Towards Proactive Text-to-CAD Generation

Anonymous ACL submission

Abstract

Computer-Aided Design (CAD) systems are indispensable in mechanical engineering and product development processes. Nowadays, text-to-CAD methods can significantly reduce the learning cost of complex CAD systems and has attracted increasing attention. However, such methods fail to achieve alignment among user expectations, textual descriptions and CAD models. To address this limitation, we propose a new paradigm, "Proactive Text-to-CAD Generation", which first employs large language models to proactively elicit and formulate text enriched with comprehensive CAD design details, then generates CAD models from these refined descriptions. To support this paradigm, we construct the first actively interactive text-to-CAD dataset, `Proactive-Text2CAD`, which contains 4,590 high-quality dialogues. Moreover, building upon this dataset, we propose a novel agentic framework for this task, named "Proactive Agent", which is driven by a hierarchical finite state machine accompanying with three carefully designed modules. Extensive evaluation and comprehensive analysis on the `Proactive-Text2CAD` dataset demonstrate the effectiveness of both our proposed paradigm and agentic framework, with our method achieving significant improvements in both textual detail refinement and final CAD model generation quality.

1 Introduction

Computer-Aided Design (CAD) systems serve as fundamental tools in mechanical engineering and product development, revolutionizing prototyping methodologies (Robertson and Allen, 1993). In traditional CAD software (e.g., Autodesk, FreeCAD, SolidWorks and onshape), users create and modify geometric entities and constraints through graphical user interfaces (GUIs). However, this requires considerable expertise and proficiency, which can be challenging for non-specialists to master (Deng et al., 2024; Zhou and Camba, 2025).

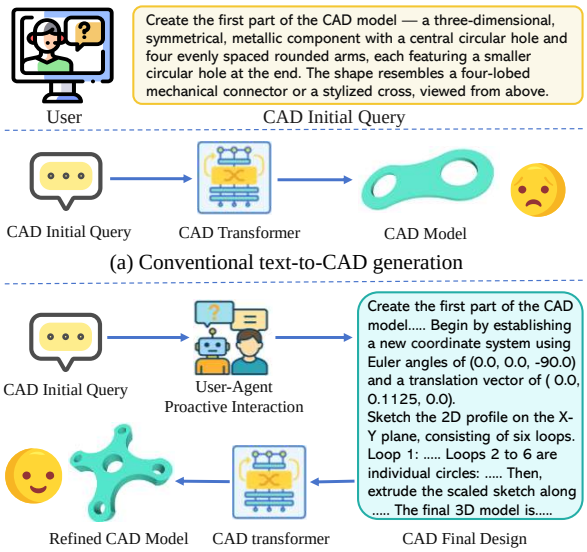

Figure 1: The conventional Text-to-CAD generation way versus our proposed proactive text-to-CAD generation way.

To address this challenge, integrating natural language input with CAD systems through highly capable large language models (LLMs), which are excel at interpreting and synthesizing structured data such as command sequences, streamlines parametric CAD generation. Recently, numerous studies have been devoted to this topic, such as Nelson et al. (2023); Khan et al. (2024b); Li et al. (2024); Kapsalis (2024). Such a text-to-CAD generation paradigm can definitely minimize the need for users to directly interact with complex GUIs, significantly lowering the barrier to CAD modeling (Zhou and Camba, 2025).

However, such a paradigm is not without its flaws, particularly in achieving alignment among user expectations, textual descriptions and CAD models. In detail, current text-to-CAD methods (Nelson et al., 2023; Khan et al., 2024b; Li

et al., 2024; Kapsalis, 2024; Badagabettu et al., 2024) can effectively handle the alignment between textual descriptions and CAD models, but they struggle to maintain satisfactory alignment when dealing with vague or abstract textual inputs. Some compelling experimental results (Khan et al., 2024b) also reveal that as textual descriptions become more abstract, the accuracy and precision of generated CAD models deteriorate significantly. This challenge is particularly pronounced because novices and non-specialists tend to provide high-level descriptions of model appearance rather than detailed parametric specifications. Moreover, even experienced CAD engineers find it difficult to produce text descriptions that are both sufficiently detailed and compliant with CAD generation principles without the aid of a visual interface. In short, merely relying on user-input text descriptions cannot bridge the gap between user intent and the final CAD model.

To fill this gap, we propose proactive text-to-CAD generation (shown in Figure 1). Such a way actively engages users in iterative questioning to seek missing CAD design details in the initial user provided textual description. By refining incomplete parametric specifications and enhancing the text's descriptive quality through this dialogue-driven process, the framework ensures higher-quality input before final CAD generation. To achieve proactive text-to-CAD generation, we make the following efforts in this paper:

First, we build the first actively interactive text-to-CAD dataset, `Proactive-Text2CAD`. Our goal is to integrate the incomplete user-provided CAD text descriptions with a proactive information-seeking dialogue, enabling the agent to ask targeted questions actively when encountering missing or unclear CAD parameters. To achieve that, based on Text2CAD's expert-level text annotation dataset and CAD dataset (Khan et al., 2024b), we further involve two key phases: (1) User initial query generation and (2) Proactive dialogues generation. Through such a dataset construction process, we obtain 4,590 high-quality dialogues, each containing an incomplete metadata entry and an active dialogue, totaling around 28,110 question-answer pairs. Building upon our dataset, we further establish a family of strong and representative baselines, which can be roughly categorized into three type (Deng et al., 2025): (1) Standard Prompting, like Proactive Prompting (Deng et al., 2023); (2) CoT Prompting, like ProCoT (Deng et al., 2023) and PS+ Prompting(Wang et al., 2023), and (3) Multi-agent Prompting, like MACRS (Fang et al., 2024).

Second, we propose a novel agentic framework for this task, named "Proactive Agent". In detail, through experimens, we find that current proactive methods, like ProCoT, PS+ Prompting and MACRS, underperform in refining textual descriptions of CAD models, especially exhibiting significant limitations in comprehensively identifying missing operational and parametric details in the CAD model, due to the complex CAD topology. For example, when the subtypes face and loop are missing in the sketch type of the CAD model, since loop is a subtype of face and a CAD model can contain multiple faces, current baselines fail to proactively ask the user for specific information about the face first. To address this issue, we involve three modules in the proposed agentic framework: strategic, tactical, and operational modules. The strategic module analyzes the current CAD design text at a macro perspective, captures missing details, and generates an agent workflow. The tactical and operational modules each contain independent modules to address specific functions, decoupling the complex workflow analysis and execution process. To interconnect these three modules and enable autonomous operation, we design a hierarchical finite state machine (HFSM) that drives the reasoning logic from workflow generation to execution through state transitions.

The primary contributions of this work can be summarized as follows:

- We propose "Proactive Text-to-CAD Generation" paradigm that engages users in iterative questioning to seek missing CAD design details in the initial user provided textual description, then generates CAD models from the refined textual descriptions.

- We construct `Proactive-Text2CAD`, the first dataset for studying actively interactive text-to-CAD generation, which contains around 4,500 high-quality dialogues.

- We present "Proactive Agent", an innovative agentic framework driven by a hierarchical finite state machine accompanying with workflow generation, task dispatching, questioning, and summarization functional modules.

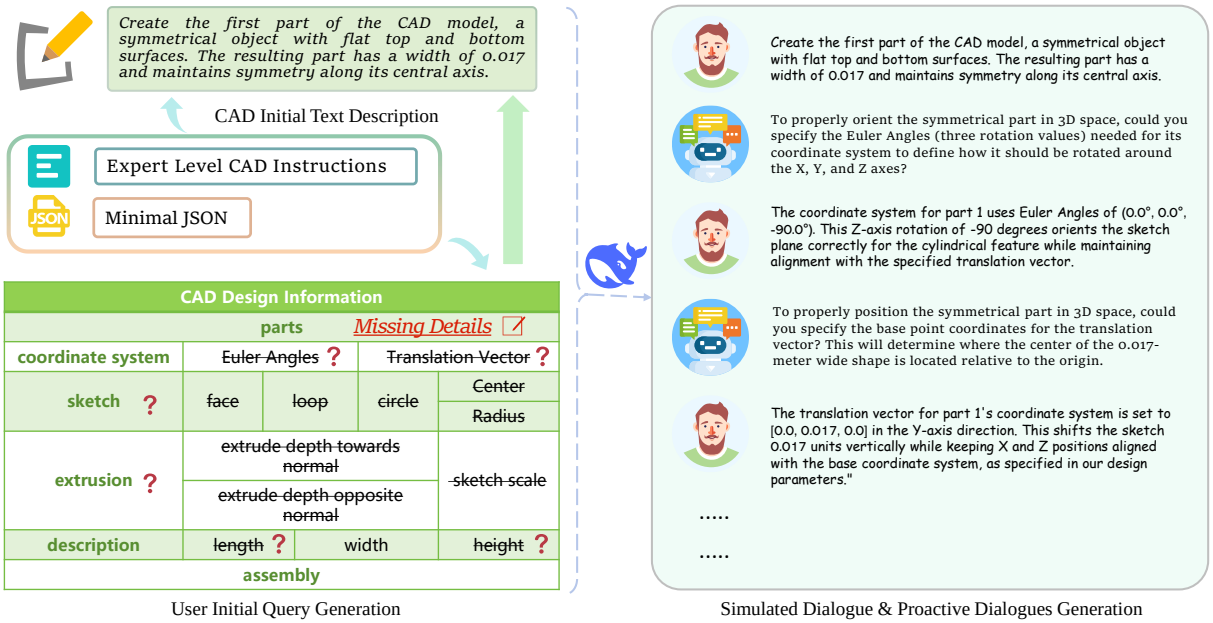

Figure 2: The whole process of `Proactive-Text2CAD` construction.

2 `Proactive-Text2CAD` Construction

2.1 Dataset Construction

Our objective is to combine incomplete user-provided initial CAD text descriptions with proactive information-seeking dialogue, enabling the agent to actively inquire when facing CAD parameter details that are missing or incomplete. As illustrated in Figure 2, the dataset construction process consists of two key phases: (1) User Initial Query Generation; (2) Proactive Dialogues Generation.

2.1.1 User Initial Query Generation

At this stage, our task is to generate user initial query with uncertain detail missing, which are divided into four specific steps:

First, through collecting the multi-level textual description dataset of CAD constructed by Text2CAD and the minimal metadata dataset (minimal json) (Wu et al., 2021a; Khan et al., 2024b), we generate an initial dataset containing precise geometric descriptions, text descriptions with relative measurements, and concise representations of shape attributes and their relational properties in CAD designs.

Next, based on the minimal metadata, we categorize the details of CAD textual descriptions into six main types: component quantity, sketch, extrusion, coordinate system, appearance description, and assembly method. Among these, the sketch, extrusion, coordinate system, and appearance description types further contain multiple subtypes (e.g., subtypes of sketch: face, loop, line, etc.) [1].

Then, we randomly delete information of main types and subtypes to create missing types. We subsequently use Deepseek-R1 (DeepSeek-AI et al., 2025) to rewrite the text descriptions based on the missing types, generating user initial query that lack the missing category information, i.e., incomplete texts[2].

Finally, since there exists sequential relationships among CAD operation types and dependency relationships among parameter types, we employ topological sorting to arrange the questioning order of missing types according to the dependency graph[3].

2.1.2 Proactive Dialogues Generation

In this stage, our task is to generate the whole dialogue, which are divided into four specific steps:

First, we utilize DeepSeek-R1 (DeepSeek-AI et al., 2025) to generate one question for each missing type to inquire about the details of the missing information. These questions serve as the questions posed by the system to the user.

Second, for each generated question, we retrieve specific information about the missing types from the initial dataset, then use DeepSeek-R1 to generate natural user responses based on this specific

[1] Detailed type information can be found in Appendix A.1.1.

[2] The rewriting prompts are provided in Appendix A.2.1

[3] Specific information can be found in Appendix A.1.1

information[4].

Third, based on the generated initial user query, we sequentially arrange the generated question-answer pairs according to the order of missing types, thereby creating proactive dialogues between the system and users.

Finally, three human reviewers are involved to verify whether: (1) the initial user queries contain all type information except for the missing types, (2) the questions cover all missing category details, and (3) the responses provide specific detailed information. This dataset construction method achieved a 77.64% pass rate in human review.

2.2 Dataset Statistics

After data processing and filtering, we obtain 4,590 samples, each containing an incomplete metadata entry and an active dialogue, totaling around 28,110 question-answer pairs. Uncertain details are randomly selected from primary and subcategories. Due to the variable number of subtypes across different CAD models and the inherent instability of category definitions, the dataset exhibits a naturally right-skewed asymmetric distribution. Detailed dataset specifications are provided in Appendix A.1.3.

2.3 Evaluation Protocols

We design several evaluation metrics at three different levels: turn-level, dialogue-level and CAD-level.

2.3.1 Turn-Level Evaluation

Following Zhang et al. (2024), we use **Clarification Accuracy (Clari. Acc.)** to measures the system's ability to actively query incomplete CAD design texts, where a score of 1 is assigned if the response is a valid question, otherwise 0. Besides, we also involve **Rule-based Score** to assesses whether responses target the correct missing attribute category, where a binary score (1/0) is determined by the presence of keywords relevant to the target attribute. Moreover, **Rough-L** (Lin, 2004) and **BERTScore** (Zhang et al., 2020) is used to quantify semantic similarity between responses and reference questions. Note that, we evaluate all turn-level metrics through both Micro and Macro averaging, where Micro-averaging computes the mean score per turn, while Macro-averaging calculates the mean score per dialogue.

[4]The prompts for simulating system questions and user responses can be found in Appendix A.2.1

2.3.2 Dialogue-Level Evaluation

Completion Rate. This metric measures the degree to which missing information in the CAD text description is completed during the dialogue, calculated as:

$$\text{Completion Rate} = \frac{N_c}{N_t} \tag{1}$$

where N_t refers to the count of initially absent CAD operations or parameter types in the user's input, and N_c denotes the count of missing categories addressed (queried) during the dialogue.

Effectiveness Rate. This metric evaluates the proportion of effective questions in the dialogue, defined as:

$$\text{Effectiveness Rate} = \frac{N_e}{N_i} \tag{2}$$

where N_e denotes the number of effective questions that satisfy all of the following criteria: (a) targeting a missing category, (b) having not been previously asked, and (c) receiving a non-"unknown" response, and N_i refers to the total number of interaction turns.

2.3.3 CAD-Level Evaluation

To evaluate the quality of the final generated CAD, we leverage **Chamfer Distance (CD)** (Fan et al., 2016; Wu et al., 2021b), which can measure geometric similarity between generated 3D CAD models and ground truth, where lower values indicate higher geometric fidelity. In addition to automatic evaluation, we also involve manual evaluation to comprehensively assess the CAD model quality. **Average Rank** metric is used in manual evaluation, which can be denoted as

$$\text{Average Rank} = \frac{1}{N}\sum_{i=1}^{N} R_i \tag{3}$$

where N is the total number of CAD models generated by different methods and R_i is the rank of the i-th model (lower values denote better alignment)

3 Methodology

3.1 Overall Pipeline

To achieve proactive text-to-CAD generation, we involve a pipeline with two parts, where Part 1 is designed to proactively query users for missing CAD design information and generate the final CAD design text, and while Part 2 is configured to employ a CAD Transformer for converting final CAD design text into CAD models.

Note that, in this paper, we only focus on Part 1 and build baselines and our proposed method. As for Part 2, we leverage the state-of-the-art method, Text2CAD(Khan et al., 2024b), to convert the final CAD design text into a CAD model.

3.2 Baseline

Following Deng et al. (2025), we involve three type of baselines: (1) Standard Prompting, (2) CoT Prompting and (3) Multi-agent Prompting.

In detail, in standard prompting, we involve **Proactive Prompting** (Deng et al., 2023) (short for "Proactive" in the experiments), which can provide the LLM with alternative options for determining appropriate actions to take in responses. In CoT prompting, besides the standard CoT (Wei et al., 2022), we further build **ProCoT** (Deng et al., 2023) and **PS+ Prompting** (Wang et al., 2023) (short for "PS+"), where ProCoT involves dynamic reasoning and planning to analyze subsequent actions for achieving dialogue objectives, and PS+ is a two-stage method that first generates both the reasoning process and potential answers through logical inference, then employs answer extraction prompts to derive the final solution. Finally, in the multi-agent prompting, we leverage **MACRS** (Fang et al., 2024) as the representative, which is a collaborative multi-agent framework that integrates four LLM-based agents to plan diverse dialogue behaviors, employing a feedback-aware reflection mechanism for agent adaptation. Note that, more details about baselines can be find in the Appendix A.4.

3.3 Proposed Method: Proactive Agent

Through experiments, we find that current baseline methods perform poorly in refining CAD detail texts and struggle to comprehensively and holistically identify the missing operational and parametric details in the current CAD model. To address the aforementioned issue, we propose a novel multi-agent framework named "Proactive Agent" (shown in Figure 3), which consists of strategic, tactical, and operational modules. To interconnect these three modules and enable autonomous operation, we design a hierarchical finite state machine (HFSM) that drives the reasoning logic from workflow generation to execution through state transitions.

3.3.1 Strategic Module

The strategic module (S_0) analyzes the current CAD design text at a macro level, captures missing details, and generates an agent workflow. There are two key phase in this module.

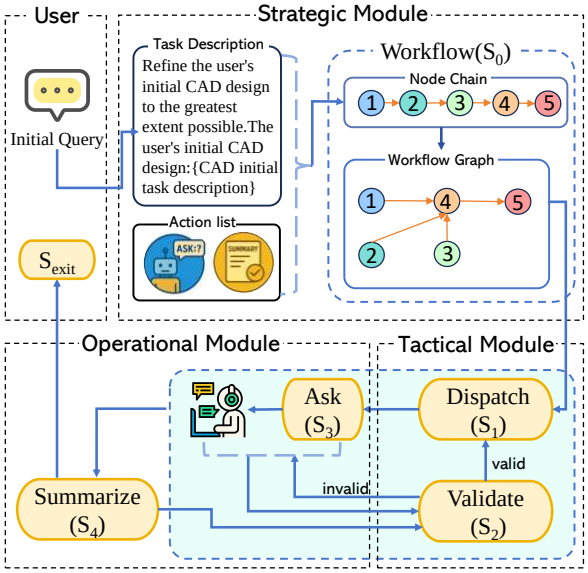

Figure 3: Our proposed agentic framework: "Proactive Agent".

Specifically, first, the strategic module generate a topological sequence $\mathcal{C}$ of the workflow graph via a give LLM_θ[5] , which can be denoted as

$$\mathcal{C}(V) \leftarrow \text{LLM}_\theta(d, t, \mathcal{A}). \tag{4}$$

In the above equation, d represents the task description of workflow generation, t represents the initial textual description of the user's CAD modeling task, $\mathcal{A}$ denotes the action set { "ask", "summarize"}, where "ask" indicates proactively querying the user for CAD information, and "summarize" indicates extracting the CAD design scheme from the dialogue history, and the nodes $V = \{v_1, v_2, \ldots, v_n\}$ represent tasks to be finished, such as "ask for the coordinate system details" and "ask for the details of the sketch to be created".

Then, based on the dependencies among the details of each CAD operation and parameter, the topological sequence $\mathcal{C}$ then forms a directed acyclic graph $\mathcal{G}$ according to topological sorting (Qiao et al., 2024), which can be denoted as:

$$\mathcal{G}(V, E) \leftarrow \mathcal{C}(V) \tag{5}$$

where $E = \{(v_i, v_j)\}$, where $1 \leq i \neq j \leq n$, represent the execution relationships between nodes (v_j must be executed after v_i).

[5]The prompt is shown in the Appendix A.2.2.

Method	Clarif.Acc. ↑		ROUGH-L ↑		Rule-based Score ↑		BertScore ↑	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
Standard	0.10	0.09	0.05	0.04	0.07	0.06	0.05	0.04
CoT	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Proactive	4.12	3.72	2.33	2.02	7.53	6.84	2.74	2.38
ProCoT	67.85	71.22	19.55	18.75	35.76	31.10	20.50	19.74
PS+	44.67	37.27	14.75	13.84	12.27	9.93	14.90	13.50
MACRS	99.54	99.74	17.24	15.72	34.68	28.90	21.33	19.93
Ours	**99.89**	**99.93**	**21.04**	**19.11**	**45.53**	**38.14**	**24.62**	**22.21**

Table 1: Experimental results at the turn-level.

Method	Completion Rate ↑	Effectiveness Rate ↑
Standard	0.04	0.07
CoT	0.00	0.00
Proactive	3.22	3.17
ProCoT	12.45	5.95
PS+	6.78	8.56
MACRS	8.27	2.68
Ours	**23.89**	**23.34**

Table 2: Experimental results at the dialogue-level.

3.3.2 Tactical Module

To achieve efficient workflow execution, the tactical module (S_{tact}) consists of two sub-modules: dispatch (S_1) and validation (S_2).

The dispatch sub-module is responsible for receiving and parsing the workflow graph $\mathcal{G}$. Via Kahn's Algorithm (kah, 1962), the parsing function τ can dispatch nodes V to to either the asking or summary sub-modules (introduced in §3.3.3, and denoted as S_3 and S_4, respectively), which can be represented as:

$$\tau(V; \mathcal{G}(V, E)) \rightarrow \{S_3, S_4\}, \tag{6}$$

The validation sub-module evaluates the question q (produced by the asking sub-module) and summary p (generated by the summary sub-module) against the task instruction v_i via the give LLM_θ[6]. Based on verification results, it triggers regeneration of q or p through corresponding sub-modules.

3.3.3 Operational Module

To complete the underlying task details at the micro-level and improve the execution efficiency of workflow G, the operational module (S_{oper}) consists of two independent sub-modules: asking (S_3) and summary (S_4).

The asking sub-module is responsible for proactively asking questions to the user. After receiving a task instruction v_i from the dispatch sub-module, the asking sub-module can leverage the give LLM_θ to generate question q[7], which can be represented as

$$q \leftarrow \text{LLM}_\theta(v_i) \tag{7}$$

The summary sub-module is responsible for generating the final CAD design solution based on the dialogue history $\mathcal{H}$. After the summary module receives the task instruction, it generates a CAD design solution in combination with the task instruction v_i, which can be represented as:

$$p \leftarrow \text{LLM}_\theta(v_i, \mathcal{H}) \tag{8}$$

where the dialogue history $\mathcal{H}$ includes the user's initial CAD text description, the questions posed by the asking sub-module, and the user's responses.

3.3.4 Hierarchical Finite State Machine

To coordinate the integration and autonomous operation of the aforementioned modules, we leverage the Hierarchical Finite State Machine (HFSM) (Alur and Yannakakis, 2001). Such a machine can employ state-based reasoning to systematically govern the workflow from generation to execution. The HFSM is formally modeled as a quintuple $\{S, O, \mu, S_0, \{S_{\text{exit}}\}\}$:

- $S = S_0 \cup S_{\text{tact}} \cup S_{\text{oper}} \cup S_{\text{exit}}$ is a hierarchical state set, where S_0 denotes the strategic module and is the initial state in the machine, $S_{\text{tact}} = \{S_1, S_2\}$ is the tractical module, $S_{\text{oper}} = \{S_3, S_4\}$ represents the operational module, and while S_{exit} denotes the terminal state.
- O represents the set of all possible outputs from the aforementioned modules S.
- $\mu : S \times O \rightarrow S$ is a transition function that determines the next state in the reasoning process based on the current state and the execution result of the corresponding module[8].

Figure 3 illustrates the whole working mechanism of the HFSM for the proposed proactive agent.

[6]The prompt is shown in the Appendix A.2.2.

[7]The prompt is shown in the Appendix A.2.2

[8]We put the detail of the transition function in the Appendix A.3 due to the page limit.

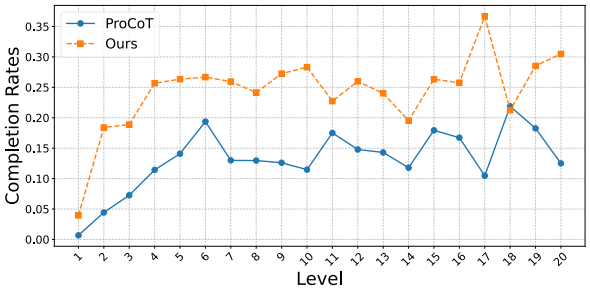

Figure 4: Completion rates for dialogues with varying numbers of missing details.

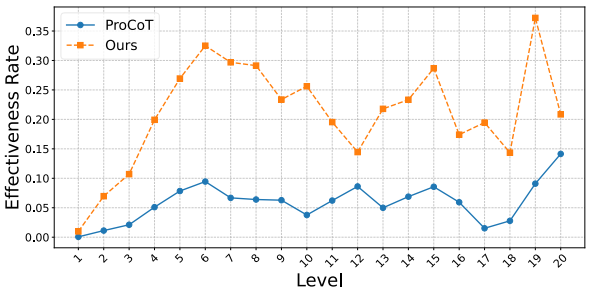

Figure 5: Effectiveness rates for dialogues with varying numbers of missing details.

Method	Median CD ↓	Mean CD ↓	Avg. Rank ↓
Initial Query	45.52	142.61	\
Standard	47.66	141.56	3.56
CoT	48.48	141.50	3.82
Proactive	36.42	129.54	2.78
ProCoT	28.00	116.13	2.20
PS+	34.73	131.80	2.61
MACRS	27.03	119.69	2.64
Ours	**20.60**	**99.70**	**1.60**

Table 3: Experimental results at the final CAD-level.

4 Experiment

4.1 Performance Comparisons

In this paper, we employ GPT-4o Mini as the primary model for our main experiments.

4.1.1 Results at the Turn-Level

The experimental results are shown in Table 1, and demonstrate our proposed proactive agent's superior performance across all metrics compared to all baseline approaches, confirming its effectiveness in identifying missing CAD design details and generating appropriate proactive queries. Notably, we find that CoT performed poorly, failing to analyze omissions or initiate proactive queries, indicating its inadequacy for this task. In contrast, MACRS and ProCoT maintain competent performance in both detail analysis and query generation, exhibiting particularly strong questioning capabilities in this specific assessment dimension.

4.1.2 Results at the Dialogue-Level

The experimental results are shown in Table 2. From the table, we find that: our proposed proactive agent outperforms all baselines in both Completion Rate and Effective Rate, demonstrating its superiority in comprehensively and accurately identifying missing information in CAD text descriptions. Besides, CoT's poor performance at dialogue-level evaluation aligns with its turn-level shortcomings, consistently failing to detect missing details or initiate meaningful inquiries. Meanwhile, we also find that MACRS is no longer performing well, particularly showing poor performance in terms of the effectiveness rate.

Furthermore, we categorize these dialogues into 20 distinct levels based on the type and quantity of missing information or required operations, conducting granular dialogue-level evaluations at each individual level. As illustrated in Figure 4 and Figure 5, our method consistently outperforms the best baseline approach, ProCoT, across all classification levels, demonstrating its effectiveness.

4.1.3 Results at the CAD-Level

Table 3 presents the experimental results at the CAD-level. From the results, we find that our method and most baseline approaches achieve superior CD values compared to the initial user queries, demonstrating that the proactive interaction paradigm can effectively improve alignment between CAD models and user expectations. Moreover, our proposed proactive agent can outperform all baselines in CD metrics, confirming its effectiveness for this task. The CD results generally correlate positively with completion rates from dialogue-level evaluation, indicating that better textual refinement leads to improved geometric alignment. Interestingly, MACRS exhibits slightly better median CD than ProCoT, likely due to its polarized performance (either excellent or poor refinement). However, ProCoT maintains a better average CD than MACRS, suggesting more consistent overall quality in CAD model refinement.

4.2 Discussion

We investigate the impact of different base LLMs on experimental outcomes by employing GPT-4o mini, Gemini-2.0-flash-lite, and Deepseek-R1.

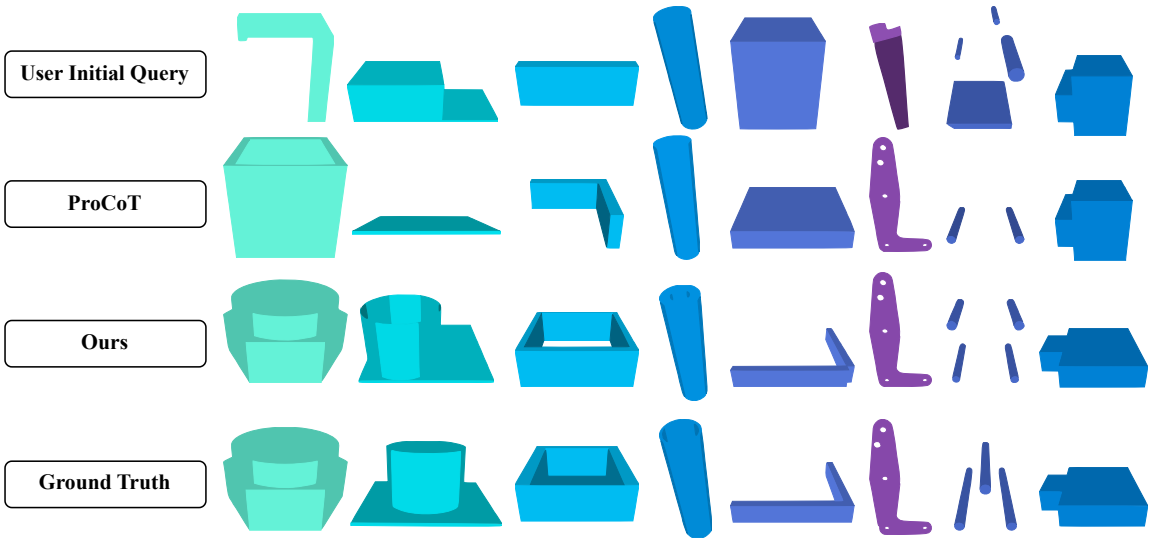

Figure 6: Case studies to illustrate the effectiveness of our method.

LLM	Method	Dialogue-level Evaluation		CAD Model Evaluation	
		Completion Rate (%)	Effectiveness Rate (%)	Median CD	Mean CD
GPT-4o mini	ProCoT	12.45	5.95	28.00	116.13
	Ours	23.89	23.34	20.60	**99.70**
Gemini-2.0-flash-lite	ProCoT	14.58	8.64	25.11	114.54
	Ours	24.76	9.54	20.91	106.24
DeepSeek-R1	ProCoT	14.31	10.22	30.55	115.44
	Ours	**48.75**	**29.71**	**17.60**	105.14

Table 4: The experimental result by changing base LLMs.

Among the baseline methods, ProCoT performs the best in the previous experiments. Therefore, we further test ProCoT and our proposed method in terms of dialogue and CAD model generation. The experimental results are shown in Table 4. From the results, we find that the base LLMs have a big impact on the experimental results. Among these LLMs, DeepSeek-R1 perform notably well. Besides, our method outperforms ProCoT in all metrics when the base LLMs are changed.

4.3 Case Study

In Figure 6, we compare the geometric appearance of the user's initial CAD text, the strongest baseline method ProCoT, our method, and the ground truth CAD model. We find that the CAD models generated by our method have more details than those generated by the user's initial CAD text and ProCoT, and are the most similar to the ground truth. More specifically, in the first case, the ground truth appears similar to a nut. The CAD models generated by the user's initial text and ProCoT deviate significantly from the ground truth, while our method generates a shape that is most similar to the ground truth. In the third case, the ground truth resembles a simple rectangular container. Compared to the user's initial text, ProCoT only generates one additional side panel, whereas our method can generate all the side panels. The only difference from the ground truth is the absence of a base plate.

5 Conclusion

In this paper, we extend the static text-to-CAD paradigm to the proactive text-to-CAD paradigm, which can engage users in iterative questioning to find missing CAD design details. Second, to achieve this paradigm, we construct the first dataset for studying actively interactive text-to-CAD generation, `Proactive-Text2CAD`. Third, we present "Proactive Agent", an innovative agentic framework for this paradigm. Through extensive experiments, we demonstrate the effectiveness of both our proposed paradigm and agentic framework.

Limitation

Our method is currently a pipeline way with two parts, where Part 1 is designed to proactively query users for missing CAD design information and generate the final CAD design text, and while Part 2 is configured to employ a CAD Transformer for converting final CAD design text into CAD models. In future work, we may explore an end-to-end way from the user's initial text to the CAD model through active interaction and implement joint optimization of the user's initial text and the CAD model.

Besides, through experiments, we find that the validation sub-module in our proposed agentic framework can increase the proactivity of inquiry, but also introduce the issue of generating invalid questions, which can slightly reduce the efficiency of our agentic framework. In future work, we will explore new method to improve the validation sub-module.

References

1962. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

Rajeev Alur and Mihalis Yannakakis. 2001. Model checking of hierarchical state machines. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 23(3):273–303.

Autodesk. https://www.autodesk.com/products/autocad/. Online: accessed 2025-04-20.

Akshay Badagabettu, Sai Sravan Yarlagadda, and Amir Barati Farimani. 2024. Query2cad: Generating cad models using natural language queries. *Preprint*, arXiv:2406.00144.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. 2023. Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration. *arXiv preprint arXiv:2305.13626*.

Yang Deng, Lizi Liao, Wenqiang Lei, Grace Hui Yang, Wai Lam, and Tat-Seng Chua. 2025. Proactive conversational ai: A comprehensive survey of advancements and opportunities. *ACM Transactions on Information Systems*, 43(3):1–45.

Yuanzhe Deng, James Chen, and Alison Olechowski. 2024. What sets proficient and expert users apart? results of a computer-aided design experiment. *Journal of Mechanical Design*, 146(1).

Haoqiang Fan, Hao Su, and Leonidas Guibas. 2016. A point set generation network for 3d object reconstruction from a single image. *Preprint*, arXiv:1612.00603.

Jiabao Fang, Shen Gao, Pengjie Ren, Xiuying Chen, Suzan Verberne, and Zhaochun Ren. 2024. A multi-agent conversational recommender system. *arXiv preprint arXiv:2402.01135*.

FreeCAD. https://www.freecadweb.org/. Online: accessed 2025-04-20.

Timo Kapsalis. 2024. Cadgpt: Harnessing natural language processing for 3d modelling to enhance computer-aided design workflows. *arXiv preprint arXiv:2401.05476*.

Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. 2024a. Cad-signet: Cad language inference from point clouds using layer-wise sketch instance guided attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4713–4722.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. 2024b. Text2cad: Generating sequential cad models from beginner-to-expert level text prompts. *arXiv preprint arXiv:2409.17106*.

Xueyang Li, Yu Song, Yunzhong Lou, and Xiangdong Zhou. 2024. Cad translator: An effective drive for text to 3d parametric computer-aided design generative modeling. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 8461–8470.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Matt D Nelson, Brady L Goenner, and Bruce K Gale. 2023. Utilizing chatgpt to assist cad design for microfluidic devices. *Lab on a Chip*, 23(17):3778–3784.

onshape. https://www.onshape.com/en/. Online: accessed 2025-04-20.

Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Benchmarking agentic workflow generation. *arXiv preprint arXiv:2410.07869*.

David Robertson and Thomas J Allen. 1993. Cad system use and engineering performance. *IEEE Transactions on Engineering Management*, 40(3):274–282.

SolidWorks. https://www.solidworks.com/. Online: accessed 2025-04-20.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Rundi Wu, Chang Xiao, and Changxi Zheng. 2021a. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6772–6782.

Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. 2021b. Density-aware chamfer distance as a comprehensive metric for point cloud completion. *Preprint*, arXiv:2111.12702.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. 2024. Ask-before-plan: Proactive language agents for real-world planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10836–10863, Miami, Florida, USA. Association for Computational Linguistics.

Jiwei Zhou and Jorge D Camba. 2025. The status, evolution, and future challenges of multimodal large language models (llms) in parametric cad. *Expert Systems with Applications*, page 127520.

A Appendix

A.1 Dataset Specifications

A.1.1 User Initial Query Generation

We first define the main types and subtypes of CAD operations and parameters, and randomly remove the type information included in the minimal metadata. Then, we perform a topological sort on the CAD types based on the dependencies between CAD type information.

Main Types and Subtypes of CAD Operations and Parameters. We adopt the same representation method proposed by Khan et al. (2024a), which uses a sketch-and-extrude format. Each 2D sketch consists of multiple faces, each face consists of multiple loops, and each loop either contains a line and an arc or a circle. Loops are always closed (i.e., the start and end coordinates are the same). The specific descriptions of the main types and subtypes in our dataset are presented in Table 5.

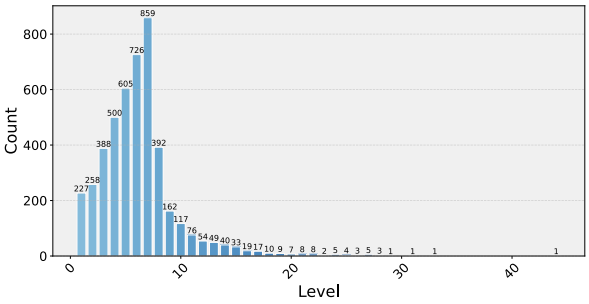

Figure 7: The statistical distribution of our dataset.

Dependencies Between CAD Types. The dependencies between CAD types are shown in Figure 8. After randomly deleting missing types (including both main types and subtypes), we use topological sorting to order the missing types, laying the groundwork for the sequence of proactive dialogue generation.

A.1.2 Proactive Dialogues Generation

To facilitate the reproducibility of our dataset, we employ DeepSeek-R1 to generate proactive dialogues with users. The system proactively asks users questions based on the missing type objects, and users retrieve corresponding information from the initial dataset to answer the system's questions. The prompts for the system's responses and user inquiries are shown in Table A.1.1.

The initial text of the user, namely the initial text description of the CAD design, is adapted from the text descriptions in the initial dataset. We also use DeepSeek-R1 for this adaptation. The prompts for the adaptation are shown in Table 7.

A.1.3 Dataset Statistics

We classify the samples according to the number of missing types in each sample. Since the subtypes within the main types vary across dataset samples (e.g., different sketches may contain different numbers of loop subtypes), which makes it inconvenient to count, we uniformly consider the absence of one main type and one subtype as a single type of missing information, and include it in the count of missing types.

Our dataset contains approximately 4,590 dialogues, with 28,110 question-answer pairs. The distribution of the number of missing types per

sample is naturally right-skewed and discrete, concentrated in the range of 4 to 9, with a long right tail, consistent with the characteristics of a skewed discrete count data distribution. The specific distribution of the dataset is shown in Figure 7.

A.2 Prompt List

A.2.1 Dataset Construction

Prompts for generating dialogues between the simulated system and user are presented in Table 6. Prompts for rewriting the text of the initial dataset to generate CAD initial queries are presented in Table 7.

A.2.2 Proposed Method: Proactive Agent

The prompts for generating workflows by the strategic module are presented in Table 10. The prompts for dispatching tasks by the dispatch sub-module, for validating task completion by the validation sub-module, for initiating questions by the asking sub-module, and for generating summaries by the summary sub-module are presented in Table 9.

A.2.3 Evaluation

The prompts for simulating user responses in the dialogue-level experiments are presented in Table 8.

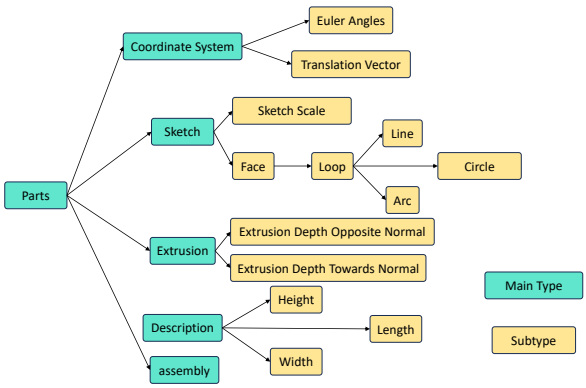

Figure 8: Dependencies between CAD types.

A.3 Details of Our Methods

The transition function μ of a Hierarchical Finite State Machine (HFSM) can be represented as:

$$\mu(S_i, y) = \begin{cases} S_0 & \text{if } S_i = S_{\text{exit}} \\ & \quad \text{and } y = t, \\ S_1 & \text{if } S_i = S_0 \\ & \quad \text{and } y = \mathcal{G}(V, E), \\ S_3 & \text{if } S = S_1 \\ & \quad \text{and action} = \text{ask}, \\ S_4 & \text{if } S = S_1 \\ & \quad \text{and action} = \text{summarize}, \\ S_2 & \text{if } S = S_3 \\ & \quad \text{and } e = q, \\ S_2 & \text{if } S = S_4 \\ & \quad \text{and } e = p, \\ S_1 & \text{if } S = S_2 \\ & \quad \text{and action} = \text{ask} \\ & \quad \text{and task completion is valid}, \\ s_{\text{exit}} & \text{if } S = S_2 \\ & \quad \text{and action} = \text{summarize} \\ & \quad \text{and task completion is valid}, \\ S_3 & \text{if } S = S_2 \\ & \quad \text{and action} = \text{ask} \\ & \quad \text{and task completion is invalid}, \\ S_4 & \text{if } S = S_2 \\ & \quad \text{and action} = \text{summarize} \\ & \quad \text{and task completion is invalid}, \\ S_{\text{exit}} & \text{otherwise.} \end{cases} \tag{9}$$

In this context, y represents the input to the current state, and the current state transitions to the next state through the transition function μ combined with the input y.

The overall execution logic of our hierarchical finite state machine (HFSM) begins with the user's initial CAD design text description t, which first enters the workflow sub-module of the strategic module. At this point, it is in the initial state (S_0), where it plans how to ask the user for missing information in order to ultimately generate a complete CAD design text, thereby generating the workflow of the operational sub-module. Then, the generated workflow is input into the tactical module and enters the dispatch sub-module for workflow parsing. The workflow parsing generates a series of subtasks, which the dispatch sub-module assigns to the operational sub-module. At this point, it transitions to state S_1. Next, the asking sub-module or summary sub-module of the operational module, which receives the sub-tasks, executes the tasks, transitioning to state S_3 or S_4. The completion of tasks by the asking and summary modules is verified by the validation sub-module of the tactical module, which means transitioning to state S_2 for

validation. If the validation sub-module determines that the task of the asking sub-module is complete, it re-enters the dispatch module, transitioning to the next task assignment state S_1. Otherwise, it returns to the asking sub-module to regenerate the question q. The validation of the summary module is similar to that of the asking module, except that when the validation sub-module determines that the task is complete, i.e., the final CAD design solution has been generated, it transitions to the termination state S_{exit}, completing the entire execution logic of the HFSM.

A.4 Baseline

Since no existing large language model (LLM) methods currently address proactive querying or interactive capabilities for CAD applications, we establish six baseline approaches based on our newly constructed Interactive-CAD dataset and novel CAD generation paradigm. These baselines incorporate both commonly-used and state-of-the-art (SOTA) LLM-based proactive dialogue methodologies.

Standard Prompting(Deng et al., 2023): Given a task description q, the user's initial CAD text description t, and dialogue history C, we instruct the LLM to perform CAD proactive querying and generate response r. The task descriptions and prompt templates are provided in Table 12 of Appendix A.4.1. This prompting scheme can be formally represented as:

$$p(r \mid q, t, C)$$

CoT(Wei et al., 2022): A chain-of-thought prompting approach that generates intermediate reasoning steps to derive the final response. In the task description, we require the system to simulate the next response based on the current dialogue history. The prompt template is provided in Table 13 of Appendix A.4.1.

Proactive Prompting(Deng et al., 2023): Proactive Prompting is designed to provide the LLM with alternative options for determining appropriate actions to take in responses, formally represented as:

$$p(a, r \mid q, t, C, A)$$

Given the task description q, the user's initial CAD text description t, dialogue history C, and a set of possible dialogue actions A, this approach instructs the LLM to: (1) select the most suitable dialogue action $a \in A$, and (2) generate the corresponding response r. To adapt this to CAD proactive querying tasks, we define the dialogue actions as either "querying the user about missing CAD design details" or "summarizing the CAD design solution". The prompt templates are provided in Table 14 of Appendix A.4.1.

ProCoT(Deng et al., 2023): The proactive chain-of-thought prompting scheme (ProCoT) involves dynamic reasoning and planning to analyze subsequent actions for achieving dialogue objectives, formally represented as:

$$p(c, a, r \mid q, t, C, A)$$

where c denotes the cognitive description of the decision-making process for subsequent actions, while q, t, C, A, and r maintain the same definitions as in **Proactive Prompting**. For CAD proactive querying tasks, we define c as the analysis of missing detail types in the current CAD design. The prompt templates are provided in Table 15 of Appendix A.4.1.

PS+ Prompting(Wang et al., 2023): Plan-and-Solve Prompting (PS prompting) is a two-stage methodology that first generates both the reasoning process and potential answers through logical inference, then employs answer extraction prompts to derive the final solution. We adopt PS+ prompting with more detailed instructions, defining the task questions as: (1) "identifying missing details in the current CAD design" and (2) "formulating user queries about these missing details", with the dialogue history incorporated into the questions to provide complete contextual information about the current CAD design. The reasoning prompt templates and answer extraction prompts are detailed in Table 16 of Appendix A.4.1.

MACRS(Fang et al., 2024): Multi-Agent Conversational Recommender System (MACRS) is a collaborative multi-agent framework that integrates four LLM-based agents to plan diverse dialogue behaviors, employing a feedback-aware reflection mechanism for agent adaptation. Specifically, MACRS incorporates four specialized agents designed to perform distinct dialogue functions: questioning, small talk, recommendation, and planning. For proactive querying in CAD design tasks, we define the user profile U_p in MACRS's memory module as representing a CAD modeler, whose objective is to create a fully detailed CAD model. The

system initializes the interaction using the user's initial CAD text description as input. The questioning agent, small talk agent, and recommendation agent generate three candidate responses (R_{ask}, R_{chat}, and R_{rec}) based on dialogue history and the user profile. The planning agent (π_p) then performs multi-step reasoning to select the most appropriate response from these candidates, determining the final system response R_s:

$$R_s = \pi_p(I_{\text{plan}}, R_{\text{ask}}, R_{\text{rec}}, R_{\text{chat}}, D_h, A_h)$$

where I_{plan} represents the instruction for the planning agent π_p, D_h denotes the dialogue history, and A_h corresponds to the history of dialogue actions. The prompt templates for each agent can be found in Table 11 of Appendix A.4.1.

A.4.1 Prompts in Baseline Methods

The prompts for standard prompting are listed in Table 12, for CoT in Table 13, for Proactive prompting in Table 14, for ProCoT in Table 15, for PS+ prompting in Table 16, and for MACRS in Table 11.

Main Category	Description
Parts	The number of parts included in the current CAD model.
Coordinate System	Defines the coordinate system of the part, which includes the following subtypes: **Euler Angles**: Three parameters (θ, ϕ, γ) that determine the orientation of the sketch plane. **Translation Vector**: Three parameters (τ_x, τ_y, τ_z) that describe the translation of the sketch plane.
Sketch	Defines the geometry of the 2D sketch. **Face**: Defines a face that contains a closed loop, which includes the following subtypes: **Loop**: Defines a loop composed of lines, arcs, or circles. **Line**: Contains start and end coordinates. **Arc**: Contains start, mid, and end coordinates. **Circle**: Contains center and top-most coordinates. **Sketch scale**: The scaling factor for the 2D sketch.
Extrusion	Defines the parameters for the extrusion operation, which includes the following subtypes: **Extrusion depth towards normal**: The extrusion depth in the direction of the normal of the sketch plane. **Extrusion depth opposite normal**: The extrusion depth in the direction opposite to the normal of the sketch plane.
Description	Provides a description of the CAD model, which includes the following subtypes: **Length**: The length of the part. **Width**: The width of the part. **Height**: The height of the part.
Assembly	The assembly relationships between parts.

Table 5: Main types and subtypes of CAD operations and parameters.

System Simulation
You are an AI assistant helping a user design a CAD model. The user has provided some information, but the {category} is missing. Generate a natural, conversational question asking the user to provide the missing information. Make your question specific to the context of the CAD model being designed. Keep your question concise and focused only on the missing {category}. Please only generate the question without any additional content.
User Simulation
You are a user designing a CAD model. An AI assistant has asked you about missing {category} information. Referential Design: {information about the missing type in the initial dataset} Please generate a natural, conversational response where you are playing the role of a user answering a question. Use the parameters from the referential design provided above in your response. Keep your answer concise and focused only on the {category}. Please only generate the answer without any additional content.

Table 6: Prompts for simulating system-User interaction.

User Initial Query Generation
I need to update a CAD model description query to reflect recent modifications. Certain information has been removed from the original JSON data. **Task:** Please revise the query to include only the information that remains available in the current JSON, ensuring all non-null details and relevant parameters are retained. **Original JSON:** {initial metadata} **Current JSON:** {metadata with partially missing types} **Original query:** {Initial Text Containing Complete CAD Information} **Requirements for the Updated Query:** 1. Include only the information still present in the current JSON. 2. Exclude any fields that are now null. 3. Maintain relevance to CAD model design. 4. Ensure clarity and conciseness. **Please only reply with the modified query without generating any additional text.** **Example:** **Original query:** "Create the first part of the CAD model, a rectangular prism with a curved side. Begin by creating a new coordinate system with Euler angles of [0.0, 0.0, 0.0] and a translation vector of [0.0, 0.0, 0.0]. Next, create a 2D sketch on the X-Y plane of the coordinate system. The sketch consists of four lines. The first line has a start point at [0.0, 0.0] and an end point at [0.6, 0.0]. The second line has a start point at [0.6, 0.0] and an end point at [0.6, 0.375]. The third line has a start point at [0.6, 0.375] and an end point at [0.0, 0.375]. The fourth line has a start point at [0.0, 0.375] and an end point at [0.0, 0.0]. Scale the 2D sketch by a factor of 0.6. Then transform the 2D sketch into a 3D sketch with Euler angles of [0.0, 0.0, 0.0] and a translation vector of [0.0, 0.0, 0.0]. Finally, extrude the 3D model with an extrusion depth towards the normal of 0.075 and an opposite normal depth of 0.0. Scale the sketch by 0.6. The first part of the CAD model has the following dimensions: a length of 0.6 units, a width of 0.6 units, and a height of 0.075 units, with a curved side. The part is centered in the image." **Modified query:** "Create the initial part of the CAD model, which is a rectangular prism featuring a curved side. Start by defining a new coordinate system with Euler angles set to [0.0, 0.0, 0.0]. Then, generate a 2D sketch on the X-Y plane of this coordinate system. The sketch comprises four lines: The first line starts at [0.0, 0.0] and ends at [0.6, 0.0]. The second line extends from [0.6, 0.0] to [0.6, 0.375]. The third line extends from [0.6, 0.375] to [0.0, 0.375]. The fourth line connects [0.0, 0.375] back to [0.0, 0.0]. Apply a scaling factor of 0.6 to the 2D sketch. Next, convert it into a 3D sketch with Euler angles of [0.0, 0.0, 0.0]. Proceed by extruding the 3D model with an extrusion depth along the normal direction of 0.075, while keeping the opposite normal depth at 0.0. Scale the sketch again by a factor of 0.6. The resulting first part of the CAD model has the following dimensions: a width of 0.6 units and a height of 0.075 units, maintaining a curved side. The part is centrally positioned within the image."

Table 7: Prompts for generating user CAD initial text.

User Simulation in the Dialogue-level Evaluation
Assume you are a user who needs to respond to the system's question.
The system's question is: "{system_message}"
Your designed corresponding parameters are: "{parameters}".
Please generate a response to the system's question based on your designed parameters.
If the parameter is "unknown" or other non-parameter information, respond with "unknown".

Table 8: Prompts for dialogue-Level evaluation of user simulation.

Dispatch Agent
You are a dispatcher agent. Your job is to classify the task into one of two categories based on the tags in the preceding task <>: 'ask' or 'summarize'. Return only the category name without any additional text.
Classify the following task into either 'ask' or 'summarize' category: {task}
Validation Agent
You are a validation agent. Your job is to determine if the output meets the requirements of the task. Return only 'yes' or 'no' without any additional text.
Task: {task}
Output: {output}
Does this output fulfill the task requirements? Answer with 'yes' or 'no'.
Asking Agent
You are an assistant for CAD operations. Generate a clear and concise question based on the task description to ask the user.
Generate a question based on task instruction:
{task instruction}
Summary Agent
You are an assistant for CAD operations. Generate a comprehensive summary based on the conversation history and the task description.
Generate a summary based on the task instruction and the conversation history.
Task instruction: {task instruction}
Conversation history: [...]

Table 9: Prompts for asking, summary, validation, and dispatch.

Workflow Generation

You are a helpful and intelligent task planner, and your target is to decompose the assigned task into multiple subtasks for task completion and analyze the precedence relationships among subtasks.

At the beginning of your interactions, you will be given the task description and actions list you can take to finish the task, and you should decompose the given task into subtasks that can be accomplished using the provided actions. And then, you should analyze the precedence relationships among these subtasks, ensuring that each subtask is sequenced correctly relative to others. Based on the analysis, you should construct a workflow consisting of the identified subtasks to complete the task.

You should use "Node:
1. <subtask 1>
2. <subtask 2>" to denote subtasks, and use (x,y) to denote that <subtask x> is a predecessor of <subtask y>, (START,x) to indicate the beginning with <subtask x>, and (x,END) to signify the conclusion with <subtask x>. Remember that x, y are numbers.

Your response should use the following format:
Node:
1.<subtask 1>
2.<subtask 2>
...
Edges:(START,1) ... (n,END)

Now it's your turn.

Task: Refine the user's initial CAD design to the greatest extent possible.
The user's initial CAD design: [...]
The action list you can take: ['ask','summary']
'ask' means to inquire from the user to obtain specific CAD modeling information, such as coordinate system, sketch, extrusion, assembly, etc., as well as more detailed information like Euler Angle, extrude depth opposite normal, sketch scale, length, etc.
'summarize' refers to formulating a CAD design plan by integrating the CAD design information from the conversation history.

Remember that the format of the Node must be:
Node:
1: <action type> subtask 1
2: <action type> subtask 2
...
Edges:(START,1) ... (n,END)

Table 10: Prompts for generating workflows.

Asking Agent
CAD details design task: Based on the conversation history and CAD design priority rules, generate a question for the user regarding the highest-priority CAD design detail that has not yet been completed in the conversation history. The given conversation history is [...] You are a knowledgeable and enthusiastic CAD design details recommender chatbot. Your goal is to engage in friendly, casual conversation about CAD design details. Follow these guidelines: - Don't say that you can't give recommendations directly. - As you are a chatbot, speak casually but not too informally. - Respond appropriately to the seeker's answers in line with your role. You should elicit CAD design details by asking questions. If user asked any question at previous turn, You should answer the question. If there is nothing to respond to, please use the response "Alright!" Response should be equal or less than 15 words.
ChitChat Agent
CAD details design task: Based on the conversation history and CAD design priority rules, generate a question for the user regarding the highest-priority CAD design detail that has not yet been completed in the conversation history. The given conversation history is [...] You are a knowledgeable and enthusiastic CAD design details recommender chatbot. Your goal is to engage in friendly, casual conversation about CAD design details. Follow these guidelines: - Don't say that you can't give recommendations directly. - As you are a chatbot, speak casually but not too informally. - Respond appropriately to the seeker's answers in line with your role. You should elicit CAD design details by asking questions. If user asked any question at previous turn, You should answer the question. If there is nothing to respond to, please use the response "Alright!" Response should be equal or less than 15 words.
Planning Agent
CAD details design task: Based on the conversation history and CAD design priority rules, generate a question for the user regarding the highest-priority CAD design detail that has not yet been completed in the conversation history. The given conversation history is [...] You are a knowledgeable and enthusiastic planning agent decide which response to generate. Your goal is to engage in friendly, casual conversation about CAD design details. Follow these guidelines: - Don't say that you can't give recommendations directly. - As you are a chatbot, speak casually but not too informally. - Respond appropriately to the seeker's answers in line with your role. response from asking agent:{asking agent response} response from chit-chatting agent:{chit-chatting agent response} From the conversation history, determine whether user CAD design details is sufficient or not. Must choose one of the candidate responses based on three different dialogue acts. These three dialogue acts are: asking, or chit-chatting. If there is nothing to respond to, please use the response "Alright!"

Table 11: Prompts for MACRS.

Standard Prompting
Your current task is to determine the user's intentions and satisfy their needs based on the provided conversation between the user and the system. The given conversation history is [...]. Please generate the response. If there is nothing to respond to, please use the response "Alright!"

Table 12: Prompts for standard prompting.

CoT
Your current task is to determine the user's intentions and satisfy their needs based on the provided conversation between the user and the system. The given conversation history is [...]. Let's think step by step. If there is nothing to respond to, please use the response "Alright!"

Table 13: Prompts for CoT.

Proactive Prompting
Your current task is to determine the user's intentions and satisfy their needs based on the provided conversation between the user and the system. The given conversation history is [...]. You may choose to either inquire about a missing detail in the current CAD design or, if the current CAD design is already complete, simply respond with "Alright!"

Table 14: Prompts for proactive prompting.

ProCoT
Your current task is to determine the user's intentions and satisfy their needs based on the provided conversation between the user and the system. The given conversation history is [...]. First, let's analyze step by step whether there are any missing details in the current CAD design. Then you may choose to either inquire about a missing detail in the current CAD design or, if the current CAD design is already complete, simply respond with "Alright!"

Table 15: Prompts for ProCoT.

PS+ Prompting
Your current task is to determine the user's intentions and satisfy their needs based on the provided conversation between the user and the system. The given conversation history is [...]. Let's first understand the user's intentions and devise a plan to satisfy their needs. Then, let's carry out the plan to satisfy their needs step by step.

Table 16: Prompts for PS+ prompting.