Automatic Building Energy Model Development and Debugging Using Large Language Models Agentic Workflow

Liang Zhang[1,2*], Vitaly Ford[3], Zhelun Chen[4], Jianli Chen[5,6]
1. University of Arizona, Tucson, Arizona, U.S.
2. National Renewable Energy Laboratory, Golden, Colorado, U.S.
3. Arcadia University, Philadelphia, Pennsylvania, U.S.
4. Drexel University, Philadelphia, Pennsylvania, U.S.
5. Tongji University, Shanghai, China
6. University of Utah, Salt Lake City, Utah, U.S.
* Corresponding author: liangzhang1@arizona.edu

Highlights

- Explore the use of large language models (LLMs) to automate building energy modeling
- Develop an LLM-in-the-loop workflow based on LLM planning for complex procedures
- Accurately translate building description to an EnergyPlus model in a case study
- Outperform naive prompt engineering and other planning in accuracy and reliability

Abstract

Building energy modeling (BEM) is a complex process that demands significant time and expertise, limiting its broader application in building design and operations. While Large Language Models (LLMs) agentic workflow have facilitated complex engineering processes, their application in BEM has not been specifically explored. This paper investigates the feasibility of automating BEM using LLM agentic workflow. We developed a generic LLM-planning-based workflow that takes a building description as input and generates an error-free EnergyPlus building energy model. Our robust workflow includes four core agents: 1) Building Description Pre-Processing, 2) IDF Object Information Extraction, 3) Single IDF Object Generator Suite, and 4) IDF Debugging Agent. These agents divide the complex tasks into manageable sub-steps, enabling LLMs to generate accurate and reliable results at each stage. The case study demonstrates the successful translation of a building description into an error-free EnergyPlus model for the iUnit modular building at the National Renewable Energy Laboratory. The effectiveness of our workflow surpasses: 1) naive prompt engineering, 2) other LLM-based workflows, and 3) manual modeling, in terms of accuracy, reliability, and time efficiency. The paper concludes with a discussion on the interplay between foundational models and LLM agent planning design, advocating for the use of fine-tuned, specialized models to advance this field.

Keywords

Building energy modeling, complex system modeling, large language model, generative artificial intelligence, agent planning

1 Introduction

The building sector represents a significant portion of global energy consumption and carbon emissions. According to recent estimates, buildings account for approximately 40% of total worldwide energy usage and are responsible for around 33% of global carbon emissions [1]. This substantial contribution highlights the critical role that energy efficiency and sustainable practices in building design, construction, and operation must play in global efforts to mitigate climate change and achieve environmental sustainability goals.

Building Energy Modeling (BEM) is crucial for decarbonization efforts. BEM serves as a key tool in strategies to reduce carbon emissions in our built environments. By providing a detailed analysis of a building's energy consumption patterns, BEM allows stakeholders to make informed choices that enhance energy efficiency and promote environmentally sustainable design and operations.

BEM is a complex field demanding a comprehensive understanding of building science, building engineering, and equipment. It delves into the interconnected nature of a building's systems, particularly heating, ventilation, and air conditioning (HVAC). Consequently, BEM users require a strong foundation in these systems and the physical principles governing their interactions. This knowledge is crucial for accurately capturing building characteristics and predicting energy consumption. Additionally, proficiency in specialized BEM software presents another hurdle. Each program has its own intricacies, language, and operational complexities. Mastering these tools requires a substantial time investment, potentially limiting accessibility for those lacking the background or resources. The challenge is further intensified with the growing sophistication of modern buildings. Complex mechanical systems, novel materials, and connections to broader energy networks elevate the demands on BEM practitioners. Sustainability concerns add another layer, as users must stay abreast of emerging technologies and navigate an evolving regulatory landscape. As a result, BEM has become an expertise-driven field. It necessitates a broad and deep knowledge base encompassing multiple disciplines. This multifaceted nature can pose a significant barrier for newcomers and even experienced professionals. Yet, only a few papers are found to focus on the automation of BEM tools. For example, Kamel and Memari [2] developed a tool that automates BEM and provides detailed outputs by integrating Building Information Modeling with modified versions of energy simulation tools such as EnergyPlus and OpenStudio. This tool accepts gbXML files and automatically calculates heat transfer through individual building envelope components, including windows, doors, and walls. However, its reliance on structured input (gbXML) limits its broader application.

The advent of tools like GitHub Copilot [3] empowered by large language models (LLMs) has dramatically transformed the programming landscape by lowering the barriers to generate programming code, suggesting a similar approach could revolutionize BEM. The concept of a “BEM copilot” offers an enticing solution to democratize BEM by simplifying the generation of the necessary textual input data for energy models. For instance, EnergyPlus [4], a leading BEM tool, utilizes text-based input data in IDF (Input Data File) format to characterize buildings. This text-oriented nature of BEM inputs aligns well with the capabilities of LLMs, allowing for the automated generation of these inputs. This approach could significantly reduce the technical barriers, making BEM more automated and accessible to a wider audience and stakeholders. One relevant study was identified in our literature review. Jiang et al. [5] translated natural language description of buildings to established building models of various geometries, occupancy scenarios, and equipment loads based on a fine-tuned LLM. Although achieved high accuracy of 100%, the work only processes simplified geometer and building description, which needs to be improved for complex modeling scenarios.

With the proven capability of LLM in code/text generation and reasoning in many engineering modeling areas, this approach still present challenges when directly used in BEM. The generation of input files for BEM is an intricate engineering process that involves great expert knowledge and numerous steps. The intricacy of this task directly influences the performance of LLMs; as the complexity increases, the ability of the LLM to produce accurate and useful results often diminishes [6, 7]. Consequently, the sophistication of creating input files for BEM may exceed the LLM's capabilities to replicate the nuanced engineering knowledge and sequence of steps required. This highlights the need for innovative applications and frameworks that can effectively harness LLMs to navigate the complexities of real-world engineering processes, as suggested by the proposed typologies in the field of agentic reasoning. Agentic reasoning design patterns for a complex engineering process include four categories, according to Andrew Ng: 1) reflection [8, 9], 2) tool use [10, 11], 3) planning [12, 13], and 4) multi-agent collaboration [14, 15]. Although a simple application of LLM agentic workflow to modify a single IDF object (EnergyPlus object) was once studied by Zhang et al. [16], a systematic application of an agentic LLM designed for the whole complex BEM process has never been discussed.

For the second challenge, the constraint posed by the input/output token limitations of LLMs and their respective costs can pose difficulties for BEM. A token in this context refers to pieces of text that LLMs process as single units, approximately equal to four characters or three-quarters of a word. Many models have predefined limits on the length of input and output they can process, which becomes problematic when dealing with the extensive data typical of BEM files, such as IDF files. For instance, a large-sized office reference building's data can encompass about 120,000 tokens, reaching or exceeding the capabilities of many commercial LLMs, such as GPT-4o model, support the token limit of 128,000. Furthermore, the financial cost associated with processing such voluminous data through commercial APIs can be prohibitive. For example, if processing 120,000 tokens, GPT-4 models would incur a cost of approximately $0.6 for input and $1.8 for output per session, based on the rates of $5.00/1M tokens for input and $15.00/1M tokens for output as of June 3, 2024, if using API from “gpt-4o” and “gpt-4o-2024-05-13” models. Some commercial LLMs support an extra-large token size, Claude 3 being one of the more prominent ones in this regard with support up to 200,000 tokens at the time of writing (the beta of a newer version of Google Gemini AI supports a million tokens but it is yet to be tested, compared, and released). To process 120,000 tokens on the Claude’s best model would cost $1.8 for input and $9.0 for output per session.

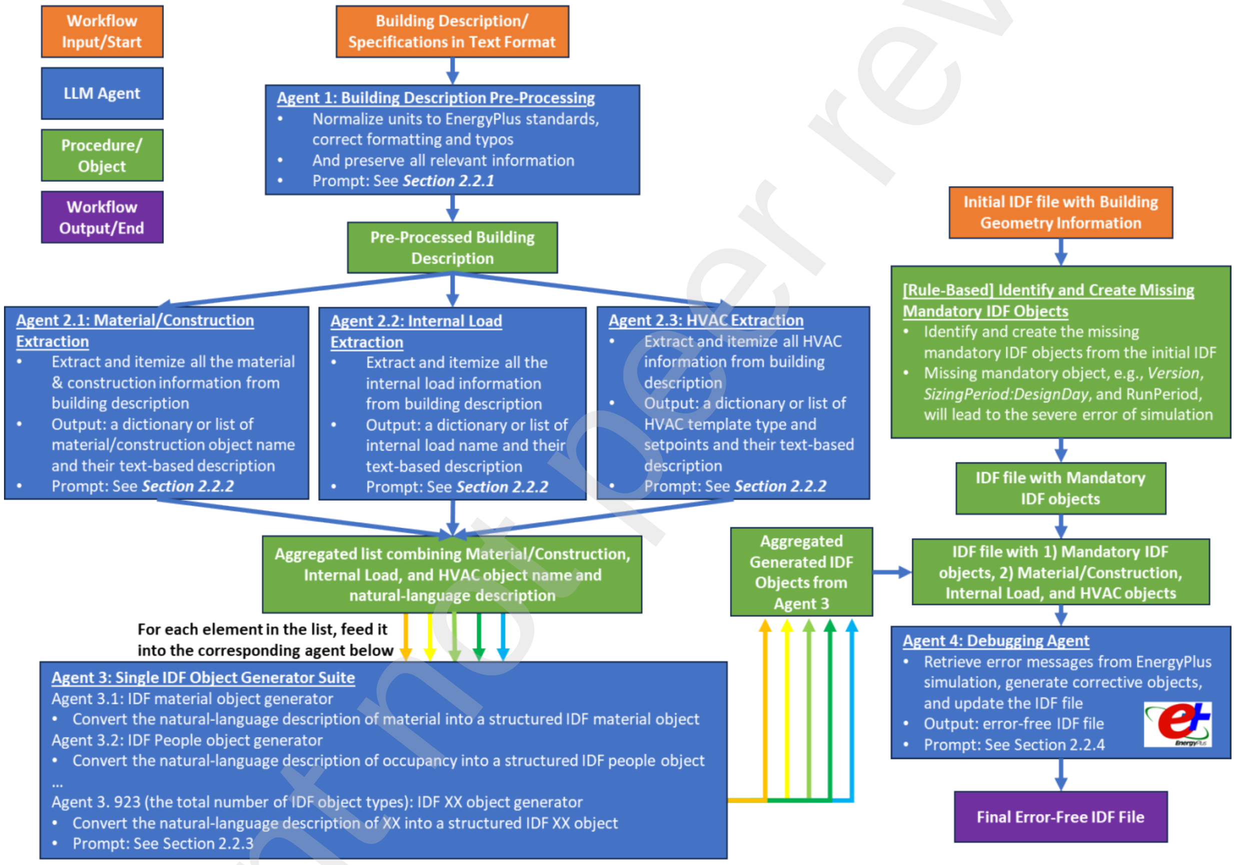
While these costs may not seem substantial, incorporating an engineered process with repeated generation could lead to expenses of hundreds of dollars just to operate one model. Therefore, a more efficient strategy is essential to manage BEM data effectively within the limitations of current LLM technologies. The research hypothesis of this paper posits that LLM agentic workflow have the potential to generate structured, syntax-compliant text formats as input data for complex BEM scenarios. This study aims to explore innovative solutions to these two challenges mentioned earlier, focusing on the potential of LLM multi-agent systems and planning techniques with economic use of tokens. By leveraging these advanced agentic LLM methodologies, the study proposed an LLM-planning-based workflow that seeks to overcome the inaccuracy issue in complex engineering procedures and the cost limitations, and unlock the full potential of LLMs in the development and debugging of building energy models. Specifically, the workflow automates the generation of EnergyPlus input data IDF file from natural-language-based building description. The workflow is designed with four core agents: 1) Building Description Pre-Processing, 2) IDF Object Information Extraction, 3) Single IDF Object Generator Suite, and 4) Debugging Agent. These agents divide the complex tasks into manageable sub-steps, enabling LLMs to generate accurate and reliable results at each stage, and generate the final error-free IDF simulation file as the final output.

The paper is structured as follows: Section 2 presents a detailed description of the methodology and workflow. Section 3 introduces three case studies that apply this methodology. Section 4 contains the discussion, and Section 5 concludes with remarks on future work."

2 Methodology

This section outlines the LLM-planning workflow designed to automate modeling and debugging for EnergyPlus input IDF file format. It details how LLM agents are developed and structured to form a cohesive planning framework. Section 2.1 describes the entire workflow, while Section 2.2 focuses on the specific roles and functions of each LLM agent.

2.1 Workflow



Workflow Input/Start
LLM Agent
Procedure/ Object
Workflow Output/End
Building Description/ Specifications in Text Format
Agent 1: Building Description Pre-Processing
• Normalize units to EnergyPlus standards, correct formatting and typos
• And preserve all relevant information
• Prompt: See Section 2.2.1
Pre-Processed Building Description
Agent 2.1: Material/Construction Extraction
• Extract and itemize all the material & construction information from building description
• Output: a dictionary or list of material/construction object name and their text-based description
• Prompt: See Section 2.2.2
Agent 2.2: Internal Load Extraction
• Extract and itemize all the internal load information from building description
• Output: a dictionary or list of internal load name and their text-based description
• Prompt: See Section 2.2.2
Agent 2.3: HVAC Extraction
• Extract and itemize all HVAC information from building description
• Output: a dictionary or list of HVAC template type and setpoints and their text-based description
• Prompt: See Section 2.2.2
Aggregated list combining Material/Construction, Internal Load, and HVAC object name and natural-language description
For each element in the list, feed it into the corresponding agent below
Agent 3: Single IDF Object Generator Suite
Agent 3.1: IDF material object generator
• Convert the natural-language description of material into a structured IDF material object
Agent 3.2: IDF People object generator
• Convert the natural-language description of occupancy into a structured IDF people object
...
Agent 3. 923 (the total number of IDF object types): IDF XX object generator
• Convert the natural-language description of XX into a structured IDF XX object
• Prompt: See Section 2.2.3
Aggregated Generated IDF Objects from Agent 3
Initial IDF file with Building Geometry Information
[Rule-Based] Identify and Create Missing Mandatory IDF Objects
• Identify and create the missing mandatory IDF objects from the initial IDF
• Missing mandatory object, e.g., Version, SizingPeriod:DesignDay, and RunPeriod, will lead to the severe error of simulation
IDF file with Mandatory IDF objects
IDF file with 1) Mandatory IDF objects, 2) Material/Construction, Internal Load, and HVAC objects
Agent 4: Debugging Agent
• Retrieve error messages from EnergyPlus simulation, generate corrective objects, and update the IDF file
• Output: error-free IDF file
• Prompt: See Section 2.2.4
Final Error-Free IDF File

Figure 1 illustrates the LLM-planning workflow designed for generating and debugging BEM input files, specifically focusing on IDF files, derived from natural-language descriptions of buildings. The process initiates with two key inputs: 1) the natural-language description of the building, treated as unstructured data, and 2) an initial IDF file that includes geometry and zone information. Geometry generation is not discussed or considered in this paper due to its complexity and heavy reliance on graphical user interfaces, which are difficult for LLMs to replicate. Instead of generating geometries from descriptions, they are typically created using tools like OpenStudio or DesignBuilder, or imported from other data formats. The initial IDF is assumed to include the geometry and zone information generated earlier. Below are the detailed steps.

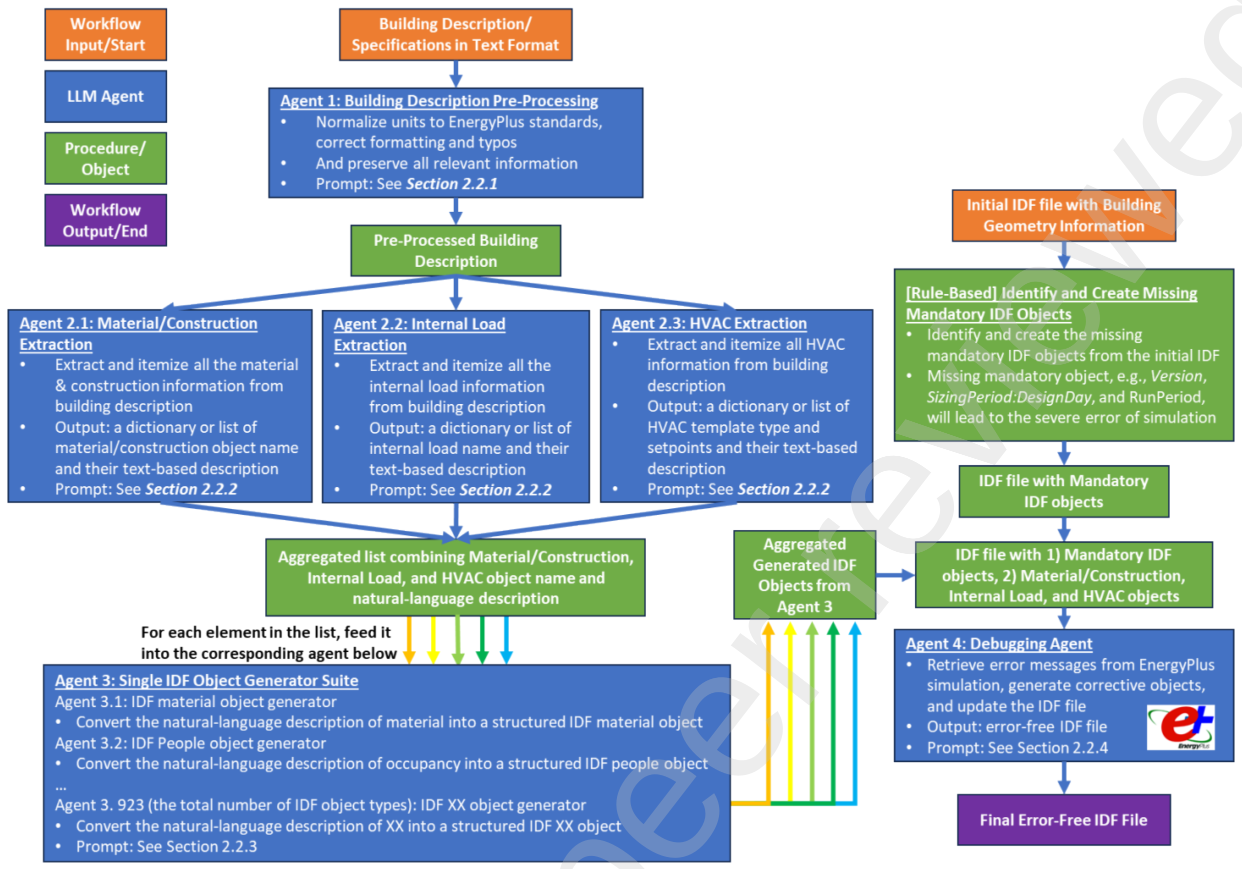

Workflow Input/Start
LLM Agent
Procedure/ Object
Workflow Output/End
Building Description/ Specifications in Text Format
Agent 1: Building Description Pre-Processing
Normalize units to EnergyPlus standards, correct formatting and typos
And preserve all relevant information
Prompt: See Section 2.2.1
Pre-Processed Building Description
Agent 2.1: Material/Construction Extraction
Extract and itemize all the material & construction information from building description
Output: a dictionary or list of material/construction object name and their text-based description
Prompt: See Section 2.2.2
Agent 2.2: Internal Load Extraction
Extract and itemize all the internal load information from building description
Output: a dictionary or list of internal load name and their text-based description
Prompt: See Section 2.2.2
Agent 2.3: HVAC Extraction
Extract and itemize all HVAC information from building description
Output: a dictionary or list of HVAC template type and setpoints and their text-based description
Prompt: See Section 2.2.2
Aggregated list combining Material/Construction, Internal Load, and HVAC object name and natural-language description
For each element in the list, feed it into the corresponding agent below
Agent 3: Single IDF Object Generator Suite
Agent 3.1: IDF material object generator
Convert the natural-language description of material into a structured IDF material object
Agent 3.2: IDF People object generator
Convert the natural-language description of occupancy into a structured IDF people object
...
Agent 3. 923 (the total number of IDF object types): IDF XX object generator
Convert the natural-language description of XX into a structured IDF XX object
Prompt: See Section 2.2.3
Aggregated Generated IDF Objects from Agent 3
Initial IDF file with Building Geometry Information
[Rule-Based] Identify and Create Missing Mandatory IDF Objects
Identify and create the missing mandatory IDF objects from the initial IDF
Missing mandatory object, e.g., Version, SizingPeriod:DesignDay, and RunPeriod, will lead to the severe error of simulation
IDF file with Mandatory IDF objects
IDF file with 1) Mandatory IDF objects, 2) Material/Construction, Internal Load, and HVAC objects
Agent 4: Debugging Agent
Retrieve error messages from EnergyPlus simulation, generate corrective objects, and update the IDF file
Output: error-free IDF file
Prompt: See Section 2.2.4
Final Error-Free IDF File

Figure 1. Diagram of the developed LLM-in-the-loop workflow for BEM input generation and debugging

First, the workflow begins by gathering natural-language descriptions of a building from modelers, detailing internal loads, envelope materials, energy systems, and more. **Second**, the Building Description Pre-Processing Agent (Agent 1, Section 2.2.1), standardizes these descriptions by converting all units to those compatible with EnergyPlus and cleaning the data. **Third**, the IDF Object Type and Description Extraction Agent (Agent 2, Section 2.2.2) methodically identifies and categorizes potential IDF objects within the description to ensure no element is missed, using specialized sub-agents for materials, loads, and HVAC systems. **Fourth**, the agent called Single IDF Object Generator Suite (Agent 3, Section 2.2.3) then systematically generates IDF objects from these descriptions using predefined templates for each IDF object type, ensuring all IDF object types are correctly formatted. **Finally**, the Debugging Agent (Agent 4, Section 2.2.4) iteratively checks and corrects the generated IDF file in the EnergyPlus simulation, using error analysis and correction sub-agents until no errors remain, completing the workflow.

2.2 Agent Details: Engineered Prompts

As described in Section 2.1, there are 4 agents utilized in the workflow, including 1) Building Description Pre-Processing, 2) IDF Object Type and Description Extraction, 3) Single IDF Object Generator Suite, and 4) Debugging Agent. The following sections provide more details about these agents in the form of engineered prompts.

2.2.1 Building Description Pre-Processing

Since the building description is unstructured and unpredictable, especially the units that are defined differently, the Building Description Pre-Processing Agent (Agent 1) is a critical step due to the diverse nature of building descriptions. One significant aspect of this diversity is the variation in units used, which is crucial because incorrect unit conversions can lead to errors in the generated IDF objects. The following prompt focuses solely on converting all units to match those used in EnergyPlus, ensuring unit consistency throughout the process. While this stage encompasses more than just unit conversion—such as data cleaning, merging, making the language more concise, and improving readability—these aspects are not covered in the example below. The prompts are provided in the Python-based programming language syntax.

~~~
Agent_1 = f"""This agent automatically converts a building description to another one
with the correct SI unit used in EnergyPlus.

If the unit in the building description is already a valid SI unit used in EnergyPlus,
then do nothing and return the exact same building description.

Below is the original building description
```{building_description}```"""
~~~

2.2.2 IDF Object Type and Description Extraction

This agent, IDF Object Type and Description Extraction (Agent 2), methodically extracts and itemizes all potential IDF objects from the building description to ensure no objects are overlooked. The itemized IDF objects include the type of IDF object, and the extracted natural language description for the object (not real EnergyPlus readable IDF object yet). Our observations indicate that directly converting building descriptions into IDF objects frequently results in missing and inaccurate IDF objects. To enhance accuracy, Agent 2 employs sub-agents—Agent 2.1, 2.2, and 2.3—which extract specific types of information independently: 1) materials/construction, 2) internal loads, and 3) HVAC systems.

Each agent has placeholders to get extra information to complete the prompts, including 1) {user_description}: the postprocessed building description, 2) {zone_list_str}: extracted from the initial IDF file with zone defined, 3) {common_object_list}: this is a user defined list that indicates the suggested object type for object creation.

After extracting this information individually from Agent 2.1, 2.2, and 2.3, we concatenate the generated IDF object as a list. The output from Agent 2 is a dictionary or list containing the names of the material/construction objects and their text-based descriptions.

```
Agent_2_1 = f"""I want to process user description of their building to ONLY create a
dictionary about internal load (people, lights, electric equipment, infiltration), the
dictionary's keys are the EnergyPlus IDF object type and its name to be created in tuple
format (object type in str, object name in str, object name cannot be the same as object
type), and the values to be detailed description or requirement of the IDF object, in
string format.

Only generate object type mentioned in commonly used internal load objects list attached
at the end of the prompt. Here is the user description: {user_description}

- Specific Requirements:
1. Extract schedule as much as possible from the description, use 'Schedulelimit' as the
schedule limit.
2. Create an 'Always_On' schedule, and assign it to object without specified schedule,
use 'Schedulelimit' as the schedule limit. In the dictionary, generate schedule object
first.
3. ALWAYS mention what the exact schedule to be used in the detailed description
4. One object for ONLY one zone. Do NOT use one object to describe multiple zones.
5. ALWAYS mention the exact zone name in the detailed description. The complete zone
list is: {zone_list_str}
```

```
6. For People Object, if not specified, mention in the description to use 'Always_On'
just for 'Activity Level Schedule Name'.
7. Extract as much information from the user description as possible in the detailed
description.

Commonly used internal load objects list: {common_internal_load_object_list}
"""
```

```
Agent_2_2 = f"""I want to process user description of their building to ONLY create a
dictionary about internal load (people, lights, electric equipment, infiltration), the
dictionary's keys are the EnergyPlus IDF object type and its name to be created in tuple
format (object type in str, object name in str, object name cannot be the same as object
type), and the values to be detailed description or requirement of the IDF object, in
string format.
Only generate object type mentioned in commonly used internal load objects list attached
at the end of the prompt. Here is the user description: {user_description}

- Specific Requirements:
1. Extract schedule as much as possible from the description, use 'Schedulelimit' as the
schedule limit.
2. Create an 'Always_On' schedule, and assign it to object without specified schedule,
use 'Schedulelimit' as the schedule limit. In the dictionary, generate schedule object
first.
3. ALWAYS mention the exact schedule to be used in the detailed description
4. One object for ONLY one zone. Do NOT use one object to describe multiple zones.
5. ALWAYS mention what the exact zone name in the detailed description. The complete
zone list is: {zone_list_str}
6. For People Object, if not specified, mention in the description to use 'Always_On'
just for 'Activity Level Schedule Name'.
7. Extract as much information from the user description as possible in the detailed
description.

Commonly used internal load objects list: {common_internal_load_object_list}
"""
```

```
Agent_2_3 = f"""I want to process user description of their building to ONLY create HVAC
System related objects, the dictionary's keys are the EnergyPlus IDF object type and its
name to be created in tuple format (object type in str, object name in str, object name
cannot be the same as object type), and the values to be detailed description or
requirement of the IDF object, in string format.

Only generate object type mentioned in commonly used internal load objects list attached
at the end of the prompt. Here is the user description: {user_description}

- Specific Requirements:
1. ALWAYS mention the exact zone name list in the detailed description. The complete
zone list is: {zone_list_str}
2. ALWAYS create setpoint object before HVAC template object, and in HVAC template
object, mention what setpoint object is used.
3. HVAC system template name must use the same name as the zone name.

Commonly used internal load objects list: {common_HVAC_object_list}
"""
```

2.2.3 Single IDF Object Generator Suite

With Agent 2 extracting all objects and their natural-language descriptions in an organized and accurate manner, the subsequent step involves iterating over each object to generate individual IDF objects. Designed for this purpose, Agent 3 facilitates the actual generation of IDF objects one by one. Although it is collectively referred to as Agent 3, it comprises 923 different sub-agents, each corresponding to one of the total numbers of IDF object types. Agent 3 utilizes templates, enabling any type of object to use this framework to create its corresponding agent. As can be seen from the agent, it systematically structures the conversion process from natural language descriptions to formal IDF objects, ensuring that all required and optional fields are appropriately addressed and formatted in each IDF object.

The design logic of this agent is detailed as follows. Ideally, the building description should provide sufficient information for the IDF object creation, ensuring all mandatory values are available. However, in many real-world scenarios, building descriptions may not be detailed enough to fully support object creation, often resulting in numerous undefined values. Our strategy, therefore, involves using the IDD file (a template that defines the structure, data types, and allowed values for all input data used by EnergyPlus) to determine which fields can be assigned default values, similar to how the IDD file assumes default values for IDF objects. We ensure that these defaults are utilized; user-specified information is then integrated into the object. Finally, for those parameters that neither have default values nor are provided by the user, the agent will make assumptions or use placeholders to those fields. This completion clarifies the role of the agent, highlighting its systematic approach and its vital role in ensuring data integrity and completeness. The above decision process is defined by "Some general rules when generating the object" shown in the agent below.

In terms of the placeholders, {object_type} indicates the object type such as "People"; {object_name} indicates the name field of the object, such as "People_for_Zone_1"; {specific_requirement} is the extracted piece of information related to the natural language description of the object extracted from building description; {IDD_dict[object_type]} is the attached IDD file description providing the ground truth or rule of IDF object creation.

~~~
Agent_3 = f"""
   This agent automatically generates text-based idf object type '{object_type}' for
EnergyPlus Simulation. The object name is '{object_name}' (if None, meaning the object
does not require object name). The IDD file of {object_type} is provided in triple back-
ticks. IDD file defines the fields of the object, including their definition, data
type/format, mandatory or not, etc.

  The specific requirement for this object is: {specific_requirement}.

  Some general rules when generating the object:
  - Start with filling out default values as many as possible defined by IDD.
  - Fill out or replace the value that can be extracted from user's input.
  - If the field is not mandatory and is not defined by the specific requirement, please
do not put a value in the field.
  - If the field is mandatory but not defined by the specific requirement, please set
it to "TBD" + mm(month)dd(day)yy(year)hh(hour)mm(minute)sss(second in one decimal
places) such as TBD0501241203111.
  - Even though some fields are optional or not defined by the specific requirement,
keep them in the object definition.
  - In the generated object, comment after "!" for each field about why the value is set
or is left blank.

  First, generate the object in txt format. In the end after the object is generated,
tell the user:
  - Which fields are set by user.
  - Which fields are mandatory but not defined.
  - Which fields are set to their default values.
  - Which fields are left blank.

  ```
  {IDD_dict[object_type]}
  ```
~~~

```
  """
```

2.2.4 Debugging Agent

The Debugging Agent ensures the generated IDF file is error-free. If an error is detected, Agent 4 analyzes the .err file from the EnergyPlus simulation, indicated by the placeholder of "{severe_n_fatal_error_str}", to identify the problematic class and its object name in a list. Along with the current IDF file, which is attached in the end using placeholder "{current_IDF_file_str }", the agent can create the corrected IDF objects only for wrong objects in string format. The reason of just creating new IDF objects for the wrong objects is to reduce output token size. This correction process is repeated before proceeding to the next iteration. Iterations are crucial as EnergyPlus errors may not manifest simultaneously; some errors only appear after others have been resolved. The process concludes once the IDF file is verified to be error-free, marking the end of the workflow.

~~~
Agent_4 = f""" The following is the fatal or severe error message:
```{severe_n_fatal_error_str}```

Based on the current IDF file (attached in the end), create the corrected IDF objects only for the wrong objects in string format, and put it between ``` ```, do not include any other things or comments.

For each IDF object you modify, record its original information in a list of tuples [('object_type_1', 'object_name1'), ('object_type_2', 'object_name2')], and put it between ///[]///, do not include any other things or comments.

- Specific requirements:
1. The maximum number of IDF objects generated is 10, and the maximum length of the list should also be less than 10.
2. You must ensure that the other fault-free objects are strictly kept unchanged.

```{current_IDF_file_str}```
  """
~~~

3 Case Study and Results

This section provides comprehensive case studies on energy modeling of actual buildings using the developed agentic LLM workflow. It includes the energy modeling of iUnit, an experimental modular building, executed with commercial LLMs including GPT-4 and Gemini. The first case study evaluates the Agent 3 Single IDF Object Generator Suite to confirm its capability in generating fundamental elements for BEM. The second study examines the entire debugging process, critical for workflow usability. The third assesses the entire developed workflow, while the fourth tests the workflow with Gemini instead of GPT-4 to evaluate its reliability across different LLMs.

3.1 Building and LLM Settings

The building energy model to be built is based on a modular studio apartment experimental test facility, referred as the iUnit. The iUnit (floorplan shown in Figure 2) was located at the National Renewable Energy Laboratory (NREL) that can be moved indoor for guarded testing or outdoor for environmental testing. iUnit is used to conduct validation-grade energy simulation experiments. iUnit has 378 square feet with three thermal zones, equipped with a packaged terminal air conditioner (PTAC). The construction, internal loads, and HVAC systems are detailed in Section 3.2. The building energy simulation engine for iUnit is EnergyPlus 22.2.0.

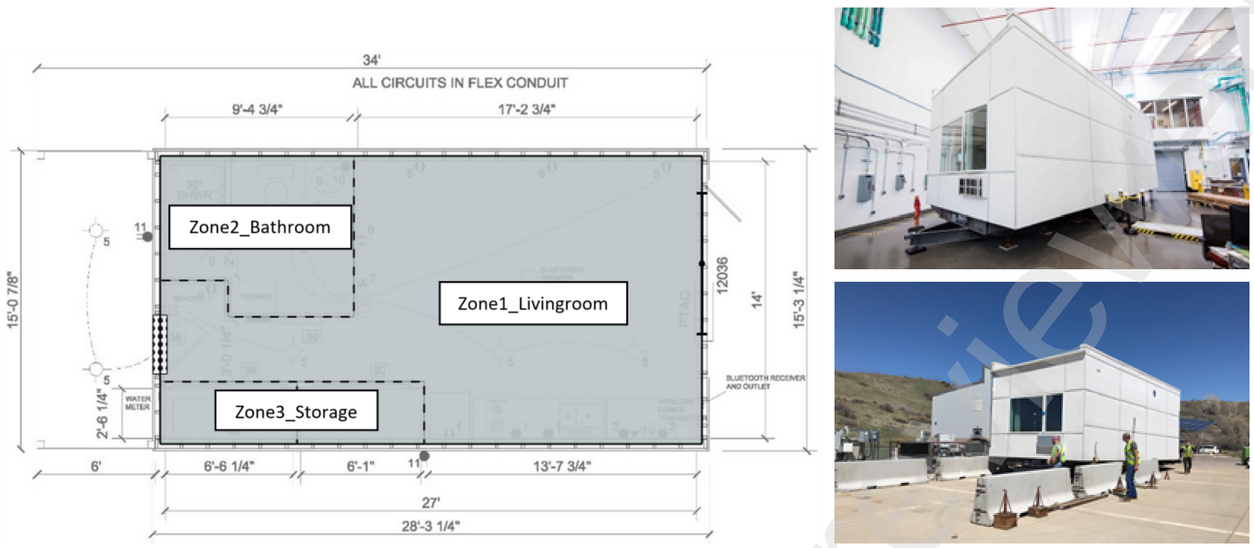


Figure 2. iUnit floor plan and appearance

We utilize three GPT models through API: 1) GPT-4 model gpt-4-0613, 2) GPT-4 model gpt-4-turbo-2024-04-09, and 3) Gemini 1.0 Pro. GPT-4-0613 is an advanced iteration of OpenAI's Generative Pre-trained Transformer models, captured on June 13th, 2023, and optimized for understanding and generating human-like text across a wide range of topics and contexts, with enhanced support for function calling. GPT-4-Turbo-2024-04-09 is a refined version of OpenAI's Generative Pre-trained Transformer models, captured on April 9th, 2024, and enhanced for rapid response and high-accuracy performance in generating and understanding human-like text across diverse subjects and contexts. Gemini-1.0-Pro is a sophisticated LLM from Google AI, excelling at a wide range of tasks, including understanding complex instructions, generating creative text formats, and providing informative summaries. The case study will use these LLMs for agents and compare their performance in Section 3.4. We use LangChain, a Python library designed for building and orchestrating AI applications (especially LLM applications), as the key infrastructure to build agents, connects agents, and evaluate agents.

3.2 Detailed Building Description as Workflow Input

The description describes: 1) the material and construction, 2) internal load, and 3) HVAC system. The goal of the developed LLM-based workflow is to automate the modeling process with minimum human participation and finally generate an error-free EnergyPlus input file, or IDF file.

```
Project 1: EnergyPlus Modeling of iUnit
1. Project Description
The goal of Project 1 is to create EnergyPlus Model of iUnit based on the information
and measurement given in this document. iUnit is a modular apartment located at the
campus of National Renewable Energy Laboratory (NREL) that can be moved indoor for
guarded testing or outdoor for environmental testing. NREL is using iUnit to conduct
validation-grade energy simulation experiments.

2. iUnit Details
2.3. Constructions
2.3.1. Floor
Table 1. Details on floor construction layers
  Floor Layer 1 (Outside): Rodent Barrier (HDPE)
   Roughness: Smooth; Thickness: 0.05 in; Conductivity: 3.472 Btu·in/hr·ft2·R; Density:
59.3 lb/ft3; Specific Heat: 0.4347 Btu/lb·R
```

Floor Layer 2: Insulated Joist Layer
Roughness: Smooth: Thickness: 14.462 in; Conductivity: 0.3323 Btu·in/hr·ft2·R; Density: 6.336 lb/ft3; Specific Heat: 0.2357 Btu/lb·R
Floor Layer 3: OSB
Roughness: Smooth: Thickness: 1.75 in; Conductivity: 0.826 Btu·in/hr·ft2·R; Density: 33.96 lb/ft3; Specific Heat: 0.289 Btu/lb·R
Floor Layer 4 (Inside): Plywood
Roughness: Smooth; Thickness: 0.25 in; Conductivity: 0.8 Btu·in/hr·ft2·R; Density: 33.96 lb/ft3; Specific Heat: 0.289 Btu/lb·R

2.3.2. Roof
Table 2. Details on roof construction layers
Roof Layer 1 (Outside): EPDM Rubber
Roughness: Smooth; Thickness: 0.1 in; Conductivity: 1.3889 Btu·in/hr·ft2·R; Density: 63 lb/ft3; Specific Heat: 0.48 Btu/lb·R; Solar Absorptance:0.3
Roof Layer 2: OSB
Roughness: Smooth; Thickness 0.5 in; Conductivity: 0.8264 Btu·in/hr·ft2·R; Density: 33.96 lb/ft3; Specific Heat: 0.289 Btu/lb·R
Roof Layer 3: Insulted Truss Layer
Roughness: Smooth; Thickness 37.375 in; Conductivity: 0.3425 Btu·in/hr·ft2·R; Density 3.296 lb/ft3; Specific Heat: 0.1900 Btu/lb·R
Roof Layer 4 (Inside): Drywall
Roughness: MediumSmooth; Thickness: 0.625 in; Conductivity: 1.1111 Btu·in/hr·ft2·R; Density: 49.94 lb/ft3; Specific Heat: 0.26034 Btu/lb·R

2.3.3. Exterior Wall
Table 3. Details on Exterior wall construction layers
Wall Layer 1 (Outside): Exterior Finish OSB
Roughness: Smooth; Thickness: 0.625 in; Conductivity: 0.8264 Btu·in/hr·ft2·R; Density: 33.96 lb/ft3; Specific Heat: 0.289 Btu/lb·R; Solar Absorptance: 0.5
Wall Layer 2: Rigid Insulation
Roughness: Smooth; Thickness: 1 in; Conductivity: 0.2 Btu·in/hr·ft2·R; Density: 2.2786 lb/ft3; Specific Heat: 0.358 Btu/lb·R
Wall Layer 3: OSB
Roughness: Smooth; Thickness 0.5 in; Conductivity: 0.8264 Btu·in/hr·ft2·R; Density: 33.96 lb/ft3; Specific Heat: 0.289 Btu/lb·R
Wall Layer 4: Stud Layer
Roughness: Smooth; Thickness 5.5 in; Conductivity: 0.2927 Btu·in/hr·ft2·R; Density 7.541 lb/ft3; Specific Heat 0.2573 Btu/lb·R
Wall Layer 5 (Inside): Drywall
Roughness: MediumSmooth; Thickness: 0.625 in; Conductivity: 1.111 Btu·in/hr·ft2·R; Density: 49.94 lb/ft3; Specific Heat: 0.26034 Btu/lb·R

2.3.4. Interior Wall
Interior Wall Layer 1 (Inside): Plywood_2
Roughness: Smooth; Thickness: 1 in; Conductivity: 0.8 Btu·in/hr·ft2·R; Density: 33.96 lb/ft3; Specific Heat: 0.289 Btu/lb·R

2.3.5. Window
U-factor = 0.223 Btu/ft2·hr·R
Solar Heat Gain Coefficient = 0.43
Visible Transmittance = 0.42

2.5.6. Door
· Insulated steel exterior door
· Conductivity = 0.20408 BTU in/ft2 F h
· Density of Expanded Polystyrene: 24.0 kg/m^3
· Specific heat of Expanded Polystyrene: 1210.0 J/kg*K

2.4. Internal Loads
2.4.1. People
Internal load "People" strictly follows the National Renewable Energy Laboratory's business hours, 1) Monday–Thursday, 8 a.m.–5 p.m., 2) Friday, 8 a.m.–4 p.m., 3) Closed

```
on Saturday and Sunday. There are on average two persons in iUnit during business hours
and no people in time of the rest.

2.4.2. Lighting
Assume lighting capacity by yourself since it is not measured. This kind of assumption
of missing values is very common in EnergyPlus. Refer to ASHRAE standard. The lighting
on/off schedule is assumed to be the same as the business hours.

2.4.3. Electric Equipment
Assume the capacity by density.

2.5 Infiltration
The infiltration will be defined using IDF object ZoneInfiltration:DesignFlowRate. In
this object, according to the onsite measurements, I = 1, Fschedule = always on
(1,1,1,1,…), A = 0.03, B = 0.003, C = 0, D = 0. Apply the infiltration to all zones.

2.6 HVAC System
Use the following HVAC template to design your own HVAC system for iUnit.
- Packaged terminal air conditioner (PTAC) systems with optional hot water boiler
The setpoints of each zone are listed as follows:
- Zone1_Livingroom is conditioned with dual cooling and heating setpoint. The cooling
setpoint is 76F and the heating setpoint is 72F.
- Zone2_Bathroom is conditioned with a single heating setpoint. The heating setpoint is
74F.
- Zone3_Storage is an unconditioned room without any setpoints.
```

3.3 Test: Agent 2 IDF Object Type and Description Extraction

Agent 2 methodically extracts and itemizes all potential IDF objects with natural language description from the building description introduced in Section 3.2 to ensure no objects are overlooked. Among Agent 2, Agent 2.1 generates materials and construction items. Agent 2.2 generates internal loads items. Agent 2.3 generates HVAC items. The itemized IDF objects from Agent 2.1, Agent 2.2, and Agent 2.3 are listed in Table 1, Table 2, and Table 3, respectively.

Table 1. Itemized IDF objects related to materials and construction generated by Agent 2.1 with natural language description from the building description

No.	Object Type	Object Name	Natural Language Description
1	Material	Rodent_Barrier_HDPE	Roughness: Smooth; Thickness: 1.27 mm; Conductivity: 0.607 W/m·K; Density: 949.3 kg/m3; Specific Heat: 1819 J/kg·K
2	Material	Insulated_Joist_Layer	Roughness: Smooth: Thickness: 367.3 mm; Conductivity: 0.0461 W/m·K; Density: 101.6 kg/m3; Specific Heat: 986 J/kg·K
3	Material	OSB	Roughness: Smooth: Thickness: 44.45 mm; Conductivity: 0.1147 W/m·K; Density: 544 kg/m3; Specific Heat: 1209 J/kg·K
4	Material	Plywood	Roughness: Smooth; Thickness: 6.35 mm; Conductivity: 0.111 W/m·K; Density: 544 kg/m3; Specific Heat: 1209 J/kg·K
5	Construction	Floor_Construction	Layers: Rodent_Barrier_HDPE, Insulated_Joist_Layer, OSB, Plywood
6	Material	EPDM_Rubber	Roughness: Smooth; Thickness: 2.54 mm; Conductivity: 0.243 W/m·K; Density: 1009 kg/m3; Specific Heat: 2008 J/kg·K; Solar Absorptance:0.3
7	Material	Insulted_Truss_Layer	Roughness: Smooth; Thickness 949.35 mm; Conductivity: 0.0476 W/m·K; Density 52.8 kg/m3; Specific Heat: 794 J/kg·K
8	Material	Drywall	Roughness: MediumSmooth; Thickness: 15.875 mm; Conductivity: 0.194 W/m·K; Density: 800 kg/m3; Specific Heat: 1089 J/kg·K
9	Construction	Roof_Construction	Layers: EPDM_Rubber, OSB, Insulted_Truss_Layer, Drywall
10	Material	Exterior_Finish_OSB	Roughness: Smooth; Thickness: 15.875 mm; Conductivity: 0.1148 W/m·K; Density: 544 kg/m3; Specific Heat: 1209 J/kg·K; Solar Absorptance: 0.5
11	Material	Rigid_Insulation	Roughness: Smooth; Thickness: 25.4 mm; Conductivity: 0.0349 W/m·K; Density: 36.5 kg/m3; Specific Heat: 1497 J/kg·K
12	Material	Stud_Layer	Roughness: Smooth; Thickness 139.7 mm; Conductivity: 0.0407 W/m·K; Density 120.6 kg/m3; Specific Heat: 1076 J/kg·K

13	Construction	Exterior_Wall_Construction	Layers: Exterior_Finish_OSB, Rigid_Insulation, OSB, Stud_Layer, Drywall
14	Material	Plywood_2	Roughness: Smooth; Thickness: 25.4 mm; Conductivity: 0.111 W/m·K; Density: 544 kg/m3; Specific Heat: 1209 J/kg·K
15	Construction	Interior_Wall_Construction	Layers: Plywood_2
16	WindowMaterial:SimpleGlazingSystem	Window_Material	U-factor = 1.26 W/m2·K; Solar Heat Gain Coefficient = 0.43; Visible Transmittance = 0.42
17	Construction	window_construction_1	Layers: Window_Material
18	Material	Insulated_steel_door	Conductivity = 0.356 W/m·K; Density of Expanded Polystyrene: 24.0 kg/m^3; Specific heat of Expanded Polystyrene: 1210.0 J/kg*K
19	Construction	Door_Construction	Layers: Insulated_steel_door

Table 2. Itemized IDF objects related to internal loads generated by Agent 2.2 with natural language description from the building description

No.	Object Type	Object Name	Natural Language Description
1	Schedule:Compact	Business_Hours	Schedule for business hours: Monday-Thursday 8 a.m.-5 p.m., Friday 8 a.m.-4 p.m., Closed on Saturday and Sunday. Use 'ScheduleLimit' as the schedule limit
2	Schedule:Compact	Always_On	Schedule that is always on (1,1,1,1,...). Use 'ScheduleLimit' as the schedule limit
3	People	People_iUnit	Average of two persons during business hours, no people at other times. Apply to all zones: ['Zone1_Livingroom', 'Zone2_Bathroom', 'Zone3_Storage']. Schedule: 'Business_Hours'. Activity Level Schedule Name should use 'Always_On'
4	Lights	Lights_iUnit	Lighting capacity assumed based on ASHRAE standard. Apply to all zones: ['Zone1_Livingroom', 'Zone2_Bathroom', 'Zone3_Storage']. Schedule: 'Business_Hours'
5	ElectricEquipment	Equip_iUnit	Electric equipment capacity assumed by density. Apply to all zones: ['Zone1_Livingroom', 'Zone2_Bathroom', 'Zone3_Storage']. Schedule: 'Always_On'
6	ZoneInfiltration:DesignFlowRate	Infiltration_iUnit	Defined using the formula I = 1, Fschedule = always on (1,1,1,1,...), A = 0.03, B = 0.003, C = 0, D = 0. Apply to all zones: ['Zone1_Livingroom', 'Zone2_Bathroom', 'Zone3_Storage']. Schedule: 'Always_On'

Table 3. Itemized IDF objects related to HVAC template generated by Agent 2.3 with natural language description from the building description

No.	Object Type	Object Name	Natural Language Description
1	HVACTemplate:Thermostat	Thermostat_Livingroom	ThermostatDual setpoint thermostat for Zone1_Livingroom with cooling setpoint at 24.4°C and heating setpoint at 22.2°C
2	HVACTemplate:Thermostat	Thermostat_Bathroom	Single setpoint thermostat for Zone2_Bathroom with a heating setpoint at 23.3°C
3	HVACTemplate:Zone:PTAC	PTAC_Livingroom	Packaged Terminal Air Conditioner system for Zone1_Livingroom with dual setpoint control, cooling setpoint at 24.4°C and heating setpoint at 22.2°C
4	HVACTemplate:Zone:PTAC	PTAC_Bathroom	Packaged Terminal Air Conditioner system for Zone2_Bathroom with a single heating setpoint at 23.3°C

3.4 Test: Agent 3 Single IDF Object Generator Suite

Agent 3 Single IDF Object Generator Suite facilitates the generation of single IDF objects one by one based on the extracted itemized IDF object information from Agent 2 (listed in Table 1, Table 2, and Table 3). Agent comprises 923 different sub-agents, each corresponding to one of the total numbers of IDF object types. Agent 3 utilizes templates, enabling any type of object to use this framework to create its corresponding agent. We will test whether it systematically structures the conversion process from natural language descriptions to formal IDF objects, ensuring that all required and optional fields are appropriately addressed and formatted in each IDF object. The following results show its conversion of the description of occupancy to “People” IDF object named “People_iUnit”. The results show that it accurately generates the object based on the requirement (“Natural Language Description” column in Table 1, Table 2, and Table 3), and also follow the rules we defined in the prompt template about default values settings and mandatory value settings.

```
People,
  People_iUnit, !- Name
  Zone1_Livingroom, !- Zone or ZoneList or Space or SpaceList Name
  Business_Hours, !- Number of People Schedule Name
  People, !- Number of People Calculation Method
  2, !- Number of People
  , !- People per Floor Area
  , !- Floor Area per Person
  0.3, !- Fraction Radiant
  , !- Sensible Heat Fraction
  Always_On, !- Activity Level Schedule Name
  3.82E-8, !- Carbon Dioxide Generation Rate
  No, !- Enable ASHRAE 55 Comfort Warnings
  ZoneAveraged, !- Mean Radiant Temperature Calculation Type
  , !- Surface Name/Angle Factor List Name
  , !- Work Efficiency Schedule Name
  ClothingInsulationSchedule, !- Clothing Insulation Calculation Method
  , !- Clothing Insulation Calculation Method Schedule Name
  , !- Clothing Insulation Schedule Name
  , !- Air Velocity Schedule Name
  , !- Thermal Comfort Model 1 Type
  , !- Thermal Comfort Model 2 Type
  , !- Thermal Comfort Model 3 Type
  , !- Thermal Comfort Model 4 Type
  , !- Thermal Comfort Model 5 Type
  , !- Thermal Comfort Model 6 Type
  , !- Thermal Comfort Model 7 Type
  , !- Ankle Level Air Velocity Schedule Name
  15.56, !- Cold Stress Temperature Threshold
  30; !- Heat Stress Temperature Threshold
```

The following result shows the “Material” object named “Plywood”, which is correctly generated.

```
Material,
  Plywood, !- Name
  Smooth, !- Roughness
  0.00635, !- Thickness {m}
  0.111, !- Conductivity {W/m-K}
  544, !- Density {kg/m3}
  1209, !- Specific Heat {J/kg-K}
  0.9, !- Thermal Absorptance
  , !- Solar Absorptance
  ; !- Visible Absorptance
```

The result shows the “HVACTemplate:Zone:PTAC” object for “Zone1_Livingroom”, which is also correctly generated.

```
HVACTemplate:Zone:PTAC,
  Zone1_Livingroom, !- Zone Name
  Thermostat_Livingroom, !- Template Thermostat Name
  autosize, !- Cooling Supply Air Flow Rate
  autosize, !- Heating Supply Air Flow Rate
  , !- No Load Supply Air Flow Rate
  , !- Zone Heating Sizing Factor
  , !- Zone Cooling Sizing Factor
  Flow/Person, !- Outdoor Air Method
  0.00944, !- Outdoor Air Flow Rate per Person
  0.0, !- Outdoor Air Flow Rate per Zone Floor Area
  0.0, !- Outdoor Air Flow Rate per Zone
  , !- System Availability Schedule Name
  , !- Supply Fan Operating Mode Schedule Name
  DrawThrough, !- Supply Fan Placement
  0.7, !- Supply Fan Total Efficiency
  75, !- Supply Fan Delta Pressure
  0.9, !- Supply Fan Motor Efficiency
  SingleSpeedDX, !- Cooling Coil Type
  , !- Cooling Coil Availability Schedule Name
  autosize, !- Cooling Coil Gross Rated Total Capacity
  autosize, !- Cooling Coil Gross Rated Sensible Heat Ratio
  3.0, !- Cooling Coil Gross Rated Cooling COP
  Electric, !- Heating Coil Type
  , !- Heating Coil Availability Schedule Name
  autosize, !- Heating Coil Capacity
  0.8, !- Gas Heating Coil Efficiency
  0.0, !- Gas Heating Coil Parasitic Electric Load
  , !- Dedicated Outdoor Air System Name
  SupplyAirTemperature, !- Zone Cooling Design Supply Air Temperature Input Method
  14.0, !- Zone Cooling Design Supply Air Temperature
  11.11, !- Zone Cooling Design Supply Air Temperature Difference
  SupplyAirTemperature, !- Zone Heating Design Supply Air Temperature Input Method
  50.0, !- Zone Heating Design Supply Air Temperature
  30.0, !- Zone Heating Design Supply Air Temperature Difference
  , !- Design Specification Outdoor Air Object Name
  , !- Design Specification Zone Air Distribution Object Name
  None, !- Baseboard Heating Type
  , !- Baseboard Heating Availability Schedule Name
  autosize, !- Baseboard Heating Capacity
  None; !- Capacity Control Method
```

Due to the multiple objects to create (a total of 29. See Table 1, Table 2, and Table 3) and limited space for the paper, the paper does not show the rest of the objects in detail. All of the 29 objects are created successfully without errors. Only one “Material” object called “Insulated_steel_door” has a “TBD” value (to be decided value) for the material thickness. This is required by the Agent 3 that when any mandatory information is missing, “TBD” value should be assigned to it. In the building description, no thickness information is provided, so this value is set to “TBD.” Missing mandatory information is a very common problem in BEM. In order to make the object run without errors, Agent 4 will take care of all the “TBD” values based on the reasonable assumption or user’s input, which will be introduced in Section 3.5.

```
Material,
  Insulated_steel_door, !- Name
  MediumRough, !- Roughness, default value used as not specified
  TBD0501241203111, !- Thickness, set to 'TBD' as it is a required field but not specified
  0.0295, !- Conductivity, set by user
  24.0, !- Density, set by user
  1210.0, !- Specific Heat, set by user
  0.9, !- Thermal Absorptance, default value used as not specified
  0.7, !- Solar Absorptance, default value used as not specified
  0.7; !- Visible Absorptance, default value used as not specified
```

3.5 Test: Agent 4 Debugging Procedure

In Agent 4, we primarily focus on two major error types: 1) reference errors and 2) value errors, including missing values. Although there are many different error types in EnergyPlus modeling, this paper concentrates on these two common error types within the Agent 4 workflow. A similar approach can be extended to address those errors as well.

3.5.1 Reference Error

Reference errors are a common issue due to the hierarchical structure of IDF objects. For instance, an internal load object must link to a schedule object, and an HVAC template object requires a setpoint object. Although no reference errors were found in the modeling task for the case study, aside from the "TBD" issue outlined in Section 3.4, we conducted an independent test. This involved intentionally creating incorrect reference field values in IDF objects to check if Agent 4 could correct them. A clear example is provided below: a building surface must correctly refer to a construction name. There was a discrepancy between “R13_WALL” and “R13WALL.” Agent 4 successfully corrected “R13WALL” to “R13_WALL,” ensuring consistency.

```
Construction,
    R13_WALL,                  !- Name
    R13LAYER;                 !- Outside Layer

BuildingSurface:Detailed,
    Zn001:Wall002,            !- Name
    Wall,                     !- Surface Type
    R13WALL,                  !- Construction Name
    Main Zone,                !- Zone Name
    ,                         !- Space Name
    Outdoors,                 !- Outside Boundary Condition
    ,                         !- Outside Boundary Condition Object
    SunExposed,               !- Sun Exposure
    WindExposed,              !- Wind Exposure
    0.5000000,                !- View Factor to Ground
    4,                        !- Number of Vertices
    15.24000,0,4.572000,  !- X,Y,Z ==> Vertex 1 {m}
    15.24000,0,0,  !- X,Y,Z ==> Vertex 2 {m}
    15.24000,15.24000,0,  !- X,Y,Z ==> Vertex 3 {m}
    15.24000,15.24000,4.572000;  !- X,Y,Z ==> Vertex 4 {m}
```

3.5.2 Object Value Error

Although there are no incorrect object value errors in the modeling task for the case study, such as using numerical values in categorical fields where a string is required, we developed a simple case study based on the official EnergyPlus example IDF file, “1ZoneDataCenterCRAC_wApproachTemp.idf”. This involved intentionally changing the correct value type to an incorrect one. Specifically, we altered the 'Begin Month' from 1 to 'kk', an erroneous data type. Agent 4 successfully corrected this from 'kk' to 1. Similarly, another correction was made from 'abc' (a string) to '0.0' (a float).

```
RunPeriod,
    Jan,                      !- Name
    kk,                        !- Begin Month
    1,                        !- Begin Day of Month
    ,                         !- Begin Year
    1,                        !- End Month
    31,                       !- End Day of Month
    ,                         !- End Year
    Tuesday,                  !- Day of Week for Start Day
    Yes,                      !- Use Weather File Holidays and Special Days
    Yes,                     !- Use Weather File Daylight Saving Period
    No,                      !- Apply Weekend Holiday Rule
    Yes,                     !- Use Weather File Rain Indicators
    Yes;                     !- Use Weather File Snow Indicators

Building,
    Bldg,                    !- Name
    abc,                     !- North Axis {deg}
    Suburbs,                 !- Terrain
    0.05,                    !- Loads Convergence Tolerance Value {W}
    0.05,                    !- Temperature Convergence Tolerance Value {deltaC}
    MinimalShadowing,        !- Solar Distribution
    30,                      !- Maximum Number of Warmup Days
    6;                       !- Minimum Number of Warmup Days
```

3.6 Test: the Developed Agentic Workflow Compared against a Single Prompt

In this section, we demonstrate the necessity of an agentic workflow by comparing it with a single-prompt approach. We conducted 10 tests of the entire developed agentic workflow and observed the success rate in generating error-free and correct IDF files. As shown in Table 4, all 10 trials using the developed workflow successfully produced accurate and error-free IDF files, achieving a 100% success rate. However, due to the inherent self-consistency issues of LLMs, the models generated prior to the intervention of debugging agents varied. The "Erroneous object(s) before debugging agent" column in Table 4 lists the different errors encountered in each run, including: 1) zero door thickness, as this detail was omitted from the building description; 2) incorrect HVAC template setpoint references; 3) construction-material reference errors; 4) incorrect activity level schedule references in the "People" object; and 5) failure to create an "Always_On" schedule. Despite these variations, all errors were effectively resolved by Agent 4, the debugging agent. These results emphasize the effectiveness of a well-designed LLM agentic workflow in addressing self-consistency issues. While it cannot ensure uniformity and accuracy at every step, it can reliably produce accurate final outputs through a continuous feedback loop. Additionally, we assessed the time efficiency of the LLM agentic workflow compared to two human modelers: a student new to BEM (who takes BEM class and the modeling of iUnit is his final project) and an experienced modeler who actually built the official iUnit model. The student required two weeks to construct and debug the model, with debugging accounting for more than two thirds of this duration. The experienced modeler completed the model in one day. In contrast, the agentic workflow completed the model in 9 minutes, with 6 minutes allocated to IDF generation and 3 minutes to error correction.

Table 4. Test results of the developed agentic workflow

Run No.	Erroneous object(s) before debugging agent	Error-free after debugging?	Accurate IDF?
1	Door thickness is 0	True	True
2	Door thickness is 0, HVAC template setpoint reference error	True	True
3	Door thickness is 0	True	True
4	HVAC template setpoint reference error	True	True
5	Construction-material reference error	True	True
6	People object activity level schedule reference error	True	True
7	"Always_On" schedule is not created	True	True
8	People object activity level schedule reference error	True	True
9	Door thickness is 0	True	True
10	Door thickness is 0, People object activity level schedule reference error	True	True

Second, we compare the whole developed agentic workflow with the single prompt method without agentic workflow. The single prompt is shown as follows and its diagram is shown in Figure 3.

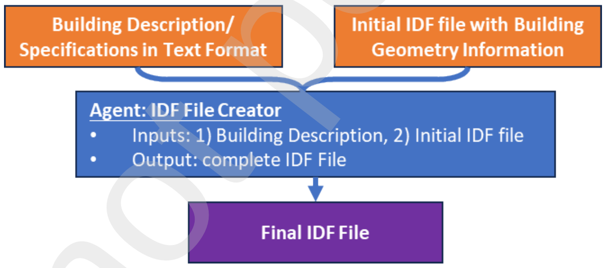
```
Single_Prompt = f"""

You are an EnergyPlus modeling expert, and you will use 1) the provided building information described in a sectioned natural language, as well as 2) an initial IDF file with building geometry information, to create an error-free IDF file in text format.

Here is the building description: {building_description}

Here is the initial IDF file in text format: {initial_IDF}"""
```

In terms of the results, there are altogether 33 errors in this IDF file. The only accurate part of the IDF file is the material and construction creation. The most sever issues include: 1) sever syntax error of single IDF objects not following IDD format (as shown in the "People" object below), 2) no schedule created and paradox schedule reference (as shown in the "People" object below, no "Office Occupancy" schedule is created), 3) casually putting numbers to fields (as shown in the "HVACTemplate:Zone:PTAC" object below). The performance is far less accurate than the agentic workflow, indicating the necessity of splitting each task into sub-tasks. The long context using a single prompt distracts the IDF creation task and leads to poor performance. More discussions can be found in Section 4. It is also worth mentioning that we are using GPT-4 model "gpt-4-turbo-2024-04-09" as the foundation model because it allows maximum 128,000 tokens to deal with the long input and output length and "GPT-4-0613" only allows 8,192 tokens which cannot be used for this task. Looking back at the agentic workflow, it has less limitations on the token limit, making this method more adaptable to more foundation models.



Building Description/ Specifications in Text Format
Initial IDF file with Building Geometry Information
Agent: IDF File Creator
Inputs: 1) Building Description, 2) Initial IDF file
Output: complete IDF File
Final IDF File

Figure 3. Prompt engineering without LLM Planning

```
People,
  iUnit People,                     !- Name
  Zone1_Livingroom,                 !- Zone or ZoneList Name
  Office Occupancy,                 !- Number of People Schedule Name
  people,                           !- Number of People Calculation Method
  2,                                !- Number of People
  ,                                 !- People per Zone Floor Area {person/m2}
  ,                                 !- Zone Floor Area per Person {m2/person}
  0.3,                              !- Fraction Radiant
  ,                                 !- Sensible Heat Fraction
  Activity Level Schedule;          !- Activity Level Schedule Name

HVACTemplate:Zone:PTAC,
  Zone1_Livingroom,                 !- Zone Name
  Thermostat Schedule,              !- Template Thermostat Name
  autosize,                         !- Cooling Supply Air Flow Rate {m3/s}
  autosize,                         !- Heating Supply Air Flow Rate {m3/s}
  ,                                 !- No Load Supply Air Flow Rate {m3/s}
  PackagedTerminalAirConditioner,   !- Zone Cooling Design Supply Air Temperature Input
Method
```

```
  14,                              !- Zone Cooling Design Supply Air Temperature {C}
  ,                         !- Zone Cooling Design Supply Air Temperature Difference {deltaC}
  PackagedTerminalAirConditioner,  !- Zone Heating Design Supply Air Temperature Input
Method
  50,                              !- Zone Heating Design Supply Air Temperature {C}
  ,                         !- Zone Heating Design Supply Air Temperature Difference {deltaC}
  None,                            !- Zone Cooling Sizing Factor
  None,                            !- Zone Heating Sizing Factor
  Sum,                             !- Outdoor Air Method
  0.00944,                         !- Outdoor Air Flow Rate per Person {m3/s-person}
  0.01,                          !- Outdoor Air Flow Rate per Zone Floor Area {m3/s-m2}
  0.1,                             !- Outdoor Air Flow Rate per Zone {m3/s}
  DesignDay,                       !- System Availability Schedule Name
  Fan:OnOff,                       !- Supply Fan Operating Mode Schedule Name
  DrawThrough,                     !- Supply Fan Placement
  0.7,                             !- Supply Fan Total Efficiency
  500,                             !- Supply Fan Delta Pressure {Pa}
  0.9,                             !- Supply Fan Motor Efficiency
  SingleSpeedDX,                   !- Cooling Coil Type
  ,                                !- Cooling Coil Availability Schedule Name
  autosize,                        !- Cooling Coil Rated Capacity {W}
  autosize,                        !- Cooling Coil Rated Sensible Heat Ratio
  3,                               !- Cooling Coil Rated COP {W/W}
  Electric,                        !- Heat Pump Heating Coil Type
  ,                                !- Heat Pump Heating Coil Availability Schedule Name
  autosize,                        !- Heat Pump Heating Coil Rated Capacity {W}
  2.75,                            !- Heat Pump Heating Coil Rated COP {W/W}
  -8,                      !- Heat Pump Heating Minimum Outdoor Dry-Bulb Temperature {C}
  5,                       !- Heat Pump Defrost Maximum Outdoor Dry-Bulb Temperature {C}
  ReverseCycle,                    !- Heat Pump Defrost Strategy
  Timed,                           !- Heat Pump Defrost Control
  0.058333,                        !- Heat Pump Defrost Time Period Fraction
  Electric,                        !- Supplemental Heating Coil Type
  ,                             !- Supplemental Heating Coil Availability Schedule Name
  autosize,                        !- Supplemental Heating Coil Capacity {W}
  21,              !- Supplemental Heating Coil Maximum Outdoor Dry-Bulb Temperature {C}
  0.8,                             !- Supplemental Heating Coil Fan Placement
  50,                              !- Supply Plenum Name
  ,                                !- Return Plenum Name
  None,                            !- Baseboard Heating Type
  ,                                !- Baseboard Heating Availability Schedule Name
  autosize,                        !- Baseboard Heating Capacity {W}
  MinimumFlowWithBypass;           !- Damper Heating Action
```

3.7 Test: the Developed Agentic Workflow using Gemini Model

In the current case study, all agents operate using GPT-4 Model gpt-4-0613 or gpt-4-turbo-2024-04-09, as introduced in Section 2.2. For this experiment, we employ the Gemini 1.0 Pro model to re-execute all components. However, it was observed that the Gemini model fails to successfully operate Agent 3, specifically in generating single objects correctly. The results, demonstrated in the subsequent sections, reveal that the generated objects do not conform to the appropriate syntactical structure, rendering them unusable. This issue may stem from Gemini's lack of training on IDF files. This emphasizes the critical dependence of agent performance on the underlying foundation model. Consequently, both the prompt structure and the foundation model significantly influence an agent's performance and should be concurrently considered during the agent design stage.

```
A1 ,    \field Name
! Set by user
People_iUnit
```

```
A2 ,   \field Zone or ZoneList or Space or SpaceList Name
! Set by user
['Zone1_Livingroom', 'Zone2_Bathroom', 'Zone3_Storage']
A3 ,   \field Number of People Schedule Name
! Set by user
Business_Hours
A4 ,   \field Number of People Calculation Method
! Set to default value
People
N1 ,   \field Number of People
! Set by user
2
N2 ,   \field People per Floor Area
! Left blank
N3 ,   \field Floor Area per Person
! Left blank
N4 ,   \field Fraction Radiant
! Set to default value
0.3
N5,  \field Sensible Heat Fraction
! Set to default value
autocalculate
A5 ,   \field Activity Level Schedule Name
! Set by user
Always_On
N6 ,   \field Carbon Dioxide Generation Rate
! Set to default value
3.82E-8
A6 ,   \field Enable ASHRAE 55 Comfort Warnings
! Set to default value
No
A7 ,   \field Mean Radiant Temperature Calculation Type
! Set to default value
ZoneAveraged
A8 ,   \field Surface Name/Angle Factor List Name
! Left blank
A9 ,   \field Work Efficiency Schedule Name
! Left blank
A10,   \field Clothing Insulation Calculation Method
! Set to default value
ClothingInsulationSchedule
A11,   \field Clothing Insulation Calculation Method Schedule Name
! Left blank
A12,   \field Clothing Insulation Schedule Name
! Left blank
A13,   \field Air Velocity Schedule Name
! Left blank
A14,   \field Thermal Comfort Model 1 Type
! Left blank
A15,   \field Thermal Comfort Model 2 Type
! Left blank
A16,   \field Thermal Comfort Model 3 Type
! Left blank
A17,   \field Thermal Comfort Model 4 Type
! Left blank
A18,   \field Thermal Comfort Model 5 Type
! Left blank
A19,   \field Thermal Comfort Model 6 Type
! Left blank
A20,   \field Thermal Comfort Model 7 Type
! Left blank
A21,   \field Ankle Level Air Velocity Schedule Name
! Left blank
```

```
N7,  \field Cold Stress Temperature Threshold
! Set to default value
15.56
N8;  \field Heat Stress Temperature Threshold
! Set to default value
30
```

3.8. Limitations of Case Studies

The case studies presented in this paper are subject to several limitations that should be considered. First, each case study begins with the assumption that the model geometry is already established. This presupposition may limit the generalizability of our findings, as the initial conditions are not derived from the LLM-based workflow. Additionally, to simplify the complexity of the system in the case studies, only HVAC templates were utilized. This approach may not capture the full intricacies of more customized or advanced HVAC system configurations that are often required in real-world scenarios. In fact, detailed modeling of HVAC systems presents significant challenges, even for experienced practitioners. For example, an error as simple as mistakenly size a water loop pump 100 times too large can lead to fatal errors due to temperature being too high. Such errors, although they appear straightforward, require a deep understanding of the system's physical dynamics to diagnose and resolve effectively. Another significant limitation is the lack of user interaction; the current model operates without further engagement with the user (such an interaction can also be represented as a separate Agent), which could be necessary for refining outputs and integrating user-specific requirements. Finally, although the methods used for evaluation and validation of the models are robustly designed for the specific building of iUnit, a more comprehensive validation approach across more diverse building and modeling cases would be needed to strengthen the findings and ensure their applicability in practical applications.

4 Discussion

4.1 Advantage of Agentic Workflow

As indicated by the results in Section 3.6, the agentic workflow is essential to EnergyPlus modeling as a complex engineering modeling process. LLMs often struggle with increasingly complex multi-step tasks when using a single prompt due to several theoretical constraints. Firstly, these models operate within a finite context window, which limits their ability to consider all necessary details as task complexity increases. Additionally, the higher the number of tokens, the harder it is to search for the "needle in the hay" in terms of special requirements, context details, and complex instructions. Secondly, they lack an inherent understanding of the tasks, merely generating text based on statistical likelihoods from their training data, which might not include sufficient examples of similar complex tasks. This leads to difficulties in handling ambiguity and implicit knowledge that are often crucial in multi-step operations. Furthermore, without persistent memory or external databases, they cannot maintain or refer back to the previous steps within a task, complicating the management of sequential dependencies and decision-making processes required for complex tasks. Thus, as complexity grows, the model's performance degrades due to these cumulative limitations in the context management, task understanding, and memory retention. An agentic LLM workflow can address these challenges by actively managing task context, incorporating feedback loops for continuous improvement, and integrating external memory systems to handle complex, sequential dependencies effectively.

Additionally, the case study demonstrates that the agentic workflow effectively incorporates expert knowledge in a straightforward manner. This approach remains efficient even when the authors possess limited knowledge of EnergyPlus. It is anticipated that integrating additional domain expertise from experienced modelers could enhance this procedure further, leading to a more refined structure and improved processes.

4.2 Challenges for Agentic Workflow

The primary goal of utilizing LLMs is to minimize the engineering effort and expertise required, ideally allowing fewer experts to accomplish the same volume of tasks. However, the current capabilities of LLMs are limited; they do not perform optimally without supplementary agentic design, as demonstrated in this paper. This necessity for additional design and adaptation to make LLMs functional contradicts their original purpose—to reduce the need for extra effort. Therefore, while LLMs promise efficiency and resource reduction, realizing these benefits fully depends on overcoming their present limitations through enhanced design strategies, requiring huge amount of domain expertise from experienced modelers in the LLM agent development stage.

The costs associated with using LLMs are significant, stemming from both API calls and the need for powerful local computing resources such as expensive GPUs. These expenses are compounded by the energy-intensive nature of operations and the processing of tokens. For example, processing a large-size reference office building's data, which involves 50k words or 67k tokens, incurs costs of \$0.67 for input and \$2.01 for output at the utilized models' rates of \$10.00 per 1M input tokens and \$30.00 per 1M output tokens. Moreover, these costs can escalate further due to the necessity of multiple iterations to correct errors.

Besides, the self-consistency issue within LLMs, particularly in maintaining parameter consistency like "temperature" settings across different parts of the model, remains a fundamental challenge. Adjusting the "temperature" parameter to zero or near zero is one approach to mitigate this issue. However, it points to an inherent need to fundamentally address how LLMs manage internal consistency to enhance their reliability and applicability in complex simulations like BEM. This requires not only adjustments within the models themselves but also a thoughtful design of workflows, deep analysis of edge cases and exceptions, and a fundamental improvement in LLM that can mitigate these inconsistencies effectively.

LLM is a rapidly evolving field, and breakthroughs in LLM technology can significantly alter downstream applications, including BEM automation. For instance, recent advancements introduced at the conclusion of this paper, such as a new attention mechanism mentioned in the latest work [17], have resolved the token limit issue, fundamentally changing the design process to accommodate these limits. Moreover, as LLMs improve in reasoning, coding, and understanding human nuances, the future workflow of agents may further be simplified. Consequently, in addition to focusing on robust design to enhance agent performance, a more long-term effort should be directed towards improving LLMs themselves, which leads to the discussions in Section 4.3.

4.3 Need for Small-Size, BEM-Specialized LLMs

There are two main directions for improving LLMs. Firstly, from a capability perspective, LLMs should better understand domain-specific tasks to minimize manual engineering involvement. Secondly, they should be more cost-effective. In this study, we utilized one of the most advanced LLMs, the GPT-4 model. However, the cost of API calls is high, and we observed that the GPT-4 model sometimes provides more capability than needed and at other times fails to adequately comprehend the prompts. The primary issue is that GPT-4 is not ideally suited for this task; it is overly sophisticated for general applications yet insufficiently specialized for basic building energy simulation and building science knowledge: it is a Swiss army knife, not a scalpel for BEM.

These considerations suggest a future development pathway for LLMs: domain-specific models with fine-tuning. Fine-tuned LLMs can be significantly smaller in size for inference (e.g., 7 billion parameters compared to GPT-4's 1.76 trillion), which would accelerate inference times and reduce costs. Additionally, these LLMs can be specifically trained with domain-relevant data to enhance performance, thereby understanding the problems more effectively and reducing the need for complex, self-defined engineering processes in their application. Jiang et al. [5] fine-tuned a small LLM with only a 1 billion parameter model to successfully generate a simplified case-specific EnergyPlus IDF input file, indicating the feasibility of this future technical route.

5 Conclusions

This paper has explored the innovative application of Large Language Models (LLMs) within the realm of Building Energy Modeling (BEM), presenting a specialized agentic workflow designed to automate the generation and debugging of EnergyPlus input files. The development and testing of this workflow underscore the potential of LLM agentic workflow to enhance the efficiency and accessibility of complex engineering tasks, such as BEM, which traditionally require substantial expert knowledge and manual input. The case study demonstrates successful translation of a building description into an error-free EnergyPlus model for the iUnit modular building at the National Renewable Energy Laboratory. The effectiveness of our workflow surpasses: 1) naive prompt engineering, 2) other LLM-based workflows, and 3) manual modeling, in terms of accuracy, reliability, and time efficiency. The paper concludes with a discussion on the interplay between foundational models and LLM agent planning design, advocating for the use of fine-tuned, specialized models to advance this field.

The limitations outlined in this research paper primarily concern the operational constraints and potential areas for improvement within the deployment of LLM for BEM. High costs from LLM API usage and local computing needs, alongside challenges in maintaining internal consistency like temperature settings in models, necessitate robust design strategies and fundamental improvements in fine-tuned LLM models to realize their potential in applications like BEM automation. Looking forward, the continued evolution of LLM capabilities promises to further simplify and streamline workflows in complex system modeling. As these models become more adept at handling specialized tasks with greater precision and less human oversight, the potential for widespread adoption and application across various sectors looks increasingly feasible.

Reference

1. Citaristi, I., *International energy agency—iea*, in *The Europa directory of international organizations 2022*. 2022, Routledge. p. 701-702.
2. Kamel, E. and A.M. Memari, *Automated building energy modeling and assessment tool (ABEMAT).* Energy, 2018. **147**: p. 15-24.
3. Nguyen, N. and S. Nadi. *An empirical evaluation of GitHub copilot's code suggestions*. in *Proceedings of the 19th International Conference on Mining Software Repositories*. 2022.
4. Crawley, D.B., L.K. Lawrie, F.C. Winkelmann, W.F. Buhl, Y.J. Huang, C.O. Pedersen, R.K. Strand, R.J. Liesen, D.E. Fisher, and M.J. Witte, *EnergyPlus: creating a new-generation building energy simulation program.* Energy and buildings, 2001. **33**(4): p. 319-331.

5. Jiang, G., Z. Ma, L. Zhang, and J. Chen, *EPlus-LLM: A large language model-based computing platform for automated building energy modeling.* Applied Energy, 2024. **367**: p. 123431.
6. Saparov, A., R.Y. Pang, V. Padmakumar, N. Joshi, M. Kazemi, N. Kim, and H. He, *Testing the general deductive reasoning capacity of large language models using ood examples.* Advances in Neural Information Processing Systems, 2024. **36**.
7. Yang, C., X. Wang, Y. Lu, H. Liu, Q.V. Le, D. Zhou, and X. Chen, *Large language models as optimizers.* arXiv preprint arXiv:2309.03409, 2023.
8. Madaan, A., N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, and Y. Yang, *Self-refine: Iterative refinement with self-feedback.* Advances in Neural Information Processing Systems, 2024. **36**.
9. Shinn, N., F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, *Reflexion: Language agents with verbal reinforcement learning.* Advances in Neural Information Processing Systems, 2024. **36**.
10. Patil, S.G., T. Zhang, X. Wang, and J.E. Gonzalez, *Gorilla: Large language model connected with massive apis.* arXiv preprint arXiv:2305.15334, 2023.
11. Yang, Z., L. Li, J. Wang, K. Lin, E. Azarnasab, F. Ahmed, Z. Liu, C. Liu, M. Zeng, and L. Wang, *Mm-react: Prompting chatgpt for multimodal reasoning and action.* arXiv preprint arXiv:2303.11381, 2023.
12. Wei, J., X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q.V. Le, and D. Zhou, *Chain-of-thought prompting elicits reasoning in large language models.* Advances in neural information processing systems, 2022. **35**: p. 24824-24837.
13. Shen, Y., K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, *Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face.* Advances in Neural Information Processing Systems, 2024. **36**.
14. Qian, C., X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun, *Communicative agents for software development.* arXiv preprint arXiv:2307.07924, 2023.
15. Wu, Q., G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, *Autogen: Enabling next-gen llm applications via multi-agent conversation framework.* arXiv preprint arXiv:2308.08155, 2023.
16. Zhang, L., Z. Chen, and V. Ford, *Advancing Building Energy Modeling with Large Language Models: Exploration and Case Studies.* arXiv preprint arXiv:2402.09579, 2024.
17. Munkhdalai, T., M. Faruqui, and S. Gopal, *Leave no context behind: Efficient infinite context transformers with infini-attention.* arXiv preprint arXiv:2404.07143, 2024.