进度管理：软件生命周期 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/272/2021_2022__E8_BF_9B_E5_BA_A6_E7_AE_A1_E7_c41_272394.htm 软件生命周期(SDLC，软件生存周期)是软件的产生直到报废的生命周期，周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段，这种按时间分程的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。但随着新的面向对象的设计方法和技术的成熟，软件生命周期设计方法的指导意义正在逐步减少。 一、软件生命周期(SDLC)的六个阶段 1、问题的定义及规划 此阶段是软件开发方与需求方共同讨论，主要确定软件的开发目标及其可行性。 2、需求分析 在确定软件开发可行的情况下，对软件需要实现的各个功能进行详细分析。需求分析阶段是一个很重要的阶段，这一阶段做得好，将为整个软件开发项目的成功打下良好的基础。"唯一不变的是变化本身。"，同样需求也是在整个软件开发过程中不断变化和深入的，因此我们必须制定需求变更计划来应付这种变化，以保护整个项目的顺利进行。 3、软件设计 此阶段主要根据需求分析的结果，对整个软件系统进行设计，如系统框架设计，数据库设计等等。软件设计一般分为总体设计和详细设计。好的软件设计将为软件程序编写打下良好的基础。 4、程序编码 此阶段是将软件设计的结果转换成计算机可运行的程序代码。在程序编码中必须要制定统一，符合标准的编写规范。以保证程序的可读性，易维护

性，提高程序的运行效率。 5、软件测试 在软件设计完成后要经过严密的测试，以发现软件在整个设计过程中存在的问题并加以纠正。整个测试过程分单元测试、组装测试以及系统测试三个阶段进行。测试的方法主要有白盒测试和黑盒测试两种。在测试过程中需要建立详细的测试计划并严格按照测试计划进行测试，以减少测试的随意性。 6、运行维护 软件维护是软件生命周期中持续时间最长的阶段。在软件开发完成并投入使用后，由于多方面的原因，软件不能继续适应用户的要求。要延续软件的使用寿命，就必须对软件进行维护。软件的维护包括纠错性维护和改进性维护两个方面。 二、软件生命周期模型 任何软件都是从最模糊的概念开始的：为某个公司设计办公的流程处理；设计一种商务信函打印系统并投放市场。这个概念是不清晰的，但却是最高层的业务需求的原型。这个概念都会伴随着一个目的，例如在一个"银行押汇系统" 的目的是提高工作的效率。这个目的将会成为系统的核心思想，系统成败的评判标准。99年政府部门上了大量的OA系统，学过一点Lotus Notes的人都发了财（IBM更不用说了），但是更普遍的情况是，许多的政府部门原有的处理模式并没有变化，反而又加上了自动化处理的一套流程。提高工作效率的初衷却导致了完全不同的结果。这样的软件究竟是不是成功的呢？ 从概念提出的那一刻开始，软件产品就进入了软件生命周期。在经历需求、分析、设计、实现、部署后，软件将被使用并进入维护阶段，直到最后由于缺少维护费用而逐渐消亡。这样的一个过程，称为"生命周期模型"（Life Cycle Model）。 典型的几种生命周期模型包括瀑布模型、快速原型模型、迭代模型。瀑布模型（Waterfall Model

）首先由Royce提出。该模型由于酷似瀑布闻名。在该模型中，首先确定需求，并接受客户和SQA小组的验证。然后拟定规格说明，同样通过验证后，进入计划阶段…可以看出，瀑布模型中至关重要的一点是只有当一个阶段的文档已经编制好并获得SQA小组的认可才可以进入下一个阶段。这样，瀑布模型通过强制性的要求提供规约文档来确保每个阶段都能很好的完成任务。但是实际上往往难以办到，因为整个的模型几乎都是以文档驱动的，这对于非专业的用户来说是难以阅读和理解的。想象一下，你去买衣服的时候，售货员给你出示的是一本厚厚的服装规格说明，你会有什么样的感触。虽然瀑布模型有很多很好的思想可以借鉴，但是在过程能力上有天生的缺陷。 迭代式模型 迭代式模型是RUP推荐的周期模型，也是我们在这个系列文章讨论的基础。在RUP中，迭代被定义为：迭代包括产生产品发布（稳定、可执行的产品版本）的全部开发活动和要使用该发布必需的所有其他外围元素。所以，在某种程度上，开发迭代是一次完整地经过所有工作流程的过程：（至少包括）需求工作流程、分析设计工作流程、实施工作流程和测试工作流程。实质上，它类似小型的瀑布式项目。RUP认为，所有的阶段（需求及其它）都可以细分为迭代。每一次的迭代都会产生一个可以发布的产品，这个产品是最终产品的一个子集。迭代的思想如上图所示。 迭代和瀑布的最大的差别就在于风险的暴露时间上。"任何项目都会涉及到一定的风险。如果能在生命周期中尽早确保避免了风险，那么您的计划自然会更趋精确。有许多风险直到已准备集成系统时才被发现。不管开发团队经验如何，都绝不可能预知所有的风险。"（RUP）二者的区别如下图

所示：由于瀑布模型的特点（文档是主体），很多的问题在最后才会暴露出来，为了解决这些问题的风险是巨大的。"在迭代式生命周期中，您需要根据主要风险列表选择要在迭代中开发的新的增量内容。每次迭代完成时都会生成一个经过测试的可执行文件，这样就可以核实是否已经降低了目标风险。"（RUP）快速原型（Rapid Prototype）模型是我喜欢采用的另一种模型。快速原型模型在功能上等价于产品的一个子集。注意，这里说的是功能上。瀑布模型的缺点就在于不够直观，快速原型法就解决了这个问题。一般来说，根据客户的需要在很短的时间内解决用户最迫切需要，完成一个可以演示的产品。这个产品只是实现部分的功能（最重要的）。它最重要的目的是为了确定用户的真正需求。在我的经验中，这种方法非常的有效，原先对计算机没有丝毫概念的用户在你的原型面前往往口若悬河，有些观点让你都觉得非常的吃惊。在得到用户的需求之后，原型将被抛弃。因为原型开发的速度很快，设计方面是几乎没有考虑的，如果保留原型的话，在随后的开发中会为此付出极大的代价。至于保留原型方面，也是有一种叫做增量模型是这么做的，但这种模型并不为大家所接受，不在我们的讨论之内。 上述的模型中都有自己独特的思想，其实现在的软件组织中很少说标准的采用那一种模型的。模型和实用还是有很大的区别的。 软件生命周期模型的发展实际上是体现了软件工程理论的发展。在最早的时候，软件的生命周期处于无序、混乱的情况。一些人为了能够控制软件的开发过程，就把软件开发严格的区分为多个不同的阶段，并在阶段间加上严格的审查。这就是瀑布模型产生的起因。瀑布模型体现了人们对软件过程的一

个希望：严格控制、确保质量。可惜的是，现实往往是残酷的。瀑布模型根本达不到这个过高的要求，因为软件的过程往往难于预测。反而导致了其它的负面影响，例如大量的文档、繁琐的审批。因此人们就开始尝试着用其它的方法来改进或替代瀑布方法。例如把过程细分来增加过程的可预测性。