To a large extent, buildings consist of repetitions of identical or sufficiently similar elements. In its previous versions, Autodesk Revit software already included a number of tools that allow for repeating elements, albeit with some geometric or category-specific limitations, such as arrays, divided surfaces. The new Divide & Repeat workflow of Autodesk Revit 2013 offers a more open and versatile functionality that can be used in a host of different contexts. This class discusses various patterns for creating Divide & Repeat rigs consisting of a combination of reference points, divided paths, and divided surfaces. We will take a detailed look at each pattern and its practical application in architectural design by examining examples of iconic buildings and identifying how certain aspects of those buildings could be created with the Divide & Repeat functionality. The class also touches on the methodological implications of creating rigs and components for the discussed patterns.

Learning Objectives
At the end of this class, you will be able to:

• Identify repetition strategies based on design intent
• Devise rigs for Divide & Repeat based on repetition strategy
• Devise repeater components based on repetition strategy
• Explain repetition in the context of case studies of iconic existing buildings

About the Speakers
Andreas has been working with CAD & BIM applications since the mid-nineties. He began using Autodesk® Revit® in 2007 and since then has never wanted to do another project without it. Andreas teaches various classes on Building Information Modeling and parametric design at RWTH Aachen University/Germany where he has been working as teaching/research fellow for the chair of Computer Aided Architectural Design (CAAD) for about a decade now. Since 2011, he has also been teaching BIM and parametrics workshops at conferences and other universities. Andreas is one of the founding partners of the Aachen-based architectural office IP arch, which completed its first major projects in 2010. Currently, he is rather busy trying to write his PhD thesis.

dieckmann@caad.arch.rwth-aachen.de

Zach Kron is an architectural designer and a software analyst for Autodesk. Since 2007, he has worked on Autodesk® Revit® and associated platforms with product designers and developers creating feature functionality, exploring parametric design practices, and generally enjoying technology. In addition to internal teaching and curriculum development at Autodesk, Zach has helped create and run workshops at MIT, ACADIA, Autodesk University and WoodStEx, and maintains the Buildz blog. Before joining Autodesk, Zach worked as a designer in several
Boston-area architectural offices on projects ranging from furniture to bridges. He has more than fifteen years of professional experience in design.

Zachary.Kron@autodesk.com
High Level Concepts & Some Context

Repetition in Architecture
To a large extent, buildings consist of repetitions of identical or sufficiently similar elements. Most often in architectural documentation, each of these elements is “typical”, with many many occurrences of the same item that gets pulled off a truck. For things like standard furniture, doors and windows, the only distinguishing factor is the location. Their placement is often patterned, or determined by a general rule: a column grid starts at one place and repeats every so many feet, window x must occur 20 times on each wall on each floor.

Such identical elements are usually placed manually or using an array tool. In Autodesk® Revit®, to the extent that the entire model tracks and categorizes these elements, this is “smart” repetition, but the elements themselves have little knowledge of where they are or any customized adjustment to their context. For identical, off-the-shelf elements, this is certainly sufficient.

But there is also a role for “smarter” repetition. If repeated elements are aware of where they are, windows that know they are 50 feet above ground or columns that know they are carrying 20 floors above them, they could react to their specific unique context. The windows could become larger or smaller, the columns could get more or less reinforcement. Elements can stay constant in some ways, and variable in others.

Adding custom information to individual elements is also well understood as a process that can be accomplished without fancy tools. Manual data entry is a low risk, predictable method. For instance, if you know that it takes two minutes per column to label and change instance values, and there are 1000 columns in your building, it will take 33.3 hours. If you get started now, you know you can make your Friday deadline. We have all made some kind of calculation like this in the past.

While much of what we are going to show you can do manually, there are more efficient, automated approaches to this kind of customization of large numbers of elements. There are some tradeoffs in flexibility of the systems, and there are of course always risks in diverging from predictable but inefficient methods. However, with the Repeater functionality we can do things more quickly and easily, with the capacity to change the root repetition generator, to get more interesting results less painfully.

Repetition in Autodesk® Revit®
Autodesk® Revit® is first and foremost a BIM application. That means that the majority of tools available are specialized building feature based tools in a highly structured environment that categorizes building elements semantically, according to their function (wall, roof, etc.). Autodesk® Revit® 2010 introduced the conceptual massing environment, which has a more general approach and is designed for 3D conceptual, often freeform modeling.

Both of these environments have tools that allow us to repeat elements. While most of these tools in the project environment are highly specialized (curtain systems, railings, beam systems - with the exception of the array tool), their counterpart in the conceptual massing environment,
the divided surface, has gradually become more open in terms of possible element categories with each release.

**Parametric Patterns**

In Robert Woodbury, Onur Yuce Gun, Brady Peters, and Roham Skeikholeslami’s book „Elements of Parametric Design“, the authors explore what it means to work/design/model/transform in a medium that is about change. Their thesis is that parametric thinking is an old method of designing and that computers simply enable this methodology. In typical CAD packages it is easy to create lots of geometry, but it is very difficult to change geometry. It is this shifting and changing of data structures as a response to intentions and performance requirements that makes for great design tools, and is the backbone of parametric design.

Woodbury discusses 14 patterns as significant conceptual tools in the execution of parametric designs. Each of the patterns is a discrete illustration of a set of associations that result in geometry. Woodbury’s examples are drawn from Bentley’s Generative Components, and they have been reproduced in McNeel & Associates’ Grasshopper plugin for Rhinoceros by Tsung-Hsien Wang. During the last two years, Zach has been re-presenting Woodbury’s patterns in Autodesk® Revit® (and its smaller cousin, Project Vasari) to demonstrate the power of the platform as a design tool in addition to being a documentation tool.

In the context of this class, the following four patterns have the most relevance:

- **Point Collection**
  Intent: Organize collections of point-like objects to locate repeating elements.
  Use this pattern when you are able to think about the size and location of repeating elements in terms of a set of defining points.

- **Placeholder**
  Intent: Use proxy objects to organize complex inputs when making collections.
  Use this pattern when you are able to describe the multiple inputs to an model through a smaller number (preferably one) of abstract proxy objects.

- **Reactor**
  Intent: Make an object respond to the proximity of another object.
  Use this pattern when you want to make an object respond to the presence of another object.

- **Increment**
  Intent: Drive change through a series of closely related values.
  Use this pattern when you are making collections of related parts.

Learn more about Woodbury’s parametric design patterns here:

- [http://www.elementsofparametricdesign.com](http://www.elementsofparametricdesign.com)
Introduction To Divide & Repeat

Basic Knowledge
Prior to Autodesk® Revit® 2013, the process of placing a large number of adaptive components used to be very time-consuming as they had to be placed one at a time. Although adaptive components could also be control-copied on one or several paths (provided their placement points were situated on said paths), even this significantly quicker workflow involved a lot of manual adjustments to align the placed components properly.

Autodesk® Revit® 2013 has introduced the new Divide & Repeat workflow. It allows for the rapid repetition of individual objects along a path or across a surface. Some simple ground rules:

• The Divide & Repeat functionality is available in in-place masses, conceptual masses, adaptive components and pattern-based elements.

• Paths and surfaces have to be divided prior to repeating elements on them. Dividing paths is another new feature in Autodesk® Revit® 2013.

• Repeatable elements need to contain placements points (i.e. adaptive components or pattern-based elements).

• The placement points must be placed on the nodes of the path(s) or surface(s). When using divided surfaces, the visibility for their nodes will have to be activated manually.

• Optionally, placement points can be placed on reference points as well, as long as at least one placement point is hosted on a node. These placement points will remain attached to the respective reference points in every instance of the repeated component and will not follow the path/surface.

For more basic information on Divide & Repeat please refer to Wikihelp:

• http://wikihelp.autodesk.com/Revit/enu/2013/Help/00001-Revit_He0/0135-Prelimin135/0204-Conceptu204/0319-Associat319

Designing with Divide & Repeat
The Divide & Repeat functionality surpasses the classic array functionality that is limited to workplane-based straight or radial paths in that it allows for any given path in space to be used for the repetition of elements. Moreover, elements can be hosted on several paths and can thus assume a different shape in every repeated instance. Repeating elements on one or several divided surfaces effectively results in a two-dimensional array of objects. Note that combining a divided path and a divided surface for repeating elements generally does not make sense, as the repetition on the divided path can only be one-dimensional.

Hosting a repeater on a path or surface in combination with one or several reference points yields an array of repeated elements, parts of which will remain fixed at the same spot (the reference point) while others will follow the path or surface. This technique is well suited for the
use of **reporting parameters** in order to have the spatial relation between reference point and nodes on a path or surface drive the geometry of the repeated component. When working with reporting parameters, remember that strong reporting parameters (i.e. reporting parameters that can be used in formulas) need to reference host geometry only. In adaptive components only placement points are considered host geometry, whereas in pattern-based elements the reference lines already included in the family template qualify as well, allowing for angular measurement on top of length measurement. If necessary, read up on reporting parameters here:


All in all, the new functionality opens up a world of possibilities and makes for a more effective use of adaptive components. But how to approach the design of a repeater? By examining a few examples of repetition strategies we’ll try to highlight some of the topics and issues to think about when choosing a repetition strategy and devising repeater components or rigs for repetition.

**Learning by Example**

*Tiny disclaimer: This is not an exercise in precision. None of the following examples claim to be entirely accurate facsimiles of their real-world counterparts. They are merely as accurate as they need to be in order to illustrate how a particular building or an aspect thereof could be created in Autodesk® Revit® using the Divide & Repeat functionality. The same goes for the case studies examined further below.*

**Example #1: Single Path & Reference Point**

To start with something blatantly obvious, the first step should alway be to **break down the design into repeatable components**. In the case of the very visibly segmented hull of the Montreal Olympic Stadium, this step becomes a no-brainer. Next, **identify the path(s) along which the component is to be repeated** in order to achieve the desired result, in our case a simple ellipse. This illustrates one of the strengths of the Single Path & Reference Point repetition pattern: unlike in a classic array we can now define paths that do not need to be straight or radial and at the same time **repeat components around an excentric reference point** if we so choose. While the hull segments of the real Montreal Olympic Stadium do not point toward the center of the ellipse but are instead spaced more or less evenly on two elliptical paths, choosing the Single Path & Reference Point repetition pattern gave us the opportunity to make this small point about excentricity.
The repeater component itself is designed to create a **curved component surface along the path** as long as the path itself is curved. In order for this to work the component needs to have three placement points on the nodes of the path. Learn more about curved repeater components here:

- [http://buildz.blogspot.com/2012/05/repeat-and-divide-prt-i-curved-panels.html](http://buildz.blogspot.com/2012/05/repeat-and-divide-prt-i-curved-panels.html)

When creating repeater components, bear in mind that **placement point orientation matters**. Placement points will behave differently on divided paths and divided surfaces and this will subsequently affect elements hosted on the placement points. By default they inherit their orientation from their host. In our case the component's placement points are hosted on the nodes of the divided path and would thus orient themselves tangentially to the path. Changing the placement point orientation to „Vertical on Placement“ fixes that. Read more about placement point orientation here:


### Example #2: Multiple Paths

The Multiple Paths repetition pattern is usually the right choice if you need **more control over the shapes of the repeater component instances**. In its original design, the Air Force Chapel could easily be repeated on a single straight path or, easier yet, with the classic array tool. The approach shown here deliberately makes use of a rig that is hosted on the four edges of an extruded quadrilateral so as to be able to distort the shape of the building simply by changing the shape of the extruded form. Basically, the **mass acts as a controller** for the shape of the repeater. We will see this same technique being used again in case study #1.

The placed instances of the rig in turn form the backbone for the divided paths. In order to keep the creation and placement of the actual facade component simple, the facade pattern was **broken down into a single repeatable component**. This might also help when further refining the component as it would most probably involve fewer repetitive tasks as compared to re-editing a more complex component. Placing the component clockwise and counterclockwise yields the **different component orientations** necessary to mimic the facade of the chapel. In a setup such as this one, where the component shape is determined entirely by the spatial relations between the placement points, placement point orientation usually does not matter.
Prior to Autodesk® Revit® 2013, achieving a complex patterning without placing a large number of adaptive components by hand would usually involve one of two techniques:

- The superimposition of several layers of divided surfaces. Depending on the number of divided surfaces occupying the same space, this can quickly become hard to manage. Also, depending on the element pattern and the component's geometry, there could be border tiling issues (see case study #4 for a neat workflow to resolve those).

- The nesting of divided surfaces. Nested divided surfaces have one major disadvantage compared with non-planar surfaces, though: their edges will not follow the shape of the parent surface but rather the edges of the panel they are nested in.

The Single Surface repetition pattern allows for complex patterning on a single divided surface without the challenges mentioned above. The roof structure of the Centre Pompidou Metz consists of a pattern of intersecting beams. Of course, in reality these beams are continuous and would therefore need to be modeled differently than shown here, but nonetheless it is a good example of how to **break down a complex pattern into more manageable parts** and at the same time avoid border tiling issues. The roof's hexagram pattern can be broken down into six single triangles. Due to the interwoven structure of the beams, the **placement order varies** for the different triangle instances in order for the various beams to run continuously on their respective levels. When looking closely at the pattern, it also becomes obvious that the **hexagons overlap in one direction**, so the repeatable pattern actually does not resemble a hexagram at all. Note that when repeating multiple elements on a surface, placing all your components in a single rectangular field on the surface will usually be enough information for Autodesk® Revit® to be able to distribute that pattern across the whole surface.

You should also take a look at Zach's take on Shigeru Ban's Nine Bridges Clubhouse which has a similar roof structure, albeit without interwoven components. This was done prior to Autodesk® Revit® 2013 simply with a divided surface and a half-step pattern and the post also explains in more detail how to tackle the shape of the divided surface:

If you want to learn more about repeater patterns (and you should!), Tim Waldock has started posting a series of elaborate overviews of repeater patterns on his blog „RevitCat“ that is definitely worth checking out:


**Example #4: Single Surface & Multiple Reference Points**

Repetition patterns that include one or several reference points usually make use of the reactor pattern (see above), although exceptions may exist (see example #1). With the use of reporting parameters, the Single Surface & Multiple Reference Points pattern can be utilized to have components that are hosted on the surface react to their proximity to several reference points. In the case of the One Ocean Pavilion, the facades of each of the cylindrical turrets include an array of triangular openings that center around two spots on the respective facades and change their size depending on how close they are to these spots.

Setting up a formula for the component that can express this relationship involves the following steps:

- Finding the **average distance** to the reference points (using reporting parameters)
- Determining the **shortest average distance** (comparing all average distances)
- Defining a **threshold** value (controlling the maximum distance to a reference point that will still yield an opening)
- Creating a **condition for the existence of an opening** (based on threshold value and shortest average distance)
- Defining a **minimum and maximum opening** (based on an inverse relationship to the proximity of the reference point). This usually works **best with normalized curve parameters** as they will always express the opening value as a percentage of the component's edge length and (if set up correctly) will not produce impossible results that could otherwise break your geometry.
- Calculating the **current opening value** (based on shortest average distance and minimum/maximum opening). Note that there should be a **safeguard** in place here (based on the condition mentioned above) for panels that do not meet the threshold requirement.
• Controlling the **position of the void** (based on the condition mentioned above). Note that **voids do not have a visibility property**. Therefore they must be moved outside of an object in order to not interact with it.

Repeating the components seamlessly on the surface does not work as described in example #3 as they can not be arranged in one rectangular field. In this case, each component has to be placed manually once and then repeated separately.

**Example #5: Multiple Surfaces**

The Multiple Surfaces repetition pattern is great for components the shapes of which are determined by two datums. It basically provides the means to control a component’s shape graphically through the shapes of two or more divided surfaces. Depending on design intent, the rigging for the divided surfaces will be more complex than the component’s rigging itself. In the case of the Shanghai Expo UK Pavilion this was certainly the case.

In a way, this pattern is very similar to the Multiple Paths repetition pattern (see example #2): usually the component shape is entirely controlled by the host surfaces and therefore placement point orientation does not matter much here. The biggest challenge in using this pattern is most likely to **make the orientation and node numbering of the various divided surfaces match** as otherwise you might run into unexpected results where placement points on different surfaces will repeat in different directions. Often enough, it will suffice to rotate one of the surfaces’ system by 90°, 180° or 270° degrees. Also note that planar divided surfaces switch their orientation once they are made non-planar.

With its large number of repeating elements, this example is a real CPU killer. Have a look at another example here that is a bit more CPU-friendly:


**Case Studies & Applications**

While the examples in the previous section mainly serve to illustrate some of the various repetition patterns available with Divide & Repeat, the case studies take a more in-depth look at
the parametric aspects of the new functionality and aim to show applications for one of the following:

- Execution of existing ideas (case studies #1, #2, #3 & #4)
- Rule-based form generation and/or patterning (case studies #1, #2 & #3)
- Analysis & fabrication (case study #5)
- Speculative form exploration (case study #6)

**Case Study #1: Linear Reactor**

The Garibaldi Exhibition Building was chosen as a case study due to the nature of its facade. The gill-like facade elements gradually open up more the closer they are to the center of the elongate building, making it a good example of how to control component behavior by measuring deviation from a given path. In Autodesk® Revit® this effect can be achieved by employing the Multiple Paths & Multiple Reference Points pattern. Similar to example #2, a distorted extruded mass acts as both rig and controller. Its longitudinal edges are used as divided paths while a divided path that connects two vertices at the opposing sides of the mass describes a straight path from one end of the building to the other.

The component itself is designed to be placed on two sets of neighboring nodes with five points for each set: four on the mass edges and the fifth on the aforementioned straight path. The distance between the fourth and the fifth placement point (and the ninth and the tenth respectively) describes the deviation of the mass from a straight path at that particular point in space by means of a reporting parameter. This deviation is later used to control the amount of torsion in the component, so the method used here can be classified as a reactor parametric pattern. And as each repeater references a different node to measure deviation (unlike in example #4 where only static reference points are referenced), let's call it a linear reactor.

Additionally, the component has two more placement points that are placed on static reference points hosted on the midpoints of the straight path and its neighboring mass edge in order to have the maximum deviation available as another reporting parameter in all repeater instances.

The torsion in the component is created with a similar method as described for the openings in example #4: the deviation at each set of points is divided through the maximum deviation of the mass, effectively yielding a percentage value for each set. This value is used as a normalized curve parameter to move points that control the component shape toward the inside and outside of a specified zone within the component's rig. A conditional statement sets up an additional rule for the component's behavior at both ends of the building while the amount of torsion can be controlled through a numerical parameter (yet another percentage value).
Case Study #2: 1D Increment
Herzog & de Meuron’s Signal Box may not seem like the project best suited to showcase the parametric aspects of Divide & Repeat. Its facade structure is relatively regular and one would not have to think too hard about how to build it in Autodesk® Revit®. It can, however, serve as a testbed to demonstrate how the increment parametric pattern (see above) that previously involved the manual numbering of elements can be automated by utilizing the Divide & Repeat functionality.

Variation #1: A Simple Increment Pattern

As nodes are by default distributed evenly on a divided path, it is a simple enough operation to deduce the number of a repeater instance if we have the distance between two adjacent nodes and one of their distances from the starting point of the path available. This way we can express the element number as an integer value. When placing the fifth placement point, be sure not to host it on the first node of the divided path, but instead on the reference point of the reference line that defines the path (using the tab key if necessary). In our first variation of the Signal Box facade, we feed the element number into a simple sine function that drives the offset for each panel. In order to have a decent amount of control over the component’s behavior, we modify the sine function with a few tweaks:

• We want the formula to always yield results between 0 and 1, yet another percentage value, so we change $\sin(x)$ to $(\sin(x) + 1) / 2$.

• We want to feed an integer parameter into the formula, but the sine function expects an angle parameter. If we describe the length of a whole 360° cycle as an integer value, we’re in business:
  \[(\sin(\frac{n}{\text{cycle_length}} \times 360°) + 1) / 2\]

• We want to be able to control where the cycle starts. This is easily done by adding an angular offset value:
  \[(\sin(n / \text{cycle_length} \times 360° + \text{cycle_offset}) + 1) / 2\]

• Finally, we want to define a minimum and maximum offset for the panel:
  \[(\sin(n / \text{cycle_length} \times 360° + \text{cycle_offset}) + 1) / 2 * (\text{max_offset} - \text{min_offset}) + \text{min_offset}\]
Variation #2: Liberation From The Grid

We can use a similar technique as shown in variation #1 to create a seamless array of repeater components the shapes of which do not need to follow the nodes of the paths they’re hosted on. In order for this to work we need to make the family aware of the entirety of the paths. This can be done by recreating the paths within the component family and have the geometry travel on them through the use of normalized curve parameters. Thus our component has four placement points for the start and end points of both paths and another two to be placed on adjacent nodes on the divided path. For the distribution algorithm we decided on a gradient pattern which quickly led to using an arcus tangent formula:

$$\frac{\tan(x \cdot n)}{\tan(n)}$$

This formula has an interesting property in that it will yield 0 for x=0 and 1 for x=1 for all values of n (except n=0). This means we can use n to control the steepness of the curve and if we limit our possible input values for x to a range between 0 and 1 we will always receive values between 0 and 1, too. This in turn enables us to use the result of this formula to express the position of each component hosted on a path in the form of a normalized curve parameter. As each component has two placement points hosted on the nodes of the divided path, we can use reporting parameters to determine their distance to the start point of the path and we can do the same for the overall length of the path by measuring the distance between the placement points on either end of the underlying reference line. As in variation #1, make sure to only host placement points 5 & 6 on the nodes of the path. We can then express x as the distance from one node to the path’s start point divided by the overall distance of the path, which will always gives us that value between 0 and 1 we need for the formula.

The components shown here both rely on being hosted on straight paths. However, with a few modifications (and a few more placement points) this technique can be applied to curved paths as well - as long as the component always contains one straight divided path to take the necessary measurements.
Case Study #3: 2D Increment

Now that we have seen how the increment parametric pattern can be used with a divided path, let’s see if it can be translated to a divided surface as well. A rectangular divided surface is basically like a coordinate system: the position of every cell can be described in x and y coordinates. Unfortunately, this information is not readily available in the components’ properties, so we’ll have to use some reporting parameters to dig it up.

Our quadrilateral panel gets an additional three placement points. When placing the component on the divided surface, the additional points are placed on reference points in bottom left, top left and top right corners of the divided surface. Together with the bottom right placement point of the panel itself, they now form two triangles. The heights of these triangles meet in the bottom right corner of the panel and effectively describe the x and y coordinate we’re looking for. As we do not know any of the angles in the triangles, we need to use a formula that does not need angles to compute the height of a triangle, such as this one:

\[ h_a = \sqrt{2 \times (a^2 \times b^2 + b^2 \times c^2 + c^2 \times a^2) - (a^4 + b^4 + c^4)} / (2 \times a) \]

After placing and repeating the component, we now have all we need to compute each component's current row and column number as well as the total number of rows and columns. With this information we can also number the elements consecutively:

\[ \text{element\_number} = \text{current\_column} + (\text{current\_row} - 1) \times \text{total\_columns} \]

What can we do with this information now? For one thing, we could turn these parameters into shared parameters and schedule them. But our case study, the Spanish Pavilion in Aichi, is about complex patterning, not about scheduling. The pavilion’s facade is made up of hexagonal panels some of which are shaped irregularly. Not considering the panel colors, the panels are organized in a 2 x 3 repetition pattern - let’s call this a meta-panel. The color variations are achieved by switching the order of panel colors for different occurrences of the meta-panel. In Autodesk® Revit®, we can build a meta-panel that has several types to achieve exactly that effect. To connect the position-aware panel and the meta-panel we create a third panel that computes the information from the position-aware panel and controls the behavior of the meta-panel, which we’ll call the rule-maker. This isn’t really necessary and, yes, we could achieve...
everything with a single, very complex panel family, but nesting the panels gives us a clean
element hierarchy that enables us to easily swap out the rule-maker and the meta-panel if we so
choose.

With the information available from the position-aware panel, the rule-maker can now set
up conditions that determine which type of the meta-panel gets to be displayed in which
repeater instance. In our case, we started with a simple sine wave pattern, then changed its
frequency, later inverted all the panel values on the outer edges of the surface and finally
changed the values for every third panel. These are just some examples of which types of rules
could be set up using this technique.

The technique shown here does have one limitation: it „only“ works for planar rectangular
divided surfaces (i.e. 90% of all buildings). The path-based approach shown in case study #2
could, however, be modified to a nested family structure and would then yield the same results
for (some, not all) types of non-planar surfaces.

Case Study #4: Corner Wrap
Corner conditions in any system usually call for custom handling. It is rare that the condition that
holds for the interior condition is the same. Even more troublesome is the situation where a
special unifying solution is needed for abutting edge conditions.

For this case, the repeated element needs to do two things: identify the edge that is shared by
both sides for each panel, and describe the pattern in which the solution for this shared edge is
solved, the edge of the overall form.

The corner condition requires six placement points to define a single panel: four points for each
side of the corner, with two of them “shared” as the common edge. In the miter condition
pictured, points 1, 2, 5, and 6 define one side, 2, 3, 4, and 5 define the other. In the fillet
condition, this same combination of points is necessary to define a tangency.

Once the panel is created, the Repeater needs to define the collection of edges it needs to
cover, the overall edge of the form. For two coincident divided surfaces, there is no shared edge
element or common feature that the model already is aware of. To illustrate this, sew the six
point component between two surfaces that are made from unconnected forms.
It looks like a corner condition, until you pull the two surfaces apart.

This is exactly the same condition that is defined with a single form hosting two divided surfaces. As far as the divided surfaces are concerned, there is no shared edge. But if you place two panels, an edge pattern is established.

Repeated
Case Study #5: Unroll / Unfold

Unrolling or unfolding simple and complex geometry is often needed to create study models, validate design feasibility or even for direct fabrication methods. This case study demonstrates two methods to “get flat”, one using planar quadrilateral panels, the other using using simple developable strips.

Unrolling developable strips is operationally much simpler, as it relies largely on built-in functionality in McNeel and Associates' Rhino, using carefully constructed Revit geometry. Complexity comes in the initial setup. Not all surfaces can be broken up into flattenable strips, but certain classes of geometry are quite predictable in this regard. Planar surfaces, revolves, translation, and scale translation surfaces can all be sliced into clean strips. This study demonstrates how to design and isolate these strips for export.

Using a toroidal section (a subclass of revolves), the surface is subdivided using Divided Surface and nodes are turned on. For this form, two orientations of developable strips are possible, horizontal or vertically oriented.

A 14 point adaptive component surface can be used in this model for either orientation.
After placement of the component in either the horizontal or vertical orientation, and hitting Repeat, for one of two resulting sets of components.

Rhino can then be used to “unroll” these surfaces, but the export must be carefully configured. First configure your DWG/DXF export settings.
Making sure that in the “Solids” tab is set to export ACIS solids, NOT Polymesh. Then isolate and export the 3D view as a 2007 format DWG (as of this writing, I was only able to try this in Rhino 4, and DWG files in 2010 or 2013 format did not succeed in bringing across the necessary geometry).

After opening in Rhino, type UnrollSrf into the command prompt and select a surface. The surface will be re-projected onto the horizontal surface in its unrolled state. If constructed correctly, there should be 0% change in surface area for the resulting form.

The same workflow can also be done for more complex forms, such as Scale Translation, as shown in this blog post:


Other references:

- [http://buildz.blogspot.com/2012/05/paper-models.html](http://buildz.blogspot.com/2012/05/paper-models.html)
Case Study #6: Mapping

By using reporting parameters and repeat, one can create forms and shapes unlike the standard ones available in the Revit pallet. This case study is more abstract, demonstrating how the Repeat functionality can be used to explore a particular set of forms. While the method is inspired by work done with spherical inversion, it is not actually inversion (http://en.wikipedia.org/wiki/Inversive_geometry).

The basis of this workflow is an idea of adding the coordinates of one surface to the local coordinates of a second form. Most simply illustrated, if the distance between point 1 and point 2 is measured, that distance can be re-represented by making a point 3 that is of equal distance from 2 as 1 is from 2.

Making this principle into a two point component, combined with Repeat, for any point on Line 1, the distance between it and Line 2 can be represented as a series of points that describe a Line 3 that is the addition of Line 1 and 2.

If this is done with a component that has four points defining two points on each surface, then a more closely defined Line 3 appears.
2D can be expanded to 3D by increasing the mapping of points to three or four sets of paired points (pictured below).

The placement becomes more complex, as do the resulting forms. However, the principle is the same throughout.
A Very Short Summary
In essence, Divide & Repeat is a toolset that automates already existing functionality in Autodesk® Revit®. While the large-scale deployment of adaptive components previously used to involve a significant amount of manual labor, it is now a matter of a few clicks. This, in turn, makes it a lot easier as well as more convenient and viable to experiment with the functionality and explore its multifarious applications.

The creation of one and two dimensional arrays of objects that are free from the geometric constraints of the classic array tool is probably the most obvious application of Divide & Repeat (examples #1, #2, #3 & #5). But we have also shown that, on top of the geometric aspects, the creation of more complex parametric repeaters that are aware of their spatial context is facilitated immensely, whether they make use of the reactor (example #4 & case study #1), increment (case studies #2 & #3) or mapping (case study #6) pattern. And lastly, Divide & Repeat can also be utilized to achieve (semi-) automated solutions to very specific problems, be they design related (case study #4) or documentation related (case study #5).

Troubleshooting FAQ
We believe that we have covered a lot of the topics you need to know about when creating complex parametric arrays with the Divide & Repeat functionality. Unfortunately, there is never time for everything. Here are a few more items you should be aware of:

• I can't seem to use my repeaters to generate a form like I used to with adaptive components.
  
  • **Repeated elements cannot be used for form generation.** In general, model lines contained in adaptive components can be tab-selected and used to generate new forms - this will not work for model lines in repeated components. You can, however, use the model lines contained in (shared) families nested in repeater components.

• It seems like I can't snap to any reference points in my repeaters.
  
  • **Snapping to visible nodes or reference points in repeated elements is not possible.** You can only snap to mid and end points. This also entails that nodes of nested families can not be used for hosting repeaters - but you can always nest repeaters...

• How do I rehost my repeated element?
  
  • **Repeated elements cannot be rehosted.**

• When I delete a repeater, even my original component is gone.
  
  • **Deleting a repeater will delete all of its elements, including the original component.**

• I get an error message when choosing the segment length algorithm for a spline / ellipse based path.
  
  • When dividing a path, be aware that the segment length layout algorithm does not work for splines and ellipses. Use chord length instead.
• All the instance parameters of my repeated components seem to have the same value.

  • **Instance parameters of repeated components display the same parameter values for all instances and act as type parameters when their values change.** They will, however, appear correctly in schedules.

• My repeater takes awfully long to regenerate.

  • **File size and reload times can suffer when you are working with a very complex repeated elements or generally a large number of repeated elements.** Be reasonable. Or are you fiddling around with example #5?

And most importantly the following two:

• My component won’t stand up / points somewhere I didn’t expect it to / looks all funny etc.

  • Again: **Placement point orientation matters!** Points behave differently on paths (think: tangents) and surfaces (think: normals). Try adjusting the orientation of your placement points to a different mode and see what happens.

• I am working with multiple paths / surfaces and my component will not repeat / repeats into different directions.

  • **Path & surface orientation matters!** When working with a Multiple Paths repetition pattern, make sure that all paths are oriented in the same direction (derived from drawing direction, but changeable in the path’s properties). When working with a Multiple Surfaces repetition pattern, be aware that the surface orientation (and the subsequent repetition direction) is dependent on a number of factors: planar vs. non-planar surface, surface vs. mass, type of form generation method (blend, extrusion etc.), profile vs. side face of mass, etc. First try rotating the surface grid, otherwise try a different form creation method.

**Recommendations**

**Previous AU Sessions Relevant To This Class**

Please note that a lot of background knowledge relevant to the contents of this class has been covered in AU sessions in previous years. We recommend you have a look at the following classes:

• AU2011: You Want to Model a What? Converting Real Projects into Parametric Relationships
  [http://au.autodesk.com/?nd=event_class&session_id=9368](http://au.autodesk.com/?nd=event_class&session_id=9368)
  (Mathematics & Parametric Patterns)

• AU2011: Twice Baked: Creating Your Own Adaptive Components and Panels with Autodesk® Revit®
  [http://au.autodesk.com/?nd=event_class&session_id=9221](http://au.autodesk.com/?nd=event_class&session_id=9221)
  (Point Orientation, Adaptive Components & Pattern-based Elements)
• AU2010: Parametrics Laid Bare: Panels and Adaptive Components in Autodesk® Revit®
  http://au.autodesk.com/?nd=class&session_id=7366
  (Adaptive Components & Pattern-based Elements)

• AU2010: Au Bon Panel: Baking Your Own Adaptive Components and Panels with Autodesk®
  Revit® Architecture
  http://au.autodesk.com/?nd=class&session_id=7260
  (Adaptive Components & Pattern-based Elements)

• AU2010: Fuzzy Math Essentials for Revit® Family Builders
  http://au.autodesk.com/?nd=class&session_id=7538
  (Mathematics)

Suggested Reading
You might also want to browse through some if not all of these books:

• Robert Woodbury; Elements of Parametric Design; 2010

• Jane Burry, Mark Burry; The New Mathematics of Architecture, 2010

• Helmut Ottmann, Andreas Asperl, Michael Hofer, Axel Kilian; Architectural Geometry, 2009

• Farshid Moussavi; The Function of Form; 2009

• Farshid Moussavi, Michael Kubo; The Function of Ornament; 2008