Proceedings of the Design Society, Volume 5: ICED25
https://doi.org/10.1017/pds.2025.10208

Generative AI-powered parametric modeling and BIM for architectural design and visualization

Jaechang Ko ✉, John Ajibefun and Wei Yan

Texas A&M University,USA

✉ jaechangko@tamu.edu

ABSTRACT: Parametric modeling and generative design hold promise for architecture, yet their reliance on scripting and predefined constraints has often discouraged early-stage exploration. This paper proposes a conversational AI framework to address these challenges, integrating ChatGPT into two workflows: user-driven (Revit+Dynamo) and AI-driven (Grasshopper). By transforming natural-language prompts into Python scripts or Grasshopper definitions, designers can iterate on geometry, materials, and forms without extensive coding. AI-based visualization tools such as Veras provide near-instant feedback, accelerating the loop from concept to refinement. Rather than evaluating a single software tool, this exploration highlights collaboration between architect and AI, demonstrating how large language models can augment design intent, expand the parameter space, and adapt to contextual needs.

KEYWORDS: computational design methods, artificial intelligence, visualisation, design automation, LLMs

1. Introduction

Parametric modeling and generative design methods have significantly advanced fields such as mechanical engineering and product development, where performance criteria—such as weight, structural integrity, or efficiency—are precisely defined and quantifiable (Cross, 1982). The deterministic nature of these fields enables systematic, rapid explorations of large design spaces with measurable parameters (Turrin et al., 2011).

Architecture, however, presents a distinctly different context. Buildings must respond to diverse and sometimes intangible factors: cultural context, experiential quality, occupant well-being, and site-specific environmental constraints (Fitch, 2017). Unlike manufactured products, buildings are seldom mass-produced with identical specifications; predetermined standardized parameters are thus atypical. By the time parametric schemas are established in architectural design, foundational decisions have generally been settled upon, diminishing the perceived value of further generative exploration (Gürsel Dino, 2012). Moreover, stakeholders or clients may view the exploration of dozens of parametric variations as an impractical expense unless the outcome offers functional or experiential benefits (American Institute of Architects., 2014). While certain building types—like modular housing or multi-family projects—may favor repeated design modules (Levin, 2015), these cases remain relatively specialized within the broader architectural industry.

Paradoxically, the true promise of generative methods in architecture lies in engaging them at a point where key ideas and fundamental decisions remain fluid (American Institute of Architects., 2014). If parametric modeling and generative experimentation were introduced at the earliest conceptual phases, one could envision a more expansive and meaningful search through the design space. However, the inherent complexity of scripting and managing parametric workflows frequently deters architects from employing generative techniques in these crucial early phases of uncertainty and ideation.

Recent advances in generative artificial intelligence, particularly large language models (LLMs) exemplified by ChatGPT, suggest new possibilities for overcoming these barriers (Ko & Key, 2023).

Conversational AI offers an intuitive method of interaction that allows designers to dynamically identify, define, and iteratively refine design parameters through natural-language exchanges rather than complex coding processes (Makatura et al., 2023).

Building on these insights, this paper proposes a conversational design framework specifically tailored for architectural parametric modeling, demonstrated through two complementary scenarios. The first scenario centers on a user-driven process: ChatGPT is integrated with Dynamo in Autodesk Revit (Yori et al., 2019), enabling architects to incrementally define, combine, and modify design parameters through natural-language exchanges. Once a desired parametric model is established, AI-based visualization tools such as Veras (VERAS, n.d.) provide rapid feedback on materiality and aesthetics within the BIM environment. In the second scenario, we explore a more AI-driven approach. Here, ChatGPT API is embedded directly into Grasshopper for Rhinoceros (Saremi et al., 2017), allowing the system to autonomously generate key parameters—informed by the user's high-level design intent—and produce the resulting 3D geometry. Visualization again uses Veras, but in this workflow, the designer's role shifts to guiding and evaluating the AI-suggested shapes rather than directly specifying each parameter.

2. Background and literature review

2.1. Architectural visualization

Architectural visualization has traditionally posed significant challenges for designers. Before the advent of digital platforms, drawings and physical models were the only means of conveying spatial ideas—both labor-intensive and inflexible for exploring multiple options (Yildirim & Yavuz, 2012). Rendering tools such as V-Ray, Maxwell, and Lumion revolutionized this process by enabling high-fidelity images with sophisticated lighting and material effects (Peddie & Peddie, 2019). Yet these programs often carry steep hardware requirements and can consume days of computation to produce final outputs, making rapid iteration difficult. Additionally, mastering their numerous parameters—textures, reflections, and other fine-grained details—requires substantial training (Miller et al., 2004). Even with more user-friendly systems like Enscape, high-quality results still depend on the architect's careful tuning of numerous variables (Leitão et al., 2019).

Recent progress in AI has transformed visualization further, reducing both the expertise and effort previously required. Tools like Veras or Midjourney employ generative AI AI (Ghimire et al., 2023) to produce conceptual images rapidly, allowing architects to iterate aesthetic decisions in tandem with spatial modeling (Borji, 2022). Visualization can be a dynamic partner in exploring form and appearance from the earliest phases beyond a purely downstream process in the conventional architectural design process (Jo et al., 2024). By generating multiple styles, moods, or material impressions on demand, AI-driven solutions promise not only to streamline rendering workflows but to redefine how architects conceive and communicate design concepts.

2.2. Parametric environments and AI-enhanced applications

Over the past two decades, parametric modeling platforms (e.g., Rhino+Grasshopper, Autodesk Revit+Dynamo, Blender, Houdini, SketchUp, and AutoCAD) have become indispensable in architectural practice. They offer node-based or scripted methodologies to generate complex geometries and automate repetitive tasks, all while maintaining design flexibility.

Recent advancements in computational design have further fueled this ecosystem with AI-enhanced applications aimed at reducing the technical overhead often associated with parametric workflows.(Gürsel Dino, 2012). Examples include Finch3D (Optimizing Architecture, 2020), which expedites early space planning by generating rectilinear layouts under user-defined constraints, Hypar (Hypar, n.d.), which employs cloud-based "functions" to automate building layouts or repetitive tasks, or TestFit (TestFit, n.d.), known for configuring multifamily housing or parking arrangements.

However, such solutions, despite being intended to augment architects' autonomy and creativity, risk a paradox: the AI meant to liberate designers can also limit them. While these tools may useful for automating discrete tasks, they frequently rely on predefined design logics that can inadvertently constrain a design (Bolojan, 2022), (Ko & Key, 2023). Moreover, with technology advancing at such a rapid pace, a static catalog of AI tools risks quickly becoming outdated. New domain-specific plugins are constantly emerging to address particular goals (e.g., architecture-ai (community builder, n.d.)). Next-generation models like Gemini (Team et al., 2023) promise ever more advanced generative capabilities. However, it might not the architects and designers' duty to track every new release or adapt to fleeting trends. A more fundamental question arises: *how can these tools truly empower architects as decision-makers, freeing them from repetitive tasks such as modeling or rendering, and supporting creative, context-sensitive judgments instead?* Focusing on how AI can enhance designers' capacity to conceive, evaluate, and refine architecture—rather than merely automate it—may thus be key to sustaining genuine innovation in this evolving landscape.

2.3. Reconceptualizing parametric workflows with AI

Historically, effective utilization of parametric modeling tools has required significant proficiency in programming languages such as Python or C#, resulting in a steep learning curve that often discourages architects and designers. Moreover, the complexity inherent in these tools, coupled with the frequent necessity of code adjustments and debugging, can impede the spontaneous creative exploration that is essential during early design stages (Ko et al., 2023). Updating parameters and models to meet evolving design needs is often time-intensive.

Future tools must prioritize intuitive interfaces, reducing reliance on coding and fostering seamless integration with design workflows (Schnabel, 2007). Recent developments in LLMs, exemplified by ChatGPT, suggest a potential paradigm shift in addressing long-standing challenges associated with parametric modeling and generative design workflows. For instance, Rane present a comprehensive overview of integrating ChatGPT, Bard, and other cutting-edge AI models in diverse design and engineering workflows—ranging from early conceptual studies and structural simulations to project management and construction scheduling (Rane et al., 2023). In a related effort, Lamaakal showed how LLMs can translate plain-language prompts into functional Grasshopper scripts to build a parametric modeling (Lamaakal et al., 2025).

However, architectural design seldom unfolds as a single, discrete task but rather as a confluence of interrelated activities: collaboration, iterative problem-solving, documentation, code generation, and more. Accordingly, it is not sufficient to consider how AI might optimize a single aspect of workflow in isolation. Instead, a broader perspective is required—one that situates generative tools within the full spectrum of architectural design, where multiple parameters, stakeholders, and objectives intersect (Ko et al., 2023). Hence, rather than viewing generative AI solely as a tool for scripting or code automation, this paper contends that successful integration must embrace the inherently interconnected nature of architectural work

3. Methodology

This study embeds conversational AI within widely adopted parametric modeling tools to expand architects' autonomy, allowing them to control building forms and components via natural language. ChatGPT was chosen for its stable API, broad user community, and robust developer ecosystem, reducing integration hurdles and supporting iterative, text-based design exploration. For the parametric modeling, Dynamo in Autodesk Revit and Grasshopper in Rhinoceros 3D were selected because they are long-established standards in architectural practice. Lastly, Veras was employed for near-instant AI-based visualization, rapidly converting geometry into conceptual or stylized imagery.

Two complementary scenarios have been developed (Fig. 1). The first scenario, ***User-Driven Parameters (ChatGPT + Dynamo)***, shows the approach in Autodesk Revit for Building Information Modeling (BIM). When the user (designer) provides natural-language prompts, ChatGPT translates these into Dynamo-compatible scripts, enabling an iterative process of prompt generation, debugging, parameter refinement, and ultimate visualization—all within the same architectural design task.

The second scenario, ***AI-Driven Parameters (ChatGPT API + Grasshopper)***, places ChatGPT in a more proactive, generative role. Here, Rhinoceros 3D and Grasshopper serve as the parametric environment, and ChatGPT integrates directly via an API. Rather than manually specifying parameters or scripting logic, the designer conveys broad design intentions—leading ChatGPT to infer core geometric principles and generate or adapt Grasshopper definitions with minimal human intervention.

In both scenarios, Veras provides rapid rendering or visualization, ensuring that parameter changes translate swiftly into tangible modifications of the built form.

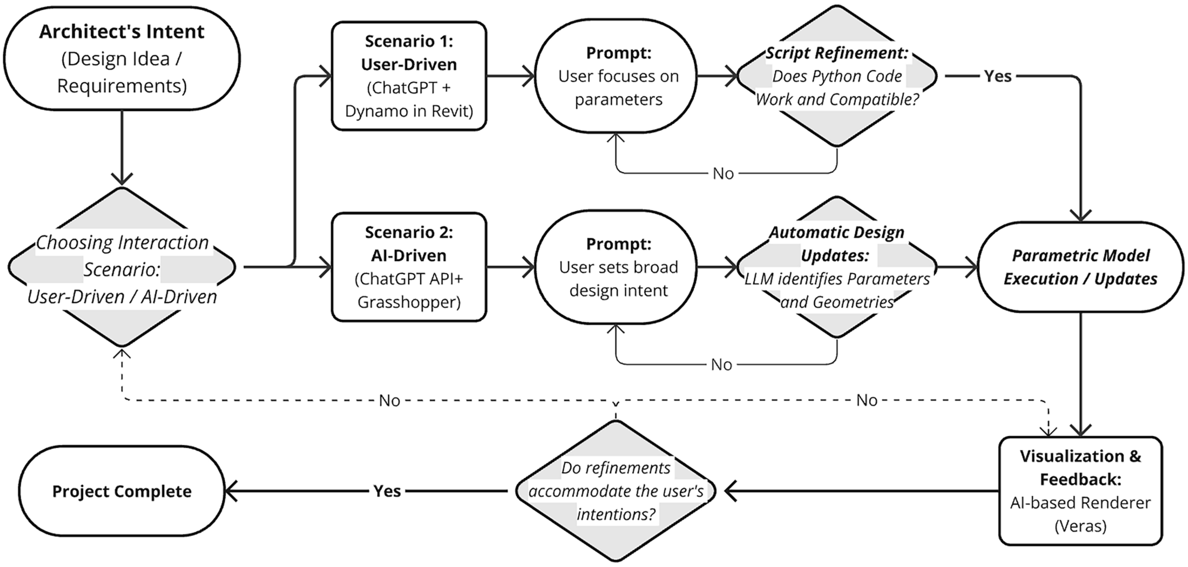

Architect's Intent
(Design Idea / Requirements)
Choosing Interaction Scenario: User-Driven / AI-Driven
Scenario 1: User-Driven (ChatGPT + Dynamo in Revit)
Prompt: User focuses on parameters
Script Refinement: Does Python Code Work and Compatible?
Yes
No
Scenario 2: AI-Driven (ChatGPT API+ Grasshopper)
Prompt: User sets broad design intent
Automatic Design Updates: LLM identifies Parameters and Geometries
No
Parametric Model Execution / Updates
No
No
Visualization & Feedback: AI-based Renderer (Veras)
Do refinements accommodate the user's intentions?
Yes
Project Complete

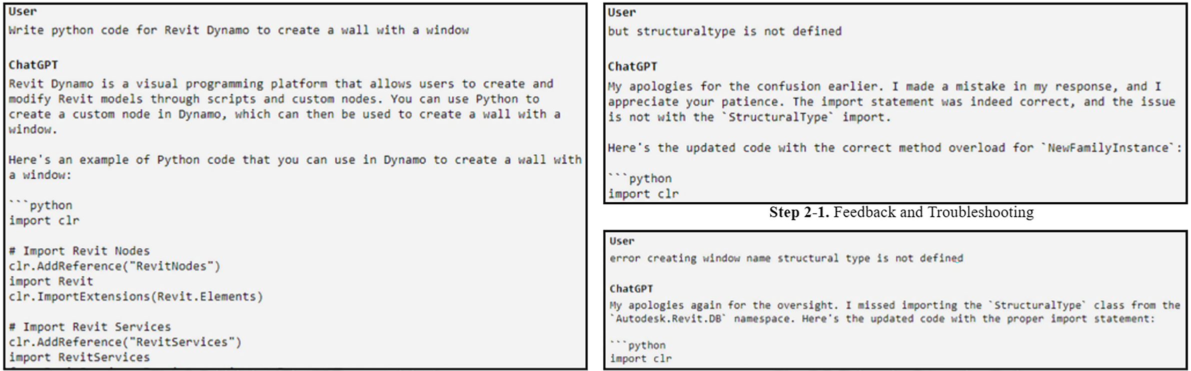
Figure 1. Flowchart for conversational AI in parametric design

4. Results and discussions

4.1. User-driven parameters (ChatGPT + Dynamo)

This section explores the use of ChatGPT (version 4.0) and Veras in the BIM Dynamo design process. ChatGPT enables an interactive design workflow, while Veras provides feedback for refining design solutions

The process begins with defining the design task. ChatGPT generates Python code for Dynamo based on user prompts. The code is tested and refined as needed, ensuring the design process functions as intended. Once the desired outcome is achieved, the interaction with ChatGPT concludes. The framework highlights how ChatGPT interprets architectural intentions, converts them into actionable instructions, and produces a BIM model.



~~~
User
Write python code for Revit Dynamo to create a wall with a window

ChatGPT
Revit Dynamo is a visual programming platform that allows users to create and
modify Revit models through scripts and custom nodes. You can use Python to
create a custom node in Dynamo, which can then be used to create a wall with a
window.

Here's an example of Python code that you can use in Dynamo to create a wall with
a window:

```python
import clr

# Import Revit Nodes
clr.AddReference("RevitNodes")
import Revit
clr.ImportExtensions(Revit.Elements)

# Import Revit Services
clr.AddReference("RevitServices")
import RevitServices
~~~

Step 1. Initial chat (ChatGPT exported data)

~~~
User
but structuraltype is not defined

ChatGPT
My apologies for the confusion earlier. I made a mistake in my response, and I
appreciate your patience. The import statement was indeed correct, and the issue
is not with the `StructuralType` import.

Here's the updated code with the correct method overload for `NewFamilyInstance`:

```python
import clr
~~~

Step 2-1. Feedback and Troubleshooting

~~~
User
error creating window name structural type is not defined

ChatGPT
My apologies again for the oversight. I missed importing the `StructuralType` class from the
`Autodesk.Revit.DB` namespace. Here's the updated code with the proper import statement:

```python
import clr
~~~

Step 2-2. Debugging Interaction Cycle

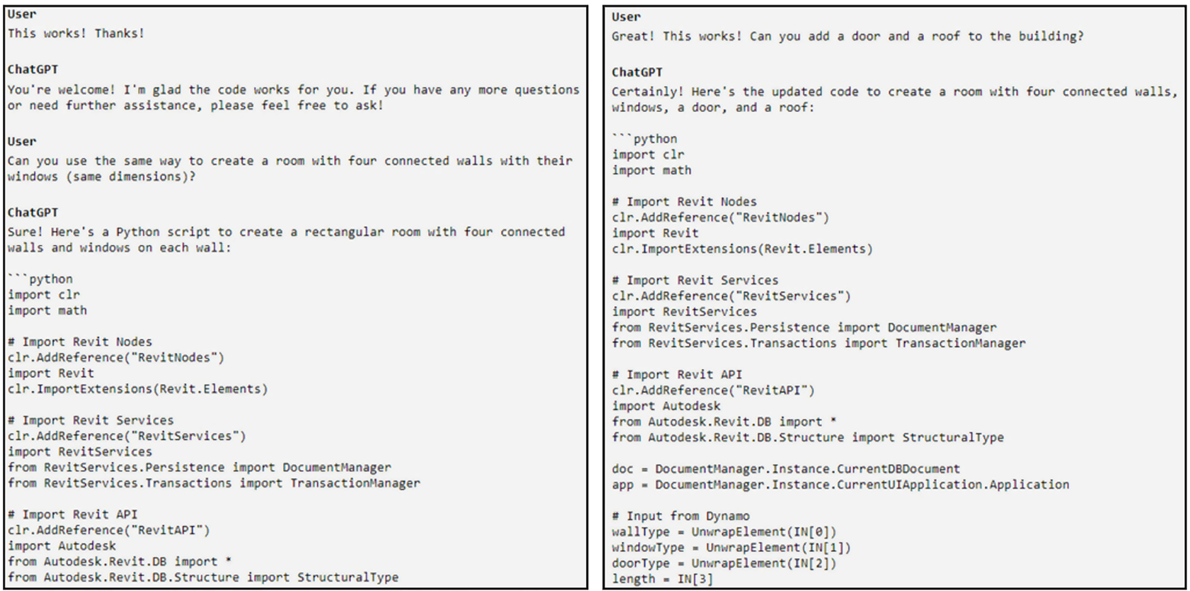
Figure 2. Initial query (Step1, left) and troubleshooting (Step2, right)

- Step 1. Initial Query and User Input Users start by entering design prompts, ranging from general ideas to specific requirements. The task begins with creating a wall and window in Dynamo (Step 1, Fig. 2).
- Step 2. User Feedback and Troubleshooting Verifying and refining the Python code is critical. Errors encountered during execution in Dynamo prompted iterative debugging. The user copied the error messages from Dynamo into ChatGPT, which provided revised code. This process continued until all issues were resolved (Steps 2-1, 2-2, Fig. 2). ChatGPT generated the complete Python code and resolved all debugging issues, except for the manual input of error messages into the chat. Future integration of ChatGPT via an API could automate this process, enabling direct interaction between Dynamo and ChatGPT for seamless troubleshooting.
- Step 3. Chat for Creating BIM of a Simple Example Design The process progressed incrementally, starting with a wall and window, then creating a room with four connected walls, and finally

adding a roof and door. Each step built upon the previous outcomes, using ChatGPT-generated Python nodes. Step 3 in Fig. 3 illustrates the creation of the room following the initial wall and window task.

- Step 4. Interactive Developments

ChatGPT demonstrated flexibility in adding new elements to the model. For instance, a request to add a door and roof to the room yielded updated Python code integrating these features. This adaptability underscores ChatGPT's capability to respond dynamically to evolving design needs, supporting an iterative and user-driven design process (Step 4, Fig. 3).



~~~
User
This works! Thanks!

ChatGPT
You're welcome! I'm glad the code works for you. If you have any more questions
or need further assistance, please feel free to ask!

User
Can you use the same way to create a room with four connected walls with their
windows (same dimensions)?

ChatGPT
Sure! Here's a Python script to create a rectangular room with four connected
walls and windows on each wall:

```python
import clr
import math

# Import Revit Nodes
clr.AddReference("RevitNodes")
import Revit
clr.ImportExtensions(Revit.Elements)

# Import Revit Services
clr.AddReference("RevitServices")
import RevitServices
from RevitServices.Persistence import DocumentManager
from RevitServices.Transactions import TransactionManager

# Import Revit API
clr.AddReference("RevitAPI")
import Autodesk
from Autodesk.Revit.DB import *
from Autodesk.Revit.DB.Structure import StructuralType
~~~

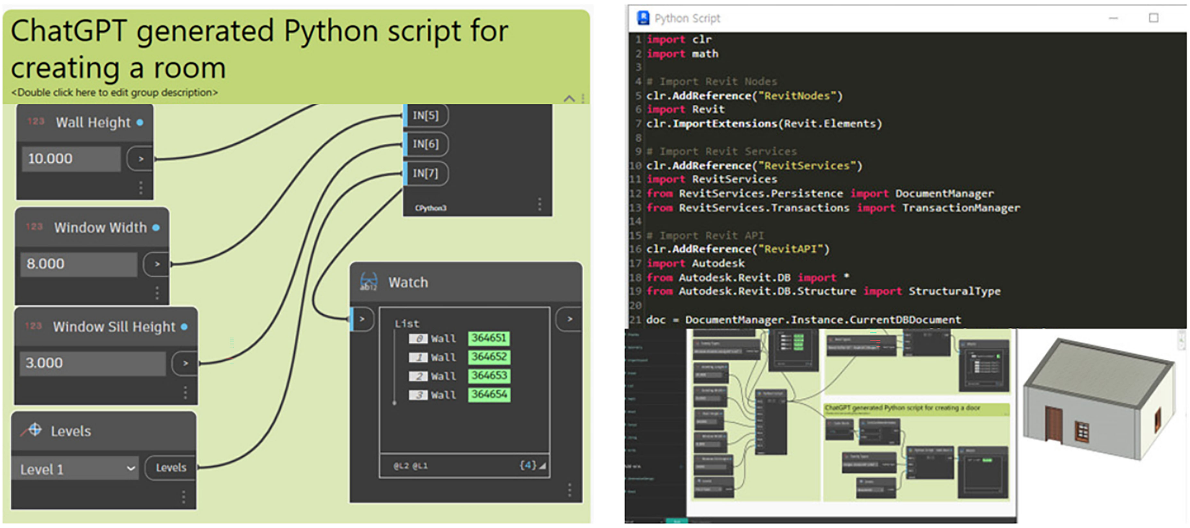
Step 3. The user asked ChatGPT to create a room, and ChatGPT responded.
(Data exported from ChatGPT)

~~~
User
Great! This works! Can you add a door and a roof to the building?

ChatGPT
Certainly! Here's the updated code to create a room with four connected walls,
windows, a door, and a roof:

```python
import clr
import math

# Import Revit Nodes
clr.AddReference("RevitNodes")
import Revit
clr.ImportExtensions(Revit.Elements)

# Import Revit Services
clr.AddReference("RevitServices")
import RevitServices
from RevitServices.Persistence import DocumentManager
from RevitServices.Transactions import TransactionManager

# Import Revit API
clr.AddReference("RevitAPI")
import Autodesk
from Autodesk.Revit.DB import *
from Autodesk.Revit.DB.Structure import StructuralType

doc = DocumentManager.Instance.CurrentDBDocument
app = DocumentManager.Instance.CurrentUIApplication.Application

# Input from Dynamo
wallType = UnwrapElement(IN[0])
windowType = UnwrapElement(IN[1])
doorType = UnwrapElement(IN[2])
length = IN[3]
~~~

Step 4. Building Expansion and Core Updates

Figure 3. Initial design process (Step3, left) and design refinement (Step4, right)

This demonstrates ChatGPT's flexibility and adaptability to the evolving design needs and goals of the user. Thus, ChatGPT plays a significant role in the iterative and dynamic process of building design, adjusting its responses and code based on user input and feedback.



```
import clr
import math

# Import Revit Nodes
clr.AddReference("RevitNodes")
import Revit
clr.ImportExtensions(Revit.Elements)

# Import Revit Services
clr.AddReference("RevitServices")
import RevitServices
from RevitServices.Persistence import DocumentManager
from RevitServices.Transactions import TransactionManager

# Import Revit API
clr.AddReference("RevitAPI")
import Autodesk
from Autodesk.Revit.DB import *
from Autodesk.Revit.DB.Structure import StructuralType

doc = DocumentManager.Instance.CurrentDBDocument
```

Figure 4. Final script and BIM models

Step 5. Implementation and Fine Tuning

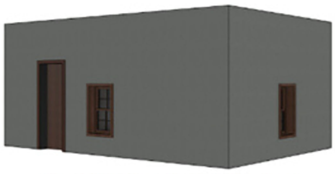
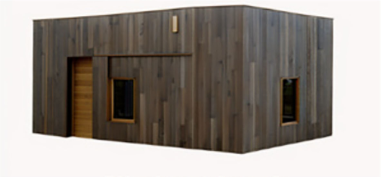
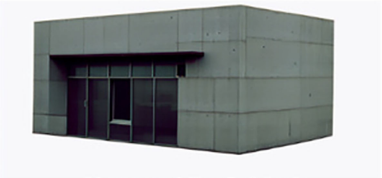
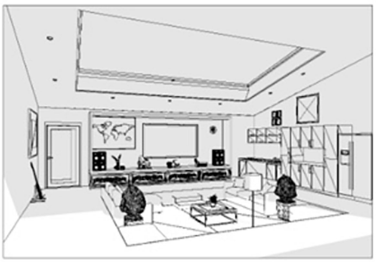
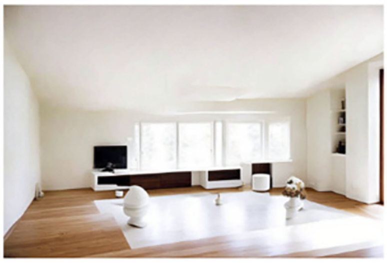
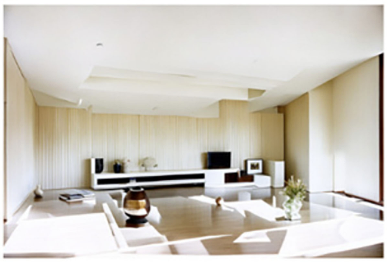
Each Python script generated by ChatGPT was implemented in Dynamo through a Python Script node. Successful execution required users to connect the appropriate input and output nodes, aligning with the script's requirements. Inputs such as wallType, windowType, dimensions, and level were specified, with data types and conditions adjusted accordingly (Fig. 4).

The finalized script produced the desired BIM models, illustrating ChatGPT's potential to facilitate architectural design. The interaction showcases how AI can assist users in exploring and realizing their design ideas efficiently and effectively (right bottom, Fig. 4).

4.2. Integrating Veras for design ideation and visualization

Following Step 5 of the previous chapter, a simple box-shaped model was produced, featuring four walls (each with a window), a flat roof, and a single door (see three-rendering images at top in Table 1). In this section, we explore how Veras, an AI-powered visualization add-in, integrates with the ChatGPT + BIM workflow to further refine and visualize this basic model.

Table 1. Based on the views from BIM (left column), and the user's prompts, Veras generated different designs' visualization (middle and right columns)

Original Revit Model - Exterior	Ext_Rendering A	Ext_Rendering B
		
Revit 3D View Screen Capture	*"Wooden retail store"*	*"Concrete Office Building"*
Original Revit Model - Interior	**Rendering C**	**Rendering D**
		
Revit 3D View Screen Capture	*"Interior contemporary living room with white paint and wood as major material"*	

Starting with a basic box-shaped model, Veras allows users to specify materials and building purposes through simple prompts. For example, Rendering A ("Wooden retail store") introduced unique wood patterns and wall variations, while Rendering B ("Concrete office building") featured dark glazing and setback openings (Table 1).

Interior spaces also benefit from Veras' feedback. Renderings C and D responded to prompts like "Interior contemporary living room with white paint and wood as major materials," suggesting changes such as filling sunken spaces or modifying skylights based on ceiling materials (Table 1).

Veras facilitates both exterior and interior design ideation by providing actionable visual feedback. By leveraging its AI-driven visualizations, users can explore a wide range of design possibilities while refining their initial concepts in a more intuitive and efficient manner.

4.3. AI-driven parameters (ChatGPT API + Grasshopper)

This section explores the integration of the ChatGPT API with Rhino Grasshopper for generating 3D architectural models via natural language input. The focus is on automating workflows and refining prompt-based design processes.

- Step 1. Import ChatGPT API in Rhino GrasshopperA trigger component and data recorder automated iterative design by sending API requests at intervals (e.g., every 2 seconds). The script generated new coordinates in each cycle, creating geometry dynamically.
- Step 2. API as ComponentsGeometries were generated directly from the API, avoiding manual intervention. Initial prompts produced text-heavy outputs, which were refined with instructions like "No text, only equations."

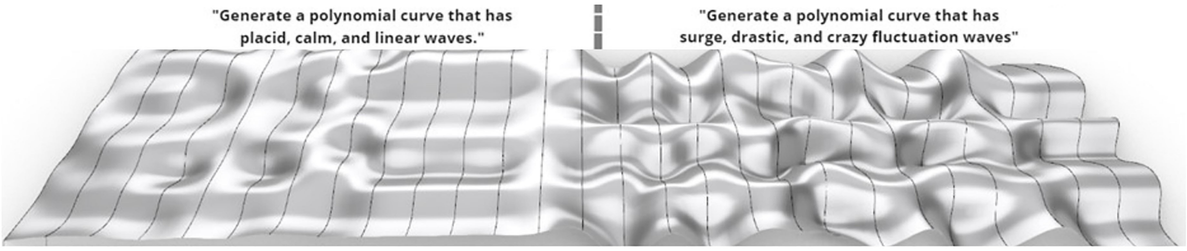
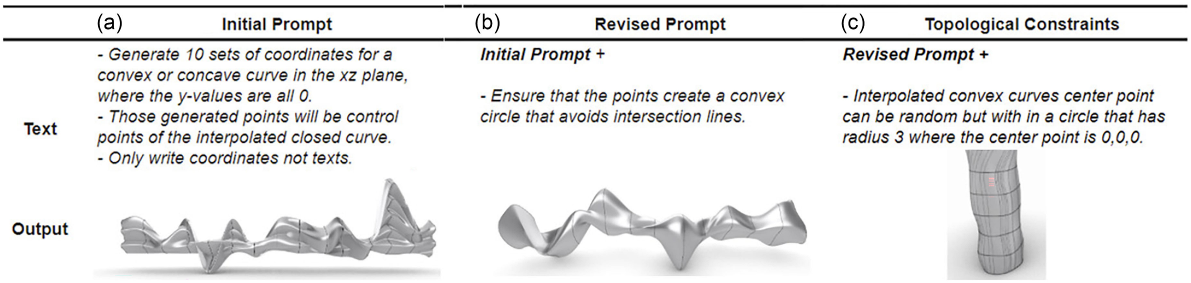

"Generate a polynomial curve that has placid, calm, and linear waves."
"Generate a polynomial curve that has surge, drastic, and crazy fluctuation waves"

Figure 5. Descriptive language implementation in ChatGPT API

Constraints, such as using sine and cosine functions, ensured consistent polynomial curves. Testing descriptive prompts (e.g., "placid, calm waves" vs. "drastic, fluctuating waves") produced visually distinct curves (Fig. 5), showing ChatGPT's responsiveness to input. This outcome demonstrates that ChatGPT can interpret human language input and generate designs that cater to specific requirements or emotions. It highlights the importance of prompt for unlocking the full potential of ChatGPT-generated architectural design, ultimately expanding the possibilities in design methods.

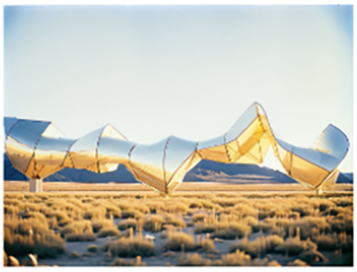
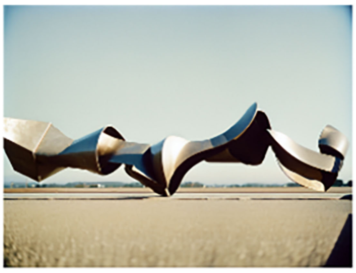
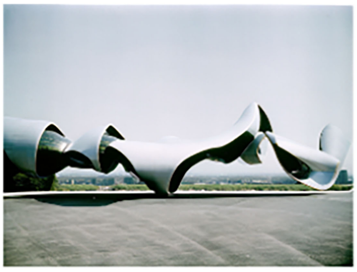
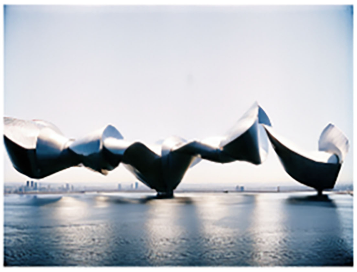
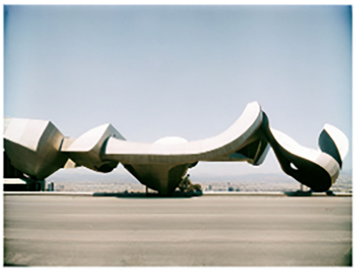
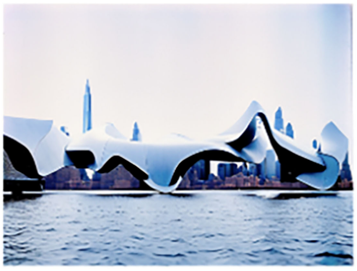
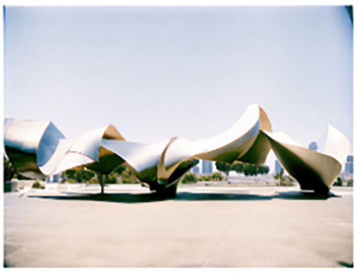
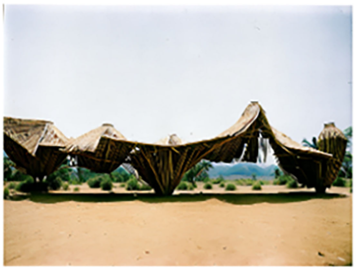
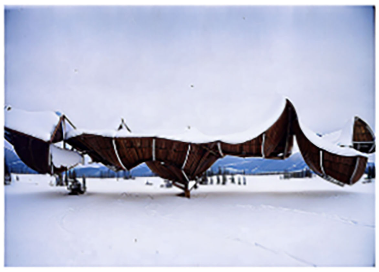
Table 2. Prompt with topological condition



	(a) Initial Prompt	(b) Revised Prompt	(c) Topological Constraints
Text	*- Generate 10 sets of coordinates for a convex or concave curve in the xz plane, where the y-values are all 0.* *- Those generated points will be control points of the interpolated closed curve.* *- Only write coordinates not texts.*	***Initial Prompt +*** *- Ensure that the points create a convex circle that avoids intersection lines.*	***Revised Prompt +*** *- Interpolated convex curves center point can be random but with in a circle that has radius 3 where the center point is 0,0,0.*
Output			

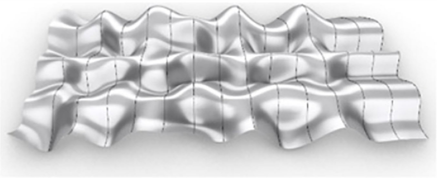
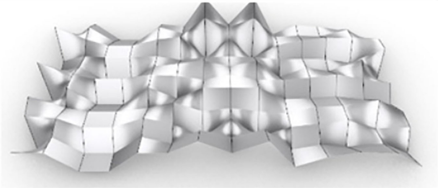
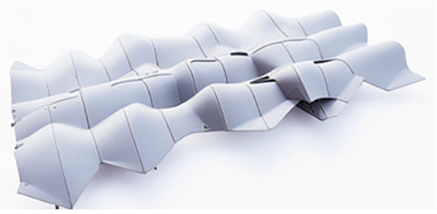
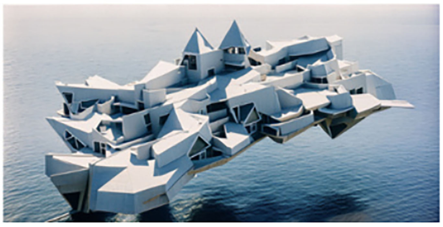
- Step 3. ChatGPT as Generative Design ToolTo create lofted building forms, prompts specified constraints like convexity and non-intersecting lines. Adjustments ensured stable structures, incorporating mandatory elements like a circular framing radius of 6 units and controlled variability within a defined radius of 3 (Table 2). The prompts were refined to include constraints like "ensuring convex" and "avoiding intersecting lines" (Table 2, row b), enabling ChatGPT to generate dynamic, non-intersecting convex or concave shapes effectively.Additional parameters for column generation required curves to include a circle with a radius of 6 units for structural stability, while the center point varied within a radius of 3 (Table 2, row c). This introduced controlled randomness, ensuring designs adhered to practical architectural requirements while maintaining structural integrity.
- Step 4. Veras VisualizationIntegrating Veras enabled rapid exploration of architectural style, materiality, context, and scale. Initially, a building was rendered without user prompts, resulting in a structure with an aluminum skin on flat terrain at sunset, with Veras autonomously adding a structural column to support the lifted mass (rendering A, Table 3). With the prompt "Black metallic building," Veras generated a darker, heavier aesthetic, featuring a thick, abstract skin and a defined entry point at ground level, enhancing structural integrity (rendering B). Inspired by architects like Zaha Hadid and Frank Gehry, prompts tailored to their styles resulted in designs emulating their iconic works (renderings C and D). Locality-specific prompts, such as "Building in downtown Los Angeles," added an LA backdrop to a concrete and metal-skinned structure, while "Zaha Hadid building in Downtown NY" rendered a Manhattan setting with cool tones (renderings E and F). Scale keywords were also tested. The term "pavilion" reduced the scale, focusing on exterior skin and structural details like stair-like openings and support elements (rendering G). Regional and material contexts further demonstrated Veras's adaptability. For example, a bamboo pavilion rendered with "Africa" featured wide spacing for ventilation, while the same prompt in the "Arctic" produced a denser structure with snow accumulation (renderings H and I).These results highlight Veras's ability to generate contextually responsive designs, enabling diverse and iterative exploration of architectural possibilities.

Table 3. Veras renderings for design exploration

Rendering A	Rendering B	Rendering C
		
No prompt	*"Black Metalic Building"*	*"Zaha Hadid Style"*
Rendering D	**Rendering E**	**Rendering F**
		
"Frank Gehry Style"	*"Building in downtown Los Angeles"*	*"Zaha Hadid building in Downtown NY"*
Rendering G	**Rendering H**	**Rendering I**
		
"Pavilion"	*"Pavilion in Africa"*	*"Pavilion in the Arctic"*

- Step 5. Veras Design FeedbackThe ability of Veras to provide immediate visual feedback proved highly effective in the iterative design process, particularly for refining complex geometries generated by the ChatGPT API (Table 4). For instance, when rendering a facade created using the prompt "Generate a polynomial curve that has surge, drastic, and crazy fluctuation waves," Veras translated the design into a shell-shaped roof (middle column, Table 4).While intriguing, the authors aimed for a more angular, protruding form, prompting a revision to the ChatGPT API prompt to include "interpolated curve degree is 0" (right column, Table 4). This adjustment produced a markedly different output, featuring sharp angles and a fragmented, jagged geometry, illustrating the utility of AI-based visualization tools for rapid feedback and iterative design.

Table 4. Comparative visualization of design iterations: basic prompt(left), addition1(middle), and addition2(right)

Basis Prompt	Basis Prompt + Addition 1	Basis Prompt + Addition 2
"Create one polynomial with three variables x, y, and z." *"Do not number the equations"* *"No text, only equations."* *"Use the * operator whenever multiplication occurs."* *"create one trigonometric polynomials with three variables each x, y, and z"* *"Only use sin and cos function not tan."*		
		

Traditionally, architects model building elements and create renderings to visualize and refine designs, repeating this labor-intensive process until achieving satisfactory results. This approach often leads to increased complexity, extended timelines, and potential errors. The integration of ChatGPT, its API, BIM, parametric modeling, and Veras proposed in this study streamlines this process. By generating complex geometric shapes through descriptive prompts and providing immediate visual feedback, Veras reduces the time and effort required for iterative design. This allows designers to focus more on creativity and less on technical scripting.

The study highlights significant advancements in design efficiency and exploration while revealing critical challenges for future research. Generative AI tools currently face limitations in addressing complex, bespoke, or customized designs, which are crucial in architectural practice. Additionally, the physical feasibility of AI-generated designs must be considered, as some outputs, such as Penrose-like constructs (DiPaola et al., 2018), may be visually impressive but structurally impossible. Addressing these challenges is essential for fully realizing AI's potential in architectural design.

5. Conclusions

This paper introduced a conversational AI framework for architectural parametric modeling, demonstrating how LLMs can reduce coding barriers in both user-driven (Revit+Dynamo) and AI-driven (Grasshopper) scenarios. By integrating ChatGPT directly into these workflows—and coupling them with on-demand visualization tools—designers can explore fluid, early-phase experimentation without being bound by rigid parametric schemas. The resulting synergy of natural-language prompts and parametric scripts expands architects’ capacity to iterate on form, material aesthetics, and functional constraints in a single, interactive environment.

Several challenges remain. First, while AI-generated geometry can be compelling, it must incorporate regulatory codes and topological constraints to ensure actual buildability—an area not deeply addressed in the present exploration. Second, LLMs sometimes produce unintended or overly abstract shapes, falling short of precise micro-scale detailing. Third, the study’s conceptual approach—though indicative of potential—has not yet been tested in a comprehensive pilot implementation, thus requiring further empirical validation in real-world contexts. Fourth, concerns about data privacy persist, as many generative AI services run on external platforms. Alternative local-model solutions (e.g., Ollama (Marcondes et al., 2025)) may mitigate these risks by preventing sensitive design data from leaving the architect’s environment. Nonetheless, the foundational premise remains: conversational AI can be a powerful partner in early-stage design, facilitating both broad creative exploration and iterative refinement, provided architects maintain critical oversight of structural, regulatory, and contextual factors.

References

American Institute of Architects. (2014). *The architect's handbook of professional practice (Fifteenth edition.)*. Wiley; WorldCat.org.

Bolojan, D. (2022). *Creative AI: augmenting design potency. Architectural Design*, 92(3), 22–27.

Borji, A. (2022). Generated faces in the wild: Quantitative comparison of stable diffusion, midjourney and dall-e 2. *arXiv Preprint arXiv:2210.00586.*

community builder. (n.d.). Architecture-ai. *Architecture AI.* Retrieved March 11, 2025, from https://chatgpt.com/g/g-BxSRxZ2zs-architecture-ai

Cross, N. (1982). Designerly ways of knowing. *Design Studies*, 3(4), 221–227.

DiPaola, S., Gabora, L., & McCaig, G. (2018). Informing artificial intelligence generative techniques using cognitive theories of human creativity. *Procedia Computer Science*,145, 158–168.

Fitch, J. M. (2017). The aesthetics of function. *In People and buildings* (pp. 3–16). Routledge.

Ghimire, P., Kim, K., & Acharya, M. (2023). Generative AI in the Construction Industry: Opportunities & Challenges (arXiv:2310.04427). *arXiv*. https://doi.org/10.48550/arXiv.2310.04427

Gürsel Dino, I. (2012). *Creative design exploration by parametric generative systems in architecture*.

Hypar. (n.d.). Retrieved March 11, 2025, from https://hypar.io/

Jo, H., Lee, J.-K., Lee, Y.-C., & Choo, S. (2024). Generative artificial intelligence and building design: Early photorealistic render visualization of façades using local identity-trained models. *Journal of Computational Design and Engineering*, 11(2), 85–105.

Ko, J., Ennemoser, B., Yoo, W., Yan, W., & Clayton, M. J. (2023). Architectural spatial layout planning using artificial intelligence. *Automation in Construction*, 154, 105019. https://doi.org/10.1016/j.autcon.2023.105019

Ko, J., & Key, S. (2023). Integrating AI/ML Techniques in Parametric Modeling for Creative Design. *Proceedings of the 14th international conference on computational creativity*, 407–411.

Lamaakal, I., Maleh, Y., El Makkaoui, K., Ouahbi, I., Pławiak, P., Alfarraj, O., Almousa, M., & Abd El-Latif, A. A. (2025). Tiny Language Models for Automation and Control: Overview, Potential Applications, and Future Research Directions. *Sensors*, 25(5), 1318.

Leitão, A., Castelo-Branco, R., & Santos, G. (2019). Game of renders: The use of game engines for architectural visualization. *Proceedings of the 24th CAADRIA Conference*, 1, 655–664.

Levin, M. S. (2015). *Modular system design and evaluation. Springer.*

Makatura, L., Foshey, M., Wang, B., HähnLein, F., Ma, P., Deng, B., Tjandrasuwita, M., Spielberg, A., Owens, C. E., Chen, P. Y., & others. (2023). How can large language models help humans in design and manufacturing? *arXiv Preprint arXiv:2307.14377.*

Marcondes, F. S., Gala, A., Magalhães, R., Perez de Britto, F., Durães, D., & Novais, P. (2025). Using Ollama. *In Natural Language Analytics with Generative Large-Language Models: A Practical Approach with Ollama and Open-Source LLMs* (pp. 23–35). Springer.

Miller, K., Przybyla, D., & Pegah, M. (2004). Soothing the many-headed beast: Improving the rendering process in the world of 3D animation. *Proceedings of the 32nd Annual ACM SIGUCCS Conference on User Services*, 381–384.

Optimizing Architecture. (2020). https://www.finch3d.com/

Peddie, J., & Peddie, J. (2019). Ray-Tracing Programs and Plug-ins. *Ray Tracing: A Tool for All*, 181–330.

Rane, N., Choudhary, S., & Rane, J. (2023). Integrating ChatGPT, Bard, and leading-edge generative artificial intelligence in architectural design and engineering: Applications, framework, and challenges. *Framework, and Challenges* (November 27, 2023).

Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software*, 105, 30–47.

Schnabel, M. A. (2007). Parametric designing in architecture. *Computer-Aided Architectural Design Futures (CAADFutures) 2007: Proceedings of the 12th International CAADFutures Conference*, 237–250.

Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., & others. (2023). Gemini: A family of highly capable multimodal models. *arXiv Preprint arXiv:2312.11805.*

TestFit. (n.d.). Retrieved March 11, 2025, from https://www.testfit.io/

Turrin, M., Von Buelow, P., & Stouffs, R. (2011). Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. *Advanced Engineering Informatics*, 25(4), 656–675.

VERAS. (n.d.). EvolveLAB. Retrieved April 30, 2023, from https://www.evolvelab.io/veras

Yildirim, T., & Yavuz, A. O. (2012). Comparison of traditional and digital visualization technologies in architectural design education. *Procedia-Social and Behavioral Sciences*, 51, 69–73.

Yori, R., Kim, M., & Kirby, L. (2019). *Mastering Autodesk Revit 2020.* John Wiley & Sons.