正确看待项目开始的前期分析 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/232/2021_2022__E6_AD_A3_E7_A1_AE_E7_9C_8B_E5_c41_232552.htm 谈过RUP原则，也搞定了客户需求，一个软件开发项目总算从客户端转由开发团队掌控。「太好了，案子终于进来了，大家开始动手做吧！」项目领导人宣布。于是办公室灯火通明，准备与开发工程进行长期抗战。但是等一等！不是要先做分析吗？「分析？没问题，我们当然也做分析。」项目领导人如此回答，同时对某人喊道：「喂，给你三天时间…不，两天内把分析做好，大家等着赶工呢！」 这样的形容或许有些夸张，不过许多开发团队看待「前置阶段」（程序撰写之前的阶段）的态度大概是如此。因为大部分人认为：反正分析又不是真正的「生产」，何必投入太多人力与时间在这种没有实际贡献的事情上？ 分析不事生产吗？其实并非如此。您是否对在前两篇文章中不断提到的「1：200」成本概念记忆犹新？同样一个问题，当它发生在前面阶段与后面阶段时，所需要投入的解决成本平均比例大约是一比两百，所以您应该不难理解在「前置阶段」中分析的重要性。 相信绝大多数的开发人员都知道分析很重要，因为它位居承先启后的战略位置，任何一个小小的错误，都可能导致开发大业功亏一篑。因此，分析并非「不事生产」；相反的，我认为从分析阶段开始就是一种生产。 分析的目的 「分析」在RUP的定义是：「正确理解问题，把需求陈述转换成软件概念，并开始对所欲建置的系统发展出一个可视化模型，以便快速、毫无衔接障碍地进入程序设计等后续阶段。」为了达到这个目标，我建议采用以

下作法： 在RUP里，这个阶段的工作称为「对象导向分析」（Object Oriental Analysis），又名「视觉塑型分析」（Visualizing Modeling Analysis）。「对象导向分析」代表从分析阶段就开始建立正确的对象导向概念，而「视觉塑型分析」则彰显分析在此是一种可视化与模型化的过程。「对象导向」（Object Oriented，简称OO）是许多教科书都会介绍的程序语言概念，因此不再赘述。至于强调视觉感的「塑型」（modeling）则是一种新的观念与作法，在此特别说明。 Model是什么？ 什么是Model？Model就是对真实物品的简化呈现，它可以是汽车模型、飞机模型等实体模型，也可以是一份设计蓝图。不论如何，其目的都是利用「视觉」来代表真实物品。对RUP而言，「视觉塑型分析」也同时拥有上述两种意涵。希望把蓝图画出来，也期望透过这个过程创造「产品雏形」。 为何要做Modeling？理由很简单：因为「视觉图像」远较「文字陈述」更容易被理解，也更接近真实物品。所以，当开发过程与产品都被「视觉塑型化」之后，就产生以下几个好处： 首先，开发成员可以按照实际模型或蓝图进行讨论，避免各自以文字或想象进行沟通而可能产生的误解。第二，可以让客户验证产品雏形是否符合真正需求，避免需求被扭曲。第三，可以预先了解及检测产品的特色、行为与功能等，进一步掌控质量。最后，可以据此发展成为最终产品。 不过，你可能会问：「可视化？做模型？哪有那么多时间？干脆直接写程序算了！」当然，如果做模型也要一头钻入实际的程序代码撰写工作，确实是蛮累人的。不过，幸好有个叫做「UML语言」的好东西，可以帮助我们完成这些工作。 UML的威力 UML（Unified Modeling Language）是

一种可以把软件开发过程中的各种产物予以可视化、特定化、建构化与文件化的语言。是由Rational的Gardy Booch、Jim Rumbaugh与Ivar Jacobson三位对象导向领域的大师级人物于1994年所发展与公布的语言，随后并经全球各程序语言专家，以及包括IBM、HP、Microsoft、Oracle等业界大厂共同参与、制订及推动，目前已是OMG（Object Management Group；对象管理组织）的公开性标准语言，被广泛应用在跨领域的软件开发过程。 那么，UML可以做到什么事情呢？ 首先，UML可以「可视化（Visualizing）」系统及系统架构。因而能够把需求、问题、行为等概念或文字描述，转换成各种互有关连的「图形」，让开发成员可以清楚知道系统的各种详细架构，让大家得以在共同基础上沟通，避免误解。 第二，UML可以「特定化（Specifying）」一个模型。也就是可以建立一个精准、毫不模糊及完整的模型，帮助进行特定功能或行为的追踪、检测与控管，确保它们不会在后续阶段被模糊或失焦。 第三，UML可以「建构（Constructing）」真正的程序代码。可以直接把UML语言「对应转换（Mapping）」成Java、C、VB等真正的程序代码或数据库，所以产品雏形就能制造出来。更棒的是，它不仅提供这种「正向工程」，也允许从程序代码转回UML的「逆向工程」。 第四，UML将整个系统架构及开发流程「文件化（Documenting）」。因为UML除了可以把整个系统及系统架构予以可视化，产生各种互有关连的大量图表外，还会瞄准所有图表的运作与互动细节，帮助掌控从需求、项目计划、测试到产品成型的所有开发流程都遵循标准作业。 从以上说明可以发现，UML可说是软件分析工程的最大利器，也是整个软件开发过程的坚强

后盾。大家都知道，软件系统日趋庞大复杂，而且往往也不是单独存在，必须与其它既有或未来的系统互动沟通，这些都让系统架构的复杂度大幅提升，不能单靠想象或文字进行描述与沟通；应该让UML发挥其强大的威力。 所以，只要会利用UML，谁能说「分析」不事生产呢？ 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com