

"Would you tell me, please, which way I ought to go from here?"

"That depends a good deal on where you want to get to," said the Cat.

"I don't much care where—" said Alice.

"Then it doesn't matter which way you go," said the Cat.

"—so long as I get *somewhere*," Alice added as an explanation.

"Oh, you're sure to do that," said the Cat, "if you only walk long enough."

Revit Families en Data Handling

Martin Plasschaert, 18 december 2015

Colophon

Author: Martin Plasschaert
Date: December 18, 2015
City: KOEWACHT
Education: HBO ACE Architectural Designer
Institute: TEC / CAD College BV
Kerkenbos 1018 B
6546 BA NIJMEGEN
024-3565677
info@cadcollege.nl
Mentor: Ir. Ronald Boeklagen
Examiners: Ing. Rinus de Boer
Ing. Wim de Goede

Resume

This final project is twofold, first described the use and capabilities of two families. These families are an elaboration of standard functionality in Revit. Otherwise, the process is described for data beyond the basic capabilities that are already within Revit, linking them to objects within the Revit model.

Preface

First let me start with a motivation for the choice of the front page, an image from Alice in Wonderland by Lewis Carroll.

The text of the cover page translating for this assignment means that Revit, and similar software are almost infinite in their possibilities, and there is often more than one way to reach a solution. Finding and then exploiting these opportunities and converting them into a practical application appear to be two different issues thereof.

Initially I chose an elaboration that at that moment I did not even know if this was possible within Revit. Determine the end point where you want to go, the solutions, the route to the destination, it must be said also that one solution does not have to be better or worse than the other, as long as it works towards the endpoint and actually functional working up the chosen project is not always proven to be a simple way.

The image itself represents a certain graphic feature I do not possess and I'm well aware of that.

Decorating images through rendering is therefore not discussed in this project.

The emphasis is as much on exploiting and, where necessary adding functionality. Sometimes, several methods have been developed which will achieve the same goal, then at that time it was a choice to discover as much as possible of the software. Then again, some parts are not further elaborated, read finished, because this further elaboration only would be a repetition of the operations already made and nothing more would be added to the deepening of the software.

This final project is the result of the courses AutoCAD & VB.NET, Revit Base and Revit Advanced as part of the HBO program ACE Architectural Designer at the TEC / CAD College in Nijmegen.

Martin Plasschaert

Martin Plasschaert

Content:

1. Introduction
 - 1.1 Used software and programming languages
2. Problem
 - 2.1 Problem family wall plate
 - 2.2 Problem family roof element
 - 2.3 Problem Data Tool
3. Purpose
 - 3.1 Purpose family wall plate
 - 3.2 Purpose family roof element
 - 3.3 Purpose Data Tool
4. Approach
5. Elaboration family wall plate
 - 5.1 Variable dimensions
 - 5.2 Variable roof pitch
 - 5.3 Optional nested families
6. Elaboration family roof element
 - 6.1 Base family roof element
 - 6.2 Variable dimensions
 - 6.2.1 Variable lath distance
 - 6.3 Variable slope
 - 6.4 Parameters nested families
 - 6.5 Visibility family components
 - 6.5.1 Visibility control by yes/no parameter
 - 6.5.2 Visibility control by object styles subcategory
7. Elaboration Data Tool
 - 7.1 Integrated Revit menu structure
 - 7.1.1 Tab
 - 7.1.2 Ribbon panels
 - 7.1.3 Pushbuttons
 - 7.1.4 Images
 - 7.1.5 Tooltips
 - 7.1.6 Quick access toolbar
 - 7.2 Data handling
 - 7.2.1 Assigning data
 - 7.2.2 Reading data
 - 7.2.3 Deleting data
 - 7.3 Selection based on data
 - 7.3.1 Highlight based on data
 - 7.3.2 Hide based on data
 - 7.4 Database evaluation
 - 7.4.1 Export to database
 - 7.4.2 Show data from database
 - 7.4.2.1 Show data extern application
 - 7.4.2.2 Show data in browser
 - 7.4.2.3 Show data in Revit UI
 - 7.4.3 Edit data in database
 - 7.4.4 Update model data from database
 - 7.5 Display production progress in virtual model
 - 7.6 Add parameters to family via API
 - 7.7 Provide parameters with extensible storage data
8. Afterword
9. Sources

1 Introduction

In the current working methods are within AutoCAD Architecture projects modelled in 3D. To these AutoCAD objects data objects can be assigned, whereupon the data can be exported to a database. Once exported to the database, the data can be processed to obtain efficient data-handling.

This implies among other things that the numbers in the model of the unique elements traced out is multiplied by the total number of needed elements, including all the underlying data associated with that particular element. Also, from the 3D objects in the model, the supplier is located and grouped orders are created per product type. The exported quantities are analysed by type and are then scheduled in a linked planning.

The results of this far-reaching automation of administrative operations are allow us to spend more time on the model so that the model intrinsically is raised to a qualitatively higher level. Of course, the above-mentioned process can then continue to exist only if the reliability of that process is complete.

Because of the marked increase in BIM projects, it makes sense to realize, in addition to existing capabilities, data handling within Revit.

Under data-handling is defined here as awarding free entries to objects and then export them to a database.

Basically this are facilities within Revit already present. However, the major drawback is that there is absolutely no guarantee the integrity of the data.

For guaranteed performance of such processes requires a secure environment is an absolute necessity. Both for the operation of the process as well as the data itself.

1.1 Used software and programming languages

The used software packages: Autodesk Revit 2015, Visual Studio 2015, SQL Server and SQL Management Studio 2012.

The languages used are VB.NET (Visual Basic) in the Revit UI.
SQL (Structured Query Language) for addressing the database.
ASP.NET (Active Server Page) for dynamic Web pages.

2. Problem

2.1 Problem family wall plate

The absence of a family which makes it possible to vary in profile, and the variable length dimension divided in to choose wall plate brackets.

2.2 Problem family roof element

The absence of a family that is so flexible and yet has the ability to generate the desired level of detail in itself. In such a way that to a 'board-level' fast for a base housing a roof element can be generated.

2.3 Problem data tool

The inability to realize within the basic amenities that are a guarantee of allocated data.

3. Purpose

3.1 Purpose family wall plate

The aim is to develop a family that:

- can have a variable width, height and length.
- can have a variable roof pitch where a plane perpendicular to the slope with a fixed length of 44 mm arises
- a choice of optimally distributed nested families residing on variable length with a mutual centre-to-centre distance. It should also display a count of individual numbers of the nested family in a schedule.

3.2 Purpose family roof element

The aim is to develop a family who:

- may have a variable width wherein the number of rafters with each other centre to centre 610 mm are distributed.
- can have a variable rafter height wherein the position of bottom tile lath 11mm above the top of the rafters remains .
- can have a variable rafter width.
- can have a variable starting upper tile lath position.
- can have a variable mutual tile lath Centre to Centre distance .
- can have a variable tile lath starting position at the location of the gutter.
- can have a variable roof pitch.
- can have a variable base.
- can have a variable board with at the location of the gutter.

3.1 Purpose data tool

The aim is to create a possibility, beyond the standard features within Revit, that enables data in a reliable manner, attributable to Revit objects.

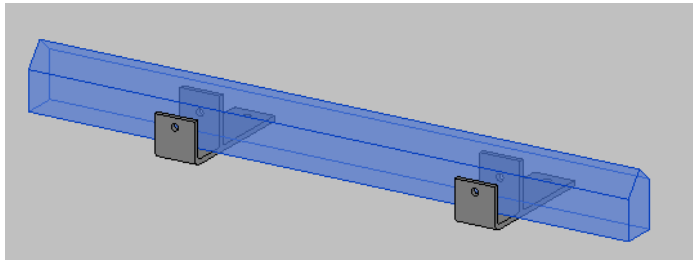
4. Approach

For the families especially first was the definition of the functional requirements needed.

After the conclusion of that description a start was made creating the family and adding the desired functionality.

With regard to the data tool it quickly showed that none or few examples could serve as a guideline. Certainly within vb.net there are no examples. The plan here was to dissect the goal in many small steps. Those small steps soon became goals in themselves. Each step was achieved through trial and error result.

5. Elaboration family wall plate



5.1 Variable dimensions

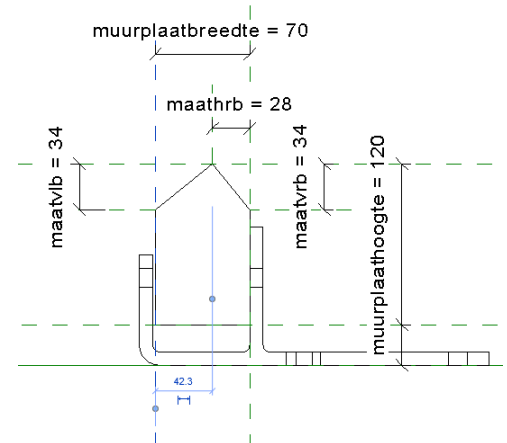
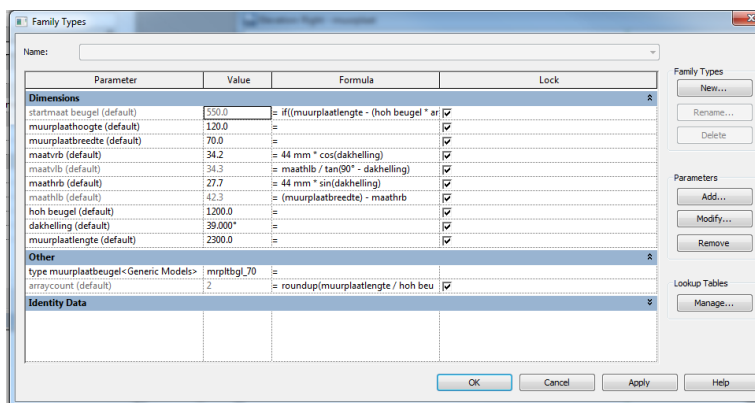
The width, height and length of the wall plate are variable by filling in a parameter which is locked to the extrusion.

5.2 Variable roof pitch

The break points of the wall plate extrusion are mathematically calculated values.

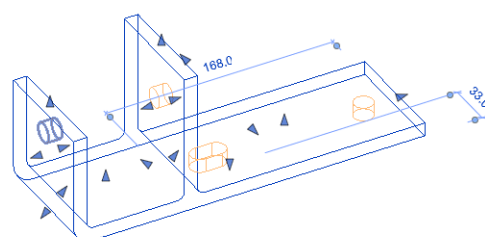
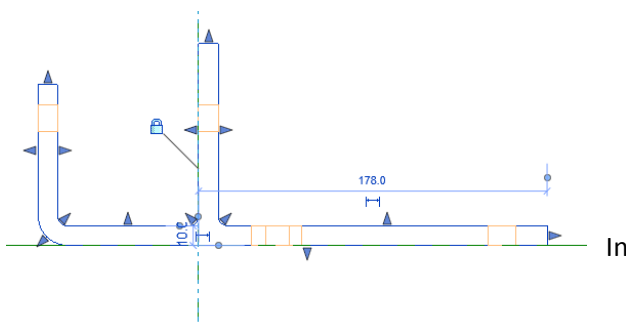
The break points are kinks of an extrusion that are locked in position by reference planes. The position of the surfaces is determined by the calculated parameter value.

Such that a surface formed with a fixed length of 44 mm that is at right angles to the roof surface.

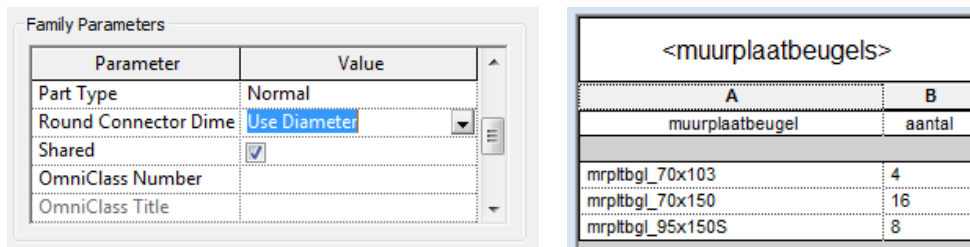


5.3 Pull down list nested families

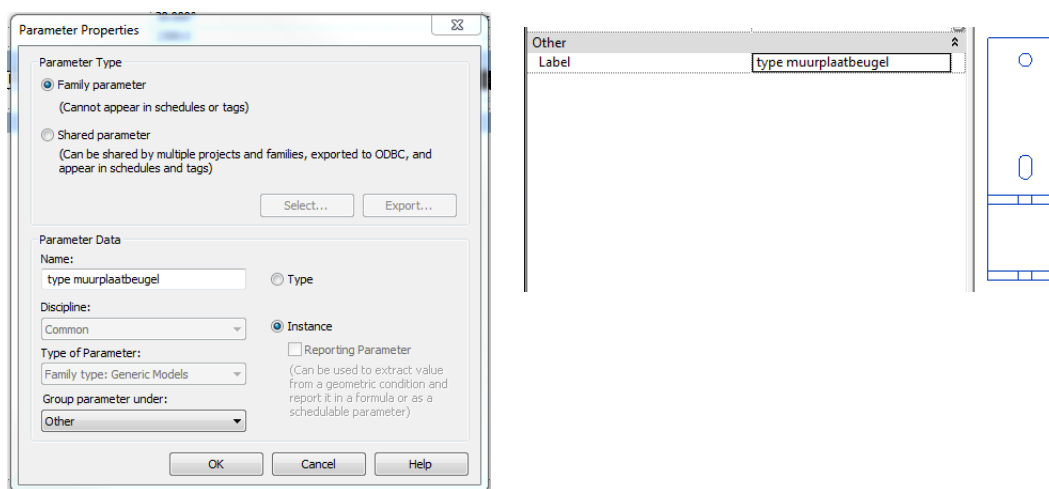
The wall plate itself is composed of an extrusion in which the holes in the bracket are made by voids. Because we want his bracket in a schedule view any further flexibility in this family is excluded.



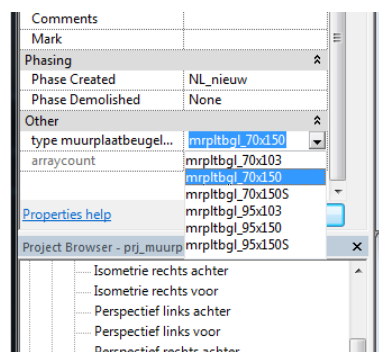
order to include the nested family in a schedule separately this family has to be "Shared". If this is not the case, only the host family will be shown in the schedule.



To the host family to have the nested families appearing in a pull-down menu, the families which will be nested must be loaded into the host family, but not installed. The parameter that fills the list box is a 'family parameter'. The parameter is filled with all the 'Generic models' families (in this case) that are loaded in the host. Finally, the parameter is linked to the placement position by means of the label of a seeded family.



The result: the list.



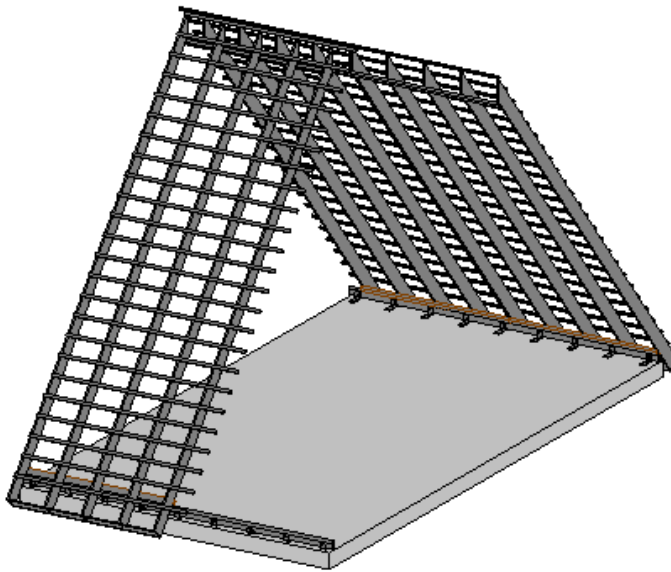
In order to distribute the brackets there are 2 parameters necessary to fill in: the wall plate length and the centre-to-centre spacing of the brackets. And two calculated parameters: the number of brackets (array count) and the start position of the first bracket (starting point bracket).

arraycount (default) | 2 | = roundup(muurplaatlengte / hoh beugel)

The number of braces is in such a way from the centre of the wall plate distributed so that the rest of the size at the ends is always smaller than half the centre-to-centre distance.

startmaat beugel (default) | 550.0 | = if((muurplaatlengte - (hoh beugel * arraycount)) / 2 > 0 mm, (muurplaatlengte - (hoh beugel * arraycount)) / 2, (muurplaatlengte - (hoh beugel * arraycount)) / 2 + (hoh beugel / 2))


6. Elaboration family roof element



6.1 Base family roof element

Because the roof element rests on the wall plate also in the model is chosen to relate the roof element on a face of the wall plate. If the roof inclination of the wall plate is changed, also the slope of the roof element changes.

6.2 Variable dimensions

Properties	
 dakelement	
Generic Models (1)	Edit Type
Constraints	
Host	muurplaat : muurplaat
Elevation	130.3
Dimensions	
onderkantpanlat	275.0
spoorhoogte	220.0
spoordoersteeknokbuiten	7236.5
spoordoersteeknokbinnen	6974.3
spoordoersteekgoot	500.0
spoorbreedte	30.0
positiescharnierregel	6948.3
panlatstartmaatnokcalculated	7262.9
panlatstartmaatnok	0.0
panlathohafstand	310.0
panlateindmaatgoot	200.0
elementbreedte	2440.0
diktevoetplank	19.0
dakhelling	50.000°
basis	4500.0
Volume	0.339 m ³
Identity Data	
Image	
Comments	
Mark	
Phasing	
Phase Created	NL_nieuw
Phase Demolished	None
Other	
panlatzichtbaar	<input checked="" type="checkbox"/>
spoorarraycount	4
panlatarraycount	24
Schedule Level	Zolder

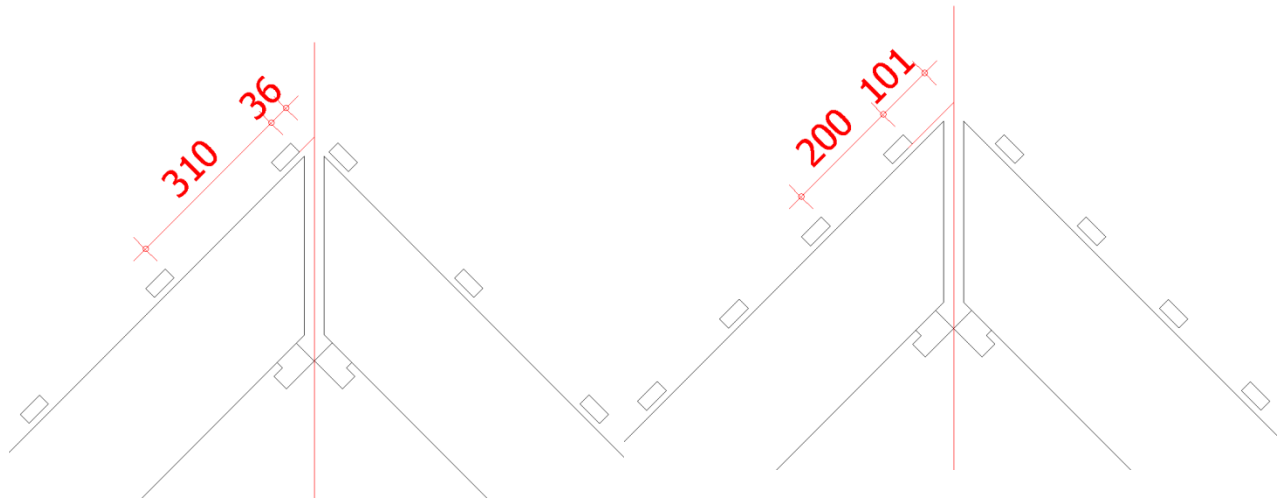
The family comprises a number of parameters which increase the flexibility of the family. A number of parameters is self-explanatory, such as 'rafter width', will adjust the width of the rafters. A number is more fully discussed below.

6.2.1 Variable tile lath distance

The position of the tile lath is coupled to the rafter height. Wherein also the starting point of the upper tile batten is determined by the parameter 'tile lath starting point ridge'. This distance is the point of intersection of the upper side of battens. The distribution of the tile laths themselves is done by an array. The dividing plane is determined by the formula below.

$$\text{panlatarraycount (default) } 23 = \text{rounddown}((\text{panlatstartmaatnokcalculated} + \text{spoordoosteekgoot} - \text{panlateindmaatgoot}) / \text{panlathohafstand})$$

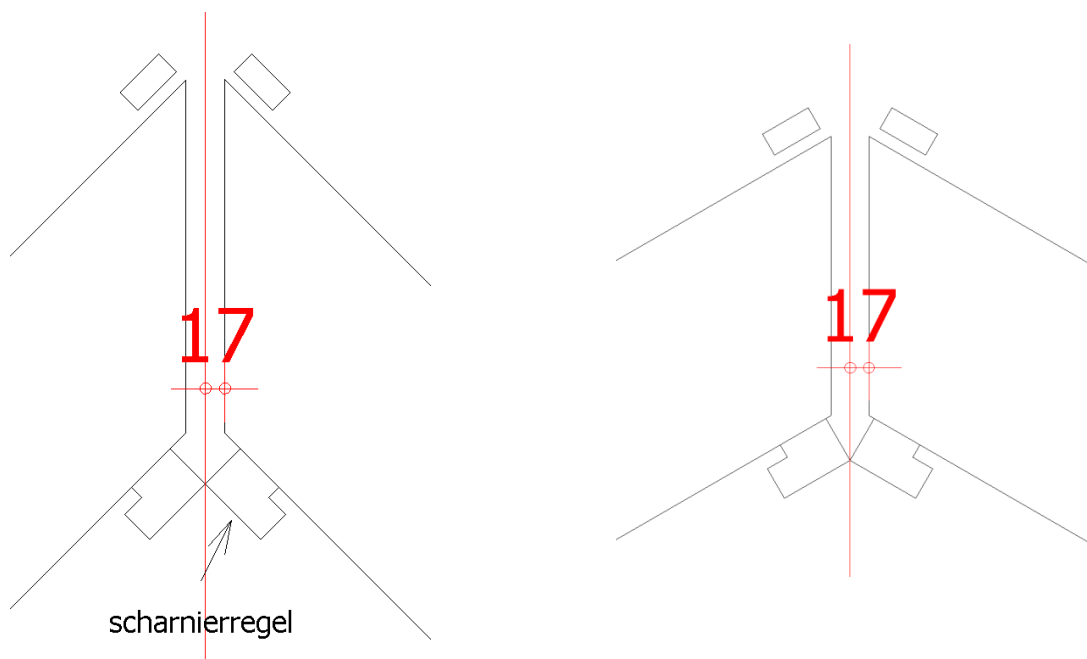
The following two images with different 'tile lath starting point ridge' and 'various tile lath centre-to-centre distance'.



6.3 Variable roof pitch

The variable roof pitch ensures that the point of intersection bottom hinge rules on the ridge line comes to lie.

And that the rafter vertically bevelled such that the vertical bevel 17 mm set back in respect to the ridge line.

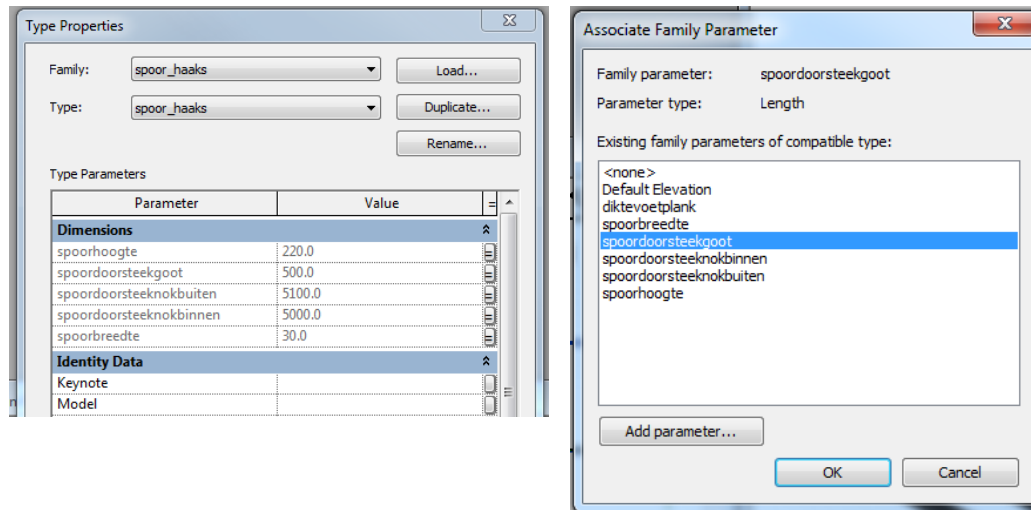


6.4 Parameters nested families

The nesting of families occurs as already described in section 5.3.

However in addition to this, we want the parameters of the nested family to change in the host family. To realize this, we combine the parameter of the nested family on the parameter of the host. Condition is that both parameters are of the same type, thus linking text parameters to text parameters. Create in the nesting family and the host family the desired parameters. Load the nesting family into the host and select an instance.

Now click on 'Edit type'. Click the '=' - button of the parameter and select the parameter to be linked. Do this for all parameters to link.



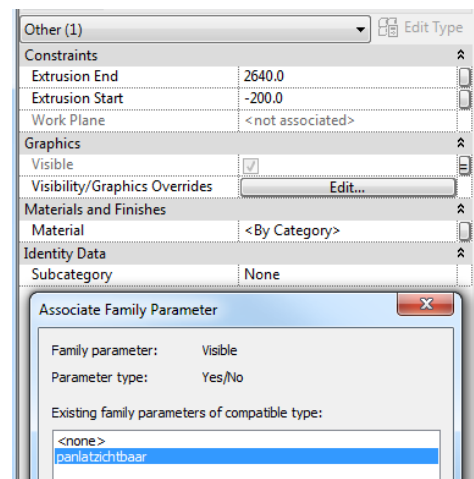
6.5 Visibility family components

There are a number of options to change the visibility / display of family components. We choose two to work out.

6.5.1 Visibility control by yes/no parameter

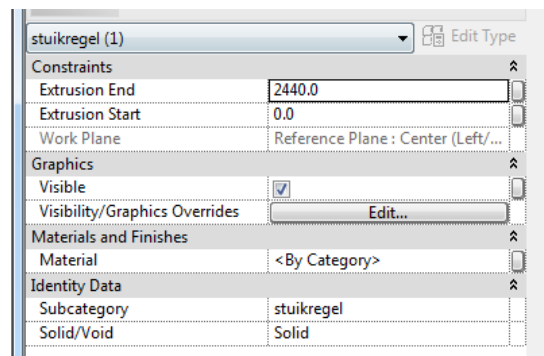
The host family has a yes / no parameter with the name 'visibilitytilelath'.

We link this parameter to select the tile lath by the tile lath in the family and at properties- visible select the 'visibilitytilelath' parameter. This method works globally in the project, i.e. for all views.



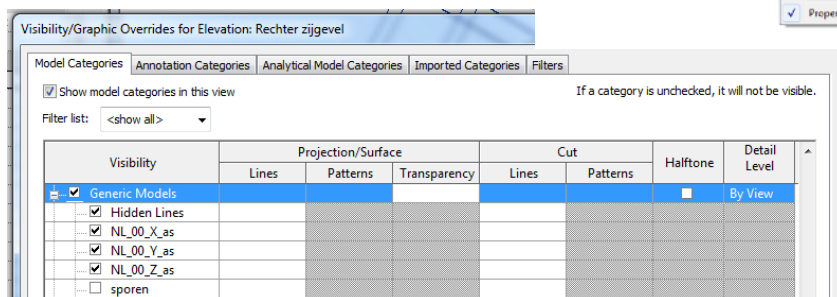
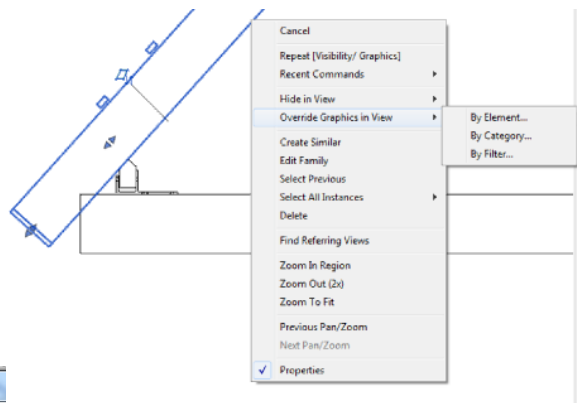
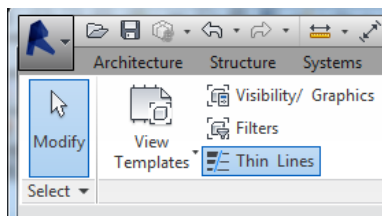
6.5.2 Visibility control by subcategory

At 'Manage' - 'object styles' it is possible to create a new subcategory. Where also the line colour or line type, material can be set. Here the colour is brown set for the 'stuikegrel'.



By selecting the object in the family, the subcategory can be changed.

By 'View' - 'Visibility / Graphics' or by selecting an element within a view, the visibility of the subcategory can be defined. This method is view dependent. Per view, it is possible to set in different settings.



7. Elaboration Data Tool

7.1 Integrated Revit menu structure

General information:

During startup Revit the 'Data Tool' menu is being loaded.

The location of the DLL Dynamic Link Library: the program file with the actual collection of software routines, which is loaded and the images shown are not encrypted fixed. In other words, there is no predefined folder where the DLL should be, this can be any directory.

Because assigning data should always work from a document, the push buttons are activated only when an open document is active.

Language: To prevent a mixture of Dutch and English words chosen is to do the dialogues in Revit in English, this is the native language of Revit. The naming of the data fields is in Dutch, because this increases the clarity and the naming right here is expressly free to fill in.

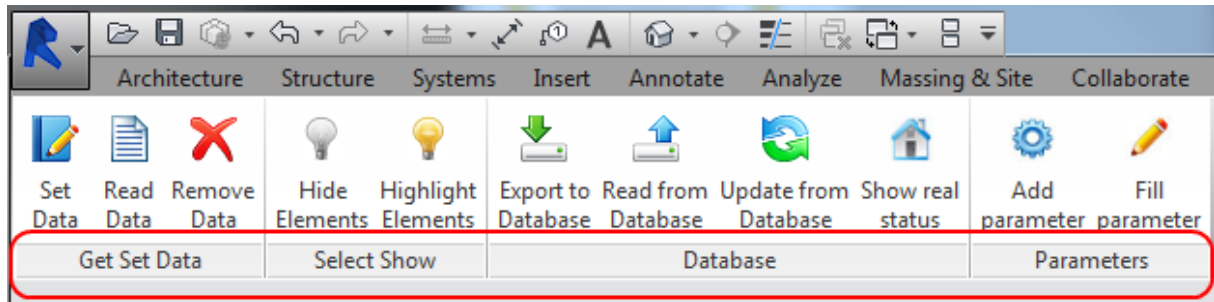
7.1.1 Tab

Times launching Revit is a tab added to the menu.

The tab is divided into 4 Ribbon Panels: Get Set Date, Select Show, Database and Parameters.

7.1.2 Ribbon Panel

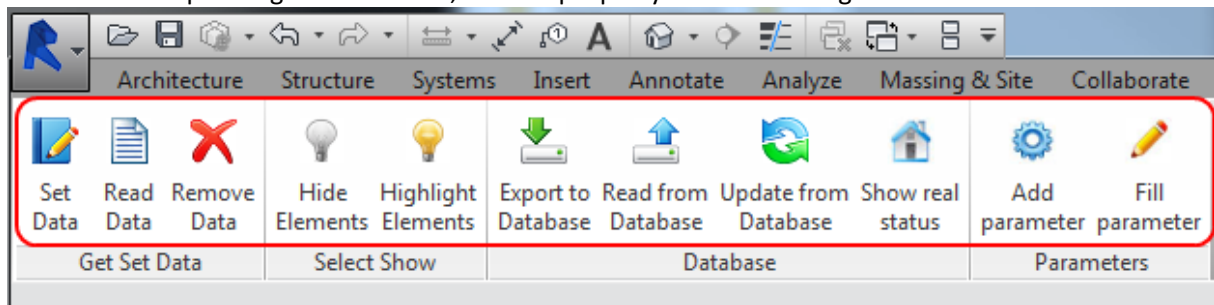
Ribbon Panels grouping the commands that belong together.



7.1.3 Pushbuttons

Push Buttons realize the execution of a command.

The text corresponding to the button, has the property 'multiline' assigned.

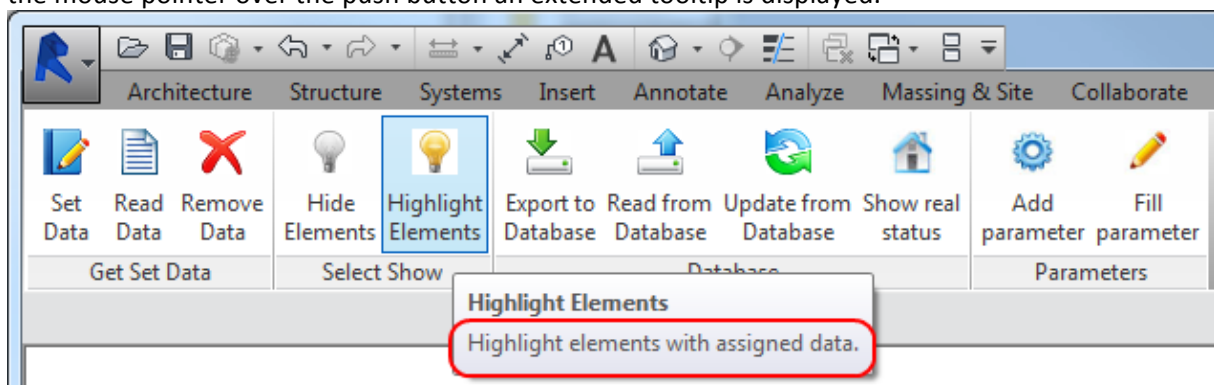


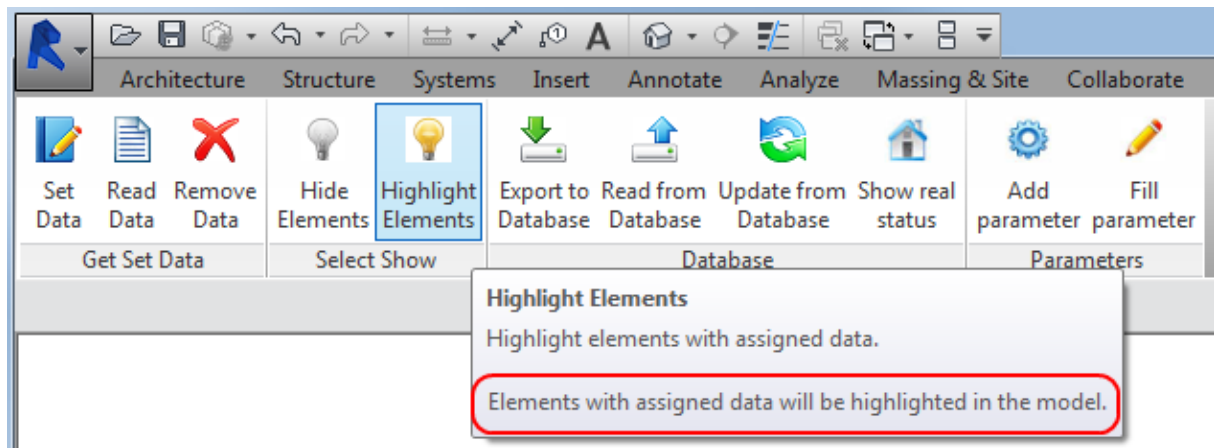
7.1.4 Images

To the push button and the corresponding commands recognizable / appealing to make, they can be provided with an image. If the push button is dragged to the shortcut menu, another image is used.

7.1.5 Tooltips

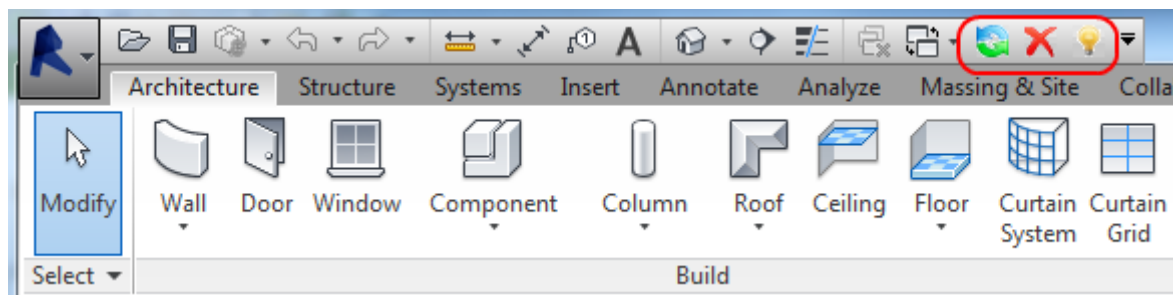
By moving the mouse pointer over the pushbutton the tooltip is activated. By now even linger with the mouse pointer over the push button an extended tooltip is displayed.





7.1.6 Quick access toolbar

The push buttons have the ability to be added to the quick access toolbar. In that case, the push button gets an alternative image. The other properties remain unchanged. The advantage is that when choosing a different tab, the pushbutton directly can be used.



7.2 Data handling

General information:

Assigning data to Revit (roof) objects by extensible storage programmed in vb.net.

There are possibilities available within Revit to assign data to objects. This may be within clearly defined frameworks such as the identity data parameters.

In addition to that there are options that offer more freedom to realize this, for example parameters. However, should any or all objects are already equipped with these parameters.

What we want now is to realize various kinds of data with complete freedom, string, numeric, Boolean, etc. to assign to the objects.

The elaborated method has been to achieve this via extensible storage. This solution was chosen because it is the most direct form of data storage. And also offers the flexibility to achieve the target efficiently.

Because the possibilities are almost infinite, it is especially important to first make a clear demarcation that achieves the desired objective, yet not get bogged down in an extended functionality.

The data to assign can be information about lot numbers, building blocks, dwelling types, routing, component codes, part numbers etc.

The fields of type string that we will assign to the objects are: project id, building number, block, house type and element coding. Type numeric: routing number.

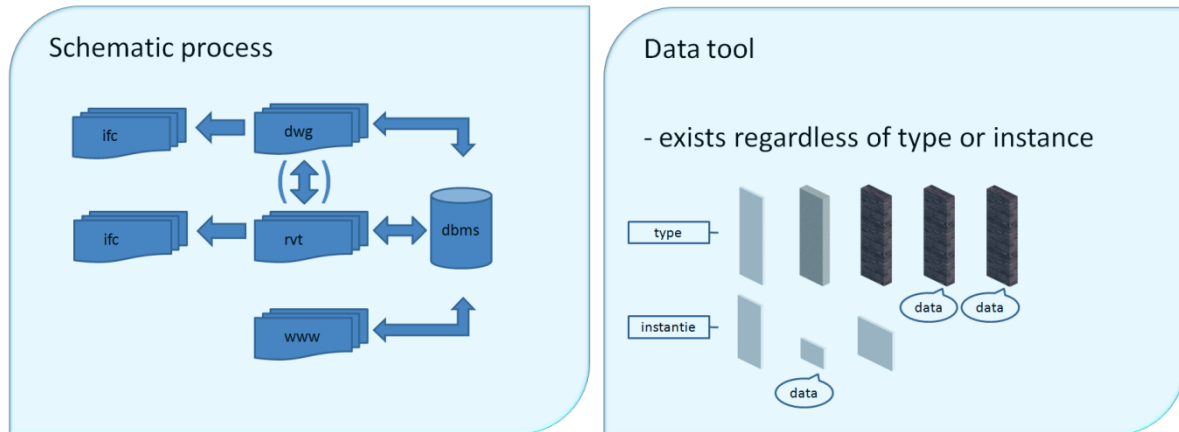
And the type of date: element ready date.

The data can be assigned to objects that are selected by the user interface. Once data is assigned to objects we realize the ability to retrieve and display this data.

Based on a selection filter of the allocated data objects can edit, e.g. pin, hide, unhide, show only etc. all objects with the element number E03.

It then exports this data into a database environment, such as SQL Server.

The processing of the data in the database environment and assigning back on the original objects. This creates a one-to-one situation between the data in the database and model.

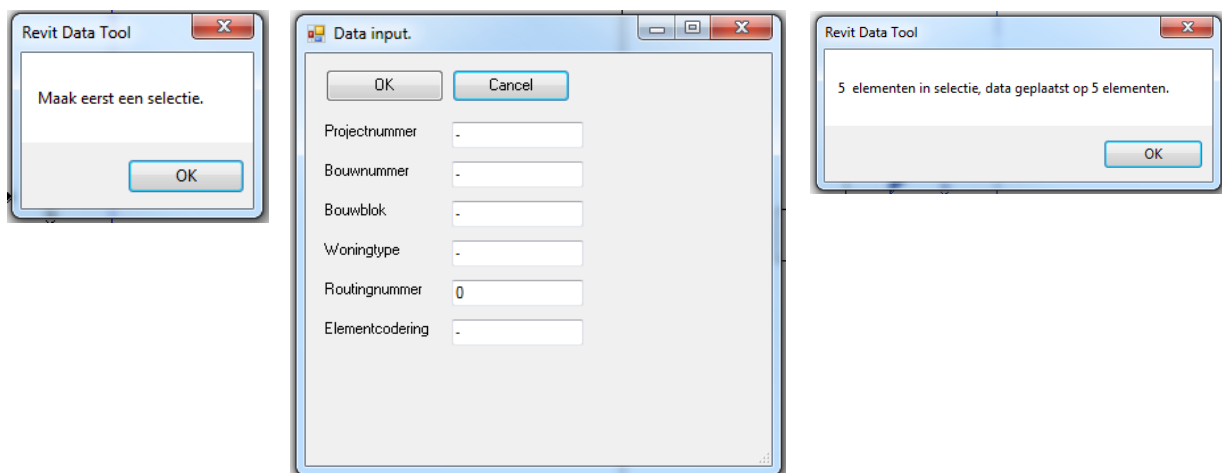


7.2.1 Assigning data



If the number of selected elements is greater than 0, the data input screen opens, when there is no selection a message is shown that first there must be a selection. After completing the data input screen, the entered data is as extensible storage schemas assigned to the selected objects.

Successively the scenes if no elements are selected, the data input screen and the notification to the successful placement of data on the objects.



7.2.2 Reading of data

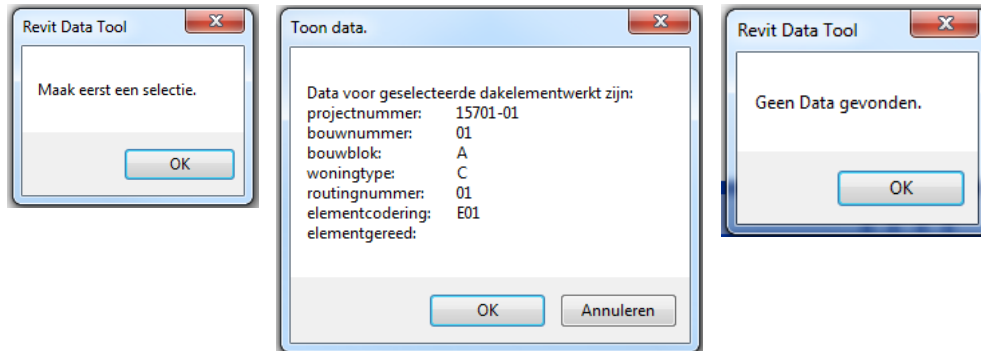
Within the selection is determined or elements contain valid data, it is read out and displayed. During the presentation of the data read zooms in on the element. The Cancel button allows you to interrupt the reading.

Method code:

First, a collection is defined by the current selection in the active document.

If the number is greater than zero is examined for each element within the collection or the element contains a (valid) data schema. If this is the case, then the value of each field is read out and displayed. The element from which the data is displayed is zoomed in to. If interim during showing the dates of the selection should be interrupted can be pressed cancel. If no (valid) scheme in the object is found a message is shown: 'no data on the element is found'.

Successively, the screens if there are no elements are selected, data is read out from an element having a valid schedule, and if the element does not have a valid schedule.



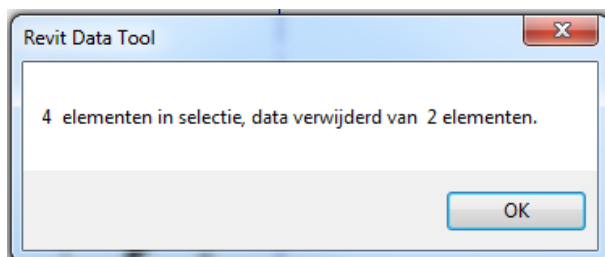
7.2.3 Erasing data



Within the selection is determined whether the elements contain valid data, if yes then it is removed from the element. After the action follows a message with the number of selected elements and the number of elements whose data has been removed.

Method code:

The content of equal to 7.2.2, with the difference that instead of reading out the (valid) schedule, the schedule is deleted. After completion of all elements in the selection screen is displayed with the number of selected elements, and the number of elements whose data schema has been removed.



7.3 Selection based on data

To select objects exist a number of possibilities, each with its own advantages and disadvantages. . For the specific filtering of data within a storage schema the data within the selection is run through. Does the data meet the criteria set in the filter, then the data is added to a collection. Next, an operation can be done with this collection.

Because this is actually a variation of the combination of which is described in chapter 7.3.2 and chapter 7.5 this is not further elaborated.

7.3.1 Hide based on data



Hide elements does the opposite of Highlight elements, namely elements within the selection that do not contain valid data schema are hidden.

7.3.2 Highlight based on data



Highlight elements shows a selection of elements containing a valid data schema. In other words, the elements that are provided with data.

Method code: cycles through all elements within the current selection. If a valid schedule data is found the element containing the data is added to a new element set. The initial selection set is replaced by the new selection set and the screen is being updated.

7.4 Database operations

Database operations can include: read, add, delete and edit data in the database.

First we create a connection to the database, we create a connection object.

If we want to read data from the database, we need something to store this data. We use a dataset object. This data set exists in the computer memory and can be compared to a table. The connection and the data set be created, but now have no interconnection with each other. We use the data adapter object.

The data adapter communicates via the connection to the database, it executes a SQL statement and stores the result of that instruction in the dataset. Below fill the data adapter 'da' dataset 'ds'. The addition, 'Project Overview' serves as an identifier. Example to retrieve data from dataset: Tables ('Project Overview') Rows (0).Item (1) shows the value from the 2nd column in the first row.

```
Dim Con = New SqlConnection(GlobalVar.connectionString)
Dim ds As New dataset
Dim da As New SqlDataAdapter
Dim SQL As String = "SELECT * from TableProject"
Con.Open()
da = New SqlDataAdapter(sql, Connection)
da.Fill(ds, "Projectoverzicht")
Con.Close()
```

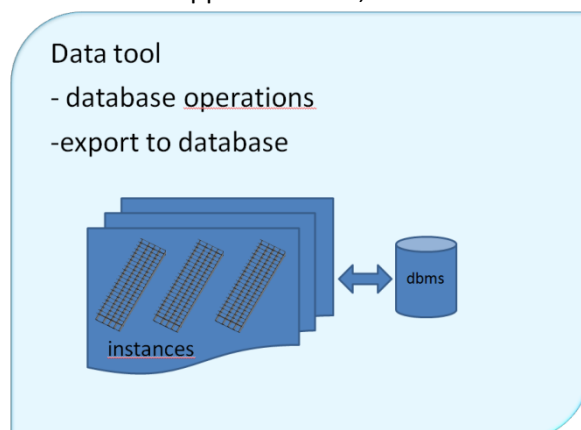
7.4.1 Export to database



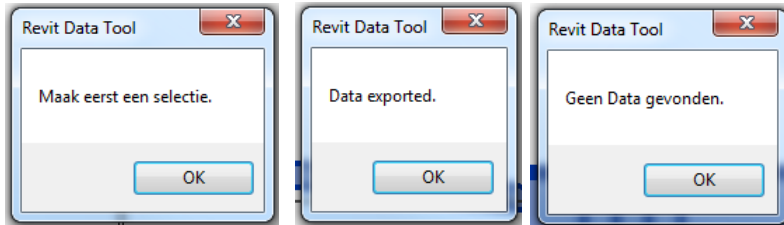
Export to Database export the data of valid schedules within the selection.

Each element is checked whether this element contains a schedule. Then, the data in the schedule are read out and added to the database.

Because we want to add records to the database, we use the Data Adapter object. Through a SQL statement, the records are added to the database. Then the connection is closed by the database. A data set is not applicable here, we do not read anything from the database.



Successively the scenes if no elements are selected, data element and get exported when an element has no valid schedule.



7.4.2 Show data from database

After the data is exported to the database, it is possible to display and edit these through different channels. These channels may include: an external stand-alone application, a web browser or within Revit.

7.4.2.1 Show data extern application

One possibility is to make a connection via an external application using the database and within that display the data. This option is only illustrative mentioned here and is not further elaborated.

7.4.2.2 Show data in browser

Another possibility is to provide the data through a browser. The advantage of this is that one is not tied to a particular location or to a particular network in order to view the data. This kind of web pages that interact with the user, dynamic pages are called, in contrast to static pages showing just their predefined layout.

Actually react dynamic pages with an answer to a particular question. For example: show all elements building number 04, etc.

Postbus 17
4587 ZG
KLOOSTERZANDE
0114-683099

BIM

Home Details Projecten CSV Standhoek BIM BIM-data Hout% Revit data

Onderstaand kan uit de keuzelijst een projectnummer worden geselecteerd.
Na selectie van een projectnummer wordt alle rechtstreeks uit het Revit model geëxporteerde data getoond.

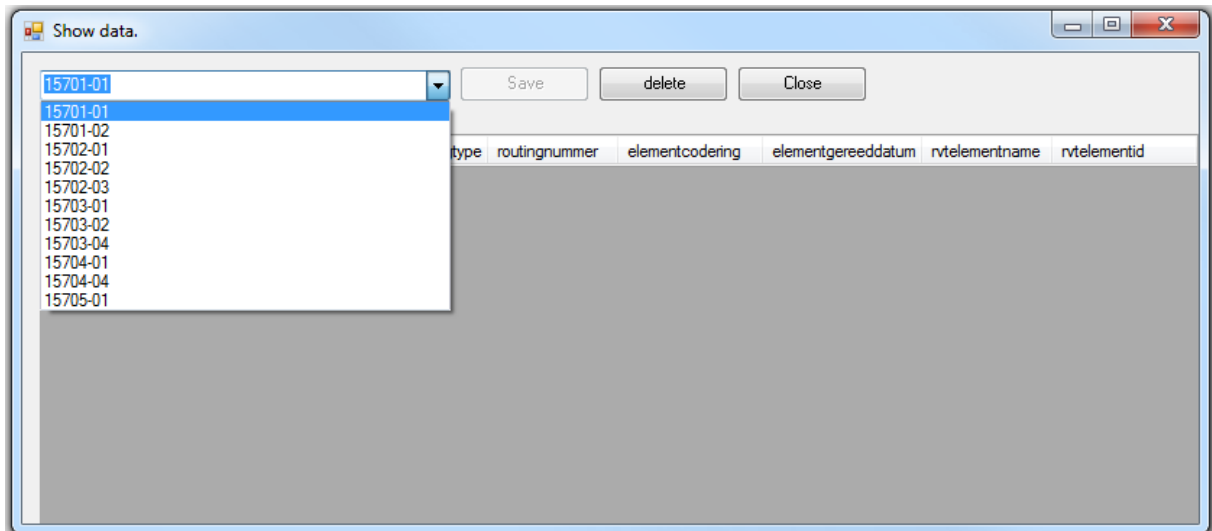
Selecteer een project: 15701-01

projectnr	bouwnr	bouwblok	woningtype	routingnummer	elementcodering	elementgereeddatum	rvtelementname	rvtelementid
15701-01	01	A	C	01	E01		dakelement	435426
15701-01	01	A	C	01	E01		dakelement	435426
15701-01	01	A	C	01	E01		dakelement	435426
15701-01	01	A	C	01	E02		dakelement	435426
15701-01	01	A	C	01	E02		dakelement	435426
15701-01	01	A	C	01	E02		dakelement	923664
15701-01	01	A	C	01	E03		dakelement	965940
15701-01	01	A	C	01	E03		dakelement	965940
15701-01	01	A	C	01	E03		dakelement	435480

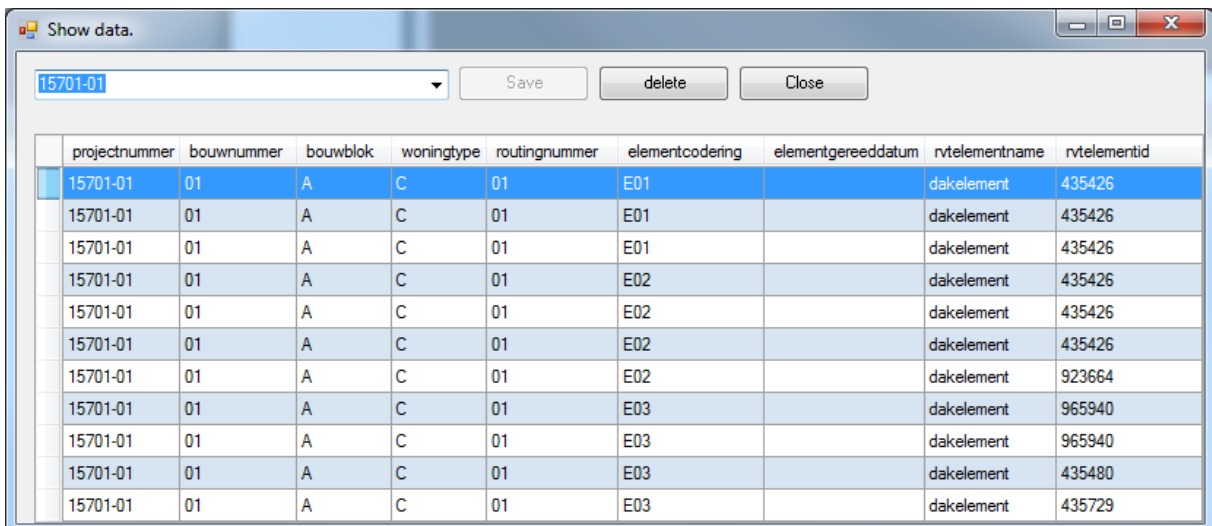
7.4.2.1 Show data in Revit UI



The 'Read from database' button opens a screen. While the screen is being opened, on the background the pull-down is filled with project numbers contained in the database.



After selecting a project, the associated data in this project, exported from the Revit model, is displayed.



7.4.3 Edit data in database

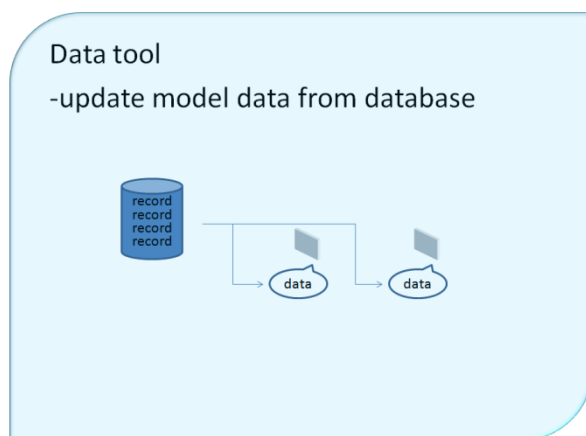
Data is displayed in a data grid view. By default, when you open the form, the Save button is not activated. When changing data, the button is activated. By saving the data in the database is updated and the 'save' button to put not active. This is to indicate that there are no unsaved changes have been made yet. After saving the changed data, the button is inactive again, until there are changes in the data provided, etc.

projectnummer	bouwnummer	bouwblok	woningtype	routingnummer	elementcodering	elementgereeddatum	rvtelementname	rvtelementid
15701-01	01	A	C	01	E01	18-12-2015	dakelement	435426
15701-01	01	A	C	01	E01	18-12-2015	dakelement	435426
15701-01	01	A	C	01	E01		dakelement	435426
15701-01	01	A	C	01	E02		dakelement	435426
15701-01	01	A	C	01	E02		dakelement	435426
15701-01	01	A	C	01	E02		dakelement	923664
15701-01	01	A	C	01	E02		dakelement	965940
15701-01	01	A	C	01	E03		dakelement	965940
15701-01	01	A	C	01	E03		dakelement	435480

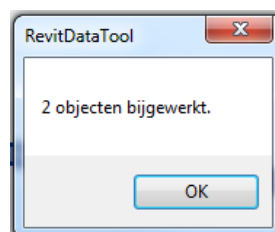
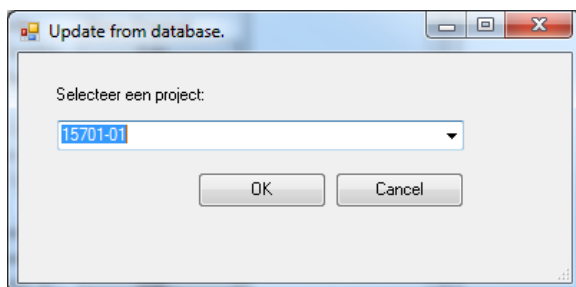
7.4.4 Update model data from database




Changes made to the database can be synchronized from the database with the data in the model. This means that the added dates in the 'element ready date' field are updated in the respective extensible storage schemes in the model.



Method code: After opening the database connection is a SQL statement executed that retrieves all the records from the table associated with the selected project number and have a completed element date. The captured data set is completed with the schedule of every element ID from the model contained in the dataset is updated. After going through the entire dataset is a message with the number of updated features shown.



7.5 Display of production progress in virtual model

 After synchronizing with the external database within the model the appropriate elements have a completed 'element ready date'. Herein, it is possible to make a selection. Now we can the selection in a view.

Show real status Thus we can in almost real-time virtual model of the current state of the to display production. In other words, what is now visible in the model is ready been reported in the production.

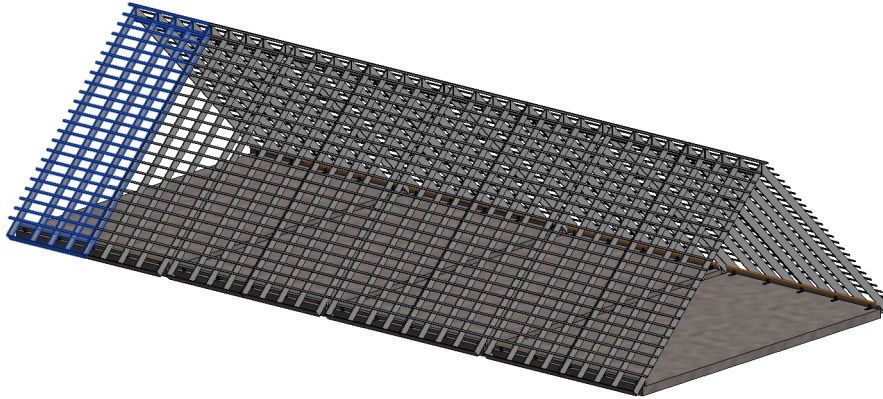


Image of the representation of the complete model.

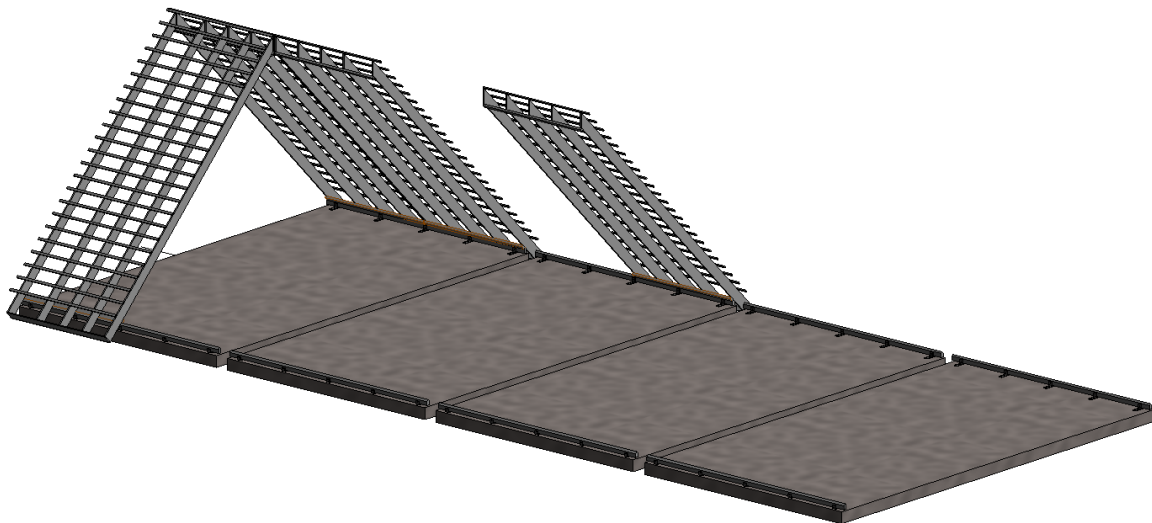



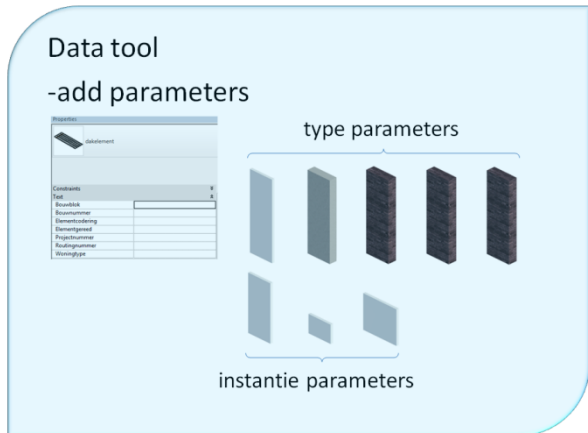
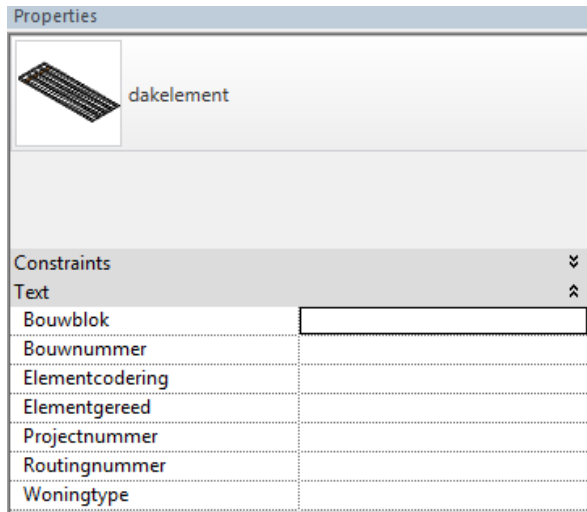
Image of the appearance of the model reported finished.

7.6 Add parameters to family via API

 The data thus far allocated to the elements is only at the relevant element display by means of the function 'Read data'. To now display the data to third parties who do not have the application to display the data and to export the extensible storage data to IFC models, the

Add parameter data is assigned to the corresponding elements. To this end, the required parameters are first assigned to the respective family. If the current selection contains more than one object, a message is displayed that only one object can be selected. This is because the change is done per family. The family is in the background in the family-editor opened. If the family already has a parameter with

the same name the making of the parameter would generate an error message, this is therefore checked first. Then, the parameter with the property 'instance' is added to the family. This is done for all the necessary parameters to create. The family document is now closed and then loaded into the current project.

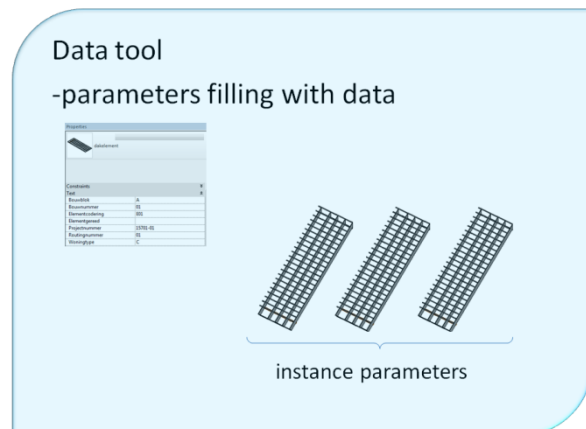
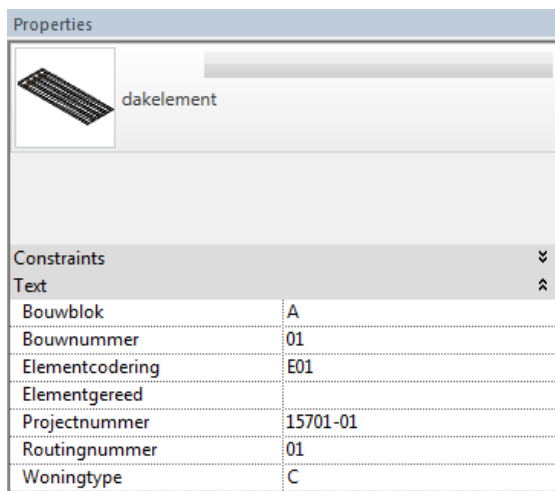


7.7 Parameters filling with extensible storage data



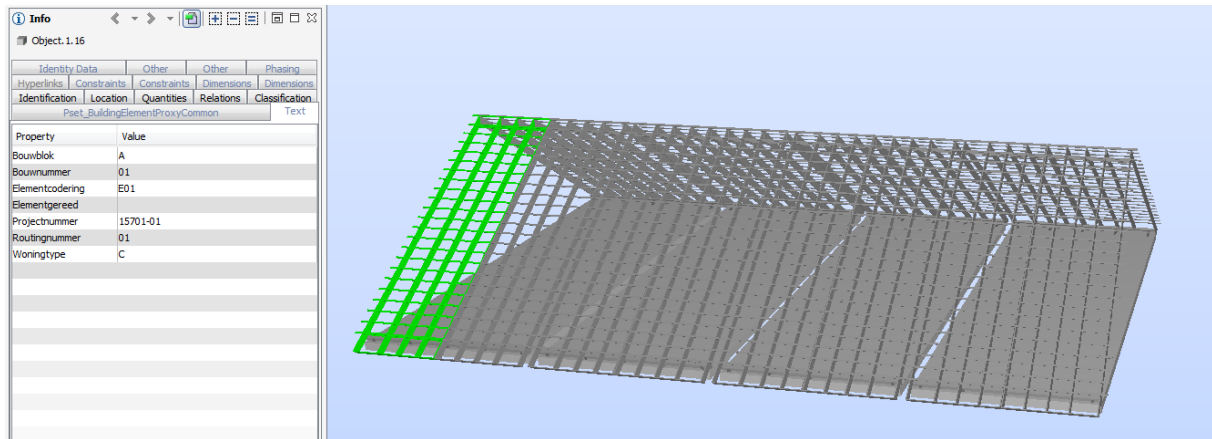
Fill
parameter

After the necessary parameters have been added to the relevant family, the parameters can be provided with the relevant data. In the model the relevant instance of the family for the project has to be selected. The data then is placed in the appropriate parameter.



The now filled with data parameters are exported along with the corresponding element.

Display of the IFC model contains the exported data.



8. Afterword

All facets as described in my first draft for this project are actually developed. Aforementioned is not completely finished, it is only a beginning to exploit the functional capabilities within Revit.

As for many years being an AutoCAD user it is obvious to make a comparison between the two applications. Revit offers more in extending from the BIM-concept in a more accessible way, while AutoCAD seems to be more 'consistent' and more structured with respect to the drawing itself.

To control a production environment directly from Revit, based on data from the model, extensible storage is a necessity. Although Revit in base offers the possibility to perhaps achieve the same, these capabilities provide sufficient collateral for a secured process.

9. Sources

Kris Riemslogh, Software Architect HSBCAD, inspirer cover page.

Ronald Boeklagen, Revit 2015. Nijmegen, Nederland: TEC / CAD College.

Kean Walmsley, Software Architect for a number of AutoCAD-based products, several articles gathered under: through-the-interface.

Jeremy Tammik, member of the Autodesk Developer Network ADN team, several articles gathered under: The Building Coder.