The Truth Is In The Model

Utilizing Model Checking to Rate Learning Success in BIM Software Courses

Andreas Dieckmann[1], Peter Russell[2]
[1,2]Chair for CAAD, RWTH Aachen University
[1,2]http://caad.arch.rwth-aachen.de
[1,2]{dieckmann|russell}@caad.arch.rwth-aachen.de

Model checking is one of the core methodologies of Building Information Modelling (BIM). It allows us to quickly evaluate models based on custom criteria. While there are known examples of the integration of model checking into the course content of design studios, there is no literature on utilizing this methodology to help in the grading process of BIM software courses. This paper presents a project that applies model checking to the task of rating the learning success of students in such a course. The main project goals were increasing the objectivity of the ratings and reducing the time necessary to process a large number of submitted models. The paper describes a possible approach to categorizing and organizing model checks in an educational context and outlines a proven and tested workflow for automating the rating and feedback process.

Keywords: *BIM Education, Grading, Evaluation, Model Checking, Automation*

CONTEXT

Nowadays, Building Information Modelling (BIM) is being taught at an ever increasing number of architecture schools: the overall concepts and processes as well as the use of specific BIM software. A lot has been written about how to teach BIM, with a lot of papers focussing on ways of integrating BIM into design studios (Barison & Santos, 2010) - but nothing about how student work is graded, specifically the quality of the BIM models students submit for review.

One of the main advantages of BIM over conventional CAD is the semantic nature of the model which permits querying the model like a database. The process of verifying that a BIM model meets certain criteria or rules is known as model checking. Typical use cases for model checking include collision detection, egress path checking etc. (Hjelseth & Nisbet, 2010), but also lends itself to more basic applications like model content checking.

This paper is about applying the aforementioned method to the task of rating the learning success of large numbers of students in BIM software courses using the tools available within the BIM environment itself.

PROJECT PARAMETERS

From elective to mandatory subject

At RWTH Aachen University, BIM.Basics - the introductory BIM course - always used to be an elective with 30 to 40 students per semester since it was first launched in 2008 (Dieckmann, Russell, & Wittenberg,

2011). That changed in 2012 when the new mandatory module Integrated Planning was introduced into the second and third semester of the Master of Architecture curriculum. BIM.Basics became the only mandatory course in that module as all other courses in the module (e.g. "Model-based Building Performance Optimization", "Model-based Procurement & Tendering" etc.) build upon the foundations laid in BIM.Basics. Accordingly, every semester approximately 100 students now enroll in the class (see fig. 1). Alternatively, it can still be taken as an elective before the Integrated Planning module so as to allow interested students to enroll in a larger number of electives in that module.

BIM.Basics introduces the students to the topic of Building Information Modeling - its history, its principles, its methodology. However, the larger (practical) part of the course is dedicated to learning to work with BIM software. The approach here is strictly project-oriented: Throughout the term, students continuously develop a project. They are encouraged to bring their own (current) project or, alternatively, they can work on a brief provided by the Integrated Planning module. The class content is structured into four phases:

1. **Project setup** (three weeks, learn how to setup a project and model the site, building volumes and site elements),
2. **Building model** (four weeks, learn how to model a building using building elements),
3. **Model representation** (three weeks, learn how to derive 2D and 3D representations from the model and format them),
4. **Model data** (three weeks, learn how to enrich the model with additional data, how to extract data from the model and how to author custom components).

At the end of each phase, the students have to submit their BIM model for review through a website. That means that each time 100 submitted projects have to be evaluated and - in the first three phases - individual feedback for the students needs to be generated. This feedback is focussed mainly on the correct use of the BIM software package and whether the submitted work meets the criteria set in the assignment.

The issue of scalability

When the course was still an elective with 30 to 40 students per semester, this process was manageable. A teaching assistant (TA) would assess the submitted models according to a check list and the lecturer would subsequently rate the overall quality of each submission. However, this process does not scale well to a course of 100 students per semester. The two main reasons for this are the following:

- **Lack of objectivity:** Due to the large number of submissions (ca. 400 per semester),

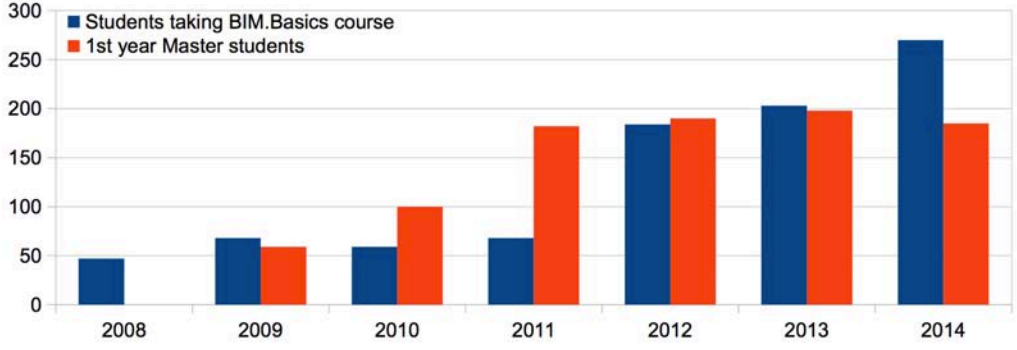

Figure 1
Number of students taking introductory BIM course versus 1st year Master students at the Faculty of Architecture, RWTH Aachen University

several TAs are needed to get the assessment done in a timely manner. However, according to an evaluation performed on the assessment scores of the summer term of 2013, average check results vary significantly between different TAs. The evaluation of the scores showed more than 18% deviation from the overall average score over the course of the entire term, although the TAs were even reassigned to other students randomly after each phase (see fig. 6). That much is clear: the TAs show varying levels of leniency toward their peers when it comes to assessing the quality of their work as well as varying levels of success in identifying issues in the submitted BIM models.

- **Expenditure of time:** The manual checks performed on the model by the TAs are complex. They involve switching model views, isolating elements or element categories, checking element properties and - most importantly - a lot of decision-making. Hence they are very time-consuming. Also, with each additional student, the demand on the lecturer's time grows, too - especially if he may need to (partly) re-check the TAs' assessments (and possibly increase their objectivity).

Therefore the evaluation process needed to be automated as much as possible within reasonable time constraints [1]. The main goals were increased objectivity and saving of time. Additionally, the engineer's mind set in to the question, *"Could we use the BIM software itself to achieve these goals?"*

APPROACH

Model check categorization

In general, most model checks performed in this project could be classified as what Hjelseth and Nisbet (2010) refer to as model content checking (with some exceptions that belong to the domain of validating model checking). Based on the evaluation check lists from previous semesters and the course assignment, necessary checks were grouped into three categories:

1. **Mandatory tasks.** These checks form the base score. Students are penalized for missing items in this category.
2. **Optional tasks.** These checks act as positive modifiers to the base score. Missing items in this category do not have any influence on the final score.
3. **Common mistakes.** These checks act as negative modifiers to the base score. Only if the model is free of mistakes does this category have no influence on the final score. Having carried out BIM courses for seven years, the authors have seen a plethora of common mistakes that have made it onto the check list.

Thus, the overall score can be expressed in a very simple formula:

$$\text{Score} = \sum_{\text{Mandatory}} + \sum_{\text{Optional}} - \sum_{\text{Mistakes}} \qquad (1)$$

The checks can also be categorized according to the type of information that is extracted from the model:

- Type 1: **The presence (or absence) of certain elements** (or component categories) in the model (*"Has the student used interior walls?"*). This would result in a true / false rating.
- Type 2: **The evidence of the use (or non-use) of certain tools** or features of the software (*"Do all exterior walls have multiple material layers assigned to them?"*). This would result in a true / false rating, too.
- Type 3: **The evidence of the correct (or incorrect) use of certain tools** or features of the software (*"Have roofs been modelled as floor slabs?"*). Again a true / false rating.
- Type 4: **The quality of the implementation of certain component categories** in the

model (*"What is the overall quality of modelled walls in this model?"*). This could also take into account the complexity of the model and would be rated as either positive, neutral or negative.

While checking for type 1 information answers the question whether a student has used a certain object category or tool at all, checking for type 2 and 3 information will answer the question if he or she has managed to gain in-depth knowledge of that tool. However, only type 4 information will answer the question if the student is able to apply the skills and knowledge gained during the course to an actual architectural problem.

Automation strategy

Yet another way of categorizing the checks was by assessing whether a particular check could be automated or not. It turned out that for the mandatory and optional tasks, non-automated checks are almost exclusively qualitative assessments. In the common mistakes category, however, some checks could not be automated due to the nature of what needed to be checked - in some cases because the software development of the check algorithm would have taken disproportionately long, in other cases because a human eye is needed to make the decision if something is really a mistake or not. For the first iteration of the model check automation approximately two thirds of all checks were categorized as automatable which later proved to be correct albeit with some deviation between the respective phases (see fig. 2).

Converting the manual checking routines of previous semesters to semi-automated routines also meant an increase in the total number of checks performed. Where previously there might have been only one (rather complex) manual check needed to rate a certain aspect of the model, there would now be several automated checks and quite likely one (albeit extremely simple) manual check. As an example, let's examine how topography used to be checked manually and how that process has been automated.

The manual check of a topography element involved the following actions:

1. Determining if there is a topography element
2. Determining if the topography element includes subregions
3. Determining if the subregions have different materials assigned to them
4. Determining if the topography is entirely flat
5. Rating the overall quality of the topography elements in the model

Using a check list, the TA would then assign an overall rating for topography based on his findings that could have one of the following values: 1 (positive), 0 (neutral) or -1 (negative). With the new automated approach, the software will automatically obtain the information from steps 1 through 4, leaving only the qualitative assessment in the hands of the TA. In this particular case, all results obtained through automation will have a true / false rating (1 / 0), while the TA's assessment will continue to have values of 1, 0 or -1. The results are then fed into a formula that computes the end result for the topography elements:

$$\mathrm{Topo}_{\mathrm{Result}} = \overline{\mathrm{Topo}_{\mathrm{Auto}}} \cdot \frac{\mathrm{Topo}_{\mathrm{Quality}} + 2}{2} \qquad (2)$$

While previously the results would range from -1 to 1 (with only three possible values: -1, 0 and 1), they now range from 0 to 1.5 with a host of possible values in between, making for a more fine-grained overall score. More importantly, the qualitative assessment merely acts as a modifier now, placing a stronger weighting on the automated check results. All check results are stored in a spreadsheet (see fig. 3) that computes the following data:

- The results for each element category
- The results for each check category (mandatory tasks, optional tasks & common mistakes)

- The overall score from the three check categories (see formula 1)
- The rank of each model score in the entire list of models
- A qualitative rating of each model based on the overall score
- A qualitative rating of each model based on its ranking

Additionally, the TAs can include comments for internal use (i.e. information about each model that is not reflected in any of the check results).

IMPLEMENTATION

Dealing with idiosyncrasies

In practice and research, model checks are usually performed on IFC files using dedicated software like Solibri Model Checker (SMC) [2]. A good overview of rule-based checking systems (albeit with a focus

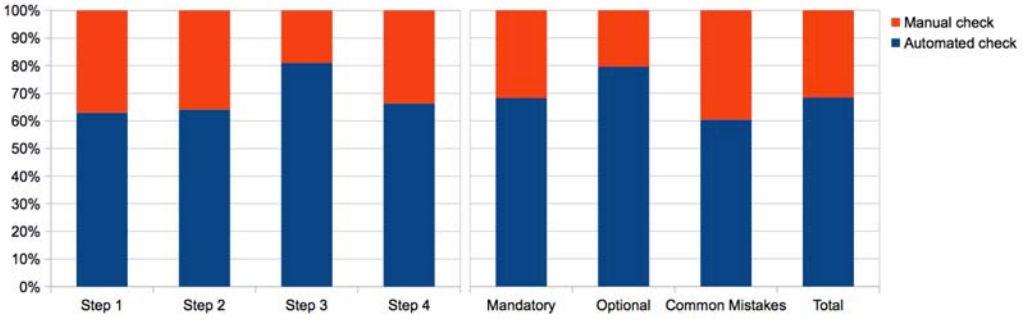

Figure 2
Percentage of automated checks by category and phase

Figure 3
Screenshot of the grading spreadsheet for phase 1

on code compliance) can be found in Eastman et al. (2009). There are good reasons for performing the check on an IFC file, the chief among them being interoperability.

However, in the context of a BIM software class that teaches the use of a specific proprietary BIM authoring software, running the models though an IFC based model checking software may not be the wisest choice. Every software has its idiosyncrasies that are reflected in the output we create with the software. When teaching a software and trying to evaluate the resulting output, those idiosyncrasies therefore need to be taken into account, because they may actually tell us a story about how the software was used. With regard to BIM authoring software, the original model will most likely reveal more in that respect than an exported IFC model.

For the BIM.Basics course, the software of choice is Autodesk Revit [3]. Elucidating the choice of software is neither part of this paper nor does it bear much relevance as the principles in this research are transferable to other authoring environments, but suffice it to say that the authors are fluent in several BIM authoring tools and had their reasons.

With Revit, one of many good examples for the aforementioned idiosyncrasies is the topic of view templates. As the name suggests, view templates are an infrastructure to store common settings for views in the model, thereby greatly facilitating the consistent formatting of multiple views of the same type (e.g. all floor plans of scale 1:200). Clearly, this is an important skill to teach as it has a profound impact on the economical use of the software (as opposed to formatting each floor plan separately). However, this type of information will not make it into an exported IFC model and could therefore not be checked by SMC or similar software.

It was therefore obvious that in order to achieve the broadest check results (and give the most valuable and tool-specific feedback to the students), the model check would have to be performed on the Revit model itself. This way, it could be ensured that very specific cases could be tested regarding the use (or mis-use) of certain tools within the software.

Choice of development environment & subsequent implementation

It just so happens that Revit already has a model checking tool in the form of a plugin called Model Review since its 2010 release [4]. This plugin is available to all subscription customers of Revit. Basic checks can be authored through the Model Review user interface, but more complex custom checks can only be achieved with plugins. These plugins require a significant amount of coding in C Sharp or Visual Basic on top of the Revit API (Bell, 2011). Each plugin has to be built as a dynamic link library (DLL) using Microsoft's Visual Studio [5].

An alternative presented itself in the form of Dynamo [6], a recent open source project that is mainly marketed as a computational design tool. In essence, Dynamo is a visual programming environment (very similar to Grasshopper [7]) that was originally developed to run on top of Revit but has since been released as a standalone application as well. Among other features, Dynamo allows for the creation of custom Python-based components that in turn can access the Revit API.

In the end, Dynamo was chosen over Model Review for the following reasons:

- **Ease of use.** At least for non-programmers, working with Dynamo is more intuitive than coding in Visual Studio.

- **Accessibility.** At some point in the future, it may make sense to offer the model checking package to students for self assessment. While the access to Model Review is limited to subscription customers, Dynamo is an open source software that already has a web-based infrastructure for sharing custom code in place.

- **Economy.** Dynamo comes with a full library of ready-to-use built-in functions for things like logic, list management etc. Hence, only

specific queries of the Revit model needed to be built.

Implementing the model checks required building a total of roughly 50 functions (not counting sub-functions), referred to in Dynamo as custom nodes (see fig. 4), of which over 80% have Revit-specific functionality while the rest can be classified as general purpose. Since the model checker for the final project submission (see fig. 5) includes 67 checks and most of the model checks make use of several of these custom nodes (three per check on average), it is obvious that the reusability rate of the custom nodes is rather high. All of the custom nodes have since been publicly shared through Dynamo's package manager [8] and their functionality has been documented online [9].

Workflow

For each of the four phases, the resulting workflow for the teaching assistants entails the following steps for each checked model:

- Open a Revit model
- Run the appropriate Dynamo graph for the current phase
- Running a model check will generate a number sequence that contains the results of every individual aspect checked for the current phase. That number sequence can then be copied to the clipboard.

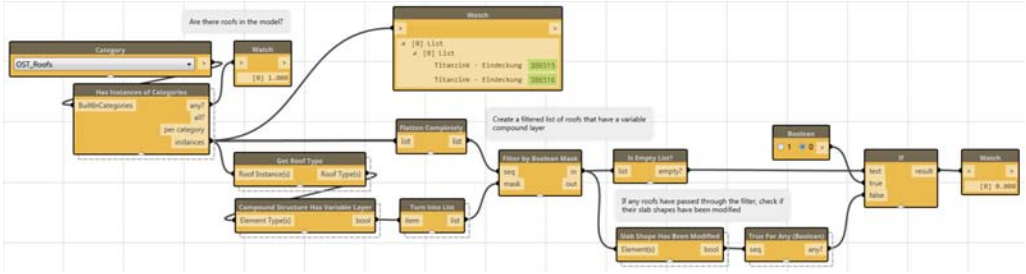

Figure 4
Example of a Dynamo-based model check (examining roofs). Custom nodes can be identified by a dashed shadow outline.

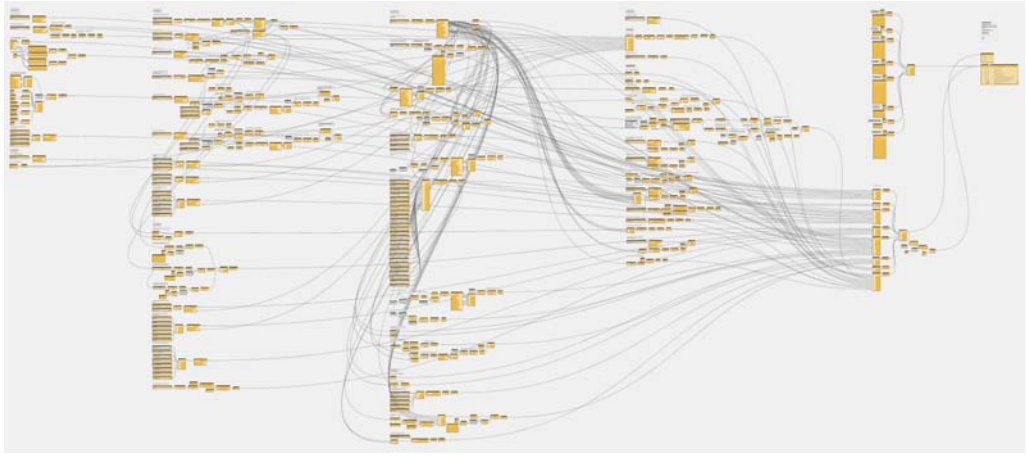

Figure 5
Screenshot of the Dynamo graph checking the final project submission

- Switch to the spreadsheet for the current phase and paste the results into the appropriate line.

- Perform the necessary manual checks for the given model and enter their results into the spreadsheet. The spreadsheet includes a checklist for each phase outlining the criteria for each manual check.

- Score and ranking are automatically computed on the fly by the spreadsheet.

- Where necessary, add some comments on the current model about encountered issues not covered in the model checks.

When all teaching assistants have submitted their spreadsheets, the lecturer has to perform the following actions:

- Visually examine all models and the respective TA comments

- Where necessary, write some individual feedback for the student for issues not covered in the model checks (only phases 1 through 3)

- Send the individual feedback to the students (only phases 1 through 3). This process can easily be automated using the Mail Merge Wizard available in most office suites. To that end, the spreadsheet contains a lookup table that gets populated with boilerplate text based on the individual check results for each student.

- Grade the work based on the check results and individual examination of each model (only phase 4)

CONCLUSION

Preliminary results

The presented automation process has been in use for only one full semester, so there is no long-term evaluation of the new grading process yet. However, the following results are already obvious:

- Automating a large part of the process results in more objective results, i.e. the deviations between the individual assessments of the various TAs have a significantly reduced effect on the overall assessment of each submission (see fig. 6). The goal of increased objectivity has therefore been achieved.

- The estimates of saved time were more or less correct, depending on the submission phase. Some submission phases have a higher automation rate than others (see fig. 2), therefore the time savings vary. Across the entire semester, the average rate of time savings amounts to roughly 50%. Thus, the goal of speeding up the model review process has

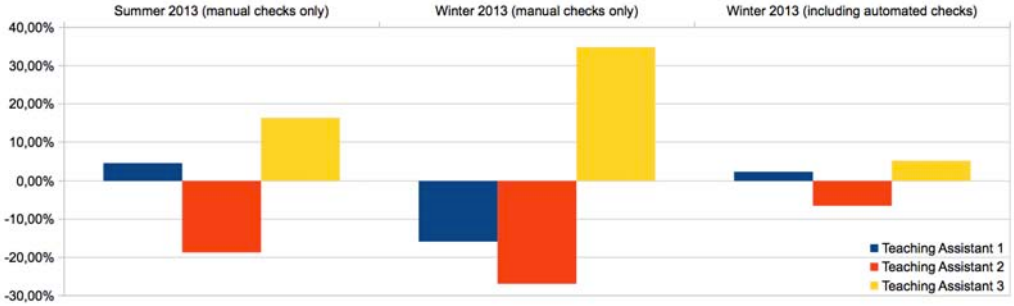

Figure 6
Deviation of average model check results from the mean by teaching assistant (anonymized)

been achieved, too, in turn proving the economic viability of automating the process.

- As has been described above in the topography example, automating the model checking process has led to a significantly larger number of model checks. A welcome side-effect of this change is that the feedback generated out of these more detailed checks is considerably more precise than feedback in previous semesters, thereby giving the students a better chance at improving their models for the final submission.

- Automating the model checks has actually made it possible to check several additional aspects of the models that previously were too tedious and / or time-consuming to check by hand, again giving students better opportunities to improve their models.

Boeykens et al. (2013) have already shown how model checking can be meaningfully integrated into a studio context. The research in this paper has added the case of BIM software classes. How is this research transferable? The principles laid out in this paper can, of course, be applied to BIM software classes in general (without having to use Revit). While the method of implementation and the rate of automation will most likely vary depending on the software package used, it is foreseeable that the objectivity of grades will increase and that the automation process will lead to time savings. As mentioned earlier, another likely development in the future may be to provide such a model checking tool to the students directly for continuous self assessment. Another possible area for model checking in education could be the assessment of models created in a design studio like it has become customary in many architectural competitions ever since the Vestbanen competition in 2009/2010) [10].

Further work

There is, of course, always room for improvement. At the time of writing, the model checking process is only semi-automated in that a separate model check needs to be run for each model. Although this in itself is already a vast improvement compared to an entirely manual evaluation of each model, it is not ideal yet. Accordingly, a solution for batch processing of models is currently under development. It makes use of the Revit Test Framework (RTF) [11], a tool that was initially developed for unit testing new Dynamo builds. Although unit testing still is its main purpose, the RTF can be used for automating other operations on Revit models as well. Using the RTF will make it possible to process an entire folder structure of models. The Dynamo-based model checks had to be altered slightly in order to automatically update a CSV file after checking a model, thus building a list of examined models that can later be imported into the master spreadsheet. It is expected that this tool can go into operation during the current semester, once again reducing the amount of time spent with each model significantly.

The model checks themselves form another area of potential improvements. After evaluating the individual feedback from the last few semesters (i.e. comments on issues in the models that are not covered by a model check yet), several issues have been identified that could be automatically assessed in the future.

Acknowledgement

Many thanks to Ian Keough, the father of Dynamo, for helping out on the development of the batch processing of models and for creating Dynamo in the first place.

REFERENCES

Barison, MB and Santos, ET 2010 'BIM Teaching Strategies: An Overview of the Current Approaches', *Proceedings of the International Conference on Computing in Civil and Building Engineering (ICCCBE) 2010*, Nottingham, UK, pp. 577-583

Bell, RR 2011 'Writing Your First Autodesk Revit Model Review Plug-In', *Autodesk University 2011*, Las Vegas, NV

Boeykens, S, de Somer, P, Klein, R and Saey, R 2013 'Experiencing BIM Collaboration in Education', *Computation and Performance – Proceedings of the 31st*

eCAADe Conference (Vol. 2), Delft, Netherlands, pp. 505-513
Dieckmann, A, Russell, P and Wittenberg, B 2011 'Educating the Masses: Teaching BIM from Undergraduate to Postgraduate Level', *Autodesk University 2011*, Las Vegas, NV
Eastman, CM, Lee, JM, Jeong, YS and Lee, JK 2009, 'Automatic rule-based checking of building designs', *Automation in Construction*, 18(8), pp. 1011-1033
Hjelseth, E and Nisbet, N 2010 'Overview of Concepts for Model Checking', *Applications of IT in the AEC Industry, Proceeding of the 27th International Conference - CIB W78*, Cairo, Egypt
[1] http://xkcd.com/1205
[2] http://www.solibri.com/products/solibri-model-checker
[3] http://www.autodesk.com/products/autodesk-revit-family/overview
[4] http://thebuildingcoder.typepad.com/blog/2009/11/model-review.html
[5] http://www.visualstudio.com
[6] http://www.dynamobim.org
[7] http://www.grasshopper3d.com
[8] http://www.dynamopackages.com
[9] https://github.com/CAAD-RWTH/DynamoSamples
[10] http://www.aec3.com/de/referenzen/Vestbanen-en.htm
[11] https://github.com/DynamoDS/RevitTestFramework