第 12 章 SE 负责监管计算机系统的构建

热身问答

在阅读本章内容前，让我们先回答下面的几个问题来热热身吧。

初级问题

SE 是什么的缩略语?

中级问题

IT 是什么的缩略语?

高级问题

请列举一个软件开发过程的模型。

怎么样？被这么一问，是不是发现有一些问题无法简单地解释清楚呢？下面，笔者就公布答案并解释。

答案

初级问题：SE 是 System Engineer（系统工程师）的缩略语。

中级问题：IT 是 Information Technology（信息技术）的缩略语。

高级问题：软件开发过程的模型有“瀑布模型”“原型模型”“螺旋模型”等。

解释

初级问题：在计算机系统的开发过程中，SE 是参与所有开发阶段的工程师。

中级问题：一提到 IT，通常就意味着充分地运用计算机解决问题，但 Information Technology（信息技术）这个词中并没有包含表示计算机含义的词语。

高级问题：本章将会详细地介绍应用瀑布模型的开发过程。

本章重点

从第 1 章到第 11 章，讲解的都是各种各样的计算机技术。在作为本书最后一章的第 12 章，请允许笔者再介绍一下将这些技术组合起来构建而成的计算机系统，以及负责构建计算机系统的 SE（System Engineer，系统工程师）。本章不仅有技术方面的内容，更会涉及商业方面的内容。对于商业而言，没有什么可称得上是绝对正确的见解，因此本章的叙述中也多少会含有笔者的主观想法，这一点还望诸位谅解。

“将来的目标是音乐家!”——正如以前新出道的偶像歌手都会有这句口头禅一样，过去新入行的工程师也有一句口头禅，那就是“将来的目标是 SE !”在那时 SE 给人的印象是计算机工程师的最高峰。可是最近，想成为 SE 的人似乎并没有那么多了。不善于与客户交谈，感到项目管理之类的工作很麻烦，觉得穿着牛仔裤默默地面对计算机才更加舒坦等原因似乎都是不想成为 SE 的理由。SE 果真是那么不好的工作吗？其实不然，SE 是有趣的、值得去做的工作。下面就介绍一下身为 SE 所需要掌握的技能以及 SE 的工作内容吧。

12.1 SE 是自始至终参与系统开发过程的工程师

所谓的 SE 到底是负责什么工作的人呢?《日经计算机术语辞典 2002》（日经 BP 出版社）中对 SE 做出了如下的解释。

> SE 指的是在进行业务的信息化时，负责调查、分析业务内容，确定计算机系统的基础设计及其详细规格的技术人员。同时 SE 也负责系统开发的项目管理和软件的开发管理、维护管理工作。由于主要的工作是基础设计，所以不同于编写程

序的程序员，SE 需要具备从硬件结构、软件的构建方法乃至横跨整个业务的广泛知识以及项目管理的经验。

简单地说，SE 就是自始至终参与系统开发过程的工程师，而不是只负责编程的程序员。所谓系统，就是“由多个要素相互发生关联，结合而成的带有一定功能的整体”。将各种各样的硬件和软件组合起来构建而成的系统就是计算机系统。

至今为止，有些业务依然是靠手工作业进行的，引进计算机系统就是为了提高这类业务的效率。SE 在调查、分析完手工作业的业务内容后，会进行把业务迁移到计算机系统的基本设计，并确定详细的规格。SE 负责的工作是项目管理和软件开发管理，以及引进计算机系统后的维护，而制作软件（编程）的工作则交由程序员完成。

也就是说，SE 是从构建计算机系统的最初阶段（调查分析）开始，一直到最后的阶段（维护管理）都会参与其中的工程师。比起只参与编写程序这一工作的程序员，SE 所参与的工作范围更加广泛。为此，SE 就必须掌握从硬件到软件再到项目管理的多种多样的技能。

表 12.1　SE 所需的技能和程序员所需的技能

职业	工作内容	所需技能
SE	调查、分析客户的业务内容 计算机系统的基本设计 确定计算机系统的规格 估算开发费用和开发周期 项目管理 软件开发管理 计算机系统的维护管理	倾听需求 书写策划案 硬件 软件 网络 数据库 管理能力
程序员	制作软件（编程）	编程语言 算法和数据结构 关于开发工具和程序组件的知识

12.2 SE 未必担任过程序员

正如其名，SE 虽然也是工程师，但他们并不同于孜孜不倦地处理具体工作的专业技术人员。可以说 SE 是一种更接近“管理者”的职业，负责管理技术人员。若以建设房屋为例，程序员就相当于木匠，而 SE 则相当于木匠师傅或是现场监理。但是请不要误解，SE 未必比程序员的职务高。从职业规划上来说，也不是所有的程序员将来都会成为 SE。

确实有人是从程序员的岗位转到了 SE，二十几岁时是程序员，三十几岁时当上了 SE。但是也有人是从 SE 的新手成长为 SE 的老手，二十几岁时担任小型计算机系统的 SE，三十几岁时担任大型计算机系统的 SE。说到底 SE 和程序员是两个完全不同的职业。在企业中，若说 SE 部门有一条从负责人到科长再到部长的职业发展路线，那么程序员部门自然也会有一条与之相应的从负责人到科长再到部长的职业发展路线。

但是在现在的日本，几乎已经找不到还在制作 OS（Operating System，操作系统）或 DBMS（Database Management System，数据库管理系统）这类大型程序的企业了，所以企业中程序员部门的规模通常都不大，多数情况下是隶属于 SE 部门或其他管理部门的，甚至有时企业还会把整个编码工作委托给外包公司。因此，很少有人能够以程序员的身份升迁到部长的职位，从而造成了程序员成为 SE 的下属这样的现状。

12.3 系统开发过程的规范

上一节我们提到 SE 是从最初的阶段直至最后的阶段，自始至终都参与构建计算机系统的工程师。而本节将要讲解的是计算机系统是由怎样的开发过程构建而成的。无论任何事都需要规范，即便未能按其实践，规范的存在也算是一种参考。这里介绍的有关计算机系统开发

过程的规范叫作“瀑布模型”。如图 12.1 所示，在瀑布模型中要进行 7 个阶段的开发。虽然实际开发中可能未必如此，但规范毕竟是规范。

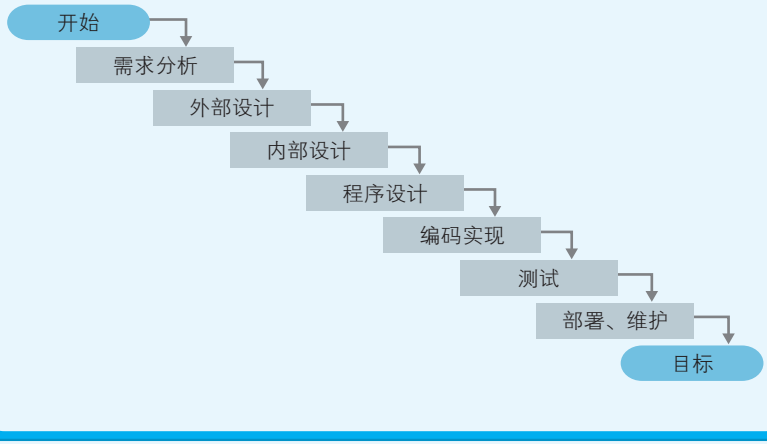

图 12.1　瀑布模型中的 7 个阶段

在瀑布模型中，每完成一个阶段，都要书写文档（报告）并进行审核。进行审核时还需要召开会议，在会上由 SE 为开发团队的成员、上司以及客户讲解文档的内容。若审核通过了，就可以从上司或客户那里得到批准，继续进入后续的开发阶段。若审核没有通过，则不能进入后续的阶段。一旦进入了后续的阶段，就不能回退到之前的阶段。为了避免回退到上一阶段，一是要力求完美地完成每一个阶段的工作，二是要彻底地执行审核过程，这些就是瀑布模型的特征。这种开发过程之所以被称为“瀑布模型”，是因为开发流程宛如瀑布，一级一级地自上而下流动，永不后退。如图 12.2 所示，开发过程就好像是开发团队乘着小船，一边克服着一个又一个的瀑布（通过审核），一边从上流顺流而下漂向下游。而坐在船头的人当然就是 SE 了。

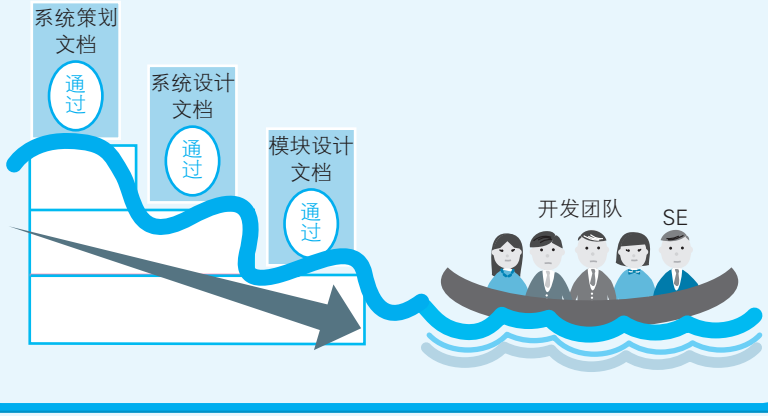

图 12.2 瀑布模型的示意图

12.4 各个阶段的工作内容及文档

下面介绍瀑布模型各个阶段的工作内容及所要书写的文档的种类（如表 12.2 所示）。

表 12.2 各个阶段所要书写的文档

阶段	文档
需求分析	系统策划文档、系统功能需求规格文档
外部设计	外部设计文档
内部设计	内部设计文档
程序设计	程序设计文档
编码实现	模块设计文档、测试计划文档
测试	测试报告
部署、维护	部署手册、维护手册

在“需求分析”阶段，SE 倾听将要使用计算机系统的客户的需求，调查、分析目前靠手工作业完成的业务内容。作为本阶段的成果，SE 要书写“系统策划文档”或是“系统功能需求规格文档”。

接下来是设计计算机系统，该过程可以分为 3 个阶段。虽然看起来有些啰嗦，但规范终归是规范。第一个阶段是“外部设计”，进行与从外部观察计算机系统相关的设计。设计内容包括系统处理的数据、显示在画面上的用户界面以及打印机打印的样式等。第二个阶段是“内部设计”，进行与从内部观察计算机系统相关的设计。内部设计的目的是将外部设计的内容具体化。在计算机行业中常会提及“外部”和“内部”，一般情况下，把从用户的角度看到的东西称为“外部”，把从开发者的角度看到的东西称为“内部”。也许这样说会更容易理解，外部设计设计的是用户看得到的部分，而内部设计设计的是开发者看得到（用户看不到）的部分。第三个阶段是“程序设计”，为了用程序将内部设计的内容实现出来而做出的更加详细的设计。作为以上 3 个设计阶段的结果，SE 要分别书写“外部设计文档”“内部设计文档”和“程序设计文档”。

再接下来，就进入了“编码实现”阶段，要进行的工作是编写代码，由程序员根据程序设计文档的内容，把程序输入到计算机中。只要经过了充分的程序设计，编程就变成一项十分简单的工作了。因为所做的只是把程序设计书上的内容翻译成程序代码。作为本阶段的文档，SE 要书写用于说明程序构造的“模块设计文档”和用于下一阶段的“测试计划文档”。这里所说的模块，就是拆解出来的构成程序的要素。

到了“测试”阶段，测试人员要根据测试计划文档的内容确认程序的功能。在最后编写的“测试报告”中，还必须定量地（用数字）标示出测试结果。如果只记录了一些含糊的测试结果，比如“已测试”或是“没问题”，那么就难以判断系统是否合格了。

在定量地标示测试结果的方法中，有“涂色检查”和“覆盖测试”等方法。“涂色检查”的做法是一个个地确认“系统功能需求规格文档”中的功能，如果该功能实现了，就用红笔把它涂红。“覆盖测试”则是

一种表示有多少代码的行为已经经过确认的方法。“通过涂色检查，已确认了系统 95% 的功能。剩下的 5% 虽然有问题，但已经查明了原因，可以在 1 周内修正”“已完成了 99% 的覆盖测试。由于剩余的 1% 是不可达代码（Dead Code，绝不会被执行的代码），所以可以删除”。如果能像这样给出定量的测试结果，那么就很容易判定系统是否合格了吧。

如果测试合格了，就会进入“部署、维护”阶段。“部署”指的是将计算机系统引进（安装）到客户的环境中，让客户使用。“维护”指的是定期检查计算机系统是否能正常工作，根据需要进行文件备份或根据应用场景的变化对系统进行部分改造。只要客户还在使用该计算机系统，这个阶段就会一直持续下去。在这一阶段要书写的文档是“部署手册”和“维护手册”。

12.5 所谓设计，就是拆解

下面先请诸位回到图 12.1 所示的瀑布模型，从上游到下游再回顾一遍该模型中的各个开发阶段。从需求分析到程序设计，所进行的工作都是拆解业务，把将要为计算机系统所替代的手工业务拆解为细小的要素。从编码实现到部署、维护阶段，所进行的工作则是集成，把拆解后的细小要素转换成程序的模块，再把这些模块拼装在一起构成计算机系统。

庞大复杂的事物往往无法直接做出来。这个道理不仅适用于计算机系统，也同样适用于建筑物或是飞机。人们往往要把庞大复杂的事物先分解成细小简单的要素来进行设计。有了各个要素的设计图，整体的设计图也就出来了。先根据每个要素的设计图制成小零件（程序中的模块），待每个小零件的测试（单元测试）都通过了，剩下的就只是一边看着整体的设计图，一边把这些零件组装起来了。然后再来一轮

测试（集成测试），测试组装起来的零件是否能正确地协作运转。大型的计算机系统就是这样构建出来的（如图 12.3 所示）。

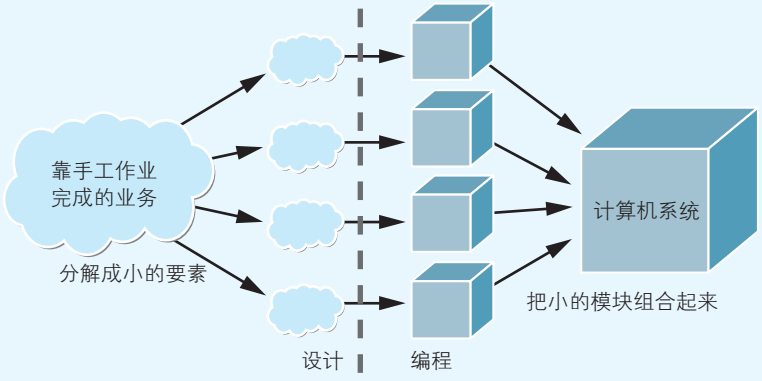

图 12.3　把小的模块组装起来，构成大的计算机系统

可以说，所谓计算机系统的设计，就是拆解。老一辈工程师们已经发明出了可作为规范的各种各样的设计方法，这些方法之间的差异只是拆解时的关注点不同。这里先把几个具有代表性的程序设计方法列在表 12.3 中。

表 12.3　具有代表性的程序设计方法

设计方法	拆解时所关注的事物
通用功能分割法	在整个计算机系统中通用的功能
STS 法	数据流（输入、变换、输出）
TR 法	事务（数据的处理单位）
Jackson 法	输入数据和输出数据
Warnier 法	输入数据
面向对象法	构成计算机系统的事物（对象）

STS: Source，Transform，Sink
TR: Transaction