typically, the interfaces for tying parameters to each other are exposed through the same views that display building components with graphical representations. Therefore, it is often not possible to establish parametric relationships with the properties of abstract objects like views. Theoretically, you can imagine tying the scale of a view to the extents of the objects in it, but the program does not yet support that kind of parametric control.

Revit Architecture: Dimensions, Doors, and Windows

The Connection

Revit Architecture introduces access to object parameters in several ways. You have seen the Properties dialog box, which every object supports. It displays all the editable properties of an object. But Revit Architecture could be a purely textual system if that were the only parameter access it provided. There is a rich graphical interface for creating and modifying the form of components. These are parametric operations, as you can confirm by checking the Properties dialog box before and after you change a wall's length.

But Revit Architecture uses another mechanism to expose parameters—dimensions. A dimension in Revit Architecture is an active parameter that you can use to control model properties. In fact, almost any linear property can be exposed and manipulated textually or graphically by means of dimensions. Dimensions also serve to introduce the model parameters that do not strictly belong to a single object. The distance between two walls can be dimensioned and then controlled as a parameter. This distance does not belong as a property to either wall and therefore does not show up in its property sheet. But the dimension does exist as a constraint in the permanent model, and Revit Architecture attempts to satisfy it.

Revit Architecture Features and Concepts to Learn

- Dimensions
  - Temporary versus permanent
  - Linear, angular, radial
  - Length syntax
  - Moving the witness lines
  - Multireference dimensions
  - Dimension handles
  - Dimensions and wall centers, faces, cores
  - Dimension properties
  - Locking dimensions
- Doors
  - Changing orientation and swing
  - Changing type
  - Other properties
- Windows
  - Placing
  - Changing type
  - Other properties
- Changing walls containing doors and windows
- Instance parameters versus type parameters
- Families and types
  - Loading families
  - Editing type parameters
    - Through the families list in project browser
    - Through Edit/New…
  - Creating new types
    - Duplicating types
    - Through the edit feature of the options bar
- Type naming conventions

Notes

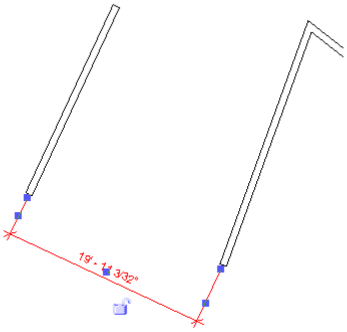
- Dimensions mean much more in Revit Architecture than they do in a CAD tool. In a CAD tool, they are purely informational. They give information about the model to a user. In Revit Architecture, dimensions are both informational and a means by which to manipulate the model. They give information to the model from the user. You make much more use of them than you do in CAD.
- All length and distance parameters can be exposed through dimensions. Some are also exposed through other means (such as a Properties dialog box), but you can always get to them by dimensioning them and manipulating the dimension.
- Everything has parameters that are editable the same ways: views, chairs, walls, levels. You name it.

Hands-on Topics

- List all the properties you think a wall ought to have. Now place a wall and open its Properties dialog box. Does your list match that of Revit Architecture? Explain any differences.
- Load a few families and place instances into your design. Pick several and make new duplicate types from them with new names. Adjust their properties until they are unrecognizable. Manipulate both type and instance parameters until you understand the difference between them.
- Workbook Unit 4.

Questions

- What different roles do dimensions play in Revit Architecture?
- Try making two parallel walls and a permanent dimension that measures their separation. Now select that dimension and try to change its value. Why doesn't it work? What would you have to do to make it work?
- Play a little with the lock symbol. What does it seem to do? We return to it in depth in Unit 17.
- When walls containing doors and windows change shape, what specifically happens to the doors and windows?



- What happens if you shorten a wall below the top of a window it contains?
- Why can't you edit type parameters directly from an instance's Properties dialog box? What would be the harm?
- Why duplicate types when you modify them? When should you not duplicate?

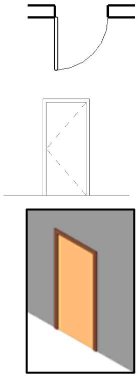
Unit 5

This unit introduces the concept of representations and gives you ideas for hands-on practice creating views and sheets in Revit Architecture.

Theory: Representations

To make use of the model, you must specify the viewing conditions under which it is to be represented. There can be no meaningful interaction with a design model without choosing a view, be it plan, elevation, section, 3D, or schedule. To be of use in a model, each object must be able to represent itself in all possible viewing modes. Or in other words, the objects must have the intelligence to send a different set of instructions to the graphic card saying "show me this way" when viewed in plan view, and a different set of instructions to the graphics display system when viewed from an elevation or isometric direction.

Take the following example: Which representation best describes a door?



"single door type 1"

The answer of course is that it looks like none of them and all of them at the same time. They are each valid representations of the same object in the model database, but they are useful under different circumstances.

Why You Don't Draw the Geometry of Objects

If you were to draw a door in plan, it would take powerful artificial intelligence for the machine to decide that those few lines and an arc should represent a door. Sometimes it would be right, sometimes wrong. Rather than incur this ambiguity, you define a door ahead of time that can draw itself in any view, and then you simply place an instance of that object into your model.

This kind of embodied "intelligence" is the best fruit of object-oriented design. The algorithm for drawing any kind of view becomes quite similar if you allow every object to be in charge of its own set of instructions to the computer's display system. You simply tell it to "draw yourself," and it recognizes what that means depending on the active view.

A View Is More Than a Representation

Beyond mere representation, a view in Revit Architecture specifies a mode of interaction with the model data. It is possible to change the heights of objects by manipulating them with the mouse in an elevation or section view, but not in plan. In a 3D view there is considerable ambiguity about what kinds of transformations a 2D mouse movement ought to imply. Revit Architecture introduces the notion of work planes to constrain the mouse to two dimensions in such cases. The point is that a *view* is really more than a view. It is an entire interface. You must choose to work in the view that makes the most sense for the kind of work you are doing. It would not make sense to work on a façade in a plan view. Complex geometries may be impossible to comprehend in anything but a 3D view. It is quite common to switch rapidly back and forth between views to make modifications to objects and their positions and properties.

Some Objects Appear in Only Some Views

Not everything that appears in one view ought to show up in every other. For example, you may add a crowd of people to your 3D renderings that you do not want represented in your plan (lest they get added to the shop drawings). This is again an object-view decision. Every view has the prerogative to ignore any specific categories or subcategories that should not show up. An object's not appearing in a view must not be mistaken for its being absent from the model. It is still there, but it is invisible and unmodifiable from the current view. You may wonder if there is a definitive way to catalog everything that is in your model if there are things that may not show up in any given view. This is a good question, and it can be addressed by turning on the visibility of as much as you can for a particular view or by using Revit Architecture software's ODBC (Open Database Connectivity) export, which exports information about all elements in the model to a spreadsheet or database.

Annotations and Edits

Certain kinds of work do not belong in the unified building model. For example, if you decide to show a specific dimension in a view, that dimension itself is a graphical element, but you might not want it to show up in a 3D view no matter what. Or you may place some explanatory text into a callout. This text belongs to this specific callout and should not be duplicated anywhere else. Revit Architecture does support the creation of text and geometry directly into a view. These are annotations and "drafting objects," and have no bearing on the building model. Annotations and drafting objects are stored with a view. They may be copied to other views. This is the mode most familiar to CAD users because it is exactly the same as using lines, arcs, text, and so forth in a traditional CAD package.

How can Revit Architecture introduce view-specific elements without destroying the benefits of the unified view-independent model? The answer is carefully. It is tempting for CAD users to revert to old habits and simply enter drafting mode in a view and start drawing. This approach is risky because the geometry created inherits none of the design intelligence that the rest of the model has. If it is drawn in plan, it does not show up in elevation. If a matching elevation detail is added, it does not update when the plan is modified. It is therefore imperative to use view-specific elements only where they are needed. There are three cases in which they make sense.

First, any information that conceptually does belong only to one view should be created as a view-specific element. That information often includes annotations, edits, and dimensions. These are not part of the building and do not properly belong to the model. The program should not even attempt to format this information to show up in another view. This kind of annotation does not bend or break the model. It is meant to be view-specific.

Second, any information that goes beyond the level of detail intended for the building model should be made view dependent. For example, you would not model every piece of hardware that makes up the cladding system, because it would overwhelm the system. Instead, you would represent this kind of detail in a representative callout. You essentially "draft" the detail into the callout. It exists only in this view as an explanation of the detail, and should the surrounding geometry change, it must be redrafted. Choosing the level of detail at which to model is a decision that affects workflow tremendously, and we return to it later in Unit 14.

Third, you clean up or draft corrections when the program displays something you don't expect. No matter how smart the program is, sometimes the graphic representation of the model in a specific view does not match what you know it ought to look like. These cases arise from errors in display, drawing conventions that the program does not recognize (imagine if doors were exotic—how would the program know about door swings?), unusual demands (such as "this door must be drawn as half open"), or non-rule-based visual preference (such as "this line should be thicker here"). Revit Architecture gives you the control to make this kind of edit, but these tools are deliberately deemphasized. Basically, each revision of the software aims to reduce the amount of this kind of work.

Multiple Similar Views with Different Properties

One way of working with views, with which you will likely become familiar, is copying them and slightly modifying their parameters. The properties of a furniture plan may be similar to those of a regular first-floor plan. You simply copy your first-floor plan view, rename the new view "furniture plan," and change the view settings so that furniture objects are visible. You have not changed your building model a bit. You have simply introduced a new view of it. And a schedule is no different. It is a view of your model that has nothing to do with geometry. It is a view (as a list) of the names and types of objects your model contains.

Sections and Elevations

Sections and elevations historically are one method of communicating some of the 3D design intent. It is common for designers creating freehand sketches to work in section simultaneously with the plan views they are sketching. With Revit Architecture you can also use this method. While in plan view when you place a section or elevation mark, you are not just placing an annotation reference, but creating a section of the building model that appears in the view lists of the project browser. Often, it is easier to work or select objects in a section or elevation view than in plan view. Because the section is a view of the building model, and not just an extraction, any changes you make in a section view affect the plan as well.

Revit Architecture: Views, Visibility, and Sheets

The Connection

Revit Architecture strongly enforces the separation between model and view. The interface terminology makes this clear. The views are listed under a category called *views,* not *drawings*. They are not drawings, but different representations of the same model. Of course, "view" is a bit of a misnomer because it is only through a schedule or graphic view that you have access to change the model. A view is more than a window into a model; it is like a window outfitted on the exterior with robot arms to change its environment. We are now going to look closely at how to work with views, which are central to BIM design.

Revit Architecture Features and Concepts to Learn

- Views, general
  - Creating new views
  - Copying with and without detail
  - Dependent views
  - Navigation and zooming
- Linking views
  - Showing underlays
- Visibility
  - Showing or hiding things
  - Reducing clutter
- Sections
  - Creating
  - Navigating
  - Crop region and far clip
- Callouts
  - Creating
  - Scale
  - Level of detail
- 3D views
  - Making cameras
  - Modifying views
  - Navigation
  - Shading
  - Quick renderings
  - Sun and shadow studies
- Sheets
  - Compositing views
  - A view can appear on only one sheet
  - Scale considerations
  - Title blocks
    - Templates
  - Sheet text
- Materials
  - Drafting fills versus surface patterns

Notes

The scale of the view is the only determiner of scale as it is placed on the sheet. This is purposeful and unrestricting. If you want it to appear elsewhere at a different scale, you have to make another view at that scale.

The distinction in Revit Architecture between annotation and model elements is fundamental. Annotation is view specific; model elements appear in all views. The trick is understanding what Revit Architecture considers annotation versus data versus view.

- If you place a door tag, this is pure annotation. It only appears in the view in which it is placed.
- If you add a level or grid mark, these appear in all views because they are datum marks.
- If you add a section mark or elevation mark you are adding a view. This is reinforced by the fact that the properties for the mark itself have the same controls and parameters as section view properties.
- Linked Revit files also have similar view-based control over visibility; refer to Unit 12.

Hands-on Topics

- Create several new 3D perspective views of your design from the previous exercise. Compose them on a sheet with an appropriate title block.
- Produce two new plans of Level 1, one showing furniture, and one at a high scale and level of detail to show wall interiors.
- Create a plan view with door tags, section marks, and some text. Duplicate this view twice. The first time use the duplicate with detailing feature. For the second copy use the simple duplicate (without detailing) feature. Study the differences between the copies. What appears and what does not appear?
- There are two different line tools. One is on the Basic tab of the design bar, the other on the Details tab of the design bar. What is the fundamental difference between these two tools?
- Workbook Unit 5.

Questions

- What functionality do sheets replace in CAD?
- Why can’t you put a plan view on more than one sheet?
- Who dictates the scale of the view on the sheet? Why?
- At what scale are elements drawn when placed directly on the sheet (not from an external view)?
- What elements belong in a sheet rather than a view?
- Why do annotations and symbols change size when the view scale is changed? Did they change size at all?
- What purpose does a dependent view have? Discuss the advantages of a parent view with two dependent views versus two views that are just duplicated with detailing.

Unit 6

This unit introduces the concept of design constraints and gives you ideas for hands-on practice creating levels, reference planes, and grids in Revit Architecture.

Theory: Design Constraints

Constraints

Much of the work of design is the creation and discovery of constraints. For a given job you may be presented with a brief, a site, and a budget. Taken alone, these are clearly not enough to generate form. Your job as an architect is to make a specific proposal for a massively underconstrained problem. Whether you are conscious of it or not, most of your early design process is the invention of a system of constraints consistent and rich enough to guide your evaluation of ideas.

Where Do They Come From?

It is helpful at this early stage to establish the dominant criteria of the exploration. Some of these criteria come directly from the brief: program, square footage, and budget. Some come from historical or theoretical influences. Some come from found conditions: site, available materials and technologies, the clients' proclivities. Some come from a trained pragmatic intuition or analysis: dominant axes, structural system, level height. And most come from the designer's personality.

Defining Relationships

If any of these initial constraints can be expressed formally, they can be encoded into parametric relationships that can drive a model. Paradoxically, establishing fixed conditions is the hardest part of generating a dynamic model. Parametric relationships enable you to design with a certain set of driving conditions, which you can modify at a later stage and retain a valid design.

Assume that you decide to design with a consistent 10-foot level height. Later you decide that the first floor ought to be pushed up by 2 feet. If you are careful to establish level heights as base conditions, and design relative to them, all you need to do is modify a single parameter. The rest of the design automatically adjusts to the new conditions. Every interior and exterior wall, every stair, and every ceiling that was properly tied to the level height updates automatically.

Data

To promote making these kinds of relationships, Revit Architecture makes it easy to establish certain kinds of data. Level heights are a good example. A framework for establishing and setting up relationships and then modifying them is included in the program. The same is true for grids, which you can lay out with considerable freedom. Then any kind of design can be made conforming to the grid. If the grid is later modified, the design changes accordingly. Both grids and levels are data elements in Revit Architecture. However, they have slight variations in behavior. A level can be intrinsically associated with a view. When placed in a view window, model elements are inherently constrained to the level associated with the view. Grids, on the other hand, can act as a reference plane for placing objects and aligning objects to.

Alignment

By now you have worked with level constraints and dimension constrains. Alignment is another common constraint. Many different elements can be aligned. Part of the Align tool function is a lock prompt in the drawing window. This lock appears immediately after you align two objects and establishes the constraint between the two aligned objects. This relationship can be between almost any two faces of objects:

- Window edge to a grid in elevation
- Face of one wall to another wall
- Edge of a piece of furniture to face of wall
- Face of wall
- Face of core, center of core, and so forth to a gridline or reference plane

Revit Architecture introduces “reference planes” to help establish such conditions. Once a reference plane has been placed into a model, objects can be placed relative to it. When the reference plane is moved, all dependent objects move too.

Revit Architecture: Levels, Reference Planes, and Grids

The Connection

Specifying invariance is the key to parametric design. To clarify: If two walls meet at a corner and one of them is moved, Revit Architecture changes the length of the other to preserve the corner. The corner is the invariant condition, which the operation of moving the first wall challenged. Revit Architecture preserved the invariance by extending or shortening the other wall. This is a simple example, but the principle is the same no matter where it appears. If a wall is constrained to begin 3 feet above level 1, and level 1 changes height, Revit Architecture preserves the stated invariant condition between level 1 and the wall. Rather than veto the change to level 1, the software changes the height of the wall to maintain constraints. Of course parameters tied to the height of the wall now change too. This is the propagation of constraints, to which we return later.

Three types of objects in Revit Architecture form the basis for a great many invariant relationships: levels, reference planes, and grids. None of these is a physical model object, but are abstract data from which to perform relative operations..

Revit Architecture Features and Concepts to Learn

- Levels: Revit Architecture software’s basic horizontal plane data (vertical control)
  - Default levels
  - Establishing levels
  - Associated views
  - Wall/ceiling attachment to levels
- Reference planes
  - Vertical planes are easy to add in plan or section
  - Horizontal planes are easy to add in elevation or section
  - Reference planes can be placed anywhere
  - Levels as reference planes
    - Needed for 2D drawing
  - Implicit reference planes for views
  - Selecting a work plane
  - Dimensioning to or from reference planes
  - Adding reference planes in sketch mode
- Alignment
  - The Align tool
  - Locking to alignments
- Grid
  - A named set of reference planes
  - Shows up by default in many views
  - Creating
  - Dimensioning
  - Labeling
  - Gridding to an underlay
- Equalizing dimensions
  - The EQ feature
- Locking walls to an equalized moving grid

Notes

- Establish the framework early.
- Levels are among the first things that need to be established in a project.
- Any drawing that takes place with a 2D toolset such as the line toolset must take place on a specific plane. Most often this is the plane of a level.

Hands-on Topics

- Constraints encode design intent into the model. Create a design based on and constrained to a regular grid. Deform the grid radically. How does the design behave?
- Make a reference plane from east to west in plan view. Switch to a southern elevation. Where did the plane go? Is it possible to work on the plane from here? Switch to an eastern elevation. Is it possible to work on the plane from here?
- Change the length of a level line in the south view of a project. Look at the north view. Are the extents of the level line changed here also? Can you find a way to make it not take effect in parallel views?
- Draw a wall with location line set to Wall Center Line on top of a gridline. Does the lock that appears serve the same or a different function from the Align tool lock?
- Workbook Unit 6

Questions

- Why are reference planes necessary?
- What does the 3D icon mean when you select a level or grid?
- Describe the rationale behind the default insertion function of structural members down. For example, when placing a concrete stem wall from level 1 to bottom of footing, or in the case of the columns in from level 2 down to level 1.
- Is there a difference between using the Align tool lock and placing a dimension, and setting it to 0" and then locking the dimension?

Unit 7

This unit introduces the concept of design information organization and gives you ideas for hands-on practice working with components, categories/subcategories, and families/types in Revit Architecture.

Theory: Design Information Organization

CAD users have become accustomed to organizing their projects by means of layers. Layers are a standard means of organizing graphical information in design tools from photo editing and illustration to CAD and engineering. Their absence in Revit Architecture may come as a surprise. Be assured, however, that this absence was not an oversight but a principled decision to replace layers with more powerful organizational systems.

To introduce the Revit Architecture system of components, groups, and categories, let's begin by outlining what layers are good for and then demonstrate how Revit Architecture moves beyond them with its own system of View Visibility Graphic settings and object styles components and subcomponents, as well as control offered by worksets and design options.

Layers enable you to group together objects that are conceptually related to each other. You can then manipulate properties of that layer to determine default characteristics of everything in that layer, such as color and lineweight. You can also keep them from displaying at all, which makes them particularly handy for keeping a complicated drawing simple while it is in process and then showing all detail at the end. It also opens them up to use as scaffolding devices, used to structure components and then be removed.

The "layer" metaphor on which layer systems are based makes the most sense in a 2D environment in which each layer can be composed on top of the one below it in a prescribed but changeable order. This kind of composition introduces an implicit quantized third dimension into 2D design in which layers are stacked and has no direct analog in 3D design. In 3D design, layers are purely organizational. Some implementations of layers are hierarchical, or at least allow for named layer groups, but most are simply linear.

Layers are extremely flexible and impose little structure by themselves. They are essentially syntactical devices. The semantics of layer use is entirely up to you. There are as many meanings for a set of layers as there are users of layers. Some users divide their work into layers depending on its material. This division can be handy because it is often possible to control material rendering properties on a per-layer basis. But then they lose the ability to organize their layers by any other criterion. Should they decide later that a different organizing principle would have made a better layer generator, they are in for a great deal of work. This is particularly harmful as a project proceeds from one stage of development to another. Often in an exploratory phase, a conceptually abstract set of layers is appropriate. Later when drawings are in preparation for output, a set of layers based on rendering details may be more useful. It is common for users to start entirely new documents and trace over their old ones just to instantiate a differently layered organization. Layers are simply inadequate to organize the many equally valid, and simultaneously operative hierarchies of building model information—spatial, functional, temporal, material, financial, and so forth.

Although you may initially miss layers, you will come to realize that this feeling is really just a symptom of mistakenly treating modeling as though it were CAD.

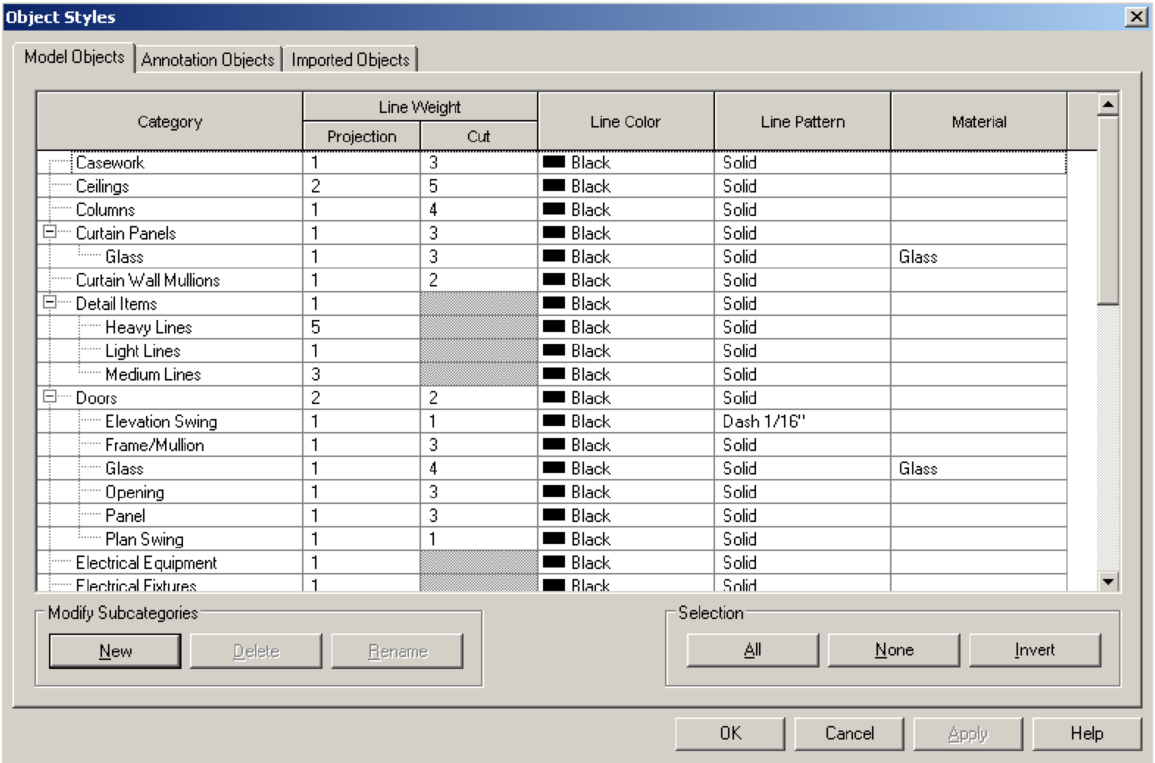
Consistency

One of the goals of BIM information organization is consistency. As mentioned earlier, layers impose no specific organizational methodology. This semantic freedom of layers introduces another problem: incompatibility of documents. No two users' layers match exactly. This introduces an enormous amount of labor in the translation of information from one set of conventions to another, often even within the same practice. For people to work on drawings created by others, they have to learn and adopt the originator's layer strategy or else spend a huge amount of time converting it to their own preferred strategy. Worse, the document may have been prepared by a user who did not know how to use layers at all (a common reality). These documents betray almost no organization whatsoever. Neither is this kind of mess always the result of ignorance. Setting up layers takes time and foresight at the beginning of a project. Often people are too hurried or the project is not yet specified fully enough to do it thoroughly. And since layering requires explicit work during drafting, mistakes are unavoidable.

An Alternative

In CAD there is little possibility for automatic organization. The software knows nothing about the geometry you are making to help organize it. However, when modeling, the software recognizes a great deal about what a designer is doing at a fairly high level, which makes possible the automatic organization of elements. In Revit Architecture you can organize objects in two primary ways. The first few units covered the Element>Family>Type>Instance hierarchical organization of the objects in the Revit Architecture building model. Revit Architecture provides another organizational method that is more specific to the display of the objects. This organization is held by object styles and gives you control over how objects are displayed.

Automatic organization requires a hierarchy of object categories and subcategories so that everything can be created into an existing abstract structure. The production of this structure requires a considerable amount of work, and if left up to users, you would again be confronted with the problem of nonportability. Revit Architecture defines such an organization, which you can add to but not modify.



Category	Line Weight		Line Color	Line Pattern	Material
	Projection	Cut			
Casework	1	3	■ Black	Solid	
Ceilings	2	5	■ Black	Solid	
Columns	1	4	■ Black	Solid	
⊟ Curtain Panels	1	3	■ Black	Solid	
Glass	1	3	■ Black	Solid	Glass
Curtain Wall Mullions	1	2	■ Black	Solid	
⊟ Detail Items	1	▨	■ Black	Solid	
Heavy Lines	5	▨	■ Black	Solid	
Light Lines	1	▨	■ Black	Solid	
Medium Lines	3	▨	■ Black	Solid	
⊟ Doors	2	2	■ Black	Solid	
Elevation Swing	1	1	■ Black	Dash 1/16"	
Frame/Mullion	1	3	■ Black	Solid	
Glass	1	4	■ Black	Solid	Glass
Opening	1	3	■ Black	Solid	
Panel	1	3	■ Black	Solid	
Plan Swing	1	1	■ Black	Solid	
Electrical Equipment	1	▨	■ Black	Solid	
Electrical Fixtures	1	▨	■ Black	Solid	

Revit Architecture categories and subcategories allow manipulation of the display of the objects in the model.

Within this organization, Revit Architecture uses the terms *category* and *subcategory* to describe the levels of structure. *Doors* is an example of a category, and *elevation swing* is a subcategory. It is this category/subcategory that enables the manipulation of visibility control used to create different representations of the building model.

Communicating Design Intent

Any given design has aspects that need to be conveyed to different parties involved in the building creation. What we need to present to the client is different from what we need to convey to the interior designer or to the structural engineer or contractor. Revit Architecture's category/subcategory feature enables us to present differing content in essentially the same view of the building model. In the workbook exercise, you work with a plan view with dimensions and tags that does not display the furniture. You create a plan view that shows the furnishings, but not the tags and dimensions.

Upside

This rigid organization has a strong upside. Clearly, the consistency aids in file exchange. But it also saves a great deal of time and energy that would otherwise have to go into creating such a hierarchy from scratch. It does not require users to waste time as they model setting the layers of objects. And a subtle advantage is that looking at the hierarchy can be didactic. You may or may not know that "L Corner Mullions" are commonly available. If you want to see what one looks like, simply place one in a curtain wall grid and take a look. This makes the category list somewhat like a building component catalog through which you can browse and select items.

Downside

Of course, rigidity has its natural downside—inflexibility. Is a table with built-in seating a table or a bunch of chairs? In Revit Architecture, if you create subcategories with this level of granularity, you cannot assign these elements to both subcategories. Although the category list is not modifiable, it is extensible. You can add custom subcategories to existing categories, but you cannot create new categories.

In addition, because it is extensible, you run the risk of subcategories getting out of hand the way layers get out of hand, with each user creating their own list of unique subcategories that makes sense only to them.

This shallowly hierarchical organization by object type is also the only organization that Revit Architecture allows. Fortunately, it is effective for many operations such as making schedules. But it does not easily enable you to group objects based on arbitrary properties. For this, Revit Architecture provides groups, nested families, or both, which is discussed in a later unit.

Other Implicit Organizations

Revit Architecture offers a few more options for organization of model elements. The first is common to most design tools—grouping. You can select a set of components and group them into a single entity. You can then copy this group or move it as a single entity. Groups in Revit Architecture can be named.

The power of organization by object type is in the fact that you can define object types narrowly. It is perfectly reasonable to create two different types that are graphically identical in order to be able to select them separately. Revit Architecture offers the ability to select all instances of a particular type, making type a convenient grouping mechanism.

Revit Architecture: Components, Categories, Subcategories

The Connection

We are going to explore the automatic design organization that replaces layers in Revit Architecture. To use Revit Architecture productively, you must become fluent with this organization and the terminology that surrounds it: Category/Subcategory, Family/Type.

In contrast to the hierarchical and visual taxonomy, Revit Architecture uses other organizational methods. These include selection sets and aggregations of objects called groups that you can manipulate collectively. Groups can be created in a Revit project and then exported as separate Revit files. Conversely, a linked Revit project file can be bound into the working project, becoming a group.

Revit Architecture Features and Concepts to Learn

Object Style dialog box
- Category
- Subcategory

Family browser
- Family
- Type

View level graphic overrides
- Category
- Subcategory

Object level graphic overrides

Revit groups
- Creation
- Editing
- Saving as a separate Revit file

Notes

- The categories are fixed.
- You may add your own custom subcategories to a category.
- Many of the subcategories are determined as part of a family definition. As you add a family into the project, it brings with it the subcategories that are defined within it. As you add more families, you see more subcategories appear in the object styles list. Reference Unit 11 for more information on subcategories in families.
- The categories are hard-coded into families. In the family editor, there is also an Object Styles dialog box where you can create your own subcategories, but you only have one category choice, which is determined by the starting point of the family—the family template or RFT file.

Hands-on Topics

- Using the concepts of categories, subcategories, and type as Revit Architecture defines them, describe a set of components not included with the program. What is the category? Is this category part of the Revit Architecture taxonomy? If not, why not? What are the subcategories? What are the types?
- Instantiate several different component types into a design you have been working on. Select a group by type and swap out the entire type. Now duplicate and change type parameters of the type. Notice all the changes that occur.

- Create a group and nest that group within another. What benefits does nesting groups give you?
- Create a group that has elements that are hosted outside the group (windows, doors). What problems does this inherently pose as you copy and mirror the group and how can you overcome them?
- Workbook Unit 7.

Questions

- What's another way to organize categories besides the way Revit Architecture does it? What are the advantages and disadvantages?
- Material settings are also available from the Object Styles dialog box but not from a view properties visibility graphics. Is this a hindrance or a good idea?
- Why does a door have an elevation swing category? In which cases would you want to show a swing? In which cases not?
- How does the Object Styles dialog box differ from the view properties Visibility/Graphics dialog box. What does each control?
- What happens if you open a family and add a component and assign it to a new subcategory? Where is the control over this new component once the family is placed in a project?
- Why use Dependent views?

Unit 8

This unit introduces the concept of domain-specific knowledge and gives you ideas for hands-on practice working with roofs in Revit Architecture.

Theory: Domain-Specific Knowledge

Rule-Based Behavior

Building information modeling implements complex, rule-based behavior such as intelligent wall and roof joining. The vocabulary and concepts built into BIM are tied specifically to the architectural and engineering profession. A user with no working knowledge of how a wall and roof join may not understand how or why a building information model is working, because the model assumes the user has a domain-specific knowledge base.

This gives the architectural design student two things to learn at the same time—both modeling and architectural practice. It is not common, for example, for students in early architectural design to consider foundations, but these are a necessary component for design in Revit Architecture if a user wants to place a building in a site. Students will be asked what kind of wall they want to use; they may not have even considered such questions when walls just seem to be two parallel lines.

Benefits

Time, energy, and good practice: Aside from the obvious advantage of saving time and effort, one advantage of this domain specificity is that it can implicitly enforce good design practice. There are things that Revit Architecture simply won't support. You can't place a door without a wall to put it in. Although this domain-specific intelligence can save time and effort, and reduce mistakes, it does come with costs.

Costs

Loss of generality: The most obvious detriment to domain-specific behavior is a loss of generality. Although CAD packages are typically suited to different domains primarily by the formal languages they support (splines versus meshes, for example), Revit Architecture is tightly bound to architecture. It would make no sense to try to do industrial design of a computer peripheral in Revit Architecture. All of Revit Architecture software's knowledge-based intuitions would be wrong. There is no "roof" on a mouse. Not everyone considers this narrowness a detriment. Other disciplines have tailored software, why shouldn't architecture? Architects probably feel they deserve software designed to handle the kinds of things they do daily at the expense of generality. But it becomes an issue if there are components of a design project that are not strictly architectural. If there is a fabric to be designed in conjunction with a project, Revit Architecture will not easily help you.

Exceptions: A more subtle problem can arise whenever you want to introduce an exception to a rule that Revit Architecture has applied. If you want to have two walls meet at a corner but not to join structurally, you have to tell the wall explicitly to not join with others. This is a loss of immediate low-level control. It is a sacrifice made to ensure that the most common controls are background tasks automated by the software and is the same kind of control that drivers of stick-shift cars enjoy over automatic transmissions. This kind of automatic action appears in much commercial software, most infamously Microsoft® Office products, which introduced the dancing paperclip that tried to intuit users' formatting intentions. The backlash that this obtrusive form of automated intelligence engendered serves as a caution to designers of software intelligence. No matter what steps software allows you to skip, it needs to allow you to retrace them one by one and change the parameters minutely. The balance between high-level ease of use and low-level control is not easy to achieve.

More to learn: Another problem with domain-specific software manifests itself in the ever-increasing sets of limited-use tools as features. Revit Architecture, for example, has separate design tools for roofs, ceilings, stairs, and curtain walls. They all have certain commonalities, but they must be learned individually because they all have different behaviors and rules. It may not be possible to fully enumerate all the rules that even one such tool embodies, let alone five of them. This can lead to a feeling that the software is unknowable and uncontrollable. It can become unclear exactly what the consequences of your action will be. In a parametric design tool like Revit Architecture, not all consequences are even visible. The same plane that appears to be in the correct place may have an unintended implicit attachment to the location of some other element.

Special cases: These niche tools also introduce myriad special cases in which some serious intelligence must come to bear. What should happen when two pitched roofs that start at different levels meet? There is more than one possibility. You have to try it in Revit Architecture to determine which one the program supports. The truth is that there are an infinite number of special cases like this, some without obvious resolution, that no amount of software intelligence can untangle. The decision to take on the problem of interpreting design intent is based on the faith that there is space enough to support a full range of design expression.

Enforcing dominant paradigms: A final, and perhaps most insidious danger of design intelligence is that it enforces dominant paradigms and discourages experimentation. Anything that pushes the boundaries of what is possible, design software that conforms strictly to existing practice has trouble supporting. It is also true that the components in the category list are ones that are known to be readily available. The more exotic the component, the less likely it is to be represented. Although it can be argued that this does not represent a bar to creativity in the same way that meter and rhyme do not strangle poetry, it does mean that there is a natural inclination to tread well-worn paths. From a practical and budgetary standpoint, this is probably great news, but for students who are struggling to find their voices, often in opposition to standard practice, it may seem unduly confining. Although in general Revit Architecture is based on the dominant paradigms, it does provide mechanisms to break away from the standard. Solid/Void forms in the software's Building Maker allow great flexibility of form. Building elements such as walls, floor, and roof are then applied to the faces of the form massing. In addition, in-place families allow for quick, on-the-fly generation of components at the edges or beyond the dominant paradigms.

Revit Architecture: Roofs

The Connection

The apotheosis of Revit Architecture software's domain specificity is its roof creation system. Nowhere (except perhaps in stair making) is such geometric complexity managed by so few parameters. As any user of CAD knows, roof geometry is painful to manage. It is in roofs that compound angular calculations spring up. More than one design student has chosen a flat roof for a project not because it was the best choice, but because it was easy to model.

Revit Architecture attempts to make most standard roof types possible to create and edit easily. It has been quite careful in the control that it offers and the control it reserves so that it can manage the roof geometry well enough to integrate with the rest of the building model. For example, walls that so specify terminate at the roof no matter how the roof flies. Intersections of different roofs and roof types are handled automatically. When Revit Architecture does what you want it to, the roof system is delightful, and when it doesn't, it can be exceedingly frustrating.

Revit Architecture Features and Concepts to Learn

- Roofs
  - Extruded roofs (Unit 6)
  - Footprint roofs
  - Cut plan profile
  - Openings
  - Slope-defining edges
  - Joining walls to roof
  - Joining roof to roof
- Advanced roofing
  - Hipped roofs
  - Round roofs
  - Slope arrows
  - Soffits, facias, and gutters
- Using in-place roofs to move beyond the traditional concept of a roof
- Low Slope Roof tool

Notes

- Slope arrows can also be used for sloping floors such as ramps.

Hands-on Topics

- Describe the difference between roofs by extrusion and roofs by footprint. In what cases will you favor one over the other?
- Create a simple design with an exotic roof. This might contain a roof by footprint meeting a roof by extrusion. To what extent can you change the originating form while the roof remains valid? How can you fix it?
- Create a building with exterior walls of type Brick on Metal Stud. Make a roof of the steel truss type. Create a section through a roof/wall connection and set its view to medium detail. Explain why the different components clean up or don't clean up. Change the roof type to the different types provided in a template, and explain the differences.
- Create a simple roof. Apply a sweep to several of its edges. Load several more sweep profiles (*Profiles>Roofs* folder). Create new types based on these new profiles and try them on different roofs.
- Workbook Units 6 and 8.

Questions

- What kinds of roofs can Revit Architecture produce with existing tools? What kinds can't it? (Use in-place families to model them.)
- Look at the properties of a roof. Is there any domain-specific terminology you don't understand?
- If you change the geometry of a roof with sweeps assigned to its edges, do the sweeps adjust appropriately? Discuss the implications of the workflow process from general to specific.
- What are the differences between using the Opening tool to place a hole in a footprint-type roof and adding a closed shape to the interior of its sketch? When would you use one versus the other?

Unit 9

This unit introduces the concept of delaying specificity and gives you ideas for hands-on practice working with massing in Revit Architecture.

Theory: Delaying Specificity

Too Much Too Soon

A common criticism of computerized design tools is that they require too much specificity too soon. But properly applied, parametric and BIM techniques can help you start quite generally and proceed gradually to higher levels of detail.

Overprecision

The argument tends to hinge on the fact that on the computer everything must at some level be specified quite precisely. In a paper sketch it is quite easy to indicate that a mark is not precise. It can be drawn lightly, shakily, or may trail off. Scale does not have to be precisely determined. There are all kinds of visual cues that indicate sketchiness. These cues are not easily available on the computer. A line looks quite precise and final no matter what the design intention.

Modeling requires the specification of much more information than drawing. It is reasonable to be concerned that all that information, specified at an early stage of design, may be prematurely limiting. Do you have to specify every building detail at once? How, for example, can you remain sketchy while still knowing the exact position of every leg of every chair in the design?

In addition, this early determination of information may convey a false sense of completeness in the design process. All the visual cues that indicate sketchiness are removed, and hence presentation of the model at even a late conceptual stage may promote the perception of completeness in the viewer. This can be good or bad: good if you are trying to convey a complete project, bad if you are still working out issues such as materials and textures. Often if you present a complete model with the intention of communicating rough massing and overall organizational ideas, design dialog will focus on materials and details instead of on the overall concepts. BIM modelers can use many different techniques to convey just the information they want to present. Revit Architecture provides view-level detail control, mass elements, and color fills among other tools to present just the information needed.

Roughing, Then Refining

Parametric design naturally supports a technique for the delaying of specificity until design elaboration stages. The idea is to start by placing generally defined components and later on switching them out for more specific versions. When you start Revit Architecture and draw a wall without specifying anything else, the default type is *Generic—8"*. That is clearly not actually a kind of wall. It is a placeholder for a wall type to be determined later. The 8-inch thickness specification is the bare minimum necessary to represent the wall properly in various views. So if you start Revit Architecture and decide to begin a conceptual design in plan, you can simply place walls without caring particularly what type of wall they represent. Later you can simply select walls and choose a different wall type from all the loaded types, and the model adjusts accordingly, even if the thickness changes. Generally, most types can easily be replaced with any other type from the same category. You can feel free to place a door in a wall without worrying at the outset what kind of door it is or how it should swing. That’s all modifiable later. The same is true of level heights: pick a number, and start designing. Level heights can be adjusted later.

This relieves the pressure to know all the correct types of objects at the time of placement. Notice that it is specifically the object-oriented parametric nature of BIM design that allows this deferral of specificity. In traditional CAD, you had better make a pretty good guess about wall thickness before you start drawing or else you're going to spend as long or longer redrawing when that specification changes.

Revit Architecture: Massing

The Connection

Revit Architecture has a group of tools to enable delayed specificity. This group of tools is collectively called Building Maker. With Building Maker you start by creating a massing model. The massing model consists of solids and voids. To the massing model you assign walls, floors, curtain systems (curtain walls), and roofs to faces of the massing.

Building Maker is a designed to tie together your massing studies and the walls, curtain walls, floors, and roof that make up the building. It has its own tools and relationships that enhance the process flow from conceptual to schematic design, allowing for a cumulative understanding of the relationship between expressive and built form as the design develops.

Building Maker features include the following:

- Curtain systems
- Floor by face
- Roof by face
- Wall by face
- Mass Editor
- Reusable mass families, including nested mass families
- Multiple mass instances that can be assigned to any workset, phase, or design option
- Schedulable mass properties, including gross volume and floor area
- More flexible creation and associativity of building elements to mass instances

The Building Maker tools enable you to start working in a sketchy way and add building components as you determine them. This capability allows a great deal of delayed specificity and enables you to start working on a project from preliminary information. You also have several options in how you bring that massing information into the rest of the modeling process. At its most integrated, the massing model becomes the generator of the building shell. This is a dream for any CAD user who has thought, “Why can’t I just draw a box and have it generate walls and floors?” With the Building Maker tools, you can.

Revit Architecture Features and Concepts to Learn

- Massing
  - Element associativity
  - Extrude
  - Revolve
  - Sweep
  - Blend
- Solids and voids
- Massing workflow
  - Levels
  - Reference planes
- Show Mass tool
- Mass as scaffold
- Mass as shell
  - Shelling logic
- Show mass/shell
- Working with building elements
  - Walls, floors, roofs, curtain systems by face

Remake function

Import of SketchUp files

Notes

- The mass is essentially a high-level sketch that acts as a geometric control parameter for the shell.
- After creating a massing and then creating components by face, you cannot change the associativity between the massing and the created components. There is a toggle on the Components Properties dialog box that alerts you that these components are related to mass, but you cannot uncheck this box. Components generated from faces remain related to those faces until the mass element is deleted from the project.
- Show Shell is automatically turned on when you create a mass but is not necessarily on if you save, close, and reopen the project.

Hands-on Topics

- Import a site map of an existing place. Produce a massing model that dissects the site into separate regions. Articulate some of the regions into building enclosures, and generate components from their faces. Now slightly change your region separation logic. What happens to your components?
- Workbook Unit 9
- Load some of the massing families (Family Library>Mass) and use them in a mass.

Questions

- What rules does Revit Architecture use to create components based on faces? How does it translate a void in a mass into a hole in a wall?
- How do the imported family masses differ from the in-place masses used in the workbook exercise? What are the advantages and disadvantages of these two types of mass objects?
- Describe why you can use the Building Maker tools on the mass with the nested SketchUp model, but not on the SketchUp model linked directly into the project.

Unit 10

This unit introduces the concept of component design and gives you ideas for hands-on practice working with the Family Editor in Revit Architecture.

Theory: Component Design

Designing custom components with their own parametric behaviors is one of the more complex topics of building information modeling. All components originate from somewhere, and understanding how they work, and how to add parametric control to custom content, is an integral part of successful modeling. In general, components are built outside the project, and then loaded in. Thus, a separate, but consistent work environment needs to exist for making the objects that eventually end up in the project. In Revit Architecture, this environment is called the Family Editor.

Going Beyond the Premade

Revit Architecture comes with a wide variety of premade types in many categories, but at some point you may want to make your own custom types. These types can be new kinds of furniture, plants, light fixtures, anything. It makes sense to define a custom type if you think you are going to want to instantiate multiples of something, particularly if the instances vary in certain predictable ways. These points of variance become the type's parameters. Although there is a difference between a type and a family, there can be no type that does not belong to a family, and in Revit Architecture, every family defines at least one type. So defining and modifying types tends to be referred to as editing families.

Modeling Variability

Making a new family is considerably more challenging than simply producing geometry. All parametric variation that the family allows must be explicitly modeled. The tools needed to produce families are quite similar to those used to model with them once they are defined, but they must be more powerful in several areas. To understand the tools used to build a family, you should first think about what you need to define to specify a type completely. To make this more concrete, let's define a new family of table.

The Challenge

Modeling parametric components is sufficiently different from modeling 3D form that it takes practice even for experienced 3D modelers to get used to. And even once they do, it is common for them to have to make several passes at a design component before it is parameterized correctly. Early structural errors tend to compound. Although this seems like a lot of work, it should be compared to the work involved in creating each potential variation that will ever be deployed by hand.

Predesign

What the complexity of these components implies is that there is a necessary redesign stage before an attempt to realize a custom component in software. Presumably at the start of this stage, a default 3D form of the component is well understood. If not, some formal research is necessary to determine an efficient way to produce the form geometrically. The next step is to determine what properties the component has that should be open to variation. This step can be quite challenging because any real component has essentially infinite variability. You have to select only the most productive axes. You can consider these the criteria by which you would select this component from a matrix in a printed catalog (thicknesses, lengths, materials, and so forth).

Specificity Versus Wide Applicability

The goal is to make the component both widely useful but specific enough not to change identity. A structural member, for example, should probably not allow so much variability that it can be used as both a plate and a beam. The long-term benefits of more specific components are that they retain reference to their type and can therefore be changed and elaborated en mass. It is unlikely that you would want to apply the same subsequent operations to two forms of a component that vary so widely that they no longer have any family resemblance. In fact, the use of the term *family* for a type with a group of related parameter sets is quite apropos. They should be a family, and no amount of parameter tweaking should be able to produce a component that does not belong.

Intrinsic Versus Extrinsic Properties

The preceding guidelines help in determining a component's intrinsic properties. As a next step it is worth considering what extrinsic properties are necessary. Rather than determine formal or material properties of the component, these properties specify where and how the component integrates with others. For example, in Revit Architecture, a window instance specifies not only its own dimensions, but also its sill height relative to a starting level. Why should a window store its own relative height? That has no direct bearing on the window's dimensions or form. The idea is that if you want a window instance to have numerically controllable height, you should expose it as a parameter. Later you can automatically set the heights of all windows of this same type to the same value, which has the effect of aligning them on each level. It is worth considering making a separate window type of the same form as an existing type if you know you will use several of them at a different height. That way the height of each type can be controlled centrally.

Modeling Invariance

A final predesign step is to determine the fixed relationships between the parameters you have specified. Which of them are tied to others in specific ways? Some of these may seem too obvious to specify, but recall that the machine lacks any sort of human intuition for what is an acceptable condition. Consider a dresser with short legs. What should happen when a user adjusts the overall height of the dresser? Should the whole form scale upward, lengthening both the legs and the body, or should the legs stay fixed at their initial length as the body grows and shrinks? These are the kinds of questions that require resolution at this stage. They cannot be avoided because when a component is modified, it must by definition exhibit some sort of behavior. If you have not applied design intent to this behavior, it is going to be arbitrary and unlikely to be helpful.

Complexity

A final set of concerns to apply to parametric component design is the level of complexity you want to support. It is possible, in fact common, to nest components. You may make a component that consists of nothing more than a grid of other existing components. This kind of hierarchical relationship allows for the creation of highly complex components. It is easily possible to specify so much complexity in a common component that the software bogs down. Avoid this temptation toward completeness. Although BIM design hinges on the idea of a complete building model from which specifications can be read, the level of detail must stop somewhere lest you end up with models that are so complete they are not useful. As Martin Fowler put it in his book on modeling for engineering, "Comprehensiveness is the enemy of comprehensibility" (*UML Distilled, 3rd Ed., 2004*).

If you design a custom doorknob and place it into your model as the default for a specific door type that you use everywhere, the software has to produce and manage the geometry for this doorknob everywhere it appears, which is likely to be a waste of computational power no matter how nice the doorknob is. The best way to handle this kind of situation is to model the knob separately, note textually or in a simple diagram that the door contains it but not to model it into the door, or to substitute a geometrically simple placeholder such as a box in your door component design. Every piece of geometry in a Revit Architecture family has visibility parameters that control when the geometry is visible. This capability enables control based on the type of view (3D, plan, elevation) and detail level (coarse, medium, fine). Your custom doorknob, for example, could be included in the door family but specified to be visible only in fine views.

Kinds of Parameters

It is important to understand the breadth and variety of parameters you can bring into play for a component. Every parameter has a type, which constrains the kinds of values it is allowed to take on. The most common parameters are dimensional. These store simple linear values. Areas and volumes are the results of several one-dimensional measures instead of being specified directly as square or cubic unit parameters. Another kind of parameter you can use is a material. This may not be a parameter that you apply mathematical operations to, but it can be a parameter nonetheless. A third, and tremendously useful form of parameter is the Boolean value. A Boolean value can be either true or false, but nothing else. It is used to model discontinuous variation based on a condition. If a certain condition is met, then do one thing; otherwise do another. An example could be a component that is modeled as a ramp until its inclination is made too high, at which point it becomes a stair.

All the preceding considerations are quite general. They apply to the construction of parametric components, from architectural building components to software systems. The actual process of doing the parametric modeling, however, is somewhat involved and domain specific, and its details are quite particular to the software that implements it. Revit Architecture uses an interesting model that exposes parameters by means of dimension measures.

Errors

The intricacy of parametric modeling makes it unsurprising that some of the errors it can lead to are subtle and difficult to track or remedy. Experimenting without foresight with the parametric tools undoubtedly leads to error messages saying that constraints cannot be satisfied. That is because it is easy to introduce conflicting constraints by way of relationships that are satisfied in some conditions but not in others. For example, you could make a triangular form, constrain one angle measure and also the side length opposite it. Now if you try to change that side length, the angle constraint can no longer be met. The program complains.

Overconstraint

Overconstraint errors do not appear as errors until an impossible condition is reached, at which point the program complains, and when it does, the offense may be deeply buried within layered dependencies. The problem is that there is no way to know when this will occur, if ever (although you can bet it will happen when you’re late on a deadline), so it behooves the designer to pick relationships carefully and judiciously so as not to overconstrain the model.

There is also the likelihood that some values your parameters could take on would lead to impossible geometries. If in the table example earlier, you specify a leg height taller than the overall height, what should the result be? Probably you’d like to be notified that an impossible condition has been requested and offered the opportunity to change the offending values. Revit Architecture comes close to this functionality.

Revit Architecture: Family Editor

The Connection

The Family Editor is like a private workshop separate from the rest of the model for working on custom components. The Family Editor is where you create and test design pieces that will be placed into the model. This tool is commonly used for repeating elements with certain parametric variations. Cabinetry that does not match the default types shipped with Revit Architecture is a good project for the Family Editor.

The Family Editor positions users as more than the client of premade parameterizations. They become the author of implicit relationships that they or others can subscribe to. It is a powerful role, and becoming familiar with the Family Editor is the first step in understanding the full potential of parametric design. Users who are frustrated with what they perceive to be limitations in Revit Architecture are often not yet comfortable with the Family Editor.

Revit Architecture Features and Concepts to Learn

- Family composition
  - 3D geometry tools
    - Solids: Extrude, Revolve, Sweep Blend
    - Voids: Extrude, Revolve, Sweep Blend
  - Symbolic lines
  - Visibility control
    - By visibility property
    - By visible property
    - By subcategory
  - Reference planes and reference lines
  - Aligning with constraint, without constraint
  - Dimensions
    - Lables (parameters)
    - Static locks (explicit constraint)
  - Text and model text
  - Parameters
    - Types
    - Formulas with “if, then” logic
    - Implicit parameters
    - Naming
- Family creation process and flow
  - Getting into the Family Editor
    - Stand-alone
    - From within a project
  - Family component templates
- Design process
  - Sequence: ref lines, dims>parameters, flex, solids/voids, flex, symbolic lines
  - Draw first, parameterize later
- Edit sketch and dimension
- Flexing the model
- Working with families
  - Loading and reloading families
  - Editing families that have been placed in a project
  - Deleting families that have been placed in a project
  - Making types in the family
  - Making types while in a project

Notes

- Always lay down a framework of reference planes first.
- Lock the geometry to the reference planes.
- Assign the dimensions to the reference planes and not the geometry.
- Flexing the model: Once you have made a component, you need to modify its parameters and pull on its edges to see how it reacts. Often, a component does not behave the way you think it ought to. If it doesn't, you need to add or change the parametric relationships. This process is called "flexing the model," and it is an essential step. Flex early and flex often. This is the best way to keep from overconstraining your family.

Hands-on Topics

- Create a custom wall sweep based on a newly defined profile type.
- Build a set of parameterized furniture, including chairs with and without arms, a table, and a desk. Vary the parameters to see what kinds of variations you can produce. Store the most successful variations as types.
- Create a family from a component template. Create a similar object using in-place families. What are the differences?
- Workbook Units 10 and 11.

Questions

- What do the component family templates offer? How do they differ?
- Explain why there are no wall or roof templates.
- Are there things in the templates that you don't understand? What are they?
- Compare and contrast template-based component families and in-place families. Why would you use one instead of the other?

Unit 11

This unit introduces the concept of propagation of constraints and gives you ideas for hands-on practice working with alignment, locking, and constraints in Revit Architecture.

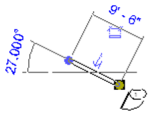
Theory: Propagation of Constraints

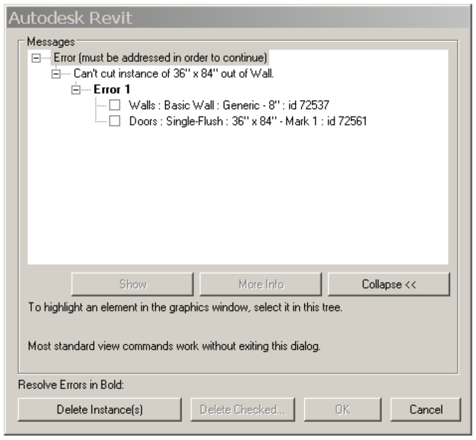
You have seen that parametric design allows for the imposition of constraints on objects' parameters. We are now going to look in more depth at the kinds of constraints that are possible and how those constraints propagate through a model.

Validity Constraints

Certain conditions must hold for objects to be well defined. Lengths should stay at or above zero, for instance. These constraints are implicit limits on parameter values and do not have to be explicitly specified. When they are violated, you are notified that the last transaction failed (see the databases topic for a definition of transaction), and the modification that created the bad condition is rejected. It can be hard to determine the root cause of a validity failure because the modification driving the design change may be far from the site of the consequent failure. It may take some time walking up the dependency tree to find the culprit.

For example, the error at right was generated because a door was placed in a wall and then the wall was made shorter than the width of the door. The relevant part of the error report is, "Can't cut instance of 36" x 84" out of Wall." The system can't cut a door out of a wall that is narrower than the door. This is a clear validity constraint violation. In this case, the cause is clear, but it is easy to imagine far more complicated cases in which it is quite difficult to debug a design. Clicking Cancel here undoes the attempt to shorten the wall.




Autodesk Revit

Messages
- Error (must be addressed in order to continue)
  - Can't cut instance of 36" x 84" out of Wall.
    - **Error 1**
      - □ Walls : Basic Wall : Generic - 8" : id 72537
      - □ Doors : Single-Flush : 36" x 84" - Mark 1 : id 72561

To highlight an element in the graphics window, select it in this tree.

Most standard view commands work without exiting this dialog.

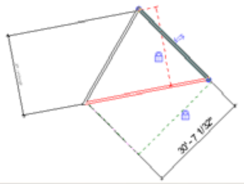
Resolve Errors in Bold:

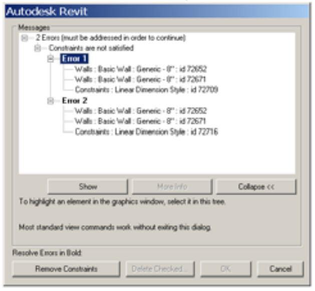
Constant Constraints

A constant constraint is a parameter set to a value that cannot change. Revit Architecture makes it quite easy to constrain dimensions, linear or radial, to a constant value. Overconstraint is an issue in the Family Editor, and constant constraints are an easy way to produce overconstraint in more basic modeling.

Unless the program has explicit rules for changing an object's parameter value, all unbound properties are constants.

In the example to the right, overconstraint errors resulted from constraining dimensions on two of three sides of a triangle (notice the blue lock icons) and trying to move the third. It is clear that such an edit cannot work. One of the constant constraints would have to be violated. Again this is an easily debugged case, but setting up more constraints than is absolutely necessary is a good way to get into trouble down the line.




Autodesk Revit

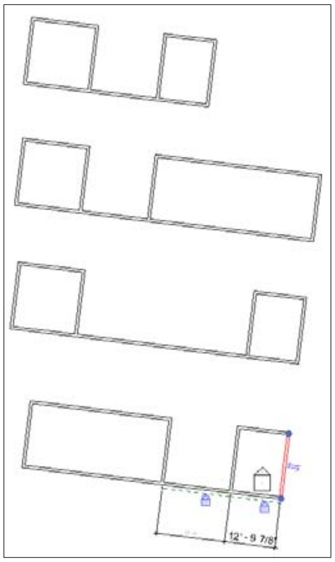
Messages
- 2 Errors (must be addressed in order to continue)
  - Constraints are not satisfied
    - **Error 1**
      - Walls : Basic Wall : Generic - 8" : id 72652
      - Walls : Basic Wall : Generic - 8" : id 72671
      - Constraints : Linear Dimension Style : id 72709
    - **Error 2**
      - Walls : Basic Wall : Generic - 8" : id 72652
      - Walls : Basic Wall : Generic - 8" : id 72671
      - Constraints : Linear Dimension Style : id 72716

To highlight an element in the graphics window, select it in this tree.

Most standard view commands work without exiting this dialog.

Resolve Errors in Bold:

Underconstraint



The sneaky cousin of overconstraint is underconstraint. The software never reports an error from underconstraint, so how do you know you have encountered it? Underconstraint becomes obvious when you make a change to a design and something else in the design changes or doesn't change in an undesirable way. Constraint is one of the most powerful tools parametric design offers to control the behavior of a design in the presence of change. If something changes counter to your wishes, then you need to constrain it in a way that makes such a change impossible. For example, all three lower designs on the right are modified versions of the top design, in which the right wall has been pulled farther to the right. The differences are all of constraint. In the third one down, the width of the room is constrained on the right. In the bottom one, both the room width and the distance between the rooms are constrained. If the left room width were also constrained, this change would result in an error of overconstraint. But if the second design from the top is not what you were looking for, your design would have been underconstrained relative to the desired change.

Propagation of Constraints

As is evident from these examples, constraints cause changes to propagate. One constraint's satisfaction may require a change to an entirely different section of the design. The cascade of changes that a single change kicks off can be difficult to predict, particularly if constraints are in place in other parts of the design.

Bound Property Constraints

Bound properties are tied directly to each other. If one takes on a new value, so does the other. This could be expressed mathematically as A = B + (constant). This is a common and useful kind of relationship. It is easily reversible, that is, if one of the parameters changes, it drives the other, no matter which direction the link was made. A parameter can just as easily depend on two or more other parameters as one. Say A = B + C. As soon as this is true, however, you lose the reversibility of the relationship. It is clear what the consequence of changing B or C is (simply recalculate A). But what should be the result of changing A? Should you change only B, only C, or half each of B and C? There is no correct answer. This is an inherent ambiguity in such uninvertable functions.

Directionality

This means that a relationship like A = B + C implies a directionality. B and C drive A. A may change, and depending on what B and C represent graphically, there may be a convention that helps describe what the obvious consequence to each is when their sum changes. If B and C are two lengths of plank that together sum to the length of A, perhaps it makes sense to change B and C when A changes, retaining the ratio between B and C. That is a well-constrained, solvable problem.

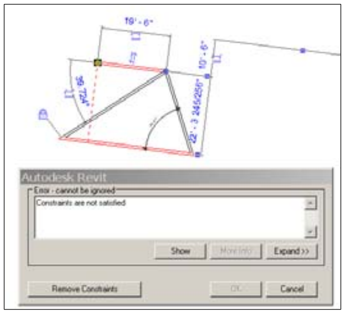
Statements of parametric constraint are the subject of a great deal of thought in computer science that falls under a few names, including constraint solving and declarative programming.

Declarative Programming

Placing constraints in a parametric design is a kind of declarative programming. Declarative programming gets its name because all the statements in declarative languages are statements of invariant truths (such as A = B + C). The language then is itself responsible for finding a solution set that satisfies all the specified constraints. For instance, if you were to set A to 6 and ask for the value of C, you would hope the declarative language system would be good enough to say, C = 6 – B, which doesn't give you a value for C, but does let you know what relationship the specific value of A has implicitly established between them. It has been proven that declarative programming is capable of producing any algorithm that can be programmed on a computer, but many algorithms would be hard to write this way. The proof that you can doesn't mean much.

Constraint Solving

The action the program takes to satisfy all the constraints the user has specified is called *constraint solving*, and it is a complicated business. In fact, it is pretty much impossible to do it perfectly. Given a sufficiently complicated set of constraints, you can probably make edits that are technically possible to achieve but that the software's solver cannot find. The case to the right is a small example. There is a simple solution to the requested edit, but the software fails to find it. This is not to blame Revit Architecture. General geometric constraint solving is a monumentally difficult problem, and Revit Architecture often performs nobly.



Constraints are not satisfied

Implicit Constraints

Revit Architecture makes the designer's life easier by setting up a variety of implicit constraints based on guesses about design intention. When you draw a wall that hits another at an endpoint, Revit Architecture sets up a corner constraint. In the figure to the right, the only thing that changed in the bottom copy was that the bottom wall moved. The other wall extended automatically to maintain the corner constraint. This is an implicit constraint because it was never manually specified. Implicit constraints are harder to control than explicit constraints because they are not made visible to modify.

Cyclic Dependencies

It is easy to introduce constraints that are impossible to satisfy because they are cyclic. As a simple example, suppose you say constrain "x" so that it is equal to "y + 1." Now you say constrain "y" so that it is equal to "x + 1." This constraint cannot be accomplished because the definition is cyclic. Y's value depends on x's, and x's depends on y's. There is no solving such a system. If constraints are applied without care, such a situation can easily arise in parametric modeling.

Revit Architecture: Alignment, Locking, and Constraints

The Connection

You saw earlier that the theoretical basis of parametric relationships was the establishment of invariant conditions. Revit Architecture exposes a means to define invariant constraints of component alignment and dimension through a visually obvious system of locks. All dimensions, be they linear or radial, display an open padlock, which can be closed with a click. Once so locked, Revit Architecture does its best to maintain that dimension. If you try to make an edit that necessarily violates a previously imposed constraint, Revit Architecture issues a warning and refuses to allow it unless the constraint is removed.

Underconstraint leads to models that allow too much variation and are not intelligently responsive to change. Overconstrained models are brittle and generate error messages whenever they are modified. With practice, you will find the happy medium that allows you to create efficient building models that are both flexible and responsive.

Revit Architecture Features and Concepts to Learn

- Dimensions
  - Locking dimensions
  - Equality dimensions
  - Removing dimensioned constraints
- Alignment
  - Locking alignment
- Angular constraints
- Chains of propagation
- Family Editor
  - Weak reference
  - Strong reference

Notes

- The Family Editor sometimes cannot find existing solutions to parametric variation. You have to reparameterize in a different way or try a different variation.
- A lock is displayed immediately after you use an Align tool. This lock disappears once any other action is taken.
- Inherent locks:
  - There are many instances of inherent constraints. If you draw a sketch line on top of a reference line, a lock automatically appears, enabling you to lock the sketch to the reference plane.
  - If you use the Pick tool (as opposed to the Draw tool) as you are creating the symbolic lines, a lock appears.

Hands-on Topics

- Create a design change propagation machine. Make a change to the start side and watch the changes propagate. What happens if you change the “output” side? Can Revit Architecture solve the inverse problem? Can you redesign the propagation machine in such a way that allows you to change the output side?
- Design a small building that changes its glazing with the sun angle. It’s okay if the physical mechanisms required to realize the design are nonexistent.
- Start a component family and add two more reference planes. Create a solid form extrusion. As you create the sketch lines, place them over the reference planes, but do

not lock them. Add dimension parameters and finish the sketch. Flex the family. Are the locks needed? Why or why not?

- Workbook Units 10 and 11.

Questions

- What kinds of relationships can you express with the constraints Revit Architecture offers? What can't you?
- The notes mention two different kinds of constraints that appear automatically when you are adding lines in sketch mode. Revit Architecture anticipates that the action will probably require constraints but does not assume these conditions are locked. What other actions have a similar effect?
- When you draw four walls and add a roof by footprint, the roof is constrained by default to the walls. In this case Revit Architecture assumes this to be a constrained relationship. Can you break this relationship? How? Name some of the other relationships Revit Architecture assumes to be constrained (in other words, does not show locks, but is constrained)?

Unit 12

This unit introduces the concept of interdependencies and gives you ideas for hands-on practice working with sites in Revit Architecture.

Theory: Interdependencies

A topic related to both establishing fixed references and delaying specificity is interdependencies. Designs achieve richness of detail in response to many different pressures: site, program, budget, and design rationale. All bear simultaneously on the finished design, but it is useful to be able to draw them into the design process slowly in whatever order you prefer.

Site, for Example

Take site, for example. Site may be a condition that needs to be present from the outset, or you may be able to experiment with a totally decontextualized space. Design tools must be flexible enough to enable you to add and change site independent of other concerns. Site modeling should not use exactly the same toolset as building modeling but the two should be related by parametric relationships.

Building Reacts to Site

There are any number of ways you can imagine building design reacting to site. Levels may be placed according to topography or surrounding buildings. Site has everything to do with the selection of a structural system. An entire design rationale can be built up from intense analysis of site and context. In this way a building model can be parameterized by site.

Site Reacts to Building

It is also true, however, that site is not unchanged by building. Building foundations excavate site extensively, and many proposals call for landscaping and road or parking creation. Forms on a site affect the light conditions of all surrounding buildings. So it is clearly not a one-way influence of site on building.

Directionality of Influence and Parametric Design

Parametric modeling is an excellent tool for interdependencies provided there is a strong directionality of influence. That is, if site is a strong driver of building form, it makes sense to model the site and then draw parametric relationships from that into the building design. In this case site pushes building form, but the building design does not enforce constraints back onto the site in a feedback loop. In principle, this kind of feedback loop is not destructive. In fact it exists to some extent in every real-world condition of influence, but to model it in software is more challenging. Most parametric designs are made to be driven in a single direction. Mutual influence is much more difficult to model, and it requires simultaneous forward- and back-propagation of constraints.

Conflicting Concerns

Beyond indeterminacy of influence, there is another challenge in necessarily conflicting concerns. In daily life we deal with conflicting constraints continually and mostly with grace. Real-world design problems are complex enough that they too must deal with simultaneous conflicting constraints. The role of the architect is, of course, to prioritize these constraints and resolve each to whatever extent is appropriate.

Soft Constraints

This kind of compromise is possible because most real-world constraints are soft constraints. They may have a distinct satisfactory value (“be here at 12:00”), but there is a scale of penalty for their lapse. (It is different, then, to arrive at 12:05 versus 1:30.) Even project budgets must be considered somewhat soft. If they were always strictly capped, there would be many buildings without roofs.

Expression

It is unusual for parametric design software to support this kind of soft constraint because the business of even expressing them quantitatively enough to describe them to a computer is challenging (“keep this angle close to 90 degrees”). Further, even if soft constraints are specified in a precise enough language for the machine, the problem of optimization that their simultaneous pressures imply is not always computationally solvable (particularly not in real time).

Beware Optimization

Different strategies such as “fuzzy logic” have been proposed to solve this kind of system of soft constraint. But the problem with their application to architecture is that the problem space (the set of all building forms) is too large and amorphous to support algorithmic optimization. In this arena, the human being is yet to be challenged by any machine intelligence. And anyone trying to sell you automated design “optimization” should be viewed with suspicion.

The Return of the Human

That is not to say that soft constraints have no place in computational design, but their role in practice is necessarily more analytical than generative. Given a set of soft constraints, you can evaluate any design proposal relative to axes such as thermal efficiency or cost. However, many factors remain—such as the overall experiential quality of a space—that no set of computational criteria can evaluate. That kind of evaluation must be left to humans.

Revit Architecture: Site

The Connection

An obvious place to begin addressing interdependent influences is with site, which often represents as much of a given as any other project constraint. Revit Architecture provides a full set of site tools that are fundamentally different from its building-design tools. However, the sites produced are model objects and can be used as parametric drivers of building design.

Revit Architecture Features and Concepts to Learn

- Toposurface
    - Imported
    - Drawn
    - Elevation points
    - Properties
    - Contours and labels
- Split/merge surface
- Graded region
- Property lines
- Building pad
- Site components
    - Parking
    - Graded region
    - Trees and plants
- Import and Export of AutoCAD® drawing format files
- Linking one Revit Architecture file into another

Hands-on Topics

- Design a small building whose levels are tied to site conditions. Change the site.
- Create a toposurface by picking four points. Set the point elevation to a higher elevation and add more points. Set the elevation to something off the contour interval, and add more points. What relationship is there between the points and the contour lines?
- Select and delete some points. How does the toposurface react?
- Place a tree on a toposurface in the site plan view. Move the tree in this view. Cut a section through the site that looks at the tree. Tile the section and plan views. Move the tree in each view. Does the tree stay associated with the toposurface in both views? Can you move the tree off the toposurface?

Questions

- How do the site tools differ from 3D CAD tools?
- What design intelligence does site have in Revit Architecture?
- Why is a property line an object and not simple linework?
- When would you split a toposurface versus creating a subregion?

Unit 13

This unit introduces the concept of unique and customized form, and gives you ideas for hands-on practice working with in-place families in Revit Architecture.

Theory: Unique Form

Usage Constraints

Building information modeling can be a proscribed process. Things must occur in a certain order and must conform to what the software can support. What if you want to add geometry (it may be as simple as a wedge-shaped wall) that the program does not natively support? The software must provide some mechanism for operating outside its built-in definitions or else it becomes too restrictive. Fundamentally, the idea is that when design reaches outside the program's area of expertise, the software's helpfulness should degrade gradually rather than simply not dealing with conditions its designers did not anticipate.

Decontextualized Types

A user's need to extend a BIM design tool's reach is mostly satisfied by the ability to create custom parametric components as previously outlined. But the creation of component types ought to be reserved for cases in which a single component or parametric variations are to be multiply instantiated. Their place of future instantiation is unknown at the time of their family's definition, which limits the amount of contextual information to which the type can be sensitive. For example, suppose that there is one geometrically complex wall in a project that curves back at the top so as to smoothly join a horizontal slab above it. The wall tools built into Revit Architecture do not offer such a custom wall. Generalizing this one-off is far more work than simply modeling the single instance in place.

Custom Types in Context

Fortunately, Revit Architecture enables you to use the formal tools directly in the context of your design, using existing objects as a reference layer. This technique is called an *in-place family*. Like drafting views, this technique should be used sparingly. Not only does it defeat all the design intelligence of the system, but it also calculates extremely slowly. Too much of this can lead to unwieldy models with no advantage over CAD.

When to Use an In-Place Family

It is sometimes difficult to tell when an in-place family is necessary. In general, it is better to avoid them as much as possible, since they negate much that is valuable in BIM design. They are parametrically variable, but they have no identity beyond their ad-hoc parameterization with which to influence automated design changes. Generally, use them when custom work is so specific that no part of it can be reused in another circumstance. It is worth thinking about various parameterizations of various aspects of the family to be sure that no part of it should really be a general parameterized component. When you are satisfied that your needs are truly unique, it is time to use an in-place family.

Design Considerations

As you create your in-place family, you have to define how it displays in each view in which it appears or you can create the 3D geometry of the in-place family and have Revit Architecture use that to determine the family's appearance in all views. This is no less work than the construction of a general custom type. At the least, your in-place family should vary smoothly with any conceivable change to its context. The practice of flexing the model that you learned when making custom components must be done here too if you have created constraints between the in-place family and geometry not in the in-place family. To do that, however, you may need to make experimental changes to your context to gauge the response of your in-place component.

In-Place Families Versus Drafting and Detail

In-place families provide an escape hatch in case the program does not support the desired form. Easier escapes are available in case a form simply does not display as desired in a specific view. These escapes are discussed in Unit 14.

In-Place Families Versus Generic Model Component families

Revit Architecture provides a component family called generic model templates. Using one of these templates, you can create one-off designs but retain the ability to drive types within the family. The trade-off with in-place families is that you do not have access to the types. However, you do have access to the entire building model as you create the geometry, enabling you to use it as reference as you create the geometry.

Revit Architecture: In-Place Families

The Connection

The Family Editor represented a kind of open workshop dissociated from model context in which to work on custom parameterized components. In-place families are similar except the workshop *is* the model. All the tools of the Family Editor are exposed, but they can be deployed in direct reference to the underlying model elements, giving you considerable geometric flexibility. This operation is profoundly different from creating a new generalized component type.

In-place families can be deployed multiple times in a single model. They can be copied and pasted into another project if needed. They should be considered one-offs, and they sacrifice one of the two tremendous powers of parametric design. They retain their ability to flex and react parametrically with the rest of the model, but they sacrifice their generic reusable quality. (You cannot define types for in-place families.) This may be totally appropriate. It may be that you need some particular geometry for a particular situation. If so, in-place families are the answer.

Revit Architecture Features and Concepts to Learn

Creating in place families
Parameterizing
Controlling view
Modifying geometry
In-place families interacting with the model
Adding inserts

Notes

- An in-place family is a family created in the context of the current project. The family exists only in this project and cannot be loaded into other projects, but can be copied and pasted into other projects. By creating in-place families, you create components unique to a project or components that reference geometry within the project. For example, if you need to create a reception desk that must fit between walls in a room, you could design it as an in-place furniture family. If the original design ever changed, the in-place family could be edited to change accordingly. Alternatively, you could constrain the in-place family geometry, as you created it, to the two walls.
- When you edit an in-place family in a project, you select the entire family first and then click Edit Family. This brings you into the Family Editor. To then edit individual elements

of the in-place family, select that element, and click Edit Sketch. You are now in sketch mode, editing that element.

Hands-on Topics

- Produce a wedge-shaped wall.
- Produce a barrel vault roof that turns a corner.
- Integrate both of these families into a design so that when the design changes the in-place family responds appropriately.
- Produce an in-place family that interferes with the geometry of your design. What tools does Revit Architecture offer to resolve this kind of geometric conflict?
- Create an in-place family that is constrained to elements in the building model. Are there any conditions you can create with this combination that you cannot create using the generic model family component and anchoring the elements of the family to the building model?

Questions

- The goals of parameterizing in-place families are different from parameterizing external families. How? What are the differences? What kinds of parameters are appropriate for each? (This can be framed as a question: “Whom are you serving with this component?”)
- Create several different categories of in-place families (wall, roof, and so forth). How do these behave differently from native instances of these objects?

Unit 14

This unit introduces the concept of details and gives you ideas for hands-on practice with drafting and creating linework in Revit Architecture.

Theory: Detail

The Limits of Modeling

As powerful an idea as a fully specified building model is, there must be a point at which the specification of detail ends within the building model components. You may place doors, and those doors likely have hinges, but to have those hinges appear in anything other than a tight detail of the door assembly is not only uninformative, but also harmful to the program's performance. Specifying too much detail on repeated components is a good way to make a large model so slow as to be unusable. Some high-end software systems do manage this level of detail with tremendously complex algorithms to automatically filter the information relevant to the current view from tremendously large data sets in real time. Revit Architecture has some such capacity in its ability to specify a level of detail per view, which filters what displays. But it is not capable of dealing with the kinds of data sets that are used to fully specify something like an airplane, which can easily run into the tens of gigabytes.

Level of Detail

Revit Architecture software's unsuitability to aircraft design is neither surprising nor inherently limiting, but it does force the user to decide the level of detail appropriate to model. A coarse but well-organized model can easily be elaborated simply by further detailing of its components. It is a good idea to start somewhat rough as described in the Delaying Specificity unit.

Graphical Overlay

The alternative to specifying fine detail as a full model component is to add it as a purely graphical overlay directly into a view. Doesn't this violate everything you have been told that makes BIM design beneficial? How, for example, does this overlay remain in sync with model changes? Well, it won't. That is why detailing of this kind should be done sparingly, only once in a representative detail callout to be referenced elsewhere, and late in the design development process. A distinction worth holding on to is that drafting work like this is done *onto* a view rather than *into* it as a model changes. Changes to the fundamental model are passed through the view back to the database. Although drafting work is indeed stored in the database, it is stored as a raw description of geometry. It doesn't show up in schedules, and it can't display in any other view except the one *onto* which it was drawn. Consider detail work like graffiti, spray painted over the top of a window looking at the building model (a view) to deliver a message. Or separate "static" views that can be referenced from callouts. A standard detail can live in a view where no part of "the model" will ever be present.

Back to CAD?

The tools available for drafting directly onto a view are the most similar to traditional CAD as any in Revit Architecture. That is because once you are drafting onto a view, you've already forfeited all the value of BIM design, and you might as well be using CAD for this local bit of drafting. These tools are nowhere near as sophisticated as the similar tools in a CAD program, and that is mostly purposeful. It is often better to figure out how to solve a modeling problem properly than to give up and resort to drafting, though if the results are going to be shown in one view only it can be more efficient to use the drafting tools.

Modifying Automatic Rendering

Another circumstance in which drafting is necessary is when the program’s default display of components in the selected view does not correspond to what you want to see. This can be a matter simply of lineweights, in which case Revit Architecture offers special tools to help. It may also be the result of an unusual condition that the Revit Architecture graphics engine does not treat the way you want it to, or simply a non-rule-based desire for a specific variance. You may want to show an outline of a component from another view that would not normally show up in this view just for registration.

If Revit Architecture consistently draws a component differently from your preference, you should take the time to modify the component definition itself (object style). This is a more systematic way to make the change. That way it propagates throughout all views.

Revit Architecture: Drafting and Linework

The Connection

If in-place families represent a partial release from the strictures of BIM design, drafting and linework represent its total abandonment (with the exception of Detail Components). Again, this is simply a caution. Often these tools are appropriate to use. But be prepared to reassume the mantle of reference-maintainer as your views become less closely tied to your model.

Revit Architecture Features and Concepts to Learn

Creating the detail view
Detailing tools
Tracing details from callouts
Detail components
Linework
Redoing details

Notes

- Use drafting views sparingly. The geometry you create in them is not a part of the building model.
- The drafting and linework tools are similar to CAD tools but are based on parametric use. In some cases they are not nearly as powerful as, say, tools in AutoCAD software. Because of their parametric nature, however, they surpass standard CAD tools in other ways.
- There are two major differences in detail views. A section or callout tool creates a new view that displays the building model. A drafting view does *not* display the building model. It is purely a space to draw 2D linework and detail components.

Hands-on Topics

- Produce the best-quality section of one of your designs as possible, using custom linework and drafting details if necessary.
- Many architecture magazines include details of the projects they are featuring that month. Duplicate one of those.
- Create a drafting view in a project. Load several detail components from the family library. Create a detail with these components.
- Workbook Unit 14.

Questions

- When in your experience so far have you needed these tools?
- Many of the detail component tools use parameters. Like the shelf unit built in a previous exercise, how do these parameters control length, width, and depth of the subcomponents in the detail object?
- How do the parameters found on a detail component family differ from those on a building model component family?

Unit 15

This unit introduces the concept of sequences and gives you practice designing as you would build in Revit Architecture.

Theory: Sequence

Construction sequence is rigid, as you have learned either through experience on site or in a class on materials and construction. Nothing is done out of order. You can't build Level 3 before Level 2. CAD doesn't much care about sequence. You could start your model with the roof and build down. Part of the exhilarating freedom of computational design is the total lack of physical constraint. Questions of a design's structural validity must be left to a secondary analysis stage or farmed out completely to engineers.

Building information model design, in contrast to CAD, cares about sequence. It brings back sequence as a primary force in model construction. The sequence may not strictly follow the future construction sequence (it would not be advised on the job site for instance, to pour solid concrete walls first and then punch holes in them for your doors and windows), but at a macro level, Revit Architecture does encourage a process that rehearses building construction.

This sequence dependency is not an arbitrary imposition on the part of the developers of Revit Architecture software. Parametric design is always sequence-driven. The choice to match the parametric sequence with a building sequence, however, is the contribution of Revit Architecture developers. This sequencing of the building model also aids in staging design intent as you create the building elements.

Construction History

Many design tools allow objects to be parameterized by construction history. In a construction history–based modeler (there are many, notably Autodesk® Maya® software), the program retains information about the objects by which a form was generated. For example, if a circle is swept along a spline to form a tube, the tube retains an implicit parametric connection to the circle. If that same circle is later modified, say flattened into an ellipse, the tube updates into an elliptical tube. Essentially, the history of construction is encoded by parametric dependencies. The originating forms become the formal drivers of the system.

Inflexibility

This system is powerful but fundamentally inflexible. You can carefully set up a whole hierarchy of dependencies such that a person's height could drive an entire house design, for example. But if there is one mistake in the parameterization of an early component, the only solution is to delete everything back to the point of that component's construction and rebuild it properly. It is a great way to waste time and become frustrated.

Full Parametrics

Revit Architecture exposes a fuller version of parametric design in which the relationships are not fixed at the time of objects' creation. You can modify them or tie them to other parameters at any time. If you say that the length of a wall is to match the length of another component, it would be convenient, but not necessary, for that other component to exist first. You can create the wall first and establish its parametric dependency later. The ability to establish a parametric dependency in a second step after the initial creation of a component requires a system that exposes parameters as first-class objects and a system for establishing invariant relationships between them.

Placeholders

Revit Architecture does, however, enforce certain rules at the creation of components. Some components such as doors and windows are "hosted" in other components like walls. You cannot instantiate a hosted component without specifying a host. It's simply not possible. Nor is it possible to change a hosted component's host after it has been placed. This makes some conceptual sense if doors and windows are to be considered exceptions to a wall's primary function of separation. But it does impose an order on your thinking. You cannot compose a heap of windows by themselves. You cannot say, "No matter what is here in the final design, there must be an opening right here in it." This is not a sequence that Revit Architecture supports. But this problem is solved the way it would be in CAD. A solid placeholder object is placed where the opening ought to go. Later when it is time to carve out the opening, the placeholder's inverse may be used.

Learning Construction Sequence Through Modeling

As mentioned earlier, sequence plays an important role in both construction and parametric modeling. Traditionally, construction sequence is taught as a discipline separate from design, but it may be useful to take a lesson from construction into design. In construction, everything must have a foundation on which to rest. The first-floor walls must sit on something. It can be said that the condition of the foundations determines much of the first-floor walls—where they can and can't go. Were you to design the first-floor walls before the foundation, you would be implicitly defining a support condition that may or may not be possible. Although this does indeed happen all the time, it means that either the cost of the project increases to enact an extreme foundation solution, or the designer goes back to the drawing board to integrate the design condition that should have been considered initially.

Limiting Indeterminacy

This kind of indeterminacy is the fruit of designing out of construction order. Revit Architecture is designed so that users start low and build up. Building pads help define foundations, which help determine walls. Roofs can be defined without reference to existing walls but are easier to define with walls in place. The whole system is designed to make modeling follow construction as much as possible. Buildings designed this way from the start have a less difficult process of rationalization in design development.

Revit Architecture: Design as You Would Build

The Connection

Revit Architecture supports a design process that nearly predicts a building process. This is intentional and quite useful in practice. Adhering to it maximizes the helpfulness of Revit Architecture toolsets and limits the possibility that you are implicitly introducing impossible preconditions as you design out of order. In essence, designing in sequence achieves design rationalization as you go.

This can be seen in contrast to the now-famous Frank Gehry process: the master makes the extravagant paper model and hands it to his team whose job it is to rationalize the design without sacrificing its sculptural qualities. Gehry does not use Revit Architecture, because it is not an appropriate tool for rationalizing extremely free-form design. Instead it is a tool for producing prerationalized design that gets formally complex in its process of development.

If the tools were powerful enough, the prerationalization could simulate constructability requirements. Gravity could be modeled and used to test material strength and stability. It would be just like virtual building. Construction sequence would become part of formal design. This level of integration is in the future, and Revit Architecture has begun the process.

Revit Architecture Features and Concepts to Learn

- Site
- Footprint
  - Building pad and site
- Foundations
- Grids
- Levels
- Build up from the base
- Roof is last
  - Connecting roofs
- Wall layers and rules
- Wall-structure editing
- Modifying vertical structure

Hands-on Topics

- Try building a small building starting from the roof and working down. What problems do you encounter? Are these intrinsic to Revit Architecture software’s manner of dealing with parametric design or to the discipline itself?
- Make a custom, vertically compound wall type and use it in a design.
- Create a vertically compound wall from level 1 to level 2 and a different wall from level 2 to level 3. Align and lock the faces of these walls together. Place a “stack wall” wall type. How do these assemblies differ? How are they the same?

Questions

- The architectural objects in Revit Architecture are designed to be implemented in a process similar to construction. The structural objects are designed to be used from the roof down. Why the difference?

Unit 16

This unit discusses the idea of architecture as engineering and gives you ideas for hands-on practice working with formulas in Revit Architecture.

Theory: Is Architecture Engineering?

Yes. Just kidding. But you see it's a silly question to try to answer definitively. We can structure the discussion by comparing ideologies, methodologies and techniques, and domains of practice.

An irresponsibly reductive history of engineering in architecture: Architecture and engineering have been entwined as long as either has existed. Shelter is itself a technology. In the Renaissance the architect was expected to also be the engineer. And some were fabulous. To construct the duomo in Florence, Filippo Brunelleschi introduced several mechanical innovations for hoisting the lanterns that were used for centuries afterward. The rise of mechanical industry, however, drove an increasing professionalization of what were previously informal trades such as mechanical engineering. A separation grew between architects and structural and civil engineers. Rather than fight this trend, early in the 20th century architecture gleefully abdicated responsibility for engineering, instead engaging technology as metaphor ("machine for living"). Engineering was seen to be a matter of finish or rationalization to be applied after the work of design was done. Mid-century, a few pushes for engineering to enter the design process came from outside architecture from figures such as Buckminster Fuller, but were easily sidelined as crackpot for their overly analytical view of human behavior and utopist idealism. Later, in the 1960s rhetorical architects such as Archigram made outrageous technoarchitectural proposals that were never intended to be built. Lately, however, something has changed.

Are They Closer Than They Used to Be?

Both parametric and BIM design were engineering design methodologies before they made it over to architecture. Engineering began moving over to parametric technologies in the 1980s, and by now it is the dominant paradigm across most engineering disciplines. Why has it taken architecture so long to catch on? And why has it now become fashionable for architects to engage engineers and computer programmers as partners in the design process?

Software for Engineers

Much of the answer has to do with changes in available software and training. Software written for engineers tends to assume a fairly technical audience, and architects have often resisted being too technical lest they lose their artistic credibility. Until quite recently engineering software typically did not pay much attention to graphical user interfaces and often relied on a great deal of purely numerical interaction. Another problem has been architecture's radically underconstrained problem space. The more specific performance bounds that can be put on a problem, the better software is able to help a human user. Architectural problems are often poor in this sort of well-defined constraint and require a more exploratory mode.

Software for Artists

Another realm that engineering software has typically not focused on is rendering. But the new economics of entertainment has engendered a breed of software designed to appeal to graphic artists. These tools are gentle enough for architects to approach. Such software tends to be driven by image production. Engineering tolerances are not important as long as the rendering looks good. This is a seductive realm, and many architects have gone a long way abusing 3D animation software for architecture simply because the images it can produce are so compelling. It is worth noting that in general these programs operate on the simplest geometric model that can be rendered convincingly—a surface model. That means that every object in a scene is defined only by the surfaces that bound it. There is no facility for mass, density, or materiality except as a surface treatment. Architecture modeled as a set of surfaces really can go only as far as rendering. It is not a representation that can generate anything beyond image.

A Hybrid?

Architects share concerns with both artists and engineers. Like artists, their public is largely a lay public whose desire for image cannot be ignored. But at the same time they need to work with a representation that can produce more than image. Fortunately, software for architects that marries the two is arriving.

Economics

The economics that now plays a part in bringing together architecture and engineering until recently served more to separate the disciplines. Buildings are essentially one-offs, custom jobs every time. Small efficiencies in design do not have the same benefit that such efficiencies achieve for mass production. What building project could compete in economic scale with a single military weapons system? The commonalities among building projects are far more subtle and require more sophisticated software systems. These systems are becoming available.

Evaluation

A prime divide between architecture and engineering is evaluation of the products. The evaluation criteria of any engineering project are clearly spelled out in its design brief. That way success or failure can be objectively measured against a metric. There can be no such objective measure in architecture. Its measure of success as a finished product is really only its client and public reception. However, architecture and engineering share objective concerns during their design and manufacture phases. Time and cost of design and manufacture are shared concerns. Modularity and reuse of components is a design ideal that architects now are embracing as well. BIM design serves this kind of modular design particularly well as modules can be parametrically defined and deployed into differing conditions.

Quantitative Methods in a Qualitative Discipline

This entire discussion frames a larger question: To what extent is architecture a qualitative discipline to which quantitative methods do not apply? There is no one answer. Different architectural practices position themselves at different places on this. Some are highly sculptural, while some claim a strong energy-efficiency position. Those practices that make a point of employing quantitative methods incur a burden that not all of them deal responsibly with. There is no numerical method capable of general optimization of architectural form. There are simply too many variables and too many qualitative constraints, such as aesthetics. Therefore any optimization that is performed must be done toward a small set of specific variables relative to a fixed design condition already chosen. There is as much subjectivity in these choices as there is in sculptural design. That does not mean that optimization is useless. Indeed, it is only responsible to try to optimize design decisions for energy concerns. However, these optimizations serve only to ameliorate the damaging practice of building in the first place. So to the extent that numerical methods are useful as generative and evaluative techniques for architecture, they must not be confused with optimization.

Revit Architecture: Formulas

The Connection

Formulas substantially expand the range of expressible parametric relationships. One length may be derived as the square root of another, for example. Along with their power, formulas introduce tremendous complications of data flow and limit back-propagation. If you are not careful, it is easy to wind up with an overdetermined and inflexible design.

Revit Architecture Features and Concepts to Learn

Formulas in the Family Editor
Valid formula syntax
Units
If-then logic
Functions
Propagation

Notes

- You can enter formulas for calculating parameter values. Formulas may include numerical constants and other parameter names. For example, the family could have two parameters named Length and Width. The value for Width might be calculated with the formula Length/2.
- Formulas are case sensitive. If the parameter names have initial capitals, you must enter them in the formula with initial capitals. For example, the parameter name is Width. You type width * 2 in the formula. This is ambiguous to Revit Architecture, and it does not accept the formula.
- Formulas support the following arithmetic operations: addition, subtraction, multiplication, division, exponentiation, logarithms, and square roots.
- Formulas also support the following trigonometric functions: sine, cosine, tangent, arcsine, arccosine, and arctangent.
- For numerical values in formulas, you can enter integers, decimals, or fractional values.
- Legal formula examples:

  Length = Height + Width + sqrt (Height*Width)

  Length = Wall 1 (11,000 mm) + Wall 2 (15,000 mm)

  Area = Length (500 mm) * Width (300 mm)

  Volume = Length (500 mm) * Width (300 mm) * Height (800 mm)

  Width = 100 m * cos(angle)

Hands-on Topics

- Produce a building with a series of components (doors, windows, floors, and so forth) that are each related to the previous one by a formula. Change the initial component’s dimensions, and watch the series update.
- Workbook Unit 16 exercises

Questions

- What are the limits on legal formulas?
- What kinds of design concerns can be expressed in formulas and what can’t?

Unit 17

This unit introduces the concept of databases and gives you ideas for hands-on practice using worksets in Revit Architecture.

Theory: Databases

There are more databases than you think, particularly if you are inclined to be broad in your definition. They are everywhere. There are thousands in your computer. There are thousands in everyone else's computer. There are maybe a hundred in your phone. A thousand in your PDA. There are a few large ones that everybody's concerned about, like the credit rating, real-estate, and IRS databases, but the vast majority are the little local ones that get no attention at all.

Why Is This Relevant?

This is relevant because to work in a piece of BIM design software is to engage directly with a database. And the general principles of client/server architectures apply quite well to the discipline. They will also be of interest in the discussion of multiuser capabilities of design software.

Database Defined

Broadly defined, a database is any system into which data can be tagged with a reference and stored indefinitely to be retrieved later by reference. There isn't much to that, and a great many structures in your computer's memory certainly qualify.

They are used in any circumstance in which information is not totally transient. If you have a phone book application on your phone, it has a database. Increasingly, it has become clear that most of the computer applications in the world are thin front ends put onto database systems. Insurance agents use them to write policies, stores use them to track sales and customers, and credit card companies record everything.

In more mundane terms a database is a method to store, organize, and display information. The building information model is nothing more or less than this. Any BIM software's success must then be determined in part by how well the thin veneer of user interface enables users to interact with that data.

Client-Server Architectures

A well-designed database separates client from server. The server's job is to respond to clients' requests for data by supplying it, and to apply client changes to the data. Although this is a gross simplification, generally we consider a single server per database serving many clients. This structure enables multiple users (clients) of a single database to simultaneously operate on the data without introducing different local versions of it. This factor becomes exceedingly important in the information-organization component of a building project, which is a tremendous burden. Different teams always need simultaneous access to the same data, often with privileges to edit it. The depth of the difficulty in trying to sync up slightly different versions of giant documents cannot be understood without experiencing it firsthand. Couldn't a design document as database help the situation out a great deal?

Basic Principles

The client-server architecture introduces a layer of mediation between the user and the data. It means that the actual format or size of the dataset may not be apparent to the user. You could run a simple phone book off the national Social Security database and not know that it contained anything other than 20 people's phone numbers. Alternatively, you could use it as the database to drive the Social Security system. The two uses are not mutually exclusive, and may be simultaneous. These two applications function the same way as "views" in BIM, as a filter on a large dataset. What queries the database (the building model), displays the data in a way that makes sense to the user (a plan, section, and so forth), and allows the user to modify that data in whatever way is appropriate to the view.

View-Model as Client-Server

It is more than an accident that database clients and views share these characteristics. A view in BIM design is a client of the model database. That is exactly why manipulations in one view show up in all others—because the view translates the manipulation into a database request that changes the underlying data. Other views are clients of the same data and reflect that change.

There is no such thing as a complete "view" of your model. The model is itself the only complete representation of itself, and it is so large, interconnected, and machine-encoded that it is not comprehensible to a human being. So any view of the model is by definition incomplete, and that is what makes it useful to a human user. A view serves to filter the information in a model based on certain criteria and to represent it in a way that makes sense to the user. Modifications that you make to the model are mediated through the current view. The view is responsible for translating the user's operation into a request for a change to the underlying model.

Relational Data

Revit Architecture software's database is relational. That means that information in it can refer to other pieces of information in it. The relational nature of the Revit Architecture data is what enables it to establish parametric relationships.

Multiple Users

The separation of model and view has profound consequences for the simultaneous manipulation of projects. If a single model can serve multiple views belonging to one user, why can't it serve views belonging to multiple users? The answer is it can. With Revit Architecture worksets multiple users can access the database, or building model, at the same time. The software tracks the rights to each object, preventing two users from simultaneously editing the same construct.

At the same time, the ideal of multiple users for the same model is of fundamental importance. One of the primary causes of error in any industrial practice comes about because of different copies of data that get out of sync. The more copies, the harder the synchronization job becomes. Ideally, as is the case of the Revit Architecture central file, there should be only one copy of the data ever, or at least only one that can be modified.

Locking and Checking Out

One way to approach this ideal without tying all users to a single machine is to enable users to "check out" sections of a model. They can then make modifications only to that portion, and when they are done, they must check that section back in. That makes the modification into a single atomic transaction that can be logged or reversed if necessary. Other users may not check out any sections of the model that are already checked out. They must be checked in before anyone else can modify them. This is essentially a component-locking mechanism that locks pieces of the model while a single user controls them and then unlocks them when they are checked back in. The process of checking in a model section is considered a single database transaction. At the time of checking in the changes to the model, Revit Architecture calculates whether these changes are acceptable (cause no problems with the inherent building constraints). If any part of it fails for any reason, the user can accept or reject the change, depending on the severity of the conflict in the building model.

Exporting the BIM Database to Other Formats

Although many systems fall into the broad definition of BIM—a “system into which data can be tagged with a reference and stored indefinitely to be retrieved later by reference”—the nature of BIM in Revit Architecture facilitates the transfer of this information from one database format to another. Revit Architecture allows you to export information contained in the model into various formats depending on the information you want to convey. The lowest common denominator of this database is a raw data dump. The numbers, quantities, and areas associated with the model can be exported to an ODBC-compliant database. Other software packages can then use this database to generate material and quantity takeoffs.

IFC: If the information about the building model is 3D and spatial, you can use IFC (Industry Foundation Class) export to move the geometric information into other software packages. As more companies take advantage of the Revit Architecture API (application programming interface), more programs are being written to take advantage of both raw and geometric data.

Export to gbXML (Green Building eXtensible Markup Language) is an example of exporting to a third-party application that uses this information to provide whole building energy analysis of projects.

Revit Architecture: Worksets

The Connection

Worksets use a fundamental advantage of database-driven design technologies. By drawing strict interdependency boundaries around components, it is possible to carve a project into pieces that can develop independently while preserving a single consistent model. The boon that this represents to teams working on large projects cannot be underestimated

Revit Architecture Features and Concepts to Learn

Process Flow

Create model (how complete the model is before you create worksets is up to you).
Enable worksets (not undoable).
Review default worksets.
Create any more worksets you think you will need.
Assign parts of the BIM to the worksets you just created.
Create a central file on a server that everyone on the project has access to.
“Check in” (make noneditable) all the worksets.
Save and close the central file.
Each team member opens the central file and creates a local copy with Save As.
Users either check out entire worksets or “borrow” smaller sets of objects.
As work proceeds, each team member saves to central location often.

Making worksets
Making worksets editable
Working from a local copy
Borrowing elements

Making requests and accepting others' requests
Checking in changes
Noneditable worksets
Adding elements to the workset
Selecting elements in worksets
Visibility
Seeing latest changes
Selectively opening
Project rollback

Notes

When worksets are first enabled for a project, the following worksets are created automatically:

Views: For each view, a dedicated view workset is created. It automatically contains the view's defining information and any view-specific elements such as text notes or dimensions. View-specific elements cannot be moved to another workset.

Family worksets: One for each family loaded in the project.

Project standards worksets: One for each type of project setting—materials, line styles, and so forth.

Shared levels and grids: User-defined workset that initially contains grids and levels existing at the time the project is shared.

Workset1: User-defined workset that initially contains everything not included in the other worksets.

You can change the names for Workset1 and Shared Levels and Grids at any time. View, family, and project standards worksets cannot be renamed.

Decisions made when sharing a project and setting up user worksets can have long-lasting effects on the project team. In general, when setting up worksets, you should consider the following:

- Project size
- Team size
- Team member roles

Default workset visibility: You can maintain good performance of the project more easily if you plan worksets appropriately and use them correctly. Establishing practical policies on how all team members access and create new worksets in the project helps to maintain performance for existing users and ease the process of introducing new team members to the project.

Project size: The size of your building may affect the way you create worksets for your team. You do not need to make separate worksets for each floor of the building; however, in a multistory structure, you may want to create separate worksets for a set of building elements that appear only on one floor, such as a tenant interior.

If the floor plate of the project is large enough that you need to split it with match lines to fit it on sheets, then you may want to consider creating separate worksets for the building on each side of the match line.

Team size: The initial size of the team affects how you choose to structure the project's worksets. You should certainly have at least one workset for each person, not including the Project Standards, Shared Levels and Grids, or View worksets. A typical project has at least three or more worksets for each person. With element borrowing, users can make elements editable without having to own entire worksets, so designing workset granularity becomes less important.

Revit Architecture enables a single team leader to develop a concept and then share the file with members of the team. In addition, team size usually increases again as projects progress from design development to documentation stages. Views are managed automatically by view worksets. Each view automatically becomes its own workset. View-specific elements automatically go on the view's workset. This capability enables many users to work on detailing and documentation at one time.

Team member roles: Typically, designers work in teams with each assigned a specific functional task. Each team member has control over their portion of the design and uses Revit Architecture to coordinate changes between them. Workset structure for the project should reflect team members' tasks.

Default workset visibility: Performance improves if certain worksets are not visible by default, for example, a complex furniture layout. This technique maintains performance by eliminating additional time required to redraw those elements in all views of the project.

To set visibility correctly, determine whether the elements in the workset will be displayed on many deliverables or only a few. Under this guideline, you might have an exterior workset visible by default, while a furniture-3rd floor would not be. The elements from exterior would be visible in many plans, sections, and elevations. Items from furniture-3rd floor might be visible only on one furniture plan.

To determine if a workset is visible in the current view, open the Visibility dialog box and click the Worksets tab. If the workset is not visible in the view, there is no check mark in the box next to the workset name. Beware that elements can easily be moved from one user workset to another and elements can be accidentally placed on the wrong user workset. If workset visibility is used to prevent display of furniture, it is possible for some furniture to end up in a different workset or some walls to be put on the furniture workset. Controlling visibility by category is less likely to result in such surprises.

Element Borrowing

In Revit Architecture, you can borrow elements from worksets that are not editable. Element borrowing is a process of making elements editable on the fly by obtaining permission from either the central file or users who own those worksets. On attempting to make an element editable, Revit Architecture grants the request if no other user has that element editable. Otherwise it posts a message saying you must seek permission from the user who has the workset editable. Users can grant or deny requests in the Editing Requests dialog box. In Revit Architecture, you must notify the user of requests via email or Instant Messenger. The other user does not receive any automatic notification of your request.

Adding elements to worksets: In Revit Architecture, you can add things to any workset without making the workset editable. You can then modify these newly added elements until you save them to the central file (Save to Central). Once an element has been saved to the central file, you can't change it without making it or its workset editable. Following are additional examples of things you can do without having to make the workset editable:

- You can create new types (floor, roof, wall, component, and so forth) from the Properties dialog box. You can also add view-specific elements, such as dimensions.
- You can edit the newly created types in the Properties dialog box; however, once the changes have been saved to the central file, the properties of the newly created types are no longer editable until the corresponding project standards workset is editable.

- You cannot modify existing types in the project unless the corresponding project standards workset is editable.

Use the Make Elements Editable buttons in error messages to make specific project standards worksets (that is, individual materials or wall types). Use the option to relinquish project standards, view, and family worksets when saving to central to assure that these worksets are available to others on the project.

Groups: Groups have a type workset and an instance workset that do not have to be the same. All elements in a group are in the group instance's workset. To edit the group, you must have the group type workset editable. To modify the elements in a group, the group instance workset must be editable. You can determine which workset the instance or type is in by accessing the element properties. If you use element borrowing to check out a group instance, Revit Architecture automatically borrows all elements in the group. The type can be borrowed from the project browser or from the Make Editable callback requesting the type.

Hands-on Topics

- Create a small project. Enable worksets and save to a central file. Open the central file and save as a local file. For a single user, is there any advantage to working with worksets? Are there disadvantages?
- Using worksets, plan and design a small project as a class or in small groups.
- Export to an ODBC-compliant database, and open with Microsoft® Access™.
- Export a building model to the gbXML format. Log onto www.greenbuildingstudio.com and run an energy analysis of your project.
- Workbook Unit 17.

Unit 18

This unit discusses some of the differences between the school environment and the workplace, and gives you ideas for hands-on practice creating tags and schedules in Revit Architecture.

Theory: Nongraphic Data—Schedules, Tags, and Legends

Students as Radicals

What you are asked to do in school is different from what you are asked to do in an architectural practice. Whereas architecture school is as interested in fostering your individual expression and critical thinking, the early years of practice are largely devoted to your learning a trade. If you aren't avant-garde in school, someone will want to know why, and if you are in practice you may be asked to leave. It's easy to be radical when you don't have to build. Often the freedom of expression open to students leads them to imitation of the styles of the current darling boutique practices. Today these few firms are making wild forms with compound curves. There is a widely held belief that it is important to express new possibilities of form-making simply because they are new (although whether they are truly new is quite arguable). Wild form seems to be its own justification.

Rationalization and Construction in School

Suffice it to say that whatever benefit Revit Architecture offers in the design rationalization and construction phases of a project is lost on students. It's not that they don't understand or appreciate it; it's just that they don't really care. Nobody is asking them to make schedules or construction documents. Occasionally they produce details and callouts, but not usually. Much of the power of a unified building model comes to the fore when dealing with a design team and an environment that deals with real time frames and budgets, and actual building components. What does a student really care whether a window is custom or off-the-shelf? Why not clad the structure in titanium? Who's paying for it?

Design Logic Follows Construction Logic

In discussion with some of the developers of Revit Architecture I asked them about the restrictions on free-form modeling that component-based design implies. Let's take a wedge-shaped wall as a convenient example of something that Revit Architecture doesn't directly support without the use of an in-place family. They asked me, "How are you going to build it?" And they're right. If my "wedge-shaped wall" is really a triangular arrangement of sheetrock, why don't I model it that way? It's more accurate anyway.

Caring for Constructability

This is asking a lot from students who may not know exactly what their formal ideas imply in terms of construction practice. Should they be asked to consider this? By supporting the common state of the art of construction, Revit Architecture implicitly asks them to. The authors of Revit Architecture claim that the default state of the tools in the program supports 95 percent of the current construction in the world. That's probably true. And it's also probably true that student designs lie 65 percent inside the 5 percent that Revit Architecture doesn't support. Students will be the ones who challenge the software formally.

Rhetorical Architecture

Much of student architecture is rhetorical and bears little or no relation to the exigencies of architecture as a practice, but rhetorical architecture is not limited to students. There is a whole class of architects who do not build, but rather produce design as commentary. Take Daniel Libeskind's early work. His tangled messes of lines were not architecture as building description but architecture as argument. Revit Architecture is not likely to be the best tool for rhetorical projects because it is so closely tied to constructability.

Tabular Data Is a View

Although it is unlikely that students will be asked to produce schedules or other tabular displays of the data in their buildings, it is worth their seeing how Revit Architecture supports that kind of organization of data if only to drive home the point that a textual view of a model is a view like any other rendered view.

Revit Architecture: Tags, Schedules, and Legends

The Connection

The work of applying tags to building components, areas, and views, and the construction of textual tables from these is fundamentally one of reference. Reference was how we began talking about the benefit of BIM design in the first place. Revit Architecture software's ability to track all components once and for all alleviates a great deal of the pain of this kind of bookkeeping. As with worksets, students may have a hard time getting interested in tags and schedules, but it is worth their seeing them if only to help illuminate the tremendous effort that leads from design to realization. In the educational environment, color fills may be useful to present graphically the design intent to fulfill programmatic requirements. Color fills rely on having the room tags placed in at least one view.

Legends, on the other hand, are useful tools for displaying the graphical information in a coordinated fashion for things like material boards and presentation design notes.

Revit Architecture Features and Concepts to Learn

- Tags
  - Door tags
  - Room tags
  - Custom tags
- Rooms
  - Room names
  - Areas and volumes
  - Room tags
- Schedules
  - What they are
  - Defining schedules
  - Schedule Properties tabs
  - Applying a phase to a schedule
  - Formulas in schedules
- Color Fill
  - Creating
  - Changing representation
  - Changing colors
- Legends
  - Creating legends
  - Adding symbols, dimensions, and text to legends

Notes

- Schedules represent just another view of your model data, just like a plan or section.
- Legends are inherently 2D. You cannot place objects in legend views directly from the modeling and basic toolbars. You must use the legend tools to do so.

Hands-on Topics

- Create a legend for a current project and use it as the basis for a material selection presentation.
- Workbook Unit 18.

Questions

- Why use tag and schedule? The answer may seem obvious, but in fact software like Revit Architecture calls this into question. What possibility for the replacement of tags and schedules could Revit Architecture offer?

- What happens when you move a room tag to a different room? If you delete a wall between two rooms?
- Why does Revit Architecture prohibit you from drawing a wall with the wall tool directly in a legend?
- How could you use schedules and tags in architecture school? Can you see how schedules could be used beyond construction documentation? Why not use schedules as feedback to the design process? What schedules can you create that enable you to see information that is not readily visible in a model view?

Unit 19

This unit introduces the concept of time and change in design, and gives you ideas for hands-on practice working with walkthrough and phasing in Revit Architecture.

Theory: Time

Time and Change

Parametric design opens up the possibility for models to vary by not only physical parameters but also abstract quantities like time. It is possible using parametric design tools to produce a single model that changes over time.

Animation

If you consider time to be a continuously increasing count, it is easy to imagine this becoming animation. In fact, 3D animation can simply be framed as a 3D model parameterized by time. Animation has tremendous allure to architects because it can often convey space more completely than a still image. The use of animation tools has become quite standard for the presentation of architectural design. Often these animations are mildly parametric. The sun may move across the sky. The camera may swoop through a space and turn. A crowd of people may even be animated through a space. Interestingly, here all the parametric variation is applied to entities extrinsic to the design. Neither the camera, nor the sun, nor the people are part of the building model. The building model stays absolutely fixed.

Is This Realistic?

Yes and no. We all know that buildings are dynamic entities that change from moment to moment. Doors are opened and closed. Systems turn on and off. Finishes weather. But to the outside observer, a static building is not such an offensive approximation. There are, however, circumstances, such as construction and renovation, in which buildings change radically quite quickly. Notably these are the stages in which designers are involved. Wouldn’t it be fascinating to be able to parameterize a model by “phase” of construction?

Phasing

Consider a renovation. You can model the existing condition as the “existing” phase. Some of the walls must be removed. This is done in the “demolition” phase, after which there is a large open space. Then in the “new construction” phase you can build new walls in different places. All these elements can be present in the same model, although they do not temporally coexist. You have the choice to show components from any or all phases. Even if the components from a phase are not currently showing, they are parametrically bound to the model, so that even while invisible they are updated.

Time as a Property

The implications are tremendous. In traditional CAD, you would have to make three separate models for the three phases. If the base model were to change, you would have to apply that change to three separate files. The information coordination quickly becomes a nightmare. This is again the problem of reference. In a BIM system a single model can maintain multiple mutually exclusive states all parametrically tied to the same base.

To step through the phases of a project is to see a different kind of animation in which nothing extrinsic to the model is parameterized by time. It is the model itself that changes.

Revit Architecture: Phasing

The Connection

Revit Architecture offers tools that enable you to deal with both kinds of parameterized time as detailed earlier. Here you see the power of phasing to show a compelling narrative of architectural work.

Revit Architecture Features and Concepts to Learn

Walkthroughs
- Creating a walkthrough path
- Editing the walkthrough path
- Editing walkthrough frames
- Displaying walkthrough view during edits

Phases
- Phases and phase filters for each view
- View phase properties
- Phases with schedules
- Phase properties for modeling components
- The phases command
- Creating phases
- Phase filters

Notes

- All this could be done with layers. The syntax is quite similar: Every object is assigned to a phase (layer), and then phases (layers) may be shown or hidden, or have their appearance modified. But layers add time-based semantics.
- A few phases should suffice. You are in danger of making a mess with too many phases.
- Phasing enables you to specify and name time frames in which parts of the model take different states. For example, starting with an existing floor plan, you can demolish parts in the demo phase and build new things in the new construction phase. Or perhaps there are significant support structure built that will later be removed.

Hands-on Topics

- Create a building that uses phasing. Use New Existing Demo and Temporary types of phase status found in a standard default template.
- Create a master plan from massing elements. Create a new phasing system of Phase 1, Phase 2, Phase 3, and so forth. Set up sheets with views that illustrate change and growth in the master plan.
- Workbook Unit 19.
- Work through the installed tutorial (Help>Tutorials, then browse Viewing and Rendering>Rendering Views and Creating Walkthroughs).
- Produce a walkthrough of one of your projects.
- Use phasing to show an animated sequence of building process that includes significant scaffolding.

Questions

- What advantages are there to the phasing system over layers?
- Any disadvantages?
- What phases does it make sense to specify for most projects?

Unit 20

This unit introduces the concept of variation and gives you ideas for hands-on practice working with options in Revit Architecture.

Theory: Variation

Unknown Constraints

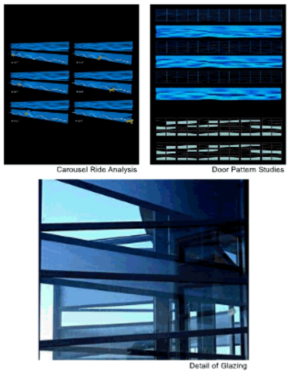
In a recent lecture, Greg Pasquarelli of ShoP told about a project for which they designed a highly customized window system for a client before they knew how much the client was going to be able to budget for it. It is not at all uncommon for architects to have to make a proposal before all the potentially constraining information is available.

Parameterizing Budget

What they did was an exceptional illustration of the power of parametric design. The cost of the design was fairly strictly correlated to the number of glass panels they divided the area into. They were able to produce this relationship as a well-defined mathematical relationship so that they could calculate the cost of any design automatically. Next they developed a parameterized design whereby they could vary the number of panels up and down to hit a certain price point. At one budget the design would look one way, at another budget, slightly different.

A Design Machine

He then went to the client not with a design, but with a design machine capable of producing a design that fit the constraints of the project no matter what they turned out to be. The idea of the architect as a designer of machines for design is powerful and compelling. An architect could produce a machine to produce housing that fits constraints of site and budget anywhere it would be placed. Ultimately, the parametric design machine becomes a way to take advantage of the astounding speed and economy of computation to spread the designers’ intentions more widely than they could without it.



Studies for and detail of parameterized glazing system by ShoP.

Generic Design

These parameterized design machines are a form of generic design. They encode all that is explicitly invariant in a theoretical design space and leave the rest to exploration. The explicit separation of the invariants from the variables is a process that necessarily exposes exactly what the real design intention is. The set of invariants composes the kernel of design decisions that makes one family of proposals acceptable and others irrelevant.

The Product of Generic Design

This idea of producing design machines as designs begins to call into question what is a final design. Why not show a whole family of different forms produced from the same parametric model and call them the same design? At some level they are. Variation like this can be a good way to offer control or its illusion to clients who crave it.

Revit Architecture: Options

The Connection

Options in Revit Architecture is a tremendously powerful feature that enables you to try out multiple designs simultaneously. These can be detailed to any extent as literal options to present to a client, making the design software serve the architectural practice's agility and flexibility to changeable concerns.

Revit Architecture Features and Concepts to Learn

Option set
Primary option
Secondary options
Editing an option
Make Primary command
Accept Primary command
View visibility overrides
Scheduling options
Options
  Creating
  Filtering

Notes

- Options relate directly to a kind of generic design discussed earlier. The skeletons of design can be specified concretely and then different optional features can be applied.
- When using design options, keep in mind the inherent relationships and constraints between the objects. If you put four exterior walls in a design option but do not include the floor slab or roof, modifying the location of the walls will not update the floors or roof.

Hands-on Topics

- Using options, make a single building scaffold, and produce three substantive options that branch off it.
- Create a group from the elements that make up the entry or other portion of the building. Duplicate the group and redefine it by adding and changing elements in the group. How does this differ from using design options?
- Workbook Unit 20

Questions

- How is the options system similar to and different from the phasing system?
- How does the options system compare to the use of layers for the same purpose?