Shattuckite

需求文档

SHADOC-002,SDP, 第五组

版本 *0.0-20-g2988d69*

表 1: **版本变更历史**

版本	提交日期	主要编制人	审核人	版本说明

Contents:

---

1 概述

本节讨论关于需求分析的元 (Meta) 信息或是方法论。

1.1 基本的原则

需求分析必须满足以下两个目标:

1. 清楚的描述客户需要什么
2. 为之后的设计工作奠定基础

1.2 需求与产品

在代码未交付前, 产品存在吗?

产品是存在的, 不过并不是以代码的形式存在, 而是以思想的形式存在。每一个程序, 在首次在计算机上运行前, 肯定在客户的脑袋里运行过无数次了。

> “需求”和“产品”是同构的, 开发团队最终递交的产品,

不过是换了一种计算机能够理解的方式来表达用户的需求。

需求就是运行在客户大脑的程序, 产品就是运行在计算机上的程序。开发团队的工作就是把需求“翻译”成产品。

1.3 需求文档的边界

从需求到产品, 是一个逐渐逼近的过程。开发团队会对从客户处得到的信息进行连续的处理和加工; 而这种处理和加工是分阶段的。不同阶段，处理和加工的产物不同。终极产物自然是“产品”, 而中间产物则包括各种各样的文档。

当团队成员打开 IDE 输入代码时, 毋庸置疑，我们正处于编码阶段。但是，我们该如何划分“需求分析”和“设计”这两个阶段?《需求分析》文档和《设计文档》的边界在哪儿?

我们将以是否涉及计算机和电子信息工程专业知识, 来作为《需求分析》文档和《设计文档》的边界。也就是在撰写本文档的过程中，我们会尽量避免出现那些无法向没有 EECS 学科背景客户解释的信息，而是用严谨的自然语言描述客户的需求, 并尽量使这种描述是易于进行后续的加工处理的。

1.4 需求挖掘

[#TODO 此处添加需求挖掘的过程, 表述一下我们将如何从用户处获取信息]

1.5 分析方法

本团队会从用户和数据两个维度对需求进行建模。

我们将分别在 [#TODO 需要添加交叉引用] 和 [#TODO] 详细讨论这两种分析方法的具体实践。

2 面向用户的需求建模

2.1 概述

面向用户的需求分析是对用户本身进行建模以及对用户提供信息的初次加工。

标准化用例

为了规范化文档的撰写, 方便使用自动化工具处理文档, 我们将使用标准化用例的方式对从不同用处获得的需求进行记录.

标准化用例将以 *场景*为基本单位进行记录。一条记录应当包含以下字段

用例名称

Key	`name`
类型	`string`

用简短的语言概述该用例。原则上该字段

1. 使用一个动宾短语
2. 不出现主语

一但用例的名称确定, 将不能修改。

参与用户

Key	`user`
类型	`string[]`

记录在本用例中, 与系统产生交互的用户。

由于可能会有多个用户同系统交互, 所以使用列表类型存储数据。

标识相同的用户在列表中多次出现，会被认为是逻辑上同属一类但物理独立的用户。在分析系统的权限问题时可能会出现这种情况。

参与用户能且仅能是在用户画像 一节中明确定义过的用户。

场景

Key	`scene`
类型	`string`

使用自然语言描述用户使用软件时的场景。该字段应当是若干句子的集合，描述用户

1. *Where* 身处何处?
2. *Which* 使用何种设备与系统进行交互?
3. *How* 用户如何与系统交互?(用户进行何种操作?)
4. *Why* 交互的动机是什么?
5. *What* 期望得到的结果是什么?

此外，该字段中可以补充对参与用户角色的描述。

异常

Key	`exception`
类型	`[[string,string,string]]`

描述用户无法得到期望的结果时的情况。

用一个三元组列表存储该字段。一个三元组表示一个异常，三个元素分别为异常描述和异常原因及异常反馈。

异常列表内可能会出现异常描述相同但异常原因不同的项。

前提场景

Key	prerequisites
类型	string[]

描述触发本场景的充分条件。该字段是用例名称 列表, 其中的值必须被明确定义过。

2.2 用户画像

本项目的用户包括 普通用户, 普通工程人员,“开发人员 “ 及 课程教授。本小节将对每一种用户分别讨论用户:

1. 特点
2. 诉求
3. 会与系统的何部分产生交互
4. 需求提交方法

普通用户

普通用户是本软件的最终使用者。普通用户没有任何 EE&CS 专业知识背景, 仅仅能够与手机和计算机进行基本的交互 (例如填写并提交表单, 通过 GUI 访问程序页面等)。普通用户主要关注:

1. 系统易于使用, 易于配置
2. 系统出错率低（或是系统出错后可以在不被察觉的情况下修复）

普通用户会和系统的终端产生交互。系统的全部在用户眼中只是手机或电脑上的 GUI 程序。

由于用户的特点, 普通用户仅能够使用模糊的自然语言描述理想中的系统该如何工作。我们需要通过访谈或是问卷的方式, 获取普通用户的需求数据, 并对这些数据进行整理与建模，然后记录为标准化用例。

在本项目的开发中, 会让文档撰写人员扮演普通用户的角色，然后自问自答。

普通工程人员

普通工程人员是在软件分销商或系统集成商的工作人员。根据经验来看, 中小型城市的工程人员, 并不总具有 EE&CS 专业背景知识; 相对普通用户, 他们能够付出更多的时间来学习系统的使用与配置, 他们接受使用更难于使用的接口以提高工作效率。这类用户主要关注:

1. 操作文档是否完善
2. 开发团队是否有完善的错误管理机制及错误修复的速度
3. 用于管理系统的接口效率是否足够高
4. 系统的部署过程是否足够简单

普通工程人员的需求是具有明确的目标导向的, 他们会使用严谨的自然语言描述自己的需求。我们可以通过访谈或是问卷的方式, 获取普通工程人员的需求数据, 然后在确保可实现性的情况下对信息稍作加工整理, 记录为标准化用例。

在本项目的开发中，会让不参与编码工作的团队成员充当普通工程人员的角色尝试为假想的用户部署本系统, 提出新的需求并直接记录为标准化用例。

开发人员

开发人员是开发团队之外，有兴趣为项目贡献代码或是想要修改该系统以适应额外需求的人员。开发人员拥有专业知识背景，且具有完全理解系统工作原理的能力。开发人员主要关注:

1. 测试代码是否完善
2. 系统的架构是否合理
3. API 文档是否完善
4. 代码质量

开发人员的需求将由开发团队本身, 在进行项目开发的过程种进行整理与记录。

课程教授

课程教授是特殊的用户, 是本项目名义上的委托人 (甲方)。课程教授将主要关注

1. 开发进度是否满足要求
2. 文档完善程度
3. 功能的完成度
4. 开发团队的管理与组织

课程教授的需求将通过开发任务说明的相关文档来获取。由于课程教授的需求已经被合理的组织且记录在了《软件工程-嵌入式方向-大作业考核说明》这一文档中, 因此, 本文档将不讨论这一特殊用户的需求。

3 面向数据的需求建模

3.1 概述

基于数据的需求建模将聚焦于以下问题

- 数据分类
- 数据产生
- 数据流转
- 数据持久化

此外需要注明, 本节讨论的数据为 *用户数据*。*用户数据*可以直接被用户感知, 与用户产生交互。系统中存在用户可直接访问的，用于对 *用户数据*进行增/删/改/查的接口。

3.2 数据类型

用户数据可以被划分为两类。

业务数据

业务数据是和系统所提供的功能相关的数据。包括 (但不限于) 以下几类数据:

传感器数据

温度/湿度/空气质量等传感器采集到的物理量。

音视频数据

安防摄像头采集到的音视频二进制码流。

执行器数据

执行器的状态。例如电磁继电器触点的开关状态，线性导轨的位置等。

事件

事件是一种特殊数据, 当满足某些特定条件后会被触发。事件在逻辑上由物理设备 (例如传感器，执行器等) 产生。事件是主动产生的。

命令

命令是一种特殊数据, 命令可能由系统或者用户发出。

命令可以是主动产生的 (由程序逻辑产生)，也可以是被动产生 (由用户操作产生) 的。

引入命令的原因是为了使用户能够控制系统中的执行器。

配置数据

配置数据是和系统工作方式相关的数据。

采样率

对于类似于温度/湿度/空气质量传感器等以固定间隔产生时间离散数据的装置, 用户应当可以配置其产生两个数据的间隔。

事件触发条件

设备触发事件的条件。举例说明，用户应当可以配置当某一温度传感器的温度高于或低于设定阈值的时候，该传感器产生一个事件。

事件回调

特定事件触发后应当采取的动作。

持久化策略

储存数据的方式。对于传感器，用户可能希望数据库获取新数据的频率低于数据产生的频率，以在保证实时性的前提下减少存储负担。

对于音视频数据，用户可能只希望保留某一时段的记录文件。

账户数据

系统应当支持多用户管理。用户的基本数据, 例如用户名，密码，用户, 权限等, 应当被合理的存储。

QA

命令和事件有什么不同

事件 描述系统状态变化, 重点是 **发生了何种变化**; 命令 描述一个操作, 重点是 **想要系统如何变化**

命令和配置数据有什么不同

命令 是暂时性的, 在命令被接收设备处理后就不再生效; 配置数据 是持久性的, 在下次更改配置前，当前配置始终生效。

3.3 数据产生

业务数据产生

业务数据主要会从真实的物理设备或是软件系统本身产生.

物理设备产生的数据

传感器数据，音视频数据 以及执行器数据 均是由真实物理设备产生的。

软件系统产生的数据

事件 和命令 均是由软件系统本身产生的。