# 校务问答机器人质量属性

## 一、非功能需求与质量属性定义

### 1.1非功能需求（NFR）概述

1. **定义**

非功能需求（Non-Functional Requirements, NFR）是指不直接描述系统具体功能，却影响系统整体运行质量、用户体验、技术架构和长期可维护性的需求。相较于功能需求（如 “查询课表”“提交反馈”），非功能需求更关注系统 “如何实现” 和 “如何持续稳定运行”，是衡量系统是否成功的核心指标。

1. **核心作用**

用户体验保障：如响应速度、界面易用性直接影响师生使用意愿；

技术架构基础：并发处理能力、数据安全机制决定系统能否支撑校园高频访问；

合规与扩展：符合教育行业数据安全标准，支持未来功能迭代（如新增国际交流模块）。

### 1.2 校务问答机器人典型非功能需求

|  |  |
| --- | --- |
| **维度** | **具体需求示例** |
| 性能 | 考试周期间支持 1000 并发用户 / 秒，平均响应时间≤3 秒 |
| 可靠性 | 系统年可用性≥99.9%，故障恢复时间≤15 分钟 |
| 安全性 | 学生成绩数据加密传输与存储，符合《个人信息保护法》 |
| 易用性 | 新生 10 分钟内掌握核心功能，语音交互支持方言识别 |
| 兼容性 | 兼容微信 / 钉钉小程序及 IE11 以上浏览器 |

## 二、质量属性（IEEE 软件质量度量方法）

### 2.1 IEEE 标准定义

根据IEEE Std 830-1998《软件需求规格说明》及ISO/IEC 25010:2011《系统与软件工程 系统与软件质量要求和评价》，软件质量属性分为8 大特性，用于量化评估系统质量：

1. **功能性（Functionality）**

定义：系统满足明确或隐含需求的能力，包括适用性、准确性、互操作性等。

校务场景：问答准确率≥85%（覆盖 90% 高频校务问题）；与教务系统 API 对接，实时同步课表数据。

1. **可靠性（Reliability）**

定义：系统在规定条件下、规定时间内完成规定功能的能力，包括成熟性、容错性、恢复性。

校务场景：服务器年停机时间≤8 小时（可用性≥99.9%）；突发流量时自动触发熔断机制，避免系统崩溃。

1. **易用性（Usability）**

定义：用户理解、学习、使用系统的难易程度，包括易识别性、易学性、易操作性。

校务场景：界面元素符合师生认知习惯（如学生端采用活泼配色，教师端支持快捷键操作）；语音交互误识别率≤5%，支持 “智能纠错” 提示。

1. **效率（Efficiency）**

定义：系统资源利用效率，包括时间性能、资源利用性。

校务场景：复杂查询（如跨学期成绩分析）响应时间≤8 秒；空闲时段 CPU 利用率≤30%，降低校园云服务器成本。

1. **维护性（Maintainability）**

定义：修改系统的难易程度，包括易分析性、易修改性、稳定性。

校务场景：知识库更新后，新政策问答准确率提升≥10%；代码模块解耦，新增 “就业服务” 功能开发周期≤2 周。

1. **可移植性（Portability）**

定义：系统从一种环境迁移到另一种环境的能力，包括适应性、易安装性。

校务场景：支持从校园私有云迁移至公有云，配置文件复用率≥90%；小程序兼容 iOS/Android 主流版本，适配成本≤5 人 / 天。

1. **安全性（Security）**

定义：保护系统数据和资源免受未授权访问的能力，包括保密性、完整性、抗抵赖性。

校务场景：敏感信息（如留学生护照号）动态脱敏显示；管理员操作日志留存≥6 个月，支持审计追踪。

1. **兼容性（Compatibility）**

定义：系统与其他系统或组件协同工作的能力，包括共存性、互操作性。

校务场景：无缝对接学校现有的统一身份认证系统（如 CAS）；支持与校园 OA 系统共享通知数据，接口适配周期≤3 天。

### 2.2 非功能需求与质量属性的映射

#### 技术特征

|  |  |  |
| --- | --- | --- |
| 非功能需求 | 对应 IEEE 质量属性 | 量化指标 |
| 考试周并发处理 | 效率、可靠性 | 1000 用户 / 秒并发，响应超时率＜1% |
| 数据加密存储 | 安全性、可靠性 | AES-256 加密，数据泄露风险等级≤1 级 |
| 新生快速上手 | 易用性、功能性 | 新手引导完成率≥95%，操作错误率＜3% |
| 跨学期数据查询 | 效率、兼容性 | 多系统数据同步延迟≤1 小时 |

### 2.3 校务项目质量属性优先级

#### 校务项目质量属性优先级

|  |  |  |
| --- | --- | --- |
| **优先级** | **质量属性** | **核心场景** |
| P0（强制） | 安全性、可靠性 | 学生成绩查询、隐私数据管理 |
| P1（关键） | 易用性、效率 | 高峰期问答响应、多端交互体验 |
| P2（优化） | 可移植性、维护性 | 未来对接职业教育模块、国际交流功能扩展 |

## 三、开发者视角质量属性：可维护性、可移植性、可重用性、可测试性

### 3.1 可维护性（Maintainability）

#### 定义

系统易于修改、升级和故障修复的能力，确保开发团队能高效响应需求变更（如新增校务政策问答、优化 LLM 模型）。

#### 技术实现

代码架构：

采用微服务架构，将 LLM 模块、数据库模块、前端交互解耦（如 Spring Boot 后端接口独立部署），单个模块升级不影响整体系统。

知识库采用结构化存储（如 JSON 格式），支持管理员通过后台可视化界面批量更新政策文档，无需代码修改。知识库初期基于公开文档构建，暂不包含非结构化数据（如视频会议记录）；

文档规范：

代码注释覆盖率≥80%，关键逻辑（如 RAG 技术实现）需附设计说明；

维护《技术债务清单》，记录待优化点（如 “当前模型对‘辅修政策’理解准确率 75%，需在下一迭代优化”）。

#### 量化指标

需求变更平均响应时间≤2 个工作日（如新增 “国际学生签证咨询” 功能）；

模块修改故障率≤5%（通过自动化回归测试保障）。

### 3.2 可移植性(Portability)

#### 定义

系统从一个环境迁移到另一个环境的便捷性，包括硬件、软件、网络环境的适配。

#### 技术实现

容器化部署：

采用Docker 容器化，打包 Spring 后端、Vue 前端、MySQL 数据库为标准化镜像，支持一键部署至校园私有云或阿里云 / 腾讯云。

配置文件分离（如.env 格式），环境变量（数据库地址、API 密钥）可动态切换，无需修改代码。

跨平台兼容：

前端基于uni-app 框架开发，一次编码适配微信小程序、钉钉小程序、Web 端；

LLM 模型支持PyTorch 框架迁移，未来可平滑切换至其他模型（如从 Qwen 迁移至 ChatGLM）。

#### 量化指标

跨云平台迁移成本≤5 人 / 天（如从校园云迁移至公有云）；

浏览器兼容性测试通过率≥95%（覆盖 Chrome、Edge、IE11）。

### 3.3可重用性（Reusability）

#### 定义

系统组件、模块或代码可被其他项目或场景重复使用的能力，降低开发成本。

#### 技术实现

通用组件设计：

LLM 接口模块：封装通用的 “问题解析 - 模型调用 - 答案格式化” 流程，可复用于其他校园 AI 项目（如智能心理咨询机器人）；

权限管理模块：基于 RBAC 设计，支持不同角色（学生 / 教师 / 管理员）权限逻辑复用。

代码仓库管理：

在 GitHub 创建组织级代码仓库，标注可复用组件（如 “/common/llm-utils”“/front-end/components/question-list”）；

制定《组件复用指南》，说明依赖项（如需安装 PyTorch 2.0+）和集成步骤。

#### 量化指标

核心组件复用率≥60%（如数据库操作模块、用户认证模块）；

第三方项目集成平均耗时≤3 天（如对接后勤报修系统）。

### 3.4可测试性（Testability）

#### 定义

系统组件、模块或代码可被其他项目或场景重复使用的能力，降低开发成本。

#### 技术实现

自动化测试体系：

单元测试：对 LLM 模型接口编写 Pytest 用例，覆盖核心逻辑（如多轮对话上下文记忆）；

集成测试：使用 JMeter 模拟 500/1000 并发场景，验证服务器响应和数据库性能；

UI 测试：通过 Selenium 自动化测试网页端交互流程（如登录→提问→点赞反馈）。

调试工具：

后端集成Spring Boot Actuator，实时监控接口性能指标（如响应时间、吞吐量）；

前端支持浏览器开发者工具调试，记录组件渲染耗时和错误日志。

#### 量化指标

自动化测试用例覆盖率≥90%（功能测试 + 非功能测试）；

缺陷平均修复时间≤4 小时（严重级别缺陷）。

### 3.5开发者质量属性与项目规划的联动

|  |  |  |
| --- | --- | --- |
| 属性 | 当前技术方案 | 未来迭代规划 |
| 可维护性 | 采用 MVC 分层架构，模块间依赖≤3 层 | 引入低代码平台，支持无代码配置知识库规则 |
| 可移植性 | Docker+Kubernetes 部署 | 开发 “一键迁移” 脚本，支持 5 分钟内环境切换 |
| 可重用性 | 拆分通用 LLM 工具包（已发布至 PyPI） | 开源部分组件（如热搜排序算法），吸引社区贡献 |
| 可测试性 | 每周执行自动化冒烟测试 | 集成 CI/CD 流水线，代码提交时自动触发测试 |

### 3.6风险与应对

技术债风险：若紧急需求导致代码耦合度升高，需每季度进行架构评审，预留 10% 开发时间重构关键模块。

依赖兼容性风险：第三方库（如 PyTorch）升级时，通过版本锁定和兼容性测试矩阵确保平滑过渡。

## 四、用户视角质量属性：有效性、高效性、灵活性、完整性、互操作性、可靠性、强壮性、可用性

### 4.1 有效性（Availability）

#### 定义

系统在任何时间可被用户访问的能力，确保师生随时获取服务。

#### 用户场景

核心需求：

学生凌晨复习时查询 “考试安排”；

教师周末提交 “调课申请” 时获取机器人指引。

#### 量化指标

服务可用性≥99.9%，年累计不可用时间≤8 小时；

紧急维护提前 2 小时通知用户，非紧急维护安排在凌晨 0-5 点。

#### 实现策略

采用云服务器集群 + 负载均衡，自动切换故障节点；

提供网页端、微信小程序、钉钉小程序多入口，避免单一渠道故障。

### 4.2 高效性（Efficiency）

#### 定义

系统快速响应用户请求的能力，减少等待时间。

#### 用户场景

核心需求：

选课高峰期查询 “课程剩余容量”，秒级返回结果；

语音提问 “图书馆闭馆时间”，即时语音回复。

#### 量化指标

普通问答响应时间≤3 秒，95% 请求≤5 秒；

复杂查询（如跨学期成绩分析）≤8 秒。

#### 实现策略

缓存高频问题（如 “奖学金申请流程”）至 Redis，命中率≥90%；

对 LLM 模型进行轻量化优化，减少推理耗时。

### 4.3 灵活性（Flexibility）

#### 定义

系统适应不同用户需求和场景的能力，支持个性化交互。

#### 用户场景

核心需求：

国际学生用英语提问 “签证延期流程”，获得双语回答；

行政人员批量导入 “新生入学须知” 至知识库，快速更新。

#### 量化指标

支持多语言切换（中英日韩语），翻译准确率≥90%；

知识库内容更新延迟≤2 小时（管理员提交后）。

#### 实现策略

设计可配置的交互模板，支持管理员自定义回答风格（如正式 / 活泼）；

提供 API 接口，允许第三方系统（如招生系统）推送定制化问答内容。

### 4.4完整性（Integrity）

#### 定义

系统确保数据准确、完整且未被非法篡改的能力，保障信息可信度。

#### 用户场景

核心需求：

学生查看 “成绩排名” 时，数据与教务系统完全一致；

教师提交 “科研经费报销指南”，版本更新可追溯。

#### 量化指标

数据同步准确率 100%（与源系统对比）；

敏感信息泄露风险等级≤1 级（通过渗透测试验证）。

#### 实现策略

对接教务系统时采用双向校验（如课表数据 MD5 校验）；

知识库内容需经 “提交 - 审核 - 发布” 三阶段流程，记录操作日志。

### 4.5互操作性（Interoperability）

#### 定义

系统与其他校园系统协同工作的能力，避免信息孤岛。

#### 用户场景

核心需求：

通过微信小程序直接调用 “校园卡余额” 接口，无需重复登录；

机器人回答 “宿舍报修” 时，自动跳转后勤系统工单页面。

#### 量化指标

与 3 个以上校园系统（教务、学工、后勤）实现单点登录；

跨系统数据传输延迟≤1 秒。

#### 实现策略

遵循校园统一身份认证标准（CAS），集成 OAuth2.0 授权；

开发通用中间件，适配不同系统的数据格式（如 JSON/XML）。

### 4.6可靠性（Reliability）

#### 定义

系统在正常负载下稳定运行的能力，减少意外故障。

#### 用户场景

核心需求：

考试周期间， thousands of students 并发查询成绩，系统不崩溃；

机器人连续运行 365 天无内存泄漏或服务中断。

#### 量化指标

平均无故障时间（MTBF）≥10000 小时；

压力测试下（1000 并发），服务器 CPU 利用率≤80%。

#### 实现策略

采用微服务架构，单个模块故障不影响整体服务；

定期进行混沌工程测试（如模拟网络延迟、服务器宕机）。

### 4.7强壮性（Robustness）

#### 定义

系统处理异常输入或压力时的容错能力，避免崩溃或错误响应。

#### 用户场景

核心需求：

用户输入乱码或格式错误时，提示 “请重新输入有效问题”；

网络波动时，自动重试请求并提示 “正在加载，请稍候”。

#### 量化指标

异常输入拦截率 100%，友好错误提示覆盖率≥95%；

弱网环境下（2G 网络），界面加载失败率≤5%。

#### 实现策略

前端增加输入校验（如正则表达式验证学号格式）；

后端启用熔断机制（Hystrix），限制异常请求对系统的冲击。

### 4.8可用性（Usability）

#### 定义

系统易于理解和使用的程度，降低用户学习成本。

#### 用户场景

核心需求：

新生首次使用时，10 分钟内完成 “课程查询” 操作；

老年教师通过语音指令 “查询监考安排”，准确率≥95%。

#### 量化指标

新手引导完成率≥90%，操作错误率＜3%；

界面导航层级≤3 层，关键功能入口可见性≥90%。

#### 实现策略

采用 “卡片式” 信息展示，避免信息过载；

提供语音交互 “一键唤醒” 功能，支持方言识别（如粤语、四川话）。

### 4.9用户质量属性优先级矩阵

|  |  |  |  |
| --- | --- | --- | --- |
| **优先级** | **质量属性** | **典型场景** | **用户痛点** |
| P0（critical） | 可用性、高效性 | 考试查询、紧急通知推送 | 等待超时、操作复杂 |
| P1（high） | 可靠性、完整性 | 成绩 / 隐私数据查询 | 数据错误、信息泄露 |
| P2（medium） | 互操作性、灵活性 | 多平台登录、国际师生服务 | 系统割裂、语言障碍 |
| P3（low） | 强壮性 | 极端网络环境下使用 | 偶发输入错误处理 |

### 4.10与开发者视角属性的协同

|  |  |  |
| --- | --- | --- |
| **用户属性** | **对应开发者属性** | **技术联动点** |
| 有效性 | 可移植性、可靠性 | 容器化部署、双活数据中心 |
| 高效性 | 效率、可测试性 | 缓存优化、自动化性能测试 |
| 完整性 | 安全性、维护性 | 数据加密、版本控制 |
| **用户属性** | **对应开发者属性** | **技术联动点** |

## 五、确定质量属性的优先级

目的：当质量属性需求产生矛盾时，明确优先级有助于对需求属性进行合理取舍，确保系统在资源有限的情况下，仍能满足核心质量要求，平衡各属性间的关系，避免因次要属性过度消耗资源而影响关键属性的实现。

依据 IEEE 1061 - 1998《IEEE Standard for a Software Quality Metrics Methodology》：该标准提供了一种系统的软件质量度量方法，通过分析各质量属性间的关联关系，为确定优先级提供理论支持。

以下是对各质量属性的详细分析：

### 5.1正确性（Correctness）：

正确性是软件质量的基础，它确保系统的功能和输出符合预期的设计和用户需求。如果系统的计算、逻辑或数据处理不正确，其他质量属性（如可靠性、有效性等）将失去意义。例如，在校务问答机器人中，若对 “课程成绩计算规则” 的回答错误，即使机器人响应速度快（高效性）、长时间稳定运行（可靠性），也会导致用户对系统失去信任。

### 5.2可靠性（Reliability）：

可靠性关注系统在规定条件和时间内完成规定功能的能力。它与正确性紧密相关，只有在系统正确的前提下，讨论其是否可靠才有意义（表中用 “△” 表示关联）。例如，若校务问答机器人在高并发下频繁崩溃，即使其回答逻辑正确，也无法满足用户需求。同时，可靠性也影响有效性，不稳定的系统难以保证随时可用。

### 5.3有效性（Availability）：

有效性强调系统可被用户访问和使用的程度。它依赖于可靠性（系统稳定运行才能保证可用），同时也需要正确性作为支撑（正确的功能才能真正满足用户需求）。例如，若校务问答机器人因服务器故障经常无法访问（有效性差），即便其内部逻辑正确且可靠，用户也无法从中受益。

### 5.4完整性（Integrity）：

完整性确保系统数据和资源的准确、完整且未被非法篡改。它与可靠性相关（数据完整是系统可靠运行的保障），例如，若校务问答机器人的知识库数据被篡改（完整性受损），其回答可能误导用户，进而影响系统的可靠性和可用性。

### 5.5可用性（Usability）：

可用性关注用户理解、学习和使用系统的难易程度。它依赖于正确性（正确的功能才能让用户顺利使用）、可靠性（稳定的系统让用户愿意使用）和完整性（完整的数据使用户信任系统）。例如，若校务问答机器人界面复杂难用（可用性差），即使其回答正确、运行可靠，用户也可能选择其他更易用的方式获取信息。

### 5.6可维护性（Maintainability）：

可维护性影响系统在面对需求变更或故障时的调整和修复能力。它与正确性、可靠性、可用性等属性相关，例如，若系统难以维护（可维护性差），在发现回答逻辑错误（正确性问题）或出现新的用户需求时，无法及时调整，进而影响系统的可靠性和可用性。

### 5.7可测试性（Testability）：

可测试性决定了能否高效地验证系统是否满足质量要求。它与多个属性相关，如正确性（通过测试验证是否正确）、可靠性（测试系统在不同条件下的稳定性）、可维护性（测试修改后的系统是否引入新问题）等。例如，若校务问答机器人难以测试（可测试性差），则难以保证其正确性和可靠性，也会增加维护成本。

### 5.8灵活性（Flexibility）：

灵活性使系统能够适应不同的用户需求和场景变化。它与多个属性相互影响，例如，系统的灵活性可能依赖于其可维护性（便于修改以适应新需求），同时也影响可用性（灵活的系统能更好地满足不同用户的使用习惯）。如校务问答机器人需灵活支持多语言交互，以满足国际学生需求，这一过程涉及对系统的修改（可维护性），最终提升用户体验（可用性）。

### 5.9可移植性（Portability）：

可移植性关注系统在不同环境下的迁移能力。它与可靠性（在新环境中稳定运行）、可维护性（适应新环境可能需要调整系统）等属性相关。例如，若校务问答机器人要从校园私有云迁移到公有云（可移植性需求），需确保在新环境中仍能可靠运行，且迁移过程中的调整应便于维护。

### 5.10可重用性（Reusability）：

可重用性体现系统组件或模块在其他项目或场景中的重复使用价值。它与多个属性相关，如完整性（完整且独立的模块更易重用）、可维护性（便于修改以适应新场景）。例如，校务问答机器人的 “用户认证模块” 若设计得完整且易于维护，可被其他校园应用重用，降低开发成本。

### 5.11可互操作性（Interoperability）：

可互操作性确保系统与其他系统或组件协同工作。它依赖于完整性（数据格式完整统一）、正确性（交互逻辑正确）等属性。例如，校务问答机器人需与教务系统对接（可互操作性需求），若其数据格式不完整（完整性差）或交互逻辑错误（正确性差），将无法实现有效对接，影响整体功能。

通过对这些质量属性的深入分析，在实际项目中，可根据具体需求和场景，明确各属性的优先级。例如，对于校务问答机器人，正确性、可靠性和可用性通常具有较高优先级，因为它们直接影响用户对系统的基本信任和使用体验；而可移植性、可重用性等属性的优先级可根据项目的长期规划和资源情况进行调整。