# Automation of data transfer between a BIM model and an environmental quality assessment application.

Adam Piaskowski[1*], Reinis Petersons[1], Simon Christian Swanström Wyke[1],
Ekaterina Petrova[1] and Kjeld Svidt[1]
[1]Department of Civil Engineering, Aalborg University, Denmark
*email: akptech@outlook.com

## ABSTRACT

Automating the transfer of Building Information Model (BIM) data to assess energy performance is still far from straight forward. The approach of using Industry Foundation Classes (IFC) or Green Building Extensible Modelling Language (gbXML) is only justified if the parametric data is accurate and explicit and if the designers are proficient at console coding. Mapping parameters to rigid schemas and calculating quantities and volumes can be time-consuming and require software engineering knowledge. Using Action Research Methodology, two student groups investigate methods of data transfer and parametric data derivation process. To ensure correct data transfer, we propose an API-based proprietary pre-computation which connects Revit objects and parameters with an Excel-based Indoor Environment Assessment tool (IV20). IV20 is currently under development at Aalborg University, and it serves us as a set of Business Rules, defining required inputs needed for an energy assessment. This case study focuses on automating features that perform extraction, calculation, and storage of input data, based on a set of predetermined user requirements. Dynamo Visual Programming Language (VPL) is used to extract sample information regarding room and element properties, façade orientations, and building surroundings. We conclude that Revit-Dynamo-Excel (RDE) approach can significantly automate data transfer, but it requires explicitly modelled objects, agreed exchange requirements, and a well-defined and communicated calculation method. Our approach does not eliminate IFC interoperability, but rather enhances the proprietary model information advancement, prior to export.

**Keywords:** BIM, data transformation, data exchange, visual programming, indoor environmental quality

## 1. INTRODUCTION

The Danish government has put a significant effort in promoting the sustainable renovation of existing buildings, placing a strong emphasis on the improvement of indoor environmental quality, building energy performance, and occupants' comfort (Gram et al, 2010; REBUS, 2015; Harrestrup & Svendsen, 2015). When renovating buildings, it is necessary to ensure satisfactory living conditions, and evaluate the building state. This can be done by benchmarking its existing performance against modern building regulation requirements. Concurrently, Building Information Modelling (BIM) and modern surveying technologies have reshaped the renovation workflows and have allowed setting more accurate and efficient energy performance targets (Habibi, 2017). Semi-automation, when coupled with a definitive list of inputs and outputs can assist speedy delivery of a complete energy analysis cycle.

When designing a BIM application, the standardized process involves formulating Exchange Requirements (ER) by following international standards such as Industry Foundation Class (IFC) Information Delivery Manuals (IDM), and validating information exchange against Model View Definitions (MVDs), to ensure complete and correct data transfer. MVD's are used as holistic data

subsets, validating that certain design elements are exchanged, specific to the intended use cases (Borrmann, König, Koch, & Beetz, 2018). ER can also follow custom-made standards such that follow a structure as the British BIM Protocol Employers Information Requirements (EIR) (CIC, 2013).

While the ERs are key to defining case specific inputs and outputs for energy analysis, further action towards IFC software validation against MVDs does not share its intended specificity, as the subsets are wide and generic, and released in only a few versions (Pinheiro et al., 2018). This process may work for big software vendors, well acquainted with using MVDs, but for quick analysis between specific stakeholders, this can cause inconsistencies and costly project delays. There seems to be a need for a simpler, more commonly communicated workflow, which is based on standardized principles, yet is flexible and easily implementable directly by design consultants, without the need of expensive validation methods. Although we aspire to find an effective workaround to the complexity of using Model View Definitions (MVDs), we follow the standardized Exchange Requirements (ER) workflow, proposed by the buildingSMART ISO 16739-1:2018 standard for BIM information exchange (ISO, 2018).

Validation, coupled with the rigidity of IFC schema, regional versioning, and holistic MVD's opens a window of opportunity for an alternative workflow using tools currently familiar to the industry professionals, who are often not fluent in console coding. It is why we propose to use Dynamo Revit, as a tool capable of extracting, calculating and linking building elements' properties and attributes, to MS Excel spreadsheets. We thereby aim to contribute to the subject of data transfer, by using Visual Programming Languages, instead of console coding.

The reason for choosing Microsoft Excel is its maturity and widespread adoption as a preferred file-based database tool. Similarly, the wide adoption of Autodesk Revit has placed the software amongst most popular BIM authoring tools (Eastman, 2016). Even if IFC MVDs will become more customizable and simpler to implement, Revit workflows will still be applicable as BIM authoring tools. As neutral formats i.e. *.ifc* are inadequate to re-create the native formats, the neutral formats must be augmented by native formats such as *.rvt* for Revit (Borrmann et al., 2018).

Alternative workflows proposed by Kamel & Memari, (2019) require the user to export a file in gbXML format. Pinheiro et al., (2018) recognizes the use of IFC4, after identifying that gbXML uses only centreline representation for geometry area calculations. These processes, although derived from IFC, require 3rd party software specializing in building energy analysis, such as Modelica or EnergyPlus, which must be purchased on top of the authoring tool. Using data structured in standardized IFC schema, Bazjanac (2008) recognized the need for small workarounds as amongst other issues, data content may need simplification prior to extraction.. Using IFC4 and Modelica can result in bottlenecks when using IFC for Building Energy Design, identifying a need for custom MVDs (Andriamamonjy, Saelens, & Klein, 2018). All the above methods rely on proprietary geometry, deriving from software such as Autodesk Revit. As a result, regardless of file format, data from proprietary authoring software will always be utilized to inform later design stages.

It is rarely the case that geometry data from Revit is ready for IFC export (Kassem, Kelly, Dawood, Serginson, & Lockley, 2015), regardless of Level of Detail. When creating Revit models, default Revit quantities are often calculated inaccurately or follow undesirable calculation assumptions. Ensuring information validity can be challenging, and therefore it is important to identify information requirements and calculation methods before exporting data values for energy analysis. As data input for energy assessment stems directly from Revit geometry, and the geometry can be modelled to a varied level of detail, in multiple ways and with varied parameter naming, attention is brought to following standard processes. Due to a diversity of project participants with varied BIM knowledge, even experienced Revit users make mistakes when transferring information to other professionals outside of their own expertise.

There are ways to ensure that the information agreed in ER is complete. Following IFC standardized workflow, a well-defined Information Delivery Manual can explicitly specify information content, geometry content, data types, attributes, attribute and object naming, level of detail, and other project-specific principles. IDM guidelines (Wix & Karlshøj, 2010), though specified to support IFC workflow, are a relevant tool when deriving non-IFC based Exchange Requirements. Alike IFC, proprietary models can be validated against Exchange Requirements both by the sender and the recipient of BIM data. Exchange Requirements consistency checks are there to ensure the information transfer is valid, and in the event of missing information, the cost incurred is correctly assigned to the

party responsible for error correction. Although information validation is not a part of this article, Dynamo functionality can be utilized to pre-validate, and post-validate ER parameter content.

Having a set of in-house Dynamo scripts can help an architect, or an energy consultant, verify if the model can be used for energy simulation. Creating uniformly named parameters is necessary for mapping to be permanent. Dynamo, by plugging directly to Revit API (Borrmann et al., 2018), can link model data with MS Excel in real-time. Dynamo Visual Programming Language (VPL) is using nodes which act as pre-defined code-blocks with built-in functionality and direct connectivity to the opened Revit project.

VPLs differ on their level of granularity. The more granular, the higher the detail of data processing visible to the user, at a trade-off of increased complexity. Dynamo Revit allows a user to use nodes ranging from single values such as strings or integers to complex combinations of code consisting of many sub-nodes. Dynamo Revit environment, although a part of Revit – a licensed software, imitates to some extent an open-source community, by embracing knowledge sharing ("What is Dynamo? | The Dynamo Primer," 2018).

The data can be stored within Revit parameters, or in an external database such as dRofus ("Home | dRofus," 2019)or MS Excel, and extracted when needed. Furthermore, necessary parameters that fit IFC schema can be exported from Revit to IFC standard MVD such as IFC2x3 Coordination View 2.0 (Building SMART, 2018), for further analysis. Therefore, our approach does not eliminate IFC interoperability, but rather enhances the proprietary information advancement, prior to export.

# 2. METHODS AND PROCESSES

The case study is based on Action Research Methodology (Bryman & Bell, 2011). Aalborg University is developing a thermal assessment tool called IV20 (REBUS, 2015), which aims to holistically assess a range of indoor environmentally related standards regarding thermal, visual, atmospheric and acoustic comfort. In order to reduce the time it takes to manually input energy performance data, two test groups were given a task to investigate how can Dynamo enable parameter value calculation and automate information transfers between a BIM model and Excel IV20. The Exchange Requirements were defined by the input parameters of IV20 and prototyping and testing was investigated by the two test groups. We compared the accuracy of the answers to manual methods and attempted to automate the calculations needed for parameter transfer using Dynamo VPL. We wrote several scripts to compare how they perform on varied geometry conditions. As the two groups worked on the same Exchange Requirements, at least two methods were developed to calculate the same parameters. The methods were then compared, and the results and observations are described in this article.

The essential objective of Revit-Dynamo-Excel (RDE) workflow is to reduce transfer process complexity, target specific parameters, and establishing data extraction and calculation methods. The 6 rules below describe processes necessary, prior to creating the Dynamo scripts:

1. Parameter domains and rules need be defined in the Exchange Requirements.
2. Calculation methods need be explicit and delimited to their application, i.e. calculations for straight walls will differ from calculations for slanted walls.
3. The Revit model geometry must be modelled to the agreed level of detail.
4. The Revit model must be specified to an appropriate level of information.
5. The Revit model must be verified and validated prior to information takeout.
6. All models sharing the same phase and geometric properties of the first concept must follow the same naming conventions and rules for the scripts to be effective on other projects.

The success of energy calculation greatly relies on the calculation method and preliminary data derived from the Revit model objects. The methodology assumes that the model contains architectural objects needed for initial energy calculation and that the modelling method is following a structured workflow.

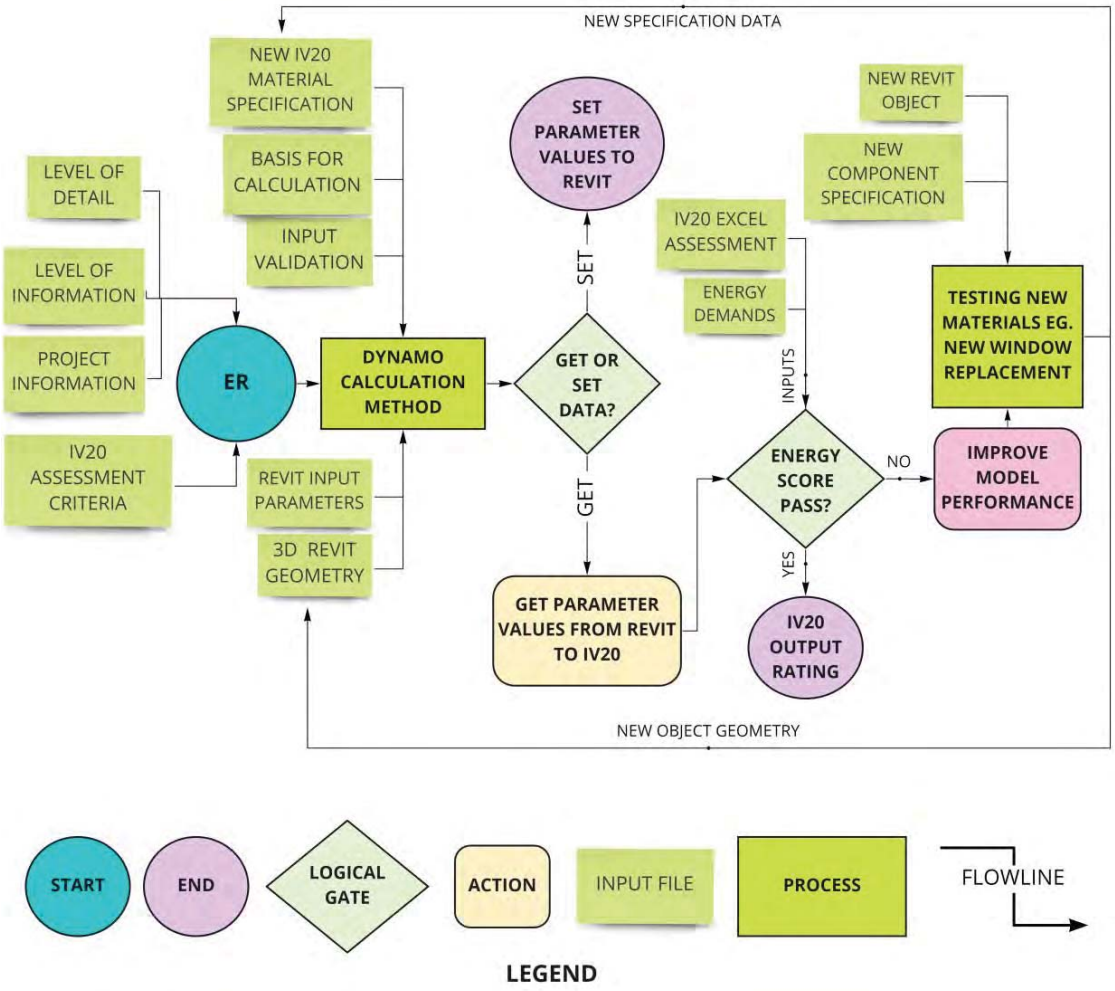


*Figure 1 Process diagram of RDE workflow.*

# 3. PROTOTYPES FOR PRACTICAL APPLICATIONS

In this section, we will look more closely into Dynamo VPL as a calculation and data transfer tool. We will attempt to describe calculation methods and processes so that the reader can learn techniques we developed. When deriving parameter values, considerations need to be noted down, to allow future scripts users to be aware of script applications and limitations.

Just like when analysing an object visually, a computer must understand what the object representation means to a certain script. For example, it is simple for a person to understand the difference between an internal and an external wall, provided there is a window with a view out. A computer can assess whether the wall is external, only if the wall property contains a parameter that specifies its function - to *exterior*. All walls in Revit of the same type will have a specified function, meaning their function is a *type* property. It is therefore detrimental, the Wall type is set correctly so that Dynamo can reason, whether the wall is *exterior* or *interior*.

It is a good idea to create a set of steps for each script, guiding its logic and chronology of creation. Below is a sample chronological workflow, paving a way for wall area script determination.

1. Exporting wall and room information from the Revit model.
2. Measuring the lengths of wall segments for each room.

3. Mapping segment lengths to specific wall instances.
4. Summing wall instances and multiplying by room height to obtain area.
5. Excluding window and door areas from walls.
6. Listing results for each room instance.
7. Converting measurement units and rounding to agreed decimal points.
8. Translating results to a tabular form.
9. Exporting results to IV20.

## 3.1. GROUPING BY ROOM OBJECT ID

Although Revit Identifier (ID) is not the same as IFC Globally Unique Identifier (GUID), as its value can be altered when updating components, its use is justified when identifying components during live operations in Revit and Dynamo. It becomes especially useful, when Dynamo highlights the code in green, meaning the information transferred is a function, not a label (string value). It is, therefore, a good practice to keep track of element ID and storing it visibly within Revit object parameters throughout the period of operation. It can serve as a dynamic reference point, which updates along with holistic model updates, yet remains static, when element count, and file version remains constant. A node called *Element.Id* can extract the ID and append its string value to the desired parameter name. Lastly, selecting the element can lead a user directly to the location of the element within Revit, having a similar functionality of the BIM Collaboration Format (BCF) element selection zoom function.

This information can be later retrieved by using *Element.GetParameterValueByName* node. Other information can be retrieved, such as *Name, Area* and, *Volume*, given the parameter exists within the Room category. Once parameters are generated, the lists can be transposed to match tabular entry sorted by the desired filtering element. In this case by room name. This process will allow grouping elements by room, enabling selection of elements only relative to their spatial placement. *List.Transpose node* swaps or transposes columns with rows, thereby changing a list of i.e. room names, areas, and volumes, to a list of room names and their corresponding parameters i.e. areas and volumes.

Using the same principle, we assign room names to windows, in order to recognize the window location filtered by belonging room, and in the same way, we can assign other parameters to the window as described below. This way, we can create a hierarchy tree where the building is a container of rooms, and a room is a container of elements such as windows and doors. Further down, windows are containers of properties within windows. This way the hierarchy classification can follow the intended IV20 schema, and data can be published accordingly to desired work sheets.

A *Data.ExportExcel* node is used to send tabular data to Excel enabling the data to be stored natively in Excel Spreadsheet (.XLS), or if needed, in an interoperable format - Comma Separated Values (.CSV). The node takes inputs for Excel file path, sheet name, starting row and columns, the data desired for export, and a *Boolean* to override current data content. It has proven useful to detach the file path with the export node, as to prevent accidental data updates when running Dynamo scripts. Consequently, data override will wipe the entire sheet blanc and should be used with intent.

## 3.2. WALL SURFACE AREAS

It is ambiguous, whether the calculation principle considers discrepancies such as wall openings, ceiling height, and internal boundary condition offset. Wall surface areas can be misleading when directly taken from the Revit model. Using the *Room.Boundaries* node, we derive the *Curve.Length* calculation of boundary room edge length. This allowed us to retrieve wall lengths to measure internal surface wall areas – directly from automatically generated rooms within Revit. We then used the ceiling height parameter to derive wall heights up to the ceiling. Alternatively, if no ceilings are present, room height can be constrained to a different parameter, agreed in the IDM.

This flexible approach demands user attention, and therefore is semi-automatic, as the operator must know which parameter is responsible for limiting room height. Similarly, depending on use, a room boundary line can skew resulting perimeter length. There is a discrepancy in length between the inner, centre and outer wall perimeter. A formula for determining offset can be added using *codeblocks*.

*Codeblocks* are python based, console coding blocks, especially useful for mathematical formulas, reducing complexity and node granularity. A *codeblock* can merge an offset parameter relative to the wall centreline, by applying a correction to the perimeter length calculation.

Once the overall surface area is calculated, the following step involves excluding areas occupied by openings, such as doors and windows. There are a couple of methods to exclude those areas, and this process can be automated if the opening elements' properties, (i.e. location) can be segmented by Revit rooms. A node called *Room.IsInsideRoom* can be used to check if certain elements are a part of a room. This can, however, fail, if the elements are recessed into walls, and the room objects consider only elements within the room boundary lines. Again, there are multiple ways of addressing this fix, and the solution may vary depending on the method.

Using *Element.GetLocation* node and *FamilyInstance.FacingOrientation*, we created two intersecting points for each door and window element with a proximity offset of +/-300mm, perpendicular to the room boundary. This method has taken all intersecting door and window elements, which had a facing orientation parallel to the room boundary, placed within the *300mm* range. This ensured that regardless of the boundary location line, items in proximity were considered to the count. This enables a *Boolean* output from the aforementioned node called *Room.IsInside.Room,* which marks objects belonging to a room as *true*. The flattened list was then filtered by *List.FilterByBoolMask* to filter true values only.

Lastly, the *area parameter* was obtained from intersecting doors and windows and the value was subtracted from the total surface area of the walls within the room. It is important to observe here, that if the room did not contain any windows, a *null* value would produce a *null* result if the list was not cleaned by using *List.Clean* node and preserving indices set to *Boolean* "*false*". This ensured that any *null* values will be removed from the list, thus preventing strings to mix with numerical formulas.

## 3.3. WINDOW ORIENTATION TO TRUE NORTH

Window orientation is essential for energy analysis, as the information can identify the exposure to sun and external objects, as well as identify window location and the angle to true north. To populate orientation angles to external doors and windows, firstly, we select only exterior doors and windows. If curtain wall panels are present in the model the same principle applies to those elements. To select exterior elements, we can filter elements by "*Function*" parameter within *Element.ElementType* node. This is important when accessing element type properties, as the function is defined at a type, and not at an instance level. If we did not use the *Element.ElementType* node, only instance parameters list will show. Exterior elements function value is equal to *1*, as it is the second element on the list i.e. *0=Interior*, *1=Exterior*, mapping numbers to function displayed in Revit drop-down Element Type Function list. By default, in Python language, thus in Dynamo VPL, the first element of every list is indexed as *zero*.

*List.Join* is then used to join filtered lists of exterior curtain walls, exterior doors, and windows. To get the element orientation we used the *FamilyInstance.FacingOrientation* node and a custom node from Clockworks package called *Wall.Orientation*. Packages are sets of nodes created by the Dynamo community to enhance Dynamo off-the-shelf capability. Note that *Wall.Orientation* node checks orientations for walls only, but the doors and windows are elements nested within walls, and thereby their orientation is transferred as a result. The resulting list of walls and wall-nested family instances are joined by *List.Join* again.

The next step is to link Revit project rotation to define the angle to true north. We used the *Vector.Rotate* node, which took the resulting List.Join and used its vectors as its first input. The second input was the axis of rotation – the Z-Axis, obtained from the *Vector.ZAxis* node, and the third input were the project coordinates, obtained from the *Coordinates.ProjectRotation* node. In order to define true north, the project rotation needs to be negative. A simple *codeblock* subtracting Project Rotation from a value of *0* will suffice.

Our final input is provided in degrees and the *Vector.Rotate* node gives out a vector which can now alter vector angle about the axis. Using the *Vector.AngleAboutAxis* node, the *Vector.Rotate* is coupled with the second input of reverse Y-Axis vector, and a rotation axis around Z-Axis Vector. The resulting angles will give values ranging from *0* to *360 degrees*. It is a requirement that the value instead uses a scale from *-180* to positive *180 degrees*, thus a *ReplaceByCondition* node can append the result

only for values above *180*, replacing the item with a formula which subtracts *360* from the original value. To validate true north rotation, it is important that the project location is set correctly to reflect real-world coordinates or that the site coordinates are published to the project from the Revit site link.

## 3.4. ROOM OVERSHADOWING

To check if overshadowing objects affect a room sun exposure, we calculate the parallel and perpendicular distances to adjacent buildings which limit sun exposure by providing desirable or undesirable shadow effects. The IV20 exchange requirement is expressed as *distance to shading object*. Given the proximate environment is geometrically modelled, using Dynamo, we select the closest face of the building and calculate the distance using the *Geometry.DistanceTo* node.

This action will require an operator to manually select faces, which may be very inefficient in projects containing multiple rooms. This process can be expanded by attributing faces to windows automatically, by mapping them to windows and mapping windows to rooms, as explained in section 3.1. This procedure would allow selecting a room, letting the programme know which window within the room is picked, and which surface represents the unobscured glass face of the window. Equally, by measuring the distance to multiple objects, it is possible to filter the shortest distance and append this measurement to the window face at hand. A loop can be created to check for all windows within a room, and all rooms within a project, and automatically derive distance measurements from each shading object. On top of distance to shading object, the IV20 required parameters specifying the height and inclination angle to of which, the building will cover the sun path, relative to its distance. We did not attempt to calculate these parameters due to time constraints, yet we suppose the calculation will follow similar principles to the above.

## 3.5. ROOM SURFACE MATERIALS

It is also possible to reverse the flow of information – and instead of populating IV20, materials can be specified from within IV20, and published to Revit. This is useful when communicating analysis results back to the design team. In order to calculate energy loss of the building, the materials of room closing elements – walls, ceilings, and floors must be recorded. In IV20, there are predefined nine types of rooms, six floor, eighteen wall, and fourteen ceiling materials, that can be mapped to Revit objects using MS Excel.

Plugging the *File path* to the IV20 spreadsheet enables reading data from Excel. We choose the sheet we want to read, and we mark *Read As Strings* with a *Boolean* set to true. We then transpose the list to obtain parameter values grouped by *Room Name* attribute. *List.RemoveFirstItem* can be used to remove list headers. We then choose the list index to match the list needed for floor finishes, wall finishes and ceiling finishes to match them to the Revit default parameters.

We then pick Revit rooms and match them with the IV20 Excel list. We make sure that the room list in Excel is matching the Room list in Revit, by using Room Element ID number. This ensures there are no two rooms having the same name. Lastly, we run the script and populate material information to Revit to set material data used for specific energy calculation in the form of a string parameter. Then the results can be observed inside of Revit schedules with the material information populated into each of the rooms. This process has a similar application to the BIM Collaboration Format used with IFC. Now that the information is populated, it is up to the design team to approve material changes.

# 4. OBSERVATIONS

In this study, we attempted to bridge some geometry data with a spreadsheet-based energy assessment tool, enabling a greater degree of automation of indoor environment assessment analysis. Dynamo scripting was used as the linking tool and observations had been made on how elements can be validated, calculated and translated to a spreadsheet. Not all parameters necessary for IV20 input were calculated, but each new script was a step closer towards more comprehensive automated data transfer.

The main purpose of this workflow was to develop a missing link between Revit model geometry and an MS Excel spreadsheet. The transfer will allow the IV20 tool to further analyze results and use the data to evaluate the indoor environment. As each calculation requires a specific approach, before all scripts are created and validated, one cannot fully rely on automating a specific task, as it may be based on incorrect quantities or methods. However, once the methods are validated, a significant improvement in quality and validity of data can be achieved.

Interpretation of models needs to be carefully analysed to yield appropriate and equivocal results. It is of major importance to agree on the level of detail and component specification, to ease, or at least bring to awareness that some object properties may be inaccurate. For example, area and volume values may be inaccurate due to built-in Revit calculation rules, placing boundary line in a different location than anticipated.

There are multiple ways of performing tasks within Dynamo, and more than one way can be correct. Explicit calculation method forms a foundation of rigid script development. Further script improvements reduce complexity and improve the transparency of the code, whilst keeping the clarity of formulas and calculations. A higher level of automation is accompanied at the trade-off of transparency, which is especially dangerous if the end-user was not involved in the creation of the scripts. Therefore, we advise that the in-house knowledge of Dynamo is shared across all involved stakeholders through commonly available standards and methodologies.

A good practice when attempting to design a new visual code script, is to create a logic map, describing issues that will be encountered during the calculation. A logic map can be communicated with the model provider using conceptual diagrams using UML or IDEF0, which clarify information flow, calculation principles and presumptions.

Although Dynamo requires some degree of programming knowledge, and a good understanding of the functionality of Autodesk Revit, thanks to the transparency of VPL interface and prompt response of Dynamo forum users, we found using Dynamo rewarding, and our understanding of the software had grown considerably in a short period of time. The community is active and keen on sharing ideas and is gladly offering help to solve problems. We discovered it a good idea to search for related issues before posting our own issues, and we observe that most computations had been encountered before, thereby enabling a short problem lifespan.

Dynamo scripts can be simplified as the user experience grows. Highly granular nodes can be compressed to blocks and calculation can be optimized by creating code-blocks. This can be done using custom nodes functionality, or by optimizing calculation techniques. For example, a curve length can be calculated as a distance between the start and end point, or a *Curve.Length* node can achieve the same, or even a more accurate result, as it will consider all curve points, allowing more widespread application, enabling calculating the length of curved elements.

# 5. CONCLUSION

We conclude that automation of data transfer is possible and achievable for industry professionals, using Dynamo as a data transfer link. The key advantage of using RDE workflow is relative simplicity, logarithmic learning curve and an in-house, software cost-reducing, development process. Additionally, unlike IFC information modelling, the data structure can be user-defined and calculated prior to IFC export. The data can be stored in use case specific parameters, that can be agreed upon using standard Exchange Requirements such as IFC IDM or EIR.

The Revit model geometry and parameters can be directly linked with MS Excel spreadsheet and the links can be sustained if the Revit models are modelled to the same level of detail, and following the same rules and parameter naming. However, to achieve a high degree of automation on a multitude of projects, thorough modelling and calculation standards need to be implemented across project stakeholders.

Knowledge transfer and transparency is key to script re-use and applicability in other projects. An in-house validation method needs to be developed, to ensure that Revit models are ready for reliable automatic data transfer when using Dynamo Revit. When creating Dynamo scripts, clear workflows must be in place, and parametric limitations must be clearly specified. Script operators must be aware

of script limitations and principles under which the scripts perform calculations, prior to drawing assumptions that the quantities or information content is coherent. Although our attempts yielded accurate results, and we succeeded in deriving some key parameters, we did not attempt to link and calculate every IV20 parameter needed for a complete thermal, visual, acoustic and lighting assessment.

Not all data transfers were fully automatic. In some early prototypes, an operator needed to switch the room in question to publish the results or select an object face for analysis. As our experience grew, we managed to automate these processes, at the same time increasing script functionality. Therefore, it is difficult to estimate if each data transfer can be fully automated by an average Dynamo user, but we can safely say that persistent Dynamo programmers will be likely capable of programming any parameter derivation. The status quo of IV20 will require all data to be manually entered and calculated from the Revit model. Therefore, even the process of automating a part of the subset, can reduce the time it takes to generate results, whilst reducing user input errors. If the requirements are met, and the method is validated, the data transfer can also contribute to the accuracy of the calculation results.

## ACKNOWLEDGMENTS

The authors would like to acknowledge the predicate for this article, which was developed by the students of Building Informatics at Aalborg University, and the authors would like to thank the contributors for their input.

## REFERENCES

Andriamamonjy, A., Saelens, D., & Klein, R. (2018). An automated IFC-based workflow for building energy performance simulation with Modelica. *Automation in Construction*, *91*, 166–181. https://doi.org/10.1016/j.autcon.2018.03.019

Bazjanac, V. (2008). IFC BIM-Based Methodology for Semi-Automated Building Energy Performance Simulation. *CIB-W78 25th International Conference on Information Technology in Construction, Santiago, Chile, July 15-17, 2008*. Retrieved from https://digital.library.unt.edu/ark:/67531/metadc897563/

Borrmann, A., König, M., Koch, C., & Beetz, J. (Eds.). (2018). *Building Information Modeling*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-92862-3

Bryman, A., & Bell, E. (2011). *Business research methods*. Oxford University Press. Retrieved from https://books.google.dk/books?id=YnCcAQAAQBAJ&printsec=frontcover&dq=isbn:9780199583409&hl=en&as_brr=3&cd=1&source=gbs_api#v=onepage&q&f=false

Building SMART. (2018). Coordination View Version 2.0 Summary — Welcome to buildingSMART-Tech.org. Retrieved May 9, 2019, from http://www.buildingsmart-tech.org/specifications/ifc-view-definition/coordination-view-v2.0

CIC. (2013). Building Information Model (Bim) Protocol. *Construction Industry Council*, 15.

Eastman, C. (2016). *BIM Handbook BIM Handbook Rafael Sacks*.

Gram-, K. (2010). 11 . Housing in a sustainable consumption perspective, 178–191. https://doi.org/https://doi.org/10.4337/9781783471270.00021

Habibi, S. (2017). The promise of BIM for improving building performance. *Energy and Buildings*, *153*, 525–548. https://doi.org/10.1016/j.enbuild.2017.08.009

Harrestrup, M., & Svendsen, S. (2015). Changes in heat load profile of typical Danish multi-storey buildings when energy-renovated and supplied with low-temperature district heating. *International Journal of Sustainable Energy*, *34*(3–4), 232–247. https://doi.org/10.1080/14786451.2013.848863

Home | dRofus. (2019). Retrieved April 24, 2019, from https://www.drofus.no/en/

ISO. (2018). ISO 16739-1:2018 Preview Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries -- Part 1: Data schema. Retrieved April 15, 2019, from https://www.iso.org/standard/70303.html

Kamel, E., & Memari, A. M. (2019, January). Review of BIM's application in energy simulation: Tools, issues, and solutions. *Automation in Construction*. https://doi.org/10.1016/j.autcon.2018.11.008

Kassem, M., Kelly, G., Dawood, N., Serginson, M., & Lockley, S. (2015). BIM in facilities management applications: A case study of a large university complex. *Built Environment Project and Asset Management*, *5*(3), 261–277. https://doi.org/10.1108/BEPAM-02-2014-0011

Pinheiro, S., Wimmer, R., O'Donnell, J., Muhic, S., Bazjanac, V., Maile, T., … van Treeck, C. (2018). MVD based information exchange between BIM and building energy performance simulation. *Automation in Construction*, *90*, 91–103. https://doi.org/10.1016/j.autcon.2018.02.009

REBUS. (2015). REBUS - REnovating BUildings Sustainably, 1–133. Retrieved from http://rebus.nu/

What is Dynamo? | The Dynamo Primer. (2018). Retrieved May 8, 2019, from https://primer.dynamobim.org/01_Introduction/1-2_what_is_dynamo.html

Wix, J., & Karlshøj, J. (2010). Information Delivery Manual Guide to Components and Development Methods. *BuildingSMART*, 84. Retrieved from http://idm.buildingsmart.comhttp//idm.buildingsmart.comashttp://idm.buildingsmart.com