第3章 快速原型方法

目标：

(1) 了解快速原型方法产生的原因。

(2) 掌握快速原型方法的概念。

(3) 掌握快速原型方法的适用范围。

3.1 快速原型方法的产生

瀑布模型对软件开发工程化做出了重要贡献，但是在实际使用过程中，仅当软件需求明确，工作能够采用线性方式完成时，瀑布模型才是一个很有用的过程模型。一旦需求不明确时，会遇到如下问题。

(1) 客户通常难以清楚地描述所有的需求。在没有实际系统呈现在客户面前时，客户无法表达细致的需求。

(2) 瀑布模型的顺序在实际项目中难以遵循。虽然线性模型可以加入迭代，但它是用间接方式实现的，结果是随着项目的推进，产生的线性模型变更可能带来混乱。

(3) 任务之间产生阻塞状态。瀑布模型的线性特性在某些项目中会导致“阻塞状态”，开发团队的一些成员要等待另一些成员完成相关任务，这样花在等待上的时间可能会超过花在生产上的时间，在线性过程的开始和结束这种阻塞状态时更容易发生。

(4) 客户无法提前接触系统。因为只有在项目接近尾声时，客户才能得到可执行的程序，对于系统中存在的重大缺陷，如果在可执行程序评审之前没有被发现，将可能造成惨重的损失，也不利于客户提早熟悉系统的使用和数据的初始化。

(5) 文档编写的工作量甚至超过程序编写的工作量。为了使文档清晰，写文档的创意、难度和耗时都不亚于编代码，这样瀑布模型将使很多项目的成本翻倍。

对于硬件来说，当机械工程师接到一个设计任务后，通常会根据要求和自己的理解，在较短的时间内按一定的比例设计并制造一个样机，交给用户确认后再成批投产，这台样机可以称为原型。同样的道理，软件工程师也可以根据初步的需求理解，在短时间内开发一个系统原型交给客户确认，边确认边开发，这样就产生了一种新的软件开发方法——快速原型方法。

3.2 快速原型方法的概念

快速原型方法的思想是先开发出一个原型系统给用户使用,通过用户反馈意见来不断修改系统直到最后成熟(见图 3-1)。它不主张将描述、开发和有效性验证等活动分开进行,而是让这些活动并行执行,同时让这些活动都能得到快速的反馈信息。

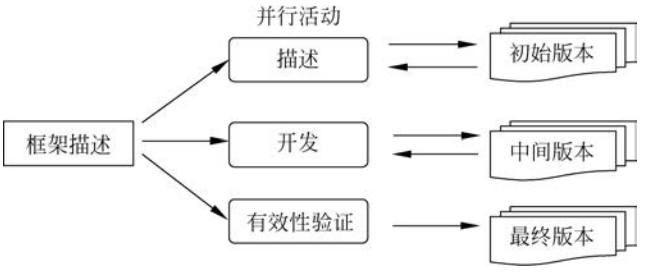

图 3-1 快速原型方法

系统原型是软件系统的初始版本,它可以用来展示一些概念,给出设计选择,发现问题及其可能的解决方案。快速原型开发是非常关键的,为的是有效地控制开发成本,而且开发人员可以较早地在原型系统上验证自己的设计。

快速原型方法打破了瀑布型方法的僵化,让各个开发过程之间和同一开发过程中的主客体之间提前融合。

(1) 融合了需求、设计、编程和单元测试等阶段,在这些阶段之间形成了一个闭环,需求驱动设计和编程,单元测试结果反过来验证需求,并诱导新的需求。这样产生如下好处。

① 导出需求。系统原型允许用户在上面实验,以便了解系统是如何支持他们工作的,在这个过程中,用户可能产生有关需求的许多新的想法,同时发现系统的优点和不足,进而提出新的系统需求。

② 验证需求。原型系统可以暴露出错误和遗漏的东西。一个经过描述的功能可能是很有用且已经是定义的,但是当这个功能模块与其他模块一起工作时,用户可能会发现他们的初始想法是错误的或是不完善的,必须修改系统描述以反映对需求的新的理解。

③ 降低风险。原型开发可以用作风险分析和降低风险的技术。软件开发中一个重要的风险来自于需求错误和需求遗漏,在后期弥补需求错误的成本是非常高的。

④ 降低成本。原型开发能减少需求描述中出现问题的数量,而且总的开发成本在有原型系统的情况下要比没有原型系统时低。

(2) 融合了开发者、系统、客户三者之间的关系,让客户提前参与到系统的开发、测试和使用当中。这样产生如下好处。

① 消除主客体之间理解的偏差。软件开发人员和用户之间的理解偏差在功能展示时显露出来,软件开发小组可能会在原型设计中发现需求的不完善和不一致。

② 支持用户培训。让用户更早地知道系统的工作方式,可以迅速展示一个应用系统对客户业务管理的可行性和作用。在最终的系统交付使用之前,原型系统可以用于用户的培训。

③ 支持文档的书写。原型可以用作书写产量和质量系统描述的基础。

④ 提高系统的实用性、可维护性和设计质量。

用原型法来提高系统的实用性和更好地满足用户需要并不一定意味着开发成本的提高。原型开发在初期阶段是表现为成本的增加，但在后期却表现为成本的降低。主要原因在于客户提出的变更减少了，因而开发中的返工现象大大减少。然而原型开发也有一些负面效果，表现在系统复用了效率较差的原型代码，而这些代码导致了整个系统性能的降低。

在区分原型开发是作为一个独立的活动还是作为主流的软件开发方法的问题上，在过去的许多年中一直比较模糊。现在许多系统是用快速原型方法来实现的，初始的版本被很快开发出来之后，经过修改和不断完善形成最终的系统。

快速原型方法和瀑布型方法的区别和联系可以总结如下。

(1) 瀑布型方法将系统的开发看成是一个整体的任务，需求分析和定义、系统和软件设计、实现和单元测试、集成和系统测试、运行和维护这些过程只有一次循环；快速原型方法将系统的开发分解成多个独立和相关的任务，需求分析和定义、系统和软件设计、实现和单元测试、集成和系统测试、运行和维护这些过程多次循环。

(2) 瀑布型方法提出的过程在快速原型方法中不是一次整体使用，而是分成多次任务分别使用，这就是瀑布型方法饱受批评而又始终存在的原因，人们一直在意识里和实践中灵活使用它。

(3) 快速原型方法是瀑布型方法的灵活使用，瀑布型方法是学院派的成果，快速原型方法是实业界的成果。

快速原型方法有如下两类。

(1) 进化式开发。进化式开发的目标是与用户一起工作，共同探索系统需求，直到最后交付系统。这类开发是从需求较清楚的部分开始，根据用户的建议逐渐向系统中添加功能。

(2) 抛弃式开发。抛弃式开发的目标是理解用户需求，然后再给出系统的一个较好的需求定义。这类开发往往从对客户需求理解较差的那部分开始。

快速原型方法的开发过程模型如图 3-2 所示。在开发过程的开始就要明确原型开发的目的，或是对用户界面的原型设计，或是对系统功能需求进行有效性验证，还可能是为了说明应用系统管理上的可用性。一个原型不可能满足上述所有的目的。如果对建立原型的目的还很模糊，管理部门和最终用户就会误解原型的功能，导致他们无法从原型开发中获得期待的益处。

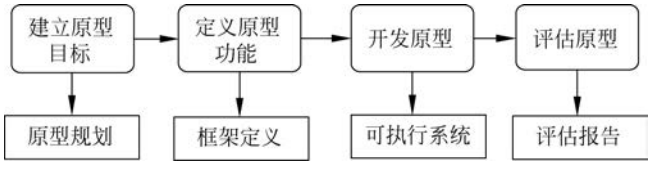

图 3-2　快速原型方法的开发过程

开发过程的下一步就是确定哪些东西要加到原型系统中，哪些应该从原型中去除。为了降低原型开发的费用和加快开发速度，需要抛开一些功能模块，而且可以降低一些性能要求，如响应时间和内存耗费，同时对错误处理和管理可以忽略或是做简单处理。除非设计原

型的目标是建立用户界面,否则可靠性标准和程序质量也不予考虑。

最后阶段的工作是原型的评估。Ince 和 Kekmatpour 建议把这项工作作为原型设计最重要的工作来抓。对用户培训的有关规定要在这一阶段给出,同时要基于原型的设计目标导出评估计划。用户需要一定的时间来适应新系统并逐渐使使用方式变得规范化。一旦使用方式规范了,他们就能较容易地发现需求上的错误和纰漏。

要让最终用户提前知道新系统将如何支持他们的日常工作是一件很困难的事情。如果该系统规模很大且复杂,那就更困难一些。

解决这个困难的一种方式是采用快速原型的系统开发方法,即在系统尚不完善时就呈现给用户,边修改边完善,在完善过程中逐渐把需求弄明白;另一种方式是采用抛弃式原型开发方法进行需求分析和有效性验证,评估一结束,就抛弃原型,重建一个完善的系统。

3.2.1 进化式原型开发

进化式原型开发的基本思路是:先给出一个系统的最初实现,让用户去使用和评论,再不断进行细化和完善,经过多个这样的反复过程后形成最后完善的应用系统,如图 3-3 所示。这种开发方法最初用于像人工智能系统那样难以描述的系统。目前这种方法已经成为软件开发的一种主流技术。

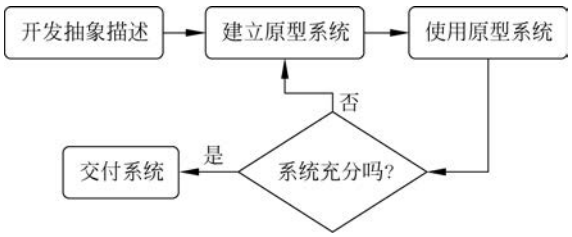

图 3-3 进化式原型开发

进化式原型开发方法对大规模、长周期的系统开发而言是最为重要的方法。在使用这种方法时要注意以下三个主要问题。

(1) 管理问题。建立大型系统软件管理机构以处理软件过程模型。软件过程模型定期产生可交付的文档来评估项目的进展状况。原型的开发太快,以至于产生大量的系统文档,这样很不经济。而且快速原型开发可能需要一些不熟悉的技术,管理者会觉得现有的开发班子使用起来有困难,因为他们缺乏这些技术。

(2) 维护问题。连续不断地对原型进行修改很可能导致系统结构的崩溃,这意味着如果某个开发人员不是从一开始就参与了该项目,他很可能难以理解系统。再者,如果快速原型开发中使用了某项专门的技术,这种技术可能会过时,不再被人们使用,这样以后再寻找具有相关知识的人来维护系统就变得十分困难。

(3) 合同问题。客户和软件开发商之间正规的合同模型是基于系统描述的。没有这样的描述,就很难拟定一个有关系统开发的合同。如果一份合同只约定开发时间和按照这个时间应付给承包商的费用,相信客户是不会满意的,因为这样可能导致系统功能滑坡和预算超支。开发者也不愿意接受一个固定价格的合同,因为他们无法控制最终用户不断改变的需求。

这些困难意味着使用进化式原型开发技术要有一个现实的态度，允许从一个小型或中等规模的系统开始做起，以便有一个较短的交货期。要把开发成本降下来，尽量提高可用性。如果用户参与到开发中，那么原型就会很贴近真实的需求。但开发单位必须意识到系统的生命周期可能会相对缩短。随着维护问题的增加，系统可能不得不被替换或彻底重写，尤其是在一个大型的软件项目中，可能有许多子承包商，进化式原型开发中的管理问题就变得难以驾驭。

3.2.2　抛弃式原型开发

基于抛弃式原型开发的软件过程模型如图 3-4 所示。这种方法在降低总的生存期成本（在产品经济有效使用期间所发生的与该产品有关的所有成本）的情况下增强了需求分析过程。在这种开发方法中，原型的作用是弄清楚需求和为管理人员评估过程风险提供额外的信息。经过评估，原型被抛弃，不再作为系统开发的基础。

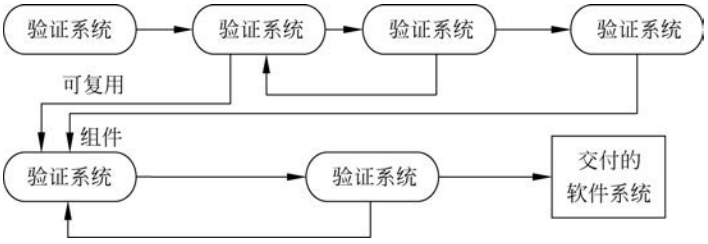

图 3-4　抛弃式原型开发

这种系统原型的方法最常用于硬件系统中。在决定进行一个相当昂贵的系统生产之前，原型被用作设计验证。一个电子系统原型往往利用现成的组件来做，而在决定投产时再制作专门用途的集成电路以实现该系统。

然而，抛弃式软件原型通常不作为设计有效性验证，而是用于获取系统需求，原型与最终系统相去甚远。这些原型必须尽快拿出来，以使用户能够尽早反馈对系统描述的意见。用户只对抛弃式原型中的功能感兴趣，这些功能经过原型设计而得到深刻理解，但质量标准和性能指标在原型中被忽略。原型开发与最终系统开发使用的语言也往往不一样。

图 3-4 中的过程模型假设原型是从粗略的系统描述开始的，接着进行交付实验，然后再修改，直到用户对其功能满意为止。在这一阶段，阶段性的过程模型被采用，从原型中提炼需求，而最后系统却要重新建立。原型中的组件也许会用于最终系统中，这样能够降低一些开发成本。

原型除了能导出系统描述外，有时原型实现本身就是系统描述。客户往往对承包商这样要求："照这个系统给我做一个。"抛弃式原型方法存在以下问题。

（1）为了尽快拿出原型，系统可能做了许多简化，因而不可避免地会遗漏一些重要特性。事实上，有些对安全要求极高的系统很难用原型来表现其中某些重要部分。

（2）在客户和承包商之间没有一个能写进合同的对于原型实现的合法规定。

（3）非功能需求，如可靠性、鲁棒性和安全性，在原型实现中不会得到充分反映。

开发一个可执行的抛弃式原型通常要遇到的问题是：原型的使用方式和最终系统的使

用方式可能不一样。一般测试人员都对系统非常了解,他们不代表系统用户。对原型评估所做的培训工作可能是不够的。如果原型运转缓慢,评估人员可能调整他们的工作方式,以避免那些费时的操作,在最终系统中他们可能用另一种模式进行操作。

开发人员有时受到来自管理者的压力,不得不交付抛弃式原型给用户使用,尤其是在最终版本交付推迟的情况下。然而这样做是不明智的,原因如下。

(1) 在原型开发过程中,不太可能使原型的非功能需求,如性能、保密性、鲁棒性、可靠性等满足需要。这些指标在整个开发过程中常被忽略。

(2) 在开发过程中,快速变更的结果必然是没有文档。仅有的设计描述就是原型代码,这不足以应付长期的系统维护。

(3) 原型开发中的变更有可能破坏系统的结构。这样,系统的维护将很困难,费用也很高。

(4) 机构内的质量标准对原型往往不加限制。

抛弃式原型不一定必须是在需求工程过程中很有用的可执行的软件原型。纸上的用户界面模型在帮助提炼界面设计和设计使用情景时依然表现得十分有效。这种原型非常经济,而且几天之内就可以完成。这个技术的延伸是"Oz 向导[沃兹原型(Wizard of Oz prototyping)]"的原型方法,该方法只用于开发用户界面。用户与界面间的交互传递给某个人,由他来翻译并以合适的方式输出。

图 3-5 进化式和抛弃式原型开发

进化式原型开发和抛弃式原型开发方法的两点重要区别如图 3-5 所示。

(1) 进化式开发的目标是给用户一个实用的系统。这就意味着原型开发必须从对用户需求把握最准的部分做起,最优先处理这部分工作;而对用户需求把握程度较差的部分和模糊的需求安排得稍后些,可以在用户有明确要求之后再处理。

(2) 抛弃式原型开发的目标是验证和导出需求。此时应该从理解得不够好的那部分需求开始实现,因为你的目标是要从中发现问题,对明确的需求就没有必要去做原型。

这两种开发方法的另一个重要区别是在系统的质量管理方面。抛弃式原型从定义上可以看出其具有较短的生命周期,从原型向正式系统的转换必须要快,不需要长期的维护。差的系统性能和可靠性是可以接受的,只要对理解需求有帮助。

进化式原型开发从一些主要的简单需求开始,在对原型的讨论过程中不断发现新的需求,添加新的功能,逐步完善原型,最终该原型就变成了一个完善的、满足所有需求的系统。这种开发方法自始至终都不需要详细的系统描述,而且在多数情况下也没有形式化的需求文档。进化式原型开发方法目前已经成为基于 Web 系统和电子商务系统的常用开发方法。

相比之下,抛弃式原型方法的目标是帮助提炼和澄清系统描述。原型过程主要有制作、评估和修改三个阶段。评估结果作为进一步进行系统描述修改的依据,经过评估和修改的过程,最后把系统描述定型于系统需求文档中。一旦需求文档描述完成,原型就不再使用,而是被抛弃。

3.3 快速原型方法的案例

下面介绍使用快速原型方法开发多媒体课件的案例。

学习是一个不可预测的过程，在不同人和不同情景下表现各不相同，因此教学软件的需求分析阶段是较为困难的，一般不可能第一次就能得到恰当的分析结果。原型可以对教学策略进行较早期的评价，可用于用户界面的设计和导航设计，这两点对于教学软件是特别重要的。原型还能够有效沟通开发人员和教学设计人员及课件用户之间的思想，是实现教学思想、教学经验与计算机技术统一和结合的基础，符合教学设计和软件过程天生的重复和迭代特征。同时，原型也是进行教学试用、教学效果评价的最基本条件，它为教学软件的快速、高质量开发起到不可替代的作用。

1. 利用多媒体制作工具实现课件原型的一般步骤

(1) 快速分析。根据要开发教学软件的学科教学特点和教学要求，决定原型要着重设计的方面，然后选定合适的多媒体制作工具，同时根据原型的使用目的确定采用抛弃式还是进化式原型，一般推荐使用进化式原型。

(2) 构造原型。根据设计要求，利用选定的多媒体制作工具制作出所选定教学内容的教学模块的外观模型(最初原型)。

(3) 运行评价。运行原型，通过学科教师、教育专家及学生的检验、评价和测试，针对原型提出修改意见和需求。

(4) 修正改进。根据反馈信息修正和完善原型，直至符合教学需求。

(5) 完善原型。整理原型并提供文档，为软件下一步的运行、开发服务。

2. 一个例子——传统课程的电子化

快速原型方法应用于教学软件开发最常见的例子就是传统课程的电子化工作，传统课程的学科教师在教育教学方面已经积累了相当丰富的教学设计经验，因此转换的工作只需要用系统的方法学来指导以经济有效地进行，一旦学科教师决定开发现有课程的多媒体教学软件，具体的转换过程就涉及以下几个相关阶段。

(1) 设定教学目标，分析教学对象，给出课程的详细说明。首先，多媒体课件的教学目的和课堂教学的目的不太相同，应审视传统课堂教学中的目标：在班级教学中有哪些不能实现的目的？多媒体课件能达到这些目的吗？试着发掘计算机能达到而班级上课时不可能实现或者不能经济实现的那些目标，并且要了解多媒体课件的局限性，比如缺乏面对面的接触等，应认真现实地考虑这些目标。其次，课件的使用者不同于上课时班级中的学生，他们之间可能在背景知识、知识需求等多方面存在较大差异，并且在课件使用中是学习者自己控制学习的进程。最后，以比较详细的方式描述课程，包括章、节、练习、实践活动等各个方面，这些都需要认真地进行教学设计。

(2) 素材转换、收集和原型的建立。素材的转换和收集包括一系列从简单到复杂的活动，其中部分具有创造性要求，可能需要创建一些模板或其他类似的可复用部件，使用制作工具制作出初始的界面原型，然后按教学设计的要求逐渐演化为初始功能原型。

(3) 评估原型。在课程的原型版本完成以后,甚至没有完成时,应使用实际的学习者进行测试,不仅要测试他们对课程的主观反应,还要判断在学习过程中是否达到了教学目标。

(4) 反馈修改。判断是否达到软件所期望达到的目标,确定可以改进的具体办法,反复进行,直到一切目标能良好完成。

3. 应注意的几个问题

基于多媒体制作工具的特点,人们可以利用多媒体制作工具快速高效地完成教学软件原型的制作。但制作工具和原型的实现过程中也具有一些不足,这就要求在制作软件原型的过程中,应重点考虑以下一些问题。

(1) 如何选择合适的制作工具。不同的多媒体制作工具适用于不同的领域,对于教育工作者、学科专家及设计人员来说,合适的教学设计方案解决以后,就面临着如何恰当地选取制作工具。通常在选取制作工具时应考虑以下问题:课件的发布场合;使用哪些种类的媒体;课件的交互水平;成绩数据追踪;课件内容和容量;开发者技术水平等。即要详细分析开发项目的需求特点,如制作的多媒体软件是演示型的,还是交互型的,又或是百科全书式的电子读物等,然后根据项目的需求特点,选择合适的制作工具,好的制作工具不应当使用户将目光局限于工具本身的特点,而是能够允许用户将更多时间投入到概念和教学设计层次上,以能设计出更具魅力的教学软件。大多数教育专家认为 Macromedia 公司的 Authorware 是开发交互式多媒体教学软件的最佳工具,事实上,它也是最被广泛使用的。

(2) 使用原型开发仍应注意遵循软件工程原则。使用制作工具进行原型开发过程中,不像一般软件那样有明显的阶段性,它是一个迭代反复的过程,文档的生成与管理、设计的表述等各个方面要注意保持一致性、完备性,还应遵循国家和国际相关标准。

(3) 原型评价的具体实施。教学软件无法仅通过界面布局及简单可运行的原型版本来评估学习的效果,因为简单原型几乎无法真正工作在完成教学目标的这层含义之下。因此,对于原型版本的评价必须经过仔细计划,初期界面原型、教学试用原型要分别进行评价,评价可通过大量问卷或实际使用方式进行,评价人员既要包括软件用户(学科教师、学生),也要包括教育学、心理学及美工等方面的专家。试用评价应是一个长期不间断的过程,贯穿于整个教学软件生存期。

思考题

1. 快速原型方法产生的原因是什么?
2. 什么是快速原型方法?
3. 快速原型方法的优缺点是什么?
4. 快速原型方法的适用范围是什么?