开发者指南

开发者指南：人工智能驱动的SCARA臂生成式设计框架

本指南概述了AI驱动SCARA机械臂生成式设计系统的开发流程。该系统利用自然语言控制Autodesk Fusion 360，并通过RAG系统和专业知识库进行增强。

I. 核心架构概览

系统将主要包含三大支柱：

1. 自然语言控制CAD (Cherry Studio & MCP 服务器)：用户将通过Cherry Studio发出自然语言指令。这些指令由Deepseek-R1模型处理，然后由一个增强的fusion360-mcp-server转换为可执行的Fusion 360脚本，用于生成式设计任务。
2. 专用SCARA知识库：将构建一个本地知识库，包含SCARA运动学原理、结构设计知识、材料特性以及成功设计案例（如DrawBot项目）。豆包（Doubao）将作为此知识库的嵌入模型，知识库将存储于向量数据库中。
3. 检索增强生成 (RAG) 系统：此系统将连接到知识库。当MCP服务器接收到用户指令时，RAG系统将检索相关信息以增强发送给LLM（Deepseek-R1或RAG流程中的其他LLM）的提示，从而生成更精确、更具上下文感知能力的Fusion 360指令。

概念流程如下：用户自然语言输入 (Cherry Studio) -> Deepseek-R1 (初步NLP处理) -> MCP 服务器 (查询) -> RAG 系统 (通过豆包嵌入进行知识检索 + 提示增强) -> LLM (优化指令生成) -> MCP 服务器 (Fusion 360 脚本生成) -> Fusion 360 (执行)。

II. 阶段一：在Cherry Studio上部署MCP服务

Cherry Studio AI 是一款强大的多模型 AI 助手，支持 iOS、macOS 和 Windows 平台。快速切换多个先进的 LLM 模型，提升工作学习效率。下载及安装地址：https://www.cherry-ai.com 。此阶段的目标是使Cherry Studio（使用Deepseek-R1模型）能够通过自然语言控制Fusion 360的生成式设计功能。

A. fusion360-mcp-server 设置

1. 基础部署：

- 克隆并部署现有的 fusion360-mcp-server 项目。

- 熟悉其现有架构，特别是用于工具定义的 src/tool_registry.json 和用于脚本模板的 src/script_generator.py。

3. 开发 Fusion 360 生成式设计工具扩展：

- 核心任务是扩展 fusion360-mcp-server 以支持Fusion 360中完整的生成式设计工作流。这包括为以下功能创建新的工具定义和相应的Python脚本生成器：
- 启动生成式设计研究：初始化新研究的命令。
- 定义设计空间：
- PreserveGeometry（保留几何）：指定模型中必须保留的部分（如电机安装面、关节连接处）。
- ObstacleGeometry（障碍几何）：定义“禁止进入”区域（如电缆路径、其他组件空间）。
- 施加结构载荷与约束：
- Loads（载荷）：定义作用在模型上的力、力矩或压力。
- Constraints（约束）：定义模型的固定点或边界条件（如固定约束、销钉约束）。
- 设定优化目标与限制：
- Objectives（优化目标）：例如，“最小化质量”或“最大化刚度”。
- Limits（限制）：安全系数、质量目标（用于最大化刚度时）、模态频率下限、位移限制。
- 选择制造方法与参数：增材制造（3D打印）、减材制造（铣削）、铸造，以及相关工艺参数（如3D打印的悬垂角度、铣削的刀具直径）。
- 选择研究材料：从Fusion 360材料库中为生成式设计研究选择材料。
- 详细的现有与所需能力对比，请参考项目建议书中的 表1。

5. 测试 MCP 扩展：

- 开发测试脚本，直接调用这些新的MCP工具，以确保它们能在Fusion 360中正确配置生成式设计研究。

B. 集成 Cherry Studio 和 Deepseek-R1 与 MCP 服务器

1. Cherry Studio 作为 NLP 前端：

- Cherry Studio 将作为用户输入自然语言指令的界面。
- 集成在Cherry Studio中的Deepseek-R1模型将对这些指令进行初步解析，以识别与SCARA臂设计相关的关键意图和实体（例如，臂长、负载、优化目标）。

3. 供 Cherry Studio 使用的 MCP 服务器 API：

- fusion360-mcp-server 需要暴露一个API端点（例如REST API），Cherry Studio可以调用此端点。
- 此API将接受来自Deepseek-R1的解析后指令或结构化数据。

5. 初步指令映射：

- 在MCP服务器内部（或作为Cherry Studio的预处理步骤）实现一个基础的映射逻辑，将（由Deepseek-R1解析的）简单、直接的自然语言指令转换为新开发的MCP工具调用序列。例如，“设计一个臂长分别为200mm和150mm的SCARA臂”可能会映射为一系列草图绘制和拉伸命令，然后是生成式设计研究的设置命令。
- 这个初步的映射逻辑将在后续阶段通过RAG系统得到显著增强。

III. 阶段二：构建本地知识库

此阶段专注于创建一个全面的、可查询的、特定于SCARA机械臂及其生成式设计的知识库。豆包（Doubao）将用作嵌入模型。

A. 知识获取与结构化

1. 数据收集：按照项目建议书IV.D.1节的规划收集信息：

- SCARA运动学原理：自由度、关节类型、DH参数（示例结构见表2）、正/逆运动学方程。
- SCARA设计原则：典型臂长、关节活动范围、额定负载能力、材料选择、结构特点（RRP、RRPR）。
- 案例研究：如DrawBot项目的详细数据（臂长202mm和196mm、GRBL配置、遇到的挑战如校准、直线绘制）。
- Fusion 360生成式设计：定义几何体、载荷、约束、目标的最佳实践；参数解释；结果解读。
- 材料属性：常用工程材料的力学性能。

3. 数据处理与格式化：

- 清洗与数字化：将所有数据转换为一致的数字格式（例如Markdown、JSON）。
- 信息结构化：
- 将表格数据（如DH参数、材料属性）存储为结构化格式（JSON、CSV）。
- 对数学公式使用LaTeX或类似格式，以便LLM更好地理解。
- 分块 (Chunking)：将长文档分割成较小的、语义连贯的“块”。每个块应足够独立，以便在检索时提供有意义的上下文。

B. 使用豆包（Doubao）进行嵌入和索引

1. 嵌入模型 (豆包)：

- 利用豆包模型为每个处理后的数据块生成向量嵌入。这些嵌入能够捕捉文本的语义信息。

3. 向量数据库设置：

- 选择并部署一个向量数据库（例如FAISS、Pinecone、Weaviate、Milvus）。
- 将豆包生成的嵌入及其对应的原始文本块和任何相关元数据（来源、类型、关键词）存储在向量数据库中。
- 确保数据库配置支持高效的相似性搜索。

IV. 阶段三：实现 RAG 系统

RAG系统将通过从知识库中检索相关信息来增强LLM的输入，从而提高生成的Fusion 360指令的质量和准确性。

A. RAG 系统架构

参考项目建议书IV.C.1节中描述的RAG架构。可以考虑使用LangChain或LlamaIndex等框架来简化开发流程。

1. 检索组件：

- 输入：从MCP服务器接收查询（例如，解析后的用户指令和识别出的参数）。
- 查询编码：使用相同的豆包嵌入模型将输入查询转换为向量。
- 语义搜索：在向量数据库中执行相似性搜索，找出最相关的top-k个数据块。
- （可选）重排序 (Re-ranking)：实现一个重排序步骤，以优化检索到的信息块的顺序，提高最相关信息的排序位置。

3. 生成组件 (LLM)：

- 上下文提示 (Contextual Prompting)：原始查询（或其来自Deepseek-R1的结构化表示）和检索到的相关数据块组合在一起，形成一个全面的提示。
- LLM选择：此LLM可以是Deepseek-R1本身（如果Cherry Studio的集成允许这种增强提示），也可以是专用于RAG系统的独立LLM实例（例如，Deepseek-R1的另一个实例，或通过Azure OpenAI等服务访问的模型，如建议书中所述）。
- 输出生成：LLM生成增强后的输出。理想情况下，此输出应为一组结构化的参数或一系列精确的MCP工具调用序列，并采用易于MCP服务器解析的格式（例如JSON）。这有助于：
- 澄清模糊的用户意图。
- 基于负载建议或自动补全缺失的设计参数（例如，安全系数、材料选择）。
- 将高级需求（例如，“快速且精确”）转换为具体的工程参数（例如，模态频率限制、刚度目标）。

B. 集成 RAG 与 MCP 服务器和 Cherry Studio

1. MCP-RAG 通信：

- 当MCP服务器从Cherry Studio接收到指令（经过Deepseek-R1初步解析后），它会为RAG系统构建一个查询。
- RAG系统处理此查询并返回增强后的、结构化的指令/参数。

3. MCP 中优化的脚本生成：

- MCP服务器解析RAG系统的输出。
- 如果RAG系统请求澄清（例如，由于输入模糊），则通过Cherry Studio将此信息反馈给用户。
- 否则，MCP服务器使用来自RAG输出的精确参数和工具序列来生成最终的Fusion 360 Python脚本。

5. 迭代循环：该过程允许迭代优化。用户的反馈或后续指令可以重新触发RAG增强的生成过程。

V. 开发建议

- 模块化设计：保持各组件（MCP扩展、知识库接口、RAG流程）尽可能模块化，并定义清晰的API，以便于开发、测试和升级。API数据格式建议使用jason。
- 版本控制：对所有代码、配置，理想情况下也对知识库内容进行版本控制（使用Git）。
- 迭代开发与测试：
- 每个阶段都从最小可行产品（MVP）开始。
- 持续测试NLP理解、RAG检索准确性以及生成的Fusion 360脚本的正确性。
- 为端到端测试定义具体的SCARA设计用例和性能指标（如项目建议书V.D节，任务3.1所述）。
- 提示工程 (Prompt Engineering)：需要投入大量精力来精心设计喂给初始NLP（Cherry Studio中的Deepseek-R1）和RAG系统LLM的提示，以引导它们生成准确且有用的输出。
- 知识库维护：RAG系统的质量在很大程度上取决于知识库的质量和结构。需持续投入精力进行知识库的优化和扩展。

通过遵循此分阶段方法并利用项目建议书中的详细见解，开发人员可以系统地构建所提议的人工智能驱动的SCARA机械臂生成式设计系统。