



# AUTODESK UNIVERSITY 2015

ES10192

Navisworks Manage—Empowering Clash Detection

Simon Moreau

Ingerop, Paris

## Learning Objectives

- Learn how to organize your clash-detection matrix around subjects instead of trades
- Learn how to standardize your coordination process using selection set, filters, and plug-ins
- Learn how to automate your clash-detection sorting process
- Learn how to measure your coordination progress

## Description

Clash detection and interference checking have greatly improved how we deal with issues during the design phase. But without a comprehensive method for managing interferences, we quickly end up with thousands of meaningless clashes. To enhance your coordination process we will present some rules to improve clash detection and management during design phases. We will illustrate this class with a case study from Paris where clash detection helped to find and solve coordination problems on a complex laboratory project. This class will cover various methods for creating an efficient clash-detection matrix, import Revit software models in Navisworks Manage software, set up the associate selection sets, and run clash detection. We will see how to group clash results into meaningful reports, and how to automate this process using Navisworks software plug-in. Finally, we will see how to use these clash reports for measuring progress in the coordination during the design phase.

## Your AU Experts

**Simon Moreau** received a master's degree in civil engineering from the École Spéciale des Travaux Public in Paris, France. In 2013 after working on large international projects and developing parametric models on complex framing, Simon joined the building engineering consultancy Ingérop in the coordination department. He works with intelligent models, and he is also responsible for the development of Building Information Modeling (BIM) protocols, standards, and workflows in Ingérop's building section. Simon also writes about BIM on his blog, BIM 42, and he is a Revit software add-in hobbyist with 3 plug-ins on the Autodesk App Exchange.

## Introduction

Do you remember your first clash detection on an actual project? Unless you were working with one of the pretty model Autodesk give us along with Revit, it was probably quite ugly.

For me, at least, it was really messy. I end up with thousands of meaningless clashes, feeling like there is nothing I will ever be able to solve before nightfall or the end of the project. Grouping and sorting them was rather painful, and the resulting reports weren't that helpful to actually coordinate the project.

I was actually so disappointed by these problems that I put aside the entire clash detection subject and focus on actually coordinate my projects.

This is when I develop my first coordination motto:

There should be more emphasis in **Clash Avoidance** then Clash Detection

*Figure 1 : First Coordination Motto*

You have to actually coordinate your project, define the general layout of the building networks and set up modeling rules to avoid conflicts. After, and only after, you can think about clash detection.

Pretty much like anything BIM related, you have to define objectives for this clash detection. These objectives can be multiples:

- Support for the coordination team (obviously)
- Specific problem-solving
- Quality control
- ...

Of course, you have to be aware than you will always find more problem than you can solve. You have to sort, prioritize and manage these problems. I do believe in issue management, and solution like BIM Kubu, BIM Track, and the like are really interesting in this case.

But issue management is not clash management, and we only develop the later here.

## Organize your clash-detection matrix around subjects instead of trades

### Issues versus clashes

The whole purpose of any model-based coordination process is to find issues involving more than one entity (trade or subcontractor). These issues are commonly, but not always, found by searching for geometrical intersections, or “clashes”, between two building elements. An issue, a coordination problem that has to be solved, involves generally more than one clash.

Thanks to 3D modeling, we can find these clashes automatically, with clash detection software like Autodesk Navisworks.

While finding all geometrical intersections, clash detection software also find a lot of false positives related to how we model things in our model. A common example is a pipe crossing a wall where the opening is not modelled.

Multiple clashes per issues and false positives require us to sort and group these clashes to extract meaningful coordination issues from them. This is where I came up with my second coordination motto:

The clash is the **symptom** of the issue, not the issue itself

*Figure 2 : Second Coordination Motto*

A clash is a symptom of an issue in the building conception, not the issue itself, and most of the work when using clash detection is to extract and understand issues from a set of clash points.

The first impulse while running clash detection is to bluntly run clash tests between models. This makes sense since each model generally represent an entity (trade or subcontractors) to be coordinated with the others.

This method extracts all clashes between two models, but we don't have any control over how we detect them. These clash detections generally yield a large number of clashes that have to be manually sorted afterwards.

Another idea is to organize your clash detection around issues to be found, or “subjects”, instead of models.

A subject is a type of issue to find in the conception. For example, a subject can be

- Elements below the minimum required headroom
- Duct crossing structural framing
- Air terminal impacting steel framing

Building our clash detection around subjects allows to define what we are searching, and improve the sorting process.

Furthermore, since each subject is a well understood issue, we can prioritize detections and focus on most important subjects first.

### The clash detection matrix

#### Why a clash matrix?

To efficiently use clash detection, it is necessary to prepare it, and anticipate the sorting and grouping process. So instead of clicking on your "run" button and thinking after, you have to organize your clash tests and define what you are searching.

The usual way to organize these clash tests is a clash matrix, a double-entry table describing every planned clash test.

Here is a pretty basic clash matrix, showing only clash tests between trades.

Here, each green check mark represents a clash test to create and run.

	A	S	H	P	E
A	×	×	×	×	×
S	×	×	✓	✓	✓
H	×		×	✓	✓
P	×			×	✓
E	×				×

Figure 3 : A basic clash matrix

#### Build your clash matrix

To build your clash matrix, you first have to define what subject you want to develop. To illustrate it, I use the following subject:

- Elements below the minimum required headroom

To check if headroom is respected on every rooms, we need to find intersection between rooms and elements that can end up below the required level: beams, duct, pipe ...



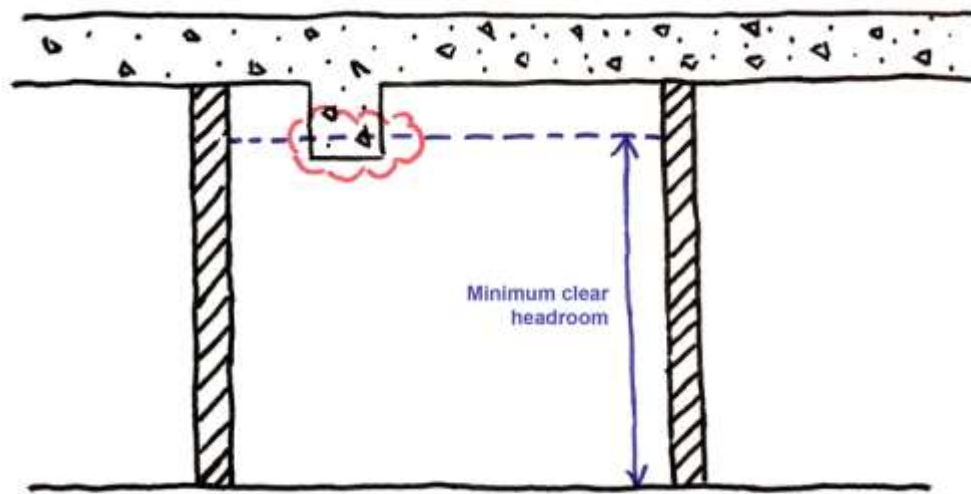
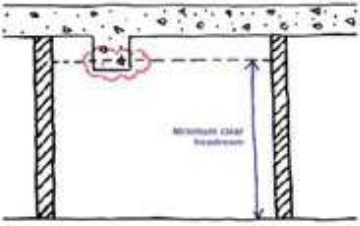


Figure 4 : Minimum required headroom

This subject involves multiple clash detection between various object categories. It can be summarized like that:



Categories	Rooms	Structural framing	Ducts	Sanitary Pipes	Hydronic Pipes	Domestic Water Pipes	Cable Trays	Flex Ducts	Air Terminals
Rooms		4	3	3	2	2	2	1	1
Structural framing									
Ducts									
Sanitary Pipes									
Hydronic Pipes									
Domestic Water Pipes									
Cable Trays									
Flex Ducts									
Air Terminals									

Figure 5 : Minimum headroom clash matrix

Each set of elements is place in table. Numbers in the cells represent priorities for these clash tests. Highest numbers correspondent to the highest priority.

So to find our “Elements below the minimum required headroom”, we will start by running a clash detection between rooms and structural framing.

This principle must be applied for all subjects you want to deal with clash detection.



## Standardize your coordination process using selection set and plug-in

### Setting up your Navisworks

Setting up your Navisworks structure is as important as setting up your Revit files. Model review and clash detection, all enabled by Navisworks models, have to be seamless, and not slow down by heavy conversion procedure.

#### Create your file structure

Navisworks has its very own file structure, and your set up must take this file structure into account.

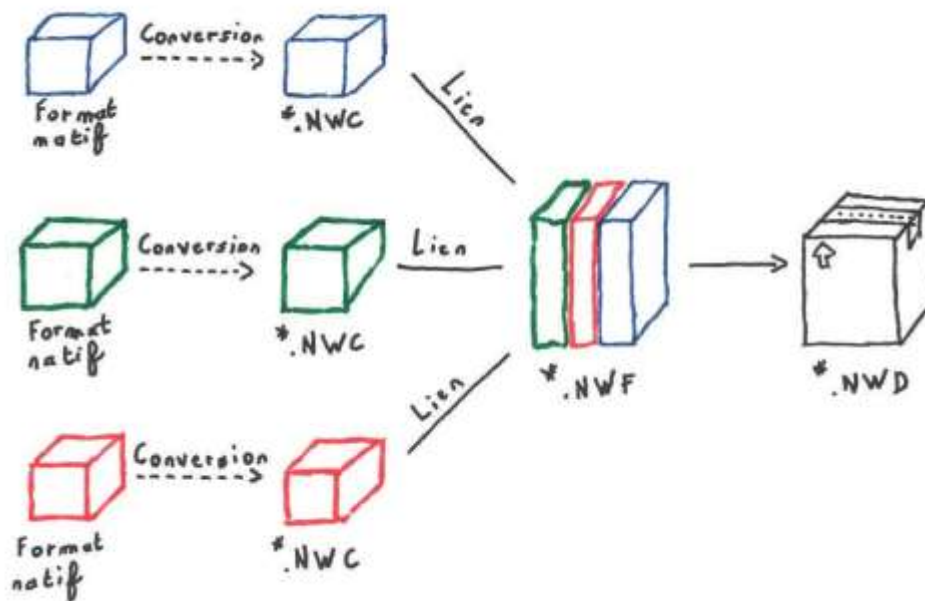


Figure 6 : Navisworks file structure

To open a file in Navisworks, you first have to convert it in NWC. This is this NWC file that will be linked into the NWF Navisworks file. You now have two solutions:

- Append the file directly in your NWF file.  
Navisworks will convert it in the background, place the corresponding cache file (NWC) beside the native file, and append it to your NWF file. Each time you open this NWF file, Navisworks will check if the native file have been updated, and update the corresponding cache file. If the native file as not been updated, Naviswoks will use the cache file directly.
- Convert the native file manually:  
You can use one of the many tool provided with Navisworks to convert your native file in NWC, then append this NWC file to your NWF file. In this case, Navisworks does not check for updates in the native file, and always use the cache file (NWC)

These two methods for creating the NWC file have both their advantages and drawback

Appending directly your CAD file directly in Navisworks is of course very convenient, but having to convert it each time you open your Navisworks quickly become tiresome.

With the second method, on the other hand, provide perfect control over your conversion process, and you always know witch file have been converted and when. I will focus on this method for the rest of my process.

Once a Navisworks cache file (NWC) is created for each of your CAD files, you place them in a file structure that mimic your main CAD file structure, and keep the same naming convention for your CAD files and your cache files.

You can know append these cache files to your Navisworks NWF file and create your coordination model.

### Create your selection sets

You have to create a selection set for every item of your clash matrix.

Since your clash matrix is organized by subject, and not by file, you cannot just select your file in Navisworks and run your clash detection. You have to prepare you clash test according to our selected subjects.

Search set or selection set?

Search set and selection both handle selection of a set of objects in a Navisworks.

The selection set save a list of selected objects:

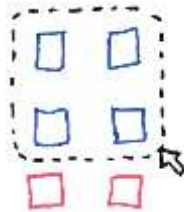


Figure 7 : Selection Set

On the other hand, the search set save a query used to retrieve these element, here: “select all blue objects”.

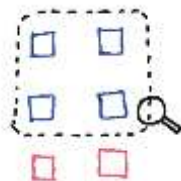


Figure 8 : Search Set

So as we update our model, the selection set keep the same elements, and the search set update with our model.



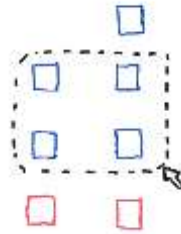


Figure 9 : Selection Set do not update

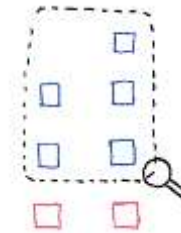


Figure 10 : Search Set update

These search sets select some object based on their properties, which is what we are searching here.

Using these search sets, we retrieve elements based on their properties.

In my example, we will search for beams below the minimum headroom.

Using the “Find Items” windows, we search for structural framing in the structural model. To do so, we search for Elements where the “Category” property match the “Structural Framing” value

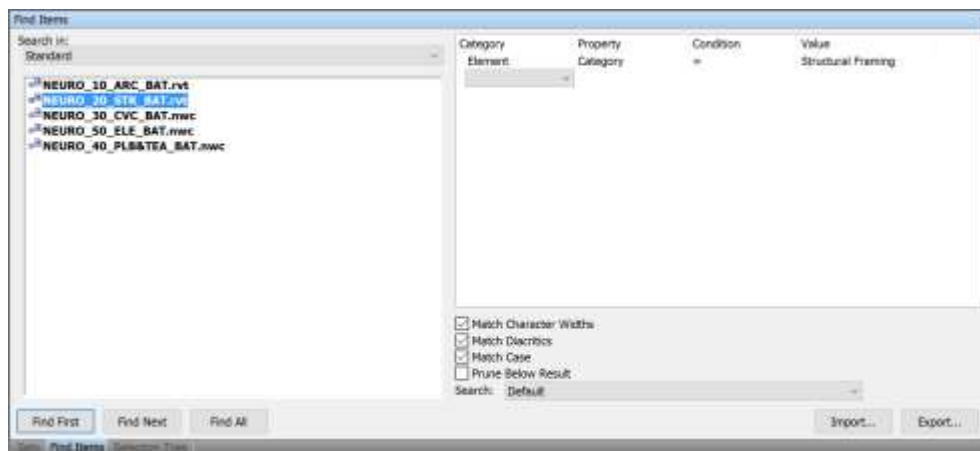


Figure 11 : Structural Framing Search Set

Likewise, we search for Rooms in the architectural model.

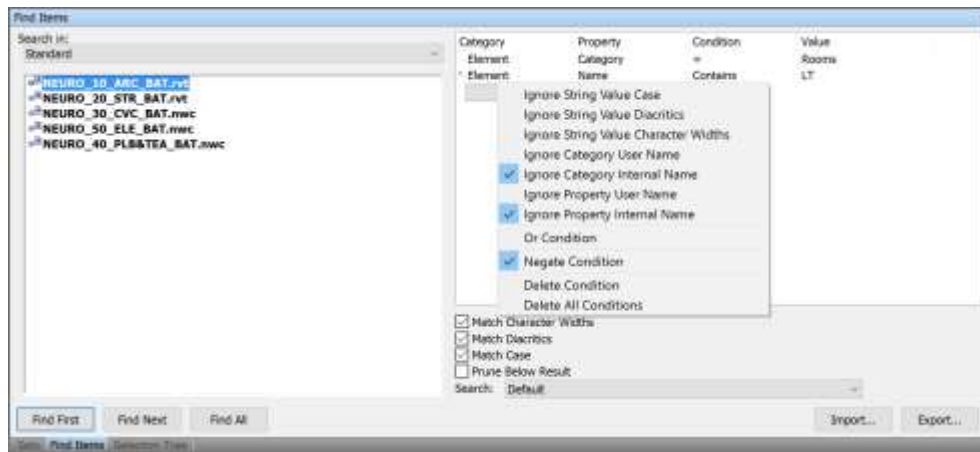


Figure 12 : Rooms Search Set

As you can see, we add a Negate condition to remove from our search the rooms with “LT” in their names, as we don’t want to take technical rooms (“Locaux techniques” in French) in our clash detection. We save these two searches by clicking on “Save Search” in the Sets window.

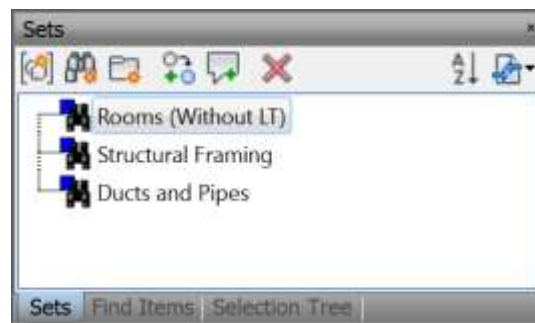


Figure 13 : Save Search Sets

These search set can be saved as an .xml file, to be reused on another Navisworks file. This is the beginning of the normalization of your clash detection process across multiple projects. As you develop subjects with clash detection, you will have more and more of these searches set already set up and ready to be deployed on your next project.

### Create your clash test

Once your selections sets are properly set up, you can create the clash test corresponding to your clash matrix.

Each intersection of the matrix relate to a clash test

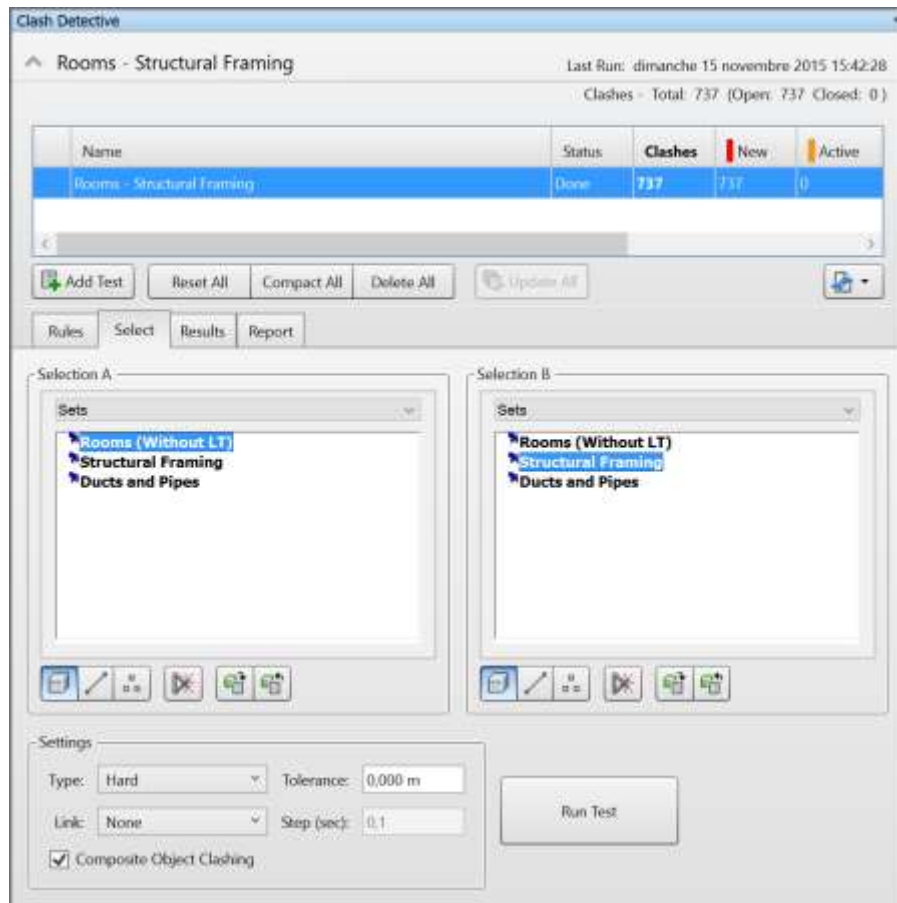


Figure 14 : Create Clash Test

## Automate model update

As CAD file are a work in progress, they are always up to date, and the Navisworks cache file have to be updated as well.

To create these NWC file, we can use the Revit plug-in to export our models to Navisworks.

But it can become quite tedious to open your Revit model every day and export it to an NWC file.

This is why I use the Navisworks Batch Utility, accessible through the Navisworks main menu:

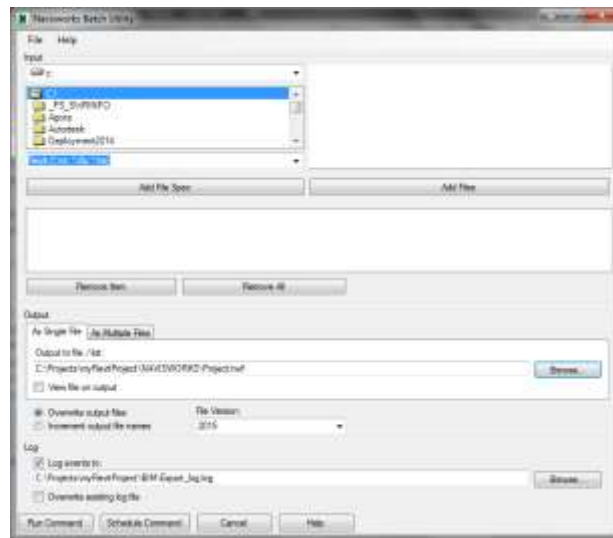


Figure 15 : Navisworks Batch Utility

I create a text file listing paths to every Revit model contained in our project folder. Back on the Navisworks Batch Utility, I open this text file to import file paths: File -> Open -> Select ListRevitFiles.txt

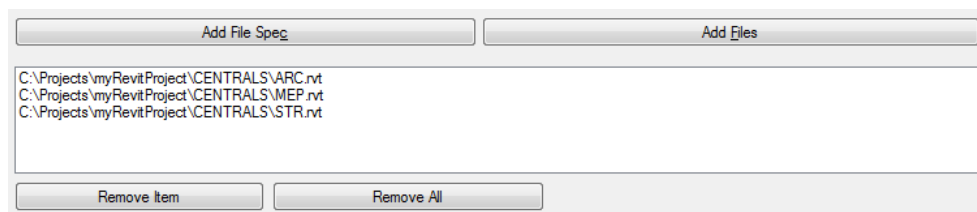


Figure 16 : Files list

As we want to create a single Navisworks File Set (.nwf), we select the “As Single File” Tab, and set the path to our future Navisworks File:

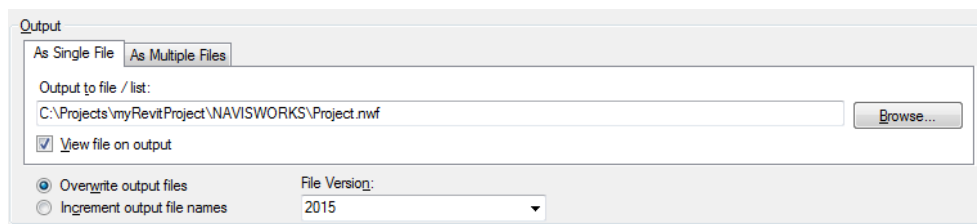


Figure 17 : Output selection

I also select “View file on output” to automatically start Navisworks when conversions are done.

We add a path to a log file in order to know what may happen, and hit “Run Command”.

After a while, Navisworks starts automatically and appends every previously created .nwc file to a new Navisworks File Set.

I am also using this feature to create a NWD file for a set of Revit file.

To do so, you just have to select the “Multiple file” tab and define a target folder for the export:

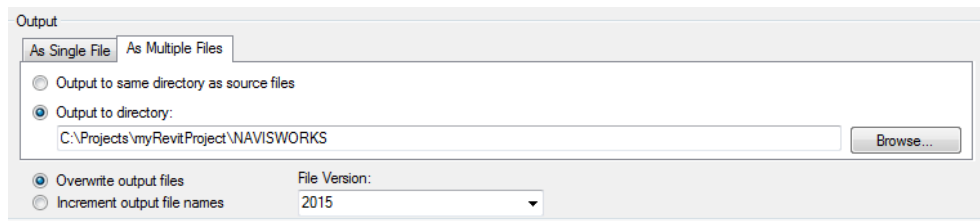


Figure 18 : Create NWD

The Navisworks Batch Utility will convert every Revit file to a NWC cache file, and made it a NWD on the run.

You can also configure the Navisworks Batch Utility to fire up at specified time every day, and automate entirely your conversion process. You will always have a fresh conversion of your Revit production file in Navisworks, without having to convert them every time you open your Navisworks model.

To have more control over the conversion process, and the location of the different files (Revit, NWC, NWD), we use a custom converter, something quite easy to do by using the Autodesk Navisworks API.



## Automate your clash-detection sorting process

### Sorting clashes

Multiple clashes per issues and false positives require us to sort and group these clashes to extract meaningful coordination issues from them.

If you are using clashes as mean to communicate issues, you also have to set up some sort of assignation process, were each clash or group of clash is assign to someone responsible for solving it.

This led us to the tedious business of screening through all clashes, removing false positives and grouping them by issue.

If this process is still largely manual, let see how to alleviate our pain by grouping clash automatically.

### Grouping clash results

#### Introducing the Autodesk Navisworks API Samples

As an introduction to the Autodesk Navisworks API, Autodesk provided a series of code samples to create plug-in for Navisworks. Apart from their pedagogic value, these samples are very useful tools on their own right.

You can find these examples on the Autodesk Developer Network. I also create a compiled version ready to load on Navisworks that you can find on BIM 42 (<http://bim42.com/2015/05/grouping-clash-results/>)

The most useful of these tools is undoubtedly the Clash Grouper. It provides various means of grouping your clash results. This is mostly an automated “Group clash involving this item”.

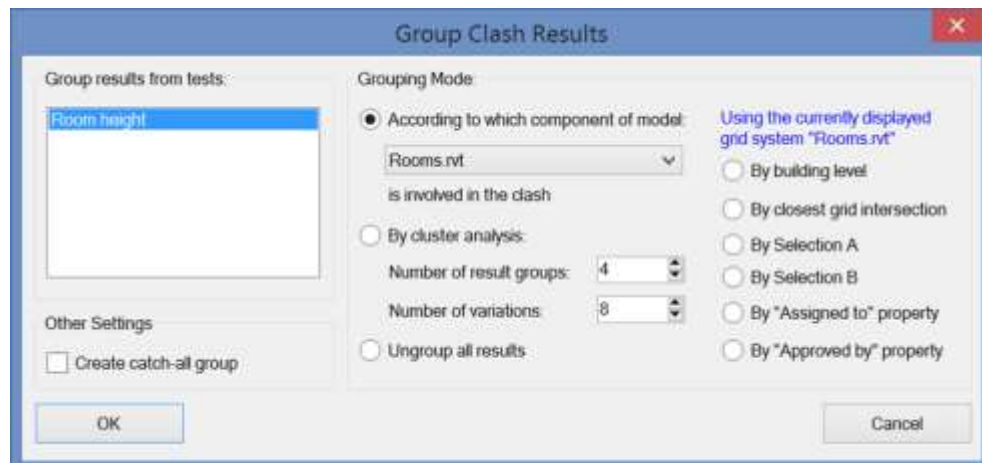


Figure 19 : Grouping clash results

To explain these grouping functions work, I run a clash detection between the blue Selection A and the green Selection B, and get seven clashes, shown here as red dot:

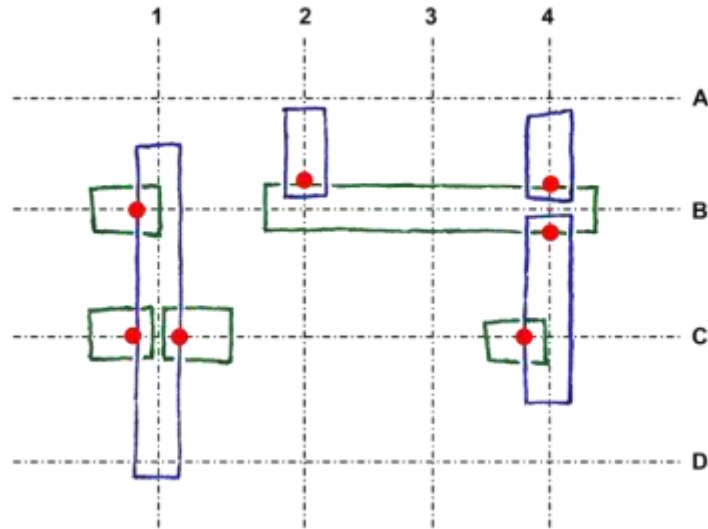


Figure 20 : An example of clash detection

I can now explore how these clashes are grouped.

#### According to which component of model

If an element belongs to the selected model, we create a group of all clashes involving this element. You can use this when your clash detection involves priority elements.

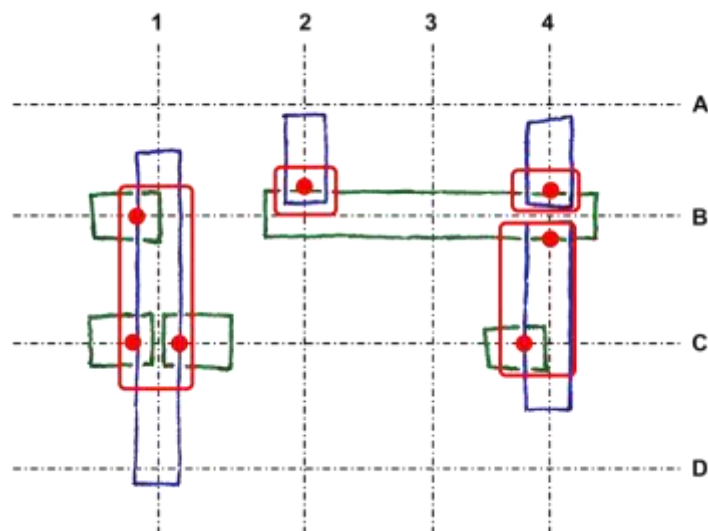


Figure 21 : Group by model

### By cluster analysis

This function search for the optimum grouping solution given the expected number of groups:

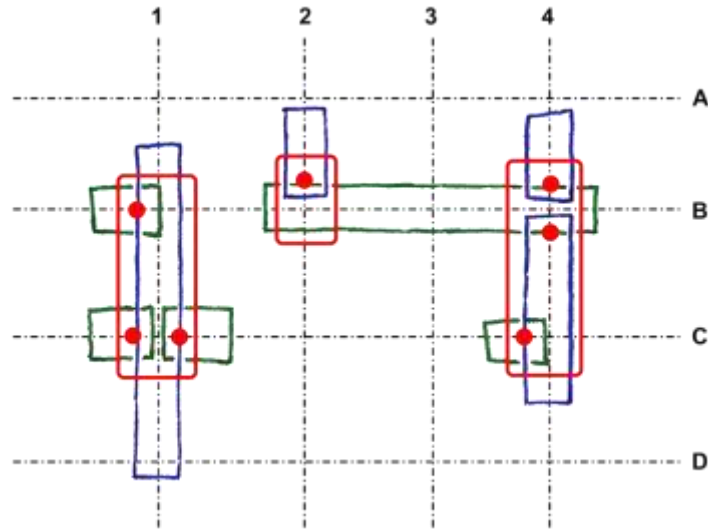


Figure 22 : Cluster analysis

### By building level/closest grid intersection

This groups clashes according to their location. Make sure you have exported Revit grids and level in your Navisworks model to enable this functionality.

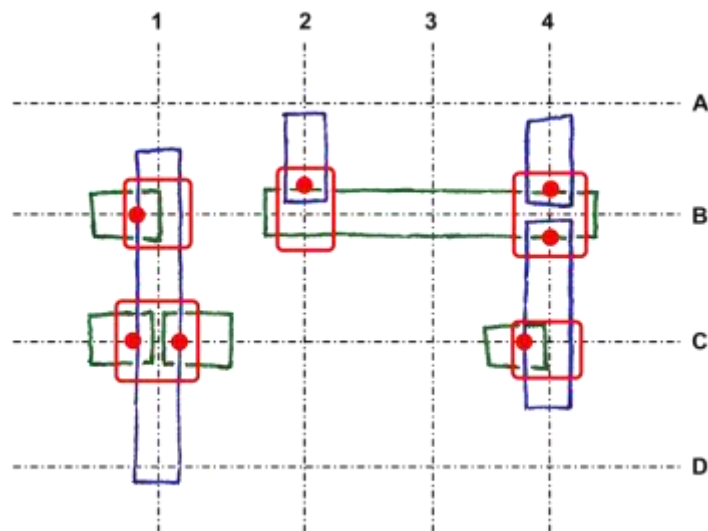


Figure 23 : Group by grid intersection



### By selection A/B

A custom development. This is pretty similar to the “model” grouping, but it use the selections used to create the clash test instead of a model. You can group all clashes involving an element of the chosen selection:

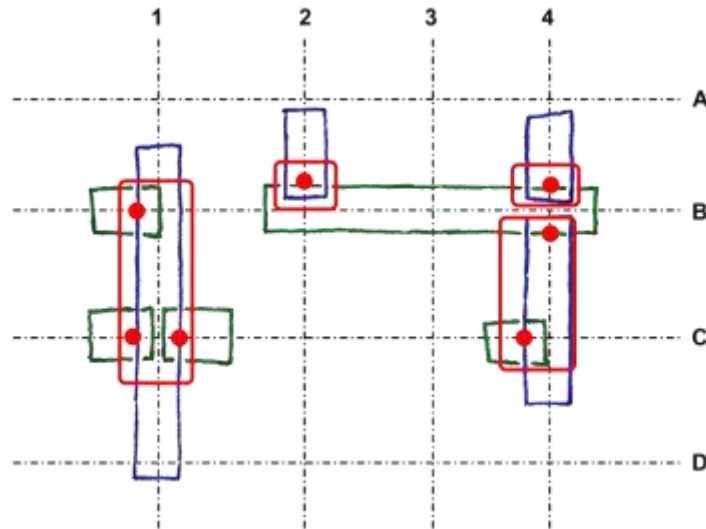


Figure 24 : Group by selection

### By “Assigned To”/“Approved By” property

Finally, if your process involve some kind of validation workflow, you can thereafter group all clashes by their assignment or their approval.

### Using the Clash Grouper

This plug-in can be an extremely powerful, especially when you are working with very specific clash detection, where the grouping method became obvious.

For example, in our minimum required headroom example, we can group the result by structural framing to give to the structural engineer the list of beam below the required headroom.

Since the grouping is now automated, you can even allow yourself to change our grouping logic in the middle of your process. For example, you can use Group by Selection to review and assign your clashes, then group them by “Assigned To” to give a report group by assignments.

## Measure coordination progress

Finding meaningful clashes is all good and well, but does not provide anything valuable if the issue is not solved. Of course, I am not here to explain how to solve issues, but more to give you how we deal with these clashes we find.

### Clash vs Issue

A clash is not an issue, but the symptom of the issue. Therefore, you should not bother with thousands of meaningless clashes, but find real issues amongst these clashes.

However, global clash detection can be useful as a performance indicator of your coordination.

### Know your progress

One of the main challenges of the engineering process involves its uncertainty. Requirements of a project can sometimes be ambiguous and will emerge over time, and measuring the progress of such an evolving process can be challenging.

One of the metrics we use to track coordination progress is a clash counter. The idea is to measure over time the number of clash remaining in our model. This clash count may not reflect the actual number of issues to be solved, but give us a metrics to ensure our coordination efforts are worthwhile.

### Collecting data

Measuring coordination progress start by collecting clash reports in a standardized format. This collection process may vary regarding your design process. In our case, the Navisworks model is updated on a daily basis along with its clash tests.

A series of clash test is used solely for reporting purpose. These clash tests do not focus on subjects like what we described previously, but count the number of clashes between trades, and reflect the evolution of our coordination progress.

Therefore, we export everyday an xml clash report containing all clashes from a series clash test defined in our Navisworks model.

This report is not very useful per se. In fact, there is generally too much clashes to sort them into something useful. And if by chance you haven't that much clashes, you probably don't need clash detection in the first place. But this clash report does nevertheless represent the state of our coordination at a given date. The fewer clashes we have, the better.

Day after day, these reports create the raw data for a journal of our spatial coordination.










Nom	Modifié le	Type	Taille
 20150127_AIG_INFRA.xml	27/01/2015 17:21	Document XML	1 889 Ko
 20150209_AIG_INFRA.xml	09/02/2015 17:38	Document XML	6 220 Ko
 20150209_AIG_P13.xml	09/02/2015 16:22	Document XML	673 Ko
 20150209_AIG_P16.xml	09/02/2015 16:10	Document XML	308 Ko
 20150209_AIG_P19.xml	09/02/2015 15:49	Document XML	41 Ko
 20150210_AIG_INFRA.xml	10/02/2015 15:40	Document XML	6 721 Ko
 20150210_AIG_P13.xml	10/02/2015 14:40	Document XML	673 Ko
 20150210_AIG_P16.xml	10/02/2015 14:33	Document XML	1 057 Ko
 20150210_AIG_P19.xml	10/02/2015 13:56	Document XML	124 Ko

Figure 25 : Daily clash reports

Periodically, we compile them into something more visual.

To do so, we create a single table (in .csv) listing every clash reported during the project, along with the date of the clash report.

We have a custom application for extracting the information from Navisworks XML reports. The process is somehow automated, but the result is the same, a large database of every clashes detected during the design.

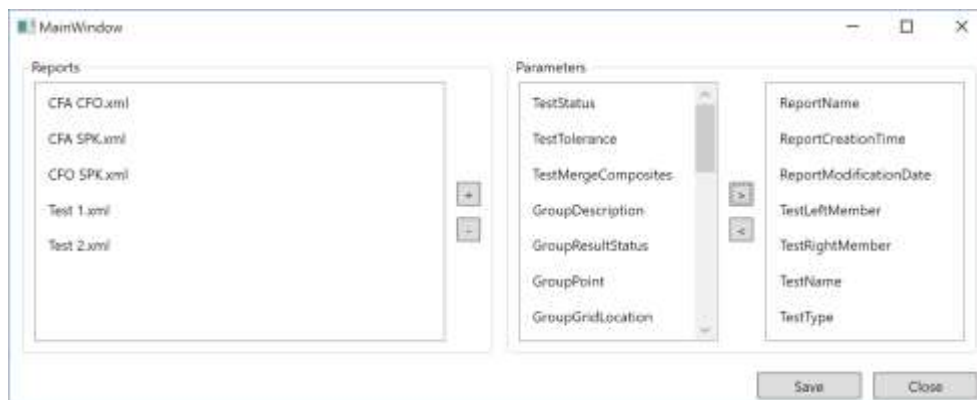


Figure 26 : The clash logger

You can download this application on GitHub (<https://github.com/simonmoreau/ClashLogger>), along with its source code.

To use it, load a set of clash report in the interface, select the properties you want to appear in your final report, and save the resulting CSV file

Once we have every clash in a handy (and pretty large) .csv file, we use Tableau to create a nice visualization out of it, and let everyone in the office follow the progress of the coordination.

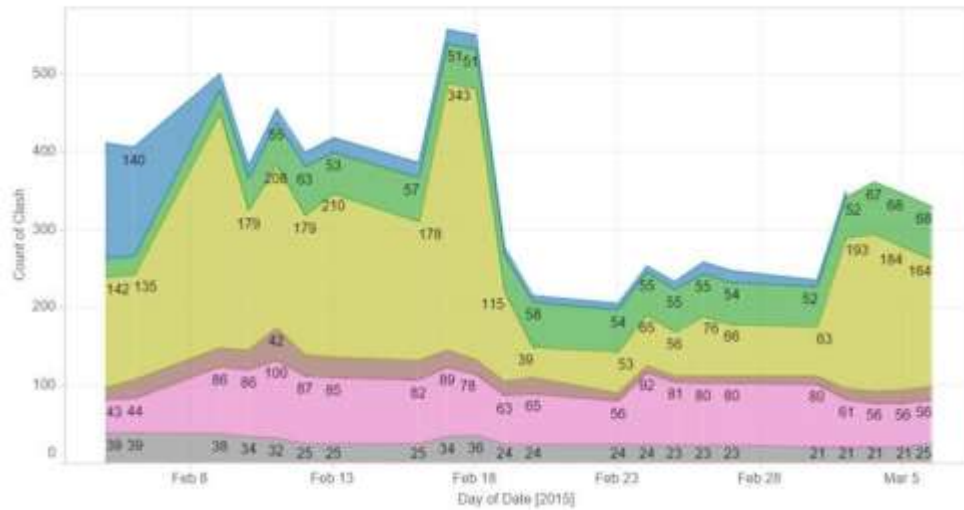


Figure 27 : Evolution of the number of clashes