Automated Generation of BIM Models

Patrick Janssen[1], Kian Wee Chen[2], Akshata Mohanty[3]
[1,3]National University of Singapore [2]CENSAM, Singapore-MIT Alliance for Research and Technology, Singapore
[1]patrick@janssen.name [2]kianwee@smart.mit.edu
[3]akshatamohanty@gmail.com

In early stages of architectural design, highly simplified minimal models are often preferred while in the later stages maximal Building Information Models (BIM) are required that include the relevant information for detailed design documentation. This research focuses on the transition from minimal to maximal models and proposes a semi-automated workflow that consist of two main steps: analysis and templating. The analysis step starts with the minimal geometric model and decorates this model with a set of semantic and topological attributes. The templating step starts the decorated model and generates a transitional BIM model which can then be readily altered and populated with high resolution building information. A demonstration of two test cases shows the feasibility of the approach.

Keywords: *BIM, parametric modelling, interoperability*

INTRODUCTION

Architectural design typically relies on different tools at different stages of the design process. In the early stages, minimal models that are highly simplified are often preferred, for two reasons. First, minimal models are easier to build, thereby minimising the 'sunk costs' for options that get discarded. Second, minimal models are much faster to analyse and simulate, thereby allowing designers to receive timely feedback. In the later stages, maximal models are required that include all relevant information for detailed design documentation. Typically, Building Information Models (BIM) are created that progressively increase the amount of information, including information from other design consultants and specialists (Eastman, 2008).

In the transition from minimal to maximal models, there is a key point when designers need to switch from conceptual modelling tools to BIM tools. Before this point, planar entities such as walls and floor slabs may be modelled as single polygons and linear elements such as columns and beams may be defined as polylines. Once the switch is made to BIM tools, entities need to be more precisely defined as building elements, requiring them to be thickened and materials and other details to be defined. We refer to this as the *materialisation process*.

Materialisation Process

In this research, we focus on materialisation processes where the tools used for conceptual modelling differ from those used for BIM modelling. There are cases where the conceptual modelling is performed within the BIM tool, which then simplifies to some extent the materialisation process. However, designers often prefer to use different tools. Com-

monly used tools include Trimble SketchUp and McNeel Rhino for conceptual design, and Graphisoft ArchiCAD and Autodesk Revit for BIM modelling.

This research aims to explore the feasibility of semi-automatically converting minimal geometric models into transitional BIM models consisting of explicitly defined building elements, but with only limited building information. Although these models may not yet be maximal in their information content, their usefulness lies in the fact that they can be directly imported into existing BIM tools, where they can be easily modified and further developed.

This materialisation process is currently difficult to automate. For buildings designs with high levels of repetition, this may not be an issue as the time taken to manually remodel the design in the BIM tool is low. However, if the design includes any more complex non-repetitive configurations, then the remodelling becomes time-consuming and error prone.

The aim of semi-automating this materialisation process is to enable more fluid and flexible workflows that do not require BIM models to be manually recreated. However, this does not require the whole modelling process to be fully automated. Instead, workflows need to be developed that minimise the effort required to remodel the design. As a test case, the research will focus on a materialisation process in which the geometric models are imported as DXF models, and the transitional BIM models are exported as Industry Foundation Classes (IFC) models. The DXF file format is a commonly used dataformat for representing 3D geometry. The IFC file format (ISO 16739:2013) is an information-rich object-based file format for representing building information. The use of an open standardised file format ensures that the approach remains workflow agnostic, allowing users to link together diverse tools and systems to support various forms of collaboration and exchange.

One of the key requirement relates to the representations used to model IFC elements. When boundary representations are used, it becomes very difficult to further modify and developed the model using BIM tools. In contrast, when parametric representations are used, the process of importing IFC models into BIM tools generates native BIM elements that retain their intelligent behaviour are therefore easy to modify. The aim is therefore to use parametric representations wherever possible, and to only revert to boundary representations where it is absolutely necessary.

Existing Approaches

Within the existing ecosystem of modelling tools and plugins, there are a number of approaches that may address some of challenges associated with the materialisation process. Three main approaches will be briefly reviewed: geometric modelling tools that support IFC export, BIM modelling tools that support geometric import, and graph-based parametric BIM modelling systems.

Perhaps the most straightforward approach is to use a geometric modeller that is capable of exporting IFC. One well know example is SketchUp, the Pro version of which now supports IFC export. Using this approach, the user simply models the building elements as objects in SketchUp, and then assigns an IFC type to each of these objects. The downside with this type of approach is twofold. First, the full 3D geometry must be modelled, and as result there is not really any advantage over modelling it directly in the BIM tool. Second, the resulting IFC elements are generated using boundary representations, which means that they can no longer be easily modified once imported into BIM tools.

Another approach is to import the geometric model directly into a BIM tool, and then convert the geometric entities to BIM elements. An example of this is Revit, which allows geometric solids to be imported as massing models into Revit. The faces of the conceptual mass can then be selected and converted to selected BIM elements such as walls and slabs. The downside is that most BIM elements cannot be created in this way. For example, openings in the faces of solids can be imported, but cannot be subsequently converted to Revit elements such as

doors and windows.

A third approach is to consider using existing parametric modelling tools as a platform for defining and customising the rules for the materialisation process. Graph-based parametric modelling systems such as McNeel Grasshopper [1], Bentley GenerativeComponents [2], Autodesk Dynamo [3], and Sidefx Houdini [4] could be used. A number of workflows and plugins have been developed for generating BIM models using such parametric modelling tools. In general, there are two approaches: tightly coupled and loosely coupled.

With the tightly coupled approach, systems are coupled through the Application Programming Interface (API) provided by the BIM system. In this case, graph-based systems communicate via the API of the BIM system, directly instantiating geometry in the BIM model each time the graph-based model is executed. There are numerous examples of this approach. Grasshopper has plugins for connecting to both ArchiCAD (ArchiCAD Connection [5]) and Revit (Hummingbird/Whitefeet [6]). Dynamo has a built-in connection with Revit, and GenerativeComponents has a built-in connection with AecoSIM [7]. The downside of all these solutions is that they only work with specific BIM tools and are therefore not workflow agnostic.

With the loosely coupled approach, systems are coupled through model exchange. The graph-based system typically generates an IFC model that can be directly imported into the BIM system. An example of this approach is the GeometryGym [8] plugin for Grasshopper. In theory, it is possible for designers to create their own customised materialisation procedures using a plugin like GeometryGym. However, the downside would be the high complexity involved in doing so. This is primarily due to the fact that graph-based parametric modelling tools are not well suited to creating such rule-based procedures. In particular, the control flow of these types of procedures is typically complex, with a heavy reliance on conditional and looping constructs, for which graph-based systems are not well suited.

Paper Overview

This research proposes a generalised system for semi-automating the materialisation process. The paper consists of three main parts, with the first part describing a proposed system, the second part focusing on a set of experiments, and the third part discussing future challenge.

PROPOSED SYSTEM

A system for the semi-automated generation of transitional IFC models from geometric DXF models is proposed. The proposed system focuses on typical building elements such as slabs, roofs, walls, doors and windows. These types of elements have the advantage that they can be defined using the parametric representations within IFC. In the proposed system, the materialisation process is divided into two parts: the analysis step and the templating step.

The analysis step starts with the geometric DXF model and generates a new model that is enriched with a set of semantic and topological attributes, referred to as the *decorated model*. The analysis rules infer these attributes from the size, orientation, and relationships between geometric entities in the DXF model. Other than adding attributes, no changes are made to the geometry by the analysis rules.

The templating step starts with the decorated model and generates the transitional IFC model. Each of the geometric entities in the decorated model are mapped to parametrically defined objects in the transitional IFC model. The template rules are matched against geometric entities in the DXF model based on attributes created in the analysis step. If a rule matches, the IFC element specified in the rule is added to the IFC model.

In general, the system is envisaged as a web-based application that can be used to materialise models. Figure 1 shows a materialisation process consisting of four main steps: 1) upload the geometric DXF model; 2) execute the analysis rules; 3) customise execute the templating rules; and 4) download the transitional IFC model.

Step 3 is important as it allows the materialisa-

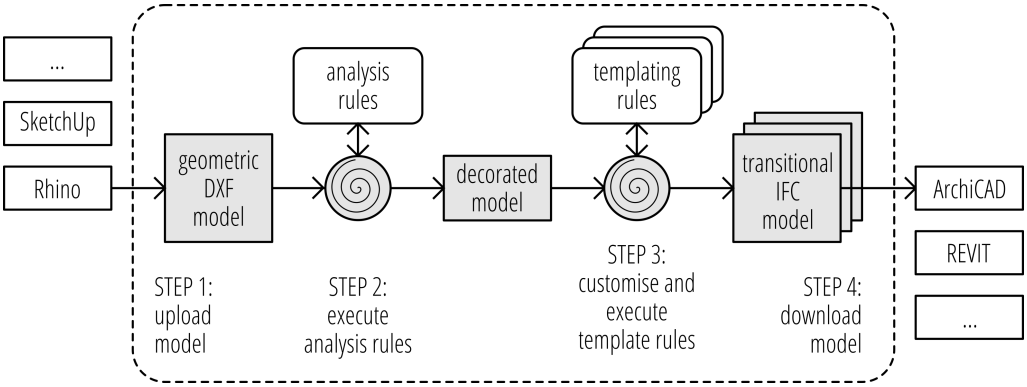


Figure 1
Proposed system for semi-automating the materialisation process.

tion process to be customised to the needs of the designer. The customised rules can take into account a variety of factors, including the scale of the project, the architectural design language, and the type of construction. Through an iterative process of tweaking rules, designers will over time be able to develop libraries of personalised rules.

In step 4, the transitional IFC model is downloaded and imported into the BIM application, where the IFC elements can be replaced with more complex native BIM objects. There is therefore no expectation that this materialisation process will create the whole model. It is more likely that the process creates a BIM model that can be readily populated with higher resolution information of the building project.

Geometric DXF Model to Decorated Model

The DXF model is assumed to contain highly simplified representations of a building following a set of modelling conventions, using only planar polygons and lines. Building elements such as walls, slabs, doors and windows are represented using polygons, while building elements such as columns and beams are represented using lines. The analysis step first assigns unique IDs to the entities in the model, and then analyses the individual entities and their topological relationships.

For lines, attributes are created to store the length and the direction vector. For polygons, attributes are created to store the number of vertices, perimeter, area, and normal vector. For polygons, specific shapes are recognised, and a series of Boolean attributes are created. These include *is-convex*, *is-rectangular* and *is-stable*. A *stable* polygon is defined as one whose lowest edge is horizontal. Lines and polygons are also categorised according to their angle of inclination, into one of four categories: *is-horizontal* (0°), *is-sloping* (1 to 45°), *is-leaning* (46 to 89°), and *is-vertical* (90°).

The relationships between lines and polygons are analysed for a given tolerance. For each type of relationship, attributes are created with the attribute value being a list of IDs of the entities involved in the relationship. Contact relationships are any lines or polygons that touch each other in some way. Attributes are created for various types of contact between points, edges, and faces. For polygons, two additional types of relationships are co-planarity and containment. Co-planarity refers to all polygons that are co-planar to a given polygon, with a single *co-planar* attribute being created for each polygon. Containment refers to a relationship between co-planar polygons, where the interior of one polygon completely contains another polygon. Attributes are created for both *contained-by* and *contains*.

Decorated Model to Transitional IFC Model

The templating step generates IFC elements from the entities in the decorated model. This step starts by creating the IFC spatial structure. The first three levels of this spatial structure are predefined, and consist of the *IFCProject*, *IFCSite* and *IFCBuilding*. Under the building level, a number of *IFCBuildingStories* need to be created. These building stories will contain all the IFC elements that belong to that storey, including slabs, walls, windows, columns and beams.

The templating step automatically groups the geometric entities in the decorated model according to building storeys they belong to. First, floors are detected by identifying all horizontal polygons with upward pointing normal. These floors are then grouped into co-planar sets, and sorted according to their z-heights in ascending order. These groups then represent the building stories. The remaining geometric entities in the model are then added to these groups according to their topological relationship to the floor polygons.

The geometric entities in each building storey group are then processed, starting with the lowest storey. For each group, an *IFCBuildingStorey* is created and the geometric entities in the group are then mapped to IFC elements using templating rules. The templating rules have two parts: a pattern and a template. The pattern part of the rule specifies the type of geometric entity (either *line* or *polygon*) and a set of attributes that the entity must have. The template part of the rule specifies one or more IFC elements to be instantiated.

The geometric representation of the IFC element is generated based on the shape, position, and orientation of the polygon or line. For example, an IFC wall [9] can be represented using a number of parametric representations. The *Swept Solid* representation of a wall defines the solid wall object by sweeping a planar profile using either linear extrusion or revolution techniques. The *clipping* representation defines the solid wall object as a result of series of Boolean operations in a Constructive Solid Geometry tree. Finally, the *IFCWallStandardCase* wall type is a parametric object that defines a wall by the centre line and parameters that specify the width and height.

In addition, a number of settings are applied to the IFC element. For example, for walls, the settings include the thickness and materials of the wall, as well as the position of the wall relative to the polygon. (For example, the polygon may represent either the wall centreline, the inner face or the outer face.)

As an illustrative example, a simplified version of the templating rules for generating IFC walls are shown below. In this case, the rules are designed to identify polygons that are walls. Such polygons must be stable, but the shape may vary, and it may be vertical or leaning. These attributes will affect the representation that is used. In addition, the 'contained-by' attribute is used to distinguish between walls and openings (such as doors and windows). For walls, the 'contained-by' attribute should be empty, while for openings, it should contain an ID of the polygon in which the opening is to be created.

- **IF** polygon *is-stable*, *is-vertical*, *is-rectangular* and *contained-by* is empty **THEN** *IFCWallStandardCase*, Representation=*Curve2D*.
- **IF** polygon *is-stable*, *is-vertical* and *contained-by* is empty **THEN** *IFCWallStandardCase*, Representation=*Clipped*.
- **IF** polygon *is-stable*, *is-leaning*, *is-rectangular* and *contained-by* is empty **THEN** *IFCWall*, Representation=*SweptSolid*.
- **IF** polygon *is-stable*, *is-leaning* and *contained-by* is empty **THEN** *IFCWall*, Representation=*Clipped*.

Similar rules exist or generate other types of elements. For example, rules for doors and windows have patterns that match polygons where the *contained-by* attribute is not empty. If a polygon is contained by a wall and its lowest horizontal edge is touching the lowest horizontal edge of the wall, it is categorised as a door. Otherwise, if it is not touching the lowest horizontal edge of the wall, then it is categorised as a window.

Default templating rules may initially be used,

but designers need to be able to customise these rules to suit their needs. Designers can make either globalised or localised customisations. Globalised customisations include general modifications to the default setting. Localised customisations only apply to certain subsets of elements, defined through the use of layers in the DXF file. For example, the designer may want to have a variety of different types of walls, with different settings. In such a case, the wall polygons can be placed on different layers, and templating rules can then be created that include the layer name in the pattern. This then ensures that these rules will only apply to polygons on those layers.

EXPERIMENTS

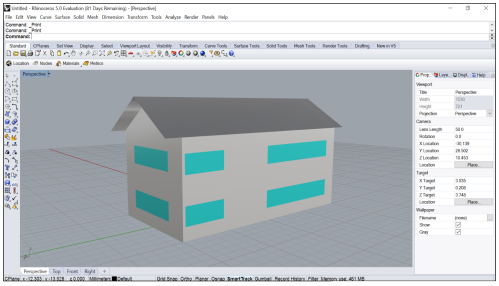
The feasibility of the proposed system is demonstrated through a number of experiments in which the creation of a set of analysis and template rules were tested. These experiments used Rhino for the conceptual modelling, Houdini for defining and executing the rules, and ArchiCAD for importing the IFC BIM model.

In these demonstrations, Houdini allowed for a fast way of prototyping and testing rules. For generating IFC models, an exporter was implemented for Houdini using the *IfcOpenShell* Python library. This allows the exported IFC model from Houdini to be imported into ArchiCAD, where plans and sections can be generated.

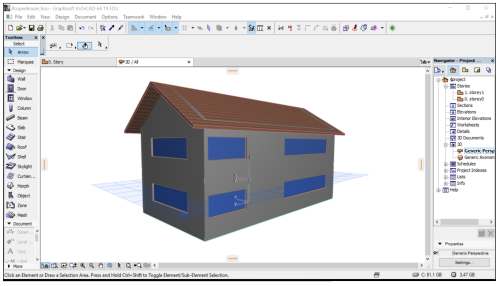
House Example

In order to illustrate how these different rules may operate, a simple example may be considered, as shown in Figure 2. The example is based on the *IfcOpenHouse* [10], where an extra level is added to increase the complexity. The example consists of two floors, six windows; three on each level, one door on the first level and a roof at the top level.

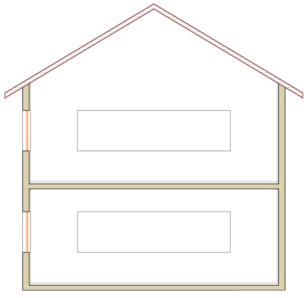
In the analysis step, the DXF model imported from Rhino is analysed and a decorated model is generated with additional attributes. For this example, there are no columns and beams, and as a result the model only contains polygons.



(a)



(b)



(c)

Figure 2
(a) House conceptual model in Rhinoceros3D (b) House IFC model in ArchiCAD (c) Section generated from the IFC model in ArchiCAD.

In the templating step, the decorated model is converted into a transitional IFC model. This step starts by identifying all the floors as horizontal polygons with upward pointing normal, of which there are two. Since these floors are not co-planar, two separate *IFCBuildingStories* are generated: a ground storey and a first storey. The remaining entities in the geometric model are then categories into the different building

storeys, depending on the height of the polygons relative to the floors. This results in 8 polygons on the ground floor (4 walls, 3 windows, and a door) and 9 polygons on the first floor (4 walls, 3 windows, and 2 roofs).

The templating rules are then applied to each storey in turn, thereby generating the various IFC elements for each floor. In most cases, the automated mapping of the geometric entities to IFC elements is straightforward. Walls are mapped to *IFCWallStandardCase*, with a thickness of 200mm. Floors are mapped to *IFCSlab* of type *Floor*, and roofs to *IFCSlab* of type *Roof*, both with a thickness of 300mm. Windows and doors are mapped to *IFCWindows* and *IFCDoors* respectively, with a thickness of 50mm. For the windows and doors, an *IFCOpeningElement* also needs to be added to the containing wall to define an opening, with a thickness that is equal to that of the wall.

The walls are slightly more complex due to the fact that different representation are required for the ground floor and first floor. On the ground floor, the walls are constant height, so an *IFCWallStandardCase* with a *Curve2D* representation can be used. In this case, the wall height is determined by the distance between the two floors. On the first floor, the walls need to be connected to the sloping roof, and as a result the walls vary in height. An *IFCWallStandardCase* with *Clipped* representation is therefore used. Figure 2b shows the transitional IFC model imported into ArchiCAD and Figure 2c shows a resolved section generated from the IFC model.

Complex Tower Example

A similar procedure as the one used for the House example is applied on a complex tower as shown in Figure 3. First, a minimal geometric model of a multistorey building was created using Rhinoceros3D.

In order to create additional complexity, a twisting tower is generated by turning the floor plates on each level (Figure 3a). In addition, columns and beams are also added, modelled as vertical and horizontal lines respectively. The default profile for the columns is a 500mm diameter circle, while for the beam it is a 500mmx500mm square.

Figure 3 shows how the minimal model can be quickly translated into a transitional IFC model. Once the IFC model is imported into a BIM application, it can be readily altered and populated with higher resolution data, and used to generate a variety of plans and sections for further development.

FUTURE WORK

The two test cases show the feasibility of a semi-automated workflow for generating BIM models from minimal conceptual models. However, there are clearly many limitations that require further research and exploration. The aim of this approach is to be able to automate the generation of BIM models for standard types of building elements and configurations, thereby allowing architects and designers to focus on the more complex parts of the modelling. However, even the standard building elements present numerous challenges.

As an example, the issue of variable wall thicknesses may be considered. Figure 4 shows a plan view of a wall intersection, with different wall thicknesses. In (a), the walls in the geometric model are aligned, the designer may want the same alignment to be maintained in the BIM model, so that the walls remain flush. However, if the polygons in the geometric model are assumed to represent the wall centre plane, then this will result in a misalignment, as shown in (b) and (c). In order for the walls to end up flush, the designer would have to misalign them in the geometric model, as shown in (d). Such an approach would clearly be highly unintuitive.

An alternative approach would be to develop more intelligent rules that could infer such conditions. For example, the rule that generates the thinner wall may be able to detect that by shifting the wall slightly to the right, this misalignment can be avoided. However, any such 'intelligent' behaviours will also need to have overrides, just in case there is a situation where a designer actually intends the walls not to be flush.

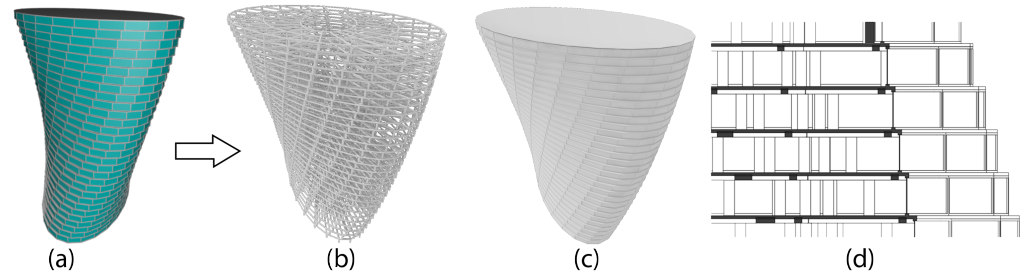


Figure 3
Complex tower example. (a) The minimal geometric model in Rhino. (b & c) The IFC model imported into ArchiCAD model. (d) A section generated from the ArchiCAD model.

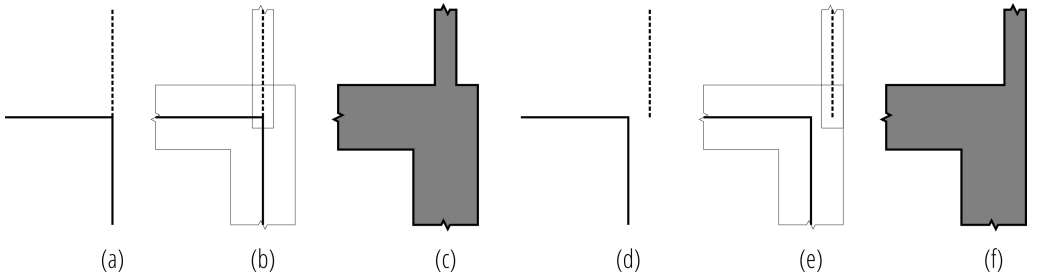


Figure 4
Misalignment of walls connections and intersections due to varying wall thickness. (a, b & c) Polygon alignment in the geometric model result in a misalignment of the walls in the BIM model. (d, e, &f) Polygon misalignment in the geometric model results in a alignment of the walls in the BIM model.

The challenge is therefore how to keep the process as a whole simple and intuitive for the designer. This requires a balance between, on the one hand, more advanced rules with intelligent inference, and on the other hand, intuitive and simple ways for the designer to override this intelligence. Future research will explore how such a balance can be achieved.

REFERENCES

Eastman, C 2008, *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, Wiley

Janssen, P and Stouffs, R 2015 'Types of Parametric Modelling', *Proceedings of the 20th International Conference of the Association of Computer-Aided Architectural Design Research in Asia CAADRIA*, Hong Kong, pp. 157-166

Patrick, J 2014 'Visual Dataflow Modelling: Some thoughts on complexity', *Proceedings of the 32nd eCAADe Conference*, Newcastle, UK, pp. 547-556

Woodbury, R 2010, *Elements of Parametric Design*, Routledge, Oxon

[1] http://www.grasshopper3d.com/

[2] https://www.bentley.com/en/products/product-line/modeling-and-visualization-software/generativecomponents

[3] http://www.autodesk.com/products/dynamo-studio/overview

[4] https://www.sidefx.com/

[5] http://www.graphisoft.com/archicad/rhino-grasshopper/

[6] http://www.grasshopper3d.com/group/hummingbird

[7] https://www.bentley.com/en/products/brands/aecosim

[8] https://geometrygym.wordpress.com/

[9] http://www.buildingsmart-tech.org/ifc/IFC4/final/html/

[10] http://blog.ifcopenshell.org/2012/11/say-hi-to-ifcopenhouse.html