原型法和面向对象的分析与设计方法（1） PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/141/2021_2022__E5_8E_9F_E5_9E_8B_E6_B3_95_E5_c29_141250.htm 原型法是在20世纪80年代中期为了快速开发系统而推出的一种开发模式，旨在改进传统的结构化生命周期法的不足，缩短开发周期，减少开发风险。原型法的理念是：在获取一组基本需求之后，快速地构造出一个能够反映用户需求的初始系统原型，让用户看到未来系统概貌，以便判断哪些功能是符合要求的，哪些方面还需要改进，不断地对这些需求进一步补充、细化和修改，依次类推，反复进行，直到用户满意为止并由此开发出完整的系统。 面向对象是近20年来国内外IT行业最为关注的技术之一，面向对象技术是一种按照人们对现实世界习惯的认识论和思维方式来研究和模拟客观世界的方法学。它将现实世界中的任何事物都视为“ 对象 ”，将客观世界看成是由许多不同种类的对象构成的，每一个对象都有自己的内部状态和运动规律，不同对象之间的相互联系和相互作用就构成了完整的客观世界。面向对象方法（Object Oriented,简称OO方法）克服了传统的功能分解方法只能单纯反映管理功能的结构状态、数据流程模型只侧重反映事物的信息特征和流程、信息模拟只能被动地迎合实际问题需要等缺点，构成以系统对象为研究中心，为信息管理系统的分析与设计提供了一种全新的方法。 在本章中将详细介绍原型法的提出背景和缘由、基本思想、基本步骤、关键成功因素以及和生命周期法的比较；介绍面向对象的基本概念和原理，面向对象的信息系统分析、设计与实施方法。一、原型法1.1 原型法的提出 20世

纪60年代末至70年代初，出现了“ 软件危机 ”，为了对软件开发项目进行有效管理，信息系统开发生命周期法诞生了。由于开发过程规范、层次清晰，系统开发生命周期法得到广泛应用。但这种方法的应用前提是需要在早期就确定用户的需求，而不允许修改，这对于很多应用系统（如商业信息系统）来说是不现实的。用户需求定义方面的错误是信息系统开发中出现的后果最严重的错误。在此背景下，提出了基于循环模型的快速原型法。1．原型法的提出背景“ 软件危机 ”出现于20世纪70年代初，“ 软件危机 ”的表现为：软件开发速度满足不了实际需求，软件成本在计算机系统总成本中所占比例逐年上升，软件产品的质量不可靠，软件难以维护，没有适当的文档资料，开发进度难以控制。 产生“ 软件危机 ”的原因在于：用户需求不明确，缺乏正确的理论指导，软件规模越来越大且复杂度也越来越高。那么如何解决“ 软件危机 ”呢？人们越来越重视软件开发方法的研究，通过多年的研究和努力，软件开发方法走向两个方面：一方面是着重研究与机器本身相关的软件开发工具，即高级语言及软件开发环境；另一方面，着重研究软件设计和规格说明等。这时系统开发生命周期（Systems Development Life Cycle , SDLC ）应运而生。它是一种用于规划、执行和控制信息系统开发项目的组织和管理方法，是工程学的原理在信息系统开发中的具体应用。 正如第三章介绍，生命周期法是一种结构化方法，把信息系统开发视为一个生命周期，把软件看作是人工制品，必然有其产生、成长、成熟、运作、消亡的生命过程。生命周期法把系统开发分为多个阶段，一般分为五个阶段：系统规划、系统分析、系统设计、系统实施。系统运行与

维护。严格按阶段进行，每个阶段都有明确的目标和任务。每一阶段完成以后，要完成相应的文档资料，作为本阶段工作的总结，也作为下一阶段的依据。这种方法特别强调阶段完整性和开发的顺序性，它要求开发者首先确定系统的完整需求和全部功能。 生命周期法具有明显的优点。它采用系统观点和系统工程方法，自顶向下进行分析与设计并自下而上进行实施。开发过程阶段清楚，任务明确，并有标准的图、表、说明等组成各阶段的文档资料。生命周期法引入了用户观点，适用于大型信息系统的开发，将逻辑设计与物理设计分开。 但是，生命周期法的应用前提是严格的需求定义方法和策略。需求定义(the Definition of Requirement )方法是一种严格的、预先定义的方法。从理论上讲，一个负责分析设计的项目小组应完全彻底地预先指出对应用来说是合理的业务需求，并期待用户进行审查、评价和认可，并在此基础上顺利开展工作。 这种严谨的需求定义方法是在一定假设的前提下形成的，它们是： (1)所有的需求能被预先定义 这一假设的确切含义是，在没有系统实际工作经验的情况下，所有的系统需求在逻辑上是可以预先说明的。在某种情况下，虽然不能保证项目参加者个人都能确知系统需求和逻辑模型，但通过大多数人对系统的建议和合理判断，完全可以描述一个明确的系统需求，所有需求都能被准确预先定义。 但实际情况，需求定义方法假设的有效性是比较脆弱的。现实中，往往提供详细说明材料的人不是本领域的专业权威和职业分析人员；去定义复杂度甚高的事情又是十分困难的；大多数用户绝非面面俱到，只能是有选择性的说明。即使预先定义工作做得很好，往往系统仍旧需要进一步地修改和经过若干次反复

，这是因为以下的事实是经常存在的。①个人对系统的认识往往与实际不完全吻合；②实地观察和使用系统会刺激用户对系统提出新的需求；③观看和经历会修改和取消对系统的事先需求。 (2)项目参加者之间能够清晰而准确地通信 严格需求定义方法的又一项重要假设是：在系统开发的进程中，项目组、项目经理、分析人员、用户开发人员、审计人员、保密分析员、数据管理员、人际关系专家等都能够清晰而有效地进行通信。虽然每个人都有自己的专业、观点和行动，但用图形／描述文档等工具，使得大家可能得到清晰、有效的沟通。 而实际情况往往是复杂的，对于共同的约定，每个人往往会有自己的解释和理解，对规格说明上应该有而尚未有的规定和说明，会有各种意见或加进个人看法。而文字叙述，如英语或汉语及其它文字描述，并非是一种准确的通信工具，即使提供了结构化的文字语言，如结构化英语以及判定表、树等较严格的通信的高级方式，当然这较叙述性的文字描述肯定是一种改进，减少了模糊性，但它仍然缺乏精确的技术上的通信语言的“ 严密性 ”、“ 专业性 ”和“ 行业性 ”。 因此，在多学科、多行业人员之间架起通信的桥梁是困难的。人们早就认识到，相互间通信的有效性的损失乃是开发过程中失败的主要原因之一。 （3）静态描述／图形模型对应用系统的反映是充分的 使用预先定义技术时，主要的通信工具是定义报告，包括工作报告和最终报告。采用叙述文字、图形模型、逻辑规则、数据字典等形式，这些具体形式因各自的技术有所不同，但其作用是相似的。 所有技术工具的共同特点是：它们都是被动的通信工具和静止的通信工具，不能表演，因而无法体现所建议的应用系统的动态特性，而

要求用户根据一些静态的信息和静止的画面来认可系统则似乎近于苛求。 因此，严格定义技术本质上是一种静止、被动的技术，要它们来描述一个有“ 生命 ”的系统是困难的。理解和评价一个应用系统的最好方式，应该是去体验它，而不仅仅是去阅读和讨论它。 综合上述各点可见，严格需求定义的合理性在许多情况下并不满足，因此建立在脆弱基础上的开发策略在实施中一旦导致系统的失败就绝非意外之事。为了更好地处理由于缺乏支持严格方法的假设而给项目带来的风险，需要探求一种变通的方法。 解决需求定义不断变化问题一种思路是在获得一组基本的需求后，快速地加以“ 实现 ”。随着用户或开发人员对系统理解的加深而不断地对这些需求进行补充和细化。系统的定义是在逐步发展的过程中进行的，而不是一开始就预见一切，这就是原型法。 