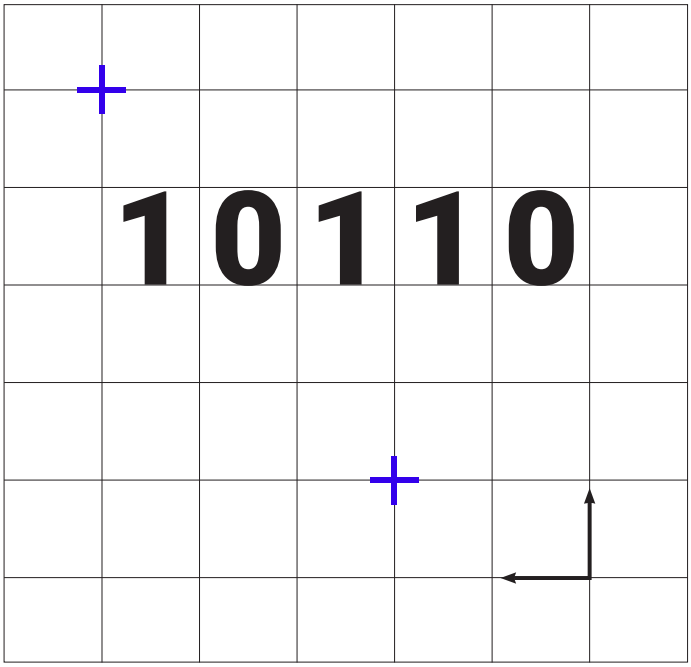
01.

How to Cite: Kawasaki, J. Y., Hirsekorn, Y., Ansre, N., & Grunwald, G. (2024). Possibilities and limits of AI-supported design processes in architecture. In H. R. Husain (Ed.), *AI-driven architecture: Pioneering the digital frontier* (pp. 01-22). Alanya University. https://doi.org/10.38027/AI-Driven-1

CHAPTER 01

Possibilities and limits of AI-supported design processes In architecture

M.A. Jan Yoshio Kawasaki, Yara Hirsekorn,
Nik Ansre and Prof. Dr. Ing. Gregor Grunwald

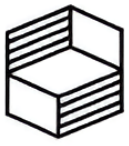


Introduction

The construction industry has been undergoing profound change for several decades as a result of advancing digitalization. This transformation also has a significant impact on the requirements and content of academic courses specializing in construction planning and civil engineering. Where analogue processes like hand drawings once prevailed, digital design methods, including Building Information Modelling (BIM), now integrate a range of digital tools and technologies, increasingly influencing the work of architects. ***(Hanke, 2024)*** In order to meet the challenges and opportunities of new digital technologies, it is necessary to develop innovative training formats and integrate them into existing curricula.

With this in mind, Jade University has designed a comprehensive range of learning opportunities in the form of modular courses, so-called knowledge nuggets. These knowledge nuggets focus on digital planning and are offered in a browser-based format for self-study. The project, which operates under the name "AUFLADEN", aims to create a comprehensive knowledge repository for students that reflects the requirements of the modern construction industry. The name "AUFLADEN" symbolizes the preparation and communication of new knowledge in the digital age.

The project has a duration of 2 years and one month and is funded by the "Stiftung Innovation in der Hochschullehre". A central topic of the project is the implementation and use of artificial intelligence (AI) and its more concrete version, a Large Language Model (LLM) like Chat-GPT 4, are increasingly used in the planning process. Large language models are artificial intelligence models that have been trained on huge amounts of text. They can generate human-like text, answer questions, translate, and much more ***(Brown et. al. 2020)***. In addition, the transfer of data obtained through AI-supported planning to production is being investigated using augmented reality (AR). In particular, the application of AR in the production of complex molds offers promising opportunities, but also specific challenges, which are analysed and documented in detail as part of this project. AI is a new and indispensable topic in research and studying at universities that requires new training formats for existing curricula ***(Reinmann & Watanabe, 2024; Grunwald et. al, 2022)***. This article highlights the practical application as well as the potentials and limitations of these technologies in the construction industry.



Stiftung
Innovation in der
Hochschullehre

Figure 01: left: Logo of the research project – Source: own figure / right: project funding institution - Source: Stiftung Innovation in der Hochschullehre

The “AUFLADEN” project

The AUFLADEN project stands for the education and training laboratory for digital planning and construction. As part of this project, an innovative platform was developed that is easily accessible via an internet browser ***(Kawasaki et al., 2023)***. This platform offers specially prepared knowledge content in the field of Building Information Modeling (BIM), created by students for students. The project is funded by the Foundation for Innovation in University Teaching and has been running at the Department of Architecture at Jade University in Oldenburg since October 2022. It develops courses that are offered under the name “BIM Game” ***(Grunwald & Heins, 2022; 2024).*** AUFLADEN is a central point of contact for students and all interested parties to familiarize themselves with the basics and advanced applications of BIM and to further their education in this forward-looking field. At the heart of the platform are the so-called “knowledge nuggets”. These knowledge nuggets consist of a variety of learning materials, including video tutorials, interactive click tutorials, practical tasks and quizzes. Each knowledge nugget focuses on a specific topic, use case or software solution in the field of BIM and thus offers a comprehensive and practical learning experience.

The AUFLADEN platform not only serves as a supportive learning tool, but also provides users with an individual and flexible learning experience. The combination of different didactic approaches ensures that the content covers both theoretical knowledge and practical application possibilities. This promotes a comprehensive understanding and competent application of digital technologies in the construction industry. The interactive and user-optimized design of the platform enables sustainable and effective learning, giving students a significant advantage in their professional development.
Alongside the learning experience in the AUFLADEN project, the exploration of topics like augmented reality (AR) ***(Hanke et al., 2023; Kawasaki & Grunwald, 2023),*** robotic process automation (RPA) ***(Heins et al., 2021; 2024),*** and the creation of digital twins has been carried out.

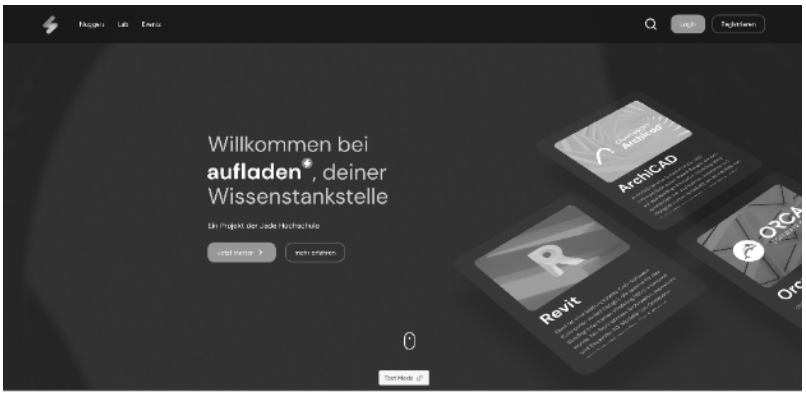


Figure 2. AUFLADEN Website - Source: www.wissen-aufladen.de

Development of the research question

In addition to the prepared knowledge content, the project offers various experimental spaces for learning and teaching digital technologies in the construction industry in the laboratory area, or lab for short. Technologies and use cases, such as the integration of augmented reality in the construction planning process or the use of artificial intelligence, are researched here together with experts and students. In addition to the results, the focus is on research-based learning and teaching. More information on all labs already carried out can also be found on the platform under the dedicated lab section ***(Grunwald, 2024).*** Within the "parametric design" labs, parametric structures in the form of sculptures, furniture or facades were designed together with the students.

The designs were created using the CAD software Rhino in combination with the visual programming interface Grasshopper. During the process, it quickly became clear that, in addition to formulating a design, the technical implementation in the software posed a major challenge. Almost all participants in the lab had no experience with Grasshopper or programming in general. The methodological knowledge for the software was supported in the lab by the AUFLADEN platform and accompanied by technical experts. The prepared click tutorials proved to be particularly helpful for the participants. Nevertheless, the necessary methodological knowledge needs to be tailored to the specific design and the functions in Grasshopper are too complex to be learned to a sufficient extent in just a few seminars. This problem was investigated in a small group within the "parametric design" lab and the approach was developed to create a Python code for Grasshopper with the help of a suitable AI. This could simplify the complex implementation of the idea in the software by describing the design of the AI and returning it in the form of code. The interface for reading Python code into Grasshopper could already be used for complex implementations. For example, ***Zheng (2019)*** implemented iterative pattern design using Python scripts decoded in Grasshopper.

Material and Methods

The primary objectives were to tap into the potential of language models, such as Chat-GPT, in order to make the concept of parametric design within the Rhino Grasshopper software accessible to a broader user base. The focus here was on designing the application in such a way that it can also be used intuitively by people without extensive prior knowledge or in-depth planning. This should significantly lower the barrier to entry into the world of complex design and democratize the creative process.

To achieve this goal and test the effectiveness of the language models in a real-world scenario, a specific question was posed to a number of advanced language models. These models included Edge Copilot, Chat-GPT 3.5, Gemini, Claude 3 and Chat-GPT 4. The selection of these different models made it possible to evaluate a wide range of answers and to analyze the respective strengths and limitations of each model in the context of automated code generation for parametric design. This approach should not only provide information about the current performance of available AI technologies, but also help to drive future developments in the interface between artificial intelligence and design software in a more targeted manner.

The question posed to the language models was as follows:

> „Give me a Rhino Grasshopper code without comments in the code. Line up 20 cubes measuring 10cm wide by 20cm long and 10cm high. Duplicate this row of cubes and stack them on top of the first row of cubes. Repeat this step until 10 rows of cubes are stacked on top of each other. “

The conversations and results of the AIs were anavlyzed and the language models were evaluated in a matrix according to defined criteria:

Requirement	Edge Copilot	Chat-GPT 3.5	Gemini	Claude 3	Chat-GPT 4	Explanation of symbols
Unrestricted availability in Germany	••••	•••••	•••••	•	•••	••••• = Works constantly and without errors
Can understand and generate Python code	••••	••••	••••	•	•••••	•••• = Works partially/ gives results with slight errors
Understands what Rhino is	•••••	•••••	•••••	•	•••••	••• = Works conditionally/ partially correctly with moderate errors
Can understand what Grasshopper and visual-scripting is	••••	••••	•••	•	•••••	•• = Works poorly/ gives few correct answers with serious errors
Can use Grasshopper Python code to generate geometric shapes	••••	••••	••	•	•••••	• = Doesn't work
Can identify and correct errors in the generated Grasshopper Python code that occur	••••	•••	•	•	•••••	
Can use an explanation to describe complex geometric shapes with Grasshopper Python code	••••	•••	••	•	••••	
Can use images to recreate the shapes in Rhino with a Python code	••	••	•	•	•••	
Rating average:	3,875	3,75	2,875	1	4,375	

Table 01. Evaluation of the AI models.

This matrix shows the selection of the optimal AI system for the specific requirements of the project application. After extensive evaluation of various AI models, ChatGPT-4 was found to be the most suitable to fulfill the complex requirements. This decision was based on a number of criteria that were specifically tailored to the needs:

1. Unrestricted availability in Germany: The AI must be usable in Germany without geographical or licensing restrictions.
2. Ability to understand and generate Python code: The AI should be able to effectively interpret and generate Python code, which is a basic requirement for integration into existing systems.
3. Understanding of Rhino software: It is critical that the AI has a solid understanding of Rhino 3D modelling software to interact seamlessly with tools.
4. Use of Grasshopper Python code to generate geometric shapes: The AI should be able to create complex geometric shapes using Python code implemented in Grasshopper.
5. Identify and correct errors in the generated Grasshopper Python code: Another important capability of AI is to identify and correct errors in the generated codes, which is essential for the reliability of the modelling process.
6. Use of explanations to describe complex geometric shapes with Grasshopper Python code: The AI should be able to describe complex geometric structures in a clear and understandable way and explain the relevant scripting methods.
7. Use images to reconstruct the shapes in Rhino with Python code: Finally, the AI should be able to develop Python code based on images that accurately reproduces the desired shapes in Rhino.

These criteria ensure that the selected AI is not only technically competent, but can also be effectively integrated into specific work flow and software environments.

After in-depth analysis of the conversations and results with various AI models, it was determined that ChatGPT-4 best met the requirements of the project application. ChatGPT-4 demonstrated consistent and error-free performance across several key areas. For example, it was able to understand and generate Python code, which is a fundamental requirement for integration into existing systems. Additionally, ChatGPT-4 demonstrated a solid understanding of Rhino software as well as Grasshopper and visual scripting, which is critical to interacting seamlessly with tools.

When evaluating the different AI models against predefined criteria, ChatGPT-4 was found to perform superior in several areas. For example, it proved to be highly reliable in applying Grasshopper Python code to generate geometric shapes and in identifying and correcting errors in the generated code. In addition, ChatGPT-4 was also able to provide explanations to describe complex geometric shapes using Grasshopper Python code, albeit not without some minor errors.

One challenge was the use of images to reconstruct the shapes in Rhino with Python code. Here, ChatGPT-4 showed partially correct results, but with moderate errors.

Despite these challenges, ChatGPT-4 proved to be the most suitable AI model for the specific requirements at the current time.

It should be mentioned that there are still few studies on AI-generated codes. Rabi (2024) conducted a quantitative study on the creation of Python code with the support of Chat-GPT, which came to the conclusion that the generated code is quite usable, but still has a lot of potential for minimizing quality problems and security issues. Rabi also urges caution when using AI-supported code. However, these findings do not apply to the specific example of the use of Python code in Grasshopper, so the security problem can be neglected in this case. However, the results are positive that a successful outcome can be achieved.

4.1. Case study 01

The first successful attempt with Chat-GPT 3.5, in which a wavy wall is generated, looked like this:

> "Give me a Rhino Grasshopper code with no comments in the code. Line up 20 cubes with the dimensions 10cm width by 20cm length and 10cm height lengthwise. Duplicate this row of cubes and stack them on top of the first row of cubes. Repeat this step until 10 rows of cubes are stacked on top of each other."

The wall created by Chat-GPT 3.5 was not wavy, but straight, which is why this information was formulated in another question.

> "Take this code and arrange the bottom row of cubes along the x-axis in a serpentine pattern. Match the rows of cubes stacked on top to the new shape of the bottom row. The output must be a."

The resulting Python code created a wavy wall in Rhino Grasshopper, but could not be replicated. Therefore, the newer version, Chat-GPT 4, was tested.

4.2. Case study 02

The following describes the successful chat history with Chat-GPT 4, in which a vase is generated. To explain to the language model how to generate a 3D model of a vase in Rhino Grasshopper, squared paper is used to define the distances and the shape of the vase and to visualize the intended result.

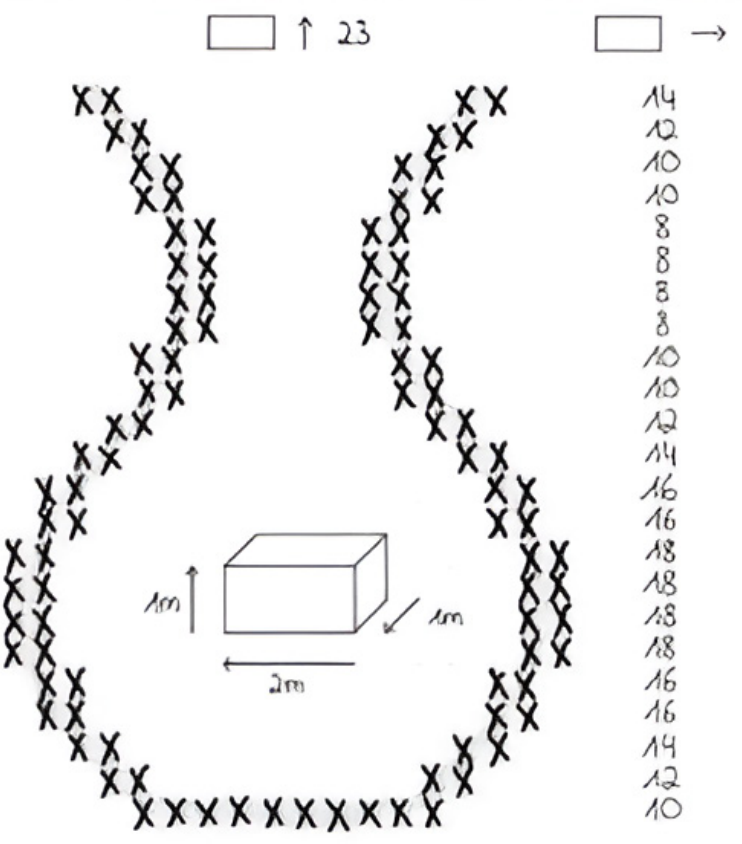


Figure 3. Sketch method - Source: own figures.

First, the framework for the shape of the vase is set by defining the desired size of the blocks. These blocks will later form the structure of the vase and in one example have the dimensions 2mx2mx1m. These dimensions serve as the basis for the overall shape of the vase. After determining the block size, a rough outer contour of the vase is sketched, typically by drawing two crosses that together represent a block to visualize a basic shape of the vase. The overall height of the vase is then determined. In this example, the vase is 23 blocks high in total. To give the language model a clear idea of the desired shape of the vase, the interior of the vase is mentally filled with blocks. The final number of blocks next to each other is then defined as the width of the individual rows, which defines the exact shape and contour of the vase.

This leads to the following question:

"Create a Python code for Rhino Grasshopper. The output is a. Arrange 2m x 1m x 1m blocks in a circle. Then stack another of these circles on top of the first and repeat this until the shape of a 3D vase is created. The construction should be 23m high. If you look at the construction in 2D, the shape would be as follows: 23m high, row 1-14 by width: 10m, 12m, 14m, 16m, 16m, 18m, 18m, 18m, 18m, 16m, 16m, 14m, 12m, 10m, 10m, 8m, 8m, 8m, 8m, 10m, 10m, 12m, 14m."

Implementing a complex modelling project using artificial intelligence requires careful control of the input data and process parameters. In a specific experiment, it became apparent that the number of inputs exceeded the processing capacity of the Chat-GPT AI model, leading to incomplete or erroneous results. This phenomenon emphasizes the need to appropriately balance the amount of data and complexity of the queries to ensure effective and accurate outputs.

Based on this insight, it was specified in a subsequent interaction that the output parameter should be defined as 'a'. This measure aimed to standardize the communication between the language model and the targeted application - in this case a modular script in a Rhino Grasshopper environment - and thus simplify the interpretation of the results. Specifying 'a' as the default output parameter should help to make the interface between generative code and graphical software more efficient and optimize the integration of dynamically generated data into the modelling process.

This iterative approach to adapting the query parameters and the subsequent refinement of the code illustrates the experimental nature of working with AI-supported design tools. It also illustrates the importance of a clear and precise specification of requirements in dialogue with advanced AI systems in order to achieve the desired results and fully exploit the power of these technologies.

The Output must be defined as “a”, as this is the default output for the GYPyhton module in Grasshopper.

After clarifying and redefining the output as ‘a’, Chat-GPT managed to generate a working code. This success led to the next phase of the process, in which the interaction with the artificial intelligence system was further refined. To maximize the applicability and usability of the generated code in the Rhino Grasshopper environment, it was then explained how the inputs can be efficiently controlled via sliders.

These sliders allow users to dynamically adjust the parameters of the model without having to edit the code directly. By integrating these controls, an interactive interface is created that allows designers to experiment with the model parameters in real time. The instructions on how to configure the sliders have been carefully formulated to ensure that the AI interprets and implements the instructions correctly.

This step underlines the importance of seamlessly integrating AI-generated code with user-friendly interfaces in software tools for digital design. It also illustrates how the precise communication of specific requirements to the AI system can significantly improve the customizability and functionality of the developed solutions. The approach of providing specific instructions for adjusting sliders demonstrates an important facet of human-AI collaboration, where fine-tuning parameters plays a central role in providing users with maximum control and flexibility.

The dimensions for the blocks are given by external sliders in Grasshopper, so change the code so that the sliders change the dimensions of the blocks by the inputs x, y and z, which are the default inputs in Grasshopper.
As a result, Chat-GPT added a sample block for the data x, y and z at the end, which had to be subsequently removed. A further request to Chat-GPT to delete the example block resulted in a working code with customizable sliders.

1. Conception of a geometric shape: Think about and decide on a specific geometric shape you want to create. Ability to understand and generate Python code: The AI should be able to effectively interpret and generate Python code, which is a basic requirement for integration into existing systems.

2. Visualization and measurement: Draw the chosen shape on paper and measure the necessary distances and dimensions to have accurate measurements for the digital implementation.

3. Definition and communication of the shape: Describe the shape to ChatGPT 4 with all relevant measurements and details to give a clear idea of the structure.

4. Software preparation: start Rhino and open Grasshopper, a visual programming environment integrated with Rhino, to model the shape. Use explanations to describe complex geometric shapes with Grasshopper Python code: The AI should be able to describe complex geometric structures in a clear and understandable way and explain the relevant scripting methods.

5. Code integration: Insert the code generated by ChatGPT 4 into a Python module (GyPython) within Grasshopper.

6. Code execution: Execute the Python module to generate the geometric shape based on the input data.

7. Troubleshooting (optional): If errors occur while running the code, you can report them to ChatGPT 4 and ask for suggested corrections. After receiving the corrected version, repeat steps 5 and 6.

8. Integration with Fologram: Connect the GyPython module to a Fologram module designed to work with augmented reality (AR) technology.

9. Hololens activation: Start your Hololens and connect to the Fologram module to visualize the geometric shape in augmented reality.

10. Viewing the shape: View the geometric shape through the Hololens to see a realistic 3D model of your design in the real environment.

Results

Within the scope of advanced software-assisted coding methods, certain customizations and functions are feasible, while others are beyond the current technological reach. Among the options that can be implemented is the ability to make detailed customizations to the code upon request.

This flexibility allows users to precisely address specific requirements, making customized programming much easier. It is also possible to identify and correct error messages that occur in the generated code. This correction function makes a decisive contribution to the efficiency and reliability of the programming process. Furthermore, continuous modifications of the code can be implemented within an interaction cycle of up to 30 responses as part of Edge Copilot, which works on the basis of Chat-Gpt 4 ***(kppowell, 2024),*** enabling dynamic adaptation to changing requirements during a session.

On the other hand, there are limitations that affect the complexity of the queries and the processing capacities. It is not possible to process more than three inputs in a query without prior fine-tuning of the AI, which underlines the need for preliminary configuration in order to effectively utilize the AI's capabilities. Also beyond current capabilities is the complex stacking of objects on demand in a way that would allow subsequent physical rebuilding using photogrammetry, although similar structures can be generated. These limitations highlight the challenges of dealing with highly complex data structures and three-dimensional modelling in real time.

As part of the research work, a method has been successfully developed with which a vase can be reconstructed from individual blocks. This reconstruction is achieved by using a hologram in conjunction with AR Glasses (HoloLens2). The implementation of this method made it possible to initiate and carry out the construction of the vase with a single request.

Figure 6 shows that the artificial intelligence has generated a functional code. Although this code does not exactly reproduce the shape specified in the ChatGPT-4 template, the artificial intelligence has nevertheless developed a complex geometry. This geometry has similarities to the original presented in

Figure 6. The development of such a geometry, despite the deviation from the original shape, underlines the ability of the A.I. to generate adaptive and innovative solutions within the given parameters.

The development of a 3D model using the Chat-GPT 4 AI language model is an accessible entry point for beginners into the world of digital modeling. This method allows users to create basic structures and simple designs without extensive prior knowledge of programming or special modeling techniques. Through intuitive interaction with the language model, users can articulate their ideas and have them translated into concrete model data.

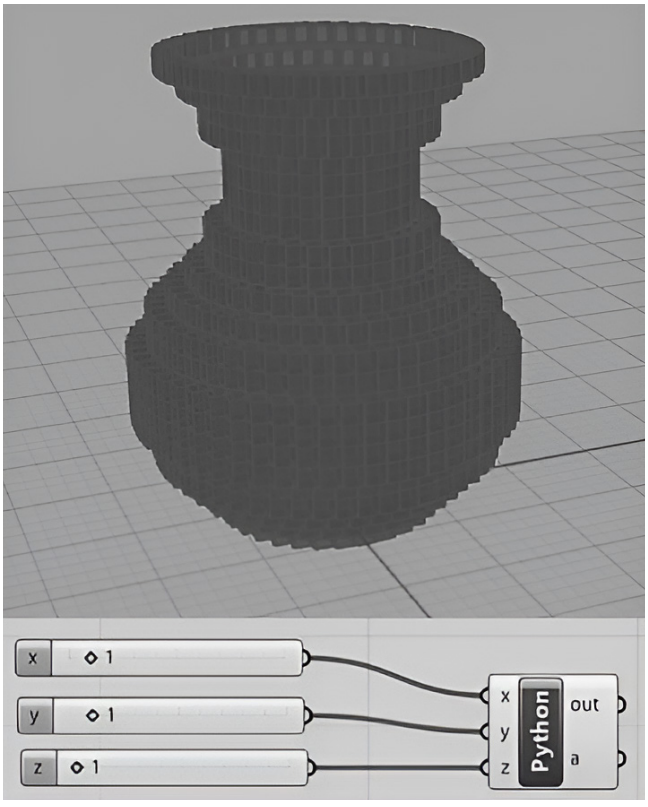


Figure 4. Final Result in Grasshopper with Sliders - Source: own figures.

```
import rhinoscriptsyntax as rs
import math

def create_vase(height, levels, widths, x, y, z):
    # x, y, z sind die Dimensionen der Blöcke, die von externen Slidern gesteuert werden
    radius_list = [width / 2 for width in widths]
    level_height = height / levels  # Höhe jeder Ebene

    vase = []

    for i in range(levels):
        radius = radius_list[i]
        circumference = 2 * math.pi * radius
        num_blocks = int(circumference / x)  # Anzahl der Blöcke pro Ebene, basierend auf der X-Dimension
        angle_step = 360.0 / num_blocks  # Winkelabstand zwischen den Blöcken

        for j in range(num_blocks):
            angle = math.radians(j * angle_step)
            x_pos = radius * math.cos(angle)
            y_pos = radius * math.sin(angle)
            z_pos = i * level_height
            block = rs.AddBox([
                (x_pos - x/2, y_pos - y/2, z_pos),
                (x_pos + x/2, y_pos - y/2, z_pos),
                (x_pos + x/2, y_pos + y/2, z_pos),
                (x_pos - x/2, y_pos + y/2, z_pos),
                (x_pos - x/2, y_pos - y/2, z_pos + z),
                (x_pos + x/2, y_pos - y/2, z_pos + z),
                (x_pos + x/2, y_pos + y/2, z_pos + z),
                (x_pos - x/2, y_pos + y/2, z_pos + z)
            ])
            vase.append(block)
    return vase

# Höhe der Vase in Metern und Anzahl der Ebenen
total_height = 23
num_levels = 23

# Breiten jeder Ebene in Metern
level_widths = [10, 12, 14, 16, 16, 18, 18, 18, 18, 16, 16, 14, 12, 10, 10, 8, 8, 8, 8, 10, 10, 12, 14]


# Erzeuge die Vase und weise sie der Ausgabevariablen 'a' zu
a = create_vase(total_height, num_levels, level_widths, x, y, z)
```

Figure 5. Code of Figure 4. - Source: own figures.

However, the use of such a language model reaches its limits when it comes to creating specific, complex models. It becomes particularly challenging when the requirements are very detailed and a great amount of information has to be processed in a single request. In such cases, the capacity of the Chat-GPT 4 language model is not sufficient to adequately capture and implement all details, which can lead to the system being overloaded.

Furthermore, the accuracy of the model implementation by the AI is limited in complex scenarios, as the interpretation of the instructions and the conversion into precise geometric data depends on the quality and clarity of the information provided. An iterative process of feedback and adaptation is often required to achieve the desired result. This is where advanced software solutions such as Rhino and Grasshopper, in conjunction with scripting modules such as Python, can be helpful to make more detailed adjustments and refine the model step by step. These tools offer the possibility of integrating programming languages and algorithmic design to make highly complex structures and precise adjustments that go far beyond basic model creation.

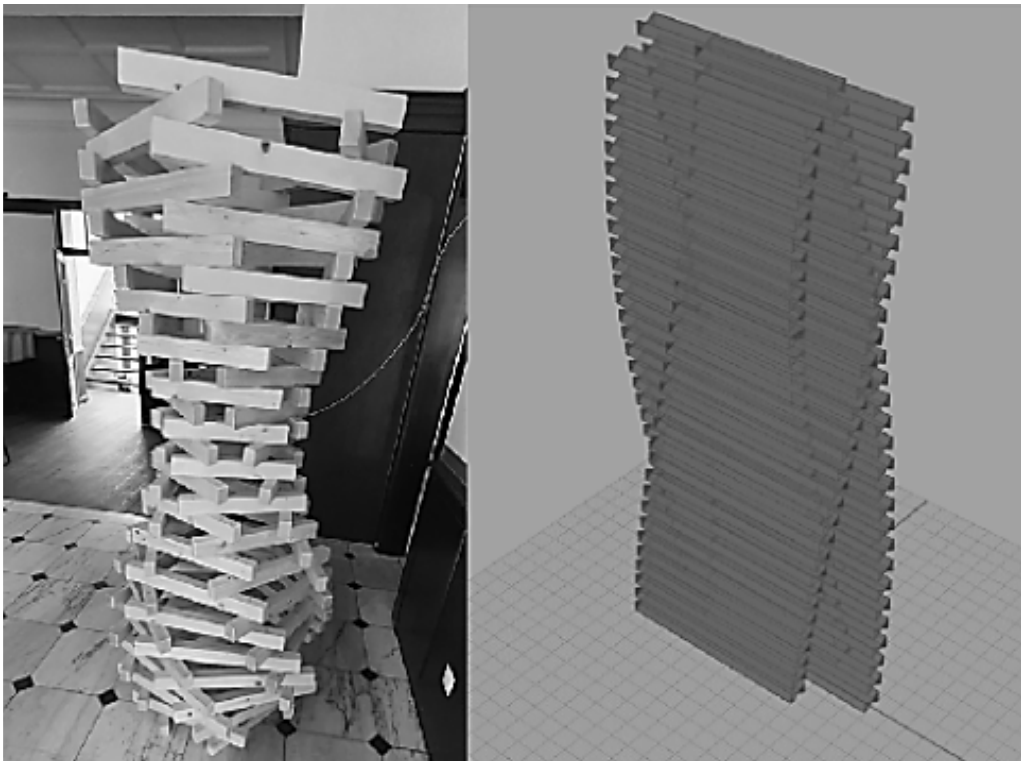


Figure 6. Real template and AI generated result - Source: own figures.

```
import rhinoscriptsyntax as rs
import math

def tower_geometry():
    # Parameters
    block_length = 50  # Length of each block
    block_width = 5    # Width of each block
    block_height = 3   # Height of each block
    num_levels = 30    # Number of levels in the tower
    rotation_angle = 5  # Base rotation angle increment per level in degrees
    height_increment = 5  # Vertical spacing between levels
    offset_distance = 10  # Distance from the center for each block

    # Function to create a single level with three blocks
    def create_level(center, base_rotation, height):
        geometries = []
        for i in range(3):  # Three blocks per level
            rotation = base_rotation + i * 120
            rad = math.radians(rotation)
            x_offset = offset_distance * math.cos(rad)
            y_offset = offset_distance * math.sin(rad)

            # Define the corners of the box
            corners = []
            for dx in [-1, 1]:
                for dy in [-1, 1]:
                    for dz in [0, 1]:
                        x = center[0] + x_offset + dx * block_length / 2
                        y = center[1] + y_offset + dy * block_width / 2
                        z = height + dz * block_height
                        corners.append((x, y, z))

            # Create the box
            try:
                box = rs.AddBox(corners)
                if box:
                    geometries.append(box)
                else:
                    print("Failed to create box at level {}, block {}".format(i, rotation))
            except Exception as e:
                print("Error creating box: {}".format(str(e)))

        return geometries

    # Initialize variables
    current_base_rotation = 0
    center_point = [0, 0, 0]
    all_geometries = []

    # Generate the tower
    for level in range(num_levels):
        level_blocks = create_level(center_point, current_base_rotation, current_height)
        all_geometries.extend(level_blocks)
        current_height += height_increment
        current_base_rotation += rotation_angle

    return all_geometries

# Output the geometries
a = tower_geometry()
```

Figure 7. Code of Figure 6 - Source: own figures.

Discussion and Conclusion

The methodology used in this study to utilize ChatGPT-4 to generate Python code for parametric design in Rhino Grasshopper raises the question of whether this methodology is transferable to other software applications that accept code inputs. While Rhino Grasshopper is a widely used platform in the field of parametric design, similar approaches could be applied in other design software environments. Future research could focus on investigating the applicability of this methodology to different software platforms and making possible adjustments to optimize the generation of code in different environments.

It should also be noted that the development of artificial intelligence in the design process is expected to evolve significantly over the next few years, which could lead to a variety of new applications. The global AI market is currently estimated to be worth over 196 billion US dollars and is predicted to grow thirteen-fold over the next 7 years ***(Howarth 2024).*** While ChatGPT-4 already has advanced code and text generation capabilities, it is likely that future versions will be even more powerful and versatile.

A promising direction for further research is to investigate the performance and effectiveness of ChatGPT 4.o in comparison to the version tested here. As technology continues to evolve, it is important to evaluate the latest models and techniques to ensure that the best available tools are used in the design process. By comparing different versions of ChatGPT, researchers and practitioners can better understand how the performance of AI models evolves over time and the impact this can have on the design process.

It was shown that ChatGPT-4 can process complex relationships effectively, especially in the area of visual scripting, as represented by the combination of Rhino and Grasshopper. These programs are often used in architectural design to create highly complex structures and shapes through parametric design.

ChatGPT-4 proved capable of understanding and applying the logic behind visual scripting and the interactions between these two software solutions. However, this is not the primary function of ChatGPT-4, so specific methods had to be developed to precisely explain to the language-based model what forms and structures should be generated. These adapted approaches allowed the model to be used effectively for specific needs.

It is assumed that this method is also transferable to other Large Language Models (LLM) and that an Artificial Intelligence (AI) developed specifically for this use case could achieve even better and more complex results. These findings suggest that the adaptability and broad applicability of LLMs such as ChatGPT-4 make it possible to tackle specialized tasks in various disciplines, especially in technically demanding areas such as computer-aided architecture.

Acknowledgement

This research was supported by the AUFLADEN project at Jade Hochschule, Oldenburg which is funded by Stiftung Innovation in der Hochschullehre.

Figure 7. Logo „Stiftung Innovation in der Hochschullehre"

Conflict of Interests

The authors declare no conflict of interest.

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. Proceedings of the 34th Conference on Neural Information Processing Systems. https://dl.acm.org/doi/pdf/10.5555/3495724.3495883

Grunwald, G., & Heins, C. (2022). BIM Game: A testing ground for specifying, modeling, evaluating, and visualizing information in IFC formats. In T. Kang (Ed.), Proceedings of the 5th International Conference on Civil Engineering and Architecture. https://doi.org/10.1007/978-981-99-4049-3

Grunwald, G., & Heins, C. (2024). BIM Game: A testing ground for specifying, modeling, evaluating, and visualizing information in IFC formats. In T. Kang (Ed.), Proceedings of the 5th International Conference on Civil Engineering and Architecture. Lecture Notes in Civil Engineering (Vol. 369). Springer, Singapore. https://doi.org/10.1007/978-981-99-4049-3_52

Hanke, T. (2024). AUFLADEN: The web portal for self-study in the field of digital planning and construction. In T. Luhmann & T. Siebert (Eds.), Photogrammetry, laser scanning, optical 3D metrology: Contributions to the Oldenburg 3D Days and the BIM Day 2024 (pp. 89–104). ISBN: 978-3-87907-750-2. https://www.vde-verlag.de/buecher/537750/photogrammetrie-laserscanning-optische-3d-messtechnik.html

Hanke, T., Kawasaki, J., & Grunwald, G. (2023). Manufacturing processes of complex shapes and structures using 3D printing and augmented reality. In H. Ahmad Nia & R. Rahbarianyazd (Eds.), 6th International Conference of Contemporary Affairs in Architecture and Urbanism (ICCAUA 2023), 14-16 June 2023 (pp. 243–256). E-ISBN: 978-605-71006-7-2.

Heins, C., & Grunwald, G. (2021). Gamification and BIM: The didactic guidance of decentralized interactions of a real-life BIM business game for higher education. ISARC 2021 Conference Paper, 38th International Symposium on Automation and Robotics in Construction. https://doi.org/10.22260/ISARC2021/0126

Heins, C., & Grunwald, G. (2024). BIM and IPA: Excerpt of an automated assessment system for an autodidactic teaching concept. ISARC 2024 Conference Paper, 41st International Symposium on Automation and Robotics in Construction. https://doi.org/10.22260/ISARC2024/0108

Kawasaki, J. Y., & Grunwald, G. (2023). Enhanced learning and teaching through AR: Case studies on design-build projects. JSEE Annual Conference. https://doi.org/10.20549/jseeen.2023.0_18

Kawasaki, J. Y., Hirsekorn, Y., Ansre, N., & Grunwald, G. (2024). AUFLADEN LAB: Parametric AI-supported design. In T. Luhmann & T. Siebert (Eds.), Photogrammetry, laser scanning, optical 3D metrology: Contributions to the Oldenburg 3D Days and the BIM Day 2024 (pp. 105–118). ISBN: 978-3-87907-750-2. https://www.vde-verlag.de/buecher/537750/photogrammetrie-laserscanning-optische-3d-messtechnik.html

Reinemann, G., & Watanabe, A. (2024). KI in der universitären Lehre: Vom Spannungs- zum Gestaltungsfeld. In G. Schreiber & L. Ohly (Eds.), Diskurse über KI-Textgeneratoren (pp. 45–61). https://doi.org/10.1515/9783111351490-004

Tasnia, T., & Grunwald, G. (2024). Evaluate the outcome of the digital learning platform AUFLADEN. In T. Luhmann & T. Siebert (Eds.), Photogrammetry, laser scanning, optical 3D metrology: Contributions to the Oldenburg 3D Days and the BIM Day 2024 (pp. 119–132). ISBN: 978-3-87907-750-2. https://www.vde-verlag.de/buecher/537750/photogrammetrie-laserscanning-optische-3d-messtechnik.html

Zheng, H., Guo, Z., & Liang, Y. (2019). Iterative pattern design via Decodes Python scripts in Grasshopper. In J.-H. Lee (Ed.), “Hello, Culture!” 18th International Conference, CAAD Futures 2019 - Proceedings (pp. 526–537). ISBN: 978-89-89453-05-5