



在线练习

第 3 章

如何定义好的软件需求

软件需求是软件开发的源头,如果源头就错了,后面的设计、编程和测试做得再好,也没多大价值。从这个角度看,需求非常重要,它决定了开发的方向是否正确、是否做正确的事——即是否开发出用户所需要的软件。准确捕捉并实现用户需求是提高用户满意度和产品市场接受度的关键,是项目成功的基础;一旦明确需求,则为开发团队提供了明确的开发指南,有助于实现预期的功能和性能。

如果定义了错误的方向,定义了错误的需求,有可能会造成软件开发工作前功尽弃,或者是后期返工很大。项目开始时,就把需求搞错了,导致项目失败,或造成开发的代价很大。例如,之前有一些经典的案例可以佐证。

- 2005 年,澳大利亚税务局(ATO)启动了一个项目来改革其商品和服务税(GST)的数据处理系统。由于需求收集不充分和需求变更管理不善,项目最终在 2009 年被取消,造成了大约 1.07 亿澳元的损失。项目失败的主要原因是需求定义不清晰,以及在项目实施过程中需求频繁变更。
- 2010 年,微软推出了 Kin 手机,这是一款面向年轻用户的智能手机。但由于需求定义错误,Kin 手机未能满足目标用户的实际需求,导致产品在市场上表现不佳,仅两个月后就被下架。项目失败的主要原因是目标市场和用户需求的误解。
- 英国国家医疗服务体系(NHS)的国家电子病历系统(National Programme for IT)项目是英国历史上最昂贵的 IT 项目之一,最终在 2011 年被取消。项目失败的主要原因包括需求管理不善和需求频繁变更。项目开始时,需求定义不够明确,导致在实施过程中需求不断变化,无法满足实际的医疗需求。此外,缺乏有效的沟通和协调也加剧了问题。
- F-35 联合攻击战斗机项目在软件开发过程中遇到了许多问题,其中包括需求的不断变化和缺乏清晰规划所导致的软件系统性能标准未能满足预期等方面的挑战。

所以,在软件研发中,需求定义是至关重要的一环。若需求定义不清晰、不稳定或经常变动,将导致软件开发过程中的许多问题,如进度延误、成本增加、设计错误等。本章将全面讨论和解决软件研发中的需求问题,从需求获取开始,逐步深入需求的各个环节,最终确保定义好软件需求,为软件后续的设计、实现打下坚实的基础,也尽量减少需求变更对项目造成的负面影响。

3.1 软件需求工程概要

本节先简要介绍软件需求工程的概要,让读者了解完整的软件需求工程的全貌,从而能

够理解本章为何有这些内容,以及各节内容之间的关系。为此,我们可以让大模型帮助生成一张“软件需求工程全景图”,如图 3-1 所示。

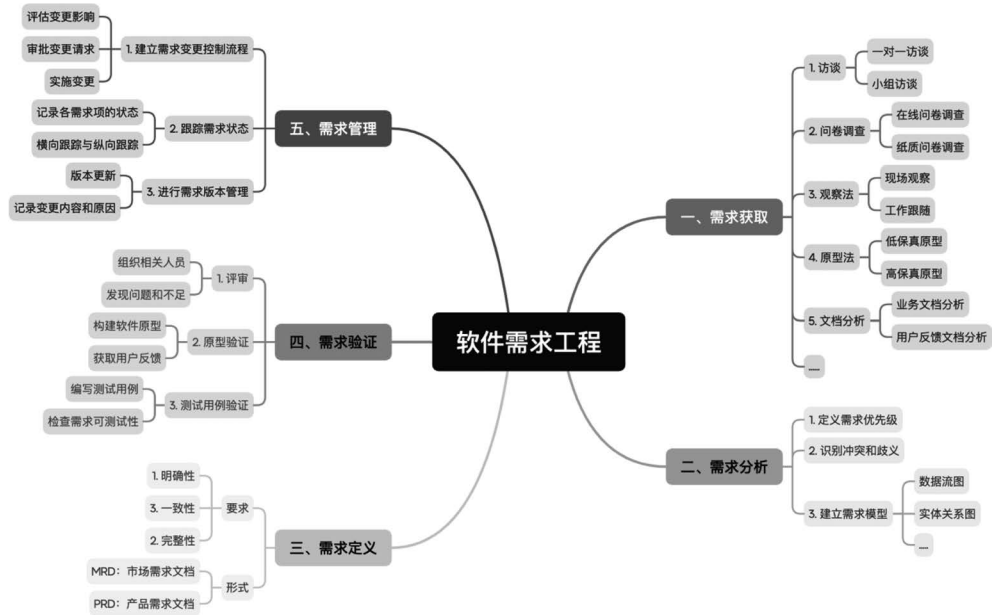


图 3-1 软件需求工程全景图

批判性思维训练

这张全景图不一定全对,但基本是对的。读者学完本章内容之后,也可以通过和大模型不断交互的方式重新生成一张更准确的软件需求工程全景图,这也就是我们经常讲的,对大模型输出的内容,都要带着批判性思维去分析和判断,然后再接受其输出结果,而且我们也是在学习、实践中成长起来的。

从图 3-1 了解到,一般将软件需求工程分为“需求获取、需求分析、需求定义、需求验证和需求管理”这 5 个阶段。需求获取会拿到第一手需求信息,需求经过整理、去伪存真,然后编写(在智能软件工程中,往往是“生成”)需求文档,再经过评审之后最终确定所定义的需求。之后,需求还可能会发生变更,需要进行跟踪管理。

根据《软件工程知识体系指南》(SWEBOK)以及相关需求工程标准,需求获取通常包括以下几个关键步骤。

(1) **定义问题**: 简要地将用户面临的问题进行记录,确保对问题有初步的了解。通过与用户的沟通,确认和澄清问题的具体细节,确保双方对问题的理解一致。

(2) **分析问题根本原因**: 旨在深入理解问题的本质,例如,采用如鱼骨图、5 个为什么等工具,系统地分析问题产生的深层原因,识别出关键因素。

(3) **分析涉众**: 以识别所有受软件项目影响的相关方,确保需求的全面性和准确性。

- **涉众识别**: 列出所有可能受项目影响的个人、团队或组织,如最终用户、管理层、技术支持团队等。
- **需求收集**: 通过访谈、问卷调查、工作坊等方式,收集各涉众的需求和期望。

- 利益分析：评估不同涉众的需求优先级，权衡各方利益，确保需求的平衡和可行性。

(4) **定义系统边界**：以明确软件系统与外部环境的接口和交互方式，确保系统功能的全面性和集成性。例如，对于一个复杂的订单系统，需要考虑这方面的问题——是否和其他系统共享数据、是否支持在线访问、如何与其他系统交互等。

(5) **确定约束条件**：以识别项目在实施过程中可能面临的限制和条件，确保需求的可实现性。例如，系统开发是否有预算限制？是否允许使用新技术？在完成时间上有限制吗？

(6) **需求验证与确认**：是确保需求准确性和完整性的关键步骤，包括需求审查、原型验证和需求跟踪。

3.2 软件有哪些需求

在具体讨论各项需求活动之前，首先需要全面、正确地理解“什么是软件需求”。需求就是对某种产品或服务的需要和要求。而 ISO/IEC/IEEE 29148:2011 则在软件工程标准词汇表中将“需求”定义为“转化或表达要求及其相关约束和条件的陈述”，而且这种要求会存在于不同层次，如软件组件需求、软件需求、系统需求等。所以 IEEE 24765 国际标准是这样解释“需求”(requirement)的：

- 为了解决问题或达到目标，用户所需的条件或能力；
- 为了满足协议、标准、规范或其他限定性文档，系统、系统组件、产品或服务需要具备的条件或能力。

其中，《项目管理知识体系指南(PMBOK)》中还增加了说明，即需求包含来自资助者(sponsor,如所在的公司或公司某高级管理人员)、客户(customer)和其他相关利益者(stakeholder)的、量化的、文档化的要求(needs)、需求(wants)和期望(expectations)。可以看到，上述需求定义中并没有提到产品的功能、非功能性，而有不少人一谈到需求，就说需要哪些功能，而忽视了资助者、客户和其他相关利益者，忽视了他们的要求和期望，这显然是错误的。而功能往往是我们设计出来的，所以可以说系统功能是我们给出的解决方案。虽然当系统功能定义并通过评审之后，可以把系统功能当作软件要实现的需求。

正确的思考方式，是从业务需求开始，然后再思考不同用户角色的需求，再到系统的功能需求和非功能性需求，这就是这里要讨论的需求层次。将需求分为三个层次来理解比较好，如图 3-2 所示。

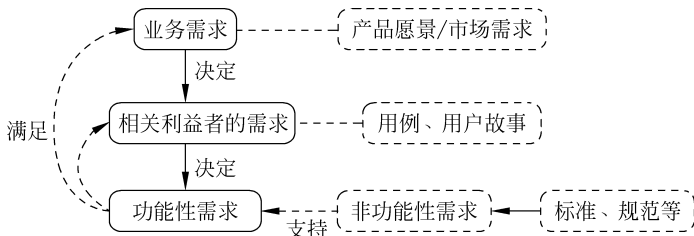


图 3-2 软件需求的层次结构(业务流程图)

- 业务需求：解决什么关键问题，核心的诉求是什么；达到什么目标，这相当于产品的愿景。

- 相关利益者的需求：每个用户角色有什么特定的需求？从用户行为去分析，即对应用例或用户故事，运维、技术支持等相关人员的需求，一般也可以并入用户角色的需求。
- 功能和非功能需求：软件为了满足上述业务需求、相关利益者的需求而具备的功能和非功能性(系统的能力)。

3.2.1 业务需求

有的需求模型会在业务需求上面增加一层：目标。目标是需求模型的最高层次，通常由企业战略和业务目标驱动。目标定义了组织希望通过软件系统实现的高层次目的。例如：

- 提高市场占有率。
- 增加客户满意度。
- 提升运营效率。

这些目标通常是定性的，反映了企业的愿景和长远规划。

为了解决问题或达到目标，也可以理解为业务需求。任何一款软件，必须有明确的目的，去解决用户的什么具体问题，提供什么样的能力帮助用户解决这些问题。其次，还要考虑到相关的协议、标准、规范等约束条件。业务需求往往包括业务流程、业务规则和业务数据，也包括业务可管理、可持续发展等需求。

(1) **业务流程**：业务流程是企业为了实现其业务目标而进行的一系列有序活动和步骤，包括每个步骤的输入、输出、活动和参与者。有效的业务流程设计对于提高企业效率和实现业务目标至关重要，我们可以为业务流程建模，如业务流程图。

(2) **业务规则**：是指导业务行为的约束条件和准则，确保业务流程的执行符合企业的政策和法规。如信贷审批流程中的信用评级规则决定了是否批准贷款，股票购买必须按100股的倍数来购买，每100股称为一手等。

(3) **业务数据**：包括在业务处理过程中输入的数据、输出的数据，如一些表单、报表等，我们可以为业务数据建模，包括实体-关系图(E-R图)、数据流图(Data Flow Diagram, DFD)。

3.2.2 用户角色需求

软件产品是为用户服务的、给用户使用的，所以我们会关注最终用户、用户需求，我们可能会问：

- 谁是软件产品的使用者？
- 用户会如何使用我们开发的产品？
- 针对每个功能，用户会如何操作？

看起来这样的提问是必要的，这里的“用户”似乎也很明确，大家都能理解，是指最终用户。但是，如果笼统地说“用户”，还是比较抽象的，其实我们忽略了特定的用户角色，不容易想象和分析用户的行为，而现实中或实际业务中存在不同的用户角色。例如，电商平台会有游客、普通买家、VIP买家和厂家直销商(卖家)、平台代理商(卖家)、平台客服、平台管理员等多种用户角色。要理解电商平台的用户需求，我们需要站在不同角色的角度、设身处地去想，才能更好地理解、把握用户的需求，才能挖掘出更多的用户需求。



观看视频

所以,在明确了业务需求之后,需要进一步区别不同的用户角色,因为不同的用户角色有不同的需求或期望。用户角色需求可以通过用例、用户故事、使用场景等方式来描述,以确保系统能够满足不同用户的具体需求和使用情境。所以,首先需要确定用户在业务处理或业务流程中扮演的角色,可以通过绘制业务流程图,识别出各种角色,从而知道不同用户扮演的角色。

(1) **用户角色 (User Role)**: 明确定义不同的用户角色,如管理员、普通用户、客户服务代表等,每种角色具有特定的权限和功能需求。

(2) **用例 (Use Cases)**: 用例是描述系统如何满足用户特定目标的一种技术性需求文档。每个用例描述了一个或多个场景,这些场景展示了系统如何与用户交互来实现一个特定的目标。例如,订单管理系统中的“创建订单”和“查看订单状态”可以是两个不同的用例。

(3) **用户故事 (User Stories)**: 用户故事是敏捷开发中用来描述用户需求的简短描述。每个用户故事通常包括一个简短的描述、价值和验收标准。例如,“作为一个客户,我希望能够快速找到产品信息,以便做出购买决策。”

(4) **使用场景 (Usage Scenarios)**: 使用场景描述了用户在特定情境下如何使用系统来完成任务。这些场景通常通过序列图或流程图来展示,以便详细说明用户与系统之间的交互和流程。

针对本书统一的案例团队协作系统(TMS),可以定义下列角色。

- 系统管理员
- 知识管理员
- 项目经理/协调员
- 内容创建者
- 协作参与者
- 访客用户

甚至可以让大模型帮助生成各角色所承担的责任,以及和其他角色之间的协作关系。

提示词: 请分析团队协作系统 TMS 的用户角色,并分析这些角色之间的协同关系

【大模型输出】

在团队协作系统 TMS 中,定义了 6 种角色: 系统管理员、知识管理员、项目经理/协调员、内容创建者、协作参与者和访客用户。以下是这些角色之间的协同工作关系:

1. 系统管理员

职责:

- 负责 TMS 的整体系统维护与管理。
- 配置系统设置和用户权限。
- 确保系统的安全性和稳定性。

与其他角色的协同如下。

- 知识管理员: 提供权限和技术支持,确保知识库的正常运行。

- 项目经理/协调员：协助配置项目管理工具和权限设置。
- 内容创建者和协作参与者：提供技术支持,解决使用中的技术问题。
- 访客用户：管理访客的访问权限,确保系统安全。

2. 知识管理员

职责：

- 负责管理和组织团队知识库内容。
- 确保知识的分类、整理和有效利用。
- 审核和发布团队博文和文档。

与其他角色的协同如下。

- 系统管理员：请求技术支持和权限设置。
- 项目经理/协调员：提供项目相关的知识和文档支持。
- 内容创建者：审核和编辑他们提交的内容。
- 协作参与者：指导他们使用知识库,提供所需的文档和资料。
- 访客用户：控制他们对知识库的访问权限。

3. 项目经理/协调员

职责：

- 负责项目的整体规划和协调。
- 分配任务并跟踪进度。
- 组织团队会议和日程安排。

与其他角色的协同如下。

- 系统管理员：配置项目管理工具和权限。
- 知识管理员：获取项目所需的知识和文档。
- 内容创建者：分配任务和收集项目相关的内容。
- 协作参与者：分配任务、监督进度并提供反馈。
- 访客用户：在必要时邀请访客参与项目讨论或提供反馈。

4. 内容创建者

职责：

- 创建和编辑团队所需的各种内容,如博文、文档、思维导图等。
- 确保内容的准确性和高质量。

与其他角色的协同如下。

- 系统管理员：请求技术支持和权限设置。
- 知识管理员：提交内容进行审核和发布。
- 项目经理/协调员：按照任务要求创建项目相关内容。
- 协作参与者：收集他们的反馈和建议,优化内容。
- 访客用户：根据项目经理的要求,可能为访客提供特定的内容查看权限。

5. 协作参与者

职责：

- 参与团队沟通和协作。

- 完成分配的任务并提供反馈。
- 参与团队博文和知识的创建与分享。

与其他角色的协同如下。

- 系统管理员：请求技术支持和权限设置。
- 知识管理员：使用和贡献知识库内容。
- 项目经理/协调员：接受任务分配并汇报进度。
- 内容创建者：提供内容创建的反馈和建议。
- 访客用户：协作参与者在必要时与访客用户共享信息或进行合作。

6. 访客用户

职责：

- 访问团队协作系统的特定部分,通常是受到邀请后参与特定项目或讨论。
- 提供外部的意见和建议。

与其他角色的协同如下。

- 系统管理员：被赋予适当的访问权限。
- 知识管理员：根据权限访问知识库的特定内容。
- 项目经理/协调员：参与项目讨论或提供反馈。
- 内容创建者和协作参与者：在必要时与访客用户共享内容和进行协作。

通过以上角色定义和协同关系,TMS 能够高效地实现团队协作、信息共享和任务管理,提高整体工作效率和协作效果。

知识点：提示词工程

提示词工程(Prompt Engineering)是自然语言处理(NLP)和人工智能领域中的一个重要概念,基于对语言、沟通和 AI 技术有深刻的理解,对 AI 系统输入的精心设计和优化,以提升 LLM 输出或任务处理上的性能和用户体验,实现更自然、更高效、更智能的人机交互。提示词工程始终围绕用户的具体目标或需求进行设计,并使用简洁、明确、无歧义的语言来构建提示词,避免不必要的复杂性和冗余,确保 AI 能够准确理解任务意图。提示词工程强调结构化构建,主要包含下列元素。

- 上下文：为 AI 提供任务背景等上下文信息。
- 任务说明(指令)：清晰表述要处理的具体任务或问题。
- 关键信息：提供完成任务所必需的核心信息、示例或数据。
- 期望结果：描述所期望的答案类型或输出格式。
- 限制条件：指出任何特定的限制或规则。

通过定量和定性的方法评估提示词的有效性,关注 AI 的响应速度、准确性和用户满意度,并建立反馈循环,利用用户和 AI 的响应来评估和优化提示词,实现持续的性能提升。当然,还需要考虑伦理和合规性、融入创新思维等。

基于此,还可以让大模型按照 SysML 规范,生成 Mermaid 格式的需求图,如图 3-3 所示。SysML(Systems Modeling Language)是一种用于系统工程的建模语言,它扩展了

UML(Unified Modeling Language)以支持系统工程的需求。Mermaid 是一个简单的图表绘制工具,虽然它不直接支持 SysML,但可以使用 Mermaid 来创建一个近似的需求图。以下是一个基于 Mermaid 的需求图示例,它展示了 TMS 中各个角色的需求和关系。

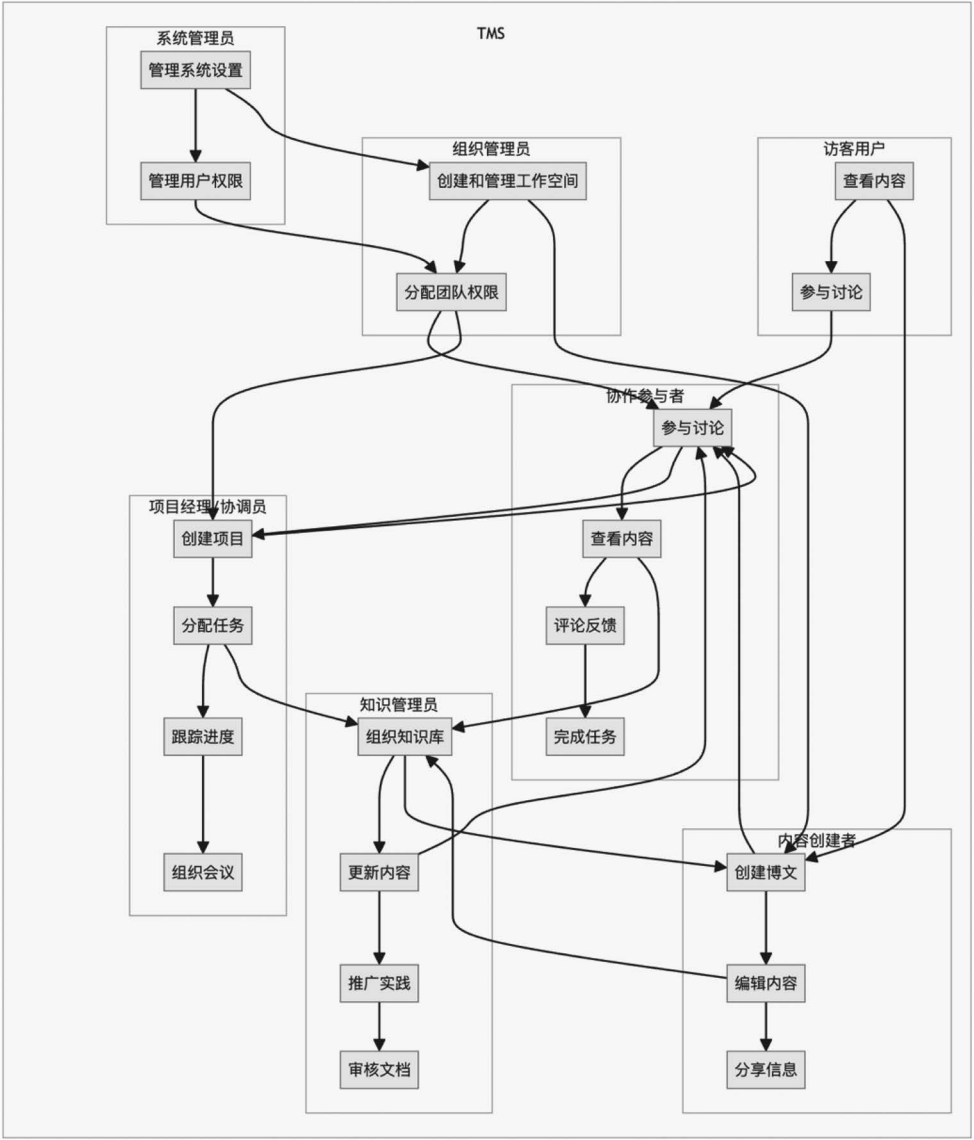


图 3-3 GPT-4o 生成的 SysML 需求图

如果觉得不对,可以通过 Mermaid 相关的工具进行编辑脚本来完善需求图。关于需求建模,后面还会有更深入的讨论,而针对用户角色需求的表示,基于敏捷开发模式,会采用用户故事来表达,后面也有专门的章节来讨论。

3.2.3 系统的功能需求

系统需求是需求模型的最底层,具体描述了软件系统需要具备的功能和特性,以满足用

户需求和业务需求。系统需求可以分为功能需求和非功能需求。

系统的功能需求(Functional Requirements)是描述软件系统必须执行的特定功能和行为的详细说明,即系统在达到其设计目标和满足用户期望时,必须具备的具体功能。这些需求通常描述了系统的行为、交互及处理数据的能力。功能需求通常以用户故事、用例或功能列表的形式呈现。

对本书的案例“TMS 团队协作系统”的功能列表：

1. 团队协作沟通功能

- 类似于 Slack 或 Bearychat 的团队沟通工具。
- 频道(组团沟通)和二级话题消息沟通。
- 私聊(一对一)。
- 支持 @消息、收藏消息、富文本消息目录。
- 有不同的频道,如外链、甘特图、任务看板等。
- 日程安排、待办事项和提醒,支持邮件通知、桌面通知等。
- 沟通消息标记表情和标签,方便分类过滤检索。
- 剪贴板上传图片,拖曳文件上传。
- 文件上传,从 CSV、Excel 导入 Markdown 表格。
- 自定义频道组、皮肤色调。

2. 团队博文(Wiki)

- 类似于精简版的 Confluence 或蚂蚁笔记。
- 博文空间,便于博文组织和权限隔离。
- 支持 Markdown、HTML 富文本、思维导图、图表等多种创作方式。
- 基于博文模板创建,可发布私有、公开模板。
- 博文目录、标签、父子级博文(支持 5 级)。
- 博文关注、收藏、版本比较、权限、点赞、分享、开放游客访问。
- 博文评论。
- 博文多人协作编辑(需开启协作权限)。
- 导出 PDF、Markdown、HTML、Excel、PNG。
- 基于 WebSocket 的博文更新实时通知。

3. 国际化翻译管理

- 翻译项目管理。
- 翻译语言管理。
- 翻译导入/导出。
- 翻译管理。

4. 其他功能

- 图像(系统界面的截图)、GIF(系统操作的动态)展示。
- 系统设置。
- 用户管理。

功能需求是软件开发的基石,它们直接影响到系统的设计、实现与测试。如果功能需求不明确或不完整,可能导致系统无法满足用户需求,进而影响项目的成功。功能需求的三个关键属性是正确性(Correctness)、完备性(Completeness)和适合性(Appropriateness)。

(1) **正确性**指的是软件系统执行的功能需求必须准确无误地符合预期的业务规则和逻辑。例如,一个在线购物平台的功能需求是“用户可以添加商品到购物车”,它意味着用户能够将其要购买的商品放入购物车后,系统能准确地反映出这一操作,包括购物车增加了相应的商品信息,包括商品的数量、价格等。

(2) **完备性**指的是软件系统的功能需求应该覆盖所有用户或客户的业务需求,没有遗漏。例如,对于一个银行的自动取款机(ATM),其功能需求应该是完备的,包括但不限于取款、存款、查询余额、转账等。如果 ATM 缺少了查询余额的功能,那么它就不满足完备性的要求。

(3) **适合性**指的是软件系统的功能需求恰到好处地满足特定的业务场景,即不能脱离业务的实际应用场景。例如,一个图书馆管理系统可能需要一个功能需求是“用户可以搜索图书”。适合性意味着搜索功能应该提供足够的搜索选项(如按标题、作者、ISBN 等),但同时不应该包含过于复杂或对大多数用户不实用的高级搜索算法,这可能会增加系统的复杂性而对大多数用户并无实际帮助。

3.2.4 系统非功能性需求

系统的非功能需求(Non-Functional Requirements)描述除功能之外的系统其他质量特性,这些特性通常与软件的内部设计和实现有关,而不是软件执行的具体任务。如性能(效率)、安全性、兼容性、可靠性等,可以参考 ISO/IEC 25000 质量模型——SQuaRE(Software Quality Requirements and Evaluation)标准。

- **效率(Efficiency)**涉及软件使用系统资源的能力和响应时间,外部体现为系统的性能。例如,一个电子商务网站的非功能性需求可能是支持每秒数千次的交易处理,同时确保页面加载时间不超过 3s。
- **可靠性(Reliability)**指软件在面对错误输入或异常情况时,仍能正常运行的能力。例如,自动取款机(ATM)的软件需要能够处理断网或电源故障,确保用户交易的完整性和资金安全。
- **易用性(Usability)**涉及软件的易理解和用户界面的友好性。例如,一个移动应用程序的非功能性需求可能包括直观的用户界面设计,使用户能够快速学会如何使用应用程序。
- **兼容性(Compatibility)**指软件与其他系统或组件交换数据和协同工作的能力。例如,一个企业级软件系统可能需要与多种数据库系统兼容,如 MySQL、Oracle 和 SQL Server。
- **安全性(Security)**涉及保护软件免受恶意攻击和未经授权访问。例如,一个在线银行系统必须具备高级加密和多因素认证,以确保交易的安全性。
- **可维护性(Maintainability)**指软件在生命周期内进行修改和升级的容易程度。例如,一个软件系统应该使用清晰的代码结构和文档,以便于未来的维护和升级。
- **可移植性(Portability)**指软件从一个环境迁移到另一个环境的能力。一个桌面应用

程序可能需要移植到移动平台,如从 Windows 移植到 iOS 或 Android。

非功能性需求对于软件的整体质量和用户体验至关重要。它们通常在软件设计和架构阶段被定义,并在整个开发过程中持续关注和实现。

3.2.5 其他需求类型

前面讨论了业务需求、用户角色需求和系统的功能、非功能性需求,这些需求都是来自客户(或用户)的需求,是软件主要的需求。但还有一些需求,来自研发团队的内部,要么是团队之前工作没做好,产生了缺陷和技术债务所带来的需求,要么是团队内部对系统部署、技术支持等提出的需求。特别是在今天 DevOps 流行,我们更应该要关注运维方面的需求。

- **运维需求**(Operational Requirements)涉及软件部署、监控、维护和支持等方面的需求,包括易部署性、可监控性、灾难恢复和业务连续性等。例如,一个 Web 应用软件能够容易被部署到生产环境中,包括自动化部署和回滚机制,以及需要不间断(24×7)的监控系统来跟踪性能指标和错误日志,确保系统稳定运行,并快速响应任何潜在的运维问题。
- **缺陷修正**(Defect Resolution)指识别、记录、修复软件中的缺陷或错误,并验证修复结果的过程。例如,一个软件产品在发布后,用户报告了一个登录问题。开发团队需要迅速响应,重现问题,定位缺陷,并发布一个修复版本。
- **技术债务**(Technical Debt)是指在软件开发过程中,由于采取了捷径或权宜之计而产生的程序结构混乱、代码可读性降低,增加了额外工作量或长期维护成本。例如,为了快速发布一个功能,开发团队可能暂时忽略了代码重构,导致代码质量下降。长期来看,这可能需要更多的时间和资源来解决由此产生的问题。在此后的迭代开发中,需要偿还技术债务。例如,在每一个新的版本中,重构之前 10%~15% 的遗留代码。

这些需求和考虑因素对于确保软件项目的长期成功、用户满意度和社会责任至关重要。它们通常需要跨学科团队的合作,包括开发人员、项目经理、质量保证工程师、运维团队和法律顾问等。

3.3 真实需求的获取

在软件工程领域,全面、准确的需求获取对于软件项目的成功起着决定性的关键作用,因为需求是源头,软件设计和实现是建立在需求之上。我们首先要做正确的事,就是要开发正确的软件,也就是要拿到正确的需求,所以需求获取作为软件开发过程中的关键环节,其重要性不容小觑。无论是新启动的软件项目,还是针对已有软件系统的升级改进,真实需求的获取都是必须攻克的首要难题。

在实际的软件开发场景下,需求的来源广泛且复杂,涵盖了用户期望、业务目标、技术限制等多个方面。由于各种因素的相互交织与影响,使得获取真实需求成为一项极具挑战性的任务。对于不同类型的软件项目,无论是从零开始的创新性项目,还是基于已有业务的优化项目,都需要一套行之有效的方法来确保需求的真实性和准确性。那么,在这样复杂的背景下,究竟怎样才能有效地获取真实需求呢?

本节将深入探讨如何系统地、科学地获取真实需求,确保软件开发过程的每一步都建立在坚实的需求基础之上。首先讨论一般情况下需求获取的过程与方法,然后就“已有的业务或市场、全新的产品”两种完全不同的情况做进一步的讨论,以帮助读者在复杂多变的环境中,准确捕捉用户的核心需求,从而避免软件研发偏离轨道,确保最终产品能够满足市场和用户的实际需要。

3.3.1 需求获取的过程

在软件开发过程中,需求获取是确保项目成功的关键步骤之一,然而,许多团队在实际操作中往往忽视这一环节,导致项目后期出现诸多问题。一般情况下,先获取需求,再进行整理与分析,有时一面获取、一面分析,发现获取不足,再补充需求。虽然这两者没有硬性隔离的界限,需求获取和需求分析两者相辅相成,但一般来说,需求获取是准备阶段,而需求分析与需求定义更紧密一些,需求定义是基于分析结果,明确地记录和描述具体的需求规格,我们会把需求分析和定义放在一起讨论,这里优先讨论“需求获取”。需求获取的重要性体现在以下几方面。

- **确保项目目标明确。**通过详细的需求获取,团队能够清晰地理解客户的期望和项目的具体目标。
- **降低项目风险。**未充分获取需求可能导致功能遗漏或误解,从而造成项目开发过程中需求变更频繁,增加项目的风险。
- **提高客户满意度。**深入的需求获取能够确保最终交付的产品真正满足客户的需求和期望,并增强客户的信任感。

尽管需求获取的重要性显而易见,但其常因为时间紧、沟通不畅、缺乏系统化工具和方法等原因而被忽视。下面就详细介绍一下需求获取的过程。但在介绍之前,也应该认识到,不同的项目背景,需求获取的过程、方法和工具有比较大的差别。例如,通常会考虑下列4种情况。

- **已有业务:**目标是优化和改进现有系统,应注重用户反馈与问题改进,通过实地调研获取具体需求。
- **开创性产品:**目标是探索全新功能或服务,重点是创意开发与用户痛点洞察,方法以创新为导向。
- **现有市场:**目标是满足行业标准或竞争产品的需求定位,注重市场竞争分析与用户细分需求的挖掘。
- **潜在市场:**目标是发掘潜在用户群或未满足的需求,需通过趋势分析、用户未知需求挖掘等方式获取需求。

1. 常见的需求获取过程

现在讨论一般情况下需求获取的常见过程——一个持续与用户、客户或潜在用户沟通的过程。在沟通过程中,将获取的需求信息记录下来。我们结合一个“Web 团队协作系统”案例来说明需求获取的常见过程与方法,它支持基于频道模式的团队沟通协作、轻量级任务看板、富文本的团队博客等功能需求。

(1) **明确产品愿景,理解核心问题。**在开发一款团队协作系统(如 TMS)之前,首先需要清楚产品的目标和愿景,明确客户希望解决的核心问题。例如,TMS 的目标是通过提供多样化的沟通方式(如频道沟通、任务看板、团队博文等),解决团队协作中的沟通障碍,提升

任务管理的效率,实现简单、有效的团队协作。

(2) **收集背景资料。**在需求获取前,通过互联网、行业报告、用户反馈等途径,收集相关领域的信息和业务背景。例如,了解当前市场上类似协作系统(如 Slack、Confluence)的功能特点、用户痛点,以及目标用户的 IT 使用习惯。这些资料有助于为后续的需求挖掘做好充分准备。

(3) **确定调查范围与用户角色。**明确产品可能影响的范围和目标用户群体,识别不同用户角色(如项目经理、开发人员、市场人员等)。例如,在 TMS 中,项目经理可能更关注任务看板和甘特图功能,而开发人员可能更需要 Markdown 支持和实时沟通功能。通过划分用户角色,为后续的调研和访谈提供针对性指导。

(4) **制订调研计划与访谈大纲。**制订详细的调研计划,包括需要了解的事项、目标用户、操作流程以及注意事项。例如,在调研 TMS 的需求时,可以设计访谈大纲,涵盖用户对频道沟通、任务管理、团队博文等功能的需求,以及他们在实际使用中的痛点和期望。

(5) **准备调研工具。**使用调研问卷、功能原型、逻辑图等工具,帮助更高效地收集和确认需求。例如,在调研 TMS 时,可以准备一个简单的原型界面,展示频道沟通、任务看板等核心功能,帮助用户更直观地表达需求。

(6) **开展调研与访谈。**通过实地调研、线上访谈等方式,深入了解用户的实际需求。例如,询问用户在团队协作中遇到的具体问题(如沟通效率低、任务分配混乱),以及他们目前的工作方式、业务场景和流程。记录用户对功能的期望,例如,是否需要实时通知、任务优先级管理等。

(7) **生成与修改调研分析报告。**根据调研记录,利用工具(如大语言模型)生成初步的调研报告,并结合实际情况进行修改和完善。例如,整理出用户对 TMS 的核心需求,如频道沟通的实时性、任务看板的可拖曳操作、博文协作的权限管理等,形成一份完整的需求文档,作为后续需求分析的输入。

(8) **持续沟通与反馈。**调研是一个反复迭代的过程,需要与项目干系人持续沟通,确保需求理解的准确性。例如,在 TMS 的开发过程中,定期与用户确认需求是否满足预期,调整调研方向,挖掘更多细节需求,如国际化翻译管理、移动端响应式设计等,确保产品功能更加贴合用户实际需求。

2. 不同场景的需求获取过程

(1) 已有业务的需求获取过程。

- 收集用户反馈:通过用户反馈、社区讨论、支持论坛等获取用户在使用过程中的真实体验和意见。
- 分析现有问题:利用数据分析工具,发现用户行为数据中存在的功能瓶颈或操作痛点。
- 开展访谈或实地观察:深入访谈实际使用系统的用户,观察他们的工作方式和使用过程中的障碍点。
- 整理需求优先级:将优化需求按照用户影响度、平台性能和商业价值等因素进行排序。

(2) 开创性产品的需求获取过程。

- 用户痛点洞察:通过深度访谈、陪同用户的行为观察,发现目前未被解决的问题和机会点。

- 头脑风暴与创意生成：组建跨领域团队，通过多轮创意思路碰撞，提炼出可能的创新方向。
- 验证和原型测试：快速制作低保真原型或功能演示产品，与潜在用户协作完成初步实验点的验证。
- 迭代开发：结合第一轮需求收集的结果，进一步优化和扩展创新产品的功能和服务细节。

(3) 现有市场的产品需求获取过程。

- 研究竞争产品：通过对竞争产品功能、定位、定价等的全面分析，了解行业痛点和标准功能。
- 用户群体细分与需求提取：明确目标市场的具体人群，通过问卷调查或座谈会获取细分群体的真实需求。
- 行业标准学习：了解行业法规或规范（如金融监管、医疗认证等），确保产品满足合规要求。
- 市场调研与功能差异化设计：通过调研找到现有市场中的未满足需求，结合用户反馈提出改进方案。

(4) 潜在市场的产品需求获取过程。

- 研究趋势与未来技术：收集行业趋势报告和技术预测，找到可能催生新市场需求的突破方向。
- 用户画像定义：确定潜在用户群体的特点（如年龄层、消费习惯、使用能力等），预测他们可能的需求。
- 从相邻领域获取灵感：通过分析相邻行业或不同场景下的需求，将已有场景的解决方案迁移过来。
- 制定假设与用户实验：基于发现的潜在需求，制定使用假设和商业模型原型，通过小规模实验验证可行性。

案例：TMS 需求获取过程

针对团队协作系统 TMS 的需求获取，可以遵循以下步骤系统地获取 TMS 团队协作系统的需求，并确保产品开发始终以用户为中心，满足市场和用户的实际需求。

(1) 理解产品定位和目标用户：首先，了解 TMS 的定位是一个基于频道模式的团队沟通协作系统，它结合了轻量级任务看板、团队博文 Wiki 以及国际化翻译管理等功能。

(2) 分析现有资料：根据提供的网页内容，分析 TMS 的主要功能和特点，包括团队沟通、博文系统、国际化翻译管理等。

(3) 识别关键功能和用户需求：识别 TMS 的关键功能，如团队协作沟通、频道任务看板、国际化翻译、实时通信等，并分析这些功能对应的用户需求。

(4) 市场调研和竞品分析：进行市场调研，了解同类产品的功能和市场表现，分析 TMS 相对于竞品的优势和不足。

(5) 用户访谈和问卷调查：设计问卷调查，收集更广泛的用户反馈；与 TMS 的现有用户或潜在用户进行访谈，了解他们的使用体验和改进建议。

(6) 观察用户行为：观察用户如何在实际工作中使用 TMS, 注意他们在使用过程中遇到的问题和不便之处。

(7) 原型测试和用户反馈：基于当前的 TMS 版本, 设计原型并进行用户测试, 收集用户对界面设计和功能实现的反馈。

(8) 分析用户反馈：对收集到的用户反馈进行归类和分析, 识别最紧迫和最重要的需求。

(9) 需求优先级排序：根据需求的重要性和紧迫性, 对需求进行优先级排序, 确定开发顺序。

(10) 撰写需求文档：将分析结果整理成需求文档, 明确每个需求的详细描述、优先级和预期影响。

(11) 与开发团队沟通：与开发团队进行沟通, 确保他们对需求有清晰的理解, 并讨论实现的可行性。

(12) 持续迭代和优化：根据用户反馈和市场变化, 持续迭代和优化 TMS 的功能, 确保产品始终满足用户需求。

(13) 利用数据分析：收集和分析用户使用 TMS 的数据, 如活跃度、功能使用频率等, 以数据支持需求决策。

(14) 合规性和版权问题：考虑到 TMS 可能用于商业用途, 需要关注并解决第三方开源依赖库的版权授权问题。

3.3.2 传统的需求获取方法

需求获取方法多种多样, 我们可以通过多样化的市场调研、深入的专题讨论、系统的信息搜索等方式来捕捉和定义需求。这些方法不仅包括收集翔实的市场信息, 还涵盖对数据的深入分析和洞察, 从而帮助企业全面了解当前的市场需求、竞争环境以及潜在的商业机会。通过结合定量与定性的研究手段, 企业能够精准识别目标用户的痛点和期望, 评估竞争对手的优势与劣势, 并发现未被满足的市场空白。这一过程不仅优化了产品或服务的定位, 还为战略决策提供了坚实的依据, 使企业能够在动态变化的市场中保持竞争力并实现持续增长。

需求获取方法很多, 如问卷调查、深度访谈、焦点小组、用户观察工作坊、原型设计、文档/报表分析、竞争对手分析、市场趋势分析、头脑风暴等。下面就侧重介绍问卷调查、深度访谈、焦点小组这三种方法, 其他方法, 读者可以举一反三, 或在网上搜索相关资料或问大模型。

1. 问卷调查

问卷调查是一种结构化的需求获取方法, 通过设计一系列有针对性的问题, 向目标用户群体收集定量和定性数据。这种方法广泛应用于市场调研、用户满意度评估、新产品开发等领域, 具有以下特点。

- 高效覆盖：能够在较短时间内收集大量用户反馈, 适合大规模数据收集。
- 成本效益：相对于深度访谈或焦点小组, 问卷调查的实施成本较低。
- 数据可量化：选择题、评分题等形式, 便于后续的数据分析和统计。

- 匿名性强：用户可以匿名填写，减少社会期望效应，获取更真实的反馈。

一般应用操作步骤如下。

(1) 目标确定：明确调查目的和需要获取的信息类型，例如，用户对现有协作功能的满意度、新功能需求、使用体验中的痛点等。

(2) 问卷设计：制定问题(如行业和团队规模、被调查者的角色、功能使用情况、用户体验、期望得到什么新功能等)，确保问题简明、具有针对性，涵盖多种题型(如选择题、开放式问题)。

(3) 样本选择：确定调查对象，确保样本具有代表性。确保问卷覆盖不同类型的用户和不同角色，以及来自不同规模和行业的团队，以获取多元化的反馈。

(4) 问卷分发：通过线上平台(如腾讯问卷、问卷星、金数据等)或线下方式发送问卷。

(5) 数据收集与分析：收集问卷回复，使用统计工具进行数据分析，提取关键需求和趋势。

(6) 结果应用：根据分析结果，制定相应的产品改进策略或新功能规划。

通过问卷调查，不仅能够高效地收集广大用户的反馈，还能精准地识别出用户的真实需求和期望。问卷调查作为需求获取的重要方法，应与其他方法(如用户访谈、焦点小组等)结合使用，以实现全面、深度的需求理解。

2. 深度访谈

深度访谈是一种质性需求获取方法，通过与目标用户进行一对一的深入对话，深入了解其需求、动机、体验和困难。这种方法广泛应用于用户研究、产品开发和市场分析，具有以下特点。

- 深入理解：能够挖掘用户的潜在需求和隐性痛点，获取丰富的背景信息。
- 灵活性高：访谈过程可以根据用户的回答灵活调整，探索更多相关话题。
- 建立信任：面对面的交流有助于建立信任关系，促使用户更开放地分享真实想法。
- 质性数据：获取详细的描述性信息，有助于形成全面的需求画像。

一般应用操作过程，分为准备阶段、执行阶段和分析阶段，其中，准备阶段的操作步骤如下。

(1) 明确目标：确定访谈的具体目标，如了解用户对现有功能的满意度或探索新的需求。

(2) 设计提纲：制定访谈提纲，包含开放性问题，确保覆盖关键话题。

(3) 选择受访者：挑选具有代表性的用户群体，确保样本多样性和代表性。

执行阶段操作步骤如下。

(1) 建立联系：通过邮件、电话等方式联系受访者，安排访谈时间和地点，尽量选择舒适的环境。

(2) 进行访谈：按照提纲引导访谈，保持开放和倾听，关注用户的每一个细节，记录重要信息(可录音以备后续分析)，确保信息的全面性和准确性。

(3) 灵活应变：根据访谈进展，适时调整问题深度和方向，深入探讨关键点。

分析阶段操作步骤如下。

(1) 整理数据：转录访谈记录，整理关键信息和共性需求。

(2) 提炼洞察：分析用户反馈，识别主要需求、痛点和改进建议。

(3) 形成报告: 编写需求分析报告, 提供具体的改进方案和行动建议。

根据前面“问卷调查”, 我们比较清楚目标、受访者、整理数据等要求, 所以这里以“TMS 团队协同系统”为例, 主要介绍如何更好地设计访谈提纲, 把它分为 5 部分, 每部分给出两三个问题示例。

(1) 基础使用情况。

- 您通常如何使用 TMS 的团队沟通功能?
- 您喜欢用什么在线方式进行沟通?
- 哪些功能是您日常工作中最常用的?

(2) 用户体验。

- 在使用 TMS 过程中, 您遇到过哪些问题或不便?
- 对于任务看板的拖曳功能, 您有何改进建议?

(3) 功能需求。

- 您希望 TMS 新增哪些协作工具或功能?
- 您认为 TMS 的哪些功能最有助于提升团队协作效率?
- 对于国际化翻译管理, 您有哪些具体的需求或期望?

(4) 整体反馈。

- 您对 TMS 的整体使用体验有何意见或建议?
- 请分享一个您希望通过 TMS 解决的具体团队协作问题。

(5) 开放问题。

- 这类工具, 您之前用过哪些? 对哪个工具印象深刻?
- 您对 TMS 整体使用体验有何意见或建议?
- 请分享一个您希望通过 TMS 解决的具体团队协作问题。

3. 焦点小组

焦点小组(Focus Groups)是一种定性研究方法, 通过组织一小群具有代表性的参与者, 在主持人的引导下进行讨论, 以深入了解他们的观点、需求和体验。这种方法广泛应用于市场调研、用户体验评估和产品开发等领域, 具有以下特点。

- 互动性强: 通过小组讨论, 激发参与者之间的互动, 挖掘更深层次的见解。
- 多样化观点: 集合不同背景和角色的参与者, 获取多元化的意见和建议。
- 灵活性强: 主持人可以根据讨论进展, 灵活调整话题和提问方向。
- 即时反馈: 能够迅速捕捉参与者的即时反应和情感态度。

一般应用操作步骤如下。

(1) 确定目标。明确焦点小组的研究目标, 如了解用户对现有功能的满意度、探索潜在需求(例如, 探索用户在协作过程中遇到的挑战和未被满足的需求)或评估新功能的可行性, 如测试用户对拟引入的“思维导图”新功能的兴趣。

(2) 设计讨论提纲。根据已制定的目标来设计详细的讨论提纲, 涵盖关键话题和具体问题, 确保讨论的方向和深度。

(3) 招募参与者。选择具有代表性的用户群体(如项目经理、开发人员、设计师和内容编辑等), 确保样本的多样性和相关性。通常每组包含 6~10 名参与者。

(4) 组织和执行。安排合适的时间和地点, 确保讨论环境舒适。由专业主持人引导讨

论,鼓励所有参与者积极发言,覆盖所有关键话题,记录关键信息(可进行录音和录像以便后续分析)。

(5) 分析与总结。对讨论记录进行整理和分析,识别出共性需求、主要痛点和创新建议,形成需求分析报告。

通过焦点小组,TMS团队能够深入了解不同用户群体的真实需求和使用体验,发现潜在的改进机会和创新方向。这不仅有助于优化现有功能,提升用户满意度,还能指导新功能的开发,增强TMS在团队协作工具市场中的竞争力。结合其他需求获取方法,如问卷调查和深度访谈,焦点小组将为TMS提供全面、翔实的用户洞察,确保产品在不断变化的市场环境中持续成长和进步。

3.3.3 针对新产品的其他方法

在新产品开发过程中,除了问卷调查、用户访谈(包括与行业专家、知名博主、科技评论员等交流讨论)和焦点小组等常见的需求获取方法之外,还会采用一些其他的方法,如趋势分析与研究、用户洞察方法等,这样能够全面捕捉市场动态与用户需求,确保我们能够理解用户的潜在需求或用户需求的变化趋势等,从而创建符合市场趋势的新产品。

1. 趋势分析与研究

趋势分析与研究通过收集和分析市场、技术、社会经济等方面的数据,识别当前和未来的市场动向,预测行业发展方向。在新产品开发过程中,趋势分析是需求获取的重要方法之一。通过识别和分析相关趋势,企业能够洞察市场动态、用户需求和科技进步,从而制定有效的产品策略,为新产品的定位、新功能的设计提供科学依据。趋势分析可以包括技术趋势、行业与市场趋势、社会文化趋势、经济趋势、社会心理趋势、环境趋势、人口变化趋势等。这里讨论对新产品更有影响的前三个趋势。

技术是推动新产品创新的核心力量。新兴技术(如人工智能、物联网、区块链技术、虚拟现实与增强现实等)的出现和发展不仅改变了产品的功能和性能,还影响了用户的使用习惯和期望,我们从中可以获取灵感和需求方向。例如,根据一份关于物联网未来发展趋势的报告,发现物联网在医疗健康领域有巨大的应用潜力,进而思考开发一款基于物联网的远程健康监测设备,满足对老年人进行实时健康管理的潜在需求。从需求获取角度看,其价值体现在以下几方面。

- 功能创新:利用新技术增强产品功能,提高用户体验。
- 效率提升:通过技术优化产品制造和运营流程,降低成本。
- 差异化竞争:采用独特技术打造差异化产品,提升市场竞争力。

行业与市场趋势,更是我们关注的另一方面。了解行业动态和市场竞争状况是新产品成功的关键。市场规模、增长率、竞争格局等因素直接影响产品的定位和战略制定,例如,我们会根据市场需求和竞争情况确定产品的定位和目标用户。在这方面,我们会注重下面4方面。

- 市场规模与增长率:评估目标市场的潜力和发展速度。
- 竞争格局:分析主要竞争对手的产品特点、市场份额和策略。
- 客户需求变化:识别用户需求的演变和新兴需求点。
- 创新与技术应用:关注行业内的新技术应用和创新动向。

社会文化趋势反映了人们的价值观、生活方式和消费习惯的变化,这些变化直接影响产品的设计和市场接受度。我们会分析社会文化的变化对人们需求的影响,关注不同年龄群体的价值观和生活方式的变化,来细分市场,精准定位产品功能。例如,以一款音乐播放软件为例,为满足年轻用户的需求,可以设计用户自定义播放界面、歌曲分享到社交平台等功能。在这方面,我们会观察以下几方面。

- 消费心理:品牌忠诚度、购买动机、消费习惯的变化。
- 生活方式变化:健康生活、工作与生活平衡、数字化生活等趋势。
- 社会价值观:多样性与包容性、伦理消费、社会责任感的提升。
- 心理健康关注:对心理健康产品和服务的需求增加。

2. 用户洞察

用户洞察方法通过深入理解用户行为、需求和偏好,确保新产品贴合目标用户的实际需求,提高用户满意度和市场接受度。主要方法包括建立用户画像、情景模拟与角色扮演、用户体验旅程地图等,下面侧重讨论用户画像、情景模拟与角色扮演。

(1) **建立用户画像(User Personas)**。用户画像是对目标用户群体的虚拟代表描述,包含用户的基本信息、行为模式、需求和痛点。提供明确的目标用户视角,有助于产品设计和营销策略制定。对新产品,可以通过假设和推测来构建潜在用户的画像。例如,设想开发一款低空飞行汽车预订应用,其潜在用户可能包括商务出行者、旅游爱好者等。其中,商务出行者可能更关注出行的效率和舒适性,希望能够快速预订到适合商务会议安排的飞行时间和座位;旅游爱好者则更注重飞行过程中的观光体验等。构建用户画像时,我们会仔细筛选用户特征,如年龄段、区域、文化水平、兴趣爱好、消费习惯、消费能力等,识别用户的共性特征和差异,细分用户群体,从而确定某一款应用更适合具有哪些特征的潜在用户(即哪一类或几类用户群体)。例如,开发一款新型的虚拟现实游戏设备,目标用户可能是年龄为18~35岁、喜欢科技和游戏、具有一定消费能力的人群。

(2) **情景模拟与角色扮演**。通过模拟用户在特定情境下的操作和决策,深入理解用户在实际使用过程中的需求和挑战。我们可以扮演潜在用户的不同角色,在各种可能的场景下使用产品,提高团队对用户需求的共情能力。例如,开发一款无人零售商店的智能购物系统,团队成员可以模拟顾客从进入商店、挑选商品、支付结算到离开商店的整个购物过程,体验过程中可能会发现顾客在商品查找、价格比较、支付方式选择等方面的需求和痛点。邀请外部人员参与情景模拟,获取更客观的反馈。例如,在开发一款新型的健康管理设备时,先基于用户画像和市场趋势,设计典型使用场景,然后邀请不同年龄段、不同健康状况的人员参与情景模拟,观察他们在使用设备过程中的行为和反应,收集他们对设备功能、操作界面、数据展示等方面的意见和建议。

(3) **用户体验旅程地图**。用户体验旅程地图是一种可视化工具,用于描绘用户在使用产品或服务过程中经历的各个阶段和触点。它不仅展示用户的具体操作步骤,还包括用户在每个阶段的情感变化、需求和潜在痛点,如图3-4所示。通过这种方式,团队可以全面了解用户的体验,识别改进机会,优化产品设计和流程。其主要环节如下。

- 事先通过用户访谈、问卷调查、现场观察等方式获取用户角色、目标、动机、行为、痛点等相关的信息。
- 识别关键阶段:划分用户与产品或服务交互的主要阶段,如认知、考虑、购买、使用、

支持等。

- 列出每个阶段的触点：确定用户在每个阶段与产品或服务接触的具体点，例如，访问网站、联系客服、使用功能等。
- 描绘用户行为和情感，记录用户在每个阶段的具体操作和决策过程。描绘用户在各阶段的情感状态，如兴奋、困惑、满意、不满等，帮助理解用户体验的高低点。
- 识别痛点与挖掘机会，如通过分析用户在旅程中的困难和不满，识别需要改进的环节，并基于用户需求和期望，找到潜在的改进和创新机会，提升用户体验。
- 最后可视化旅程地图，使用图表软件或专门的旅程地图工具，清晰展示各阶段、触点、用户行为和情感，包括用户角色、关键数据支持和具体建议。

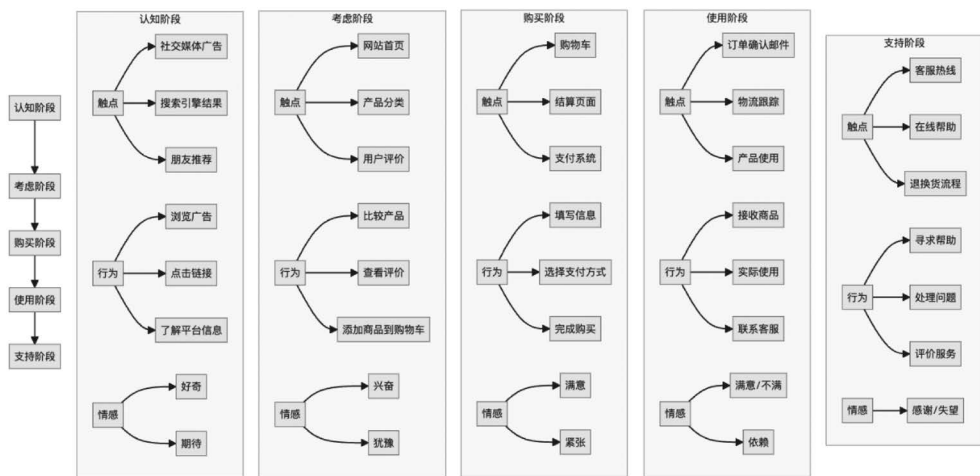


图 3-4 由大模型生成的分阶段用户体验旅程地图

3. 创意激发与头脑风暴

(1) **组织跨领域头脑风暴。**邀请来自不同背景的成员参与头脑风暴，如技术人员、设计师和市场营销专家。不同领域的专业人士能够带来多样化的视角和思维方式，激发创新性的需求想法。例如，在开发一款融合艺术与科技的产品时，可以邀请艺术家、工程师和营销专家共同参与。艺术家能够从审美和创意角度提出产品的外观设计和用户体验建议；工程师则关注技术实现的可行性和潜在挑战；营销专家从市场需求和用户接受度出发进行分析。通过多方协作，共同为产品需求定义提供丰富的创意和思路。

(2) **引入外部创意资源。**与其他行业的企业或创新团队进行合作，共同开展头脑风暴。例如，在开发一款面向教育领域的创新产品时，可以与游戏开发公司合作，借鉴游戏中的激励机制和互动方式，应用到教育产品中，以提升学生在学习过程中的趣味性和参与度。通过跨行业的合作，能够获得新的灵感和需求启示，推动产品创新。

(3) **运用类比法寻找灵感。**参考其他领域的成功产品，寻找灵感和需求启发。例如，在开发一款新型家庭娱乐设备时，可以借鉴智能手机的发展历程，思考如何将智能手机的便捷性、个性化和社交功能应用到家庭娱乐领域，以满足用户在家庭环境中的多样化娱乐需求。通过类比不同领域的成功经验，能够发现新的需求和创新点。

(4) **应用逆向思维法进行创新。**从现有产品或解决方案的不足出发，反向推导新的需求和创新方向。例如，传统健身设备通常体积庞大且功能单一。通过逆向思维，可以考虑开

发一款小巧便携、功能集成且智能化的健身产品,满足用户在不同场景下的健身需求,同时克服传统设备的缺点。逆向思维有助于发现现有产品的痛点,并基于此进行突破性创新。

3.3.4 针对已有产品的其他方法

前两节讨论的方法,如问卷调查、用户访谈、焦点小组、趋势分析、头脑风暴等,许多时候都是可以应用在已有产品的需求获取中,而且调查对象、访谈对象更清楚,甚至和用户建立了比较好的关系,这些工作更容易开展。对已有产品,我们可以去用户那里观察他们是如何使用软件的,并从中发现问题,通过交流,获取用户更多的期望;有了用户行为数据、业务交易数据,更适合采用数据分析方法。

1. 现场观察

在需求工程中,“现场观察”(On-site Observation)是一种有效的需求获取方法,特别适用于理解用户在实际使用环境中的行为和需求。现场观察是指需求工程师亲自前往用户的实际使用环境,直接观察用户如何与软件互动,记录用户的操作流程、遇到的问题以及使用习惯。通过这种方式,能够深入理解用户的真实需求和工作流程,从而发现目前软件产品中潜在的问题和改进机会。

- 真实了解用户行为:亲自观察用户在自然环境中的操作,能够获取未经修饰的真实数据,避免用户在回忆或描述时的偏差。
- 发现隐性需求:通过观察,能够识别用户未明确表达但实际存在的需求,如操作中的不便、潜在的功能需求等。
- 深入理解应用场景:了解用户在具体情境下使用软件的方式,有助于设计更符合实际需求的功能和界面。
- 促进与用户的深度交流:现场观察过程中,通过深度的交流,可以快速澄清疑问,获取更多细节信息。

在这个过程中,我们注意与用户讨论观察的结果,验证发现的问题和需求,确保理解的准确性。通过深入的讨论,可以进一步了解用户的期望和偏好,获取更多有价值的信息。同时,也要保持开放心态,避免预设假设,开放地接受和理解用户的真实需求和反馈。

2. 用户行为数据分析

用户行为数据包括两部分,后端是系统存储的日志数据,前端是跟踪用户操作软件界面的数据,可以针对两部分的数据分析,更客观、更科学地了解用户行为。

(1) 分析软件系统的日志记录,深入剖析用户操作行为。例如,在视频网站中,通过细致分析用户的播放记录、暂停次数、快进快退行为等,可以全面了解用户对不同类型视频的观看习惯和偏好。如果发现某类视频的暂停次数显著增加,可能表明该视频在某些情节设置或播放质量方面存在问题,需要针对性地进行优化,如改善视频内容质量或调整播放技术。

(2) 评估用户在页面上的停留时间与点击热点。以新闻资讯类应用为例,利用热图分析技术,可以明确用户在不同页面的停留时长、点击频率高的区域以及关注度较低的板块。如果某个频道的页面停留时间较短且点击率不高,这可能意味着该频道的内容推荐机制或页面布局存在不足,需重新审视内容展示方式或优化页面结构,以提升用户的阅读体验和参与度。

3. 业务交易数据分析

下面通过两个行业(电商业务、金融业务)的具体情况讨论,来帮助读者更好地理解基于业务交易数据分析获取已有系统的需求。

(1) 针对电商业务,深入分析订单数据及购物车遗弃率。例如,若在某个特定时间段内,购物车遗弃率突然上升,需迅速调查潜在原因。这可能包括支付流程的复杂性、商品价格的频繁波动或页面加载速度过慢等问题。基于分析结果,可以提出有针对性的优化需求,如简化支付步骤、稳定商品定价策略或提升网站性能,从而降低遗弃率,提升转化率。

(2) 在金融业务中,详尽分析投资交易数据及资金流向。以股票交易软件为例,若发现用户在特定时间段内频繁进行某类股票的交易操作,需深入了解其背后的投资策略和具体需求。这可能涉及增加高级投资分析工具、提供实时资讯服务或优化用户交互界面,以更好地满足用户的投资需求,提升软件的竞争力和用户黏性。

3.3.5 智能需求获取方法

前面已经掌握了过去常用的需求获取方法,这些方法其实也被大模型掌握了,只是大模型不能去现场和用户、专家、意见领袖等进行直接交流,但我们可以代它去执行,即在去之前,先让大模型帮我们设计一个访谈大纲、设计需要提问的问题,甚至随身带着类似 ChatGPT 或 DeepSeek 那样的 App,让这类 App 协助我们更好地完成访谈。

过去,我们采用了诸如机器学习、社会网络分析和反馈机制等技术,从开源社区的问题报告和聊天平台中挖掘隐秘的用户需求。甚至使用基于深度多任务学习的方法和基于双向 LSTM 结构的上下文感知对话模型,来提升需求获取的能力。这些探索在某些应用场景中是非常有意义的。然而,随着大模型技术的发展,现在可以通检索增强技术(RAG)和智能体,向基于大模型的工具或系统提出明确的需求,系统就会搜索相关网络资源(包括企业内部网络、知识库、文档库等)并实时整理出所需内容反馈给我们,帮助我们挖掘用户需求。

回到团队协作系统 TMS,让大模型帮忙收集或获取需求,例如,让 DeepSeek 尽可能先搜索网络资源,然后整理出一份需求列表,如下所示。



观看视频

一、团队沟通业务需求

1. 支持多种沟通场景

- 团队成员需要能够通过即时消息(如私聊、群聊)快速沟通。
- 支持异步沟通(如留言、通知),避免打断工作流。

2. 按项目或主题组织沟通

- 团队需要能够创建不同的沟通频道,按项目、部门或主题分类。
- 支持在频道内发起话题讨论,避免信息混乱。

3. 确保重要信息触达

- 重要消息需要能够通过@功能或特殊标记提醒相关人员。
- 支持消息的收藏和固定,便于后续查找。

4. 文件共享与协作

- 团队成员需要能够快速上传、下载和预览文件(如图片、文档)。
- 支持对共享文件进行分类管理和版本控制。

5. 通知与提醒机制

- 团队成员需要能够收到任务、会议或重要消息的提醒。
- 支持自定义通知方式(如桌面通知、邮件通知)。

二、轻量级任务管理业务需求

1. 任务创建与分配

- 团队成员需要能够快速创建任务,并指定负责人和参与人。
- 任务需要包含标题、描述、截止日期、优先级等基本信息。

2. 任务状态与进度跟踪

- 团队成员需要能够查看任务的状态(如待办、进行中、已完成)。
- 团队需要能够直观了解整体任务进度,避免任务延误。

3. 任务依赖与优先级管理

- 团队需要能够设置任务之间的依赖关系,确保任务按顺序执行。
- 支持根据任务优先级(如高、中、低)进行排序和分配资源。

4. 任务提醒与自动化

- 团队成员需要能够收到任务截止提醒,避免错过截止日期。
- 支持重复任务设置,减少重复性工作的手动操作。

三、博文 Wiki 业务需求

1. 博文创建与编辑

- 团队成员需要能够快速创建和编辑博文,支持富文本和 Markdown 格式。
- 支持图片、表格等多媒体内容的插入和排版。

2. 博文组织与权限控制

- 团队需要能够按项目或主题对博文进行分类管理。
- 支持设置博文的访问权限,确保敏感信息仅对特定成员可见。

3. 博文版本与协作

- 团队成员需要能够查看博文的修改历史,支持版本回滚。
- 支持多人同时编辑博文,避免内容冲突。

4. 博文评论与互动

- 团队成员需要能够对博文进行评论和点赞,促进知识分享和讨论。
- 支持将博文导出为 PDF 或 Markdown 文件,便于外部使用。

四、国际化与翻译管理业务需求

1. 翻译项目管理

- 团队需要能够创建翻译项目,管理多语言内容。
- 支持导入和导出翻译内容,便于与外部工具协作。

2. 翻译协作与审核

- 团队成员需要能够协作翻译内容,支持多人同时编辑。
- 支持翻译内容的审核流程,确保翻译质量。

五、系统管理与扩展业务需求

1. 用户与权限管理

- 团队需要能够根据角色(如管理员、普通用户)分配权限。
- 支持细粒度的权限控制,确保数据安全和隐私。

2. 系统集成与扩展

- 团队需要能够与第三方工具(如 GitHub、Jenkins)集成,实现数据同步。
- 支持通过 API 或 Webhook 实现自动化流程。

假如我们更关心“二、轻量级任务管理业务需求”中的“任务状态与进度跟踪”,可以再引导或追问下去,获得这方面更详细的业务需求,如图 3-5 所示。

针对“二、轻量级任务管理业务需求”中“任务状态与进度跟踪”,我们想获得更细致的业务需求,包括考虑当今的工作环境和开发人员的体验。请好好挖掘一下“任务状态与进度跟踪”业务需求。



已搜索到 47 个网页

针对“任务状态与进度跟踪”这一业务需求,结合当今的工作环境(如远程办公、跨时区协作、敏捷开发等)以及开发人员的体验,以下是更细致的业务需求分析:

1. 任务状态的可视化与实时更新

- 需求背景: 在远程办公和跨时区协作中,团队成员需要实时了解任务状态,避免信息滞后或误解。
- 具体需求:
 - 提供多种视图(如看板、甘特图、日历、列表等),满足不同用户的偏好和项目需求 1 4。
 - 支持任务状态的实时更新和同步,确保所有团队成员看到的信息一致 7 10。
 - 允许自定义任务状态(如“待办”“进行中”“已完成”),并支持状态流转规则的设置 1 4。

2. 任务进度的透明化与跟踪

- 需求背景: 项目管理者需要清晰了解整体进度,开发人员需要明确自己的任务优先级和截止时间。
- 具体需求:
 - 提供进度条或百分比显示,直观展示任务的完成情况 2 7。
 - 支持任务依赖关系的设置,确保前置任务完成后,后续任务才能开始 6 10。
 - 提供任务时间预估与实际耗时的对比分析,帮助团队优化资源分配 9 13。

3. 自动化提醒与通知

- 需求背景: 在快节奏的工作环境中,开发人员容易忽略任务更新或截止时间。
- 具体需求:
 - 支持任务截止日期的自动提醒(如邮件、桌面通知、移动端推送) 7 11。
 - 提供任务状态变更的实时通知(如任务被分配、完成或阻塞) 4 10。
 - 允许自定义提醒规则(如提前1天提醒、每日任务汇总等) 1 6。

4. 任务阻塞与问题管理

- 需求背景: 开发过程中常遇到任务阻塞或技术问题,需要快速解决以避免项目延误。
- 具体需求:
 - 提供任务阻塞标记功能,明确任务无法推进的原因(如依赖未完成、技术问题等) 7 13。
 - 支持问题跟踪和解决流程,确保阻塞任务能够快速分配给相关人员处理 9 11。
 - 提供阻塞任务的优先级调整功能,确保关键任务优先解决 4 10。

图 3-5 LLM 进一步获取业务需求的一个示例

5. 数据分析与报告

- 需求背景：项目管理者需要通过数据分析优化团队效率，开发人员需要了解自己的工作表现。
- 具体需求：
 - 提供任务完成率的统计报告，帮助管理者评估团队效率 2 7。
 - 支持个人工作量的可视化分析（如任务数量、耗时等），帮助开发人员优化时间管理 9 13。
 - 提供项目健康度报告（如任务延期率、阻塞任务数量等），帮助管理者及时调整项目计划 6 10。

6. 跨平台与移动端支持

- 需求背景：开发人员需要在不同设备上查看和更新任务状态，尤其是在移动办公场景中。
- 具体需求：
 - 提供跨平台支持（Web、桌面、移动端），确保任务状态和进度实时同步 1 4。
 - 支持离线模式下的任务更新，网络恢复后自动同步数据 7 11。
 - 提供移动端的快捷操作（如快速标记任务完成、添加评论等），提升开发人员的操作效率 9 13。

7. 任务历史与版本管理

- 需求背景：开发人员需要追溯任务的历史变更，管理者需要了解任务的完整生命周期。
- 具体需求：
 - 提供任务变更的历史记录（如状态变更、负责人变更、评论更新等） 4 10。
 - 支持任务版本的创建和回滚功能，确保任务信息可追溯 7 13。
 - 提供任务变更的审计日志，帮助管理者了解任务的动态变化 9 11。

8. 集成与扩展性

- 需求背景：开发人员通常使用多种工具（如GitHub、Jenkins、Slack等），需要任务管理系统与其他工具无缝集成。
- 具体需求：
 - 支持与代码托管平台（如GitHub、GitLab）集成，自动同步任务状态与代码提交 7 13。
 - 提供API接口，支持自定义集成和自动化流程 4 10。
 - 支持与沟通工具（如Slack、Microsoft Teams）集成，实时推送任务更新 1 6。

图 3-5 （续）

通过人机交互，可以收集足够的待建系统的业务需求。这依赖我们和大模型交互的能力，所幸的是，今天大模型有思考、推理能力，对提示词要求没有那么高。万一第一次交互效果不好，可以多次尝试，可以获得更好的输出。

3.4 智能需求分析与定义

需求分析与定义是软件开发生命周期中至关重要的环节，它不仅是需求获取阶段的自然延伸，更是去伪存真的过程，将零散、模糊的用户需求转换为清晰、规范、真实的需求的关键过程。通过系统化、结构化或面向对象的分析方法，需求分析与定义能够有效整合多方信息，梳理复杂需求，确保所有利益相关者对项目目标、功能范围及优先级达成一致共识。

在需求分析与定义过程中，团队可以借助多种工具和方法来提升分析的深度与广度。例如，用例图(Use Case Diagram)能够直观地展示系统功能与用户交互；活动图(Activity Diagram)可以清晰地描述业务流程和逻辑；场景分析(Scenario Analysis)则通过具体的使用场景挖掘用户的潜在需求。此外，数据流图(DFD)、实体关系图(ERD)等工具也能帮助

团队从不同维度识别关键业务数据的结构和关系。

本节将详细介绍需求分析与定义的核心方法与步骤,结合实际项目中的应用实例,深入探讨如何通过科学的需求分析提升需求的质量,避免因需求不明确或变更频繁导致的开发延误和成本超支,确保开发成果能够真正满足用户期望和业务目标。

3.4.1 去伪存真

谈到“挖掘真实的需求”,常常会提及“X-Y 问题”(X-Y problems, 见 <https://xyproblem.info>),即如果我们(特别是产品经理)不会提问,用户其实要解决问题 X,但往往用户给出了其中的一个解决方案 Y。然后我们把 Y 当作需求、把手段当目的,做出的产品是次品——不能真正解决问题,项目失败;或者在研发过程中,经过大量的交互和浪费的时间,用户最终清楚地知道,他们确实要解决 X 这样的问题,而之前提出的 Y 甚至不是适合 X 的解决方案。

出现 X-Y 问题,往往是因为缺少思考、缺少推敲,缺少再追问一句“为什么要这么做”。有些人不清楚想解决的用户需求是什么,而认为开发 Y 功能能够满足用户就直接去做 Y 的需求,不是推敲解决用户需求的 X 问题是什么。

作者常常喜欢举一个简单的例子,某个同学打球回来,往宿舍走,路上碰到另一个同学,对他说:“帮我从食堂带两个馒头”。那“带两个馒头”是这位打球同学的需求吗?其实是这位同学给出了解决方案,需求是什么?有可能是:

R1: 肚子饿了,需要充饥、填饱肚子。

R2: 肚子饿了,且要省钱。

R3: 好几天没吃到馒头,特别想吃馒头。

对于上面三个需求,其实解决问题的方案是不一样的。如果是 R1,去食堂,什么东西好带就买什么,因为都可以充饥。如果是 R2,尽量买和馒头价格相近的东西,贵的食物就不能买。如果是 R3,买其他食品都不能解决问题,只能买馒头。所以,当那位同学说,“帮我从食堂带两个馒头”,可以反问“为什么要带馒头?”或问“只能带馒头吗?”“还想吃什么?”等。

如何避免 X-Y 问题,而挖掘出真实的需求?这就要求我们不要猜测,而是善于提问、循循诱导。

- 多问几个为什么?
- 要解决这个问题是什么原因是什么?
- 现在的情况怎样? 希望得到怎样的结果?

让用户把真实需求说出来,尽可能地还原真正的需求,了解问题所在、用户的真实的内心期望。从逻辑维度(从上往下)和时间维度(从前往后)分别思考问题,甚至有更多角度来分析用户的回答,尽可能地找到正确的问题,做到有的放矢。

- **从上往下思考。**所提出的问题属于哪个领域? 这个领域最大的问题是什么? 会有哪些子问题? 对同一个问题会有不同的解决方案吗? 哪个方案更能彻底解决问题? 能解决本质问题吗? 还是各个击破?
- **从前往后思考。**这个问题是什么时候出现的? 问题的背景是什么? 之前为何没解决? 如果不解决,未来会发生什么? 什么时候解决更为合适?

3.4.2 结构化分析方法

结构化分析(Structured Analysis,SA)是软件工程中一种经典的需求分析方法,旨在通过系统化的建模技术,将复杂系统分解为更小、更易管理的部分(如模块、组件等),从而明确系统的功能、数据流和行为。以下是结构化分析方法的详细介绍。

结构化分析方法的核心思想是自顶向下、逐层分解。通过将复杂系统分解为多个层次的功能模块,逐步细化需求,直到每个模块的功能足够简单、易于实现。这种方法强调逻辑建模,而非物理实现,确保需求分析的独立性和通用性。结构化分析方法依赖于多种图形化工具,主要包括以下三种模型。

- **功能模型：数据流图(Data Flow Diagram,DFD)**,用于描述系统中数据的流动和处理过程,展示数据如何从输入经过一系列加工转换为输出。
- **行为模型：状态转换图(State Transition Diagram,STD)**,用于描述系统对外部事件的响应,展示系统状态的变化及触发条件。
- **数据模型：实体关系图(Entity Relation Diagram,ERD)**,用于描述系统中的数据实体及其之间的关系,适用于数据库设计。

基于这三个模型,可以专业地完成需求的结构化分析,其关键步骤如下。

(1) 需求获取与整理: 通过与用户交流、观察工作流程等方式获取原始需求,将需求分类整理为功能需求和非功能需求。

(2) 完成功能建模、行为建模、数据建模(下面会详细讨论)。

(3) 数据字典与加工规格说明,即定义系统中所有数据元素的属性(如名称、类型、取值范围),并描述每个加工的逻辑规则(如决策表、决策树)。

结构化分析的主要优点如下。

- **清晰直观**: 图形化工具(如 DFD、ERD)使系统功能和数据流易于理解。
- **模块化设计**: 通过逐层分解,将复杂系统简化为可管理的模块。
- **逻辑性强**: 强调逻辑建模,避免过早涉及技术实现细节。

但也存在着一定的局限性。

- **灵活性不足**: 适用于需求稳定的系统,难以应对频繁变更的需求。
- **迭代支持有限**: 传统结构化分析缺乏对迭代开发的支持。
- **实时系统适用性低**: 更适合数据处理系统,对实时控制系统的支持较弱。

目前主要采用快速迭代的方法,所以结构化方法受到的限制还比较大,这样其应用就比较少,在这里做比较简单的介绍。更详细的内容,可以参考其他的资料,或直接问大模型。

1. 数据流图

数据流图(DFD)是一种图形化技术,用于描述信息流和数据从输入到输出的过程中所经历的变换,分为描述系统与外部实体的交互的环境图(顶层 DFD)和展示系统内部的数据处理流程的分层 DFD。它不涉及具体的物理实现细节,而是专注于系统的逻辑功能。DFD 帮助开发团队和利益相关者直观地理解系统的功能、数据交互以及信息流动路径。其核心元素主要有以下几个。

- **外部实体(External Entity)**: 表示与系统交互的外部对象,如用户、其他系统或设备。通常用矩形表示。

- **加工/处理(Process)**: 对数据进行处理逻辑单元,表示对数据进行处理或转换的功能模块。通常用圆形或矩形表示。
- **数据存储(Data Store)**: 表示系统中存储数据的位置,如数据库或文件。通常用圆柱体或开口矩形表示。
- **数据流(Data Flow)**: 表示数据在系统中的流动路径。通常用箭头表示。

下面是团队协作系统的顶层 DFD、细化的任务管理业务的 DFD,分别如图 3-6 和图 3-7 所示。

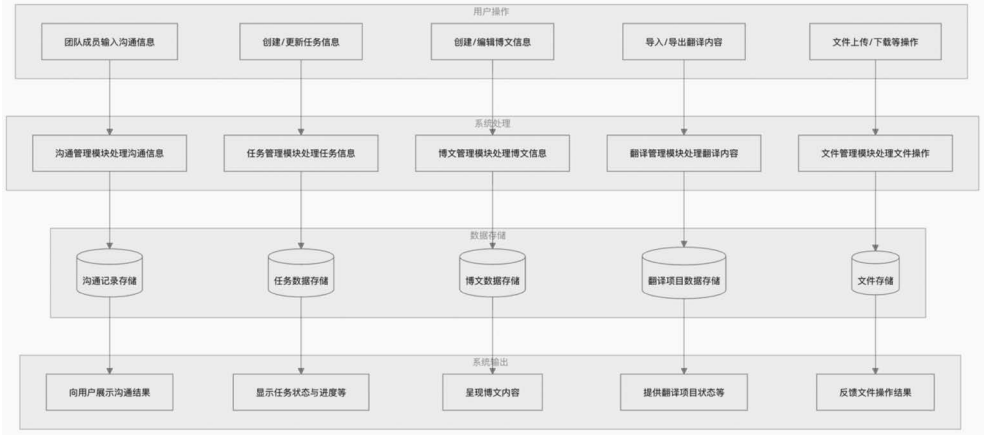


图 3-6 由豆包大模型生成的 TMS 顶层 DFD

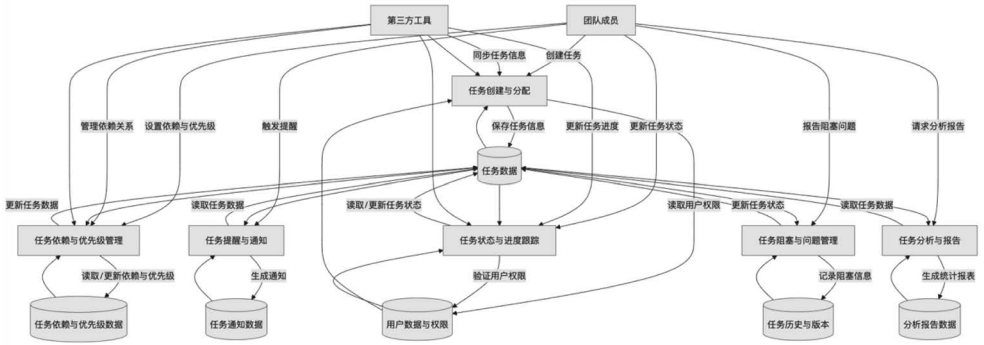


图 3-7 由 OpenAI o1-mini 生成 TMS 任务管理模块的 DFD

数据流图的绘制步骤如下。

- (1) 确定系统边界: 明确系统的范围,识别与系统交互的外部实体。
- (2) 识别数据流: 确定系统中数据的流动路径,包括输入和输出数据流。
- (3) 定义处理过程: 识别系统中对数据进行处理的功能模块,并描述其功能。
- (4) 确认数据存储: 确定系统中需要存储数据的位置,如数据库或文件。
- (5) 建立数据流动的连线: 使用箭头连接外部实体、处理过程和数据存储,表示数据的流动方向。

(6) 细化和优化: 确保图表布局清晰,避免交叉线条和混乱连接。详细命名数据流、处理过程和数据存储,确保名称具有清晰的意义。

2. 状态转换图

状态转换图(STD)是一种用于描述系统或对象在不同状态之间转换的图形化工具。它广泛应用于软件工程、系统设计和业务流程建模中,帮助理解和分析系统的动态行为。STD核心元素如下。

- 状态(State): 系统或对象在某一特定时刻的条件或情形。通常用椭圆表示。
- 初始状态(Initial State): 系统开始时的状态,用一个实心圆表示。
- 终止状态(Final State): 系统结束时的状态,用带有圈的实心圆表示。
- 事件(Event): 引发状态变化的触发条件或动作,用箭头上的标签标明事件名称。
- 动作(Action): 在状态转换过程中执行的操作。通常在转换箭头上标注,位于事件名称之后,用“/”分隔。
- 转换(Transition): 状态之间的移动,通常由事件触发,用箭头连接不同的状态,箭头上标注触发事件。

制作过程(而在智能软件工程中分为两步,先由 o1-mini 生成 Mermaid 格式脚本,如图 3-8 所示;再导入 Mermaid Live Editor 工具中生成 STD,如图 3-9 所示)如下。

- (1) 确定系统边界: 明确要建模的系统或对象范围,识别与之交互的外部实体。
- (2) 识别状态: 列举系统或对象可能经历的所有状态,确保涵盖所有关键情境。
- (3) 定义事件: 确定引发状态变化的事件或条件,包括用户操作、时间触发等。
- (4) 绘制状态和转换: 使用图形工具绘制各状态,利用箭头连接状态,标注触发事件和可能的动作。
- (5) 标注初始和终止状态: 明确系统的起始点和结束点,确保图表逻辑完整。
- (6) 验证和优化: 检查状态转换图的准确性,确保无遗漏或逻辑错误,根据需要进行优化和调整。



任务状态跟踪的状态转换图 (STD)

以下是团队协作系统 TMS 中任务状态跟踪模块的状态转换图,使用 Mermaid 语法表示。该图展示了任务在不同状态之间的转换过程,涵盖主要状态和触发事件。

```
mermaid
stateDiagram
    [*] --> 待办
    待办 --> 进行中 : 开始任务
    待办 --> 取消 : 取消任务
    进行中 --> 已完成 : 完成任务
    进行中 --> 阻塞 : 遇到阻塞
    进行中 --> 取消 : 取消任务
    阻塞 --> 进行中 : 解除阻塞
    阻塞 --> 取消 : 取消任务
    已完成 --> [*]
    取消 --> [*]
```

图 3-8 第一步大模型生成 STD 脚本

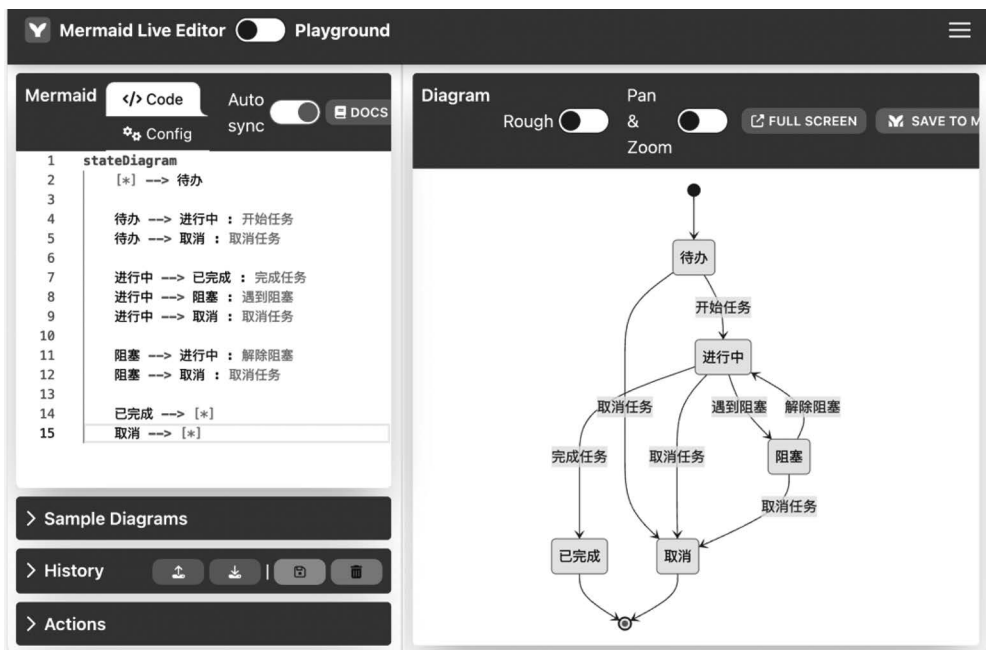


图 3-9 在工具中导入脚本生成图

3. 实体关系图

实体关系图(ERD)是一种用于建模和描述系统中数据结构及其相互关系的图形化工具。ERD 在数据库设计、系统分析和软件工程中广泛应用,帮助开发者和分析师理解和规划数据的组织方式,如图 3-10 所示,其核心元素如下。

- 实体(Entity): 系统中可以独立存在的对象或概念,通常代表一个具体或抽象的事物。用椭圆形或矩形表示,内部标注实体名称。
- 属性(Attribute): 描述实体特征的数据项,直接标注属性名称,用直线连接实体。
- 关系(Relationship): 描述实体之间关联的方式,用菱形表示,连接相关实体,并标注关系,如一对一(1:1)、一对多(1:N)、多对多(M:N)等关系。
- 基数(Cardinality): 描述实体之间关系的数量限制,即在关系线上标注 1、N(多)、M(多)等符号。
- 主键(Primary Key): 唯一标识实体实例的属性,即在该属性下画线或使用“PK”标记。
- 外键(Foreign Key): 在一个实体中引用另一个实体的主键,用于建立实体之间的关联,通常使用“FK”标记。

制作过程(记住:第一次作为提示词,可以更好地引导大模型)如下。

- (1) 确定系统边界: 明确要建模的系统或子系统范围,识别涉及的主要实体。
- (2) 识别实体: 列出系统中所有相关的实体,确保涵盖所有关键对象或概念。
- (3) 定义属性: 为每个实体列出其相关属性,确定每个属性的数据类型和约束(如是否为主键)。
- (4) 确定关系: 识别实体之间的相互关系,定义关系的类型(1:1、1:N、M:N)。
- (5) 定义基数: 为每个关系标注基数,明确实体之间的数量限制。

(6) 绘制 ERD：使用图形工具绘制实体、属性和关系，连接各元素并标注名称及基数。

(7) 验证和优化：检查 ERD 的完整性和一致性，确保所有实体、属性和关系准确无误。根据需要进行调整和优化。

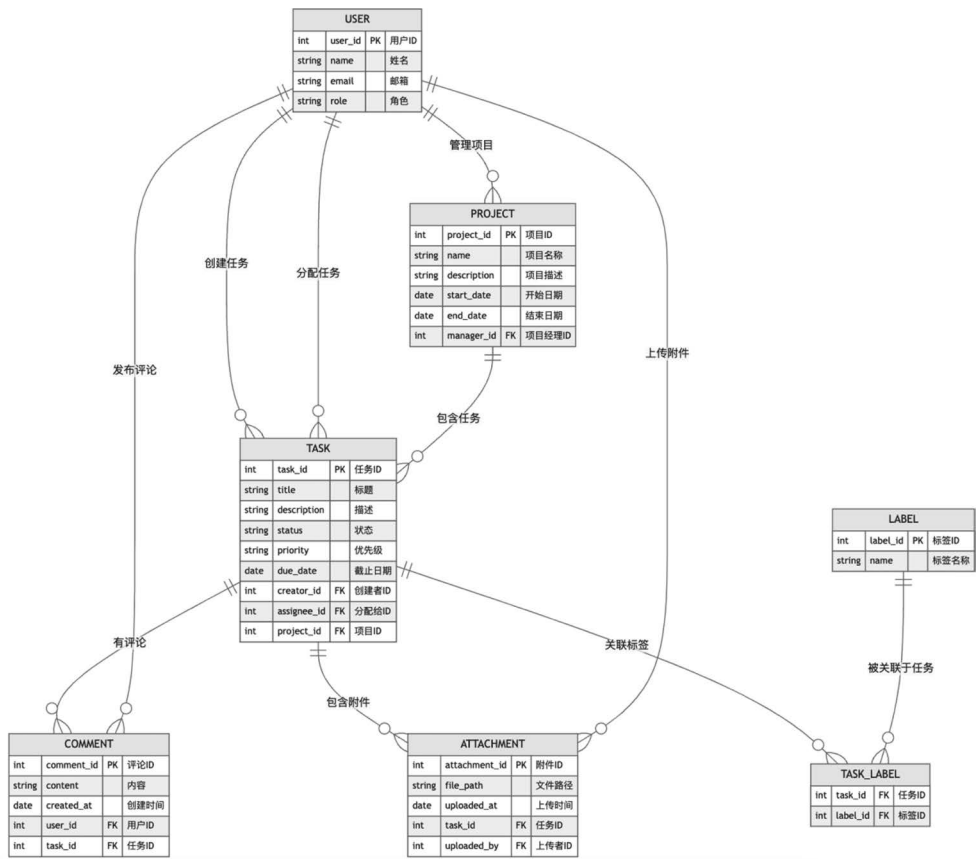


图 3-10 由 OpenAI o1-mini 生成 TMS 的 E-R 图



观看视频

3.4.3 面向对象的分析方法

面向对象的分析(Object-Oriented Analysis, OOA)是一种在软件工程中广泛应用的方法论,特别适用于需求工程阶段。通过识别问题域中的对象、类及其关系,构建出与实现无关的分析模型,为后续设计和实现提供基础。以下将详细介绍面向对象分析方法的核心概念、主要任务、分析步骤及其在需求工程中的应用。

面向对象分析的核心是将现实世界中的事物抽象为对象,并通过对象的属性和行为来描述问题。对象通过类、封装、继承等机制组织在一起,形成复杂的系统。

- **对象(Objects)**: 系统中的实体,具有特定的属性和行为。例如,在团队协作系统 TMS 中,用户、任务、项目等都是对象。
- **属性(Attributes)**: 描述对象特征的数据项。例如,任务对象的属性包括任务 ID、标题、描述、截止日期等。
- **类(Classes)**: 具有相似属性和行为的对象的抽象。例如,用户类定义了所有用户对

象的共有属性(如用户 ID、姓名)和行为(如创建任务、分配任务)。

- **方法(Methods)**: 对象能够执行的操作或行为。如任务对象的方法可能包括开始任务、完成任务、设置优先级等。
- **封装(Encapsulation)**: 将对象的属性和行为封装在一起,隐藏内部实现细节,只暴露必要的接口。例如,任务对象的内部状态(如状态、优先级)对外部不可直接修改,只能通过其提供的方法进行操作。
- **继承(Inheritance)**: 类之间的层次关系,通过继承机制,子类可以继承父类的属性和方法,减少重复代码。例如,管理员类可以继承自用户类,拥有额外的管理权限。

面向对象分析的主要任务如下。

- **识别类和对象**: 从问题域中提取类和对象,定义它们的属性和方法。
- **刻画类的层次结构**: 通过继承等关系组织类,形成类的层次。
- **描述对象之间的关系**: 包括关联、聚合、依赖等。
- **对对象的行为建模**: 通过时序图、活动图等描述对象的行为和交互。

为了完成这些任务,一般会经过下列分析步骤。

- (1) 确定系统范围与目标,明确需求背景,定义系统边界。
 - (2) 识别关键场景(用例): 确定用户角色(参与者),即找出系统的主要用户和外部交互实体,然后定义用例、描述用例,即使用自然语言描述用例的基本流和分支流。
 - (3) 构建用例模型——用例图,即从用户角度描述系统的功能需求,包括用户角色(参与者)和用例,并详细描述用例之间的交互。
 - (4) 识别分析类,从用例中提取类,分配职责。
 - (5) 建立类/对象模型,包括类图和子类等。
 - (6) 建立对象行为模型,包括时序图和活动图。
- 下面详细讨论用例图、类图、时序图和活动图等。

1. 用例图

用例图(Use Case Diagram)是统一建模语言(Unified Modeling Language, UML)中的一种静态模型,它通过图形化的方式展示系统的功能需求及其与外部实体(如用户或其他系统)的交互关系。用例图主要应用于系统分析与设计阶段,帮助开发团队理解各个参与者的角色及其与系统的交互方式,从而为后续的设计和实现提供依据。

用例图主要由以下几个基本元素组成。

- (1) **参与者(Actor)**: 与系统交互的外部实体,可以是用户、其他系统或硬件设备。
- (2) **用例(Use Case)**: 系统提供的功能或服务(但不涉及具体的实现细节),满足参与者的某种需求。通常使用动词短语命名,如“登录系统”“生成报告”。
- (3) **关系(Relationships)**: 分为以下 4 种。
 - **关联关系(Association)**,表示参与者与用例之间的交互关系,用实线连接。
 - **泛化关系(Generalization)**: 表示参与者或用例之间的继承关系,用带箭头的实线表示,适用于描述角色或功能的层次结构。
 - **包含关系(Include)**: 表示一个用例在执行过程中必然包含另一个用例,用带箭头的虚线表示,箭头指向被包含的用例。
 - **扩展关系(Extend)**: 表示一个用例在某些条件下可以扩展另一个用例,用带箭头的

虚线表示,箭头指向被扩展的用例。

用例图的设计步骤如下。

- (1) 确定系统边界: 明确系统的范围, 决定哪些功能属于系统内部, 哪些属于系统外部。
- (2) 识别参与者: 列出所有与系统交互的外部实体, 理解他们的需求和目标。
- (3) 识别用例: 根据参与者的需求, 定义系统应提供的功能。
- (4) 建立关系: 连接参与者与用例, 明确他们之间的交互方式。
- (5) 优化和细化: 检查图表的完整性和准确性, 确保没有遗漏关键功能或角色。

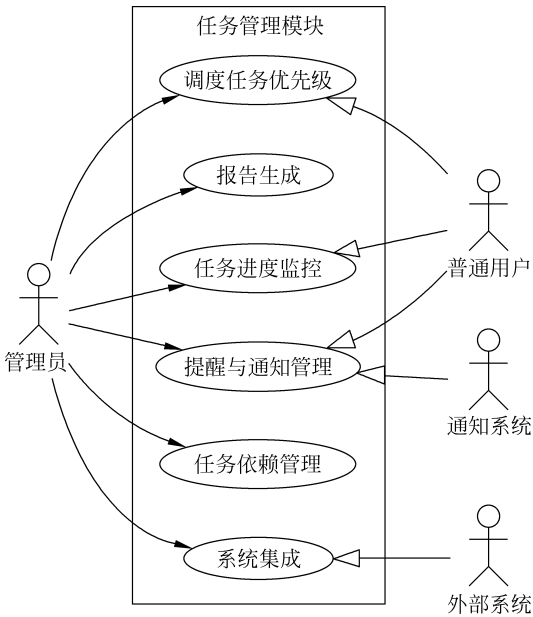


图 3-11 由 LLM 生成的用例图

如果由大模型来参与设计, 即人机交互设计, 上述过程就可以简化为以下三步。

(1) 输入准确的、完整的提示词(包括含有简要元素表示和设计思路等必要的提示信息、充分的上下文内容, 如上传业务文档)让大模型生成用例图。

(2) 研发人员检查验证(即上面第(5)步), 确定用例图是否有问题或达到要求。

(3) 如果有问题, 将问题反馈作为提示词的一部分, 重复进行上面两步; 如果达到要求就结束。

这里, 可以让大模型生成 plantUML 格式的用例图脚本, 然后导入 UML 绘图工具中生成最终的用例图。这里以团队协作系统 TMS 的任务管理业务为例, 生成用例图, 如图 3-11 所示。

2. 类图

类图(Class Diagram)也是 UML 的一种核心静态结构图, 它通过图形化的方式描述系统中的类、接口及其相互关系(如继承、关联、依赖、聚合和组合等)。类图广泛应用于面向对象的系统分析与设计, 为开发人员提供清晰的系统结构图, 指导代码编写。类图主要由以下几个基本元素组成。

(1) 类(Class): 类是具有相同属性、行为(类的方法)和关系的对象的集合。

(2) 接口(Interface): 接口是一组方法的声明, 定义了类必须实现的行为, 使用带有《interface》标识的类图符号表示。

(3) 关系(Relationships)。

- 继承(Inheritance): “Is-a”关系, 即一个类(子类)继承了另一个类(父类)的属性和方法。表示方法: 使用一条带有空心三角形箭头的实线, 箭头指向父类, 表示子类继承自父类。
- 组合(Composition): “Has-a”关系, 是一种强依赖关系, 部分对象的生命周期由整体对象控制。表示方法: 使用一条带有实心菱形箭头的实线。箭头指向整体类, 表示部分对象属于整体对象。

- **关联**(Association): 表示类之间的结构性关系,通常是一对一或一对多。表示方法:使用实线连接两个类,可以添加数量限制和角色名。
- **实现**(Realization): 表示类实现接口的关系。表示方法:使用带空心箭头的虚线,从类指向接口。
- **聚合**(Aggregation): 表示整体与部分的关系,但部分可以独立于整体存在。表示方法:使用带空心菱形的实线,从整体指向部分。
- **依赖**(Dependency): 一个类的实现依赖于另一个类,通常是临时的、较弱的关系。表示方法:使用一条带有箭头的虚线。箭头指向被依赖的类,表示一个类使用了另一个类的对象。

除此之外,还有依赖注入、友元关系(仅限于某些语言,如 C++)等。

类图的一般分析和设计步骤如下。

- (1) 识别主要类和接口,分析需求,确定系统中的主要实体和它们的职责。
- (2) 定义类的属性和方法,列出每个类的特性(属性)和行为(方法)。
- (3) 确定类之间的关系,分析类与类之间的交互,确定它们之间的继承、组合、关联、实现、聚会、依赖、依赖等关系。
- (4) 绘制类图:使用 UML 工具或手工绘制,将类、接口及其关系可视化。
- (5) 优化和审查:检查类图的准确性和完整性,确保没有遗漏关键类或关系。

类图的分析 and 绘制步骤,可以作为大模型提示词的补充,帮助大模型进行更换的慢思考,生成如图 3-12 所示的类图。输入的提示词如下。

根据规范的类图元素表示法,请认真分析“任务管理模块”的类和接口、属性和方法,确定类之间的关系(继承、组合、关联、实现、聚会、依赖等),最终生成类图,以 plantUML 格式输出。

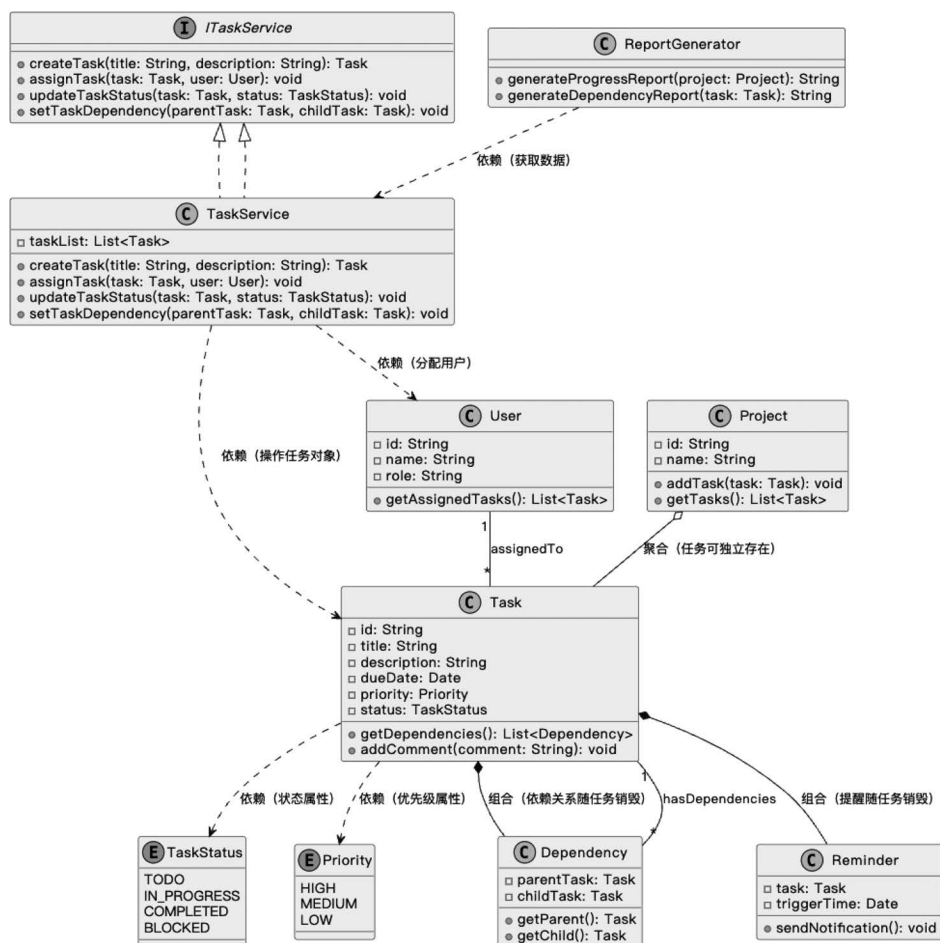
这张类图是一次性生成的,还存在一些问题,留到需求评审再揭晓其中的问题,并进行改正、优化,生成接近完美的类图。

下面是常见的设计类图的最佳实践,可以作为审核大模型生成类图的参考标准。

- 保持简洁与清晰:类图应易于理解,避免包含过多细节。关注关键类和关系。
- 使用标准 UML 符号:遵循 UML 标准符号,确保图表的一致性和规范性,便于团队协作。
- 合理命名:类、属性和方法应使用具有描述性的名称,遵循命名规范。
- 避免过度设计:只添加必要的类和关系,避免因追求完美而导致类图过于复杂。
- 分层次绘制:对复杂系统,可将类图分为多个层次或模块,逐步细化。
- 定期审查与更新:随着需求和设计的变化,及时更新类图,保持其与实际系统的一致性。

3. 时序图

时序图(Sequence Diagram)也是 UML 比较常见的 UML 视图,它通过图形化的方式展示在特定场景下系统中对象之间的动态交互过程,特别强调时间顺序和消息传递。时序图帮助开发团队捕捉用例中各步骤的执行顺序和交互方式,揭示对象之间的协作和通信,从而理解系统的运行机制,为系统的设计和实现提供清晰的指导。



时序图主要由以下几个基本元素组成。

(1) **对象(Object)**: 系统中参与交互的实体, 可以是类的实例、组件或其他系统。表示方法: 位于图的顶部, 以矩形框表示, 通常包含对象名和类名(例如 `user: User`)。

(2) **生命线(Lifeline)**:表示对象在交互过程中的存在时间,存在周期,从交互开始到结束。表示方法:垂直虚线,从对象底部延伸至交互结束。

(3) **消息(Message)**: 对象之间的通信行为,包括方法调用、数据交换或事件等。消息类型分为以下几种。

- 同步消息(Synchronous Message): 请求方等待接收方处理完成后再继续执行。用“实心箭头+实线”表示,箭头指向消息的接收方。
- 异步消息(Asynchronous Message): 请求方不等待接收方处理完成,立即继续执行。用“空心箭头+实线”表示。
- 返回消息(Return Message): 接收方处理完成后返回的结果。用“虚线箭头”表示。

(4) **激活条(Activation Bar)**: 表示对象在某一时间段内执行某个操作或方法的时间段。表示方法: 生命线上的细长矩形条, 垂直覆盖消息处理的时间范围。

(5) 组合片段(Combined Fragment): 用于描述消息传递的条件或循环结构,使用“alt”

表示条件分支,“loop”表示循环区域,“opt”表示仅在条件满足时执行(可选),“par”表示多个并发操作。

可以继续让大模型生成任务管理模块的时序图,如图 3-13 所示,而时序图的一般设计步骤如下。

- (1) **明确交互场景**: 确定要描述的业务流程或系统交互(如用户登录、支付流程),并识别参与交互的对象和外部角色(如用户、第三方服务)。
- (2) **绘制对象与生命线**: 在顶部水平排列所有参与对象,并添加生命线。
- (3) **定义消息传递顺序**: 按时间顺序从上到下排列消息,并区分同步异步和返回消息,完成“发起者发送消息给接收者,接收者处理消息并返回结果”这个过程。
- (4) **添加激活条**: 在生命线上添加激活条,显示对象在接收消息时的活动状态。
- (5) **添加逻辑控制**: 使用 alt、opt、loop 等描述分支、循环等复杂逻辑。
- (6) **优化与审查**: 检查时序图的完整性和准确性,确保交互流程清晰且无遗漏,删除冗余步骤,保持简洁性。

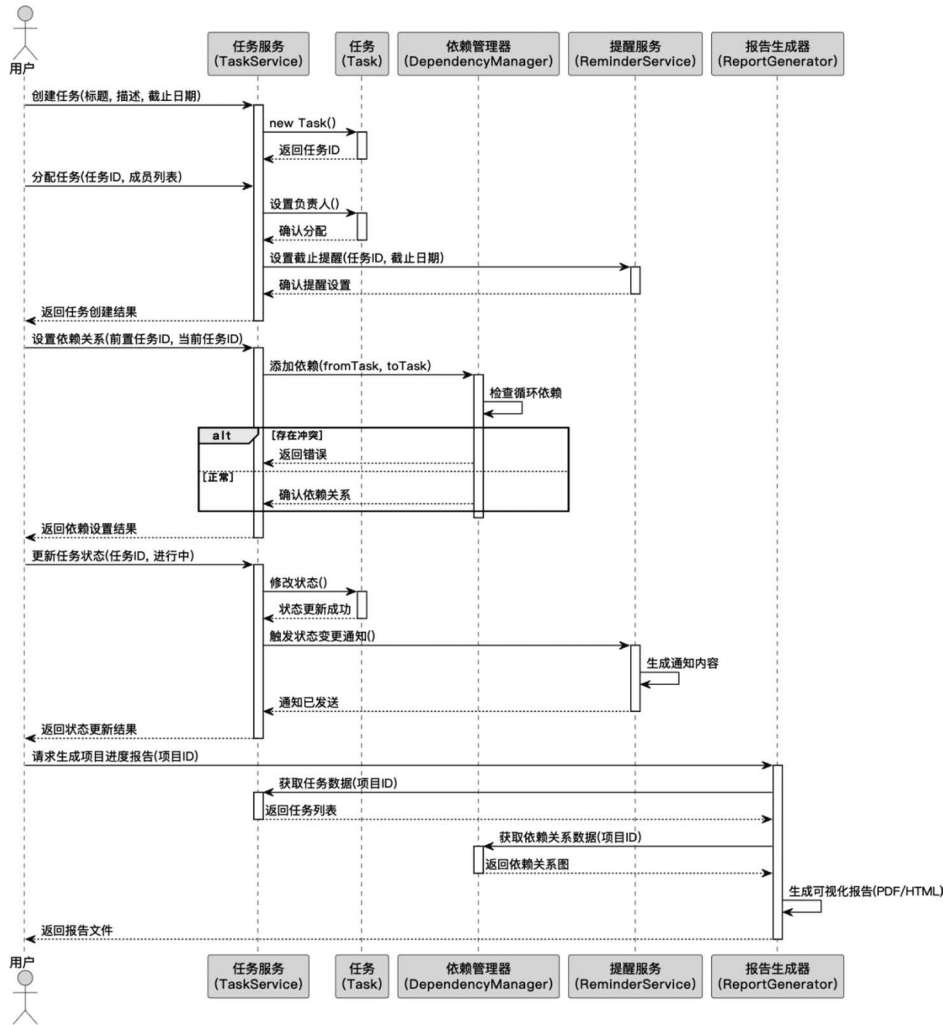


图 3-13 由 DeepSeek 生成的“任务管理模块”时序图

4. 活动图

活动图(Activity Diagram)是 UML 中的一种行为图,主要用于描述业务流程或系统操作的动态行为,尤其适合展示复杂流程中的并发、分支、同步等逻辑。它主要应用于业务流程建模,描述业务 workflow,也能应用在系统功能设计、算法流程可视化。活动图中的主要概念或元素如下。

- **活动(Activity)**: 表示流程中的一个具体操作或任务,其用圆角矩形、内部标注操作名称(如“提交订单”)表示。
- **开始节点(Initial Node)**: 流程的起点,只有一个,用实心圆点表示。
- **结束节点(Final Node)**: 流程的终点,分为活动终止节点(结束整个流程)和流程异常终止节点(结束当前分支),用空心圆套实心圆表示。
- **决策节点(Decision Node)**: 根据条件选择不同分支路径。用菱形表示,有多个出口,每个出口标注条件。
- **合并节点(Merge Node)**: 将多个条件分支合并为单一流程,也用菱形表示,但不标注条件。
- **分叉节点(Fork Node)**: 将单一流程拆分为多个并行执行的分支,用水平或垂直粗线表示。
- **结合节点(Join Node)**: 将多个并行分支同步合并为单一流程,用水平或垂直粗线表示。这里注意区分合并节点,合并节点是合并多个互斥分支(条件分支)的流程,而且任意一个分支执行完毕即可执行后续流程,而结合节点是同步多个并行分支的流程,且必须等待所有并行分支执行完毕才能继续。
- **对象节点(Object Node)**: 流程中涉及的数据或对象,之前介绍过。
- **泳道(Swimlane)**: 按参与者或系统模块划分责任区域(垂直或水平分区),并标注角色名称。
- **控制流(Control Flow)**: 表示活动之间的执行顺序,以带箭头的实线表示。
- **对象流(Object Flow)**: 表示活动与对象节点之间的数据传递。以带箭头的虚线表示,可在线上标注操作。

如果设计活动图,其过程如下。

(1) 明确流程目标: 首先确定需要描述的业务流程,然后识别核心参与者(用户、系统模块)和关键步骤。

(2) 绘制泳道划分责任: 按参与者或功能模块划分泳道,明确各区域职责。

(3) 定义节点与流: 添加开始节点,按顺序排列活动节点,通过控制流连接,并使用决策节点和合并节点处理分支逻辑,使用分叉节点和结合节点处理并行流程。

(4) 标注条件与对象: 在决策节点出口标注条件,添加对象节点表示数据传递。

(5) 验证与优化: 检查流程是否覆盖所有可能路径,确保分叉与结合节点成对出现,避免逻辑错误。

(6) 简化复杂分支,使用子活动图(Sub-Activity)分解嵌套流程。

如之前所述,利用大模型来生成活动图,主要是三步: 让大模型生成活动图的 plantUML 脚本,再导入 UML 工具生成图,然后检查活动图是否有问题。在活动图生成中,遇到问题会多一些,可能在开始有一行“left to right direction”是多余的,“back to”也不

对,需要改成相应的泳道等。经过两三次的来回,完成活动图的设计。我们甚至可以让 DeepSeek R1 打开“联网搜索”学习活动图的相关知识和表示,然后再打开“深度思考(R1)”,上传业务需求文档,结合当前业务生成活动图。图 3-14 是 DeepSeek 生成的 TMS 顶层的活动图。

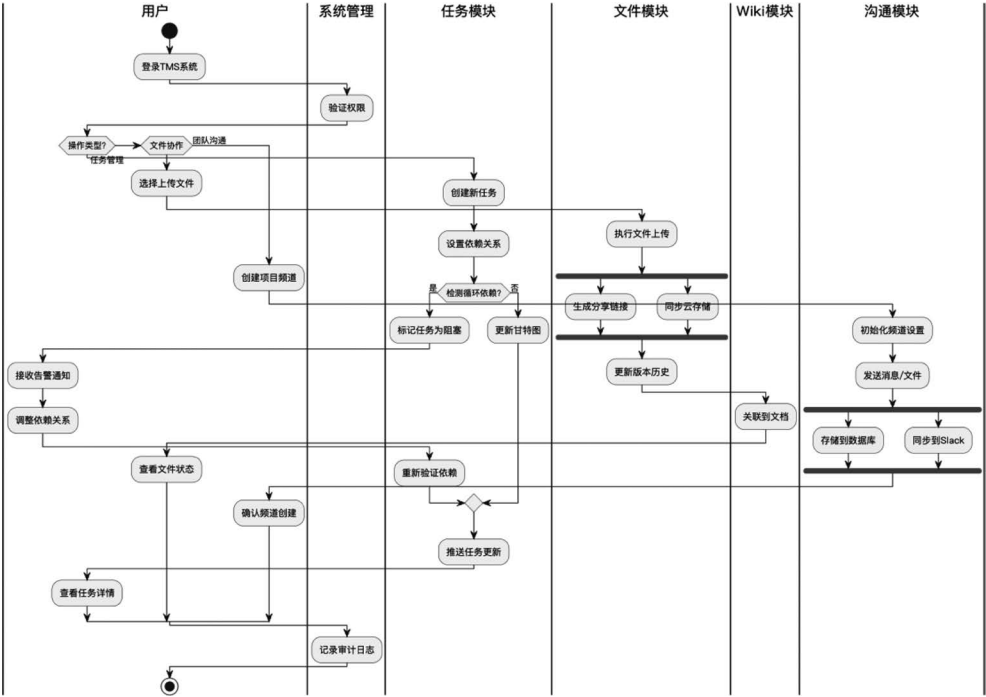


图 3-14 DeepSeek 生成的 TMS 顶层的活动图

3.4.4 面向敏捷的分析方法

在第 1 章中介绍了敏捷开发范式中的 BDD、Scrum 等落地框架,如果正在使用 BDD、Scrum(目前应用比较广泛的),那我们的需求分析就需要面向敏捷来展开,厘清史诗(Epic)、产品特性、用户故事(User Story, US)、验收标准(Acceptance Criteria, AC)等概念,并能结合业务定义特性、Epic、US、AC 等内容。

- **Epic**: 代表企业级某个业务目标,为团队提供产品愿景,指导需求优先级排序,通常跨多个迭代或版本来实现。
- **产品特性**: 具有独立业务价值的功能集合,满足用户可感知的价值,通常也需几个迭代完成,可以通过 MMF(最小可市场化特性)验证市场可行性,确保投资回报。
- **用户故事**: 从描述用户角色的行为,即某用户角色对具体功能的特定需求,遵循“**As a <用户角色>, I want to do <做什么>, so that <实现什么目标/价值>**”模板。

在敏捷开放范式中,我们从模糊的史诗逐步细化到明确的用户故事,即先确定 Epic,再将 Epic 进行分解成特性,然后基于特性细化成用户故事,而且每个层级均需验证业务价值,即遵守敏捷的“价值驱动”原则。

现在还是以团队协作系统 TMS 为例,让大模型帮我们定义本系统的 Epic,得到结果如

下,更详细的内容可以参考本书电子资料中的“TMS Epic”。

Epic 描述

作为一家快速发展的科技公司,我们的跨职能团队(开发、产品、测试、设计)分布在全球多个时区,亟须一套统一的协作系统来解决以下核心问题。

- **沟通碎片化**: 重要信息淹没在多个工具中,关键决策难以触达所有成员。
- **任务依赖混乱**: 跨项目任务优先级冲突频繁,导致资源浪费与进度延迟。
- **知识孤岛**: 文档与沟通记录分散,新成员难以快速获取上下文。
- **工具割裂**: Jira、GitHub、Slack 等工具数据不同步,操作效率低下。

Epic 愿景(或期望):

- **沟通无障碍**: 团队成员可以随时随地通过平台进行实时沟通,减少信息滞后和误传。
- **任务透明化**: 每个任务的状态、负责人和截止日期一目了然,确保责任到人,进度可控。
- **协作高效化**: 跨部门、跨地域的协作更加顺畅,项目资源得到合理分配,减少重复劳动。
- **决策迅捷化**: 管理层能够通过平台实时获取项目数据,快速响应市场变化,调整项目策略。
- **员工满意度提升**: 高效的工作流程和透明的管理机制,提升员工的工作体验和满意度,促进人才保留和发展。

近期目标: 通过 TMS 系统实现端到端的团队协作闭环,通过任务依赖自动检测减少 20% 的规划时间,提升 30% 的项目交付效率,降低跨团队沟通成本 50%。

基于这个愿景,我们让大模型尽可能列出 TMS 的特性,它列出了 4 大特性: 团队沟通业务需求、轻量级任务管理业务需求、系统集成与扩展功能和数据分析与报告工作,并展开到颗粒度更小的层次,形成第二层次的特性,如图 3-15 所示。

基于特性,可以进一步让大模型生成所有用户故事。在生成用户故事前,可以让大模型对业务进行分析,明确 TMS 中的各种用户角色。大模型没有简单地把它划分成两种角色: 普通用户和管理员,而是把 TMS 放在一个软件企业的真实环境中,结果列出了 8 种角色: 项目经理、开发人员、测试人员、产品经理、内容管理员、系统管理员、普通成员、外部协作者。其实,可以把“开发人员、测试人员、产品经理”合并为一种角色“团队成员”,这样就减少到 6 种角色: 项目经理、团队成员、内容管理员、系统管理员、普通成员、外部协作者。

再进一步,就可以按特性、按用户角色生成用户故事。例如,以前面讨论比较多的特性“轻量级任务管理业务需求”,可以针对下列 4 个特性:

- 任务创建与分配操作
- 任务状态与进度跟踪
- 任务依赖与优先级管理
- 任务提醒与自动化设置

分别按角色(如项目经理、团队成员等)来生成用户故事。大模型大概用了不到 10 分钟生成了 96 个用户故事,详见本书电子资料中的“用户故事-任务管理”。这里给出“任务创建

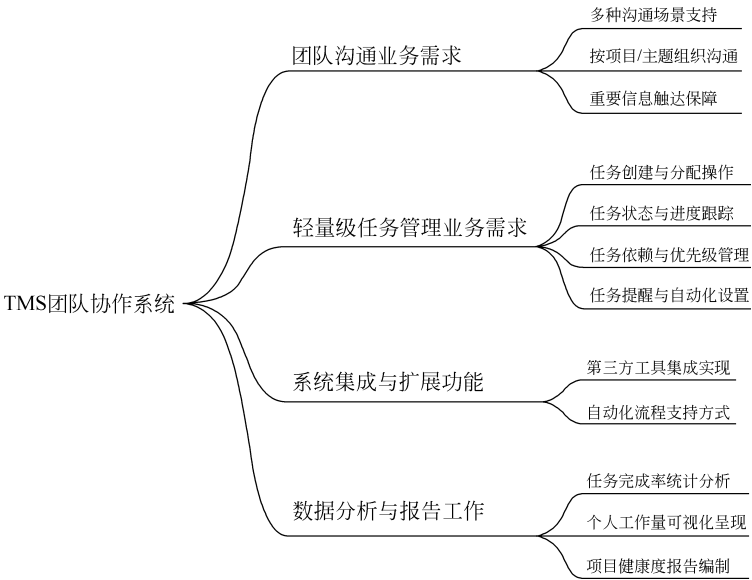


图 3-15 TMS 两个层次的特性

与分配操作”特性,每个角色 5 个用例,如表 3-1 所示。

表 3-1 “任务创建与分配操作”部分用户故事

项目经理角色	团队成员角色
用户故事 1.1 作为 项目经理,我想要 创建新任务并指定负责人和参与人,以便 确保任务有明确的责任人,推动项目按计划进行。	用户故事 2.1 作为 团队成员,我想要 快速创建新任务并提交给项目经理审批,以便 提出新的工作需求或改进建议。
用户故事 1.2 作为 项目经理,我想要 为任务设置标题、描述、截止日期和优先级,以便 清晰传达任务要求和紧急程度,优化资源分配。	用户故事 2.2 作为 团队成员,我想要 查看分配给我的任务列表,以便 了解我的工作优先级和截止日期,合理安排时间。
用户故事 1.3 作为 项目经理,我想要 批量创建和分配任务,以便 提高任务分配的效率,节省时间。	用户故事 2.3 作为 团队成员,我想要 更新任务的状态(如从“进行中”到“已完成”),以便 实时反映工作进展,保持信息透明。
用户故事 1.4 作为 项目经理,我想要 查看和编辑已创建×××务的信息,以便 根据项目进展和需求变化及时调整任务细节。	用户故事 2.4 作为 团队成员,我想要 为任务添加备注或附件,以便 记录工作细节或提供相关资料,支持任务完成。
用户故事 1.5 作为 项目经理,我想要 为任务添加标签或类别,以便 更好地组织和筛选任务,提升项目管理的可视化	用户故事 2.5 作为 团队成员,我想要 请求重新分配任务或调整截止日期,以便 应对工作量变化或遇到的阻碍,确保任务按时完成

敏捷的需求通过“用户故事”来定义,其实还没有到位,虽然感觉用户故事的颗粒度已经很小了,其实用户故事还是不可测的,即无法验证的。为了更好地澄清用户故事,确保用户故事具有良好的可测试性,需要为每一个用户增加验收标准。假如我们让大模型为“用户故事 1.1”明确验收标准,得到的结果如下。

用户故事:“作为 项目经理,我想要 创建新任务并指定负责人和参与人,以便 确保任务有明确的责任人,推动项目按计划进行。”

验收标准:

1. 任务创建界面可访问

- 前提条件: 项目经理已登录并具有创建任务的权限。
- 操作: 导航至“任务管理”模块并单击“创建新任务”按钮。
- 预期结果: 显示任务创建表单。

2. 填写必要的任务信息

- 前提条件: 项目经理在任务创建表单中。
- 操作: 填写任务标题、描述、截止日期、优先级等基本信息。
- 预期结果: 所有必填字段均可输入并验证格式(如截止日期为未来日期)。

3. 指定负责人和参与人

- 前提条件: 在任务创建表单中。
- 操作: 在“负责人”字段选择一名团队成员,并在“参与人”字段选择一个或多个团队成员。
- 预期结果: 系统允许选择已注册的团队成员,并显示所选成员的姓名。

4. 成功创建任务

- 前提条件: 项目经理已填写所有必填信息并指定负责人和参与人。
- 操作: 单击“保存”或“创建”按钮。
- 预期结果: 任务成功创建,显示在任务列表中,并且负责人及参与人收到任务分配通知。

5. 任务数据持久化

- 前提条件: 任务已成功创建。
- 操作: 刷新页面或重新登录系统。
- 预期结果: 新创建的任务依然存在于任务列表中,所有信息(包括负责人和参与人)准确无误。

这是相对灵活的验收标准。也可以要求按照 BDD(行为驱动开发)中 GWT 格式来定义用户故事的验收标准。GWT 代表:

- Given[前提条件],假定某种场景、前置条件下。
- When [操作],当什么事件发生或做什么操作时。
- Then[预期结果],那么会发生什么?

GWT 是一种基于特定的关键词(如 Given、When、Then)的结构化语言,也可以理解为需求特定领域语言(DSL),被称为“Gherkin”。基于 GWT,可以让大模型生成用户故事 1.1 的 GWT 格式的验收标准,如下所示。

场景 1: 访问任务创建界面

Given 项目经理已登录 TMS 系统

When 项目经理导航到“任务管理”模块并单击“创建新任务”按钮

Then 系统应显示任务创建表单

场景 2: 填写必要的任务信息

Given 项目经理位于任务创建表单页面

When 项目经理填写任务标题为“设计数据库架构”

And 填写任务描述为“设计并优化现有数据库架构以提升性能”

And 选择截止日期为“2025-02-15”

And 设置优先级为“高”

Then 系统应接受所有输入并验证格式正确

场景 3: 指定负责人和参与人

Given 项目经理位于任务创建表单页面

When 项目经理在“负责人”字段选择“张三”

And 在“参与人”字段选择“李四”和“王五”

Then 系统应显示所选负责人和参与人的姓名

场景 4: 成功创建任务

Given 项目经理已填写所有必填信息并指定负责人和参与人

When 项目经理单击“保存”按钮

Then 系统应成功创建任务

And 任务应显示在任务列表中

And “张三”“李四”及“王五”应收到任务分配的通知

场景 5: 任务数据持久化

Given 项目经理已成功创建任务

When 项目经理刷新页面或重新登录 TMS 系统

Then 新创建的任务应依然存在于任务列表中

And 所有任务信息(包括负责人和参与人)应准确显示

本书电子资料中提供了更多示例,请参考“用户故事 1.1~1.5 验收标准定义”。

3.4.5 场景分析方法

在普遍使用敏捷开发范式的今天,需求常常被描述成用户故事,用户故事更多地体现用户行为,而用户行为一定会发生在特定的场景下,场景可以理解为行为发生的前置条件或环境、时间和空间的组合等。场景分析(Scenario Analysis)方法就是通过描述用户在特定情境下的行为和需求,即通过构建和分析具体的使用场景,揭示用户在实际操作中的特定需求,帮助团队全面理解系统如何全面满足用户的实际需求。它不仅关注“系统需要做什么”,还深入探讨“用户在何种情况下使用系统”以及“系统如何响应用户的操作”。

1. 要素

场景分析的核心在于系统性模拟产品使用可能遇到的各种情景,其要素如下。

(1) **角色与参与者**: 明确业务中的关键角色(如用户、系统、外部服务),分析其行为动机与交互模式。

(2) **环境与约束**: 包括时间、空间、技术条件等外部环境因素,例如,“用户在地铁通勤时使用手机”这一空间约束。

(3) **事件流与分支**: 描述正常流程、异常情况及应对措施,如电商订单支付失败后的退款流程。

(4) **价值与目标**: 明确场景解决的核心问题(如提升转化率)和用户期望的最终价值。

2. 步骤

基于这些要素的描述,场景分析按下列步骤进行。

(1) 明确分析目标与边界: 确定分析范围,避免需求泛化。

(2) 识别关键变量(如用户行为、政策变化、技术趋势),通过敏感性分析筛选出对业务影响最大的驱动因素。

(3) 构建场景框架: PSPS 模型(角色-场景-痛点-解决方案)。

(4) 三步演化建模(非形式化→半形式化→形式化): 通过自然语言描述需求,逐步转换为结构化模型,适用于复杂系统。

(5) 细化场景与验证: 用户故事与用例图,通过“作为<角色>,我想要<目标>,以便<价值>”的格式描述需求,结合 UML 用例图可视化交互流程。用户故事与用例图分别在 3.4.4 节和 3.4.3 节进行了讨论。

(6) 价值流分析: 追踪用户从“了解商家”到“售后服务”的全链路体验,识别关键优化点。

(7) 冲突检测与闭环验证: 例如,在任务依赖管理中,检测循环依赖并生成告警,确保逻辑一致性。

(8) 场景优先级与策略生成: 根据发生概率与战略重要性,筛选高价值场景。

(9) 设计 MVP(最小可行产品),优先实现高频、高痛点的核心功能。

3. 应用场景与工具方法

(1) 需求挖掘。

- 痛点识别: 通过假设“无此功能时的用户痛苦程度”验证需求必要性。例如,微信的即时通信功能若缺失,用户将难以维系社交关系。
- MECE 法则: 穷尽所有可能分支(如用户登录成功/失败/超时),确保需求覆盖无遗漏。

(2) 业务流程优化。

- 活动图: 分角色绘制流程,见 3.4.3 节。
- 异常处理设计: 针对“任务依赖冲突”“系统宕机”等异常场景,预设容灾方案。

(3) 风险管理与战略规划,如跨领域协同,在轨道交通控制系统中,通过场景模拟验证 ATP(列车自动防护)系统的安全逻辑。

3.5 需求评审

许多时候,用户告诉我们的并不是他们真正的需求。例如,当用户对我们说“我需要有一个大铁锤”,很可能他是给出了解决办法,有意无意地将自己真正的意图埋藏在心里,而他

的真实需求是“要在墙上打个洞”。同时,业务有特定的领域知识、有多种不同的用户角色,其流程也可能相当复杂,我们一般也没有足够时间和每一个用户角色进行沟通、交流,这样,要完全理解用户的需求、掌控业务流程和规则,几乎是不可能做到的事情^[1]。更何况将收集到的需求要进行整理和分析,这个过程中又会丢掉一些需求信息,而且要把整理出来的需求结果写下来,又如何保证准确、完整和全面呢?

虽然有一些建模语言(如统一建模语言),但其应用不够成熟,况且项目进度压力往往比较大,项目团队也没有足够时间去完成需求建模或需求的详细描述,在敏捷中则是通过极其简单的“用户故事”来描述,还要拥抱需求变化,在需求上投入就更有限。所有这些因素,都会进一步引起需求上出现更多的问题。

在需求定义(文档)中潜在的一个问题,看似不大,但随着产品开发工作的不断推进,经过很多环节之后,小错误会不断扩大,问题可能会变得严重,我们会为此付出巨大的代价。正如俗话说“小洞不补、大洞吃苦”。而且,缺陷在前期发现得越多,对后期的影响越小,后期的缺陷就会减少得越快,最终遗留给用户的缺陷就很少。如果没有需求评审,那么在测试执行阶段会发现主要的缺陷,会有很多缺陷还遗留到产品发布之后,质量明显降低,如图 3-16 所示。

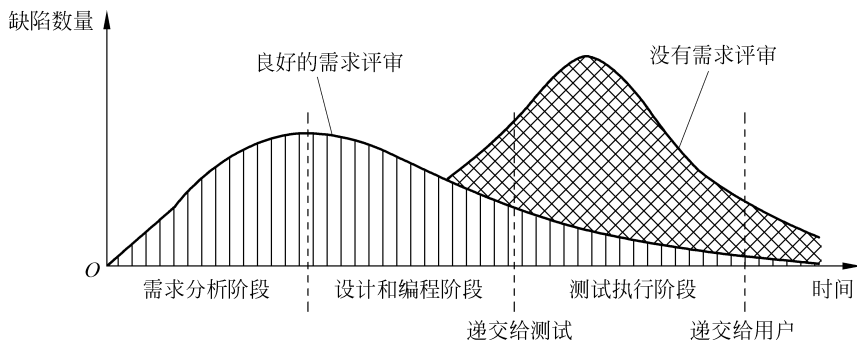


图 3-16 需求评审对缺陷分布的影响

假如在需求定义阶段犯一个错误,将一个功能定义得不合理,然后设计和编程都按照需求定义去实现了这个功能,只等到测试阶段或发布到用户那里,才发现不对。这时候,要修正问题,必须重新设计、重新编程和重新测试,代价是不是很大? 问题发现得越迟,要重做的事情就会越多,返工量就越大。也就是说,缺陷发现或解决得越迟,其带来的成本就越大。

从正面看,需求评审也能够带来不少益处。通过产品需求评审,研发人员能够更好地理解产品的功能性和非功能性需求,使软件需求能够被明确描述,其可测试性能够得到保证,避免在后期产生不同的理解而引起的争吵。通过产品需求评审,也为后期设计、编程和测试打下良好的基础,为范围确定、工作量估算等工作提供足够的信息。评审得越充分,后期需求变更的机会就减少,项目风险进一步降低。

3.5.1 如何确定传统软件需求的评审标准

通过对需求进行正确性检查,以发现需求定义中的问题,尽早地将缺陷发现出来,降低成本,并使后续过程的变更减少,降低风险。在进行需求评审之前,需要确定评审的标准。只有建立了标准,才有了评审的依据。如果没有依据,就无法判断对与错,很难更准确地、更快速地发现问题。所以,在进行需求、涉及评审时,制定一个明确的、客观的质量评判标准是必要的。

根据上面需求的层次性,可以针对不同的需求层次(业务需求、用户角色需求、功能性/非功能性需求)分别建立评审标准,这样更有针对性。例如,针对业务需求,下列各项内容都应该具备。

- 业务流程图,并体现其用户角色。
- 业务规则,涉及各项业务及其输入/输出的约束、条件、边界等。
- 业务操作,有哪些正常操作,又有哪些异常操作。
- 业务数据,包括业务规则中涉及的数据、输入/输出、外部接口等各项数据。
- 其他业务要求、影响,如业务可管理性、业务发展/变化规律等。

而对用户角色需求层次,要求用例能覆盖所有一级、二级功能项。但三个层次在评审标准上也有共同的诉求,如正确性、一致性,不存在违背客观事实、客观规律的描述,没有遗漏项,没有二义性,确保需求描述完整、清楚、准确。根据 IEEE 建议的需求说明的标准,需求评审应遵守的需求质量标准如表 3-2 所示。

表 3-2 软件系统需求质量标准

特性要求	基 本 描 述	说 明
正确性	检查在任意条件下软件系统需求定义及其说明的正确性	• 是否存在对用户无意义的功能? • 每个需求定义是否都合理、经得起推敲? • 有哪些证据证明用户提供的规则是正确的? • 假设是否有存在的基础? • 是否正确地定义了各种故障模式及其处理方式
可行性	需求中定义的功能应具有可执行性、可操作性等	• 如需求定义的功能是否能通过现有技术实现? • 所规定的模式、数值方法是否能解决需求中存在的问题? • 所有功能是否能够在某些非常规条件下实现? • 是否能够达到特定的性能要求
规范性	需求定义符合业界标准、规范的要求	• 本行业有哪些特定要求? • 是否用行业术语来描述需求? • 符合业务描述的习惯吗
可验证性	每项需求应该能找到一种方法或通过设计测试用例来进行验证,从而判断该项需求是否得到正确实现	• 系统的非功能需求(如性能、可用性等)是否有特定的指标? • 这种指标能否有办法获得? • 输入/输出数据是否有清楚的格式定义从而容易验证其精确性
优先级	每项需求因其重要性不同而其实施的先后次序是不同的,即得到一个特定的优先级	• 有没有将所需求的功能或非功能特性分为高、中、低三个优先级别? • 是不是级别越高,用户使用该功能越频繁
合理性	每项特性的合理程度	• 如果某项功能有多种实现方法,目前是否选择了最好的方法
完备性	涵盖系统需求的功能、性能、输入/输出、条件限制、应用范围等的程度,覆盖度越高,完备性越好	• 是否有漏掉的功能或输入/输出条件? • 是否考虑了不同需求的人机界面? • 功能性需求是否覆盖了所有异常情况的处理和响应? • 是否识别出了与时间因素有关的功能

特性要求	基本描述	说明
无二义性	对所有需求说明只有一个明确统一的解释。面对同一项描述,不同的人有着相同的理解。如果存在二义性,将会导致人们误解需求而开发出偏离需求的产品	- 措辞是否准确、不存在模棱两可的含义? - 是否将系统的实际需求内容和所附带的背景信息分离开来? - 需求描述是否足够清楚和明确使其已能够作为开发设计说明书和功能性测试的依据? - 为了避免歧义性,尽量把每项需求用简洁明了的用户性的语言表达出来
兼容性	软件可以和系统中的硬件和其他(子)系统无缝地集成起来	- 是否说明了软件对系统中硬件的依赖性? - 是否说明了环境对软件的影响? - 相互之间的接口都定义清楚了吗
一致性	所定义的需求之间没有相互排斥、冲突和矛盾,前后一致	- 不同需求项(描述)之间是否有矛盾? - 某些业务规则是否存在冲突? - 所规定的模型、算法和数值方法是否相容
易追溯性	每一项需求定义可以确定其来源	- 是否可以根据上下文关系找到所需要的依据或支持数据? - 后续的功能变更都能找到其最初定义的功能吗

3.5.2 如何评审敏捷需求——用户故事

从整体、宏观角度看,敏捷开发中的需求评审标准和表 3.2 所描述的内容相同。

- 从内容评审上,要求正确性、可行性、规范性、可验证性、优先级、合理性、完备性、无二义性、兼容性、一致性和易追溯性等。
- 从文档评审上,要求规范性、易理解性、一致性、准确性、易修改性等。

要善于区分用户故事中所描述的内容是用户的活动、商业价值,体现用户的真正需求,而不是解决方案。但就具体的“用户故事”的评审标准,可以概括为一个单词“INVEST”,即包含 Independent(独立的)、Negotiable(可协商性)、Valuable(有价值的)、Estimable(可估算的)、Small(足够小的)和 Testable(可测试的)^[4]。

(1) **每个故事首先是有价值的**,即对客户具有价值,帮助客户在业务上解决一个什么问题。没有价值的东西,用户自然不需要,即使设计、编程做得再好,也毫无意义。传统软件开发往往说要交付一个软件功能,而敏捷开发则强调向客户交付价值。让用户故事有价值的一个好办法就是让客户来写出这些用户故事。

(2) **一个用户故事的内容要是可以协商的**,用户故事不是合同、不是一种契约,而只是用一种简洁的方式来描述需求,不包括太多的细节。具体的细节在后期沟通时再逐步丰富。一旦客户意识到用户故事并不是一个契约,是可以协商的,客户就乐意帮助我们写出一个又一个的用户故事。

(3) **用户故事具有独立性**,能够描述很具体而完整的一个用户行为,使用户故事之间没有依赖性。如果用户故事之间存在这样或那样的依赖关系,那么当我们给一个用户故事定义其优先级、估算其工作量、安排任务等工作就变得很困难。通过用户故事的分解或组合,尽量减少其依赖性。

(4) **用户故事足够小,小到能够估算**。故事的颗粒度越大,估算越不准确,估算和实际

差异越大,带来的项目风险也就越大。如果用户故事过大,甚至一个迭代(如一个 Sprint 只有一周,5 个工作日)内都不能完成,那迭代就无法进行了。

(5) 一个用户故事要是可测试的,以便于能够验证它是否得到完整的实现。如果一个用户故事不能被验证,那么就无法判断某个工程师的工作是否已完成、这个用户故事是否能交付给客户。

除了 INVEST 标准之外,也要挖掘用户故事背后隐藏的前提、假设、约束或条件。根据用户行为及其涉及的场景、假定或条件,为每个用户故事建立其验收标准。只有列出了用户故事的各种验收标准,用户故事描述的需求才算明确、完整,才具有良好的可测试性。



观看视频

3.5.3 智能需求评审

我们手工写的文档可以交给大模型来评审,大模型生成的文档可以由我们来进行人工评审。当然,由 GPT-4o 生成的需求,可以让 DeepSeek R1 模型来评审。

如果让大模型来评审需求,可以分为以下两步。

- (1) 让 LLM 先评审需求文档、指出问题。
- (2) 再让 LLM 帮助优化文档,或提出修改意见。

我们让它评审电子材料中的需求文档:“团队协作系统 TMS 的业务需求(未评审的)”,结果 OpenAI o1 发现了 9 个主要的问题,详见电子材料中的“OpenAI o1 评审 PRD 的意见”,这里列出 9 个问题(省去说明)。

- 文档结构与编号不统一。
- 引用标识不明或缺失。
- 需求层级与业务价值的关联不够清晰。
- 缺少对非功能性需求的说明。
- 依赖管理和自动化需求较为宏观,缺少细化场景。
- 跨项目需求与单个项目需求略有交叉重复。
- 缺乏验收标准或衡量指标。
- 对用户角色和权限的描述较笼统。
- 部分需求尚未提供示例或用例。

评审其实要按标准进行,所以第二次,要求它按表 3-2 中的标准重新进行评审,评审的结果大不一样,详见电子材料中的“OpenAI o1 评审 PRD 的意见”,这里列出其评审意见的框架。

1. 正确性

- 无明确的规则验证或支撑
- 数字引用混乱且无解释

2. 优先级

- 优先级缺失或不明确

3. 合理性

- 功能描述过于笼统

4. 完备性

- 非功能需求缺失

- 部分需求仅涵盖“理想场景”

5. 无二义性

- 需求措辞有一定模糊空间
- 背景与需求混杂

6. 兼容性

- 系统运行环境与依赖描述不足

7. 一致性

- 文档内部的结构与编号不一致
- 同一需求点在多个地方出现

8. 易追溯性

- 缺少需求来源或关联说明
- 数字跳转和引用无从查询

总体改进建议

- 完善文档结构与编号
- 明确业务与非功能需求的优先级
- 细化集成与自动化场景
- 补充可量化示例与测试用例
- 添加非功能需求与兼容性说明

也可以要求它按 INVEST 标准对用户故事进行评审。例如,让 OpenAI o1 对由 DeepSeek 生成的用户故事“用户故事-任务管理”(见电子材料)进行评审,评审的结果见“基于 INVEST 标准对用户故事的评审结果”。限于篇幅,这里仅列出大模型发现的部分问题。

1. Independent(独立性)

部分用户故事之间存在隐性依赖,未明确说明。例如:

- 用户故事 PM-5(在甘特图中可视化任务依赖关系)与 PM-6(批量设置任务依赖关系)之间存在逻辑关联,后者的实现可能需要前者的支持。
- 用户故事 TM-4(任务完成后自动更新后置任务状态)依赖于任务依赖关系的设置,但未明确说明这种依赖。

2. Negotiable(可协商性)

某些用户故事的目标过于具体,可能限制了实现方式的灵活性。例如:

- 用户故事 PM-2(在看板或甘特图中查看任务的进度条和完成百分比)直接限定了使用“甘特图”或“看板”作为实现方式,缺乏协商空间。
- 用户故事 TM-9(通过自定义标签标记任务)未说明标签的使用范围,可能导致实现方式过于局限。

3. Valuable(有价值性)

部分用户故事的价值表述较为模糊,未能突出其对用户的直接好处。例如:

- 用户故事 PM-6(批量设置和调整任务的依赖关系)虽然提到了“提高效率”,但未具体说明效率提升的场景或用户收益。

- 用户故事 TM-10(查看任务的历史状态变更记录)未明确说明用户通过查看历史记录能够获得什么具体价值。

.....

在 3.4.3 节中提到图 3-12 的类图还存在问题,现在就对它进行人工评审,第一次我们发现了一些问题,通过提问或指出问题让大模型重新生成类图。

我们 review 了生成的任务管理的类图,有一些疑问或几个问题:

1. 两个 interface 的作用是什么?
2. 为什么每个 interface 有两个 realization? 是明显的错误吧?
3. 为什么 dependency 类包含 parent 和 child? 如果两个 task 平等地相互依赖,哪个是 parent? 哪个是 child? from/to 是不是比较合理?
4. ReminderService 和 Reminder 之间没有连线,可是 ReminderService 明显 Aggregate Reminder,对吧?
5. user 的 role 为什么是 string, 不是 enum?

先回答这些问题,然后尽可能纠正错误,重新生成更准确的任务管理的类图。

DeepSeek 思考了 10s,就回答了上面 5 个问题,如图 3-17 所示,然后生成了新的类图 plantUML 格式脚本。



图 3-17 Deepseek 解答问题

将 plantUML 格式 的脚本导入工具,生成的类图如图 3-18 所示,再经过检查,新的类图基本没什么问题了。在实际项目中,还要再结合上下文场景,会有少量不同的考虑,例如,可能会问 IReminderService 起什么作用? Reminder 功能有什么关注点需要分离?

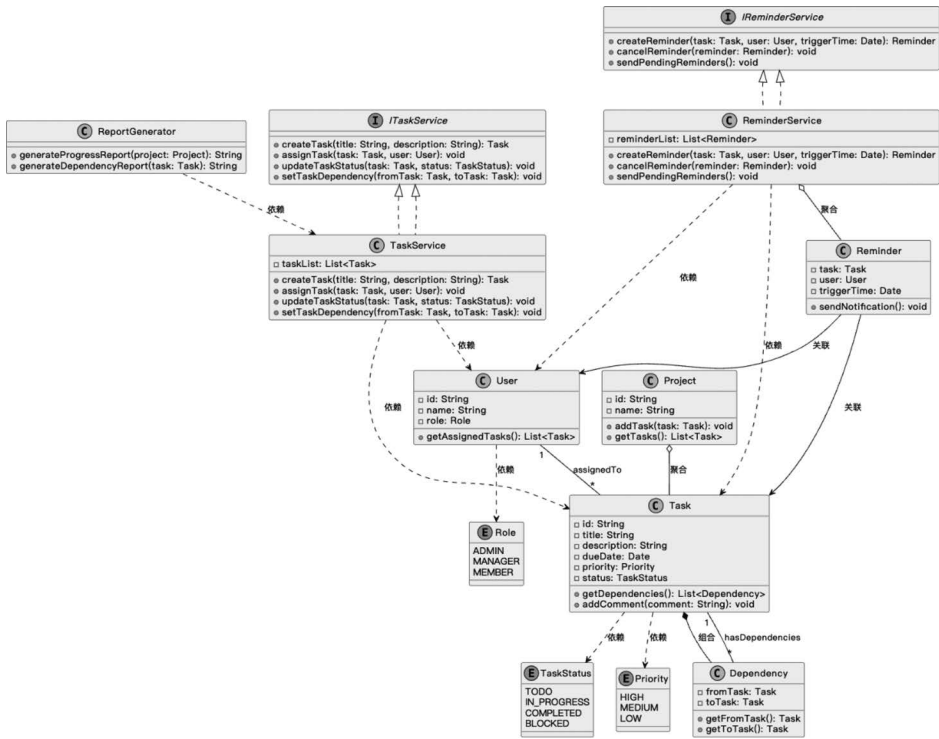


图 3-18 优化后的任务管理类图

3.6 需求跟踪与变更管理

前面介绍了需求获取、需求分析与定义和需求评审,基本完成了一个完整的需求周期。一般来说,需求评审通过了,业务需求就转换为软件系统的需求,然后我们就会基于需求去完成设计与实现,最终通过系统测试、验收测试来验证需求。在软件设计、开发和测试过程中,需求可能会发生变更,我们需要评审变更带来的风险,从而决定能否接受变更。而且,软件产品会经过多次迭代,在这个过程中,需求跟踪显得非常必要,因为有效的需求跟踪与变更管理对于确保这些需求能够准确实现和持续维护至关重要。

需求跟踪通过建立需求跟踪矩阵,系统地监控每项需求从提出到实现的全过程,确保项目的可控性和透明度。同时,需求变更控制机制则保障在项目实施过程中,任何对需求的调整都能被有序管理,减少潜在的风险和资源浪费。通过结合先进的大模型技术,需求跟踪与变更管理不仅提升了效率和准确性,还增强了团队对项目动态变化的响应能力。接下来,将深入探讨需求跟踪与变更管理的具体内容及其在 TMS 项目中的应用。

3.6.1 需求跟踪

需求跟踪(Requirements Tracing)是软件工程中确保需求在整个开发生命周期中得到

有效管理和验证的重要过程。它涉及从需求获取到最终交付的每一个环节,确保每项需求都能被追踪到其源头,并且在实现过程中保持一致性和完整性。需求跟踪的主要内容如下。

(1) **需求的识别与记录**: 将所有需求以文档形式记录(需求文档),确保每项需求都有明确的描述和背景信息。每项需求可以被定义为配置项,所以需要为它分配唯一的标识符,以便于后续的跟踪和引用。

(2) **需求与设计、代码和测试用例等的映射**: 确保每项需求都有相应的设计文档、代码和测试用例的支持,记录如何实现该需求并得到验证。如果发现了缺陷,要确保缺陷能够追溯到具体的需求。设计“纵向需求追踪矩阵”,将需求与设计、实现和测试用例、缺陷等之间的关系可视化,便于快速查找和验证。

(3) **需求变更管理**: 先要记录需求变更的请求,包括变更的原因和影响评估,然后跟踪需求的变更历史,确保所有变更都有据可查,并更新相关文档。

(4) **需求状态更新与跟踪**: 记录每项需求的当前状态(如“已提出”“正在开发”“已测试”等),确保团队成员对需求的进展有清晰的了解。其次,维护需求变更的审计日志,帮助管理者了解需求的动态变化。

(5) **需求的验证与确认**: 通过测试和评审等方式验证需求的实现情况,确保最终交付的软件符合用户的期望。在软件交付后收集用户反馈,确保需求的实现满足用户需求,并进行必要的调整。

1. 需求横向跟踪矩阵

需求横向跟踪矩阵(Horizontal Traceability Matrix)是一种用于分析和管理需求之间依赖关系、优先级关系和关联性的工具。其矩阵结构表现为横轴和纵轴都是需求项,其交叉单元表示需求间的关系类型(如依赖、冲突、关联或优先级等关系)或依赖性程度等,其主要目的如下。

- 建立需求间的依赖关系。
- 识别需求的关联性。
- 分析需求实施的先后顺序。
- 评估需求变更的潜在影响。

需求横向跟踪矩阵可以帮助我们更好地管理需求的依赖性,同时也有利于我们进行需求变更的管理。表 3-3 是根据 TMS 系统的任务管理业务而设计的需求横向跟踪矩阵示例(由大模型 openAI o1 生成),供参考。

表 3-3 TMS 任务管理需求横向跟踪矩阵示例

需求项	任务创建与分配	任务状态与进度跟踪	任务依赖与优先级管理	用户与权限管理	系统集成与扩展
任务创建与分配	—	关联(强)	关联(强)	关联(强)	关联(中等)
任务状态与进度跟踪	关联(强)	—	关联(强)	关联(中等)	关联(中等)
任务依赖与优先级管理	关联(强)	关联(强)	—	关联(强)	关联(中等)
用户与权限管理	关联(强)	关联(中等)	关联(强)	—	关联(强)
系统集成与扩展	关联(中等)	关联(中等)	关联(中等)	关联(强)	—

2. 需求纵向跟踪矩阵

需求纵向跟踪矩阵(Vertical Traceability Matrix)是一种用于需求管理和项目管理的

关键工具,用于显示需求与其设计、实现、测试和缺陷之间的关系。它提供了一种清晰的方式来追踪和记录每个需求的起源、设计、实现和测试情况,旨在帮助团队明确和跟踪需求在产品开发过程中各个阶段的源头和演变,能够有效地支持需求的验证、变化管理和风险评估。

- **确保完整性:** 确保所有需求在设计和开发过程中得到满足,并且没有遗漏。
- **改进可追溯性:** 方便追踪需求从定义到实现的整个生命周期,快速定位变更和相关影响。
- **支持需求变更管理:** 在需求变化时,快速识别受影响的部分,并评估变更的影响范围。
- **增强沟通:** 为团队成员和项目干系人等提供一个共同理解需求的基础,改善沟通与协作。

需求纵向跟踪矩阵通常包含以下主要元素。

- **需求 ID:** 每项需求的唯一标识符,用于跟踪。
- **需求描述:** 对需求的简要描述。
- **源头:** 需求的来源,可以是客户、市场调研、法律法规等。
- **设计规范:** 与需求对应的设计文档或规范。
- **实现:** 实施该需求的代码、模块或组件。
- **测试用例:** 对应需求的测试用例,以验证需求是否被实现。
- **验证结果:** 描述需求通过测试的结果(例如:通过、未通过,如果未通过,会关联一个缺陷 ID)。
- **变更历史:** 需求的任何变化和更新记录,以便审计和追踪。

表 3-4 是根据 TMS 系统的任务管理业务而设计的需求纵向跟踪矩阵示例(由大模型 openAI o1 生成),其中,纵向可以理解为时间轴,需求体现在设计、实现(编程)、测试、变更等不同阶段,供参考。

表 3-4 TMS 任务管理需求纵向跟踪矩阵示例

需求 ID	TMS-001	TMS-002	TMS-003	TMS-004	TMS-005	TMS-006	TMS-007
需求描述	支持任务创建与分配	任务状态与进度跟踪	任务依赖与优先级管理	任务阻塞与问题管理	数据分析与报告	任务历史与版本管理	集成与扩展性
源头	用户反馈	项目需求	项目需求	用户反馈	项目需求	用户反馈	项目需求
设计规范	设计文档 V1	设计文档 V1	设计文档 V1	设计文档 V1	设计文档 V1	设计文档 V1	设计文档 V1
实现模块	TaskManagement	TaskTracking	DependencyManager	IssueManagement	AnalyticsModule	VersionControl	IntegrationModule
测试用例	TC_CreateTask_001	TC_TaskStatus_002	TC_TaskDependency_003	TC_TaskBlocking_004	TC_DataAnalysis_005	TC_TaskHistory_006	TC_Integration_007
验证结果	通过	通过	BUG-001	BUG-002	通过	通过	通过
变更历史	N/A	N/A	需求描述不清晰,需重新定义依赖关系和优先级管理的具体流程	任务阻塞标记功能未实现,需增加阻塞原因的详细描述	N/A	N/A	N/A

3.6.2 需求变更管理

需求变更控制是项目管理中的一个重要环节,尤其在软件开发和团队协作系统(如TMS)中,需求的变化是常态。有效的需求变更控制可以确保项目在面对变化时仍能保持方向和目标的一致性。以下是需求变更控制的系统阐述,包括需求变更控制流程、需求变更风险评估(依赖性影响)和决策过程。

1. 需求变更控制流程

需求变更控制流程通常包括以下几个步骤。

(1) 变更请求: 用户、客户、项目干系人或任何团队成员都可以提出需求变更请求。请求应包含变更的详细描述、原因及预期效果。

(2) 变更评估: 评估变更所带来的影响,包括对项目范围、进度、成本和质量的影响。此阶段还需考虑变更对现有需求的依赖关系。

(3) 变更审查: 组织变更审查会议,邀请项目干系人共同讨论变更请求。审查的重点在于变更的必要性、可行性和潜在风险。

(4) 决策: 根据评估和审查结果,决定是否批准变更请求。

- 批准: 变更请求被接受并纳入项目计划。
- 拒绝: 变更请求被拒绝,须向请求者说明原因。
- 延迟: 变更请求暂时搁置,待进一步信息或条件成熟后再做决定。

(5) 变更实施和验证: 一旦变更请求被批准,相关团队应根据新的需求进行设计、开发和测试。通过测试验证新需求是否满足预期,包括受影响但未发生变更的需求。

(6) 变更记录: 所有变更请求及其决策和实施结果应记录在案,并确保所有相关文档和系统都得到更新,以便后续审计和参考。

2. 需求变更风险(含依赖性影响)评估

在需求变更控制中,风险评估是至关重要的一步,尤其是对依赖关系的影响。以下是进行需求变更风险评估的关键要素。

(1) 识别依赖关系: 确定变更需求与其他需求之间的依赖关系。例如,某个功能的实现可能依赖于其他功能的完成。

(2) 评估影响: 分析变更对依赖需求的影响,考虑以下几方面。

- 功能影响: 变更是否会影响其他功能的实现或性能。
- 时间影响: 变更是否会导致项目进度的延误,特别是依赖于变更的任务。
- 资源影响: 变更是否需要额外的资源或调整现有资源的分配。

(3) 风险等级评估: 根据影响程度和发生概率,对风险进行等级划分(如高、中、低)。高风险的变更需要特别关注和管理。

(4) 制定应对策略: 针对识别出的风险,制定相应的应对策略。

- 风险规避: 修改变更请求以减少风险。
- 风险缓解: 采取措施降低风险发生的可能性或影响。
- 风险接受: 在风险可控的情况下,接受风险并做好监控。

3. 变更决策原则

(1) 过程透明: 所有变更请求及其评估和决策过程应保持透明,确保所有相关人员了

解变更的背景和影响。

(2) **干系人积极参与**：关键干系人应积极参与变更评估和决策过程，以确保不同视角的考虑和共识。

(3) **数据驱动**：决策应基于数据和事实，而非主观判断。使用历史数据、用户反馈和市场趋势来支持决策。

(4) **灵活性**：在快速变化的环境中，决策过程应保持灵活，能够快速响应新的信息和变化。

(5) **持续改进**：在每次变更后，回顾变更控制流程的有效性，识别改进点，以优化未来的变更管理。

需求变更控制是确保项目成功的重要环节，通过系统的流程、风险评估和决策过程，团队能够有效管理需求的变化，降低风险，确保项目目标的实现。在团队协作系统(TMS)中，需求变更控制的有效实施将有助于提升团队的协作效率和产品质量。

3.7 业务架构

平日，人们谈得最多的是技术架构，然后是产品架构、数据架构，其实还包括应用架构、业务架构等。本节就来谈谈业务架构，即我们对业务的理解要上升到架构层次，系统地、在更高的抽象层次上去理解业务、业务模式，而不只是简单地了解各种零碎的业务知识、业务功能等。

业务架构是指组织在实现其战略目标时，所采用的结构和流程的整体框架。它通过模块化设计、流程标准化、信息流透明性、灵活性、资源分配优化和决策支持，有效降低业务的复杂性。业务架构的核心在于将业务需求与技术实现相结合，确保系统能够灵活应对市场变化和客户需求。

业务架构也可以理解为企业治理结构、商业能力与价值流的蓝图，明确定义企业的治理结构、业务能力、业务流程、业务数据和商业能力等，帮助我们识别出业务流程中包含的业务要素，更好地理解业务要素之间的关系。其中，业务能力定义企业做什么，业务流程定义企业怎么做。

3.7.1 业务架构的价值与构建

从降低业务复杂性的角度看，业务架构的价值主要体现在以下几方面。

(1) **模块化与清晰的责任划分**：通过将复杂的业务流程分解为可管理的模块，明确各模块的职责和依赖关系，减少业务活动的重叠和冲突。例如，在团队协作系统(TMS)中，业务架构将沟通、任务管理、权限控制等功能模块化，确保每个模块有明确的职责和边界。

(2) **流程标准化与优化**：通过标准化和优化业务流程，去除冗余步骤，减少不必要的复杂性，提升整体效率。例如，在任务管理业务中，业务架构通过标准化任务创建、分配和跟踪流程，减少了手动操作和重复性工作。

(3) **信息流的透明性与可追溯性**：通过设计清晰的信息流，确保数据在业务流程中的有效传递，减少信息孤岛和沟通障碍。例如，在跨项目任务管理中，业务架构通过跨项目甘特图视图和任务列表视图，确保团队成员能够清晰了解任务依赖关系和优先级。

(4) **灵活性与适应性**：通过设计灵活的业务架构，支持快速迭代和调整，确保组织能够及时响应市场变化和客户需求，减少因变化带来的复杂性。例如，在任务提醒与自动化模块中，业务架构支持灵活的任务状态流转规则和自动化提醒功能。

(5) **资源分配的优化**：通过合理的资源分配，确保高优先级任务得到充分支持，减少资源浪费和低效分配带来的复杂性。例如，在任务优先级管理中，业务架构支持根据任务优先级(如高、中、低)进行排序和资源分配。

(6) **决策支持与数据分析**：通过提供数据支持和分析工具，帮助团队做出更明智的决策，减少因决策失误带来的复杂性。例如，在任务依赖与优先级的统计与分析模块中，业务架构提供健康度报告和统计功能。

构建有效的业务架构，一般会遵循以下步骤。

(1) **需求分析**：识别和分析业务需求，包括客户需求、市场趋势和内部目标。通过与利益相关者的沟通，收集需求信息，确保需求的全面性和准确性。

(2) **定义和优化业务流程**：设计和优化业务流程，确保流程高效且能够满足需求。使用流程图、甘特图等工具可视化业务流程，便于理解和沟通。

(3) **组织结构设计与优化**：明确、优化组织内部的角色和职责，确保每个环节都有专人负责。设计灵活的组织结构，以适应业务变化。

(4) **业务领域划分与优化**：将业务活动划分为不同的领域，识别关键业务能力和资源配置。确保各领域之间的协同和信息共享。

(5) **业务服务定义**：明确组织提供的核心服务和产品，确保服务的功能、目标客户和交付方式与市场需求一致。

(6) **信息流设计**：设计信息流，确保数据在业务流程中的有效传递。确保信息流的透明性和可追溯性，以便于后续的分析 and 优化。

(7) **技术架构支持**：选择合适的技术工具和平台，支持业务流程的实施。确保技术架构能够灵活应对业务变化。

(8) **文档化与沟通**：将业务架构文档化，确保所有相关人员能够访问和理解。定期与团队和利益相关者沟通，确保业务架构的透明性和一致性。

(9) **持续优化业务架构**：如定期评估业务流程的效率和效果、讨论业务架构的有效性和改进措施，识别瓶颈和改进点。使用关键绩效指标(KPI)监控业务流程的表现，或鼓励团队成员和利益相关者提供改进建议。

业务架构通过模块化设计、流程标准化、信息流透明性、灵活性、资源分配优化和决策支持，有效降低了业务的复杂性。在团队协作系统(TMS)中，业务架构通过明确的功能模块、标准化的流程、清晰的信息流和灵活的任务管理，确保了业务活动的高效性和一致性，减少了因复杂性带来的管理难度和效率低下。通过系统化的业务架构设计和优化，组织能够更好地应对市场变化和客户需求，提升整体运营效率和响应能力。

3.7.2 业务架构的框架与工具

为了更好地落地业务架构，需要借助框架或工具来实现。下面将介绍专门针对业务架构的框架和工具。同时，也可以借助像 TOGAF、Zachman、DoDAF 等统一的业务架构框架来实施业务架构，因为业务架构是企业架构的核心组成部分。

因为像 TOGAF、Zachman、DoDAF 等框架非常丰富,由于篇幅所限,无法详细阐述其内容,这里只进行简单介绍,更详细的内容,读者可以访问官方网站获取相关的文档。

1. ArchiMate

ArchiMate 是由 The Open Group 推出的一种开放的企业架构描述语言,支持企业架构的业务、应用和技术等不同层次的描述创建架构视图,并支持与 TOGAF 等企业架构框架高度集成。其中,ArchiMate 支持业务架构的可视化和建模,展示业务架构与其他架构域的关系,并能清晰地表达信息流、业务流程、业务服务、角色和功能之间的关系,如用人形图标表示业务角色,用圆形图标表示业务服务(如订单处理服务),用圆角的矩形图标来表示业务处理,用矩形图标表示业务功能,用方形图标表示业务对象(如订单、库存),用箭头线表示关系(如执行、支持等)。

- **业务流程**是一系列有序的、动态的且具体的活动,描述“如何”完成某项任务或实现某个目标,如接收订单、发货等。
- **业务功能**是组织为实现其目标而执行的一组相关活动,描述“什么”需要被完成,而不是“如何”完成。业务功能是静态的、比较高层次的描述,通常是描述组织能力的,如订单管理、库存管理等。

由大模型生成的、基于 ArchiMate 描述符号的 TMS 业务架构图,见电子材料中的“TMS ArchiMate 业务架构图.xml”。

2. BIZBOK

BIZBOK(Business Architecture Body of Knowledge)是由业务架构协会(Business Architecture Guild)提出的框架,定义了业务架构的关键领域、术语和工具,涵盖了需求分析、流程设计与优化、组织结构设计、信息流设计、技术架构支持和文档化与沟通等内容,提供了业务架构(如业务能力、价值链、组织结构和业务流程等方面)的最佳实践和标准,包括构建业务架构的指南。

- **业务能力地图(Business Capability Map)**: 列出和描述组织的核心业务能力,帮助识别关键业务能力和资源配置。
- **价值流(Value Streams)**: 描述组织如何通过一系列的业务活动创造价值,强调从起点到终点的价值创造过程,可以用价值流映射工具来分析。
- **信息流与数据管理(Information Map and Data Management)**: 描述组织内部的业务流程中的信息流动和数据管理策略,确保数据的准确性、一致性和可用性。
- **组织结构与角色**: 描述组织的结构、角色和责任,明确各个角色在业务流程中的职责,可以用 RACI 矩阵来分析。
- **业务服务与产品**: 列出和描述组织提供的核心业务服务和产品,确保服务的功能、目标客户和交付方式与市场需求一致。
- **业务流程**: 详细描述和支持组织业务活动的一系列步骤和流程,确保流程高效且能够满足需求,用标准化的方法(如 BPMN)或服务蓝图等工具可视化业务流程。

3. DoDAF

DoDAF(Department of Defense Architecture Framework)是美国国防部发布的、用于指导体系结构开发的框架和概念模型,它通过可视化模型描述系统的各个方面(如业务视图、系统视图和技术标准视图等),确保系统的互操作性和集成性。DoDAF 2.0 提供了 8 个

视点(View Point),如图 3-19 所示,从干系人的角度,将问题空间划分为更加易于管理的更小的模块,形成众多的模型,以便不同的干系人可以从不同的角度去关注体系内特定利益相关领域的数据和信息,同时也能保持对全局的把握。其中,DoDAF 业务视点,包含 9 个模型,使之能够充分描述业务概念、资源流、业务活动、业务规则和业务事件等,这与业务架构的关键要素相符。

- OV-1 顶层业务概念图
- OV-2 业务资源流表述模型
- OV-3 业务资源流矩阵
- OV-4 组织关系图
- OV-5a 业务活动分解树
- OV-5b 业务活动模型
- OV-6a 业务规则模型
- OV-6b 业务状态转换模型
- OV-6c 业务事件跟踪模型

除了业务视点,数据与信息视点、服务视点等也和业务描述关系密切。

- **数据与信息视点**(Data and Information Viewpoint,DIV),采集业务信息需求和结构化的业务流程规则,描述与信息交换有关的信息,如属性、特征和相互关系。
- **服务视点**(Services Viewpoint,SvcV),说明系统、服务以及支持业务活动的功能性组合关系,这些系统功能或服务资源支持了业务活动,方便了信息交换。服务视点中的功能、服务资源、组件可以与业务视点中的体系结构数据关联。



图 3-19 DoDAF 2.0 的 8 个视点

4. TOGAF

TOGAF(The Open Group Architecture Framework)是由开放群组(The Open Group)推出的一个开放式的企业架构框架,提供了一套标准化的方法和工具,帮助企业设计和实施企业架构。在 TOGAF 中,业务架构是企业架构的四大核心域之一(其他三个是数据架构、应用架构和技术架构),其中,业务架构描述组织的业务流程、组织结构、业务服务和信息流

等,并分为动机、组织和行为。

- 动机:驱动力、目标、目的、度量。
- 组织:施动者、角色。
- 行为:业务服务、合同、服务质量;流程、事件、控制、产品;功能、业务能力、行动方案、价值流。

业务战略决定业务架构,它包括业务的运营模式、流程体系、组织结构、地域分布等内容。TOGAF 强调基于业务导向和驱动的架构来理解、分析、设计、构建、集成、扩展、运行和管理信息系统,复杂系统集成的关键是基于架构(或体系)的集成,而不是基于部件(或组件)的集成。

TOGAF 的 ADM(Architecture Development Method,架构开发方法)过程中,业务架构的开发和优化是重要环节。通过需求分析、业务流程设计、业务服务定义等步骤,确保业务架构与企业的战略目标一致。而且 TOGAF 提供了一系列工具和模板(如业务流程图、服务蓝图)来支持业务架构的设计和文档化,帮助企业降低业务复杂性。业务架构开发主要的工作如下。

- 选择参考模型、视点和工具。
- 开发基线业务架构和制定目标业务架构。
- 执行差距分析。
- 定义候选路线图组件。
- 解决整个架构格局的影响。
- 进行正式的涉众评审,并最终确定业务架构(架构模式)。
- 编制架构定义文档。

使用 TOGAF 进行业务架构分析,先从“识别战略”开始,即 3.3 节所讨论的“需求获取”,了解各业务部门的需求、痛点和期望。然后,进行因素分析,包括外部因素分析和内部因素分析。

- 外部因素分析:包括宏观背景、当前市场趋势、行业空间(即市场潜力和增长空间等)、竞争情况(产品功能、市场定位和用户反馈)、上下游依赖关系等。
- 内部因素分析:包括商业模式、技术壁垒、资源投入等。

我们可以尝试让大模型生成业务架构图,包括识别业务角色、理清场景、绘制流程、列出模块、功能聚类,并分别从业务角色、业务流程的角度分析各模块的功能需求、各模块之间的依赖关系。例如,在业务流程上,针对 TMS 系统,需要业务架构能详细描述任务创建、分配、跟踪等流程,并进行上下游(或各个模块之间的)依赖关系分析。分析任务依赖关系和优先级的冲突,提供解决方案。

基于上述讨论,可以让大模型为 TMS 系统生成基于 TOGAF 的业务架构图,如图 3-20 所示。

5. Zachman 框架

Zachman 框架是由 John Zachman 提出的一种分类和描述企业架构的矩阵框架。它以二维表格的形式,从不同视角(行)和不同抽象层次(列)来描述企业的各方面。虽然 Zachman 框架主要用于企业架构,但它也适用于业务架构,提供了一个不同层面和视角下的结构化方式,通过多个视角(如业务、数据、功能等)帮助我们理解组织的业务架构,帮助我

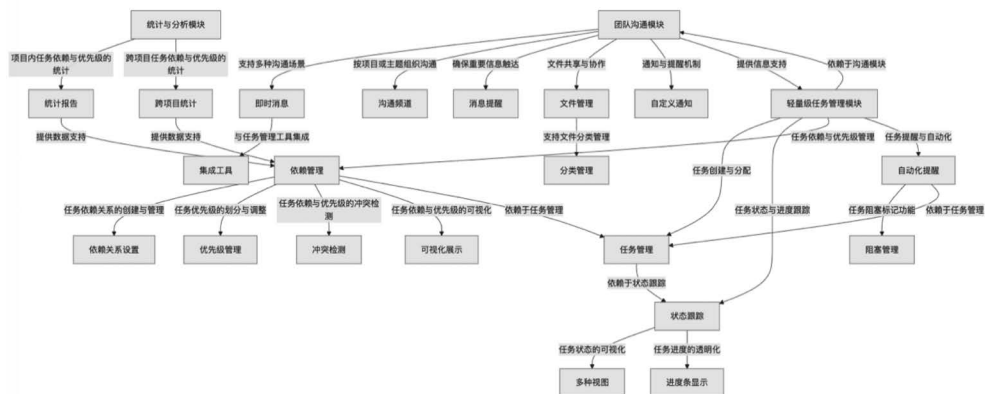


图 3-20 基于 TOGAF 生成的 TMS 业务架构图

们关注组织的业务角色、业务流程、业务服务、信息流等,特别适合面向复杂系统构建企业架构。例如,我们会从下面两个维度去构建企业业务矩阵框架。

- 参与者(涉众): 分为计划人员、所有者、设计师、实现人员、分包人、用户,分别对应范围/上下文、业务概念、系统逻辑、技术、物理、组件组装和操作类,代表了在信息系统构造过程中所涉及的人在描述信息系统时所采用的视角。
- 6 个视角(1W5H),用于描述信息系统的某一方面: 即什么内容(What)、如何工作(How)、何处(Where)、谁负责(Who)、什么时间(When)、为什么做(Why)。

这样形成 36 个单元格的矩阵,每个单元格都关注企业的一维透视图,如表 3-5 所示。

表 3-5 Zachman 框架(矩阵模型)

	数据 (What)	功能 (How)	网络 (Where)	人 (Who)	时间 (When)	动机 (Why)
范围(上下文)/ 规划者	重要事务 列表 实体:事务 分类	业务操作流 程列表 功能:业务 流程分类	业务运行 地点 节点:注意 业务场所	组织列表 人:主要 组织	重大事件 列表 时间:业务 事件	目标和战略 列表 主要目标和关 键因素
业务模型(概念)/ 拥有者	语义实体 模型	业务流程 模型	物流网络/ 系统	工作流模型	进度表	业务计划
系统模型(逻辑)/ 设计师	逻辑数据 模型	应用架构	分布式系统 架构	人机界面 架构	业务处理 结构	业务规则模型
技术模型(物理)/ 构建者	物理数据 模型	系统设计	技术架构	展现/表示 架构	控制结构	规则设计
详细展现功能模 型/组件分包人	数据定义	程序设计	网络架构	安全架构	时间/周期 定义	规则规范
产品操作/企业 用户	转换的数据	可执行程序	通信设备	培训组织	企业业务	标准

6. BPMN

BPMN(Business Process Model and Notation)是一种图形化表示业务流程的标准。它提供了丰富标准化的符号和规则,用于描述业务流程的逻辑流程和操作。通过流程图形式,清晰地展示业务流程的各个环节和逻辑关系。通过 BPMN 模型,识别和优化业务流程中的

瓶颈和低效环节。

价值流映射(Value Stream Mapping)是一种精益管理工具,通过价值流图,清晰地展示价值流的各个环节和瓶颈,也可以帮助我们识别并消除非增值活动、优化价值流。

服务蓝图(Service Blueprint)是一种可视化工具,用于描述和优化服务交付过程中的各个环节,包括前台和后台活动。服务蓝图关注客户体验,确保服务交付过程能够满足客户需求。

RACI(Responsible, Accountable, Consulted, Informed)矩阵是一种责任分配矩阵,用于合理分配或明确业务流程中各个角色的职责和权限,确保每个环节都有专人负责,避免责任重叠和冲突。

小 结

在软件开发过程中,定义清晰且准确的需求是确保项目成功的关键。第3章围绕需求的层次、获取、分析、定义和评审等方面进行了深入探讨,提供了系统化的方法和工具,以帮助团队在复杂的业务环境中有效地管理软件需求。

首先,需求层次的划分是理解和管理需求的基础。需求通常分为业务需求、用户需求和系统需求。业务需求关注的是组织的战略目标和业务流程,用户需求则强调最终用户的期望和使用场景,而系统需求则具体描述了系统应具备的功能和性能。这种层次化的需求结构有助于团队在不同层面上进行深入分析,确保所有需求都与业务目标保持一致。

其次,需求获取是需求工程中的重要环节。有效的需求获取方法包括访谈、问卷调查、观察和工作坊等。通过与利益相关者的互动,团队可以收集到真实的需求信息。在这一过程中,使用大模型辅助需求工程的能力,可以通过自然语言处理技术分析用户反馈,提取潜在需求,从而提高需求获取的效率和准确性。此外,需求获取的过程应当是迭代的,随着项目的推进,需求可能会发生变化,因此需要定期回顾和更新。

在需求分析与定义阶段,团队需要对收集到的需求进行整理和优先级排序。使用工具如需求矩阵和优先级评估模型,可以帮助团队识别关键需求并进行合理的资源分配。此时,需求的可追溯性也显得尤为重要,确保每个需求都能追溯到其来源,便于后续的变更管理和影响分析。需求定义是将需求转换为具体的功能规格和技术要求的过程。此阶段需要使用标准化的需求文档模板,确保需求描述的清晰性和一致性。通过使用用户故事、用例和功能规格说明书等工具,团队能够更好地传达需求意图,减少误解的可能性。

最后,需求评审是确保需求质量的重要环节。通过组织需求评审会议,团队可以邀请利益相关者对需求进行讨论和反馈,确保需求的完整性和合理性。评审过程中,使用大模型辅助的工具可以快速分析需求的可行性和潜在风险,帮助团队做出更明智的决策。

综上所述,我们通过对需求层次、获取、分析、定义和评审的系统化探讨,提供了一套完整的需求工程方法论。这些方法和工具,我们以团队协作系统(TMS)这个案例进行了具体展示,有利于读者更好地理解本章的内容,积累需求管理的经验。通过有效的需求管理,团队能够更好地满足用户期望,推动项目的成功实施。

思考题

124

1. 描述软件需求如何支持企业的整体业务目标。在这一过程中,如何确保软件需求与组织的战略一致性?
2. 在分析用户角色时,应该考虑哪些因素? 如何识别不同用户角色的需求差异,并确保满足各类用户的期望?
3. 阐述功能需求与非功能需求的区别。为什么非功能需求对软件系统的成功至关重要? 给出具体示例。
4. 需求获取的过程有哪些关键步骤? 在实践中,如何选择合适的方法来获取不同类型的需求?
5. 在 3.4 节中讨论“去伪存真”的分析法时,如何确保需求的真实性和相关性? 你认为哪些方法能有效识别伪需求?
6. 在项目实施中,敏捷需求评审与传统需求评审的主要区别在哪里?
7. 在需求跟踪和变更管理中,你认为企业应该如何制定有效的需求变更管理策略,以减少对项目进度和成本的负面影响?

参考文献

- [1] IEEE Computer Society. Guide to the Software Engineering Body of Knowledge v4.0.
- [2] 金芝,等. 软件需求工程方法与实践[M]. 北京:清华大学出版社,2023.
- [3] 蔡贇,等. 用户体验设计指南:从方法论到产品设计实践[M]. 北京:电子工业出版社,2021.
- [4] Arora C, Grundy J, Abdelrazek M. 2024. Advancing requirements engineering through generative ai: Assessing the role of llms [J]. In Generative AI for Effective Software Development. Springer, 129-148.
- [5] Lubos S, etc. Leveraging LLMs for the Quality Assurance of Software Requirements [J]. In 2024 IEEE 32nd International Requirements Engineering Conference (RE). IEEE, 389-397.
- [6] Marques N, Silva R R, Bernardino J. 2024. Using ChatGPT in Software Requirements Engineering: A Comprehensive Review [J]. Future Internet 16, 6 (2024), 180. ACM Trans. Softw. Eng. Methodol. The Future of AI-Driven Software Engineering • 17.