软件工程

Software Engineering

极夜酱

目录

Chapter 1 需求工程

1.1 软件开发

1.1.1 软件开发

什么是好的软件？一个好的软件需要满足三点要求：

1. 一般要求（general requirements）
2. 法律要求（legal requirements）
3. 具体要求（specific requirements）

一般要求包括软件系统需要有详细的文档、可读性高的代码、易于修改和维护的代码、能够开展测试、容易移植、使用方便等方面。

法律要求表示系统必须要遵守相关的法律法规。例如加拿大安大略省制定了残疾人无障碍法案，要求软件必须考虑照顾到残障人士的使用；加拿大反垃圾邮件立法规制了垃圾邮件、间谍软件、恶意软件和僵尸网络等问题。

具体要求即来自客户和用户具体的需求，软件系统应该要能够实现客户和用户提出的功能。

想要开发出一款好的软件，必须要拥有良好的编程能力，这可以通过大量练习提高。同时也需要具备良好的沟通和规划能力，例如在处理问题时，应该与客户或者熟悉问题的人沟通，确保理解了客户/用户的需求。编程前也可以先查找别人已经做过的相关内容，不要重复造轮子。

1.2 需求（Requirements）

1.2.1 用户故事（User Story）

用户故事是用几句话描述用户会做的一件事，以及他们为什么要这样做。用户故事是关于用户将会做什么，而不是怎么做。

用户故事一般采用这样的格式："As a [user type], I want [some action] so that [some reason]."。例如，"作为一个活跃用户，我想不用每次登录都输入账号密码，以便更方便快速地登录。"

1.2.2 需求

需求描述了软件做的某一件事，它可以是功能性（functional）或非功能性（non-functional）的。同样一个需求只需要描述做什么，而不是如何去做。

需求的作用是为了让开发者和客户都清楚最终产品想要的预期是什么。

一个好的需求应该具备以下条件：

- 分类（categorized）：使用 MuSCOW 法则对需求进行分类。
  - MUSTS：系统能够满足客户的基本需求，类似于最小可行产品（minimum viable product）
  - SHOULDS：满足客户基本或更高级的需求
  - COULDS：能够使系统变得更好的额外需求
  - WON'TS：系统不该做的事情
- 有优先级（prioritized）：对不同的需求指定一个优先级，需要遵守 MUSTS > SHOULDS > COULDS > WONT'S。
- 逻辑合理：需求之间的依赖关系需要合理，例如一个 MUST 需求不能依赖于一个 SHOULD 需求。

- 时间估计（time estimate）：一个需求应该需要在 15 天内完成，如果不能完成，应该把它分解成多个小需求。

1.3 原型设计

1.3.1 原型设计（Prototyping）

原型是指没有功能的模型，它不一定要包含一个程序的所有方面。

原型设计是与客户/用户沟通的一种工具，让客户/用户感觉他们自己也是设计者。原型设计的目的是为了验证 UI 是否能够可以被正常使用，让开发者能够以用户的角度思考问题。

1.3.2 纸上原型（Paper Prototyping）

纸上原型也称低保真原型（low fidelity prototyping），通过纸笔等工具设计，通常是 UI 设计的第一步。它具有制作过程快速、成本低、更改快速、效率高的特点。

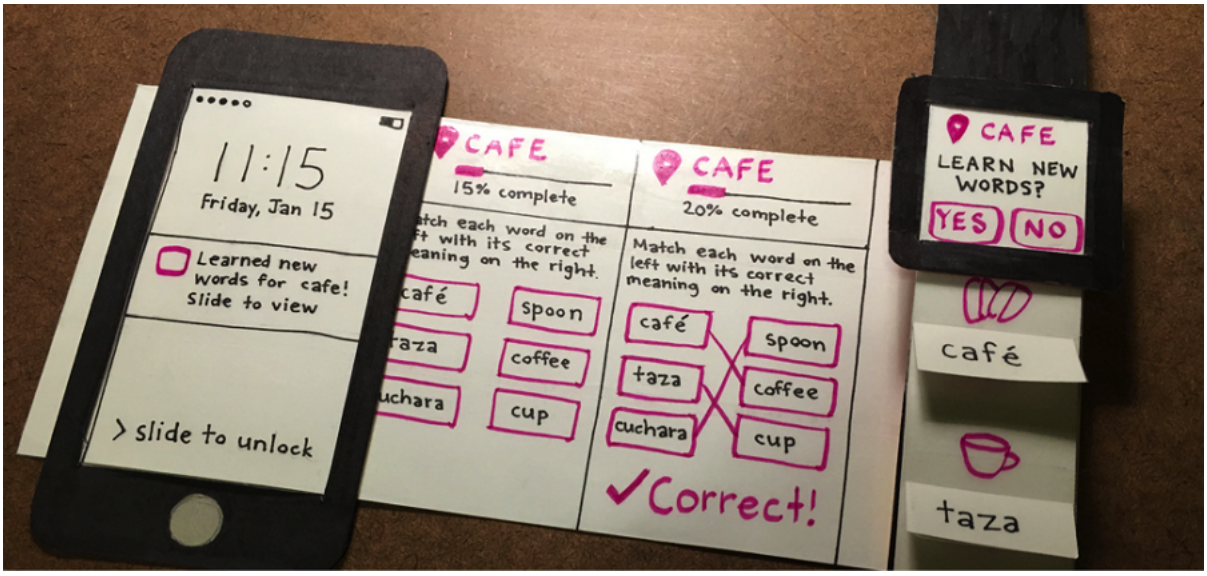

图 1.1: 纸上原型

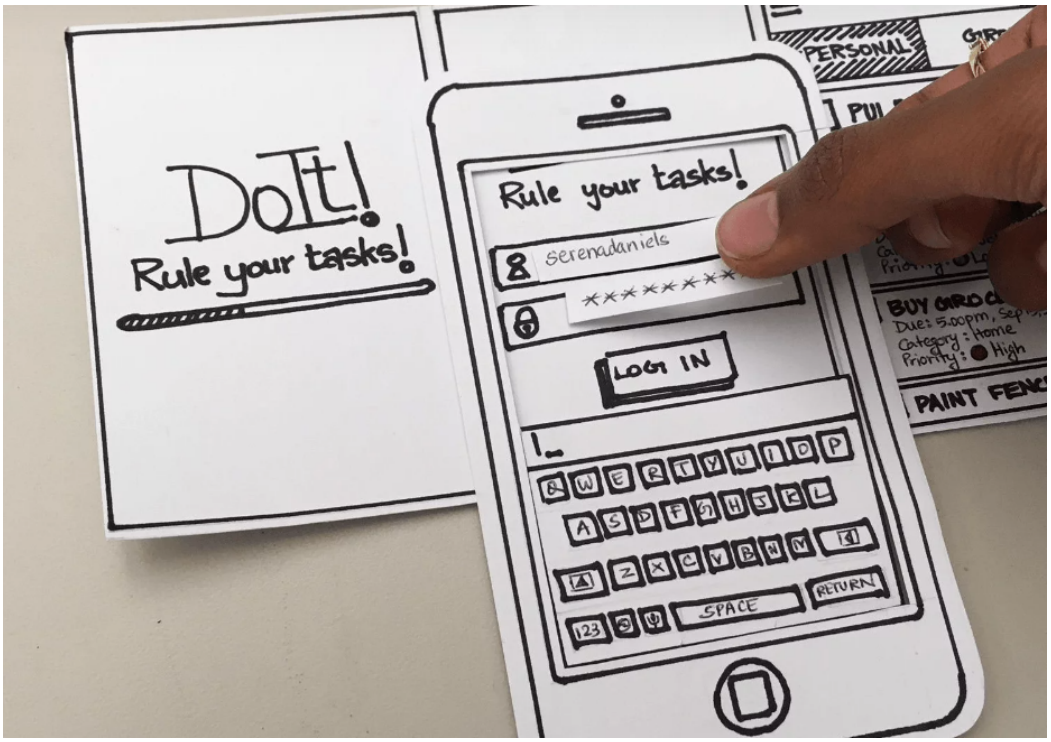

图 1.2: 纸上原型

纸上原型的目的并不是为了让 UI 更加美观，而是为了确保系统的流程是否存在问题。

在完成纸上原型后，需要让参与测试的人根据用例进行测试。在测试过程中，不要提供任何帮助或者指导，以此来验证用户交互是否流畅。

最后，记录下在纸上原型测试中哪些地方是正常的、哪些地方存在问题或遗漏的。

1.3.3 线框图（Wireframing）

线框图也称高保真原型（high fidelity prototyping），它看上去非常接近最终产品，几乎完全按照实物来制作，原型中甚至包含产品的细节、真实的交互、UI 等。

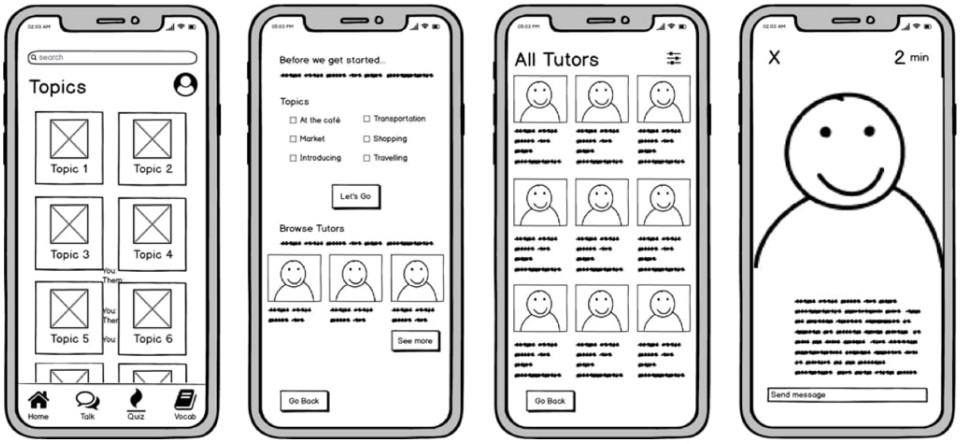

图 1.3: 线框图

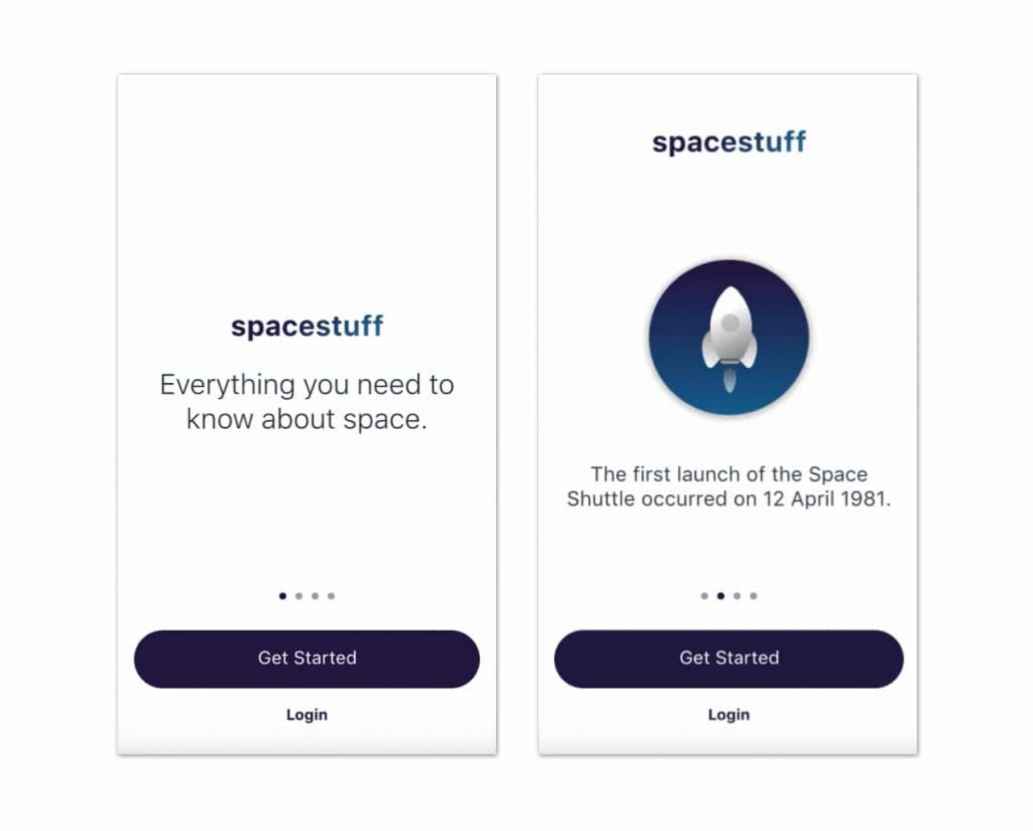

图 1.4: 线框图

高保真原型的目的在于让用户提供反馈，例如 UI 中控件的位置和 UI 的美观程度。