计 算 机 研 究 与 发 展
Journal of Computer Research and Development

https://crad.ict.ac.cn
63(5): 1097−1116, 2026

本期推荐

基于 LLM 的可解释嵌入式系统需求规约方法

王春晖[1] 赵佳琪[1] 陈小红[2] 金 芝[3,4]

[1](内蒙古师范大学计算机科学技术学院 呼和浩特 010022)

[2](上海市高可信计算重点实验室(华东师范大学) 上海 200062)

[3](高可信软件技术教育部重点实验室(北京大学) 北京 100871)

[4](北京大学计算机学院 北京 100871)

(zhijin@pku.edu.cn)

LLM-Based Interpretable Embedded System Requirements Specification

Wang Chunhui[1], Zhao Jiaqi[1], Chen Xiaohong[2], and Jin Zhi[3,4]

[1](*College of Computer Science and Technology*, *Inner Mongolia Normal University*, *Hohhot* 010022)

[2](*Shanghai Key Laboratory of Trustworthy Computing* (*East China Normal University*), *Shanghai* 200062)

[3](*Key Lab of High Confidence Software Technologies* (*Peking University*), *Ministry of Education*, *Beijing* 100871)

[4](*School of Computer Science*, *Peking University*, *Beijing* 100871)

Abstract The embedded system requirements specification serves as the cornerstone of system development, playing a crucial role in transforming high-level user intentions into clear and executable software requirements. Currently, this process remains highly dependent on manual derivation and composition by domain experts, with inherent limitations such as low efficiency and insufficient interpretability. Although automated requirements generation is regarded as a potential solution, its development still faces two core challenges: how to effectively embed domain-specific device knowledge, and how to strictly adhere to the standards of intent satisfiability and traceability to construct an interpretable requirements system. To address these issues, we propose an interpretable requirements specification method based on large language models (LLMs). By constructing a prompt chain that integrates domain knowledge and requirements engineering, our method gradually complements and refines user intentions into device, system, and software requirements level by level under human-computer interaction collaboration, ensuring the satisfiability of requirements. Additionally, it introduces the problem diagram and scenario diagram model to guarantee the traceability of requirements. Experiments on four embedded system cases show that compared with manual methods, our method reduces time cost by approximately 73% and improves the quality of requirements specification documents by 82%. Moreover, its interpretability increases by about 15% and 21% compared with both manual methods and single-step LLM methods, providing an efficient and reliable new approach for requirements specification of complex embedded systems.

Key words embedded systems; requirements specification; requirements interpretability; LLM (large language model); requirements completion

摘 要 嵌入式系统需求规约是系统开发的基石,承担着将高层用户意图转化为明确且可执行的软件需

收稿日期: 2025-12-15; **修回日期**: 2026-03-11

基金项目: 国家自然科学基金项目(62192731, 62272166, 62192730, 62162051); 内蒙古自然科学基金项目(2025MS06011)

This work was supported by the National Natural Science Foundation of China (62192731, 62272166, 62192730, 62162051) and the Inner Mongolia Natural Science Foundation (2025MS06011).

通信作者: 王春晖(ciecwch@imnu.edu.cn)

求的重要职能。当前,该过程仍高度依赖领域专家的人工推导与撰写,存在效率低、可解释性不足等固有局限性。尽管自动化需求生成被视为潜在的解决方案,其发展仍面临双重核心挑战:如何有效嵌入领域特定的设备知识、如何严格遵循意图可满足和可追溯性标准,构建具备可解释性的需求体系。为解决上述问题,提出了一种基于大语言模型(LLM)的可解释需求规约方法。该方法通过构建融合领域知识与需求工程的提示链,在人机交互协作下,逐层将用户意图补全并细化为设备、系统及软件需求,确保需求可满足,并引入问题图-情景图模型以保障需求的可追溯。在4个嵌入式系统案例上的实验表明,所提方法较人工方法节约73%时间成本,需求规约文档质量提升82%,而可解释性相较人工方法和单步LLM方法分别提升约15%和21%,为复杂嵌入式系统提供了一种高效且可信的需求规约新途径。

关键词 嵌入式系统;需求规约;需求可解释性;大语言模型;需求补全

中图法分类号 TP311.13; TP309

DOI: 10.7544/issn1000-1239.202550898 **CSTR:** 32373.14.issn1000-1239.202550898

嵌入式系统已广泛应用于航空航天、轨道交通、工业控制、自动驾驶等关键领域,其需求规约是保障系统质量的核心基础[1]。当前,嵌入式系统需求规约的核心流程依赖人工梳理用户意图,并结合设备特性、环境约束等领域知识,将抽象意图逐步精化为涵盖系统及软件层面的需求。这一过程不仅需精确界定系统功能、性能指标与约束条件,更需满足可解释性这一关键特性[2]。其核心包含2个维度:一是“意图—需求”的可满足性,即所有设备、系统及软件需求均需严格对齐并满足用户原始意图,且意图到各层级需求的精化逻辑清晰可阐释;二是“需求—意图”的可追溯性,即每项设备、系统及软件需求均明确追踪至其对应的用户意图,确保需求的来源可查、依据可信。具体而言,嵌入式系统需求规约需明确“需求如何逐层落地”以及“为何需要该需求”,即具有从上至下的可满足以及至下而上的可追溯,为后续开发、验证及维护提供可信依据。

然而,达成嵌入式系统需求规约文档的可解释面临2个方面的挑战:一是需求对意图的可满足性难保证,用户意图常以模糊化自然语言表述,而转化为硬件接口参数、实时性阈值等技术指标时涉及跨学科知识,可能因为对约束的关联度关注不够,导致最终需求集与原始意图存在“语义偏差”;二是随着系统规模和复杂性的增大,需求可追溯性难保障,需求数量从数十项增至数千项,人工逐条维护“需求—意图”追溯逻辑及跨需求依赖关系的成本增长,易出现底层需求与高层用户意图之间的追溯关系中断或不同模块需求对同一意图的解读冲突等问题,导致后续开发或维护难以理解需求的来源和可信性。

传统人工方法受限于嵌入式软件的复杂性,高效开发具有良好可解释的嵌入式软件需求文档存在困难。为此,本文设计了一种基于大语言模型(LLM)的可解释嵌入式系统需求规约方法。主要贡献包括3个方面:

1)提出知识引导LLM的需求规约框架,通过分步骤将用户意图精化为系统及软件层面的显性化细化链路,解决需求遗漏与语义偏差问题;

2)提出人机协同的需求规约方法,结合LLM自动化生成与专家交互式校验,动态维护意图和需求的关联关系,降低人工维护成本;

3)提出三层级嵌入式系统需求结构化组织模型,按“意图(任务目标)→系统需求(与环境交互)→软件需求(与设备交互)”三级架构组织需求,通过问题图建模刻画跨层级依赖关系,确保需求规约过程可解释性。

1 相关工作

1.1 需求补全与文档生成

在需求工程领域,研究者围绕需求信息的自动化补全与需求文档生成展开了广泛探索,现有研究主要分为基于数据驱动的方法与基于知识引导的方法两大类。

基于数据驱动的方法以历史需求数据、对话记录或功能描述等非结构化文本为输入,通过深度学习模型(如LLM、生成对抗网络等)学习数据中的模式,进而自动生成或补全需求信息,其核心优势在于无需人工定义复杂规则,可直接从海量数据中提取需求生成的隐性规律。例如,Voria等人[3]利用自然语言处理技术与LLM,从利益相关者会议对话中自动提取需求要素(包括功能目标、约束条件)并生成符合工程规范的系统需求,实验表明其生成的需求

在正确性、完整性及可操作性上表现良好，显著减少了人工提取的耗时与误差。Koscinski 等人[4]采用关系生成对抗网络从安全需求相关的历史数据（漏洞报告、安全规范文档）中学习安全需求的语义关联模式，合成符合专家要求的安全需求规格，通过法院案件管理系统（CCMS）的真实案例验证，表明生成的安全需求在覆盖度与准确性上均达到行业可用水平。Lian 等人[5]基于 GPT-2 模型构建，通过分析需求描述的前缀文本，结合指针网络捕捉领域术语的上下文依赖关系，实时推荐或补全后续需求内容，实验显示该方法有效提升了需求编写的效率与术语一致性。

此外，LLM 凭借其强大的自然语言理解与生成能力，成为数据驱动方法的核心技术载体。例如，Almonte 等人[6]利用 LLM 从功能需求中推导 ISO/IEC 25010:2023 标准的质量属性（如性能、可靠性），生成高相关性的非功能需求，实验生成的 1 593 条 NFRs 与行业专家评估高度一致。Habib 等人[7]通过微调 LLM 开发 ReqBrain 工具，生成符合 ISO 29148 标准的需求并支持分类，通过人工评估验证了生成需求的真实性。Lowin 等人[8]则借助 ChatGPT、BARD 等 LLM，从智能设备的产品概念目录中生成定制化需求规格，辅助解决开发中的复杂约束（如用户需求与技术限制的平衡）问题。

基于知识引导的方法以领域本体、行业标准或专家规则等结构化知识为基础，通过将知识注入生成过程，确保需求符合领域规范、语法约束及标准化要求，其核心优势在于可显式控制生成需求的质量（如术语准确性、文档合规性），弥补了数据驱动方法可能因数据偏差导致的生成缺陷。例如，Zhao 等人[9]通过构建领域本体模型（软件需求术语库、语法规则库等），将本体知识注入预训练模型 UniLM 进行微调，并结合复用机制确保生成需求包含关键领域术语，同时设计语法约束解码策略，缩小生成需求与参考规格的语义/语法差距。Elliott 等人[10]通过整合 IEEE 830-1998 标准与 SWEBOK 推荐的需求工程数据元素（如功能需求、外部接口），构建本体框架指导需求文档生成，该方法支持诸如医疗软件、金融系统等跨领域定制，并通过 7 个用例验证，3 个典型案例的需求文档生成结果符合标准要求，有效减少了手工编写的错误率。Wang 等人[11]关注嵌入式软件需求文档的生成，提出借助设备知识补全系统需求与软件需求，并通过案例研究表明其在需求一致性保持的价值。

本文提出一种融合需求工程知识与应用领域知识的需求规约方法，通过构建知识提示链引导 LLM 从意图出发，理解约束和规范要求，补全系统需求、软件需求信息，确保需求信息的可解释与可追踪。该方法不受对结构化知识完备性的依赖，同时规避数据驱动方法因数据偏差导致的生成缺陷。

1.2 需求规约与提示工程

需求规约作为连接用户意图与系统实现的桥梁，其规范化方法经历了从人工主导到自动化辅助的演进。IEEE Std 830-1998[12]标准定义了可行性、完整性等核心质量属性。为了确保规约质量，相关研究围绕模板化（如 EARS 结构化模板[13]）、形式化（如时序逻辑[14-15]）及自动化辅助（基于传统自然语言处理的分类与冲突检测工具[16]）开展。这些方法通过规范自然语言表达与结构逻辑，减少了需求歧义与遗漏，但由于嵌入式系统的硬件-软件耦合性、多领域知识交叉特性，仍存在自然语言模糊性与人工依赖成本高的挑战。例如，Bertram 等人[17]研究利用自然语言处理构建汽车需求的一致性检查工具链，Chen 等人[18]研究通过 LLM 改进需求完整性的自动化辅助，Krishna 等人[19]则设计了评估的质量属性列表，并借助 LLM 实施评估。

在自动化辅助需求规约过程中，提示工程通过优化人机交互范式，提高 LLM 对内容的理解质量以及输出结果质量。有研究者采用零样本提示技术，无需标注数据，用于生成需求文档[20]和功能规格说明[18]。也有研究者采用检索增强生成技术与对话式交互方法，生成产品需求文档[21]，以提升内容准确性和需求匹配度。还有研究者借助少样本学习技术，通过少量示例实现任务适配，生成软件规格说明[19]。

提示链（prompt chaining）通过构建“提示序列-输出依赖”的工作流，实现复杂需求任务的分步骤自动化，其核心技术要素为上下文维护与工作流编排。传统提示链侧重“任务分解—指令传递”，例如，Shakthi 等人[22]采用“少样本示例+思维链推荐”的方法，从卫星功能需求文档自动生成结构化测试用例，将卫星领域测试用例模板作为上下文，通过提取功能需求，将需求转化为测试条件，再生成可执行测试脚本。

借助 LLM 与自动化技术（如提示工程中的零样本/少样本学习、检索增强生成及提示链等方法），在需求规约文档的自动化生成（如需求文档、功能规格说明）及质量保障（如一致性检查、完整性提升、质量属性评估）方面已展现出显著效果。然而，现有方法存在用户实时反馈机制融入不足以及在不同层需求可解释方面关注不够等问题，导致下层需求对上层需求支撑和上层需求对下层需求的覆盖不够。本

文面向从意图到需求文档生成这一复杂的任务，设计知识提示链方法，通过任务拆解为多步骤流程，并针对每个步骤注入相关的知识，引导 LLM 生成细化逻辑清晰可阐释的需求规约文档。

2 方法框架

考虑嵌入式系统从用户意图到系统需求，再到软件需求的可解释生成，我们借助 LLM 的推理与解释能力，通过注入相关领域和需求工程知识，分步骤引导 LLM 从理解意图出发逐步推荐需求细节，通过与需求工程师交互，补全和生成需求文档。图 1 展示了嵌入式需求规约的过程框架。该框架以 LLM 为核心，通过“输入→知识注入→反馈→确认→输出”的交互闭环，分 5 步实现需求文档构建：设备选型、系统需求生成、软件需求生成、需求建模及需求文档生成。各步骤以意图、设备、需求等中间制品为输入，注入已有设备库、需求描述模板(系统/软件/文档)及建模知识，由 LLM 调用对应引导词模板($prompt_1$~$prompt_5$)生成反馈信息。反馈经需求工程师确认后，作为输出传递至下一阶段，未通过确认则返回重新生成，最终形成完整需求规约。本节介绍需求规约文档表示、各层需求描述模板、问题图与情景图以及设备知识库的表示，引导模板和过程在第 3 节中阐述。

2.1 嵌入式系统需求规约文档

嵌入式系统规约文档包括意图、设备、系统需求和软件需求 4 个核心维度。意图维度作为顶层设计，明确系统开发的目标，回答“为什么开发”和“系统应实现什么”。设备维度说明使用何种硬件设备与外界环境进行交互以达成任务意图的能力，包括具体的硬件型号、接口协议、环境适应性等物理约束，构成需求实现的硬件支撑。系统需求维度定义如何通过硬件设备与外界环境的交互来满足意图目标，包含功能逻辑、性能指标与安全合规要求等；软件需求维度则通过控制器与设备的交互实现系统能力，具体明确任务划分及算法逻辑。各维度之间存在系统需求分解意图、软件需求实现系统需求，系统需求、软件需求依赖设备的关系。

各维度间的逻辑关联可通过可视化需求建模方法显性化表达。问题图能够表达待开发软件及其交互环境实体的直接或间接交互，以及与交互对象之间的接口，形成待开发软件的上下文[23]。将需求视为待开发软件要解决的问题，从意图出发，通过问题分解，明确需求与上下文实体与设备的交互关系(共享的控制信号、指令、数据等现象)。行为层的动态交互过程则可通过情景图表达[23]，即描述期望交互与行为交互的过程。期望交互在系统需求层面，关注系统与上下文实体通过共享数据、控制信号或指令实现功能目标和性能指标的交互逻辑；行为交互关注控制器与硬件的交互，描述软件模块通过接口协议驱动硬件设备执行具体操作的动态过程。通过问题图与情景图的协同建模，需求文档既能清晰呈现各维度的静态结构关系，又能动态展示需求实现的行为逻辑，为嵌入式系统开发提供全面且可验证的需求框架。

综上，嵌入式系统需求文档包括意图、设备需求、系统需求、软件需求 4 个维度的需求信息，也包括各维度逻辑关联的问题图和情景图。

2.2 需求描述模板

需求描述模板是为需求描述者提供的标准化自

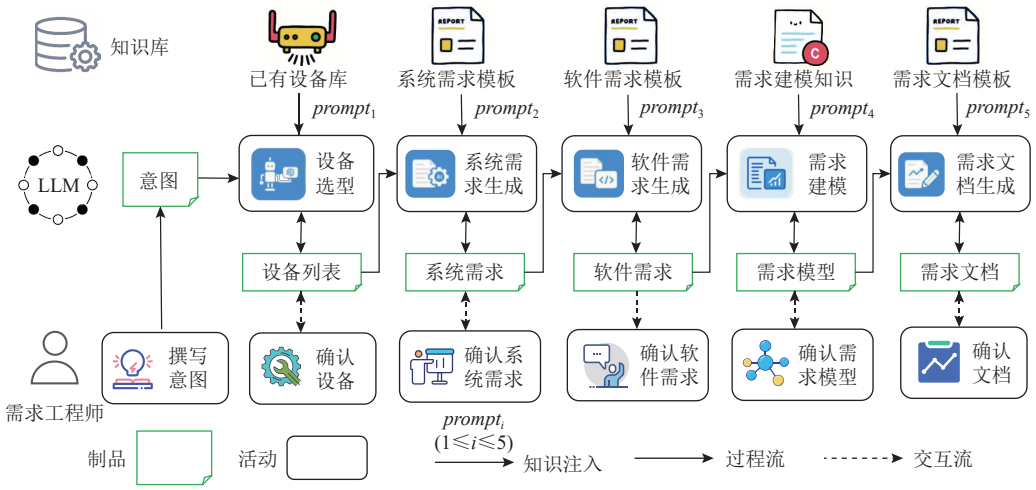


Fig. 1 Interpretable requirements specification framework for embedded systems based on LLM

图 1 基于 LLM 的嵌入式系统可解释需求规约框架

然语言句式，旨在规范需求信息的表达并确保其完整性。本文针对嵌入式需求规约内容，提供任务意图描述模板、系统需求描述模板及软件需求描述模板。基于各类模板的核心含义，分别为其定义标准化的描述句式（定义1~定义3）。

定义 1. 任务意图描述模板。规范系统核心目标及价值导向，明确关联设备资源约束与系统层级目标的追溯关系。句式：<系统>应通过<系统能力>实现/确保/支持/达成<核心目标>/达成<业务价值>。

例如："室内温度异常监测系统"应通过实时温度监测与自动报警能力实现室内温度异常（≥38℃）的识别与预警，达成降低火灾风险、保障人员安全的目的。

定义 2. 系统需求描述模板。承接任务意图，明确系统层功能与非功能目标，需绑定设备资源的硬件参数，体现"设备约束系统"的关联逻辑。标准句式：<系统>需集成/具备/通过 <设备资源>(<硬件参数>)，实现 <任务目标>，[满足<性能指标|环境要求>]。包括以下3种能力描述的句式。

①感知能力：系统需通过<传感器>实时监测<物理量|环境参数>，[采样周期设置<设备约束阈值>|误差范围<硬件指标>]。

例如：系统需通过温度传感器（单总线接口，测量范围−10~50℃）实时监测室内环境温度，采样周期设置10秒/次，误差范围为−0.5℃~+0.5℃。

②控制能力：系统需通过<执行器|驱动模块>对[控制实体] 实现 [控制动作]，[响应时间<设备模块上限>|<软件算法延迟>]。

例如：系统需通过蜂鸣器与LED指示灯对异常温度状态实现声光报警（蜂鸣器1 kHz持续鸣响，红色LED 1Hz闪烁），响应时间≤1 s。

③计算与通信：系统需支持<通信协议|接口类型>，在<延迟|带宽>约束下完成<数据|指令>传输。

定义 3. 软件需求描述模板。细化模块级实现逻辑，需关联设备状态触发条件与系统需求指标。标准句式：软件需 [实现|遵循|保障]<功能|约束>，通过<算法|协议|逻辑>满足<具体条件>；具体描述功能及非功能，可采用如下句式。

①功能需求：主要包括事件驱动型和时序控制型2类句式。

事件驱动型：WHEN<事件>(<设备状态触发条件>，持续<时间阈值>)，则<软件模块>通过<协议>调用<设备接口>执行<动作>，[响应时间<系统需求指标>]。

例如：WHEN 温度异常事件（传感器检测温度≥38℃，持续5 s），则报警控制模块通过GPIO协议调用蜂鸣器/LED接口执行声光报警动作，响应时间≤1 s。

时序控制型：WHILE<系统状态>，<软件模块>需周期性调用<设备接口>执行<动作>[周期= <设备数据更新频率>]，[数据刷新延迟<系统需求指标>]。

例如：WHILE 系统正常运行状态，数据采集模块需周期性调用传感器接口执行温度读取动作 [周期=10 s](设备数据更新频率上限0.5秒/次）。

②非功能需求：包括精度需求、空间需求、可靠性和安全性需求等。

<精度需求约束声明>::= <物理量> 采用|使用 <精度>。例如：温度采集数据采用float32类型（32位浮点精度）。

<空间需求约束声明>::= <存储设备|存储媒介> 大于|等于|不高于 <空间类型值>。例如：应用软件编译后bin文件大小小于32 KB。

<可靠性需求约束声明>::= <数据> 格式的合法性检查|<数据> <边界要求> | <数据> 不允许除零| <数组> 不允许越界 | <控制> 不允许死循环。例如：温度数据不允许超出传感器量程（−10~50℃），数组存储时不允许越界访问。

<安全性需求约束声明>::=一旦 <故障发生条件>，那么 <故障处理>。例如：一旦传感器通信中断持续10 s，那么系统需触发黄色LED故障提示。

2.3 问题图与情景图

问题图表示需求及其实现的上下文[23]，而情景图表达需求实现中的交互序约束[23]。问题图和情景图的定义见定义4、定义5。

定义 4. 问题图。问题图是描述嵌入式系统需求与上下文环境交互边界的可视化模型，可描述为六元组<CS, DS, RS, RC, BI, EI>，其中：

1）CS（控制器集合）表示软件需求中承担控制逻辑的核心模块集合，对应软件维度的实现主体。

2）DS（领域集合）可以作为系统需求、软件需求交互的发起者或接收者，需求交互的外部实体与资源抽象包括设备维度的硬件实体（如传感器、执行器）、系统维度的被控对象（如"通信模块""存储单元"），是需求落地的物理载体。

3）RS（系统需求集合）是自然语言表达的系统需求集合，对应系统需求模板中的功能目标。

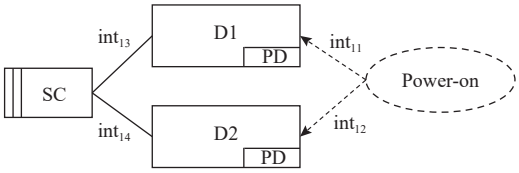
4）RC（约束关系集合）是表示需求对期望交互的引用/约束关系集合，描述需求与领域间的依赖规则，通过"需求→领域"或"领域→需求"的有向关联表达约束。

5）BI（行为交互集合）表示软件控制器与领域实体的动态交互动作，对应软件需求模板中的协议调

用逻辑。

6)EI(期望交互集合)是系统需求对领域实体的期望状态或性能指标。

图 2 展示了设备加电的问题图。针对设备加电的需求,为设备 D1 和 D2 进行加电。其中需求作用到设备上的是期望需求,即加电的状态。而控制软件(software controller, SC)作用到设备上的是需求行为交互的过程,即通过传达对设备的控制信号实施设备加电的行为。



int_{11}: <D1, SC, D1 Power-on state>[state]
int_{12}: <D2, SC, D2 Power-on state>[state]
int_{13}: <SC, D1, D1 Power-on signal>[signal]
int_{14}: <SC, D2, D2 Power-on signal>[signal]

Fig. 2 The problem diagram of the "equipment power-on"

图 2 "设备加电"问题图

问题图清晰界定了"谁(CS)在什么条件下(RC)与谁(DS)交互(BI),以达成什么目标(RS/EI)"的静态结构,为需求追溯提供可视化。

定义 5. 情景图。情景图是刻画嵌入式系统需求,实现过程中动态交互行为的模型,表达为四元组<BI, EI, CN, ES>,其中:

1)BI(行为交互集合):软件控制器与领域实体的具体动作序列(如"发送加电指令""检测电压信号"),对应软件需求模板中的协议调用与算法执行逻辑。

2)EI(期望交互集合):系统需求对行为结果的期望状态(如"设备 D1 加电成功"),要与系统需求模板中的性能指标绑定。

3)CN(控制节点集合):行为交互的触发条件与状态判断节点,包括事件节点、判断节点、动作节点等。

4)ES(边的集合):节点间的有向连接,描述时序或并发。

图 3 展示了针对设备加电需求的情景图。情景图中描述了期望的行为和行为活动。分别对 2 个设备加电可并发执行,没有先后次序的关系。

2.4 设备知识表示

设备知识是基于嵌入式系统设备属性特征构建的结构化表示。主要包含静态属性和动态属性两大类核心内容。其中,静态属性作为设备固有特征的基础描述,涵盖设备标识、设备类型及技术参数 3 个

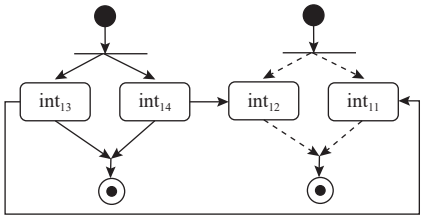


Fig. 3 The scenario diagram of the "equipment power-on"

图 3 "设备加电"情景图示例

关键层面:设备标识以厂商型号为主要标识信息,同时可关联别名以适应不同场景下的称呼需求;设备类型明确了设备的功能定位,分为传感器和执行器等类型;技术参数则进一步细化设备的技术特性,包含通信协议、状态管理参数及接口类型等具体信息,共同构成对设备静态特征的全面刻画。动态属性则聚焦设备在运行过程中的行为特性与交互规则,由功能能力、控制与操作、环境与约束、数据与交互 4 个维度组成。控制与操作维度包含设备的功能能力、具体控制指令及对应的操作步骤,定义了设备可执行的动作及实现方式;环境与约束维度明确设备正常运行所需的环境要求(如温度、湿度等)和性能约束(如响应时间、负载能力等);数据与交互维度则规范了设备的数据格式及交互流程,确保设备在数据交换过程中的准确性和一致性。静态属性与动态属性的有机结合,形成了对设备从固有特征到行为规则的完整知识体系。图 4 展示了设备知识库的层次结构。

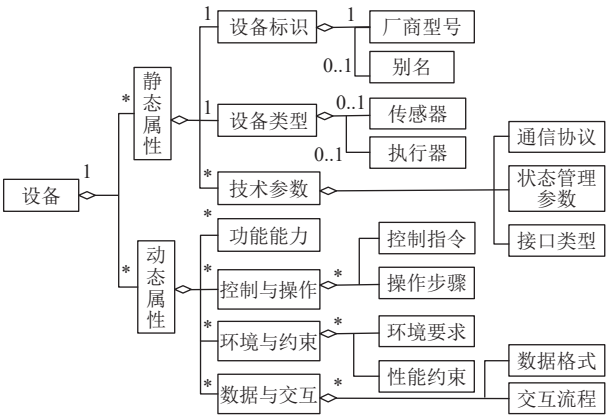


Fig. 4 Hierarchical structure diagram of the equipment knowledge base

图 4 设备知识库层次结构图

设备知识库在补充系统需求和软件需求中发挥着至关重要的作用。在系统需求层面,知识库中的设备类型(传感器/执行器)、技术参数(如通信协议、接口类型)等静态属性,直接影响系统的整体架构设

计，例如“室内温度异常监测”系统中，选用 DS1820 温度传感器（单总线接口、测量范围为−55℃~125℃、精度为−0.5℃~+0.5℃）的感知能力，决定了系统对室内温度（0℃~50℃）的监测范围和数据可靠性。环境要求和性能约束等动态属性则进一步明确了系统运行环境的边界条件，例如 DS18B20 的工作温度范围（−40℃~85℃）确保系统可适应冬季低温（0℃）与夏季高温（40℃）的室内环境。

在软件需求层面，设备的控制指令、操作步骤、数据格式及交互流程等动态属性，为软件功能模块的设计提供了直接依据，例如 DS18B20 的控制指令（如 0x44 启动温度转换、0xBE 读取数据）和数据格式（16 位二进制补码，高 8 位为整数部分，低 8 位为小数部分），定义了软件“数据采集模块”需通过单总线协议按“发送启动指令、等待转换完成、读取原始数据、校验并转换为温度”的操作步骤执行，确保温度数据的准确解析。

3 方法细节

图 1 展示的框架将嵌入式系统需求规约任务拆解为设备选型、系统需求生成、软件需求生成、需求建模及需求文档生成 5 个核心步骤，实现从用户意图到规范化需求文档的全流程。该方法以 LLM 为基础，结合设备知识库与需求描述模板，通过多轮用户交互优化结果，最终生成符合嵌入式系统开发规范的需求文档。其中，LLM 是一个多步骤引导的过程，提示框架说明如表 1 所示。

Table 1 Multi-step Guided LLM Prompt Framework

表 1 多步骤引导的 LLM 提示框架

母项	子项	内容描述
任务目标		<任务描述>
前置信息	输入内容	<具体输入内容>
	背景知识	<相关背景知识>
	上下文	<任务上下文信息>
任务拆解		<描述子步骤及依赖关系>
执行子步骤	前序结果引用	<引用信息>或无
	知识引用	<所需知识列表>
	处理过程	<执行流程描述>
	用户反馈整合	<交互反馈内容>或无
	输出结果	<当前步骤的输出>

3.1 设备选型

设备选型根据用户意图识别系统所需要的核心硬件设备，并通过用户确认形成设备列表。基于表 1 的提示框架，$prompt_1$ 的提示模板如表 2 所示。

Table 2 Instructions for $prompt_1$ Prompt Template

表 2 $prompt_1$ 提示模板说明

$prompt_1$	具体内容
任务目标	根据用户意图识别核心硬件设备，经用户确认形成结构化设备列表。
前置信息	1. 输入：user_intent 2. 背景知识：设备知识 3. 上下文：初始 user_intent
任务拆解	注入设备知识库→分解用户意图→设备匹配分析→生成候选→整合反馈
执行子步骤	1. 前序引用：无 2. 知识引用：K1 设备知识库、T1 设备推荐模板 3. 处理过程： ①注入 K1 至 LLM； ②分解意图为“系统能力+核心目标+业务价值”； ③匹配设备静态/动态属性生成候选。 4. 用户交互：工程师选择候选设备，如果对推荐不满意，再次提出问题，发起重新推荐 5. 输出：设备列表（要求提供型号/协议/接口/参数的信息，并列出推荐该设备是基于哪项系统能力、核心目标或业务价值的需要）

例如针对“室内温度异常监测系统应通过实时温度监测与自动报警能力实现室内温度异常（≥38℃）的快速识别与预警，达成降低火灾风险、保障人员安全的目的”这一意图，推荐用于实时采集室内环境温度的设备“DS18B20 温度传感器”和用于异常温度声光报警的设备“蜂鸣器”和“红色 LED”。

推荐的设备可能存在设备选型超出预算，或者交互方式等不符合预期的情况。此时，需求工程师需要勾选不满意的设备条目，并在问题提示框注明进一步推荐引导并发起重新推荐，直到所有设备满足需求后确认通过。

3.2 系统需求生成

基于已确认的设备列表，通过匹配设备能力与用户意图，定义系统级功能目标及性能约束，经过用户确认后形成系统需求列表。该步骤对应使用引导模板 $prompt_2$，如表 3 所示。

例如，针对“室内温度异常监测系统”的意图和设备选型，从感知能力和控制能力 2 个方面提供系统需求 SyR1 和 SyR2。SyR1:系统需通过 DS18B20 传感器实时监测室内环境温度，采样周期设置 10 秒/次，误差范围为−0.5℃~+0.5℃。SyR2:系统需通过蜂鸣器与 LED 对异常温度状态实现声光报警（蜂鸣器 1kHz 持续鸣响，LED 1Hz 闪烁），响应时间≤1 s。其中系统需求 SyR1 的推荐与“传感器最小采样间隔 0.5 s（硬件限制）”的知识相关，而 SyR2 的推荐与“蜂鸣器响应时间≤100 ms，LED 驱动电流≤20mA”硬件参数相关。

Table 3 Instructions for *prompt*$_2$ Prompt Template

表 3 *prompt*$_2$ 提示模板说明

prompt$_2$	具体内容
任务目标	基于意图和设备选型，形成系统需求列表。
前置信息	1. 输入：步骤 1 设备列表+user_intent 2. 背景知识：T2 系统需求模板 3. 上下文：步骤 1 设备列表
任务拆解	注入系统模板→意图转化系统能力→匹配设备参数生成需求→整合反馈
执行子步骤	1. 前序引用：步骤 1 设备列表 2. 知识引用：T2 系统需求模板、K1 设备知识库 3. 处理过程： ①注入 T2 至 LLM； ②意图转化为“感知/控制/通信”能力；将业务价值转化为可量化性能指标； ③结合步骤 1 设备列表参数分析功能逻辑（传感/处理/执行等）的可实现性，补充相关约束（如设备通信协议要求、环境适应性条件），生成候选系统需求列表。 4. 用户交互：工程师核对候选系统需求条目，发现存在“不完整”“描述不全面”等问题，再次提出问题，发起重新推荐 5. 输出：系统需求列表（基于系统需求描述模板描述系统需求，要求：标注系统需求的意图来源以及属于哪类能力（感知/控制/通信））

Table 4 Instructions for *prompt*$_3$ Prompt Template

表 4 *prompt*$_3$ 提示模板说明

prompt$_3$	具体内容
任务目标	拆解系统需求为模块化软件需求，明确交互逻辑与设备约束。
前置信息	1. 输入：步骤 1 设备列表+步骤 2 系统需求+user_intent 2. 背景知识：T3 软件需求模板，K1 设备知识库 3. 上下文：步骤 2 系统需求列表
任务拆解	注入软件模板+设备知识→拆解模块→生成软件需求→整合反馈
执行子步骤	1. 前序引用：步骤 1 设备列表、步骤 2 系统需求 2. 知识引用：T3 软件需求模板、K1 设备知识库 3. 处理过程： ①注入 T3+系统需求至 LLM； ②解析系统需求，将系统级能力映射到软件逻辑（感知/处理/执行/通信），选择 WHEN/WHILE 类型； ③结合设备接口协议与控制指令，生成可执行的软件交互逻辑（模块调用顺序、事件触发机制、数据流动路径），编写事件驱动型（WHEN）、时序控制型（WHILE）、非功能需求，形成完整的软件模块级需求集。 4. 用户交互：工程师确认规范性、完整性以及是否存在非功能缺失等情况，决定是否发起下一轮交互 5. 输出：结构化软件需求列表（基于软件需求描述模板表达软件需求，要求：标注软件需求的系统需求来源以及软件需求的类别）

在系统需求生成阶段，当 LLM 基于设备列表生成候选系统需求后，需求工程师核对发现超出意图范围、功能模块逻辑冲突（如报警响应时间≤0.5 s 与设备通信延迟 0.8 s 矛盾）、一些描述不完全等情况，可在问题提示框注明继续引导信息，如“进一步检查是否存在逻辑冲突，请仔细核对”发起重新推荐；LLM 基于这些提示重新匹配设备参数与系统需求，生成新候选列表，直到所有需求条目符合技术可行性和用户意图核心目标，工程师确认通过后形成最终结构化系统需求列表。

3.3 软件需求生成

对系统需求进行模块化拆解，明确软件与设备、外部实体的交互逻辑，需满足设备接口约束与系统性能指标。该步骤对应使用引导模板 *prompt*$_3$，如表 4 所示。

例如，“室内温度异常监测系统”将系统需求“监测环境温度”“声光报警”分解为“报警控制模块”以及“数据采集模块”两个功能性需求以及相关的非功能需求。其中，SsR1 和 SsR2 是功能需求，而 SsR3 和 SsR4 是非功能需求。SsR1: WHEN 温度异常事件（传感器检测温度≥38℃，持续 5 s），则报警控制模块通过 GPIO 协议调用蜂鸣器（PB0 引脚）和 LED（PA5 引脚）执行声光报警动作，响应时间≤1 s; SsR2:WHILE 系统正常运行状态，数据采集模块需周期性调用传感器单总线接口执行温度读取动作 [周期=10 s]（设备数据更新频率上限 0.5 秒/次）; SsR3: 温度采集数据采用 float32 类型（32 位浮点精度）; SsR4: 一旦传感器通信中断持续 10 s，则系统需触发黄色 LED 故障提示（PA6 引脚）。

3.4 需求建模

需求建模通过可视化方式整合设备、系统及软件需求，构建系统与外部实体的交互边界及动态行为逻辑。该步骤中，前序结果引用为全量上下文信息（意图、设备、系统需求、软件需求）；知识引用为需求建模知识（K2（含问题框架方法）、上下文图/问题图模型抽取知识及绘制规则[24]）。引导模板 *prompt*$_4$的设计如表 5 所示。

例如，图 5 展示了“室内温度异常监测系统”的问题图。图中包括外部实体、设备、需求、交互信息（数据、控制指定）等信息。

图 6 展示了需求“报警控制”模块针对 LED 指标灯控制的情景图。该图中上部展示了分解后的“LED 报警控制”问题图，包括设计领域“指示灯控制”对 LED 灯实施控制以实现控制行为。图下部分描述了期望需求和行为需求的过程。

3.5 需求文档构建

对前序步骤所有需求制品进行结构化整合，生成符合规约要求的文档，支持用户回溯修改。该步骤中，前序结果引用为需求建模阶段输出的可视化模型及全部上下文信息。知识引用为需求规约模板（包括任务意图、设备清单、需求列表、模型说明 4 类信息）。引导模板 *prompt*$_5$的设计如表 6 所示。

Table 5 Instructions for *prompt*$_4$ Prompt Template

表 5 *prompt*$_4$ 提示模板说明

prompt$_4$	具体内容
任务目标	可视化整合全量需求，构建交互边界与动态行为逻辑。
前置信息	1. 输入：步骤 1~3 全量结果+user_intent 2. 背景知识：K2 需求建模知识（问题图/情景图规则）；T4 问题图和情景图表达格式模板 3. 上下文：步骤 3 软件需求列表+历史上下文
任务拆解	注入建模知识+全量上下文→生成问题图→生成情景图→一致性检查+整合反馈
执行子步骤	1. 前序引用：步骤 1~3 全部结果 2. 知识引用：K2 需求建模知识，T4 模板 3. 处理过程： ①注入 K2+历史上下文至 LLM； ②从意图、系统需求以及软件需求的描述中识别问题概念和边，生成系统级问题图； ③基于问题图，理解行为交互和期望交互，识别交互逻辑，生成模块级情景图； ④根据问题图，建立意图、系统需求以及软件需求的追踪关系。 4. 用户交互：工程师查看问题图、情景图，发现不完整和不一致等问题，决定发起下一轮交互 5. 输出：问题图/情景图+结构化说明

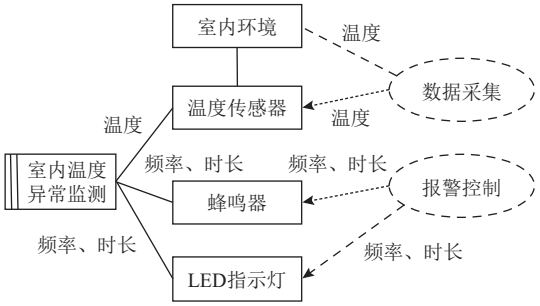


Fig. 5 “Indoor temperature anomaly monitoring system” problem diagram

图 5 “室内温度异常监测系统”问题图

用户可调整需求描述，通过检查追溯链查看完整性，并观察可追踪性。输出结果为需求规约文档，包含任务意图、设备清单、系统/软件需求、需求模型及追溯链。

整合上述步骤，算法描述如算法 1 所示。

算法 1. 基于知识提示链的嵌入式系统需求规约算法。

① *context* ← *user_intent*;
② for $i \in \{1,2,3,4,5\}$
③ *curr_template* ← *T*[*i*];
④ *curr_kb* ← *K*[*i*]; /* 当 *i*=4 时，*kb* 为 *K*[2]，其他情况 *kb* 为 *K*[1]*/
⑤ *curr_template.inject*(*context* + *curr_kb*); /*注入前序结果与当前知识库*/
⑥ loop:
⑦ *candidate_result* ← *LLM.generate*(*curr_template.messages* + *context*);
⑧ print(“候选结果：{*candidate_result*}”);
⑨ *user_feedback* ← *input*(“反馈（确认/修改/提问）：”);
⑩ *curr_template.messages.append*((“user”, *user_feedback*));
⑪ if *user_feedback* == “确认”
⑫ *structured_result* ← *curr_kb.extract*(*candidate_result*); /*从候选结果中人工选取的结果*/
⑬ *context* ← *structured_result*;
⑭ break;
⑮ else if *user_feedback*==“修改”

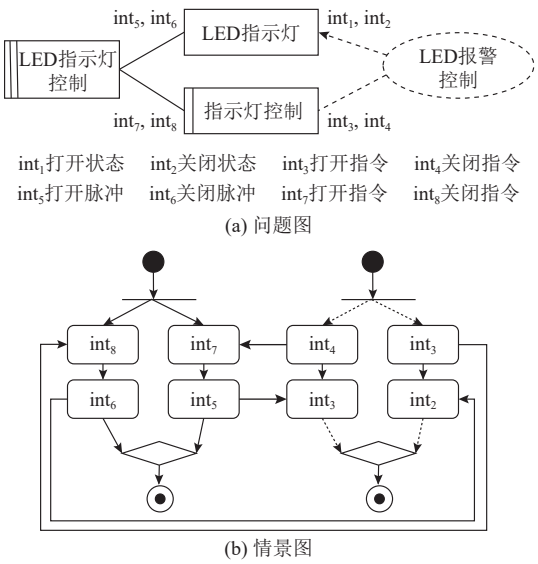


Fig. 6 Problem diagram and scenario diagram for “LED alarm control” requirement

图 6 “LED 报警控制”需求的问题图及情景图

Table 6 Instructions for *prompt*$_5$ Prompt Template

表 6 *prompt*$_5$ 提示模板说明

prompt$_5$	具体内容
任务目标	整合全部制品，生成符合规范的需求文档
前置信息	1. 输入：步骤 1~4 全部结果+user_intent 2. 背景知识：T5 需求规约模板 3. 上下文：步骤 4 建模结果+历史上下文
任务拆解	注入文档模板+全部上下文→生成初稿→构建追溯链→整合反馈
执行子步骤	1. 前序引用：步骤 1~4 全部结果 2. 知识引用：T5 需求规约模板 3. 处理过程： ①注入 T5+全部上下文至 LLM； ②生成文档初稿（意图/设备/需求/模型）； ③构建“意图→系统需求→软件需求”追溯链。 4. 用户交互：工程师根据文档中存在的“不完整”“不一致”“不充分”等问题，提供问题反馈，或发起重新推荐 5. 输出：完整需求规约文档（含追溯链）

⑯ *curr_template.messages.append* (*candidate_result*); /*在候选结果中增加信息*/
⑰ *context* ← *context.update* ({*modified_content*});
⑱ else if *user_feedback*==“提问”
⑲ *question* = *user_feedback*(“问题”); /*人工提供问题*/
⑳ *llm_answer* ← *LLM.generate*(*question*); /*LLM 再次生成回答*/
㉑ *curr_template.messages.append*(*llm_answer*); /*将 LLM 生成的回答追加到原有信息中*/
㉒ *context* ← *llm_answer*;
㉓ end loop
㉔ end for
㉕ return *context*。

4 实验验证

本节详细介绍实验验证过程。首先介绍实验数据，包括老人看护系统、太阳搜索系统、智能家居系统、战术控制系统 4 个嵌入式系统。其次，介绍研究问题及针对每个研究问题开展的实验设计。随后，介绍用于开展实验的工具。最后，对实验结果进行展示和分析。

4.1 实验案例

本文针对老人看护系统[2](elderly care system, ECS)、太阳搜索系统(solar search system, SSS)和智能家居系统[25](smart home system, SHS)、战术控制系统[25](tactical control system, TCS)四个案例开展实验研究。这 4 个案例分别来自健康监护、特定工业控制、智能家居领域和军事指挥与无人作战控制领域。其中，老人看护系统是教学案例，太阳搜索系统来自真实企业项目，另外 2 个项目来源于公开需求文档数据集，它们具有不同的规模，见表 7(使用设备数为 *DN*、系统需求数为 *SyN* 和软件需求数为 *SoN*)。我们从原始的需求描述中总结和提取了这 4 个案例的用户意图。初始意图描述如下：

1) 老人看护系统。通过实时监测生理指标、健康阈值预警，实现对老人生活健康的监测，保障老人身体健康以及预防突发风险的目标。

2) 太阳搜索系统。通过获得最佳光能采集角度、能及时响应地面操作员指令调整卫星状态，达到实时感知太阳位置并动态调整卫星姿态以追踪太阳的目的。

3) 智能家居系统。通过环境感知与自动调节优化室内温湿度及照明等参数提升居住舒适性。

4) 战术控制系统。通过构建分级无人机控制能力、任务规划与实时监控能力、多数据链协同控制能力以及 C4I 系统(Command, Control, Communications, Computers and Intelligence, 即指挥、控制、通信、计算机与情报系统)互操作能力，实现对无人机飞行与载荷的全流程控制、任务动态调整、跨区域通信保障与情报共享，在确保信息安全与系统可靠性的前提下，达成战场态势感知增强、联合指挥协同和精确作战支撑的目标。

Table 7 Scale Comparison of Experimental Cases in Terms of Device and Requirement

表 7 实验案例在设备与需求层面的规模比较

案例	*DN*	*SyN*	*SoN*
ECS	3	7	11
SSS	4	14	14
SHS	9	24	25
TCS	15	21	62

分别为这 4 个系统构建了知识库。目前，这 4 个系统已有设备数量分别为 10 个、10 个、18 个、36 个。每个设备描述了如图 4 所示的各类属性特征。我们统计了这 3 个已有设备知识库的规模(字符数)分别为 1 530 个、1 050 个、2 400 个和 6 500 个。

4.2 实验设计

为了检验本文方法的效率以及质量，提出了 4 个研究问题：

问题 1：方法在需求规约文档生成的效率和质量如何？

设计人工需求规约组(人工组)与本文方法(工具组)的对照实验，分别记录 2 组在同样“意图”和相同知识库背景下的需求规约过程时间开销，以及获取了具有相同格式的需求规约文档。人工组分为 4 组，每个项目包括 2 名硕士研究生(1 名为高年级学生，1 名为刚入学的低年级学生)，在无工具的环境下独立完成需求规约过程；工具组则使用本文方法提供的工具(详见 4.3 节)，由另外 4 名学生分别针对 4 个案例独立完成。

实验记录 2 组完成文档所需时间并对文档的质量进行评估。如表 8 所示的评价方法基于 IEEE830 标准[12]，包括无歧义性、完整性、一致性、可验证性、可修改性等维度，并根据嵌入式系统的特征进行了

Table 8 Explanation of Quality Evaluation Indicators for Embedded Software Requirement Documents

表 8 嵌入式软件需求文档质量评价指标说明

类型	子类	评价方法
无歧义性	语言模糊	含主观词的需求数量占比（如“快速响应”“友好界面”）
	不确定	含不明确的量词或程度词的需求占比（如“若干”“尽可能”“部分”“大约”“不少于”）
	指代歧义	含模糊代词（如“它”“该系统”“他们”）的需求占比
完整性	成分完整	软件需求中缺乏关键细节（如输入、输出、约束条件等）的占比
	非功能需求覆盖	涵盖性能、安全性、可靠性等维度的非功能需求类型的占比
一致性	冲突需求	需求条目间的逻辑矛盾数量在总需求条目的占比
	句式一致	采用标准句式风格的需求占比
可验证性	可测试	可测量的需求占比，可测量要求具有可观测的指标，且指标可感知
	验证条件明确	可验证的需求占比，可验证要求需求具有具体标准（如时间、数量、环境约束等），能直接用于判断是否处于达标的状态
可修改性	信息冗余度	重复出现的语义相同的需求数量占比
	独立性	不需要拆分为多个需求的数量占比

细化。该规则和评价策略提供给 LLM 进行评估。从表 8 中的各项指标的评价方法来看，各项指标的取值越小，反映出存在的问题越少。

问题 2：设备知识库在提升规约效率方面的作用如何？

为验证方法中设备知识库在需求规约效率提升方面发挥的作用，实验通过量化“需求信息推荐采纳率”和“交互次数”2 个核心指标，在 4 个案例中进行对比分析。采纳率指标定义如式(1)所示。

$$\text{采纳率} = \frac{\text{LLM 推荐且被采纳的需求条目数}}{\text{总需求条目数}} \times 100\%。 \tag{1}$$

我们基于方法中 LLM 反馈和人工交互的记录来量化采纳数(即反馈在交互工具中被人工确认采纳的数量)、再“提示”数(在交互工具中再次发起推荐申请的数量)和总的需求条目数。其中，需求条目数为设备列表、系统需求项和软件需求项的数量。推荐采纳率反映方法减少人工编写工作量的能力；而交互次数反映了当前推荐结果是否存在“需求冗余或失控”现象，需要进一步定位问题，重新推荐满足需要的信息。

问题 3：方法在确保需求可解释方面的能力如何？

为验证方法对需求可解释性的保障能力，实验通过“支撑度”和“覆盖率”分别在案例中进行对比分析。具体指标定义如式(2)和(3)所示。

$$\text{支撑度} = \frac{\text{被下层需求支撑的上层需求数}}{\text{总上层需求条目数}} \times 100\%, \tag{2}$$

$$\text{覆盖率} = \frac{\text{可追溯至上层需求的下层需求数}}{\text{总下层需求条目数}} \times 100\%。 \tag{3}$$

支撑度反映是否存在“意图或系统需求未落地”的情况，体现需求是否被有效细化与传递。也就是，上层需求的核心意图被下层需求实际支撑的比例越高，表明需求意图在细化过程中缺失越少，系统行为与用户目标一致性越高。覆盖度反映是否存在“无来源的孤立需求”，体现需求向下推荐。若可追溯至上层需求的下层需求数量占总下层需求数的比例越高，表明孤立或冗余需求越少，需求体系结构越完整。同时，下层需求与上层需求之间的关联越紧密，需求演化过程中的信息丢失风险越低。

本文针对意图、系统需求、软件需求 3 个层面量化支撑度与覆盖率。针对支撑度，分别将意图作为系统需求的上层需求，将系统需求作为软件需求的上层需求分别进行量化。式(2)中被下层需求支撑的上层需求表示上层需求分解为 1 至多个下层需求，且下层需求依据上层需求满足和达成上层需求。而在覆盖率方面，将系统需求作为意图的下层，而将软件需求作为系统需求的下层分别进行量化。式(3)中下层对上层需求的追踪表示下层需求可追溯到某一个上层需求，且该上层需求是它的源。

为验证本文提出的框架对需求文档可解释方面的优势，对比人工方法、单步 LLM 生成方法和本文提出的多步引导的方法。其中，单步 LLM 生成方法将用户意图和嵌入式系统需求规约过程要求直接输入 LLM(无多步引导与知识注入)，依赖模型自身能力完成需求规约。3 种方法生成的需求文档将输入给 LLM，采用表 6 所示的提示策略构建追溯链，并人工确认可满足和可追踪关系，分别度量支撑度和覆盖率，并进行量化对比。

问题 4: 不同 LLM 对本文方法的影响如何?

为了探究不同 LLM 对本文方法的影响, 设计对比实验, 重点考察不同大模型在方法核心步骤中推荐效果差异和性能表现。在本文中, 我们分别使用 GLM 系列[26](GLM-4.7-flash、GLM-4.7、GLM-5)、Qwen3 系列[27](Qwen3-1.7B、Qwen3-8B、Qwen3-32B)以及文心系列[28]LLM 进行对比实验与综合分析。其中, GLM 系列模型由智谱 AI 发布, GLM-4.7-flash 为轻量化版本, 具有响应速度快、推理成本低等特点, 而 GLM-4.7 和 GLM-5 则属于能力更强的大模型, 在复杂语义理解与推理任务中表现较为突出。Qwen3 系列模型由阿里巴巴公司推出, 提供从 1.7B、8B 到 32B 等不同参数规模的版本, 具有良好的开源生态与可扩展性, 其中参数规模越大的模型通常在复杂语义理解任务中具有更稳定的表现。文心系列模型则由百度公司研发, 属于商业化部署的 LLM, 在中文理解与生成任务中具有一定优势, 并提供轻量化模型以适应不同算力环境。通过选择不同规模与不同类型的 LLM, 评估本文方法在多种模型条件下的适用性与稳定性。

在实验设置上, 本实验不考虑人工交互过程, 而是以 4 个案例经过多次人工确认后形成的基准作为评价标准。该基准涵盖设备选型、系统需求和软件需求 3 个方面。实验过程中, 分别记录每个引导步骤后的模型反馈, 并通过准确率、召回率和 $F1$ 值 3 个指标进行评估, 如式(4)~(6)所示。

$$\text{准确率} = \frac{\text{模型正确推荐的数量}}{\text{推荐的总数量}}, \tag{4}$$

$$\text{召回率} = \frac{\text{模型正确推荐的数量}}{\text{基准中应推荐的总数量}}, \tag{5}$$

$$F1 = \frac{2\times\text{准确率}\times\text{召回率}}{\text{准确率}+\text{召回率}}。 \tag{6}$$

4.3 实验环境与工具

为了支持本文方法, 便于人与 LLM 的交互, 我们开发了适用性的工具。该工具主要包括知识注入的知识存储、需求信息推荐、需求模型的构建和需求信息的整合的核心模块, 构件图如图 7 所示。主要包括知识库基于知识的需求推荐、知识存储、模型构建和需求信息融合四大子系统。其中, 知识存储子系统提供各层级需求描述模板、设备知识和模型知识 3 类知识支撑, 提供这些知识的存储、维护的相关功能; 需求推荐子系统基于用户意图和设备信息, 自动推荐设备选型、系统及软件需求; 模型构建子系统生成需求可视化的上下文图和问题图; 最终通过需求信息融合子系统整合多源信息, 输出结构化的需求规约文档。

工具采用基于 Java 的编程语言, 遵循 MVC (model-view-controller) 的设计思想, 设计了前端交互

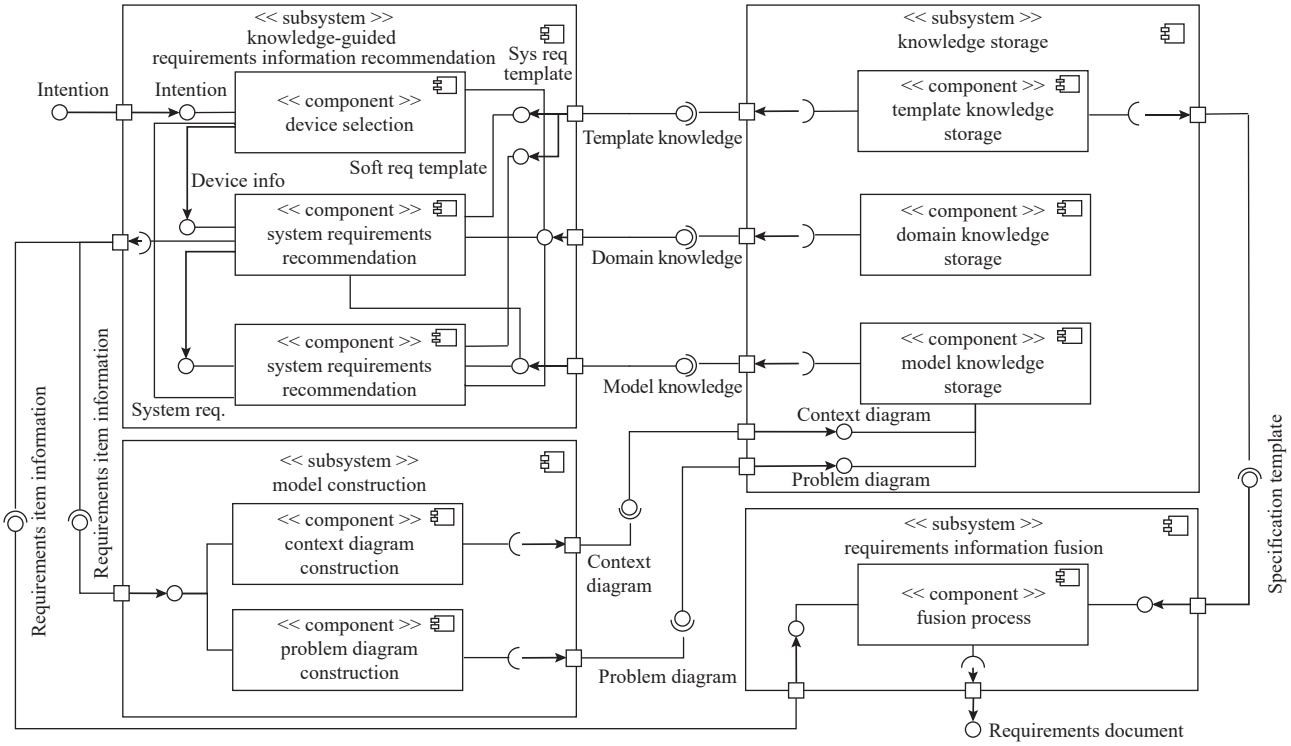


Fig. 7 The architecture diagram of the tool

图 7 工具的构件图

界面和数据库。在需求推荐子系统中我们选择了低部署成本、快推理速度且满足业务要求(详见问题 4 部分的结果分析)的文心系列的 ERNIE Lite(ernie-lite-8k)作为核心语言模型。经过参数调整，上下文窗口设置为 8 000 tokens，实际输入约 6 000 tokens，温度参数设置为 0.7。

4.4 实验结果与分析

1) 问题 1 的实验结果及分析

我们对老人看护系统、太阳搜索系统、智能家居系统、战术控制系统 4 个案例分别统计低年级学生、高年级学生以及本文方法在提供相同意图和设备知识、文档模板的情况下，开展需求规约过程(不包括建模)。表 9 展示了 3 组实验对象完成任务的时间。

Table 9 Time Spent on Manual Writing and Our Method Under the Same Intention

表 9 相同意图下人工撰写和本文方法所花费时间 min

案例	低年级学生	高年级学生	本文方法
ECS	83	105	15
SSS	101	97	18
SHS	98	71	25
TCS	135	93	49

从表 9 可以看出，本文方法在时间开销方面整体显著优于低年级和高年级学生，人工写作的平均用时为 98 min，而借助本文方法提供的工具，平均用时为 27 min(包括 LLM 推荐和人工交互的时间)，相比人工写作，工具写作平均可以节省约 72.7% 的时间。除了老人看护系统，其他 3 个案例中，高年级学生的总体用时低于低年级学生，但依然远高于本文方法。这表明，尽管高年级学生具备更扎实的专业基础和文档撰写能力，但在撰写嵌入式系统需求文档这一复杂任务中，仍然需要花费大量时间。

此外，我们还对人工撰写的文档与本文方法生成的文档进行了质量评估，采用表 8 所示的质量评价指标。表 10 展示了 4 个案例中无歧义性、完整性、一致性、可验证性、可修改性等维度的质量评估结果。其中，DN，SN 和 R 分别表示缺陷数量、需求总数以及占比。从表 10 的结果可以看出，本文方法在生成的需求总体数量高于人工的方法。即本文方法在 4 个案例中，平均需求数量为 38.05，而低年级学生和高年级学生生成文档的需求数量分别平均为 16.09 和 16.18。本文方法是人工方法生成需求数量的 2.37 倍和 2.35 倍。此外，从平均的需求问题占比来看，本文方法、低年级学生、高年级学生平均分别为 2.02%、11.12% 和 11%。本文方法相比人工平均水平在写作质量上的提升为 81.7%(质量差除以平均人工质量的百分比)。综上，在生成需求的数量以及质量整体方面，本文提出的方法均显著优于人工撰写，尤其在需求覆盖广度与错误控制方面表现突出。

2) 问题 2 的实验结果及分析

针对 4 个案例，分别在有无知识库的作用下执行本文方法，以分析知识库在需求规约文档写作效率提升方面发挥的作用如何。在实验中，采用“采纳率”

Table 10 Quality Evaluation of Requirement Documents Written Manually and Those Written by Our Method Under the Same Intention

表 10 相同意图下人工撰写和本文方法撰写的需求文档质量评价

类型	子类	案例	低年级学生			高年级学生			本文方法		
			DN	SN	R/%	DN	SN	R/%	DN	SN	R/%
无歧义性	语言模糊	ECS	3	15	20	1	18	5.56	2	28	7.14
		SSS	0	15	0	0	21	0	2	30	6.67
		SHS	0	14	0	0	21	0	0	37	0
		TCS	3	33	9.10	1	18	5.56	0	83	0
	不确定	ECS	2	15	13.33	0	18	0	0	28	0
		SSS	2	15	13.33	1	21	4.76	0	30	0
		SHS	3	14	21.43	4	21	19.05	1	37	2.70
		TCS	4	33	12.12	2	18	11.11	0	83	0
	指代歧义	ECS	1	15	6.67	0	18	0	1	28	3.57
		SSS	0	15	0	0	21	0	0	30	0
		SHS	0	14	0	0	21	0	0	37	0
		TCS	0	33	0	0	18	0	0	83	0

表 10（续）

类型	子类	案例	低年级学生			高年级学生			本文方法		
			DN	SN	R/%	DN	SN	R/%	DN	SN	R/%
完整性	成分完整	ECS	3	9	33.33	4	9	44.44	0	16	0
		SSS	4	8	50	3	12	25	0	19	0
		SHS	2	7	28.57	6	8	75	2	22	9.09
		TCS	1	16	6.25	1	9	11.11	2	62	3.23
	非功能需求覆盖	ECS	0	3	0	0	3	0	0	3	0
		SSS	0	3	0	0	3	0	0	3	0
		SHS	1	3	33.33	1	3	33.33	0	3	0
		TCS	1	3	33.33	1	3	33.33	0	3	0
一致性	冲突需求	ECS	0	15	0	3	18	16.67	2	28	7.14
		SSS	0	15	0	0	21	0	0	30	0
		SHS	2	14	14.29	1	21	4.76	0	37	0
		TCS	2	33	6.06	0	18	0	2	83	2.41
	句式一致	ECS	1	15	6.67	0	18	0	0	28	0
		SSS	0	15	0	0	21	0	0	30	0
		SHS	0	14	0	0	21	0	0	37	0
		TCS	0	33	0	0	18	0	0	83	0
可验证性	可测试	ECS	2	15	13.33	8	18	44.44	1	28	3.57
		SSS	4	15	26.67	3	21	14.29	1	30	3.33
		SHS	4	14	28.57	5	21	23.81	0	37	0
		TCS	4	33	12.12	3	18	16.67	2	83	2.41
	验证条件明确	ECS	4	15	26.67	1	18	5.56	2	28	7.14
		SSS	0	15	0	0	21	0	2	30	6.67
		SHS	0	14	0	0	21	0	0	37	0
		TCS	3	33	9.10	1	18	5.56	1	83	1.20
可修改性	信息冗余度	ECS	2	9	22.22	2	9	22.22	0	16	0
		SSS	0	8	0	2	12	16.67	2	19	10.53
		SHS	0	7	0	0	8	0	0	22	0
		TCS	0	16	0	0	9	0	0	62	0
	独立性	ECS	3	15	20	1	18	5.56	2	28	7.1
		SSS	1	15	6.67	5	21	23.81	0	30	0
		SHS	1	14	7.14	1	21	4.76	1	37	2.7
		TCS	3	33	9.10	2	18	11.11	2	83	2.41
平均			1.5	16.09	11.12	1.43	16.18	11	0.68	38.05	2.02

和“交互次数”作为度量的指标，表 11 展示了有无知识库的作用下，设备需求、系统需求以及软件需求的采纳率对比数据。其中，RN, EN, SN, AR 分别表示推荐数量，确认采纳数量、总数以及采纳率。

从表 11 可以看出，在 4 个案例中，使用设备知识库后，3 类需求的采纳率均呈现显著提升。以设备需求为例，ECS 案例中使用知识库后，AR 值从无知识库时的 17.39% 跃升至 100%，RN 从 23 个减少至 4 个，减少了 19 个，说明知识库有效降低了重复需求的提出；SSS 案例中，设备需求的 AR 值从 33.33% 提升至 100%，RN 从 12 个降至 4 个，减少了 8 个；SHS 案例中，设备需求的 AR 值从 18.92% 提升至 77.78%，RN 从 37 个降至 9 个，减少了 28 个；TCS 案例中，设备需求的 AR 值从 20.83% 提升至 93.75%，RN 从 72 个降至 16 个，减少了 56 个。系统需求方面，ECS 案例中使用知识库前后 AR 值均为 85.71%，无明显变化，但

Table 11 Adoption Rates for Three Types of Requirements Regarding the Using or Not Using the Knowledge Base

表 11 有无使用知识库的 3 类需求采纳率

案例	有无知识	设备需求				系统需求				软件需求			
		RN	EN	SN	AR/%	RN	EN	SN	AR/%	RN	EN	SN	AR/%
ECS	有	4	4	3	100	14	12	7	85.71	20	16	11	80
	无	23	4	3	17.39	14	12	7	85.71	18	14	11	77.78
	对比	19↓	0		83↑	0	0		0	2↑	2↓		22↑
SSS	有	4	4	4	100	13	11	14	84.62	21	19	14	90.48
	无	12	4	4	33.33	16	10	14	62.5	24	20	14	83.33
	对比	8↓	0		67↑	3↓	1↑		22↑	3↓	1↓		7↑
SHS	有	9	7	9	77.78	16	15	24	93.75	26	22	25	84.61
	无	37	7	9	18.92	19	15	24	78.94	27	22	25	81.48
	对比	28↓	0		59↑	3↓	0		15↑	1↓	0		3↑
TCS	有	16	15	15	93.75	24	21	21	87.5	73	62	62	84.93
	无	72	15	15	20.83	28	22	21	78.57	80	66	62	82.5
	对比	56↓	0		73↑	4↓	1↓		9↑	7↓	4↓		2↑

注：“↑”“↓”分别表示增加、下降。

SSS 案例中 AR 值从 62.5% 提升至 84.62%，提升了 22.12 个百分点，SHS 案例中 AR 值从 78.94% 提升至 93.75%，提升了 14.81 个百分点，TCS 案例中 AR 值从 78.57% 提升至 87.5%，提升了 8.93 个百分点，软件需求方面，ECS 案例中使用知识库后 AR 值从 77.78% 提升至 80%，提升了 2.22 个百分点；SSS 案例中 AR 值从 83.33% 提升至 90.48%，提升了 7.15 个百分点；SHS 案例中 AR 值从 81.48% 提升至 84.61%，提升了 3.13 个百分点；TCS 案例中 AR 值从 82.5% 提升至 84.93%，提升了 2.43 个百分点。整体来看，设备知识库对设备需求的优化效果最为显著，能大幅缩小推荐范围，提升采纳率；此外，对系统需求和软件需求也有不同程度的正向影响，其中系统需求的提升幅度在不同案例中存在差异，软件需求的提升幅度相对较小，但仍能有效优化需求质量。

在实验过程中，本文方法在不同的阶段实施了交互活动，即针对推荐结果进行再次提问以便于结果的再次反馈，通过再次推荐和采纳，减少人工撰写需求的工作量。表 12 展示了在不同阶段（设备需求、系统需求以及软件需求）交互次数。其中，IN 表示交互次数（新增加问题）。从表 12 可以看出，从交互次数来看，有设备知识库的情况下，交互次数较少（有设备知识平均为 1，而无设备知识平均为 1.75），交互次数平均减少 42.86%。

此外，我们也统计了交互的原因，主要分为 3 种情况：精度类、完整性以及模糊性。分别针对反馈的需求描述不精确、数据不正确或者表达不完全或者

Table 12 Three Types of Requirements Interactive Number of the Using or Not Using the Knowledge Base

表 12 有无使用设备知识库的 3 类需求交互次数

案例	设备需求		系统需求		软件需求	
	IN（有）	IN（无）	IN（有）	IN（无）	IN（有）	IN（无）
ECS	0	2	1	1	1	1
SSS	0	1	2	2	2	2
SHS	0	2	1	2	1	2
TCS	0	1	2	3	2	2
平均值	0	1.5	1.5	2	1.5	1.75

定义模糊的情况进行追问，提示 LLM 再次生成推荐信息，以减少人工的输入。

3）问题 3 的实验结果及分析

为检验生成文档在可解释性方面的效果，我们将每个案例生成的需求文档的意图、系统需求和软件需求进行编号，并在意图、系统需求、软件需求之间建立了追踪关系，即如果系统需求达成意图，软件需求实现系统需求，则在它们之间建立一条追踪边。这样，就构建了多层的追踪链。根据追踪链，我们统计了意图到系统需求以及系统需求到软件需求之间的可追溯数（即存在追踪边的数量）。同时也反向从系统需求到意图以及软件需求到系统需求统计了可满足的数量（即查看需求描述，核对精化逻辑）。

我们针对 4 个案例，开展了对比实验。分别为人工方法（包括低年级学生 H1 和高年级学生 H2）、只

提供意图的 LLM 单步方法(LLM-Single)和本文方法(Ours)。单步 LLM 的方法将同样的意图提供给 LLM, 并设计了提示词, 让 LLM 根据意图生成系统需求和软件需求。分别针对支撑度与覆盖率进行了评价, 对比情况见表 13 所示。其中, ISN, IN, SSN, SyN, SD 分别表示意图被系统需求支撑的数量、意图总数, 系统需求被软件需求支撑的数量、系统需求总数以及支撑度。TSyN, TSoN, SoN, CR 分别表示可追溯至意图的系统需求数量、可追溯至系统需求的软件需求数量、软件需求数量以及覆盖率。ASD, ACR 分别表示平均支撑度和平均覆盖率。

从表 13 可以看出, 本文方法在可解释性的 2 个指标“支持度”与“覆盖率”上取得了优异的指标(均为 100%)。这是因为本文方法将“意图、设备、系统、软件”各层次的需求借助需求模板和设备知识, 有步骤地从上而下实施需求信息的补全, 进而生成完整的需求文档。该文档中上层需求与下层需求之间具有清晰的可满足和可追溯关系, 确保了需求的可解释。表 13 还可以看出, 人工方法 H1 和 H2 的表现不稳定, 人工方法易受个人经验与案例熟悉程度的影响, 导致在不同场景下表现差异较大。LLM-Single 的支撑度表现较为突出, 在 ECS、SSS、SHS、TCS 四个案例中的平均支撑度分别为 86%、94%、91%、91%, 但覆盖度明显不足, 对应的平均覆盖率仅为 54%、72%、72%、69%。相比之下, 本文方法通过结构化知识引导与逻辑推理机制, 有效降低了人为因素的干扰, 提升了结果的一致性与可靠性。

图 8 展示了不同方法在各案例中支撑度与覆盖率的对比趋势。总体来看, LLM 的支撑度高于人工方法, 但覆盖率不足, 表现不稳定; 而本文方法在支撑度与覆盖率方面表现优异, 即在 LLM 辅助人工决策的作用下表现突出, 进一步验证了本文方法在可理解性方面的优势。具体而言, 在支撑度维度, 平均人工方法的支撑度为 76%, 而 LLM-Single 为 90.5%。在覆盖率维度, 平均人工方法的覆盖率为 94.5%, 而 LLM-Single 为 66.8%。从平均上来看, 本文方法在可维护性指标方面, 相比人工方法提升 15 个百分点, 相比 LLM-Single 方法提升 21 个百分点。

4)问题 4 的实验结果及分析

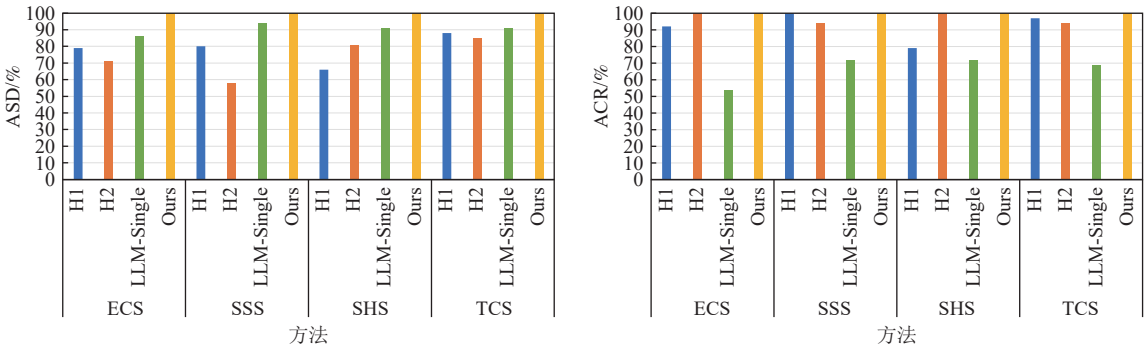
本文针对 ECS、SSS、SHS 和 TCS 四个案例, 对 7 款 LLM, GLM-4.7-flash、GLM-4.7、GLM-5、Qwen3-1.7B、Qwen3-8B、Qwen3-32B 以及 ERNIE-Lite, 在设备需求、系统需求和软件需求层的推荐性能进行了实验研究, 评价指标包括准确率、召回率和 $F1$ 值。

表 14 展示了不同 LLM 在各案例分层需求上的准确率、召回率与 $F1$ 值。实验结果显示, 在设备需

Table 13 Comparison of Manual Methods (H1 and H2), Direct Use of LLM, and Our Method in Interpretability Metrics (Support Degree and Coverage Rate)

表 13 人工方法(H1 和 H2)、单步 LLM 和本文方法在可解释性(支撑度与覆盖率)指标对比

案例	方法	意图-系统需求			系统需求-软件需求			意图-系统需求			系统需求-软件需求			汇总	
		ISN	IN	SD/%	SSN	SyN	SD/%	TSyN	SyN	CR/%	TSoN	SoN	CR/%	ASD/%	ACR/%
ECS	H1	3	4	75	5	6	83.33	5	6	83.33	9	9	100	79	92
	H2	3	4	75	6	9	66.67	9	9	100	9	9	100	71	100
	LLM-Single	4	4	100	5	7	71.43	3	7	42.86	15	23	65.22	86	54
	Ours	4	4	100	12	12	100	12	12	100	16	16	100	100	100
SSS	H1	3	4	75	6	7	85.71	7	7	100	8	8	100	80	100
	H2	2	4	50	6	9	66.67	8	9	88.89	12	12	100	58	94
	LLM-Single	4	4	100	8	9	88.89	7	9	77.78	15	23	65.22	94	72
	Ours	4	4	100	11	11	100	11	11	100	19	19	100	100	100
SHS	H1	3	4	75	4	7	57.14	6	7	85.71	5	7	71.43	66	79
	H2	4	4	100	8	13	61.54	13	13	100	8	8	100	81	100
	LLM-Single	4	4	100	18	22	81.82	16	22	72.73	24	34	70.59	91	72
	Ours	4	4	100	15	15	100	15	15	100	22	22	100	100	100
TCS	H1	14	16	87.5	15	17	88.24	16	17	94.12	16	16	100	88	97
	H2	13	16	81.25	8	9	88.89	9	9	100	8	9	88.89	85	94
	LLM-Single	16	16	100	14	17	82.35	11	17	64.71	33	45	73.33	91	69
	Ours	16	16	100	21	21	100	21	21	100	62	62	100	100	100



(a) 需求文档平均支撑度在案例中的表现对比 (b) 需求文档平均覆盖度在案例中的表现对比

Fig. 8 Comparison of average support degree and average coverage rate of different embedded requirement documents across four cases

图 8 不同嵌入式需求文档平均支撑度和平均覆盖率在 4 个案例中的对比

求层，GLM-5 与 Qwen3-8B 的表现最为突出，例如在 ECS 案例中两者的召回率均达到 100%，*F*1 值分别为 100%，显示了大模型在设备需求推荐上的优势。相比之下，GLM-4.7-flash 与 Qwen3-1.7B 的性能略低，在 SSS 案例中 *F*1 仅为 33.33%，而 ERNIE-Lite 的表现相对稳定，*F*1 值为 77.78%~88.89%，说明轻量化模型在设备需求推荐上仍具一定能力。

系统需求层整体推荐难度较大，*F*1 值波动明显，GLM-5 在 SHS 案例中 *F*1 达 60.87%，而 GLM-4.7 和 GLM-4.7-flash 多案例 *F*1 值低于 50%，显示部分模型在系统需求层存在召回不足的问题。Qwen3 系列模型表现较为均衡，其中 Qwen3-32B 在 SSS 案例中 *F*1 达 69.97%，表明更大规模模型在复杂系统需求推荐上具有优势。ERNIE-Lite 在系统需求层 *F*1 值为 50%~62%，表现中等。

在软件需求层，整体推荐性能优于系统需求层，GLM-5 与 Qwen3-32B 在 SHS 案例中 *F*1 分别达到 74.94% 和 74.41%，Qwen3-8B 在 TCS 案例中 *F*1 为 70.39%，显示了大模型在软件需求推荐的稳定性。ERNIE-Lite 在软件需求层表现相对较弱，例如在 TCS 案例 *F*1 为 58.16%，反映了轻量模型在复杂多层需求推荐任务中的局限性。

图 9 展示了 7 款 LLM 从准确率、召回率、*F*1 值 3 个核心维度在 4 个案例的设备需求、系统需求、软件需求推荐任务的平均表现。在准确率方面，所有模型都在 58%~81%，整体表现稳定。其中 GLM-5、Qwen3-8B、Qwen3-32B 三款模型表现最优，GLM-4.7、Qwen3-1.7B 次之，GLM-4.7-flash 和 ERNIE-Lite 准确率相对偏低，分别为 64% 和 59%。召回率方面，整体分布在 55%~74%，Qwen3-32B 达到 76% 的最高值；GLM-5 与 ERNIE-Lite 紧随其后，召回率约 71% 和 73%；GLM-4.7-flash 表现最弱，召回率不足 55%。在 *F*1 值方面，整体分布在 57%~75%，Qwen3-32B 的 *F*1 值最优，GLM-5、Qwen3-8 的 *F*1 值分别达到接近 73%；GLM-4.7-flash 最弱，不足 60%，其他模型总体表现在 65% 左右。

综合各案例与各层需求的实验结果可以看出，模型规模与性能呈正相关，大模型在跨层需求推荐任务中表现更为稳定；设备需求相对容易推荐，系统需求最具挑战性，软件需求表现居中；而轻量化模型如 ERNIE-Lite 在资源受限环境下仍具有一定应用价值，但在复杂需求场景下需要进一步优化。

5 讨论及有效性威胁

本文的有效性威胁主要来自 5 个方面。

1）实验案例规模和设备知识库规模相对有限。我们选取了 4 个覆盖不同复杂度和应用场景的嵌入式需求文档案例（ECS、SSS、SHS、TCS），但是这案例虽具有一定的代表性，但从规模和领域覆盖方面相对有限，需后续扩充数据集和实验规模。此外，本文采用人工构建的方法构建知识库，在知识库的完整性和精准性方面还缺乏验证。然而，本文提供的工具支持交互式的设备知识库动态维护，对知识库的进一步修正和扩展提供动态维护支持。

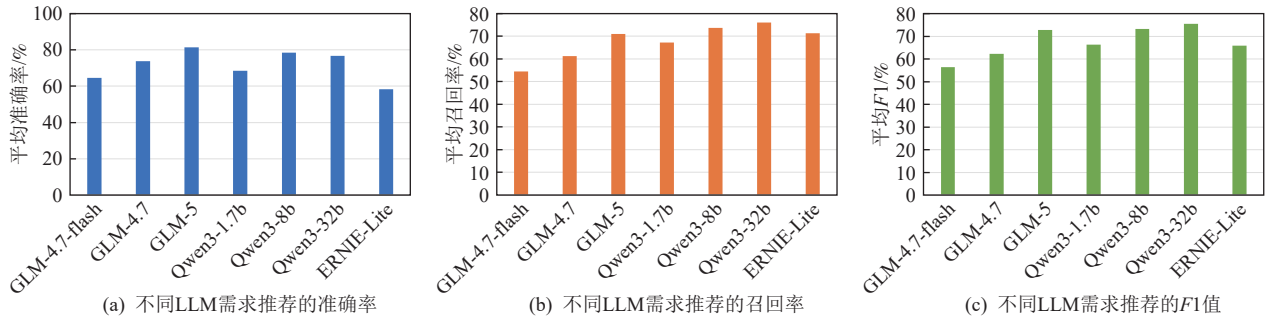
2）人工参与环节在参与人数和背景方面存在局限。人工评估涉及初学者和高年级学生，但参与人数较少，且他们对嵌入式系统的理解程度各异。初学者可能因专业术语或系统架构知识不足而对支撑度和覆盖率的评估产生偏差，而高年级学生的判断可能受个人项目经验影响，存在主观倾向。这些局限可能影响评估结果的可靠性。为缓解此问题，我们平衡了不同水平参与者的比例，但未来研究应扩大样本量并纳入专业嵌入式工程师，以提高评估的

Table 14 Performance Comparison of Different LLMs on Layered Requirements Across Different Cases

表 14 不同 LLM 在各案例分层需求上的性能对比 %

案例	模型	设备需求			系统需求			软件需求			平均 $F1$ 值
		准确率	召回率	$F1$	准确率	召回率	$F1$	准确率	召回率	$F1$	
ECS	GLM-4.7-flash	50	50	50	55.56	41.67	47.62	57.14	75	64.86	54.16
	GLM-4.7	80	100	88.89	50	25	33.33	75	56.25	64.33	62.18
	GLM-5	100	100	100	83.33	41.67	55.56	66.67	75	70.56	**75.37**
	Qwen3-1.7b	80	100	88.89	57.14	33.33	42.12	66.67	50	57.14	62.72
	Qwen3-8b	100	100	100	66.67	50	57.14	66.67	50	57.14	71.43
	Qwen3-32b	66.67	100	80	70	58.33	63.64	65	81.25	72.23	71.96
	ERNIE-Lite	80	100	88.89	50	50	50	60	56.52	58.21	65.70
SSS	GLM-4.7-flash	50	25	33.33	80	36.36	50	60	47.37	52.92	45.42
	GLM-4.7	60	75	66.67	100	36.36	53.33	72.73	42.11	53.34	57.78
	GLM-5	66.7	100	80.02	100	36.36	53.33	84.21	84.21	84.21	72.52
	Qwen3-1.7b	75	75	75	62.5	45.45	52.63	75	63.16	68.57	65.40
	Qwen3-8b	80	100	88.89	60	54.55	57.16	77.78	73.68	75.68	73.91
	Qwen3-32b	100	100	100	77.78	63.64	69.97	86.67	68.42	76.55	**82.17**
	ERNIE-Lite	66.67	100	80	55.56	45.45	50	75	63.16	68.57	66.19
SHS	GLM-4.7-flash	77.78	100	87.5	66.67	26.67	38.11	66.67	45.45	54.05	59.89
	GLM-4.7	77.78	100	87.5	71.43	33.33	45.44	80	54.55	64.91	65.95
	GLM-5	77.78	100	87.5	87.5	46.67	60.87	83.33	68.18	74.94	**74.44**
	Qwen3-1.7b	77.78	100	87.5	66.67	66.67	66.67	61.11	50	55	69.72
	Qwen3-8b	63.64	100	77.78	87.5	46.67	60.87	87.5	63.64	73.6	70.75
	Qwen3-32b	75	85.71	79.96	71.43	66.67	68.94	76.19	72.73	74.41	**74.44**
	ERNIE-Lite	63.64	100	77.78	64.29	60	62.06	57.69	68.18	62.45	67.43
TCS	GLM-4.7-flash	77.78	93.33	84.78	66.67	52.63	58.8	66.67	48.39	56.01	66.53
	GLM-4.7	76.47	86.67	81.22	77.78	36.84	50.02	62.5	56.45	59.27	63.50
	GLM-5	80	80	80	76.92	52.63	62.48	69.81	59.68	64.38	68.95
	Qwen3-1.7b	77.78	93.33	84.78	66.67	52.63	58.8	54.79	64.52	59.27	67.62
	Qwen3-8b	81.25	86.67	83.89	86.67	68.42	76.55	82.61	61.29	70.39	**76.94**
	Qwen3-32b	80	80	80	75	63.16	68.57	76.36	67.74	71.82	73.46
	ERNIE-Lite	68.18	100	81.09	55.56	52.63	54.02	1.90	66.20	58.16	64.42

注：黑体数值表示最优值。



(a) 不同LLM需求推荐的准确率 (b) 不同LLM需求推荐的召回率 (c) 不同LLM需求推荐的$F1$值

Fig. 9 Average Performance of different LLMs in equipment, system, and software requirement generation across four cases

图 9 不同 LLM 在 4 案例的设备、系统、软件需求生成平均性能

客观性。

3)LLMs 的固有随机性构成潜在干扰。本次实验选用的 7 款参数量从 1.7B 到 32B 不等的 LLM 普遍存在生成过程的固有随机性，同一 prompt 多次调

用的输出结果会存在细节差异，可能导致同一案例下的需求生成指标（准确率、召回率、*F*1值）出现小幅波动。不过从整体实验结果来看，这类随机性并未改变不同模型的性能排序，也未影响“大模型可有效提升需求生成效率”的核心结论，但在对输出一致性要求高的落地场景中，确实可能出现推荐结果偏离预期的情况，需要人工二次校验调整。本次实验已通过多轮重复测试取平均值、随机抽样核验的方式控制随机误差，未来研究可进一步通过领域微调优化模型输出分布、引入结构化提示词约束生成边界、叠加确定性规则校验模块等方式，进一步提升需求生成结果的稳定性与可复现性。

4）评估指标的设计可能存在覆盖不全的问题。当前采用的支撑度与覆盖率主要衡量需求的可解释性，需求文档的质量采用基于IEEE830标准的五维度评分法。我们对每一维度的评分法进行了针对嵌入式系统的评估量化，但其评估的全面性和精确性仍需在更广泛的场景中验证。

5）方法的人工交互成本分析需进一步深入。围绕人机交互频次、工具使用耗时、大模型辅助生成相较纯人工撰写的成本节约比例3个核心维度展开分析，验证了本文所提方法在效率与性能层面的综合优势。但当前成本分析仍存在覆盖边界的局限：方法落地所需的前期领域知识库构建质量、参与人员对嵌入式系统领域知识的熟悉程度等前置因素，都会对实际交互成本（尤其是时间成本）产生显著影响，而这类隐性影响尚未纳入本次成本评估体系，可能导致对全流程人力投入的测算存在偏差。后续研究将进一步扩大人工评估样本覆盖范围，深入分析人员专业背景、知识库成熟度等人为因素对交互成本的影响机制。

6 总结与展望

本文提出了一种基于LLM的可解释嵌入式系统需求规约框架，通过知识提示链交互式地引导LLM分步骤将用户意图细化为设备、系统及软件需求，构建问题图和情景图模型，建立需求之间的关联。构建了人工与工具对照组，实验结果表明，本文方法在时间、文档质量和可解释性方面具有优异表现。在时间维度方面，本文方法减少约73%，而可解释性相比人工方法和LLM-Single方法分别提升15个百分点和21个百分点。为复杂嵌入式系统需求工程提供了高效且可信的新途径。此外，本文还对比了不同步骤对7款不同类型和参数的LLM推荐效果的影响，通过分析准确率、召回率、*F*1值等指标可以看出，当参数越大时推荐的性能有小幅度提升。虽然本文选择了总体表现中等的文心大模型，但是得益于方法框架的通用性，用户也可根据实际场景的性能需求，灵活替换为更高性能的LLM以获得更优的推荐结果。未来将重点探索动态知识融合机制以增强设备知识适配性、扩展实验规模和面向复杂系统真实案例，进一步验证和提升需求文档的工程实践价值以及探索需求的动态演化和支持可复用。

作者贡献声明：王春晖负责方法整体设计、实验分析及论文撰写；赵佳琪完成案例数据收集、整理以及实验实现；陈小红和金芝提出指导和修改意见。

参 考 文 献

[1] Chen Xiaohong, Liu Shaobin, Jin Zhi. A survey of requirements description for embedded systems[J]. Journal of Software, 2025, 36(1): 27−46 (in Chinese)
（陈小红, 刘少彬, 金芝. 嵌入式系统的需求描述综述[J]. 软件学报, 2025, 36(1): 27−46）

[2] Jin Zhi, Liu Lin, Chen Xiaohong, et al. Software Requirements Engineering: Methods and Practice[M]. Beijing: Tsinghua University Press, 2023 (in Chinese)
（金芝, 刘璘, 陈小红, 等. 软件需求工程方法与实践[M]. 北京: 清华大学出版社, 2023）

[3] Voria G, Casillo F, Gravino C, et al. RECOVER: Toward requirements generation from stakeholders' conversations[J]. IEEE Transactions on Software Engineering, 2025, 51(6): 1912−1933

[4] Koscinski V, Hashemi S, Mirakhorli M. On-demand security requirements synthesis with relational generative adversarial networks[C] //Proc of the 45th IEEE/ACM Int Conf on Software Engineering (ICSE). Piscataway, NJ: IEEE, 2023: 1609−1621

[5] Lian Xiaoli, Ma Jieping, Lv Heyang, et al. ReqCompletion: Domain-enhanced automatic completion for software requirements[C] //Proc of the 32nd IEEE Int Requirements Engineering Conf (RE). Piscataway, NJ: IEEE, 2024: 142−154

[6] Almonte J T, Boominathan S A, Nascimento N. Automated non-functional requirements generation in software engineering with large language models: A comparative study[J]. arXiv preprint, arXiv: 2503.15248, 2025

[7] Habib M K, Graziotin D, Wagner S. ReqBrain: Task-specific instruction tuning of LLMs for AI-assisted requirements generation [J]. arXiv preprint, arXiv: 2505.17632, 2025

[8] Lowin M, Mihale-Wilson C. Functional requirements and recommendations for AI-based sustainable energy solutions[C]//Proc of INFORMATIK 2024. Bonn: Gesellschaft für Informatik, 2024: 1705−1716

[9] Zhao Ziyan, Zhang Li, Lian Xiaoli, et al. Reqgen: Keywords-driven

software requirements generation[J]. Mathematics, 2023, 11(2): 1−22
[10] Elliott R A, Allen E B. Creating a software requirements specification document using an ontology based methodology[J]. International Journal of Advanced Research in Science, Engineering and Technology, 2016, 3(9): 2616−2630
[11] Wang Chunhui, Zhao Jiaqi, Chen Xiaohong, et al. Generating requirements documents for embedded systems: A device knowledge-guided approach[C] //Proc of the IEEE 32nd Int Requirements Engineering Conf Workshops (REW). Piscataway, NJ: IEEE, 2024: 200−205
[12] IEEE Computer Society. IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications[S]. New York: IEEE, 1998
[13] Mavin A, Wilkinson P, Harwood A, et al. Easy approach to requirements syntax (EARS)[C] //Proc of the 17th IEEE Int Requirements Engineering Conf. Piscataway, NJ: IEEE, 2009: 317−322
[14] Kang K C, Ko K I. PARTS: A temporal logic-based real-time software specification and verification method[C] //Proc of the 17th Int Conf on Software Engineering. Piscataway, NJ: IEEE, 1995: 169−176
[15] Mahmud N, Seceleanu C, Ljungkrantz O. Specification and semantic analysis of embedded systems requirements: From description logic to temporal logic[C] //Proc of the 15th Int Conf on Software Engineering and Formal Methods. Berlin: Springer, 2017: 332−348
[16] Xuan Zhen, Wang Tianci, Wang Chunhui, et al. A tool for automatically identifying semantic conflicts in user stories by combining NLP and BERT model[C] //Proc of the 32nd IEEE Int Requirements Engineering Conf (RE). Piscataway, NJ: IEEE, 2024: 484−487
[17] Bertram V, Kausch H, Kusmenko E, et al. Leveraging natural language processing for a consistency checking toolchain of automotive requirements[C] //Proc of the IEEE 31st Int Requirements Engineering Conf (RE). Piscataway, NJ: IEEE, 2023: 212−222
[18] Chen Boqi, Babikian A A, Feng Shuzhao, et al. LLM-based satisfiability checking of string requirements by consistent data and checker generation[J]. arXiv preprint, arXiv: 2506.16639, 2025
[19] Krishna M, Gaur B, Verma A, et al. Using LLMs in software requirements specifications: An empirical evaluation[C] //Proc of the IEEE 32nd Int Requirements Engineering Conf (RE). Piscataway, NJ: IEEE, 2024: 475−483
[20] Yong G, Jeon K, Lee G. Prompt engineering for zero-shot and few-shot defect detection and classification using a visual-language pretrained model[J]. Computer-Aided Civil and Infrastructure Engineering, 2023, 38(11): 1536−1554
[21] Meck A M, Precht L. How to design the perfect prompt: A linguistic approach to prompt design in automotive voice assistants—an exploratory study[C]//Proc of the 13th Int Conf on Automotive User Interfaces and Interactive Vehicular Applications. New York: ACM, 2021: 237−246
[22] Shakthi S, Srivastava P, Kumar R, et al. Automated test case generation for satellite FRD using NLP and large language model[C] //Proc of the 4th Int Conf on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME). Piscataway, NJ: IEEE, 2024: 1−8
[23] Yang Xiao, Wang Xiaoqi, Chen Xiaohong, et al. Combined verification method for consistency of complex embedded system requirements[J]. Journal of Software, 2025, 36(4): 1413−1434 (in Chinese)
(杨晓, 王小齐, 陈小红, 等. 复杂嵌入式系统需求一致性的组合验证方法[J]. 软件学报, 2025, 36(4): 1413−1434)
[24] Jin Dongming, Jin Zhi, Chen Xiaohong, et al. ChatModeler: Human-AI collaborative iterative requirements elicitation and modeling method based on large language models[J]. Journal of Computer Research and Development, 2024, 61(2): 338−350 (in Chinese)
(靳东明, 金芝, 陈小红, 等. ChatModeler: 基于大语言模型的人机协作迭代式需求获取和建模方法[J]. 计算机研究与发展, 2024, 61(2): 338−350)
[25] Ferrari A, Spagnolo G O, Gnesi S. PURE: A dataset of public requirements documents[C]//Proc of the 25th IEEE Int Requirements Engineering Conf (RE). Piscataway, NJ: IEEE, 2017: 502–505
[26] Zeng Aohan, Xu Bin, Wang Bowen, et al. ChatGLM: A family of large language models from GLM-130B to GLM-4[J]. arXiv preprint, arXiv: 2406.12793, 2024
[27] Yang An, Li Anfeng, Yang Baosong, et al. Qwen3 technical report[J]. arXiv preprint, arXiv: 2505.09388, 2025
[28] Sun Yu, Wang Shuohuan, Feng Shikun, et al. ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation[J]. arXiv preprint, arXiv: 2107.02137, 2021



Wang Chunhui, born in 1979. PhD, professor. Her main research interests include software engineering and requirements engineering.
王春晖, 1979 年生。博士, 教授。主要研究方向为软件工程、需求工程。



Zhao Jiaqi, born in 2000. Master candidate. Her main research interests include software engineering and requirements engineering.
赵佳琪, 2000 年生。硕士研究生。主要研究方向为软件工程、需求工程。



Chen Xiaohong, born in 1982. PhD, professor. Her main research interests include requirements engineering, and knowledge based software engineering and formal method.
陈小红, 1982 年生。博士, 教授。主要研究方向为需求工程、基于知识的软件工程和形式化方法。



Jin Zhi, born in 1962. PhD, professor. Her main research interests include requirements engineering and software automation.
金　芝, 1962 年生。博士, 教授。主要研究方向为需求工程、软件自动化。