Operations Research Forum (2024) 5:52
https://doi.org/10.1007/s43069-024-00332-w

RESEARCH

Proposing a Computational Modeling Framework for Generating Masonry Wall Units, Enhancing the Information Within a BIM

Austin D. McClymonds[2] · Somayeh Asadi[1,2] · Robert M. Leicht[2]

Received: 9 March 2024 / Accepted: 21 May 2024 / Published online: 8 June 2024

Abstract

In recent decades, the construction industry has undergone a technological shift incorporating innovative technologies, such as robotics. However, information requirements must be met to integrate robotics further. Currently, building information models (BIM) contain substantial project information that can be leveraged for robots to create construction tasks, but for some building systems, the level of development (LOD) is inadequate to support these new requirements. Therefore, this study proposes a framework to increase the LOD of building systems by considering location information (X, Y, Z), orientation, material type, and component I.D. The computational modeler, Dynamo, is leveraged to increase the model's LOD, extract information, and facilitate robotic task execution in the future. A case study is presented for multiple masonry room configurations developed in Autodesk Revit, where masonry units are generated and placed into design locations based on the geometry of the wall system. The case study used concrete masonry units (CMU) and standard brick. The number of partial-sized and full-sized blocks for each configuration was recorded, along with the computational time required to generate the units. It was observed that room configurations with more openings had longer computational times when compared to rooms constructed from the same material. After running the script, the model is reviewed to ensure accuracy and prevent overlaps or gaps in the model. The workflow provides insight into the methods used to interpret model geometry and extract information.

Keywords Computational modeling · BIM · Robotic construction · Information exchange · Masonry construction

This article is part of the Topical Collection on *Math for SDG 9 - Industry, Innovation and Infrastructure*

Extended author information available on the last page of the article

1 Introduction

Digitization in the construction industry is not a new concept, as building information models (BIM) have been an essential source of project information. BIM, by definition, is the digital representation of model characteristics of a building, serving as a source of information throughout a project lifecycle [1]. In addition to digitization, researchers are combining it with automation. Automation, or more specifically robotics, has been the target of recent research in the construction industry. One study reviewed efforts in the construction industry, documenting the various uses for construction robots, ranging from bricklaying to tile-placing robots. Additionally, they detailed unsolved technical problems, notably interoperability between different information systems [2]. As a result, Construction 4.0 has been established with its ability to combine factors of both automation and digitization [3], which has ushered in new applications in artificial/virtual reality (AR/VR), cyber security, big data analytics, laser scanning, automation, and robotics [4]. A recent study by McKinsey Institute identified a direct link between the digitization of BIM and robotics as a technology map. However, the bridge between BIM and robotics in construction lacks exploration, suggesting improving interoperability [5]. Methods to bridge the gap between BIM and construction robots require further exploration into the standard methods and procedures. Therefore, this study seeks to explore the bridge between construction robots and BIM further.

To facilitate BIM to robot construction, the level of development (LOD) of the 3-D model must be considered. According to BIMFourm, model LOD is defined as the degree to which a component specification, geometry, and associated information are detailed in the 3-D model of the project. Additionally, BIMFourm publishes an open standard for LOD to increase a given project's interoperability [6]. However, building systems are rarely modeled to the same LOD within the BIM model. Wall systems are a prime example, where the LOD is represented as a simplistic wall texture typically in two dimensions, while mechanical systems have most of the individual components modeled. This results in models that require material and geometric supplemental information to increase the LOD [7]. In general, additional information to increase the LOD is provided from the project specifications,however, the required information is derived from workers' experience or rules of thumb that are not recorded in the specifications or the model. Computational modeling provides a potential solution for adding supplemental model information by increasing the LOD to overcome this challenge. Computational modeling is the process of changing the shape of model geometry as soon as the dimension value is modified [8]. Software programs like Dynamo, Rhino, and AutoCAD 3D use generative design to leverage content libraries to generate new model content, promoting model development. Additionally, modelers can be classified by how they modify and interact with the model and are organized into three categories: modelers (based on objects, limited access to parameters, programs include ArchiCAD and Revit), semi-restricted modelers (greater freedom and can intervene on design operations, programs include 3DS max and AutoCAD), and free modelers (complete freedom and design capabilities, programs include Dynamo and Grasshopper) [9].

This study aims to design and evaluate a framework for a computational modeling process to increase the LOD of wall systems, specifically masonry walls, within BIM based on model parameters, such as system geometry and materials. To achieve this goal, free modeler programs are used for computational modeling to achieve an LOD of 400, which contains information required to construct the building system. The process employed by this study seeks to integrate supplemental information sources, including material content libraries and technical specifications. Three challenges have been identified in pursuing this goal: (1) interpretation of model geometry and increasing LOD, (2) integrating external data sources and material content libraries, and (3) information interoperability and standardization within the model and among various programs [10, 11]. Two objectives were developed for this study: first, to increase the LOD of a masonry wall system developed in a 3-D model, and second, to provide the means to extract this information from the BIM model to facilitate robotic construction. A case study using generative design methods was considered wall configurations modeled in Autodesk Revit to incorporate the individual masonry units of the system, utilizing Dynamo. While the methods used to develop the Dynamo script are specific to masonry wall systems and are indented as proof of concept, similar approaches can be taken for additional building systems. The procedure presented in the study only generates the masonry units. Future studies will seek to incorporate additional system components, such as mortar joints and lintels.

2 Literature Review

Parametric modeling is not a recent innovation in the construction industry, with early applications being tracked back to the 1990s with Autodesk Inventor and Bentleys MicroStation [12]. In more recent research, endeavors have further integrated parametric modeling with technology, such as robotics. Methods exist to transfer information from a BIM model, spreading across a broad application platform that utilizes computational modeling. Kalkan Okur et al. [13] reviewed these applications and use cases and found that parametric modeling can facilitate design optimization, change model parameters, generate model content, convert CAD content, and update models quickly across the entire project [13]. A typical use case of computational modeling is to facilitate design optimization. One example investigated a toolbox approach for the relationship between design optimization and data science in Grasshopper utilizing plugins and material content libraries. In this approach, they determined the existing functionality of parametric design, which is the interpretation of model geometry (curves, surfaces, lines, and solids), interpreting to perform a simulation, and mathematics to combine numerical design variables [14]. Another use case associated with computational modeling is to transform and alter the geometry in a model. In a study by McClymonds et al. [15], they defined a preliminary information exchange process between a BIM model and a robot, finding the need to increase the LOD to facilitate the transfer of information in robotic construction. The case study developed used a manual approach to increase a model LOD but cited that third-party applications could automate this process [15]. However,

challenges are associated with utilizing computational modeling to generate and extract model content with third-party applications.

The first challenge concerns the interpretation of model geometry and increasing LOD. Model geometry is the shape, structure, and interaction between systems in the model. One study found that native features of Dynamo are insufficient to interpret the geometry to increase a model to LOD 400, leveraging Python to interpret geometry; however, little insight was given into the geometry process. Additionally, they provided insight into the capabilities of computational modeling, which are the ability to fill information on sheets, place families and components, import/export models, and leverage Python [16]. Another study found that the BIM model may not contain sufficient details or may be lost while undergoing information exchange, specifically mentioning industry foundation class (IFC) schema and citing interoperability issues between project stakeholders [7]. Davtalab et al. [17] developed custom software to extract model data supporting additive manufacturing, which uses the BIM model as a data source. The model's geometry was simplified into 2D planes that indicated wall locations [17]. While this study did not directly involve computational modeling, it investigated the information exchange for BIM to robotic construction. It found that model geometry must be simplified for some use cases.

The second challenge for model content generation is integrating external data sources and material content libraries. Zhang and Xing [18] defined the requirements of a material content library, stating that it serves as a centralized repository for a product or material containing detailed information, including specifications, manufacturer data, and other relevant information. Therefore, a predefined model can be created in place of a generic component. In another study by Sharif and Gentry (2015), they developed a material content library as a masonry construction database consisting of blocks containing material properties, manufacturer, geometry, and textures. However, Sharif et al. [19] determined that generating custom units for areas of complex geometry is beneficial for those locations. Locations for custom masonry unit generation would be for partial-sized units or containing non-typical masonry attributes [19]. Later, in a study done by Kim and Chin [20], they found that there are typically two methods for developing material content libraries,the first model was developed based on dimension and constraints providing detailed information for a Revit Family, and the second model was reliant on the geometric description language described by parameters and algorithms [20]. However, Venkatraj and Dixit found in their study that the information contained within the material content library is based on the specific use case, which influences model parameters and the overall database [21]. Another study identified industry expertise and planning rules as input into their modeling system to develop a workflow to generate floor tiles in an apartment, locating areas that need to be cut to fit. They simplified their process from a 3D to a 2D model and used Grasshopper to generate the locations [22]. Notably, the information in the material content library is based on the building system and will require additional research on the methods to integrate it into the computational modeling process for BIM to construction robots.

Additionally, while material content libraries are considered a primary external data source, other sources must be included, such as topological/GPS data, weather data, robot capabilities, logistical information, specifications, assembly data, and

sequencing information [10, 11]. A study by Kim et al. [23] sought to integrate a robot operating system (ROS) with BIM for robotic task planning for drywall painting. A customized XML file was exported and used as the data source for the sequencing information. It was used in a simulation rather than combined with the BIM model [23]. However, the construction schedule must be integrated into the process to aid with sequencing to improve this process. In another study, Karimi et al. [24] investigated the ontology of data exchanges for robotic navigation utilizing a script developed with Dynamo to create semantic topological data in IFC files to aid in robotic navigation and data collection. Notably, the methods employed to integrate these external data sources vary, and each took the initiative to help with standardization and address interoperability concerns.

The third challenge stems from interoperability issues and standardization, which presents a considerable challenge in the construction industry, impacting the integration of innovative technology. In a study conducted by Ren and Zhang [7], they focused on developing a framework to address interoperability issues while transferring information between an architectural and structural model [7]. In another study, Tibaut et al. [25] found that interoperability must be considered in all construction project phases to facilitate information exchanges better. However, formats like IFC support simple models, resulting in missing information when the exchange occurs [25]. Anane et al. [26] recently investigated interoperability related to computational design driven by BIM to help bridge the gap between robotics and construction. They found that current tools are insufficient for complex projects and inadequate for data management,finding more work is required to improve the process [26]. Notably, standards are being developed and maintained by BuildingSMART to support the IFC schema, which is one of the adopted formats for exporting information [27]. However, while efforts have been made to increase interoperability and standardization, the variation in approaches in the previous studies shows that more effort is required.

This section documented the challenges of previous studies, including the interpretation of model geometry and LOD, integration of material content libraries and external data sources, and interoperability issues between programs. Considering the challenges described in this section, it becomes clear that while methods exist to increase the LOD of models, more work is required to address the challenges defined by this study. Interoperability is critical when transferring information between a BIM model and a robot, especially when considering a system of lower LOD. Therefore, this study develops a method that increases a model LOD of development, limiting the areas where interoperability issues can occur. This study creates a method to address these challenges using computational modeling to increase the LOD and extract information from a 3D model developed in Autodesk Revit.

3 Computational Modeling Framework

Computational modeling is a powerful tool that allows for rapid modifications to a BIM by changing model parameters, such as an element's size, shape, and properties. Additionally, by leveraging computational modeling, design accuracy and efficiency can be improved, facilitating collaboration and communication among

project stakeholders. However, building systems are not equally represented within a BIM; some systems are more conceptual and have a low LOD, while others that contain construction information have a higher LOD. To facilitate robotic construction and leverage the data, the LOD must be sufficient to support the construction within a BIM. To this end, this study utilizes Dynamo as a computational modeler to generate masonry units based on the geometry properties of a wall, thereby increasing the LOD. A framework was developed in Fig. 1 that presents the method used by computational modeling for data visualization within Revit and, eventually, facilitates robotic task creation by adding location and material information to the model. While the entire process is depicted, this study focuses on the generation and extraction elements in the framework and does not use the information for robotic task execution. This study extends a previous study that defined a system architecture for BIM to robotic construction [10, 11].

The framework shown in Fig. 1 can be extrapolated further and divided into three distinct phases, forming the phases shown in Fig. 2, which include (1) model setup, (2) model development, and (3) information exchange. Phase one (i.e., model setup) gathers the entire model and ensures it was developed correctly, enabling the script to run as expected and determine the requirements for the process. The model is reviewed manually to ensure it was developed to work with the script's structure. Phase two (i.e., model development) interprets the model geometry and places the components into the design locations determined by analyzing the wall texture shown in the original Revit model, and phase three (i.e., information exchange) extracts and formats information from the model for a robot. The following three sections provide further details about the inputs and outputs of each phase and the development process. An example is used in the following section to help illustrate the steps shown in Fig. 2.

Additionally, any task marked with an asterisk in Fig. 2 was done manually. Notably, this research primarily focused on automating the tasks listed in Phase 2. Not all tasks shown in Fig. 2 were completed in this study, including format, sequence

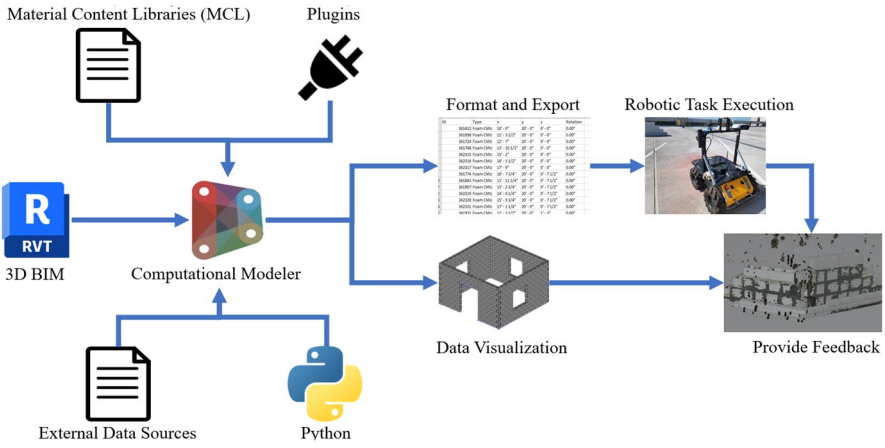

Fig. 1 Computational modeling framework for BIM to robotic construction integration

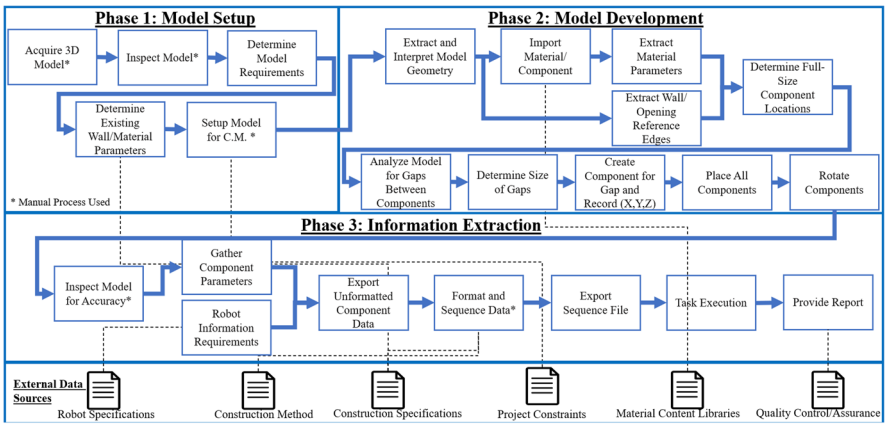

Fig. 2 Computational modeling workflow to generate model content

data, and task execution. These tasks were not undertaken as they extended beyond the scope of this study. The work presented by this workflow expands on the work completed in a previous study [28].

Dynamo plugins were leveraged in this study to support the script's development. While this study primarily used the built-in nodes, Table 1 lists the plugins used and summarizes their purpose. These packages and the creation of custom nodes allowed for a more straightforward geometric interpretation of model geometry and promoted the information extraction process. Additionally, using plugins reduced the number of Python scripts needed to facilitate the generation of the masonry units.

3.1 Phase One (Model Setup)

Model setup prepares the model to generate elements in phase two. For this study, an 8″×8″×16″ (20.3 cm×20.3 cm×40.6 cm) CMU room was developed in Autodesk Revit. The material of the wall is represented as a 2D texture and does not contain the location or material information at a component level for individual wall

Table 1 Dynamo plugins used

Plugin Name	Plugin Purposes
Clockwork	List management and determining element locations.
Dynamo Iron Python 2.7	Convert the wall directions from a vector into an angle.
Modelical	Extract the direction of the wall as vector.
RIE	Interact with bounding boxes for door and window openings.
Bimorph Nodes	Run Python scripts.

elements. The room configuration is shown in Fig. 3 and serves as the initial model for each workflow phase where on left (A) is the 3-D representation and the right (B) is the plan view.

Once the model is acquired, it is manually inspected to determine the supplemental information requirements. As stated, the wall material is represented as a texture and, by BIMFourm LOD guidelines, is an LOD 200. However, to contain information related to the construction process (i.e., component location and material information), the model must be LOD 400. The existing material parameters are determined based on the naming schema standard published by the National Concrete and Masonry Association (NCMA, 2017) to determine the correct wall material. For this study, wall materials reference NCMA standards for naming convention; therefore, an example would have an $8'' \times 8'' \times 16''$ *CMU* listed as the wall material in Revit.

The last step for phase one is to ensure the model is configured correctly. First, as observed during development, the method used for modeling the walls initially in Revit affects the geometric interpretation of the wall, impacting component placement. For instance, for this study, the walls were modeled with the location line set to the finish face exterior. Additionally, to ensure that the masonry units are generated correctly at corners, Fig. 4 was developed, which demonstrates how each wall must be modeled to facilitate the computational modeling process where each wall intersects the previous one. The walls, organized in the order they were modeled, are represented by a color: blue is wall one, orange is wall two, green is wall three, and red is wall four. A circle represents a start location, while a triangle is the end location. Identifying each wall's start and end points is required to establish the criteria for generating the individual wall components using the wall's exterior edge as the reference. For the computational modeling script to run correctly, it is crucial that the next wall begins where the previous one ended, or it would result in overlapping blocks. The start and end locations from the wall are used to determine the locations of the individual components for the first course. For the secondary course, these blocks are offset at a distance equal to half the block's length, resulting in

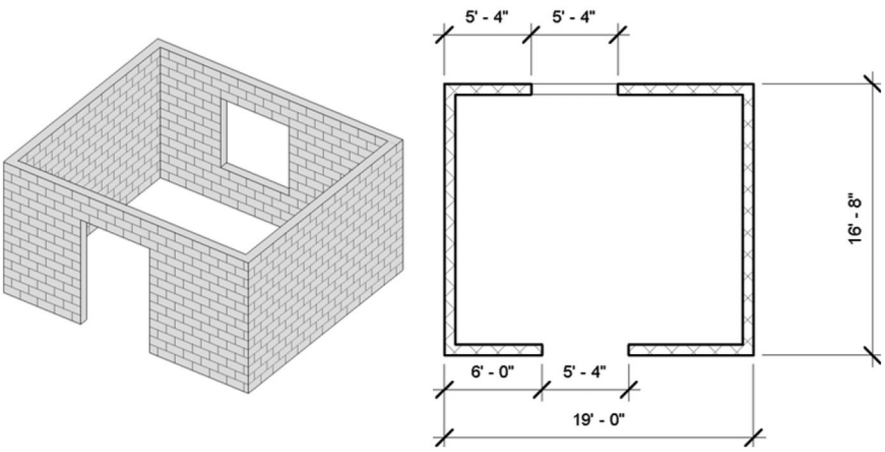

Fig. 3 **A** — Left 3-D representation of room configuration. **B** — Right Dimensioned 2-D plan of room configuration

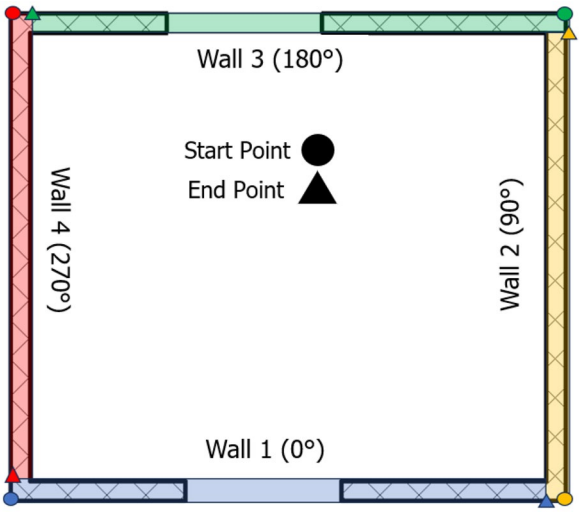

Fig. 4 Reference edge start locations modeled

a running bond pattern. Once the model setup is complete, phase two can begin. It should be noted that developed models were designed for the Dynamo script. A model obtained from an outside source must be manually reviewed and edited to ensure the computational modeling script runs correctly.

3.2 Phase Two (Model Development)

Model development is phase two of the computational modeling process, which imports components from the material content library; extracts material parameters, such as the masonry unit's length, width, and height; interprets geometry; and generates model content into their final locations. The first step of this phase is to gather and input the walls into Dynamo; the following nodes allowed this to occur: "All Elements of Category," which was attached to the "Walls" node to gather all the elements quickly. However, these nodes can be switched to allow the user to select specific walls. To import material from the material content library, a custom node was created in Python that interpreted the wall material and imported it into the model from the material content library. Once the material is imported, its dimensional parameters are extracted. The following paragraphs detail the geometric interpretation process for wall geometry shown in Fig. 5.

After determining the dimensional parameters of the wall material, the geometry of the wall is interpreted, and reference lines for the horizontal exterior edge are extracted. The start and end points for each reference edge, identified earlier in Fig. 4, are represented as either a circle (start point) or a triangle (endpoint). A filter removed reference lines that could cause units to overlap during component generation, such as the references above and below openings. Once the filtering is completed, the locations for each block are determined by first importing the full-sized component from the material content library. Only full-sized blocks ($8'' \times 8'' \times 16''$ CMU) are imported, while partial-sized units (length $< 16''$) are automatically

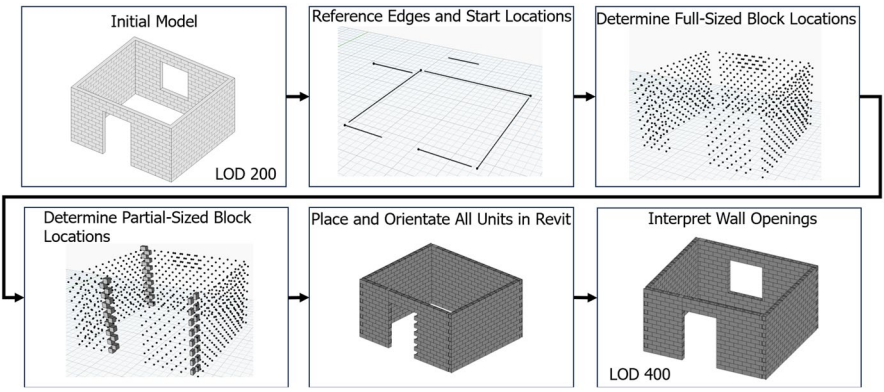

Fig. 5 Process for interpreting model geometry for a masonry wall

generated and converted into a family. Therefore, the process for determining the locations differs for generating full and partial-sized units.

Locations of full-sized units are determined prior to locating partial-sized units. To determine the locations, points are arrayed from the start location of the reference edge at a distance equal to the block's length, assuming the masonry unit's nominal size. The current iteration of the script does not generate mortar joints, which requires additional revisions to incorporate. If a point is located and extends beyond the reference edge or if the distance between the last point and the reference edge endpoint is smaller than the length of the block, it is removed. Additionally, a running bond pattern is represented in the model; therefore, the point location of the alternating rows is offset by a distance equal to half a block length, which results in two different row configurations in the wall. The script implemented a slider to adjust the offset amount, changing the bond configuration. Therefore, points located for the first course of blocks (blocks at ground level) are offset half a block length and up the height of a block. Once points are determined, they are duplicated for the remainder of the courses that construct the wall. The number of courses is dependent on the height of the wall.

To determine the location of the partial blocks, gaps between the end point of the reference edge and the endpoint of the nearest full-sized blocks are located. If a gap is located, the length is measured from the reference edge endpoint to the endpoint of the previous full-sized block in that course. Those points are recorded and translated along the z-axis at a distance equal to a block's height, creating a rectangle that is extruded by a full-sized block depth. The block is converted into a Revit generic model, and parameters are created for its dimensions (length, width, and height), and the partial-sized units are automatically generated in the identified locations. Additionally, the full-sized units are generated at their identified point locations. Once the partial-sized and full-sized units are generated in the BIM, they are orientated based on the direction of the wall they construct. Regarding the robot, the locations, orientations, and types of blocks would be used to generate robotic tasks in future studies.

A bounding box was created for each opening that intersects the wall, such as window and door openings. All components generated within the opening's bounding box were compiled into a list. If a component intersects the opening, it is removed from the model. However, if a full-sized element only partially intersected the opening, it was replaced with the proper-sized partial block. The method for determining the partial size follows the same protocol as the rest of the partial elements. Once all units are generated and orientated, information about material and location parameters is extracted.

3.3 Phase Three (Information Extraction)

Information extraction is the final step, which extracts information from the Revit model to facilitate robotic task creation. After generating the model content, the model is manually inspected to determine whether the units were placed into the proper design locations and ensure there are no overlapping units or gaps in the walls. The blocks were generated over the original wall model for testing and verification to ensure they were placed within the geometry bounds. The original wall was then removed, leaving the generated blocks in place.

After review, elements are compiled into a list, where information is exported as a.CSV file, which includes coordinates (point location for full-sized and partial CMU), orientation (rotation in degrees), type (i.e., $8'' \times 8'' \times 16''$ CMU), and identification number (I.D.) The coordinates reference the block's front left bottom corner and are based on Revit's localized system. Table 2 is a subsection of ten elements showing the unformatted extracted information, which does not currently represent the construction sequence. They are extracted in order of component I.D. Regarding the rotation for this case, 0 degrees refers to the south wall, 90 degrees to the east wall, 180 degrees to the north wall, and 270 to the west wall based on the script's configuration. The information extracted from the BIM can be used to create tasks

Table 2 Extracted location and material information for room configuration

I.D	Type	Coordinates (Feet and Inches)			Coordinates (Meters)			Rotation (Degrees)
		X	Y	Z	X	Y	Z	
12195453	8X8X12	20' – 0 15/16"	39' – 10 1/4"	1' – 4 "	6. 12 m	12.15 m	0.41 m	0
12195454	8X8X12	20' – 0 15/16"	39' – 10 1/4"	2' – 8"	6. 12 m	12.15 m	0.81 m	0
12195469	8X8X8	24' – 8 15/16"	23' – 10 1/4"	1' – 4"	7.54 m	7.27 m	0.41 m	0
12195470	8X8X8	24' – 8 15/16"	23' – 10 1/4"	2' – 8"	7.54 m	7.27 m	0.81 m	0
12195492	8X8X4	38' – 0 11/16"	23' – 10 1/4"	2' – 0"	11.60 m	7.27 m	0.61 m	0
12195493	8X8X4	38' – 0 11/16"	23' – 10 1/4"	3' – 4"	11.60 m	7.27 m	1.02 m	0
12195503	8X8X16	37' – 0 11/16"	40' – 6 1/4"	0' – 0"	11.30 m	12.35 m	0 m	180
12195504	8X8X16	35' – 8 15/16"	40' – 6 1/4"	0' – 0"	10.90 m	12.35 m	0 m	180
12195708	8X8X16	19' – 4 15/16"	40' – 6 1/4"	0' – 0"	5.92 m	12.35 m	0 m	270
12195709	8X8X16	19' – 4 15/16"	39' – 2 1/4"	0' – 0"	5.92 m	11.94 m	0 m	270

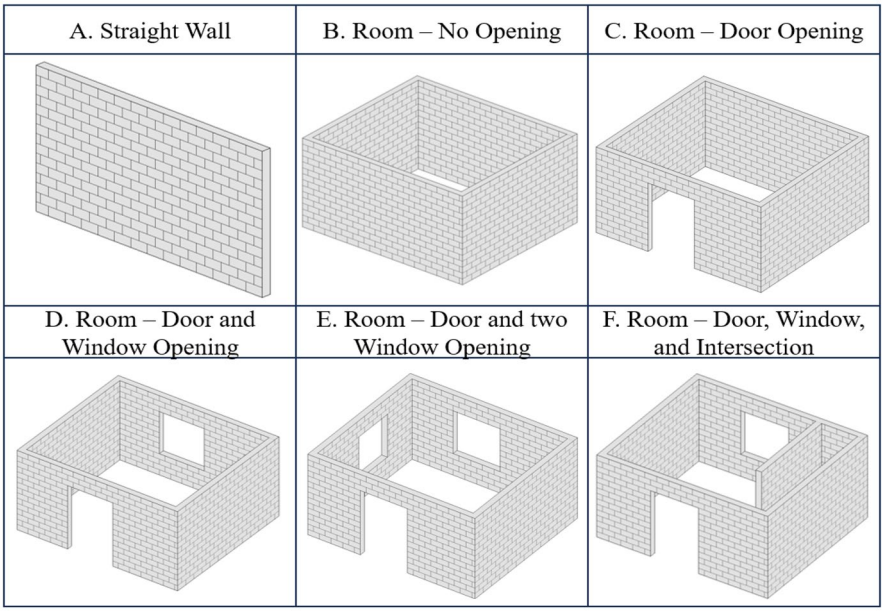

Fig. 6 Initial room configurations

for the robot, such as progress detection, material delivery, or self-performing tasks. However, the tasks conducted by the robot are heavily dependent on the robot.

4 Results

For this study, six different room/wall configurations were developed and modeled in Autodesk Revit, presented in Fig. 6. This study will refer to them all as room configurations from this point.

The room configurations were designed to slightly increase the model's complexity and evaluate the script's ability to generate model content. For this study, the simplest case is A, where only one wall is modeled, which then advances to a room containing four walls. From there, additional openings or intersections are added to the model to increase complexity further and evaluate the script. The computational modeling interpreted the wall geometry to generate the models above, which were constructed out of $8'' \times 8'' \times 16''$ (20.3 cm × 20.3 cm × 40.6 cm) CMU. The wall configuration was designed with a running bond pattern. The straight CMU wall (Configuration A) was a control for this study as it was used as the primary configuration for testing early script iterations and provided a baseline for comparison. Room configuration B added three walls, creating a box. Room configuration C added a door opening to the front wall, as early testing indicated that the additional opening increased the computational intensity of the process. Room configurations D and E introduced one and two window openings, respectively, to increase the

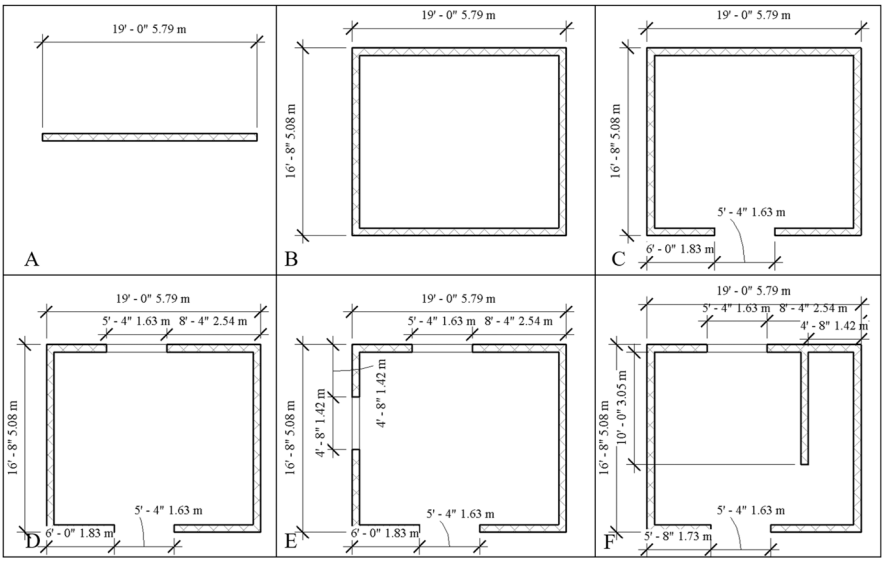

Fig. 7 **A**–**F** Plan view of control room configurations with dimensions

computational intensity of the process further. Finally, room configuration F added a wall to configuration D to show if additional walls are more computationally intense when compared to openings.

Dimensions for each configuration are provided as a 2D plan in Fig. 7 (variable 1 — initial), where the height of each wall is 10 ft (3.048 m). The sill height of the window openings is 2′8″ (0.81 m) and measures 4′8″ (1.42 m) in height by 5′4″ (1.62 m) in length. All door openings are 5′4″ (1.62 m) in length and 7′4″ (2.23 m) in height. These dimensions were chosen to align the openings with the bottom of a course of blocks. Dimensions can vary from what was used in this study (length and width) and the number of openings. However, it was observed that the height of the wall should end on a full course of blocks; otherwise, an additional row was generated.

Additional testing was conducted on each of the room configurations. First, the wall length was increased (variable 2 — increased room size), and we doubled the size of each wall to determine its impact on computational time. All window and door openings remain constant in size. Second, the material that constructed the wall was decreased in size (variable 3 — block size). For this variable, the CMU was replaced with 2–5/8″ × 4″ × 8″ (6.7 cm × 10.2 cm × 20.3 cm) standard brick and used the dimensions for the initial configuration. The final variable was only applied to configurations with an opening (C, D, E, F), where the wall length was doubled, and the openings' length was increased by 150%. The opening was extended an equal amount on both sides. In total, there were 21 configurations to be run by the script. Each variation was assigned a number: initial setup 1, increased wall length 2, block size 3, and increased opening and wall length 4. The wall height was not varied in this study; however, additional testing presented comparable results to varying lengths. Therefore, only the variation in wall length is shown in this study.

The computational modeling script was run for each room configuration and variation, generating the masonry units into their design locations. All models were designed in accordance with the framework developed by this study, which allowed for the interpretation of model geometry. The script identified the locations of partial-sized units and generated a generic model family in the Revit family for each unit. The units were named based on the standardized naming convention for masonry units, making them easily identifiable. To ensure that all blocks were generated within the bounds of the walls, the original wall was not removed automatically, and the generated blocks were superimposed into the existing wall. The model was manually inspected for overlap, gaps, and misplaced blocks. Once complete, the original modeled walls were deleted. Figure 8 shows the result of running the computational modeling scripts for each configuration and variation labeled by room configuration letter and variation number.

Table 3 summarizes the results for each room configuration (A–E). The first row lists the computational time, representing the time taken to generate all units for a room configuration. Total full-sized blocks represent the amount of 8″×8″×16″ (20.3 cm×20.3 cm×40.6 cm) CMU or the amount 2–5/8″×4″×8″ (6.7 cm×10.2 cm×20.3 cm) standard bricks generated by the computational modeler. Total partial-sized blocks are the summation of all partial-sized blocks generated. Additionally, a breakdown by length is provided for each partial-sized block. The dimensions for height and depth are dependent on the wall material. For instance, CMU is 8″×8″ (20.3 cm×20.3 cm) while brick is 2–5/8″×4″ (6.7 cm×10.2 cm)). The last row shows the summation of all full- and partial-sized units for each room configuration, which was verified against the total number of elements in Revit after generation. Additionally, the table provides a reference name and symbol for each configuration; for example, the straight wall is "A," and the initial variable is "1." Therefore, the first configuration is labeled "A1," which is represented as a red circle.

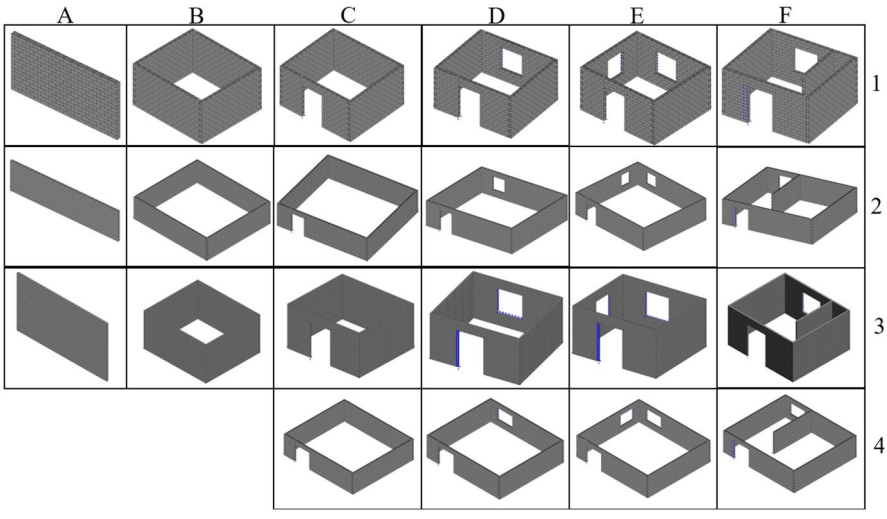

Fig. 8 Room configurations and variations post computational modeling — LOD 400

Table 3 Generated content for each room configuration

Variable	Recorded Values	A. Straight Wall	B. Room - No Opening	C. Room - Door Opening	D. Room- Door and Window Opening	E. Room - Door and Two Window Opening	F. Room - Door, Window Opening, and Wall Intersection
1. Intial	Referance Name/Symbol	A1	B1	C1	D1	E1	F1
	Computational Time (Seconds)	28	72	100	150	148	157
	Total Units Generated	225	780	747	735	699	843
	Full-sized Units Generated	203	750	702	674	630	761
	Partial Sized Units Generated	22	30	45	61	69	82
	12" Length Blocks	7	30	15	23	23	29
	8" Length Blocks	7	0	15	15	23	36
	4" Length Blocks	8	0	15	23	23	17
2. Increased Room Size (Length Doubled)	Referance Name/Symbol	A2	B2	C2	D2	E2	F2
	Computational Time (Seconds)	35	91	143	162	169	174
	Total Units Generated	435	1590	1557	1529	1505	1761
	Full-sized Units Generated	420	1560	1497	1461	1429	1673
	Partial Sized Units Generated	15	30	60	68	76	88
	12" Length Blocks	0	0	0	0	0	21
	8" Length Blocks	15	30	60	68	76	58
	4" Length Blocks	0	0	0	0	0	9
3. Block Size (Standard Brick)	Referance Name/Symbol	A3	B3	C3	D3	E3	F3
	Computational Time (Seconds)	98	290	298	391	361	622
	Total Units Generated	1305	4770	4500	4320	4234	5074
	Full-sized Units Generated	1260	4680	4374	4170	4062	4827
	Partial Sized Units Generated	45	90	126	150	172	247
	4" Length Blocks	45	90	126	150	172	247
4. Increased Room (Wall Length Double) and Opening Size (1.5 x window Length)	Referance Name/Symbol			C4	D4	E4	F4
	Computational Time (Seconds)			150	173	191	209
	Total Units Generated			1533	1489	1467	1725
	Full-sized Units Generated			1473	1421	1369	1629
	Partial Sized Units Generated			60	68	98	96
	14" Length Blocks			0	0	9	0
	12" Length Blocks			0	0	8	29
	10" Length Blocks			0	0	8	0
	8" Length Blocks			60	68	36	50
	6" Length Blocks			0	0	6	0
	4" Length Blocks			0	0	8	17
	2" Length Blocks			0	0	23	0

4.1 Discussion

Reviewing the results showed that increasing the overall number of elements increases the computational time of the model. Figure 9 was developed to show the trends between each variation of a wall configuration. In Fig. 9, each configuration is denoted by a number and color, such as red for the straight wall (A) and green for the room with a door, window, and wall intersection (F). In addition, each variable was assigned a symbol, such as variable one, represented as a circle for the initial configuration. At the same time, three is represented as a triangle representing the block size. Therefore, an orange diamond would represent configuration C2 or the room with a door opening with increased wall lengths. The same reference name and symbols are used here in Table 3. The trendline was created in Microsoft Excel using the least squares method to fit points based on the number of blocks generated for each configuration compared to the computational time. Reviewing each wall configuration shows that most data points are in close proximity to their respective trendlines. For example, all variations for the straight wall configuration (A) appear on the trendline. This also occurs for all data related to configuration C (C1 to C4). Additionally, the smallest configuration, A1, had the masonry units generated in the least amount of time, 28 s. The configuration that took the longest had the most substantial number of overall units to place, which was F3 at 622 s. In addition, all wall configurations constructed out of the brick took the most extended amount of computational time in accordance with how the script was developed.

Additionally, upon review of Table 3, a strong direct correlation between model complexity and the number of elements appears when compared with computational time. Figure 10 was developed to visualize the correlation between computational time and model complexity using the symbols and reference names from Table 3 and Fig. 9.

Figure 10 presents all four variables, each showing a similar trend. This chart shows that while increasing the overall number of elements impacts computational time, it is not the only factor. The trends show that adding additional elements to the model, such as an opening or an additional wall, increased model complexity and computational time. This trend aligns with the methods used for interpreting the model geometry that the computational modeler uses, as the script first places full-sized units in openings, then in the following step, interprets the opening geometry and deletes unnecessary blocks. Additionally, gaps are automatically measured, and partial-sized units are generated around the permitter of the opening as required. However, the results show that computational time is increased if more building elements are generated for a room configuration. Additionally, there is an overlap between the lines for variables 4 and 2 due to the similarities between the computational time and the total number of elements generated.

The method used by this study has limitations, which require further revisions and iterations of the computational modeling script. The script successfully placed all the masonry units for the room configurations developed by this study; however, they were designed and modeled to simplify the computational demands. The script requires the upper and lower edges of the opening to coincide with the edge of a course of blocks. Further refinement to the script would be required to manage edges of openings that do not coincide with the edge of a course of block. Notably, testing all possible wall configurations in this study is impossible. However,

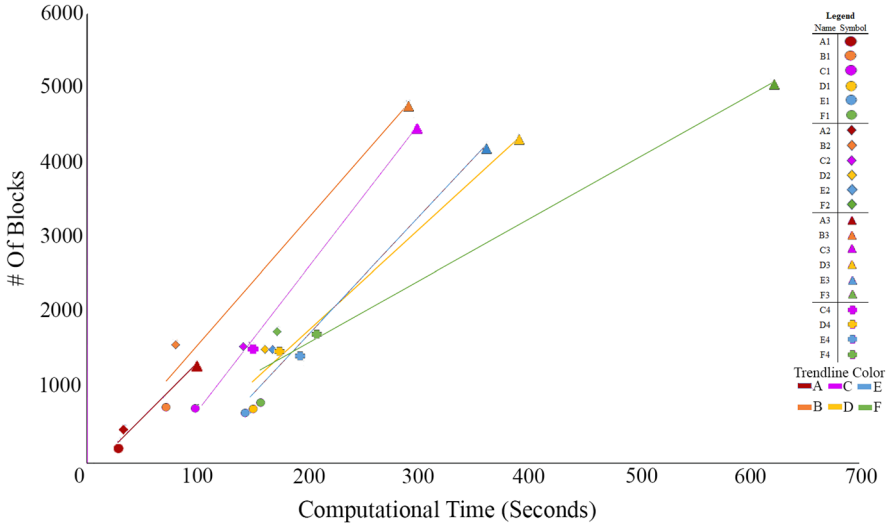

Fig. 9 Impact of increasing number of elements on computational time

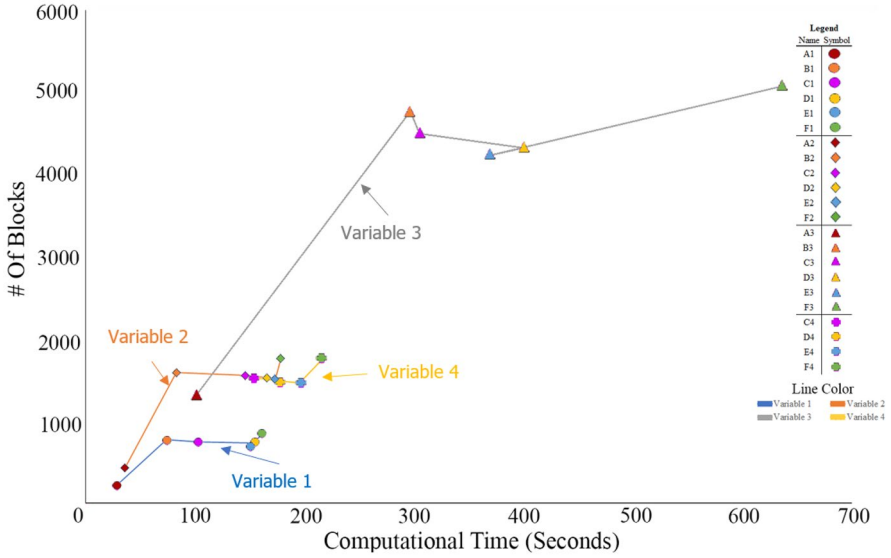

Fig. 10 Number of elements vs. computational time with increased model complexity

additional testing is ongoing to increase the script's capabilities, allowing for additional wall shapes and sizes. Additionally, the script was developed to manage more common bond patterns, such as running or stack bonds. In that case, the script must be adjusted to oversee more complex bond patterns, such as Herringbone or English Bond. Finally, the developed computational modeling script generated the masonry units within the original wall, which was kept in place to ensure no blocks extended beyond their bounds. Once verified, the original wall was removed. However, the removal of the original wall can easily be automated.

Additionally, this system was developed for masonry wall systems and requires adaptation for additional wall systems; therefore, the type of material is a limiting factor. Each building system has its unique criteria for construction, which must be considered to implement computational modeling. While the methods used to set up the project and extract wall geometry are adaptable for other wall systems, the process relies on the dimensions of the masonry units and the construction method. Additionally, the components generated were dry-set and used nominal sizing. The model should include mortar between the units to transition into a more realistic representation, and additional tolerances and specifications must be integrated. The current script was not developed to generate mortar, so the nominal size of the block was considered a three-eighths-inch mortar joint.

However, with the content represented in the model, information can be extracted and developed into tasks for the robot. As mentioned, the data is extracted in the order of component I.D.; however, this does not represent the order in which the wall was constructed. Therefore, the information required to create a construction task for the robot must be sequenced. The methods to automatically sequence the extracted data into robotic construction tasks extend beyond the scope of this

research and remain for future work. Additionally, once the information is extracted and sequenced, it must be converted into tasks for the robot. The methods extend beyond the scope of this study; however, they depend on the robotic system, and the robot must be able to adapt to the tasks according to the dynamic nature of the construction site. Also, regarding interoperability between the BIM and construction robot, the required information format depends on the robotic system and task objective. The information required could vary based on the task; however, this study identified a few constants, which include the location, orientation, material type, and material I.D., which allows for quick identification of any component in the BIM. Therefore, additional work is required to improve interoperability and standardization to enable robotic task execution using information from a BIM.

Despite these limitations, the computational modeling script was run on multiple room configurations. All masonry units were placed correctly for each room configuration, showing the viability of this method for increasing LOD. Notably, the same computer configuration was used to develop the model and run the computational modeling script, which utilized Windows 10 Pro edition, AMD Ryzen 7 3700x, 1 T.B. Samsung SSD, Nvidia GeForce GTX 1070 TI, and 64 G.B. Corsair Vengeance Pro DDR4 3600 RAM. The specifications are noted as different systems could result in variations in computational time. Finally, this study focused primarily on utilizing Autodesk Revit and Dynamo and did not investigate additional BIM authoring software such as Graphisoft Archicad or Bently Microstation. Additional work is required to standardize all software platforms; however, this extends beyond the scope of this study.

5 Conclusion

This study implemented computational modeling to generate masonry units based on the geometry of a masonry room in the BIM to support the extraction of information to create tasks for robotic construction. While this study did not develop the construction tasks, it established the initial steps to generate the information required to facilitate robotic construction in the future. Typically, a masonry wall is represented as a texture in the model; however, utilizing computational modeling, the LOD of the model was increased. As such, the model was enhanced to contain information pertaining to the construction process, such as component location, type, I.D., and orientation. However, to be considered LOD 400, additional components of the wall system would need to be generated, such as lintels or mortar joints. To this end, a framework was developed to present the process undertaken by Dynamo to generate model content, which was divided into three phases: (1) model setup, (2) model development, and (3) information extraction. First, all information was generated and contained within the original model, and only the information required to support the creation of tasks was extracted to reduce the loss of information. Second, the computational modeling script accesses the material content library and imports the correct family into the model based on the material's name. This process could be improved in future iterations of the script. The final challenge identified involved the interpretation of model geometry and increasing the LOD, which was the study's

primary objective. In addition, the computational modeling script developed in this study is available on Git Hub [29].

Future work will further develop the computational modeling script to integrate additional material content libraries and the potential for generating components of additional wall systems. Additionally, there is the potential to integrate optimization strategies into the process, such as having a worker review a plan and reduce the amount of mortar in specific locations so a row of blocks would end at a window opening. This process could be automated to generate and inform workers where the mortar amount could be reduced to minimize the block cut required, providing the opportunity to decrease material usage, lower cost, and reduce potential waste. However, tolerances associated with the mortar joints must still be strictly followed and incorporated. The outlook for utilizing computational modeling and generative design with BIM to develop robotic construction tasks appears promising. In summary, this research indicates that using computational modeling to enhance the LOD in BIM models is viable, and the capabilities aid in developing BIM models that are more precise, intricate, and conducive.

Acknowledgements Thanks to Dr. Yuqing Hu for initial feedback on the development of the Dynamo script.

Author contributions AM: Conceptualization, data collection, methodology, data analysis, writing the first draft. SA.: Conceptualization, data analysis, supervision, editing the draft. RL: Conceptualization, data analysis, supervision, funding, editing the draft.

Funding This material is based on work supported by the National Science Foundation under Grant 1928626.

Data Availability The script used for the is available on Git Hub (Link in the previous paragraph).

Declarations

Competing interests The authors declare no competing interests.

Disclaimer Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. NBIMS (2015) National BIM Standard—United States® Version 3—3 Terms and Definitions. https://www.nationalbimstandard.org/nbims-us-v3

2. Saidi KS, Bock T, Georgoulas C (2016) Robotics in construction. In B. Siciliano & O. Khatib (Eds.), Springer handbook of robotics (pp. 1493–1520). Springer International Publishing. https://doi.org/10.1007/978-3-319-32552-1_57
3. Lasi H, Fettke P, Kemper H-G, Feld T, Hoffmann M (2014) Industry 4.0. Bus Inf Syst Eng 6(4):239–242
4. Sawhney A, Riley M, Irizarry J (2020) Construction 4.0—an innovation platform for the built environment. Routledge. https://doi.org/10.1201/9780429398100
5. Anane W, Iordanova I, Ouellet-Plamondon C (2022) Modular robotic prefabrication of discrete aggregations driven by BIM and computational design. Procedia Computer Science 200:1103–1112. https://doi.org/10.1016/j.procs.2022.01.310
6. Bedrick J, Ikerd W, Reinhardt J (2020) Level of development specification — BIM Forum. https://bimforum.org/resource/level-of-development-specification/
7. Ren R, Zhang J (2021) A new framework to address BIM interoperability in the AEC domain from technical and process dimensions. Advances in Civil Engineering 2021:e8824613. https://doi.org/10.1155/2021/8824613
8. Fu F (2018) Chapter Six—Design and analysis of complex structures. In F. Fu (Ed.), Design and analysis of tall and complex structures (pp. 177–211). Butterworth-Heinemann. https://doi.org/10.1016/B978-0-08-101018-1.00006-X
9. Girardet A, Boton C (2021) A parametric BIM approach to foster bridge project design and analysis. Autom Constr 126:103679. https://doi.org/10.1016/j.autcon.2021.103679
10. McClymonds A, Asadi S, Leicht R (2023) Exploring the challenges of implementing parametric modeling to support robotic construction. Annual Meeting Canadian Society of Civil Engineers. https://par.nsf.gov/biblio/10466066-exploring-challenges-implementing-parametric-modeling-support-robotic-construction
11. McClymonds A, Leicht R, Asadi S (2023) System architecture for supporting BIM to robotic construction integration. In S. Walbridge, M. Nik-Bakht, K. T. W. Ng, M. Shome, M. S. Alam, A. el Damatty, & G. Lovegrove (Eds.), Proceedings of the Canadian Society of Civil Engineering Annual Conference 2021 (pp. 225–236). Springer Nature. https://doi.org/10.1007/978-981-19-0968-9_18
12. Wierzbicki M (2011, October 6) BIM — history and trends. CONVR 2011. https://www.researchgate.net/publication/259390230
13. Kalkan Okur E, Okur F, Altunişik A (2018) Applications and usability of parametric modeling. J Constr Eng Manag Innov 1. https://doi.org/10.31462/jcemi.2018.03139146
14. Brown NC, Jusiega V, Mueller CT (2020) Implementing data-driven parametric building design with a flexible toolbox. Automation in Costruction 118:103252. https://doi.org/10.1016/j.autcon.2020.103252
15. McClymonds A, Asadi S, Wagner A, Leicht RM (2022) *Information exchange for supporting* BIM to robotic construction. 839–848. https://doi.org/10.1061/9780784483961.088
16. Monteiro A (2016, September 29) Visual programming language for creating BIM models with level of development 400. 4th Bim International Conference. https://www.researchgate.net/publication/310606700
17. Davtalab O, Kazemian A, Khoshnevis B (2018) Perspectives on a BIM-integrated software platform for robotic construction through Contour Crafting. Autom Constr 89:13–23. https://doi.org/10.1016/j.autcon.2018.01.006
18. Zhang J, Xing Z (2013) AN IFC-based semantic framework to support BIM content libraries. Proceedings of the 30th International Conference of IT in Construction (CIB W78) (pp. 8–11)
19. Sharif S, Gentry R, Eastman C, Elder J (2015, January 1) Masonry unit database development for BIM-masonry. 12th North American Masonry Conference. https://www.researchgate.net/publication/307213214
20. Kim B, Chin S (2016) Parametric library components for BIM-based curtain wall design automation module. ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, *33*, 1–6. https://www.proquest.com/docview/1823081919/abstract/4918451B7C114B2CPQ/1
21. Venkatraj V, Dixit MK (2022) Challenges in implementing data-driven approaches for building life cycle energy assessment: a review. Renew Sustain Energy Rev 160:112327. https://doi.org/10.1016/j.rser.2022.112327
22. Wu S, Zhang N, Xiang Y, Wu D, Qiao D, Luo X, Lu, W-Z (2022) Automated layout design approach of floor tiles: based on building information modeling (BIM) via parametric design (P.D.) platform. Buildings, 12(2), Article 2. https://doi.org/10.3390/buildings12020250

23. Kim S, Peavy M, Huang P-C, Kim K (2021) Development of BIM-integrated construction robot task planning and simulation system. Autom Constr 127:103720. https://doi.org/10.1016/j.autcon.2021.103720
24. Karimi S, Iordanova I, St-Onge D (2021) An ontology-based approach to data exchanges for robot navigation on construction sites. arXiv Preprint arXiv:2104.10239. https://doi.org/10.36680/j.itcon.2021.029
25. Tibaut A, Rebolj D, Nekrep Perc M (2016) Interoperability requirements for automated manufacturing systems in construction. J Intell Manuf 27(1):251–262. https://doi.org/10.1007/s10845-013-0862-7
26. Anane W, Iordanova I, Ouellet-Plamondon C (2023) Building information modeling (BIM) and robotic manufacturing technological interoperability in construction — a cyclic systematic literature review. Digital Manufacturing Technology 1–29. https://doi.org/10.37256/dmt.3120231856
27. Edirisinghe R, London K (2015) Comparative analysis of international and national level BIM standardization efforts and BIM adoption. Proceedings of the 32nd CIB W78 Conference
28. McClymonds AD, Asadi S, Leicht RM (2024) Development of a parametric modeling method for masonry wall systems to support robotic construction. 398–406. https://doi.org/10.1061/9780784485231.048
29. McClymonds A (2023) Adm5535/computational-modeling-dynamo [Computer software]. https://github.com/adm5535/Computational-Modeling-Dynamo (Original work published 2023)

Authors and Affiliations

Austin D. McClymonds[2] · Somayeh Asadi[1,2] · Robert M. Leicht[2]

✉ Somayeh Asadi
rkn3gr@virginia.edu; sxa51@psu.edu

Austin D. McClymonds
adm5535@psu.edu

Robert M. Leicht
rml@psu.edu

[1] Department of Civil and Environmental Engineering, University of Virginia, Charlottesville, VA, USA

[2] The Pennsylvania State University, University Park, PA, USA