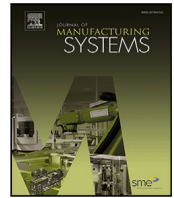




Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Technical paper

Chat with MES: LLM-driven user interface for manipulating garment manufacturing system through natural language

Zhaolin Yuan ^{a,b,c}, Ming Li ^{c,d,e,f}, Chang Liu ^a, Fangyuan Han ^a, Haolun Huang ^h, Hong-Ning Dai ^g

^a Institute of Artificial Intelligence, University of Science and Technology Beijing, Beijing, China

^b Shunde Innovation School, University of Science and Technology Beijing, Guangdong, China

^c Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, China

^d Research Institute of Advanced Manufacturing, The Hong Kong Polytechnic University, Hong Kong, China

^e Research Centre for Digital Transformation of Tourism, The Hong Kong Polytechnic University, Hung Hom, Hong Kong, China

^f Centre for Transformative Garment Production, Hong Kong Science Park, Pak Shek Kok, Hong Kong, China

^g Department of Computer Science, Hong Kong Baptist University, Hong Kong, China

^h School of Intelligent Systems Science and Engineering, Jinan University, Zhuhai, China

ARTICLE INFO

Keywords:

Interactive Manufacturing Execution System

Large language model

Text2SQL

LLM-Agent

Human-computer interaction

ABSTRACT

This paper presents Chat with MES (CWM), an AI agent system, which integrates LLMs into the Manufacturing Execution System (MES), serving as the “ears, mouth, and the brain”. This system promotes a paradigm shift in MES interactions from Graphical User Interface (GUI) to natural language interface”, offering a more natural and efficient way for workers to manipulate the manufacturing system. Compared with the traditional GUI, both the maintenance costs for developers and the learning costs and the complexity of use for workers are significantly reduced. This paper also contributes two technical improvements to address the challenges of using LLM-Agent in serious manufacturing scenarios. The first one is Request Rewriting, designed to rephrase or automatically follow up on non-standardized and ambiguous requests from users. The second innovation is the Multi-Step Dynamic Operations Generation, which is a pre-execution planning technique similar to Chain-of-Thought (COT), used to enhance the success rate of handling complex tasks involving multiple operations. A case study conducted on a simulated garment MES with 55 manually designed requests demonstrates the high execution accuracy of CWM (80%) and the improvement achieved through query rewriting (9.1%) and Multi-Step Dynamic operations generation (18.2%). The source code of CWM, along with the simulated MES and benchmark requests, is publicly accessible.

1. Introduction

The Manufacturing Execution System (MES) has undergone significant evolution since its inception in the late 19th century, becoming an integral part of modern manufacturing due to its crucial role in production process management and optimization [1]. With the rapid development of MES in incorporating more complicated functions and operational workflows, the methods by which workers interact with MES warrant more attention [2]. Efficient and friendly interaction will significantly reduce the workload of workers and the training cost for new employees.

Since the Graphical User Interface (GUI) appeared [3], it has become the primary way for workers to operate an MES. Users manipulate the graphical elements within a specific system page and receive feedback from the changeable elements. The GUI is typically

developed alongside the core components of the MES by the software engineering team. Although the GUI has evolved over several decades and is sufficiently mature, it still suffer some limitations in real developments and utilization. The first limitation arises from the programmers responsible for coding. Because MES is a kind of non-standardized and customized software, it demands substantial labor effort for developing and maintaining the GUI. Even after project delivery, users may still present various new requirements and the developers have to intervene and revise the logic of software and data access interfaces. Many MES development projects fail and are abandoned by factory users because the development team refuses to continue maintaining the systems [4]. The second issue is the learning cost and operational complexity for users. A new worker must thoroughly learn from the software guidelines and undergo training before performing operations.

* Corresponding author.

E-mail address: ming.li@polyu.edu.hk (M. Li).

<https://doi.org/10.1016/j.jmsy.2025.02.008>

Received 4 December 2024; Received in revised form 22 January 2025; Accepted 16 February 2025

0278-6125/© 2025 The Authors. Published by Elsevier Ltd on behalf of The Society of Manufacturing Engineers. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

In the utilization stage, the workload is also significant due to numerous repetitive and tedious operations. For systems designed for extremely complicated manufacturing processes, the cost for developing and using GUI is much more substantial [5].

Advanced software engineering methods are suggested to address challenges in development and implementation. From the perspective of reducing the labor cost of software development and maintenance, emerging software development techniques have explored paradigms such as no-code [6] and low-code [7] development platforms. However, these platforms face usability-plasticity dilemma. Component encapsulation allows non-programmers to swiftly create a software system through drag-and-drop interfaces. However, excessive encapsulation will adversely affect extensibility, making it impossible to meet customized needs. From a worker's experience standpoint in relation to usage, Robotic Process Automation (RPA) [8] has garnered substantial interest as a means of automating repetitive and labor-intensive business processes through software bots. However, the functions of most RPA engines are generally limited to specific workflows and certain GUI pages. They merely serve as supplemental tools to the GUI rather than substitutes for addressing the underlying drawbacks of GUI.

Recent advances in large language models (LLMs) have the potential to revolutionize the interaction between humans and information systems. This can be achieved by understanding the user requirements expressed in natural language and invoking the services within the information system. Some successful cases have been observed in Web Shopping [9] and Web Browsing [10]. Accessing an MES through a conversational interface powered by an LLM is straightforward. However, manufacturing is a typical serious and complicated scenario. It introduces two technical challenges when utilizing LLMs to manipulate a MES. First, the inherent flexibility of natural language often leads to ambiguity in user input [11]. It contradicts the need for precise instructions, which are essential for reliable operations in MES. An important research topic is enabling LLM systems to identify accurate information from incorrect or ambiguous queries. From the perspective of handling a request, a user request may involve numerous database operations and service calls, presenting a challenge for existing LLMs to accurately execute all steps. Manufacturing is a serious application domain and many operations involve atomicity and transactional properties. For the LLM Agent, the procedure for processing requests and the operations it executes have to be governed by pre-established factory management guidelines. Inaccurate operations can lead to severe consequences. These two issues must be addressed before using an LLM-based interface to manipulate the real MES.

This paper presents a system entitled "Chat with MES" (CWM) to replace the traditional GUI of MES with conversational interactions. Fig. 1 presents the differences between using CWM and traditional GUI to manipulate MES. The CWM obviates the necessity for developing both front-end and back-end software, in addition to reducing the expenses associated with system maintenance. Furthermore, we propose two technical advancements to address the challenges associated with ambiguous user inputs and complicated tasks. **(1) Query Rewriting:** This module addresses the problem of ambiguous and inaccurate user inputs by maintaining a private context base to record named entity information and rewrite users' requests. **(2) Multi-step Dynamic Operations Planning and Execution:** To effectively handle complicated user requests, the Multi-step Dynamic Operations Planning and Execution pre-plans a chain of basic executable operations with parameter placeholders and execute the operations one-by-one. Each operation involves either invoking a service from the MES's predefined services pool, generating an SQL command for direct access to the production database, or performing reasoning and inference based on context. The initial decomposition significantly enhances the controllability and reliability of the LLM and facilitates the incorporation of prior knowledge. We evaluated the CWM on a simulated garment MES and a manually designed questions-and-answers dataset involving 55 requests. The garment manufacturing execution system (MES) is equipped with a

relational database consisting of 16 tables, encapsulating the typical entities encountered in actual garment production. The designed requests encompass the major cases that may arise during actual use by the two user roles: workers, and managers. The quantitative findings indicate a high execution accuracy of the CWM at 80%, along with the enhancement achieved through query rewriting, which contributes an improvement of 9.1%, and the generation of Multi-Step Dynamic operations, which results in an advancement of 18.2%.

The contributions of this paper are threefold.

(1) We propose "Chat with MES", a novel MES operation paradigm that provides users with an interface for manipulating the MES system via natural language.

(2) The Multi-step Dynamical Operations Generation is proposed to handle complex user queries by executing a sequence of basic operations. This technique ensures the controllability and reliability of CWM in serious manufacturing scenarios.

(3) An experimental environment was established using a simulated garment manufacturing system. Furthermore, a new question-and-answer dataset comprising 55 manually crafted requests was created to serve as a new benchmark for operating a MES through conversation. The evaluation of the CWM demonstrates an impressive execution success rate of 80%, significantly exceeding the original GPT-4's performance, which registers at 60%.

The subsequent sections are organized as follows: Section 2 provides a review of existing literature concerning user interfaces within MES, as well as the application of Large Language Models (LLM) in manufacturing processes and techniques associated with LLM. Section 3 depicts the technical details of CMS including the Query Rewriting and Multi-step Dynamical Operations Generation. The experiment Section 4 provides an introduction to the simulated garment manufacturing system used for testing, along with the corresponding Q/A dataset. Next, the experimental results and analysis are presented. Section 5 discusses the limitations of CWM and provides insight into future research. Finally, Section 6 concludes this article.

2. Related work

2.1. Limitations in human-MES interaction

An integrated management interface is indispensable for each MES to ensure that the distinct components are systematically organized and accessible. Graphical user interface (GUI) serves to convey information to users and facilitate human decision-making in production management [12]. This section examines recent research concerning the creation and utilization of GUI.

In development terms, manufacturing is a tailored scenario, requiring customization of transaction logic and the GUI for each factory. To address the significant cost associated with customized development, many software development techniques have explored paradigms such as no-code [6] and low-code [7] development platforms. These platforms integrate tools for a specific kind of applications and enable end developers to create and configure corresponding applications by themselves, such as ERP, MES, and digital twin systems [13]. Although these techniques are known to be easily developed, deployed, and maintained, they nevertheless face limitations in extensibility, rendering them unsuitable for complicated manufacturing scenarios [6].

In the utilization perspective, intricate manufacturing systems are associated with significant learning expenses and operational complexities for users. As a revolutionary technique to simplify the operational processes and enhance user experience, Robotic Process Automation (RPA) [8] has garnered substantial interest as a means of automating repetitive and labor-intensive business processes through software bots. Recent work by Holzinger et al. [14] highlights the importance of human-centered Artificial intelligence (AI) interfaces. Current RPA systems are most designed to adhere to pre-defined rules and workflows of the GUI [15]. These functionalities are typically implemented

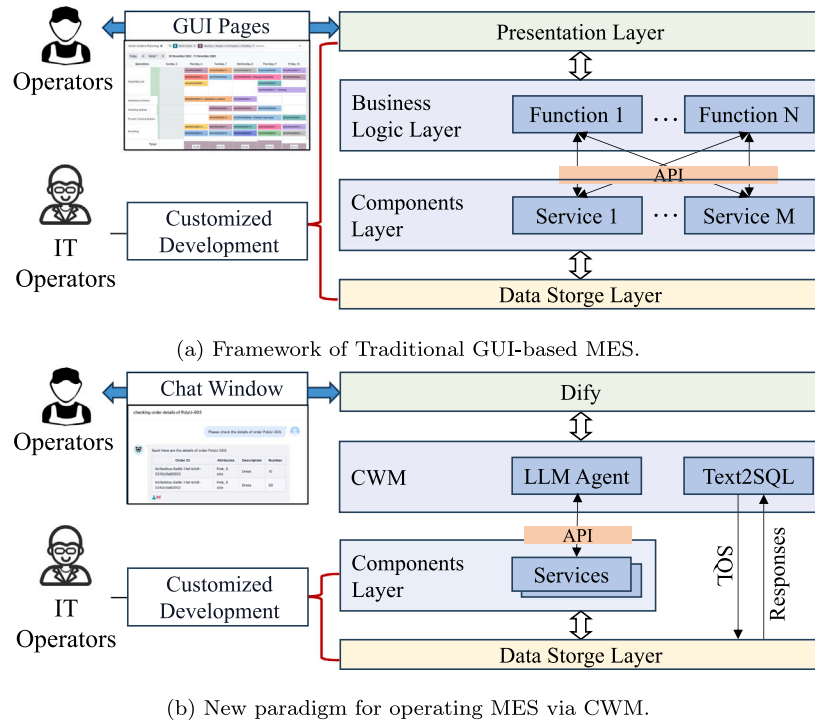


Fig. 1. The comparison between the traditional GUI-based framework and CWM.

through drag-and-drop interfaces, screenplay recording, or automation frameworks such as Selenium [16]. **The systems are intrinsically system-centered, instead of human-centered. Users must passively adhere to the rules and provide specific information, yet they cannot actively express their desires [15]. Meanwhile, RPA remains dependent on GUI interfaces, the cost of developing GUI is unavoidable.** Recent years have witnessed remarkable progress in computer vision, LLM, and LLM-Agent, which have opened up new possibilities for integrating LLMs with RPA systems towards enabling them to perceive and autonomously interact with complex applications. AI can be incorporated into the interface layer for MES to create better user-friendly experiences. Mantravadi et al. [17] proposed the concept of AI-based chatbot assistance, which aims at production coordination by assisting the shop floor workforce and providing easy information extraction in manufacturing. Their developed prototype shows that MES users can benefit from an interactive chatbot by creating more dynamic and free experiences. Colabianchi et al. [18] presented an integrated conceptual architecture to develop industrial conversational agents and illustrated the elements needed for the development. They also overlooked the application of LLMs in manufacturing conversation systems. **The research discussed above primarily stays at the conceptual or prototype development stage. A significant gap exists between the conceptual design and the development of a fully operational conversational manufacturing system.**

2.2. LLM-based Text2SQL

One primary function of MES is managing the production data by accessing the production databases. To address the wide range of potential queries from users, the proposed CWM will act as a database operating agent between the user and the database. Text2SQL parsing [19] aims at converting natural language questions into executable SQLs. Recent advances in large language models (LLMs), such as GPT-4 and Claude-2 have shown impressive results in this task. Some production-ready projects are also developed, such as DB-

GPT [20] and sqlcoder.¹ However, according to the test results from a large-scale Text2SQL benchmark, the most effective Text2SQL models achieve 54.89% in execution accuracy, whereas human experts achieve approximately 92.96% [21].

Intrinsically, Text2SQL is a code generation technology that transforms detailed and precise natural language requests into SQL commands. In practical applications, it must incorporate business-related prior knowledge alongside Text2SQL to accurately identify users' requirements and generate executable SQL commands. By designing specific workflows and incorporating additional knowledge through prompt engineering, the performance of Text2SQL in practical applications has been improved significantly. As an extension of Text2SQL, ChatBI is proposed to convert Natural Language to Business Intelligence [22], effectively addressing challenges related to a large number of columns. To address complex queries requiring multi-step database operations rather than a single SQL command, ChatDB [23] and DinSQL [24] augment LLMs to generate multi-step and dynamical SQL instructions for database manipulation. **However, existing LLM-based business intelligence systems are generally applied in non-intrusive data query scenarios to avoid the catastrophic consequences of hallucinatory data editing. In our study, manipulating MES inevitably involves a number of update, insert, and delete operations. The chosen operations and their sequence must be absolutely accurate. This places extremely high requirements on the agent. Not only does the agent need to understand all database's schema and their relationships, but it also needs to follow the production guidelines. The experimental section demonstrates that current leading general LLMs lack the ability to manage these intricate tasks.**

¹ <https://github.com/defog-ai/sqlcoder>.

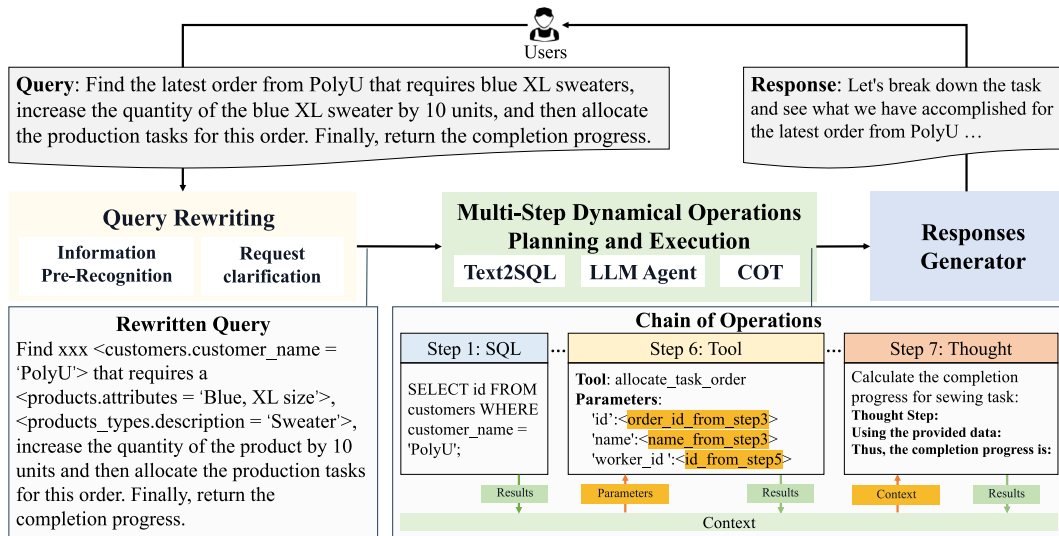


Fig. 2. Technical framework of CWM.

2.3. Application of LLM in manufacturing systems

In the context of Industry 4.0, the rapid emergence of LLM has brought forth transformative solutions for manufacturers in processing and leveraging their data efficiently. LLMs are increasingly being used as conversational agents, serving as an intuitive interface between humans and the manufacturing environment [18]. This section aims to analyze the differences between this paper and other studies that apply large models in the manufacturing scenarios. Sun et al. [25] propose an AI-driven digital-twin (DT) multi-agent architecture powered by LLMs to enhance equipment maintenance scenarios, offering novel insights into predictive and preventive maintenance strategies. Furthermore, the application of LLMs extends beyond conversational agents and into more specialized areas such as robotics programming. In a ROS (Robot Operating System) environment, Xia et al. demonstrate how LLM-driven code generation can be harnessed to enable more efficient robot operations [26]. Li et al. [27] further explore the integration of LLMs with Knowledge Graphs, leveraging the symbiotic relationship between these technologies to aid in sourcing and identifying smaller manufacturing partners. Additionally, Xu et al. [28] incorporate LLMs in collaborative design processes where LLMs generate design prompts for tools like DALL-E, facilitating the rapid and seamless creation of precise visual schemes. In the realm of industrial sustainability, Wu introduces the ProcessCarbonAgent framework, which uses LLMs to analyze root causes of carbon emissions and improve decision-making for industrial emission management [29]. These studies illustrate the transformative influence LLMs hold within numerous aspects of manufacturing, ranging from robot manipulation and design collaboration to supply chain enhancement and cause analyses. The LLM functions not only as a chatbot or writer but also as a coordinator that integrates services and processes information from multiple sources. Table 1 highlights the differentiating factors of CWM in comparison to preceding research efforts. Given that each related work addresses unique research questions, we analyze the capabilities of LLMs that are employed. CWM represents the first dialogue-oriented open-source framework endowed with functionalities including Code Generation, database access, and Tool invocation.

3. Chat with MES

The overall framework of CWM is illustrated in Fig. 2. Upon receiving a textual request from the user, the CWM undergoes three stages, Request Rewriting, Multi-Step Dynamical Operations Generation, and Responses Generation. To eliminate ambiguities from the user's original

Table 1

Comparison of LLM-based applications in the field of manufacturing.

Objective	Ref.	Chat	Text generation	Code generation	SQL generation	Tool invocation
Conversational chatbot	[18]	✓	×	×	×	×
Equipment maintenance	[25]	×	✓	×	×	×
Answering queries & generating robot code	[26]	✓	×	✓	×	×
Manufacturing service discovery	[27]	✓	×	×	✓	×
Collaborative design	[28]	✓	✓	×	×	×
Carbon emission management	[29]	✓	✓	×	×	×
Chatting with MES	Ours	✓	✓	✓	✓	✓

commands, the Request Re-writer uses an LLM to substitute entities in the request with precise descriptions that can be indexed in the database. The Multi-step Dynamical Operations Generation develops a plan with step-by-step operations to handle the request and dynamically updates these operations. The final stage, Responses Generation, summarizes the results from executing the operations and generates a textual response in markdown format for the user. This section introduces the technical details of the Request Re-writer and Multi-step Dynamical Operations Generation processes.

3.1. Request re-writer for preprocess

When users interact with the CWM, they may input non-standardized or ambiguous information, which can introduce significant information bias in subsequent database operations. For example, both "PolyU" and "Hong Kong Polytechnic University" refer to the same entity associated with a record in the customer table. If the real name stored in the database is "PolyU", once the user inputs "Hong Kong Polytechnic University" as the customer information, a generated SQL command may carry a wrong retrieve condition. Therefore, the first technical module of CWM is developed to standardize the query for Entity disambiguation.

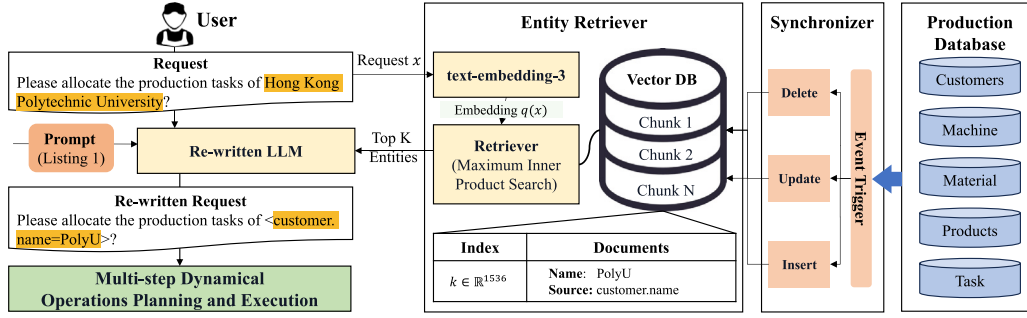


Fig. 3. Workflow of request re-writer.

The workflow of request re-writer is shown in Fig. 3. The Entity Retriever is the core of the re-writer which dynamically maintains the embedding of registered entities from the production database in a vector DB. Each entity is stored as a chunk comprising a key and the document. The document consists of the entity's name appended to the database schema and column name. The key represents the text embedding of the document generated by the OpenAI's text-embedding-3-large model. A synchronizer is introduced to monitor database updates, triggering to insertion, updates, or deletions of the chunks in the vector DB. For instance, if a customer modifies their name, it triggers a real-time update of the chunk associated with the customer's name.

Listing 1: Prompt for rewriting request

```
# role:
You are a professional question optimization
model. You can replace entity names in
questions with standard entity names.

# Standard Entity Names:
The correspondence between entity names and
possible standard entity names is as follows:
{standard_entity_mapping}

# Output Requirements:
1. Convert entity names to the most probable
standard entity names.
2. Replace the original entity names with
"<entity_type = standard_entity_name>"

# Output Examples:
1. Original Question:
Orders of PolyU
Rewritten Question:
<customers.customer_name = 'PolyU'> orders
2. Original Question:
What are the sewing tasks that use silk
Rewritten Question:
What are the sewing tasks that use
<products.product_name = 'Silk'>
3. Original Question:
What does the main warehouse have
Rewritten Question:
What does <warehouse.warehouse_name = 'Main
Warehouse'> have
4. Original Question:
What tasks does the Advanced Sowers group have
Rewritten Question:
What tasks does <working_group.name = 'Advanced
Sowers'> have

{question}

# Rewritten Question:
```

After the user inputs the request x , The text-embedding-3-large model in the Entity Retriever is utilized to convert textual query into query vector $q(x)$. Subsequently, we identify the chunks with keys most similar to the query embedding $q(x)$ by employing the Top-K Maximum Inner Product Search. Finally, we employ the LLM's zero-shot Named Entity Recognition capability to identify entities from

the user's request [30] and use the retrieved chunks to rewrite the entity information with standardized format, schema-field = "information" where the schema name, the field denotes the field name, and the "information" is the matched entity. For example, the "Hong Kong Polytechnic University" will be transformed to {customer.name = "PolyU"}. The prompt for re-writing is shown in Listing 1. If the information provided by the user does not match any registered entity, it indicates that the user's query is not recognized, and the CWM will return to generate a question to the user for confirmation.

3.2. Multi-step dynamical operations planning and execution

In an MES, a request, such as assigning a task to a worker or placing an order for a customer, typically involves a series of complex operations. To increase the controllability and reliability of CWM in handling complicated requests, this paper introduces **Multi-step Dynamical Operations Generation** to transform the natural language input from users to a series of executable basic operations in MES. Fig. 4 illustrates the procedure for generating and executing multi-step dynamical operations. This section begins with defining operations, followed by outlining pre-planning and execution processes. Subsequently, it delves into few-shot learning for integrating prior production guidelines, concluding with the final response generation.

3.2.1. Definition of operations

A basic operation is either a request for invoking a tool, or executing a SQL command for operating the database, or a thought step. As the most fundamental type of operation, 'SQL' represents accessing the MES database by generating an SQL command tailored to the specific purpose of the step. The SQL commands are generated by the LLM, which is associated with Text2SQL, a significant and emerging research area in the application of LLMs.

As the second type of operation, the tools are associated with available API services provided by the MES and identified by the LLM agent according to four labels, **tool name**, **tool description**, **parameters**, and **parameter description**. We use LLM to autonomously decide which tool to invoke and determine the appropriate parameters based on the user's query or the retrieved results from the database. In a MES, most tools comprise a series of data editing operations. Theoretically, if the Text2SQL module is intelligent enough, all requests excluding those that require external communication can be resolved by directly manipulating the databases. However, an MES system always applies a set of production-related constraints on data editing. For example, assigning a production task to a worker is always associated with inserting a material allocation record in another table. As an illegal behavior, freely generating SQLs and arbitrarily changing the records in the databases are prohibited for MES. Thus, it is essential to implement the tool operation as a means of encapsulating SQLs, ensuring that CWM actions adhere strictly to production protocols and policies.

The 'Thought' operation, the third category, complements the previous two by facilitating complex logic inference and mathematical

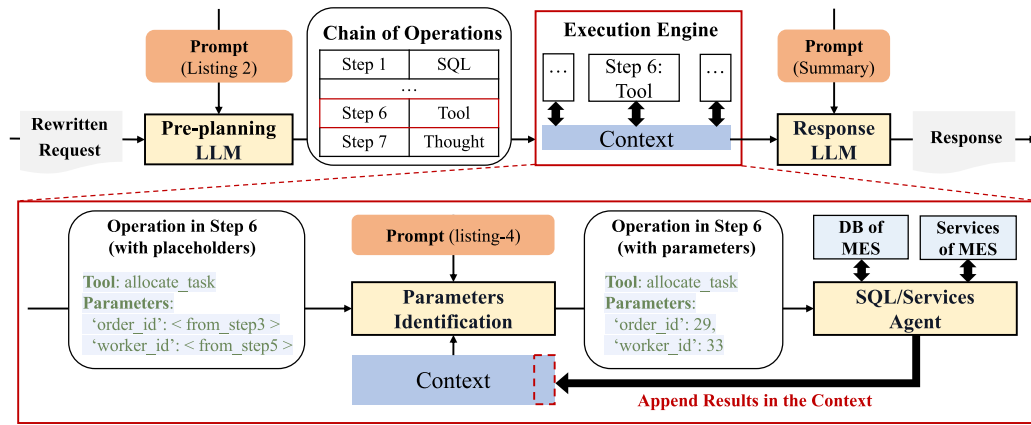


Fig. 4. Multi-step dynamical operations generation and execution.

calculations. When the user's request involves summarization or statistical analysis, both 'Tool' and 'SQL' are only capable of retrieving relevant data. Based on this, the 'Thought' step will generate a local CoT and infer the final result. An example will be shown in Section 4.2.

Listing 2: Prompt for generating the planned multi-step Operations

```
You are Chat With MES, a powerful AI assistant, a
variant of ChatGPT that can
1) decide to invoke some provided tools to solve
the query.
2) utilize the database of the Manufacturing
Execution System as external symbolic memory.
For any user query, you should always prioritize
using the given tools to complete it.
The following are the tools you can use, including
their names, descriptions, and input args.
"""
tool_allocate_task_order: Allocate the cutting
task and sewing tasks for all products
belonging to the order, and return
the allocated tasks and corresponding working
groups.
{
  'order_id': {'title': 'Order Id',
    'description': 'The order ID, an
    integer.', 'type': 'integer'},
  'order_name': {'title': 'Order Name',
    'description': 'The order name, a
    string.', 'type': 'string'}
}
% Description of other Tools.
"""
Only if you cannot handle the command/query with
any given tools, You are authorized to
directly access the database.
In this case, you are an expert in databases,
proficient in SQL statements and can use the
database to help users. The details of tables
in the database are delimited by triple quotes.
"""

-- Customer Table
CREATE TABLE 'customers' (
  'id' VARCHAR(36) PRIMARY KEY COMMENT 'Customer
  ID',
  'customer_name' VARCHAR(255) NOT NULL COMMENT
  'Customer Name'
);

% Other table creation scripts.
```

```
Please tell me what basic operations, including
sql, tool, and thought, should I use in order
to respond to the "USER INPUT". If it needs
multiple operations, please list them step by
step concisely, and indicate whether the
operation involves calling a tool or
accessing a database via SQL. If there is no
need to use any operations, reply to the "USER
INPUT" directly.
At all times, you should prioritize using the
provided tool. If the tool does not meet the
requirements, only then may you resort to
using SQL.
The output should be a markdown code snippet
formatted in the following schema, including
the leading and trailing "\'\'\'" and "\'\'\'":
'''
Step1: <Description of first step>
SQL 'SQL command for step1'

Step2: <Description of first step>
SQL 'SQL command for step2'

Step3: <Description of second step>
Tool 'Tool name, arguments for tool, purpose of
using this tool'

Step4: <Description of fourth step>
Thought 'Purpose of this intermediate reasoning
step, what intermediate results are expected.
and how to get them based on the results from
the previous steps.'
'''

Backticks are important and must be added at the
beginning and end of the command for every
step!

Here are some examples:

USER INPUT: Retrieve all orders and item details
for the Customer PolyU
ANSWER:
'''
Step1: Retrieve the customer ID for the customer
with the name "PolyU"

Execute:
SQL 'SELECT id FROM customers WHERE customer_name
= 'PolyU';'

Step2: Retrieve the details of products for each
order, including product attributes and type
description
```

```

Tool 'tool_find_order_details, {'order_id':
  <order_id>}, Find the product details of an
  order'
'''
% Omitted other few-shot examples.

USER INPUT: Find the latest order from
  <customers.customer\_name = 'PolyU'> that
  requires a <products.attributes = 'Blue, XL
  size'> <products\_types.description =
  'Sweater'>, increase the quantity of the
  <products.attributes = 'Blue, XL size'>
  <products\_types.description = 'Sweater'> by
  10 units, and then automatically allocate the
  production tasks for this order.

ANSWER:

```

3.2.2. Pre-planning operations

In Multi-step Dynamical Operations Planning and Execution, the CWM will pre-plan the operations prior to execution, Based on the prompt template from ChatDB [23], we design the prompt for generating multi-step operations. As shown in the prompt Listing 2, it establishes the context and objectives for the LLM, including the instructions to the LLM, the details on the database schema and available tools, the output formats. Certain redundant details are excluded, highlighted in red font. The content following USER INPUT represents the output from the query re-writer module.

In addition to the data schema and the description of available tools, CWM needs to understand the production guidelines, such as the order of accessing databases, the format of the data inserted into the table and some definitions and abbreviations in the garment industry. Because each garment factory has customized guidelines and the knowledge base needs to be maintained dynamically, this study employs the training-free few-shot learning [23] to inject the specific knowledge in the model. It aids the CWM in understanding the MES's operational guidelines and the relationships among schema. A deep understanding of the database structure is crucial for CWM to generalize effectively in handling arbitrary user inquiries. Few-shot learning is accomplished by presenting some representative examples as system prompts. Each example includes the chain of operations, the correct tools to invoke, and the SQL commands required to resolve the request. We introduce three few-shot learning examples in total for covering the primary difficult and representative cases of the garment MES, (1) placing orders, (2) analyzing task completion progress, and (3) retrieving product details of an order.

Listing 3: Parameters Identification prompt

```

You are now the following python function:
# "Find useful information in the results of the
  previous operating statement, and replace <>
  with the corresponding information."
"If the operation type is 'Tool' and the tool must
  be invoked multiple times with varying
  parameters, "
"please output a list of tool operations with
  distinct parameters."

def populate_operation_statement(operation_str:
  str, previous_operation_results:
  list[list[dict]]) -> list[str]:
Only respond with your 'return' value. Do not
  include any other explanatory text in your
  response.",

# Operation
INSERT INTO order_product (order_id, product_id,
  number) VALUES ('<order_id>',
  '<red_tshirt_product_id>', 50)

# Historical Context

```

3.2.3. Operation execution and dynamical parameters retrieval

After the initial chain of operations are generated, the CWM employs regex matching to extract the operations command and their types and execute each operation one by one. Apart from the first operation, subsequent operation commands might contain placeholders, such as "<customer_id_step1>", that must be determined based on the context. Before executing each command, CWM identifies these placeholders and uses LLM with prompt 3 to retrieve the specific parameters from the conversion context to update the operation command. Finally, the operations with parameters are executed by accessing the database and the services of MES. As shown in Fig. 4, the results from executing each operation are appended in a conversation context.

3.3. Response generation

After executing all operations, the context which consists of all intermediate results produced from each operation, will be fed to the Responses Generator. According to the original request from the user, the LLM-based generator will produce the final textual responses. The structured records will be automatically organized as tables with the markdown format to give a good reading experience.

4. Experiment

Different types of users have diverse business requirements when utilizing MES, such as creating or querying orders, managing production tasks and material allocation, or handling inventory management. In the experimental section of this paper, we employ the garment manufacturing as the case scenario. Based on an existing, well-established garment MES platform, we simulate the operational processes of a clothing factory and generate simulated raw manufacturing system data. Building upon this experimental platform and dataset, we simulate common requests when users utilize an MES and investigate whether CWM can effectively comprehend and process various typical business requests described in natural language in the absence of a graphical user interface (GUI). With restricted external service access and database permissions, we examine CWM's behavior and accuracy in operating within the MES system and database to fulfill user requests.

The experiment results demonstrate that CWM performs a high execution accuracy, 80%, significantly outperforming Text2SQL baseline models. Multi-step operations planning and execution significantly contributes to a high execution success rate.

4.1. Experimental setup

4.1.1. Simulated MES

To accommodate the utilization scenarios of CWM and facilitate quantitative comparisons with benchmarks, we developed a simulated garment MES by drawing upon a comparable product available in Huawei Cloud.² The constructed MES system incorporates the fundamental management components typical of garment manufacturing facilities, encompassing orders, customers, workers, materials, cutting operations, and sewing operations. The database in this simulated MES is a relational database involving 16 tables. Their relations and foreign keys are shown in the ER diagram 5. We use the GPT-4 to generate the simulated data in the database, resulting in a total of 312 records.

Table 2 enumerates the tools accessible to the LLM Agent. Given that the MES serves as a simulated platform for demonstration purposes, only four fundamental tools are presented to maintain simplicity.

4.1.2. Requests dataset

Based on the database schema of the MES and the rows inside, we designed a requests dataset including 55 requests covering common

² YiZhi MES garment intelligent production management platform in HUAWEI CLOUD.



Fig. 5. ER diagram.

Table 2
Available tools in the simulated garment MES.

Tool	Description	Parameters
find_order_details	Find the product details of an order by the order ID.	order_id : ID of order
allocate_task	Automatically allocate product order tasks to workers	order_id : ID of order order_name : name of order
store_materials	Automatically store materials in a free inventory.	material_id : ID of material number : number of stored materials
complete_task	Workers reporting the completion of specific production tasks, it automatically adjusts the quantities of finished products in the warehouse.	task_id : ID of task task_type : sewing or cutting quantity : number of completed products or work-in-progress
get_busy_workers	Find all working groups and their task counts.	None

operations in a garment factory: placing orders, checking inventory, work assignment, materials allocation, and so on. We use the term 'request' instead of 'question' because it involves many commands rather than simply posing a question. In actual production settings, requests may be initiated by two categories of users, workers and managers. Fig. 6(a) illustrates the distribution of the identities among

55 requests. As the production manager is responsible for coordinating production tasks, workers, and resources, he is the primary user of CWM. Workers use this system to confirm allocated tasks and report their working progress.

Theoretically, all requests can be addressed using basic database operations—Inserting (I), Updating (U), and Querying (Q) and their

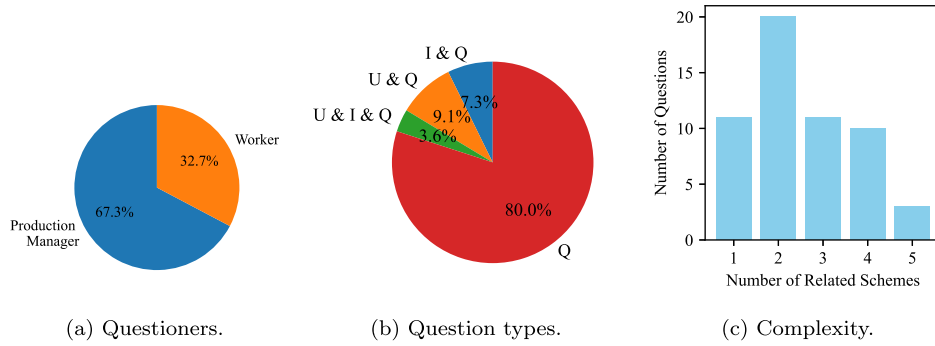


Fig. 6. The distribution of request from the perspective of the operation types, identity, and complexity. The complexity is represented by the number of tables that need to be accessed to handle this question.

combination to adjust the records in the database. Fig. 6(b) presents the distribution of the required command types involved in all requests. Most requests are data retrieval tasks which are slightly easier than the tasks requiring Inserting and Updating operations. Fig. 6(c) evaluates the complexity distribution of requests as determined by the number of database tables involved in each request. The most complex request requires the CWM to access five tables to complete it. Approximately ten requests can be easily resolved by accessing only one table.

4.1.3. Evaluation

We use the execution accuracy across all requests as the primary evaluation metric. The response given by CWM for a request is regarded as correct when it meets two conditions: (1) The CWM must adhere to the business logic of the MES and query or update the records in specific tables using either tools or pure SQL commands. (2) It must consolidate the intermediate results from these operations and deliver a coherent response to the user's request.

The ground truth for requests is developed through semi-automatically process. Initially, we use the CWM to produce both results and intermediate operation steps. These outputs are then manually examined for potential inaccuracies. In instances where errors are identified, additional prompts are consistently provided to the CWM to ensure that it yields the correct responses.

During the evaluation of the CWM and other baselines, we introduce an auxiliary GPT-4o as an adjudicator, determining whether the generated chain of operations, operations commands, and responses closely approximate the ground truth. For incorrect answers, the judge model is responsible for identifying the category and cause of the error. By analyzing the response processes of large models, we identified four types of errors:

- **Information Deviation:** The LLM fails to identify the correct entity information from the user's input. For instance, while the user's request includes the customer name 'PolyU', the officially registered name in the database is 'Hong Kong Polytechnic University'.
- **Wrong CoT:** The LLM either fails to comprehend the user's intention or constructs an incorrect logical chain to address the problem.
- **Wrong Syntax:** There are syntax errors in the generated SQL commands and tool invocation statements.
- **Else:** Other errors that do not fit into the specified categories, such as errors in Mathematical Calculation.

The four types of errors have a "short-circuit" relationship. If the model makes the "Information Deviation" error, any occurrences of "Wrong Syntax" and "Wrong CoT" in its response will not be counted. Beyond evaluating Execution Accuracy and identifying individual errors within each category, we also measure token usage for responding to all requests, as this metric is relevant to cost and significant in practical applications.

4.1.4. Details of implementing CWM

All the LLMs used in CWM, including the query rewriting module, multi-step operations planner, and response generator, are based on GPT-4o by default, with their temperatures set to 0 to reduce the stochasticity during generation. In Section 4.4, ablation studies were conducted to evaluate the selection of the foundational LLM. The agent responsible for the coordination of the provided tools is implemented utilizing the LangChain framework. We use Dify³ to construct the graphical chatting window. The source code of CWM, along with the simulated MES and benchmark requests, is publicly accessible.⁴ A test website is provided for convenient access and experience.⁵

4.2. Two cases presenting how CWM handles user's requests

Before presenting the quantitative evaluation results, this section provides two examples to demonstrate how CWM handles a user's request. Tables 3 and 4 compare CWM to the baseline GPT-4 using a sample request, by analyzing a sample request to query the raw materials consumed for orders in June 2024. Given that the total material usage is calculated by aggregating the individual usage of each product within each order, a mathematical calculation is necessary to analyze the data retrieved from the previous steps. Compared to the previous example, the planned chain of operations includes two 'Thought' steps. During Step 5, the CWM utilizes the chain of thought (CoT) methodology to deconstruct this 'Thought' task further. This procedure is executed by GPT-4o without any prompts from the CWM. Step 6 effectively distills and arranges the retrieved results into the specified format. Table 4 represents the results executed by the GPT-4o based Text2SQL model. The model generates a long SQL command containing three obvious errors marked with purple. The errors result in no results being retrieved from the database. The comparison clearly indicates decomposing complex tasks into sub-tasks can effectively reduce complexity, decrease the probability of model errors, and facilitate the incorporation of prior production guidelines into the LLM.

We specifically designed a highly challenging example to demonstrate CWM's capability in solving complex problem. As shown in Table 5, the case involves rearranging the production plan because the expected delivery time of an order has been advanced. The request involves third typical challenges. First, the description of the order "dress order of Emily" is non-standardized reference, filled with ambiguity and vagueness. In our simulated MES, the standardized order name is "Emily_Dress". CWM is required to identify the standardized entity names and formulate the appropriate retrieval SQL utilizing "Emily_Dress". The second issue is the CWM must interpret the ambiguous reference, "the group with the minimum number of tasks". The

³ dify.ai.

⁴ <https://github.com/yuanzhaolin/CWM>.

⁵ <https://udify.app/chatbot/lczRJdGZX8s7z11>.

Table 3

Example of querying the material usage.

Question:

List the raw materials and quantities used for orders in June 2024? And organize it into the form of raw materials: used quantity.

Query Rewriting: List the raw materials and quantities used for orders in June 2024? And organize it into the form of {raw materials: used quantity}.**Chain of Operations:**

Step 1: Retrieve the order IDs for orders created in June 2024

SQL 'SELECT id FROM orders WHERE created_at BETWEEN '2024-06-01' AND '2024-06-30';'

Step 2: Retrieve the product IDs and quantities for each order from June 2024

SQL 'SELECT order_id, product_id, number FROM order_product WHERE order_id IN (<order_ids>);'

Step 3: Retrieve the work-in-progress (WIP) material IDs and quantities required for each product

Tool 'tool_get_wip_materials, {'product_id': <product_id>}, Get the required work-in-progress materials for a product'

Step 4: Retrieve the original material IDs and quantities required for each WIP material

Tool 'tool_get_original_materials, {'wip_id': <wip_id>}, Get the original materials for a work-in-progress material'

Step 5: Calculate the total quantity of each raw material used by multiplying the product quantity, WIP material quantity, and original material quantity

Thought 'For each product, multiply the number of products by the quantity of each WIP material required, and then multiply by the quantity of each original material required for the WIP material. Sum these values for each raw material to get the total used quantity.'

Step 6: Organize the results into the form of {raw materials: used quantity}

Thought 'Compile the calculated quantities from Step 5 into a dictionary format where keys are raw material names and values are the total used quantities.'**Execution:****Step 1:** SQL, Retrieve the order IDs for orders created in June 2024.

Execute:

SELECT id FROM orders WHERE created_at BETWEEN '2024-06-01' AND '2024-06-30';

Database response:

```

+-----+
| id |
+-----+
| 10 |
| 11 |
| 12 |
+-----+

```

Step 2: SQL, Retrieve the product IDs and product numbers for the orders found in Step 1.

Execute:

SELECT product_id, number FROM order_product WHERE order_id IN (10, 11, 12);

Database response:

```

+-----+-----+-----+
| order_id | product_id | number |
+-----+-----+-----+
| 10 | 30 | 150 |
| 11 | 20 | 120 |
| 12 | 7 | 330 |
+-----+-----+-----+

```

Step 3: Tool, Retrieve the work-in-progress (WIP) material IDs and quantities for the products found in Step 2

Execute:

'tool_get_wip_materials, {'product_id': 30}'

'tool_get_wip_materials, {'product_id': 20}'

'tool_get_wip_materials, {'product_id': 7}'

Results from invoking the tools:

Here are the work-in-progress (WIP) material IDs and quantities required for each product:

Product ID 30: {'Material ID': 2, 'Quantity': 5}, {'Material ID': 10, 'Quantity': 1}

Product ID 20: {'Material ID': 2, 'Quantity': 3}, {'Material ID': 6, 'Quantity': 4}

Product ID 7: {'Material ID': 12, 'Quantity': 5}, {'Material ID': 6, 'Quantity': 2}

(continued on next page)

Table 3 (continued).

Step 4: Tool, Retrieve the original material IDs and quantities required for each WIP material.

Execute:

```
tool_get_original_materials, {'wip_id': 2}
tool_get_original_materials, {'wip_id': 10}
tool_get_original_materials, {'wip_id': 6}
tool_get_original_materials, {'wip_id': 12}
```

Results from invoking the tools:

```
WIP Material ID 2: {'Original Material ID': 1, 'Quantity': 0.4}, {'Original Material ID': 11, 'Quantity': 0.65}, {'Original Material ID': 3, 'Quantity': 0.2}
WIP Material ID 10: {'Original Material ID': 9, 'Quantity': 0.6}, {'Original Material ID': 1, 'Quantity': 0.35}, {'Original Material ID': 13, 'Quantity': 0.8}
WIP Material ID 6: {'Original Material ID': 5, 'Quantity': 0.75}, {'Original Material ID': 1, 'Quantity': 0.4}
WIP Material ID 12: {'Original Material ID': 11, 'Quantity': 0.7}, {'Original Material ID': 1, 'Quantity': 0.5}
```

Step 5: Thought, Calculate the total quantity of each raw material used by multiplying the product quantity, WIP material quantity, and original material quantity.

Output:

To achieve the objective, we need to calculate the total quantity of each original material used based on the orders, products, WIP materials, and original materials. Here's a step-by-step breakdown:

```
1. **Retrieve Order Details:**
  - Orders in June 2024: IDs 10, 11, 12.
  - Order 10: Product ID 30, Quantity 150.
  ...

2. **WIP Materials Required for Each Product:**
  - Product ID 30:
    - WIP Material ID 2, Quantity 5.
    - WIP Material ID 10, Quantity 1.
  - Product ID 20:
  ...

3. **Original Materials Required for Each WIP Material:**
  - WIP Material ID 2:
    - Original Material ID 1, Quantity 0.4.
  ...

4. **Calculate Total Original Materials Used:**
  - **Product ID 30 (150 units):**
    - WIP Material ID 2: 150 * 5 = 750 units
    - Original Material ID 1: 750 * 0.4 = 300
  ...

5. **Sum the Quantities for Each Original Material:**

  - Original Material ID 1: 300 + 52.5 + 144 + 192 + 825 + 264 = 1777.5
  - Original Material ID 11: 487.5 + 234 + 1155 = 1876.5
  ...
```

These are the total quantities of each original material used based on the given orders and material requirements.

Step 6: Thought, Organize the results into the form of raw materials: used quantity.

Output:

To compile the calculated quantities from Step 5 into a dictionary format, we will use the raw material names as keys and the total used quantities as values. Here is the dictionary:

```
'''python
{
    "Original Material ID 1": 1777.5,
    "Original Material ID 11": 1876.5,
    "Original Material ID 3": 222,
    "Original Material ID 9": 90,
    "Original Material ID 13": 120,
    "Original Material ID 5": 855
}
'''
```

third challenge is CWM needs to understand the business guidelines of the factory, available tools (services), and the database schema, in order to autonomously plan a series of interdependent operations to process the request.

CWM decomposed this task into 20 steps, including 15 SQL operations, 2 Tool operations, and 3 Thought operations. Due to the large number of operations, the table omits specific execution logs and results, listing only the types and descriptions of the operations used. This example sufficiently demonstrates CWM's capability to replace the GUI in MES, as it can autonomously integrate external services and

manipulate the database through natural language input, effectively handling complex requests in the manufacturing process.

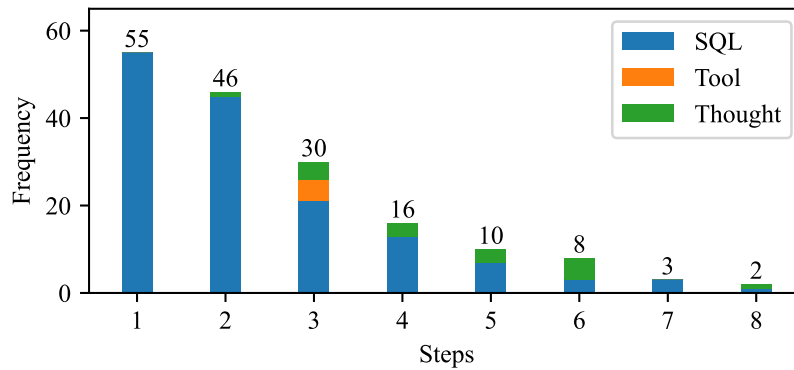
4.3. Main results

This section illustrates the quantitative performance of the CWM in comparison to baseline models as shown in Table 6. As two weak baselines, the original GPT-3.5 and GPT-4.0 are utilized as text2SQL engines to generate a single SQL command for handling the user's request. The execution accuracy of GPT-4o is 60%, significantly lower

Table 4

Example of querying the material usage(without Multi-step Dynamical Operations Generation).

Question:
List the raw materials and quantities used for orders in June 2024? And organize it into the form of raw materials: used quantity.
Query Rewriting:
List the raw materials and quantities used for orders in June 2024? And organize it into the form of {raw materials: used quantity}.
Execution:
SQL, To list the raw materials and quantities used for orders in June 2024, we need to find the orders created in June 2024, then join with order_product to get the products, and finally join with product_material to get the raw materials and their quantities.
Execute:
SELECT m.name AS raw_material, SUM(pm.number * op.number)AS used_quantity FROM orders o JOIN order_product op ON o.id = op.order_id JOIN product_material pm ON op.product_id = pm.product_id JOIN materials m ON pm.material_id = m.id WHERE o.created_at BETWEEN '2024-06-01' AND '2024-06-30' AND m.type = 'origin' GROUP BY m.name;
Results from executing the SQL:
no results found.

**Fig. 7.** Frequency of each operation type invoked in each step.

than the solutions based on the CWM architecture. This indicates that workflow engineering is essential for enabling the foundational LLM to operate an MES effectively. Unavoidably, the use of a complex workflow inevitably leads to increased token usage.

Considering that the CWM is analogous to the integration of a Request re-writer and Multiple-step operations, we have undertaken an individual assessment of each component's contribution. The Request re-writer can efficiently tackle the Information Deviation, contributing about 13% accuracy improvement. The query re-writer effectively addresses Information Deviation, contributing approximately a 9% increase in accuracy. The Multi-step Dynamic Operations Planning and Execution is the most crucial component of the CWM, reducing errors and providing an approximate 18.2% improvement. Handling the request step-by-step ensures alignment with human intent and pre-existing knowledge within a specified context, thereby leading to a reduction in errors related to both the chain of thought (COT) and computational processes.

Fig. 7 shows the total times for executing three different kinds of operations at each step for 55 requests. The process is limited to a maximum of 8 steps. Most requests, accounting for 45, can be resolved within 4 steps. In the initial stages, SQL operations are predominant, as the CWM necessitates data retrieval from the database prior to engaging in subsequent data processing and tool invocation activities. The processes involving thought typically take place in the subsequent stages when requests require mathematical computation. The integration of these three types of operations is nearly sufficient to address all requests within the CWM.

4.4. Ablation study

Within the Multi-step Dynamical Operation framework, the 'Tool' and 'Thought' operations are important in augmenting the precision of CWM's execution. To measure the impact of each operation type, we performed extensive ablation studies to assess CWM performance

when lacking one operation type. The study further examines how the fundamental LLM in CWM affects performance.

According to Table 7, excluding 'Thought' or 'Tool' operations significantly drops CWM execution accuracy. The presence of 'Tool' operations provides a marginally greater contribution compared to 'Thought'. Without thought processes, the model struggles with reasoning and calculations. As most dataset requests are straightforward, removing 'Thought' does not greatly impact performance.

As a greater role in the operations of CWM, the 'Tool' operations are important for handling the requests that require extensive prior knowledge of production, such as task allocation and material allocation. Without the 'Tool', the CWM must generate a series of complex SQL commands to simulate the functions of the missing tools. In this process, commands with incorrect syntax may be produced, and the number of 'Wrong Syntax' will increase.

Finally, we replaced the foundational LLMs in the CWM with other advanced alternatives. The results demonstrate that the intelligence of the foundation model significantly impacts the CWM's performance. The CWM based on GPT-3.5 fails to generate syntactically correct SQL. The accuracy is even lower than that of the GPT-3.5 without the CWM framework. This suggests that as the cognitive abilities of large models develop and further align with human intentions, the capabilities of CWM can advance in parallel.

5. Discussion

This study preliminarily implements a chatting interface of MES as a demonstration. However, it is still far from being applicable in real factory settings and replacing the traditional GUI interfaces. Three critical issues need to be discussed and addressed for future improvement.

1. Balance between Accuracy, Generality, and Token usage

Execution accuracy can be improved by designing more precise workflows or incorporating additional few-shot examples in prompts.

Table 5

Example of reorganize production plan.

Question: The dress order of Emily requires delivery by April 5, 2024. Please re-arrange the production plan and assign tasks to the task group with the fewest tasks currently. Provide a new production plan.
Query Rewriting: The <orders.order_name = 'Emily_Dress'> order requires delivery by April 5, 2024. Please re-arrange the production plan and assign tasks to the task group with the fewest tasks currently. Provide a new production plan.
Chain of Operations:
<p>Step 1: Retrieve the order ID for the order with the name "Emily_Dress"</p> <p>SQL 'SELECT id FROM orders WHERE order_name = 'Emily_Dress';'</p> <p>Step2: Retrieve the product details related to the order\n["tool_find_order_details, {'order_id': 3}, Find the product details of an order"]</p> <p>Tool 'tool_find_order_details, {'order_id': 3}, Find the product details of an order'</p> <p>Step3: Retrieve the current cutting task's working group ID for the order</p> <p>SQL 'SELECT working_group_id FROM cutting_tasks WHERE order_product_id IN (SELECT id FROM order_product WHERE order_id = 3);'</p> <p>Step4: Retrieve the current sewing task's working group ID for the order</p> <p>SQL 'SELECT working_group_id FROM sewing_tasks WHERE order_product_id IN (SELECT id FROM order_product WHERE order_id = 3);'</p> <p>Step5: Get the current list of all working groups and their task counts\ntool_get_busy_workers, {}, Get the current list of all working groups and their task counts</p> <p>Tool 'tool_get_busy_workers, {}, Get the current list of all working groups and their task counts'</p> <p>Step6: Compare the task counts of all cutting groups obtained in Step5 and identify the group with the fewest tasks.</p> <p>Thought 'Compare the task counts of all cutting groups obtained in Step5 and identify the group with the fewest tasks.'</p> <p>Step7: Compare the task counts of all sewing groups obtained in Step5 and identify the group with the fewest tasks.</p> <p>Thought 'Compare the task counts of all sewing groups obtained in Step5 and identify the group with the fewest tasks.'</p> <p>Step8: Retrieve the produced WIP ID from the cutting tasks</p> <p>SQL 'SELECT produced_wip_id FROM cutting_tasks WHERE order_product_id IN (SELECT id FROM order_product WHERE order_id = 3);'</p> <p>Step9: Update the status of the original cutting tasks to 0 (stopped)</p> <p>SQL 'UPDATE cutting_tasks SET status = 0 WHERE order_product_id IN (SELECT id FROM order_product WHERE order_id = 3);'</p> <p>Step10: Create new cutting tasks with the new working group ID and set the new deadline to April 5, 2024</p> <p>SQL 'INSERT INTO cutting_tasks (start_time, end_time, working_group_id, order_product_id, produced_wip_id, planned_number, status) VALUES (CURRENT_TIMESTAMP, '2024-04-05 23:59:59', 6, 21, 8, 140, 1);'</p> <p>Step11: Create new cutting tasks with the new working group ID and set the new deadline to April 5, 2024</p> <p>SQL 'INSERT INTO cutting_tasks (start_time, end_time, working_group_id, order_product_id, produced_wip_id, planned_number, status) VALUES (CURRENT_TIMESTAMP, '2024-04-05 23:59:59', 6, 29, 10, 150, 1);'</p> <p>Step12: Create new cutting tasks with the new working group ID and set the new deadline to April 5, 2024</p> <p>SQL 'INSERT INTO cutting_tasks (start_time, end_time, working_group_id, order_product_id, produced_wip_id, planned_number, status) VALUES (CURRENT_TIMESTAMP, '2024-04-05 23:59:59', 6, 13, 12, 130, 1);'</p> <p>Step13: Create new cutting tasks with the new working group ID and set the new deadline to April 5, 2024</p> <p>SQL 'INSERT INTO cutting_tasks (start_time, end_time, working_group_id, order_product_id, produced_wip_id, planned_number, status) VALUES (CURRENT_TIMESTAMP, '2024-04-05 23:59:59', 6, 5, 6, 120, 1);'</p> <p>Step14: Update the status of the original sewing tasks to 0 (stopped)</p> <p>SQL 'UPDATE sewing_tasks SET status = 0 WHERE order_product_id IN (SELECT id FROM order_product WHERE order_id = 3);'</p> <p>Step15: Create new sewing tasks with the new working group ID and set the new deadline to April 5, 2024</p> <p>SQL 'INSERT INTO sewing_tasks (start_time, end_time, working_group_id, order_product_id, planned_number, status) VALUES (CURRENT_TIMESTAMP, '2024-04-05 23:59:59', 10, 21, 140, 1);'</p> <p>Step16: Create new sewing tasks with the new working group ID and set the new deadline to April 5, 2024</p> <p>SQL 'INSERT INTO sewing_tasks (start_time, end_time, working_group_id, order_product_id, planned_number, status) VALUES (CURRENT_TIMESTAMP, '2024-04-05 23:59:59', 10, 29, 150, 1);'</p> <p>Step17: Create new sewing tasks with the new working group ID and set the new deadline to April 5, 2024</p> <p>SQL 'INSERT INTO sewing_tasks (start_time, end_time, working_group_id, order_product_id, planned_number, status) VALUES (CURRENT_TIMESTAMP, '2024-04-05 23:59:59', 10, 13, 130, 1);'</p> <p>Step18: Create new sewing tasks with the new working group ID and set the new deadline to April 5, 2024</p> <p>SQL 'INSERT INTO sewing_tasks (start_time, end_time, working_group_id, order_product_id, planned_number, status) VALUES (CURRENT_TIMESTAMP, '2024-04-05 23:59:59', 10, 5, 120, 1);'</p> <p>Step19: Check material inventory for required materials</p> <p>SQL 'SELECT pm.material_id, pm.number, wm.left_number FROM product_material pm JOIN warehouse_material wm ON pm.material_id = wm.material_id WHERE pm.product_id IN (SELECT product_id FROM order_product WHERE order_id = 3);'</p> <p>Step20: Compile the new cutting and sewing group information, new task schedule, material inventory status, and expected completion date into a report.</p> <p>Thought 'Compile the new cutting and sewing group information, new task schedule, material inventory status, and expected completion date into a report.'</p>

Table 6
Main results.

Models	Execution accuracy	Tokens	Causes of errors			
			Information deviation	Wrong CoT	Wrong syntax	Else
GPT-3.5	25.5%	319 866	3	28	9	1
GPT-4o	60%	311 176	3	14	0	5
CWM (w/o re-writer)	70.9%	439 947	3	10	1	2
CWM (w/o Multiple- Step operations)	61.8%	401 771	0	18	3	0
CWM	80%	436 057	0	7	1	3

Table 7
Results in ablation study.

Foundation model	Thought	Tool	Execution accuracy	Causes of errors		
				Wrong CoT	Wrong syntax	Else
GPT-4o	✓	✓	80%	7	1	3
GPT-4o	✓	×	65%	15	3	1
GPT-4o	×	✓	67%	16	0	2
GPT-3.5	✓	✓	16%	26	18	2

However, this approach is akin to patching, which requires significant manpower and cannot be easily generalized to other MES systems. Additionally, complex workflows or lengthy prompts lead to high LLM token consumption. The CWM is expected to evolve into a more extensible and dynamic framework that can understand database schema and production guidelines at a lower cost.

2. Who should be responsible for incorrect operations?

A typical MES requires a confirmation step before performing any data adjustment operations. The operator who confirms the operation is responsible for any unforeseen consequences that may arise. However, the CWM is a virtual worker who autonomously handles task planning and execution. A critical issue in this process is determining the responsibility when the LLM performs unreasonable operations, even if the user provided correct instructions. A viable solution is to incorporate a confirmation function in the chat window, where all details of the operations are clearly and thoroughly presented. Furthermore, it is meaningful to introduce counterfactual inference [31] to forecast the possible outcome under the hypothetical input requests. Upon the user's affirmative "YES" during the confirmation phase, responsibility for the action and any outcomes is assumed by the user.

3. Speaks little but is willing to express a lot.

In the traditional MES user interface, users must provide complete information in the submitted form, as input fields are typically set as mandatory by developers. In CWM, the free interaction mode can lead users to become lax and inattentive in providing sufficient information. The chatting agent has to keep inquiring about more information to complete the specified request, otherwise some vital information may be omitted during subsequent operation executions. If we strictly limit the question options and required fields in CWM, the characteristic of free chatting is lost, and CWM will degenerate into a typical MES interaction model. This dilemma inspires us to consider an important question, what is the best chatting interface for the CWM that is universally applicable, human-centered, and suitable for manufacturing environments? As a preliminary feasibility study, the research utilizes Dify's basic chatting page as the interface. However, there are many stakeholders for the CWM, including customers, production managers and workers. The design of human-AI interfaces must take into their tasks, identities, and personas [14]. Designing next-generation human-AI interfaces for critical applications like manufacturing is crucial for the broader utilization of LLMs.

6. Conclusion

This study proposes an LLM agent system, named Chat with MES (CWM), an interface for the operators in garment factories to manipulate a manufacturing execution system based on natural language. We introduce two key technical innovations: the Request Re-writer and Multi-step Dynamic Operations Planning and Execution. To eliminate entity ambiguity in the user's request, which may lead to incorrect query conditions, the Query Re-writer automatically aligns ambiguous entities in the users' input with the correct entities within the Manufacturing Execution System (MES), such as the names of customers, products, and materials. To handle complicated manipulation requests on MES, the proposed Multi-step Dynamic Operations Planning and Execution combines the Text2SQL, LLM-Agents and Chain of Thoughts to plan a chain of basic operations and dynamically execute the operations step-by-step. By evaluating the CWM on a simulated garment MES with 55 manually designed questions, we find that the CWM achieves a high execution success rate of 80%, compared to GPT-4's 60%. The query re-writer and Multi-step Dynamic Operations Planning and Execution techniques separately contribute improvements of approximately 10% and 18%, respectively.

In future study, we identify key areas for improvement, such as balancing execution accuracy and token usage, assigning responsibility for incorrect operations, and developing an optimal chatting interface for broad and human-centered applicability in manufacturing.

CRedit authorship contribution statement

Zhaolin Yuan: Writing – original draft, Software, Conceptualization. **Ming Li:** Writing – review & editing, Project administration, Funding acquisition, Conceptualization. **Chang Liu:** Software. **Fangyuan Han:** Software. **Haolun Huang:** Software. **Hong-Ning Dai:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is partly supported by the Innovation and Technology Fund, China (No. PRP/015/24TI) and the Innovation and Technology Commission of the HKSAR Government through the InnoHK initiative.

References

- [1] Saenz de Ugarte B, Artiba A, Pellerin R. Manufacturing execution system – a literature review. *Prod Plan Control* 2009;20(6):525–39. <http://dx.doi.org/10.1080/09537280902938613>, [arXiv:https://doi.org/10.1080/09537280902938613](https://doi.org/10.1080/09537280902938613).

- [2] Chen J, Wu H, Li Y, Wu X, Zhao Y, Shi Y, Tang S, Shi Y, Chen H. HANA-MES: A HMI-awared manufacturing execution system. In: 2021 IEEE 23rd int conf on high performance computing & communications. 2021, p. 1863–9. <http://dx.doi.org/10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00276>.
- [3] Martínez WL. Graphical user interfaces. Wiley Interdiscip Rev: Comput Stat 2011;3(2):119–33.
- [4] Shaheen BW, Németh I. Integration of maintenance management system functions with industry 4.0 technologies and features—A review. Processes 2022;10(11). <http://dx.doi.org/10.3390/pr10112173>, URL <https://www.mdpi.com/2227-9717/10/11/2173>.
- [5] Li Z, Li G, Li Z. Application research of MES in intelligent manufacturing training factory. In: International conference on adaptive and intelligent systems. Springer; 2022, p. 118–27.
- [6] Rokis K, Kirikova M. Challenges of low-code/no-code software development: A literature review. In: International conference on business informatics research. Springer; 2022, p. 3–17.
- [7] Cabot J. Positioning of the low-code movement within the field of model-driven engineering. In: Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings. 2020, p. 1–3.
- [8] Siderska J. Robotic process automation—a driver of digital transformation? Eng Manag Prod Serv 2020;12(2):21–31.
- [9] Yao S, Chen H, Yang J, Narasimhan K. Webshop: Towards scalable real-world web interaction with grounded language agents. Adv Neural Inf Process Syst 2022;35:20744–57.
- [10] Deng X, Gu Y, Zheng B, Chen S, Stevens S, Wang B, Sun H, Su Y. Mind2web: Towards a generalist agent for the web. Adv Neural Inf Process Syst 2024;36.
- [11] Yadav A, Patel A, Shah M. A comprehensive review on resolving ambiguities in natural language processing. AI Open 2021;2:85–92.
- [12] Shojaeinasab A, Charter T, Jalayer M, Khadivi M, Ogunfowora O, Raiyani N, Yaghoubi M, Najjaran H. Intelligent manufacturing execution systems: A systematic review. J Manuf Syst 2022;62:503–22. <http://dx.doi.org/10.1016/j.jmsy.2022.01.004>, URL <https://www.sciencedirect.com/science/article/pii/S0278612522000048>.
- [13] Dalibor M, Heithoff M, Michael J, Netz L, Pfeiffer J, Rumpe B, Varga S, Wortmann A. Generating customized low-code development platforms for digital twins. J Comput Lang 2022;70:101117. <http://dx.doi.org/10.1016/j.cola.2022.101117>, URL <https://www.sciencedirect.com/science/article/pii/S2590118422000235>.
- [14] Holzinger A, Kargl M, Kipperer B, Regitnig P, Plass M, Müller H. Personas for artificial intelligence (AI) an open source toolbox. IEEE Access 2022;10:23732–47. <http://dx.doi.org/10.1109/ACCESS.2022.3154776>.
- [15] Jain A, Paliwal S, Sharma M, Vig L, Shroff G. SmartFlow: Robotic process automation using LLMs. 2024, arXiv preprint [arXiv:2405.12842](https://arxiv.org/abs/2405.12842).
- [16] Sharma M, Angmo R. Web based automation testing and tools. Int J Comput Sci Inf Technol 2014;5(1):908–12.
- [17] Mantravadi S, Jansson AD, Möller C. User-friendly mes interfaces: Recommendations for an ai-based chatbot assistance in industry 4.0 shop floors. In: Asian conference on intelligent information and database systems. Springer; 2020, p. 189–201.
- [18] Colabianchi S, Tedeschi A, Costantino F. Human-technology integration with industrial conversational agents: A conceptual architecture and a taxonomy for manufacturing. J Ind Inf Integr 2023;35:100510.
- [19] Li H, Zhang J, Liu H, Fan J, Zhang X, Zhu J, Wei R, Pan H, Li C, Chen H. Codes: Towards building open-source language models for text-to-sql. Proc ACM Manag Data 2024;2(3):1–28.
- [20] Xue S, Jiang C, Shi W, Cheng F, Chen K, Yang H, Zhang Z, He J, Zhang H, Wei G, Zhao W, Zhou F, Qi D, Yi H, Liu S, Chen F. DB-GPT: Empowering database interactions with private large language models. 2023, arXiv preprint [arXiv:2312.17449](https://arxiv.org/abs/2312.17449). URL <https://arxiv.org/abs/2312.17449>.
- [21] Li J, Hui B, Qu G, Yang J, Li B, Li B, Wang B, Qin B, Geng R, Huo N, et al. Can llm already serve as a database interface? Aa big bench for large-scale database grounded text-to-sqls. Adv Neural Inf Process Syst 2024;36.
- [22] Lian J, Liu X, Shao Y, Dong Y, Wang M, Wei Z, Wan T, Dong M, Yan H. ChatBI: Towards natural language to complex business intelligence SQL. 2024, arXiv preprint [arXiv:2405.00527](https://arxiv.org/abs/2405.00527).
- [23] Hu C, Fu J, Du C, Luo S, Zhao J, Zhao H. Chatdb: Augmenting llms with databases as their symbolic memory. 2023, arXiv preprint [arXiv:2306.03901](https://arxiv.org/abs/2306.03901).
- [24] Pourreza M, Rafiei D. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. Adv Neural Inf Process Syst 2024;36.
- [25] Sun Y, Zhang Q, Bao J, Lu Y, Liu S. Empowering digital twins with large language models for global temporal feature learning. J Manuf Syst 2024;74:83–99.
- [26] Xia L, Li C, Zhang C, Liu S, Zheng P. Leveraging error-assisted fine-tuning large language models for manufacturing excellence. Robot Comput-Integr Manuf 2024;88:102728. <http://dx.doi.org/10.1016/j.rcim.2024.102728>, URL <https://www.sciencedirect.com/science/article/pii/S0736584524000140>.
- [27] Li Y, Starly B. Building a knowledge graph to enrich ChatGPT responses in manufacturing service discovery. J Ind Inf Integr 2024;40:100612.
- [28] Xu S, Wei Y, Zheng P, Zhang J, Yu C. LLM enabled generative collaborative design in a mixed reality environment. J Manuf Syst 2024;74:703–15.
- [29] Wu T, Li J, Bao J, Liu Q. ProcessCarbonAgent: A large language models-empowered autonomous agent for decision-making in manufacturing carbon emission management. J Manuf Syst 2024;76:429–42.
- [30] Xie T, Li Q, Zhang Y, Liu Z, Wang H. Self-improving for zero-shot named entity recognition with large language models. 2023, arXiv preprint [arXiv:2311.08921](https://arxiv.org/abs/2311.08921).
- [31] Del Ser J, Barredo-Arrieta A, Díaz-Rodríguez N, Herrera F, Saranti A, Holzinger A. On generating trustworthy counterfactual explanations. Inform Sci 2024;655:119898.