Developing an Automated Philippine National Building Code Compliance Check for R-1 Projects in BIM Using Visual Programming[1]

Karen Therese F. Lopez [1]
arch.ktlopez@gmail.com

Ronald S. Balane [2]
rsbalane1@up.edu.ph

Abstract

The construction industry is inefficient and has plenty of communication and coordination issues that can lead to an increase of 20–25 percent on project delivery costs (Allen & Shakantu, 2016). The design process is tedious and filled with revisions. The issue of code compliance amidst these changes and revisions further complicates the process. Code compliance checking can be cumbersome. This is mainly because most designers check 2D text and documents manually, which is very error-prone. Building Information Modeling or BIM is a 3D parametric-based methodology that is now being used by around a third of the construction industry in the Philippines (ASEP, 2013). It is strongly being used by other countries and has a high adoption rate worldwide (Kalfa, 2018). Programming and visual programming further enhances the capability of BIM to automate tasks and manipulate data. This can be used to create an actual automated code compliance check tool to address issues of compliance with building standards. The purpose of this study was to create an automated code compliance checker of the National Building Code of the Philippines for R-1 projects. The results for the automated code check were then compared with the results of the manual code check of the 2D documents of the projects, to evaluate if the developed automated code compliance checker was accurate, efficient, and feasible. The results showed that the percentage discrepancies between the two forms of code checks did not exceed 6 percent, most of which were from human modeling errors. Moreover, the automated code check took approximately five minutes per project compared to the manual code check that took approximately one hour. The developed automated code compliance checker is usable at its current state and it has potential for improvement in the future.

Keywords: BIM, Visual Programming, Compliance Check, Building Code, Automated.

[1] Karen Therese F. Lopez is a licensed architect who co-founded Arkalo, an architectural firm. She took her Master of Architecture degree in the UP Diliman, where she is currently a faculty member. Ms. Lopez's expertise is in BIM and other architectural software.

[2] Ronald S. Balane has an MA in Urban and Regional Planning and a BS in Architecture from the UP Diliman. He is currently an Assistant Professor and is a registered and licensed Architect. His research interests include new building materials and use of computers in architecture.

I. Introduction

The construction industry is one of the most inefficient sectors worldwide. It is filled with problems that are caused by communication and coordination issues (Allen & Shakantu, 2016). This is also apparent in the Philippines, where delays and issues plague the sector.

Code compliance is one of the problems in the country. It is important because it ensures that a building is stable, reliable, and usable. Unfortunately, many designers and owners disregard the code because of its inconvenience both in the implementation and the tedious process of manually checking the code (Preidel & Borrmann, 2016).

Code compliance checking is the process of checking if a building design conforms with the standards and codes that the building is subjected to. Designers and inspectors usually do this by manually checking 2D technical and textual documents and comparing them with the numerous standards and codes written in text (Preidel & Borrmann, 2016). Many codes can be subject to misinterpretation. In addition, there are plenty of codes that conflict with one another, making the process not only tedious but cumbersome and error-prone (Kim et al., 2011).

A survey by the International Code Council and National Association of Home Builders found that more residential buildings have code violations than commercial buildings. (International Code Council and National Association of Home Builders, 2013). Of note, most approved building

[1] This article is a summary of a graduate thesis for Master of Architecture completed in June 2020, A.Y. 2019–2020 under faculty adviser, Prof. Ronald S. Balane from the Building Science Studio Lab.

permits in the Philippines are that of residential projects (Philippine Statistics Authority, 2018).

Building Information Modeling or BIM is a 3D based methodology of producing Architectural, Engineering, and other Construction professional's drawing and output (Matejka & Tomek, 2016). There are several software programs available that are considered BIM tools like Autodesk Revit and Graphisoft ArchiCAD. Architecture, Engineering, and Construction or AEC professionals are already utilizing BIM for their construction projects. Some claim BIM as the solution for the construction industry's woes since it eases the process of the workflow and improves communication and visualization in projects (Allen & Shakantu, 2016). BIM makes data readily available, which helps in the coordination of design and construction issues as well as in the actual work. This is because it allows the use of standard components, which saves time and makes production more efficient (Dowsett & Harty, n.d.; Kuehmeier, 2008; Reizgevi˘cius et al., 2018). However, available BIM software also has numerous limitations which can be bypassed by support software or add-ins, like visual programming.

Visual programming is a process wherein a user may develop programs with tools, buttons, or nodes on screen that may be connected like a flow chart, which displays logical paths and associated code blocks. It functions the same way as a text-based programming or coding but is easier to understand and is more suitable for designers, architects, and engineers. Visual programming offers a lot of opportunities for the AEC. It can automate repetitive tasks, maneuver data, give access to parametric design and more complex 3D, and can even test performances (Santos, 2015; Anderle & Allen, 2017).

With the available technological tools in the AEC industry like BIM and visual programming, there is a possibility of having an actual automated code compliance checker that is convenient, flexible, and feasible.

II. Methodology

Ample research was done to study how to use visual programming in BIM. Studying the National Building Code of the Philippines was also necessary to be able to create an automation that is accurate.

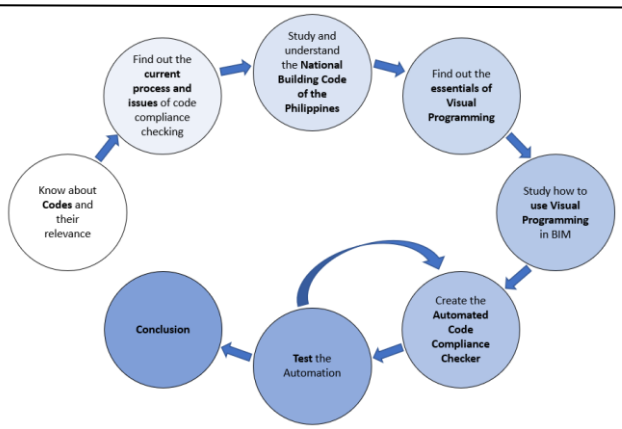

Figure 1. General Methodology.

The types of codes from the National Building Code for R-1 projects were compiled and organized. The codes were separated per element in accordance with the elements available in Revit.

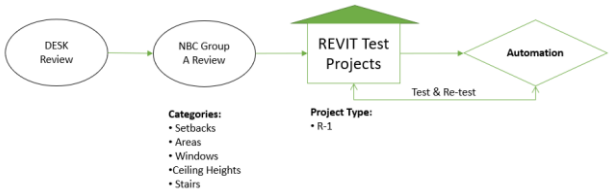

Figure 2. Categories for Automation.

The automation was filled with modeling and experimental studies because of the necessity of creating multiple BIM models for the testing and creation of the automated code checker.

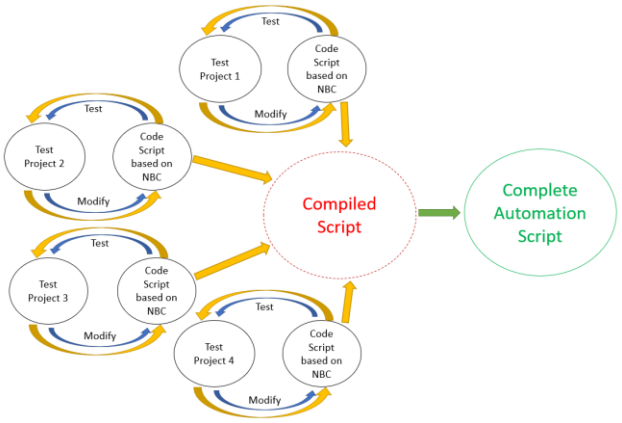

Figure 3. Automation Methodology.

Five actual approved R-1 projects were gathered to be used for the testing of the research. A reliable practicing senior design consultant was asked to manually check the 2D documents of those five projects. While the 3D model of those five projects were used to test the developed automated code compliance checker. The results for the automated code check were then compared with the results of the manual code check.

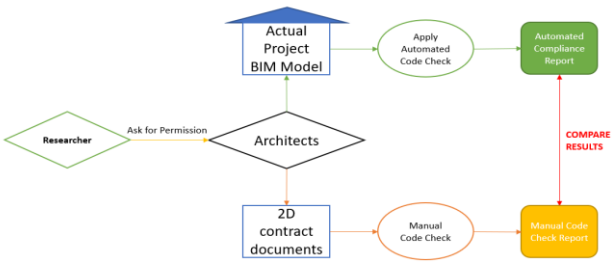

Figure 4. Testing Methodology.

This final test was used to see if the automation was useful, convenient, accurate, and flexible.

III. Gathered Data

A. Codes, Compliance, Permitting, and Revisions

Codes are a necessity because it ensures the safety and functionality of a structure. This is especially important for disaster prone countries since many damages on structures occur because of non-compliance to the code (Cox et al., 2006).

Given the changing times, construction practices, and requirements, codes should be frequently changing. However, most developing countries neglect the code leading to outdated rules and regulations.

Codes are also often confusing and conflict with other codes which could lead to misinterpretations that lead to rework and double handling. The National Building Code of the Philippines and other codes in the country like the Fire Code are not exempted from this issue. Codes should not only be harmonized and consistent with one another, they should also be simplified and easy to understand.

Compliance to the code is legally required. Unfortunately, in the Philippines, compliance is rather low because of the lack of budget and overworked building officials and inspectors. This is also the reason for the lack of code enforcement in the country. Plenty of times, compliance becomes reactive rather than proactive (Cuntapay, 2009).

To check compliance, most designers manually check their drawings and cross check those with all the codes their projects are subjected to. Some would opt to check the code before and after planning. It is also recommended that the designers know the code well so that they will avoid design choices that could go against the code requirement in the process of designing or planning their projects. Designers should also coordinate with the local government unit to make sure they abide by all local laws and special requirements.

The process of permitting in the Philippines is quite simple but it can become complicated depending on the building official or the handler. Some local government units also require different things so requirements can differ per city. Moreover, corruption is known to plague the process. Ideally, codes must be updated and coordinated with other approving bodies to negate confusion and to have consistency.

Construction projects with approved building permits in the country are mostly residential buildings. Residential buildings also have more code violations than commercial buildings (International Code Council and National Association of Home Builders, 2013).

Projects always go through revisions because of the many factors and issues in planning and design. Changes could occur not only because of code compliance but also because of other factors like value engineering or site adaptation. Most of the revisions in projects involve walls. Walls always change and even the slightest movement in walls affect other elements in a building like the room areas and ceilings. Any revision made could mean rechecking drawings to check if it is still compliant or not which can be tedious and time consuming.

B. Technology

The technological hardware and software were very essential for this study since it highly utilized computers. The study primarily used an Intel Core i7 at 3.4 GHz desktop with 16.0 GB RAM and a Windows 10 operating system. A laptop with Intel Core i5 at 1.00 GHz with 12.0 GB RAM and a Windows 10 operating system was also used occasionally. Both computer systems can create and run the automation with no problems.
BIM is a 3D based methodology of producing drawings and plans for the AEC. It has plenty of advantages because it is parametric and automated. This is likely the reason why there is a large increase in BIM usage rate worldwide. The table below shows some of the advantages and disadvantages of BIM.

Table 1. BIM Advantages and Disadvantages.

Advantages	Disadvantages
Increased efficiency	Licenses and equipment required are expensive
Reduced errors and rework	Takes time to train people and to implement
Reduced manpower	It is hard to shift to BIM especially for the older generation
Better presentation and clearer design	Return of Investment takes time
Better coordination	Dependence on user expertise

The study made use of Autodesk Revit Architecture. Revit is one of the most used BIM software as seen in the table.

Table 2. BIM Software Usage.

BIM Software	User Percentage
Revit BIM	67.08
ArchiCAD	31.69
Bentley BIM	14.79
Other	20
Tekla Structures	5.99
Digital Project	4.05
Nemetschek AllPlan	2.29

Adapted from: 'Review of BIM Software Packages Based on Assets Management,' by Kia, 2013, Retrieved from https://www.researchgate.net/publication/253058808_Review_of_Building_Information_Modeling_BIM_Software_Packages_Based_on_Assets_Management.

The version used for the study initially was Revit 2019. The research then shifted to Revit 2020. The shift was seamless with just a few edits in the automation script. There were no issues or problems that occurred.

Visual programming is a means for visual professionals like architects, designers, and engineers to write computer code even without profound knowledge on programming. Most visual programming tools are free and accessible. With visual programming, the manipulation of data within BIM to be able to apply a set of rules and be automated becomes possible. Tedious tasks like renumbering and creating sheets becomes easy and efficient to do.

The Visual Programming tool that was used for the study was Dynamo which is compatible with Revit. It is already included in Revit 2017 and above as an add-in which can be seen in Revit's ribbon panel in the Manage tab. It is free to use and can easily be downloaded on its website, dynamobim.org.

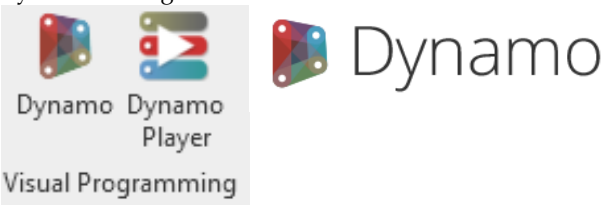

Figure 5. Dynamo.

Dynamo packages were also downloaded for the study. These are also free and can easily be downloaded in dynamopackages.com.

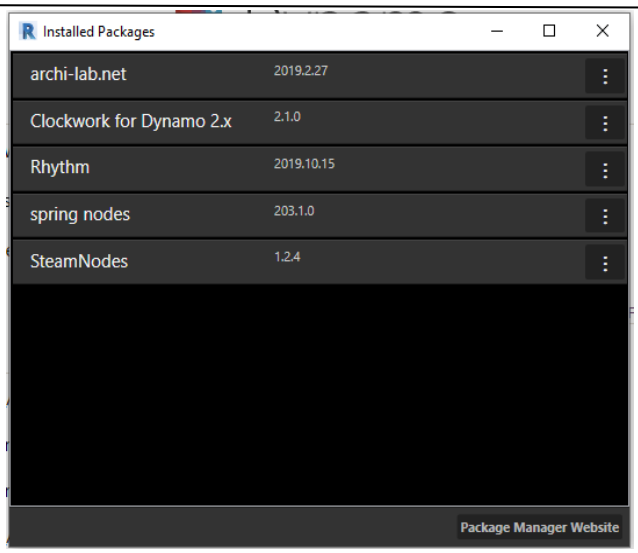

Figure 6. Dynamo Packages.

The initial version used for the study was Dynamo 2.0.1 but when the research shifted to Revit 2020, the Dynamo version was also updated to Dynamo 2.1.0. The only change that had to be done when the shift happened was the update of the selection nodes in the script.

C. R-1 Projects

The researcher gathered five actual projects that have approved building permits and are already built. All of them are R-1 single detached residential structures that are at the mid to high end range of construction. They are all quite different which is good for the automated code check test since different issues could arise from different kinds of setups. These projects were used for the testing of the automated R-1 building code compliance check.

All the necessary 2D construction documents were also provided by the five architects of the five projects. These are the documents that were approved by a building official. The specific documents that were sent were the Site Development Plan, Floor Plans, Elevations, Sections, Reflected Ceiling Plans, and the Window Schedules.

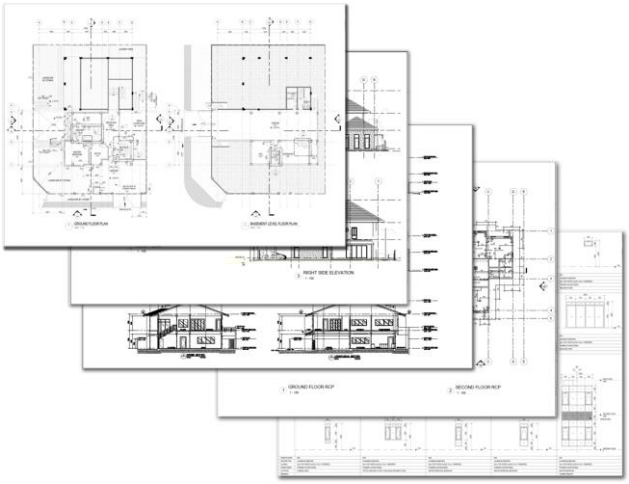

Figure 7. 2D Documents.

All of the projects were produced in BIM but only two of the five used Revit so the other three models had to be made from scratch.

Figure 8. 3D Revit BIM models of Actual Projects.

IV. Results and Analysis

Creating the automated code compliance checker involved the creation of customized parameters and plenty of data manipulation with Dynamo.

The automation of the codes was pretty complicated and involved plenty of additional parameters, gathering of parametric and automated data, and workarounds. Most of the scripts were difficult to do despite the mastery of Revit and its features.

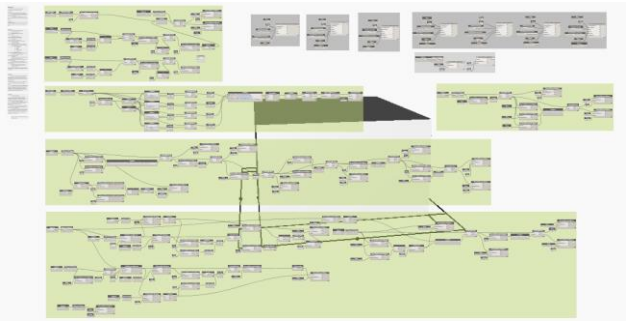

Figure 9. Automated Code Compliance Checker Script in Dynamo.

A. Automation

Parameters created in Dynamo will be in a separate script and will be run in the Revit model first before the main automation script. For some parameters that were created like the "Type" and "Natural Ventilation" parameters for the rooms, the user or the designer must manually input these parameters. After the first script which has the parameters are applied in the Revit model, the user should input the correct data in some of the parameters.

In case the designer is unavailable to input the correct data for some of the parameters that need data, the most stringent default values can be automatically placed.

Figure 10. Area Script – Room Parameter Default Values.

1. Percentage of Site Occupancy (PSO)

Automating the Percentage of Site Occupancy or PSO compliance at first seemed simple enough but was a little more complicated than expected. Luckily, Revit has a property line element with an automatically generated area.

Floor elements in Revit are used for both the structural slab, landscape, pavements, and finishes. Not all of those would

be considered as part of the built area which is bounded by the PSO. Since Revit has an automatic parameter which can be used to state if a floor is structural or not, that parameter can be used to figure out if a floor is part of the built area. Floor elements in a particular level which are structural will be considered as part of the built area.

Using the total area of the lot area generated from the automatic parameter of the property line element and the total area of the structural floor area in the first level, figuring out if a project is compliant with regards to the PSO becomes possible.

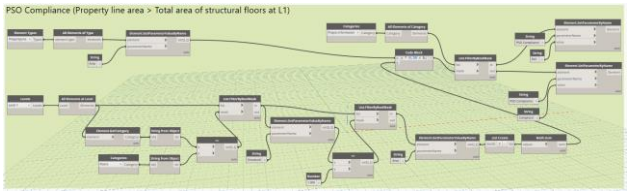

Figure 11. Dynamo – PSO Compliance Script.

2. Setbacks

Automating the setbacks were even more complicated than the PSO especially since orientation must be considered for the front, side, and rear setbacks. Revit has no innate parameter which states which side of its property line is the frontage of the lot. A complicated script which involved getting the vector value of the midpoint of each side and evaluating the location of these values had to be done to figure out which side is facing what side.

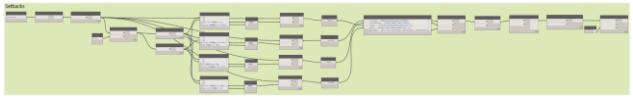

Figure 12. Dynamo – Setback Script.

3. Room Area and Window Opening Area

The room area and window area automation required several Booleans and new parameters to figure out if a room is compliant or not. The script made use of the room's automatically generated room area and a new parameter that would dictate what type of room the room was based on the types in the building code.

It also made use of a tool that could select objects within or by the room. In this case, it selected the windows of the room. From there, the automatically generated height and width of the windows were taken and multiplied with each other to get the area of the window. All the areas of the windows within the room were taken to extract the total window opening area per room. A new parameter was also created to determine if the room was naturally or artificially ventilated since the requirements within the code are different depending on its type of ventilation.

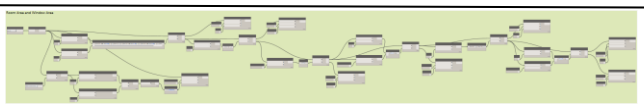

Figure 13. Dynamo – Room Area and Window Area Script.

4. Ceiling Height

In the Ceiling Height automation, the script made use of a tool that could gather elements within the room just like the one used for the window opening automation. In this case, the elements selected were the ceiling and the floor.

The basis of the result will be the highest floor and the lowest ceiling of a room. Thus, if the result is not compliant, it will not necessarily fail compliance since the highest floor and the lowest ceiling may not be in the same axis. This is the reason why the result will show "Check" instead of "Fail".

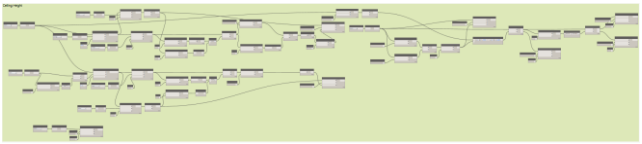

Figure 14. Dynamo – Ceiling Height Script.

5. Stairs

For the Stairs, the script will be applied on stair runs instead of the main stair since each stair run can have different risers, treads, and width. Checking the individual stair runs instead of the main stair will be more accurate. Stair Runs are nested within the main Stair family in Revit so the Compliance parameter created for all elements will not appear in the nested elements. A separate Compliance parameter for the Stair Runs had to be created. The "Actual Riser Height", "Actual Tread Depth", and "Actual Run Width" parameters of the stair runs in Revit are automatically generated. They are accurate but the Width and Tread will not have values if the Stair was created by sketch. The solution was to make stair runs with a missing value on any of the three parameters fail compliance. The user can then manually check these stairs.

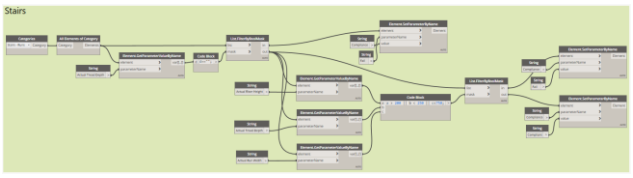

Figure 15. Dynamo – Stair Script.

B. Testing

The manual code check took some time before completion. The architect consultant checked the documents using its PDF file which is both an advantage and a disadvantage. The difficult thing with checking PDF files instead of an actual paper is that the checker was not able to simply use

a ruler or a scale to measure the drawings. The advantage however is that the drawings can easily be zoomed in and out and are much clearer than having it printed. Moreover, since the PDF files are properly scaled, the consultant was able to place or import the PDF file to a CAD program. From there the consultant was able to measure the rooms or windows digitally. This is more accurate and at times easier than measuring an actual drawing in a sheet or paper.

The average time taken for the consultant to manually check the drawings was around an hour per project. Project 2 took much longer because it had more rooms than the other projects.

Table 3. Manual Check – Time Taken Average.

	P 1	P2	P3	P4	P5	Ave
Time Taken to Check	1 hour	1 hr and 30 mins	1 hour	1 hour	1 hour	1 hr and 6 mins

For the time taken of the automated code check, it was definitely quicker. The average time taken to run it was four minutes and 23 seconds as seen in the table below.

Table 4. Automated Check - Time Taken Average.

	P1	P2	P3	P4	P5	Ave
Time Taken to Check	4 mins and 35.53 secs	5 mins and 30.43 secs	4 mins and 2.43 secs	3 mins and 58.89 secs	3 mins and 47.81 secs	4 mins and 23.02 secs

It is safe to say that the automated code check will only take around five minutes, but this will obviously vary depending on the file size of the model, the computer speed, and other variables. One factor that made it longer to run the code check in Project 2 was the fact that it had more rooms than the rest. The part where the data had to be manually inputted took at least 40 seconds longer than the rest of the projects. So basically, the more rooms and the more complicated a project is, the longer it will take to run the automated code compliance checker. However, it will be safe to assume that it will take most projects less than 10 minutes. Do take note that time taken can also increase significantly depending on the status of the model.

Some issues on both types of code checks found were due to human and modeling errors. At one instance, the 3D model was lacking some windows that were in the 2D documents. In another instance, the manual code checker forgot to include a few windows in the computation of the total window opening of a room. Some elements were also used incorrectly like the image below where the lavatory counter was made using a Floor element.

Figure 16. Project 1 – 3D Floor Error.

Despite a few issues, the results of both code checks were similar with no compliance percentages differences greater than 6 aside from Project 2 which had an outdated model.

Table 5. Manual and Automated Check – Compilation

		Project 1		Project 2		Project 3		Project 4		Project 5	
		Manual	Automated	Manual	Automated	Manual	Automated	Manual	Automated	Manual	Automated
PSO Compliance		C	C	C	C	C	C	C	C	C	C
Setback Compliance	**Front**	C	C	C	C	C	C	C	C	C	C
	Right	C	C	C	C	C	C	C	C	C	C
	Left	C	C	C	C	C	C	C	C	C	C
	Rear	C	C	C	C	C	C	C	C	C	C
BHL Compliance		C	C	F	F	F	C	F	C	C	C
Room and Opening Area Compliance %		82.35	82.35	67.57	62.16	82.61	78.26	64.00	69.23	79.17	79.17
Floor to Ceiling Compliance %		64.70	64.70	94.59	86.49	100	100	88.00	88.46	95.83	91.67
Stair Compliance %		100	100	100	100	100	100	100	100	100	100
Approximate Time Taken to Check		1 hour	4 mins and 35.53 secs	1 hour and 30 mins	5 mins and 30.43 secs	1 hour	4 mins and 2.43 secs	1 hour	3 mins and 58.89 secs	1 hour	3 mins and 47.81 secs

The automated code check was actually able to detect more code violations than the manual code check as seen below.

Table 6. Automated Check – Percent of Accuracy.

	Manual Code Check Average Compliance % (MC)	Automated Code Check Average Compliance % (AC)	Discrepancy (MC-AC)
Site	90.00	96.67	-6.67
Room and Opening Area	75.14	74.23	+0.91
Floor to Ceiling	88.62	86.26	+2.36
Stair	100	100	0

Discrepancies with positive values means that the automated code check was able to detect more code violations than the manual code check; while the negative values mean it detected less. While many of the detected violations that the automated code check found but not in the manual code check were false violations due to modeling errors. The automated code check still had some correct values that the manual code check was not able to detect. Regardless of some discrepancies, the result of both code checks is similar as seen in the graph below.

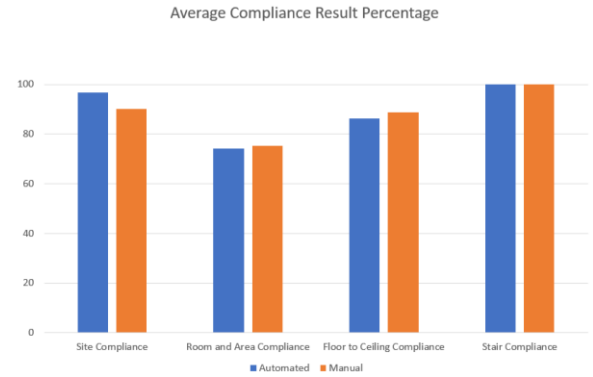

Figure 17. Code Check – Average Compliance Result Percentage.

A quick manual check of the results of the automated code check can also be accomplished to ensure that the generated result is accurate. Doing so will take approximately five to 10 minutes which will vary per person. A modified time taken which includes the quick manual check for the automated code check can be seen in the figure below.

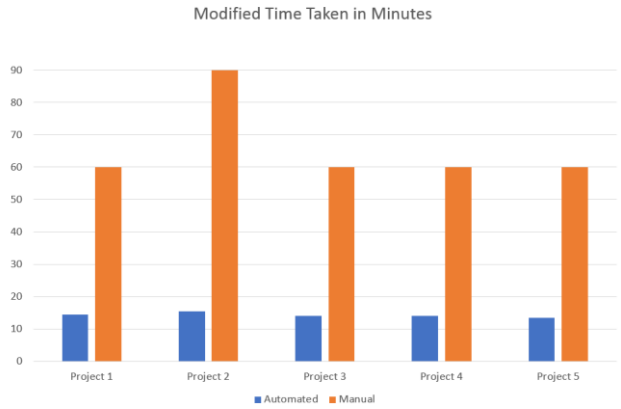

Figure 18. Code Check – Modified Time Taken in Minutes.

Even by doing that, the automated code check will still be considerably faster than the manual code check since the latter took at least an hour even with the use of excel and CAD.

C. Synthesis

A summary of the automated code compliance checker's strengths, weaknesses, opportunities, and threats are stated in the table below.

Table 7.1. Automated Code Check – SWOT Analysis.

Strengths	Weaknesses	Opportunities	Threats
Quick process.	Some setup or model cleaning may be necessary which can take time.	Faster process will mean better efficiency.	Inadequate equipment and large model sizes can drastically slow down the process.
High accuracy.	Errors may occur that will require a manual check. Results are affected by the accuracy of the model.	Greater chances of the avoidance of non-compliance in projects or designs.	Possibility of missed errors and false compliance

Table 7.2. Automated Code Check – SWOT Analysis.

Strengths	Weaknesses	Opportunities	Threats
Flexible and editable rulesets.	Difficult to edit for people with no background in Dynamo or Revit.	Has the capability of adding more rules in the script depending on the specific codes of a specific project?	Manipulation of the script could lead to a possibility of the creation of false compliance by editing the script accordingly
Uses a specific BIM software so there will be no need to convert the model.	Can only be used on Revit models.	This was created using visual programming, so it is possible to apply similar rules to other BIM software.	Updates and changes in Revit may require the constant updating of the Dynamo script.
Makes use of available and popular technology for the AEC.	Dependence on proper equipment, licenses, and experts.	Increased usage rate of BIM for the past few years makes the desire for an automated code check within BIM likely.	Shifting methods and practices can occur that may not be favorable to BIM.

The Automation of the code is convenient and useful as seen in the results especially if a designer is already using Revit. However, there are still several general key factors and possible issues that are not too obvious but still need to be emphasized and looked at.

First is the model and user variability. An incorrect manner of modeling like the use of generic masses that do not make use of Revit's innate parameters will inevitably make the automation fail in some aspects. People will do things differently and generic models or incorrect elements are sometimes being used by Revit users who are not yet adept to the software. If modeling issues are applied to any model, there will likely be problems wherein the automation script will not be able to come up with an accurate result. This issue can be solved if the user becomes aware or knowledgeable of the correct practices in modeling using BIM.

The automated code check script made use of the innate parameters within Revit whenever it could be applicable to most models. However, automatically generated parameters are not always present. There were instances wherein an automatic parameter was not present for a particular code. This meant that the automation script had to create these parameters which further complicated the script. Moreover, it added tasks for the user to do since they will have to input the fields for some created parameters as opposed to simply running the automated code check script. At this point, this is inevitable unless Autodesk adds such parameters in the future.

With regards to linked models, some models will likely have separated models that are linked with one another. This could cause some issues with the automated code check since the script may not be able to read all the elements of a linked model. Also, linked models may have different levels, families, elements, and the like which could also cause issues and duplicates. A workaround for this could be by exploding the links in a single file, but that will not solve the duplicate problem so a model cleaning may be necessary before running the automation. Luckily, Revit has a feature wherein, if the family name is the same, a copied family from a different Revit project will have the option to use the features of the family within the local Revit file as opposed to creating a duplicate of that same family. It is even possible to replace the one in the local Revit file with the family being copied.

While it is ideal to be able to have an automation script using default settings and default parameters, in actuality that will not be the case. In the automation script, some nodes that were used were from Dynamo packages like clockwork, archilab, springnodes, and more. These had to be downloaded since they were not included in the default nodes within Dynamo. The good thing with Dynamo is that all those packages are open source and easily accessible. The Dynamo community is also very helpful and answers most questions in its forum.

Another issue that could not be avoided are issues within Revit itself. While doing the setback automation, a Revit API issue wherein floors, hatches, and the like which has a sketch as boundaries, cannot be generated with a hole in Dynamo. The workaround is to create an opening on the floor to create the hole but that will not work for hatches. It is also plausible to manually create the hole in Revit and not in Dynamo. Although this specific issue did not conflict with what was needed for the R-1 codes, technical issues could be a problem for future automated code checks.

Given the following concerns that were mentioned about the automated code compliance checker, a clear guide that is easy to understand was created. The guide will allow any user that is not the researcher to be able to easily use the automated code compliance checker. The guide was created in accordance with the possible issues that may occur. Appropriate images were also added to better guide the user.

The guide has four parts: Before You Get Started, Instructions, Changes, and Disclaimer as seen in the images below.

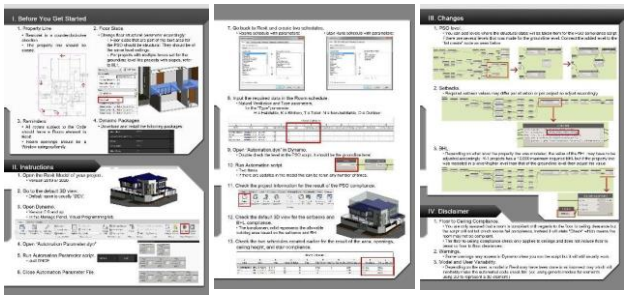

Figure 19. Guide for the Automated Code Check.

In the first part, the guide briefly discusses the things that must be done with any Revit model before starting the automated code compliance checker. It specifically discusses the property line issue and its quick recreation to solve its issue. The first part of the guide also discusses that the model's floor slab structural parameter should also be checked for the PSO compliance.

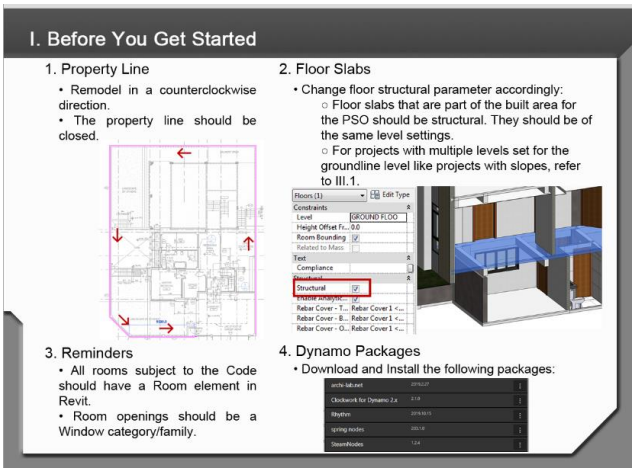

Figure 20. Guide – Before You Get Started.

The second part provides the instructions on how to use the automated code compliance checker. From opening the model to opening the Dynamo add in and scripts to running the scripts. All the required instructions will be covered.

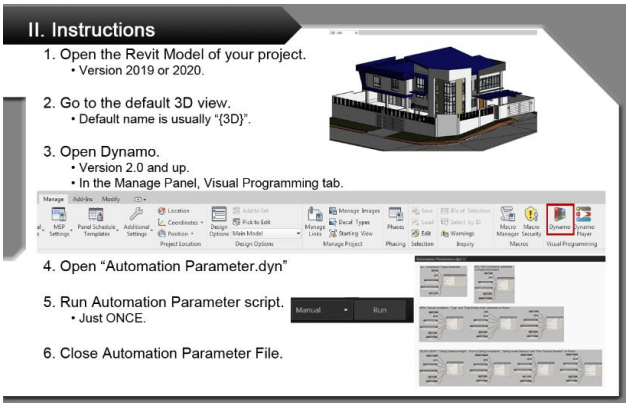

Figure 21. Guide - Instructions 1.

The schedule that must be created in Revit and the parameters that the user will have to manually fill in are all clearly discussed in the guide so that the user will not miss anything.

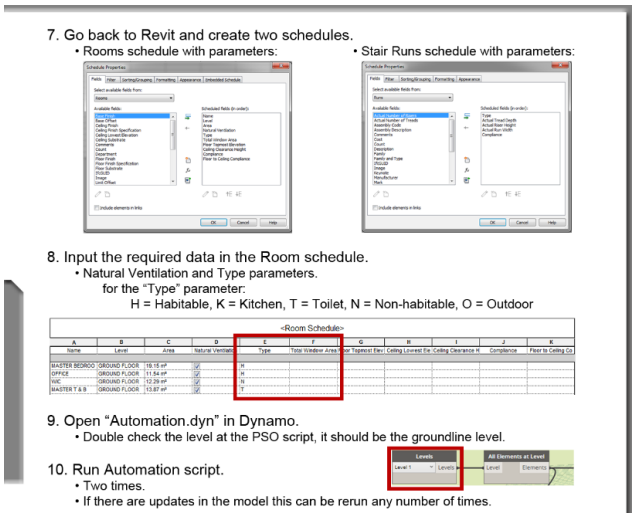

Figure 22. Guide – Instructions 2.

After the scripts are run, there are also instructions on how to check the results of the compliance report as seen in the image below.

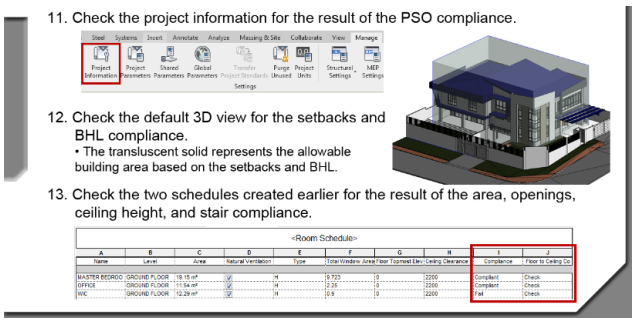

Figure 23. Guide – Instructions 3.

The third part discusses the changes that can be made depending on the particular project's situation and if there are local laws or other codes that have different values. Particularly, it discusses how to add levels for the PSO compliance, how to adjust the setback values especially for projects with multiple frontages, and the need to adjust the BHL value depending on where the property line was modeled.

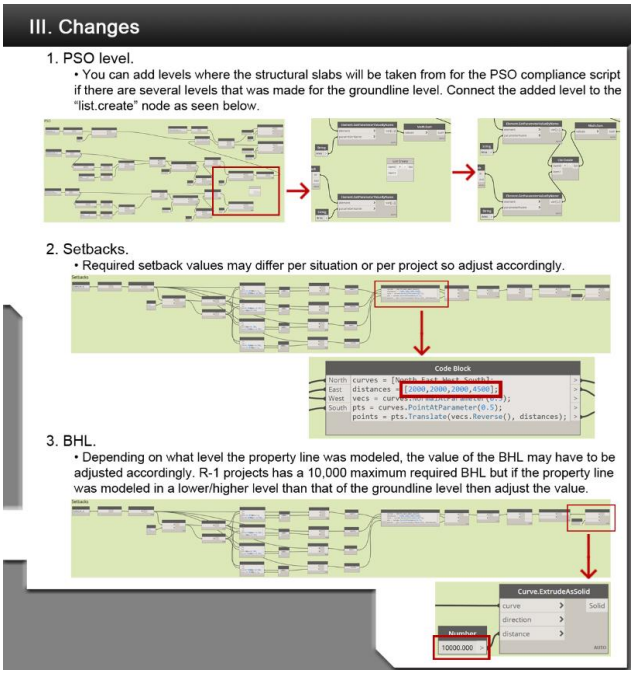

Figure 24. Guide – Changes.

For the last part, it discusses all the misconceptions that could arise with the automated code compliance checker. The floor to ceiling compliance issue is discussed and it also discusses the warnings that may appear.

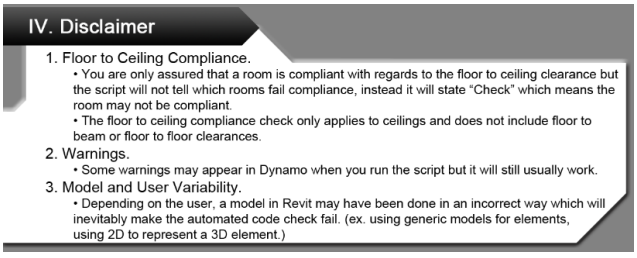

Figure 25. Guide – Disclaimer.

Notes within the Dynamo script were also placed to show the same instructions found in the guide as seen below.

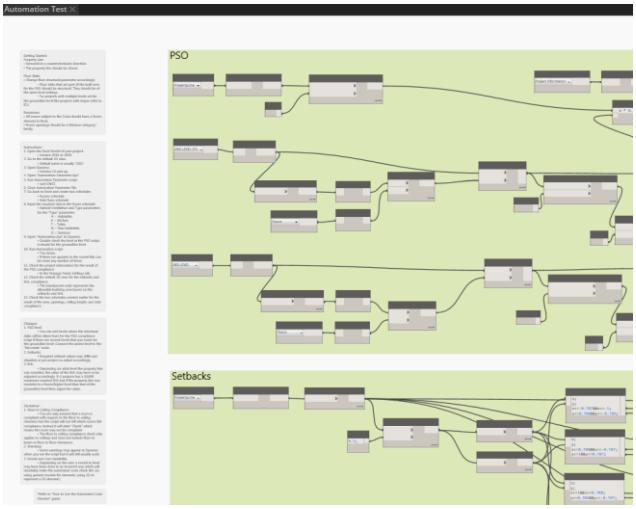

Figure 26. Dynamo – Notes.

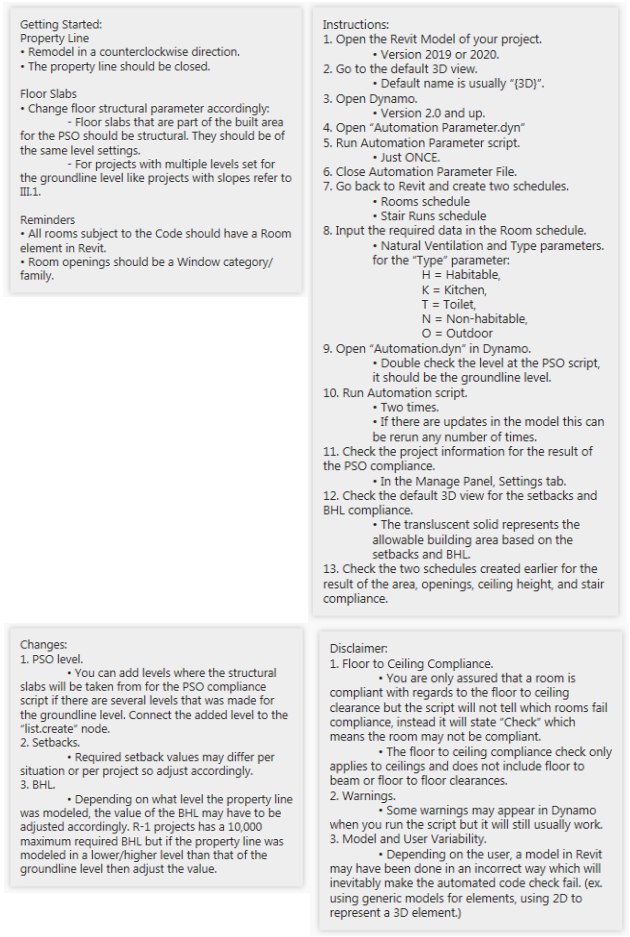

Figure 27. Dynamo – Notes: Details.

This guide will basically make sure that any user will be able to use the automated code compliance checker so that concerns that were found will be less of a problem.

V. Conclusions and Recommendations

The testing of the automated code compliance checker had desirable results which makes the development attempt successful. Result percentage variations between the two forms of code checks did not exceed six percent. Furthermore, the automated code check took only around five minutes per project or around 15 minutes including the quick manual check. Compared to the manual code check that took around an hour, the automated code check is considerably faster. There were some concerns that still need to be addressed but many of those concerns can be solved or have workarounds. The automated code compliance checker is already usable at its current state and it can still be expanded and improved on in the future.

The actual creation of the automated code check script takes up a lot of time and determining which nodes to use is more complicated than it seems. A mastery of Revit is necessary for someone to be able to properly use Dynamo for Revit especially for complicated rulesets like building codes. The solution for the study was to create a guide on how to use or edit the script especially if it has to be used for other or special projects. Since the script was already created, most changes would be on the values like setbacks, the percentages, or different clearances which have actual numbers within the script that are editable. This makes the automated code checker quite flexible and it can accommodate special codes or specific local laws that a particular project may have. The flexibility and accessibility of the developed automated code checker is very important since this is what the other available automated code checker lacked. The created guide is necessary for other users to be able to freely edit the script according to the needs of their project since script writing is not easy to do or study.

For future studies, solving some issues like the floor to ceiling compliance that considers the axis instead of just the minimum ceiling and the highest flooring can be done to be able to make sure that a room is compliant with regards to the ceiling or not. Other technical issues can be smoothed out to be more seamless and convenient. A research on the proper modeling practices in coordination with the automated code compliance checker can also be done for a future study so that models will no longer require fixing.

Furthermore, future studies can look into expanding the covered codes like adding minimum door widths, number of exits, distances to exits and more. This study can also be used as a template to create an accessible automated code compliance checker for other types of projects like commercial projects, offices, industrial projects and more. It can even be possible to include elements of the other fields like the structural and mechanical elements. Moreover, adding in the other codes like the fire code or the socialized housing code is also an option. Basically, there are plenty of

possibilities that can be done to expand on this study, but this could be the basis or foundation for those other studies.

Visual programming itself is very useful and it is highly recommended for BIM users to also learn how to use visual programming tools. It can make plenty of tedious work much easier to do and the possibility of automizing many processes can make work very efficient.

Considering how important compliance is, another possibility is to pitch the automation to the government to gain access to funds that will enable the development of every permutation.

References

Allen, C., & Shakantu, W. (2016). *The BIM revolution: A literature review on rethinking the business of construction* [Conference presentation]. 11th International Conference on Urban Regeneration and Sustainability, 920–930

Anderle, M., & Allen, R. (2017). *Improve Your Team Efficiency: 20 Practical Uses of Dynamo for Revit.* Medium. Retrieved October 4, 2018, from https://medium.com/autodesk-university/improve-your-team-efficiency-20-practical-uses-of-dynamo-for-revit-f5f4a6313ab8

Association of Structural Engineers of the Philippines, Inc. (2013). *Educating Green Building Stakeholders About the Benefits of BIM - The Philippines Experience*. Medan, Indonesia

Autodesk. (n.d.). *Revit products*. Autodesk. Retrieved July 16, 2019, from https://knowledge.autodesk.com/support/revit-products/learn?sort=score

Building and Construction Authority. (2013). *Singapore BIM Guide*. Coronet. Retrieved October 4, 2018, from https://www.corenet.gov.sg/media/586132/Singapore-BIM-Guide_V2.pdf

Choi, J., & Kim, I. (2017). *A Methodology of Building Code Checking System for Building Permission based on open BIM* [Conference presentation]. 34th International Symposium on Automation and Robotics in Construction.

Cox, R. F., Issa, R., & Ligator, J. (2006). *Top ten Florida residential building code violations*. Gainesville, Florida: M.E. Rinker, Sr. School of Building Construction.

Davenport, T. (1992). Process innovation: Reengineering work through Information Technology, Harvard Business School Press, Boston. design and LEED® rating analysis. *Automation in Construction*, *20*(2), 217–224.

Dimyadi, J., & Amor, R. (2013). *Automated Building Code Compliance Checking - Where is it at?* https://doi.org/10.13140/2.1.4920.4161

Ding, Lan & Drogemuller, Robin & Rosenman, Mike & Marchant, David & Gero, John. (2006). Automating code checking for building designs-DesignCheck. *Clients Driving Innovation: Moving Ideas into Practice* (pp. 1-16). CRC for Construction Innovation. Retrieved July 10, 2019, from https://pdfs.semanticscholar.org/d6ae/240667b08ea077a9e6950cbd3fd8cfa04534.pdf

Dowsett, R. M., & Harty, C. F. (n.d.). *Evaluation the Benefits of BIM for Sustainable Design - A Review.* http://www.reading.ac.uk/web/files/tsbe/Dowsett_TSBE_Conference_Paper_2013.pdf

Eastman, C., Lee, J., Jeong, Y., and Lee, J. (2009). Automatic rule-based checking of building designs. *Automation in Construction*, *18*(8), 1011–1033. doi:10.1016/j.autcon.2009.07.002

Greenwood, D., Lockley, S., Malsane, S., & Matthews, J. (n.d.). *Automated compliance checking using building information models*, 266-274.https://www.irbnet.de/daten/iconda/CIB20057.pdf

Kalfa, M. (2018). Building information modeling (BIM) systems and their applications in Turkey. *Journal of Construction Engineering, Management & Innovation, 1*(1), 55-66. Retrieved July 16, 2019, from https://www.researchgate.net/publication/325422848_Building_information_modeling_BIM_systems_and_their_applications_in_Turkey.

Khemlani, L. (2005). CORENET e-PlanCheck: Singapore's automated code checking system. *AECbytes "Building the Future"*. Retrieved November 15, 2012, from http://aecbytes.com/buildingthefuture/2005/CORENETePlanCheck.html

Khemlani, L. (2011). AECbytes Building the Future. *AGC's Winter 2011 BIMForum*, Part 1.

Kia, S. (2013). *Review of Building Information Modeling (BIM) Software Packages Based on Assets Management.* Retrieved July 16, 2019, from https://www.researchgate.net/publication/253058808_Review_of_Building_Information_Modeling_BIM_Software_Packages_Based_on_Assets_Management

Kilkelly, M. (2018). *What is dynamo and 5 reasons you should be using it.* Archsmarter. Retrieved October 4, 2018, from https://archsmarter.com/what-is-dynamo-revit/

Kim, J., Clayton, M., & Yan, W. (2011). Parametric Form-Based Codes: Incorporation of land-use regulations into Building Information Models. *ACADIA Regional 2011: Parametricism,* 217-223. http://papers.cumincad.org/data/works/att/acadiaregional2011_025.content.pdf

Kuehmeier, J. (2008). *Building Information Modeling and its Impact on Design and Construction Firms.*

Mallach, A. (2013, March 26). *5 Things Cities and CDCs Don't Get About Code Enforcement.* Retrieved February 16, 2019, from https://shelterforce.org/2013/03/26/5_things_cities_and_cdcs_dont_get_about_code_enforcement/

Matějka, P., & Tomek, A. (2017). Ontology of BIM in a Construction Project Life Cycle. *Creative Construction Conference 2017,* 1081–1087.

McAuley, B., Hore, A. and West, R. (2017) *BICP Global BIM Study - Lessons for Ireland's BIM Programme Published by Construction IT Alliance (CitA) Limited*, 2017. https://doi.org/10.21427/D7M049

McGraw Hill Construction. (2014). *SmartMarket Report: The business value of BIM for construction in major global markets*.

ICN-Solutions. Retrieved from https://www.icn-solutions.nl/pdf/bim_construction.pdf

McGraw Hill. (2009). The business value of BIM: Getting Building Information Modeling to the bottom line, *SmartMarket Report*, 2008

Merlan, J. (2018, October 21). *Dynamo for Revit: What it is and why you should use it.* Unifilabs. Retrieved February 5, 2019, from https://unifilabs.com/dynamoforrevit/

Moullier, T. (2014). *Improving building code implementation and compliance for more resilient buildings in developing countries: considerations for policy makers.*

Nguyen, T., & Kim, J. (2011). Building code compliance checking using BIM TECHNOLOGY. *2011 Winter Simulation Conference,* 3400-3405. https://dl.acm.org/doi/10.5555/2431518.2431922

Philippine Information Agency. (2017, September 22). *Review Committee created to strengthen current Building Code.* Retrieved February 13, 2019, from https://pia.gov.ph/news/articles/1000386

Philippine Statistics Authority. (2018, December 19). *Construction Statistics from Approved Building Permits: Third Quarter 2018.* Retrieved February 27, 2019, from https://psa.gov.ph/content/construction-statistics-approved-building-permits-third-quarter-2018-preliminary-results

Preidel, C., & Borrmann, A. (2016). *Towards code compliance checking on the basis of Visual Programming Language*. ITcon. Retrieved October 2, 2018, https://www.itcon.org/papers/2016_25.content.01707.pdf

Reizgevicius, M., Ustinovi˘cius, L., Cibulskien˙e, D., Kutut, V., & Nazarko, L. (2018). Promoting sustainability through investment in Building Information Modeling (BIM) Technologies: A Design Company Perspective. *Sustainability*,*10*(3), 600. https://doi.org/10.3390/su10030600

Santos, S. (2015). *Dynamo: The Secret to Working Smarter and Making Your Life Easier.* ArchDaily. Retrieved October 4, 2018, from https://www.archdaily.com/773079/dynamo-the-secret-to-working-smarter-and-making-your-life-easier.

Seembu. (2016, October 14). *How to get a building permit in the Philippines.* Medium. Retrieved December 3, 2018, from https://medium.com/@seembu/how-to-get-a-building-permit-in-the-philippines-f4ae3684c06d

Sweet, R. (2014). Reversing Babel, *Construction Research and Innovation Magazine, 5* (2). https://doi.org/10.1080/20450249.2014.11873932

The Economist. (2017). *Efficiency Eludes the Construction Industry.* Retrieved September 25, 2018, from https://www.economist.com/business/2017/08/17/efficiency-eludes-the-construction-industry

Thompson, G. N. (1947). *The Problem of Building Code Improvement. Law and Contemporary Problems*, 12(1), 96-110. Retrieved February 15, 2019, from https://scholarship.law.duke.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=2292&context=lcp.

US Department of Energy. (n.d.). *Compliance.* Energy Codes. Retrieved July 12, 2018, from https://www.energycodes.gov/compliance

Uraiqat, B. (2013, January 30). Cyborg architects - the effect of digital design technologies on modern architectural design and representation. *Uraiqat.* Retrieved December 1, 2018, from http://uraiqat.blogspot.com/2013/01/cyborg-architects-effect-of-digital.html

Wengerd, O. (2015, January 12). This is how much more you'll pay for Autodesk software. *WorldCAD Access.* Retrieved January 27, 2019, from https://www.worldcadaccess.com/blog/2015/01/this-is-how-much-more-youll-be-paying-for-autodesk-software.html

Yan, H., Damian, P. (2008, 16–18 October). *Benefits and barriers of Building Information Modelling* [Conference presentation]. Proceedings of the 12th International Conference on Computing in Civil and Building Engineering, Beijing, China.