## 当你要接手一个项目时，首先要搞清楚的是什么？

首先要搞清楚客户的需求

在项目的早期阶段，需求对每一个人来讲，都是含糊的，形式也是混乱的。客户并不总是

擅长于准确描述他们想要什么或需要什么，并且我们并不总是擅长于理解他人的业务含义。客户聊解他们自己的业务，但是并不总是能够向其组织外的人描述他们的业务问题。他们的描述充满了行话，并且假定一些理所当然的内容，而我们可能对此并不熟悉。同样，作为软件开发人员，我们熟知计算机解决方案，但并不总是知道可能的解决方案将会对客户的业务活动产生怎样的影响。我们也同样使用行话，并做出假设，并且有时我们认为每一个人都说同样的语言，而实际上，对同样的词，人们也有不同的理解。只有通过同每一个与系统成败相关的人讨论需求，将这些不同的观点合并成一致的一组需求，并且与风险承担者一起评审这些文档，才能使所有人都对需求是什么达成一致（见补充材料4-3表述的另一种观点）。如果我们不能就需求达成一致，项目注定要失败。

## 当你了解一个由成百上千个文件、上万行代码规模的项目时，首先要了解的是结构还是需求？

首先要了解需求 , 原因 : 由于项目非常大 , 所以项目结构大概率会很复杂 , 如果不清楚项目需求也就是不明白项目是做什么所以可能会很难理解项目的结构 , 但如果了解了项目的需求, 再看项目结构可能会清晰明了的多, 而且从代码的结构可能会对项目有新的理解 .

新接触的项目第一步应是快速深入了解整个项目。快速而深入了解一个项目的第一选择永远是阅读项目文档。一上来就看代码无异于盲人摸象。

你首先应从项目文档里面对整个项目有一个整体的认识，包括：

1.项目的目的

2.项目的环境

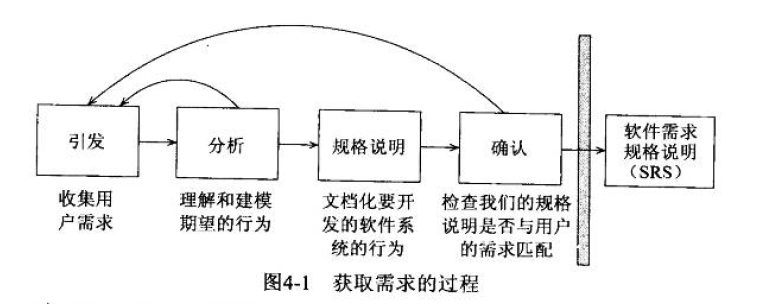
3.项目的功能模块划分

4.各功能模块的业务流程（或运行结果）

了解了以上信息之后，再去对应着各功能模块去看代码。

对于一般项目来说，代码上基本没有难度，算法研究类的另当别论。

## 获取项目需求的过程是怎样的？需求用什么工具建模？需求最终的文档是什么？



1. 建模工具 : ER图 , UML图 , 事件追踪图 , 消息时序图 , 状态机 , UML状态图 , Petri网 , 数据流图 , UML用例图 , 函数与关系 , 判定表 , Parnas表 , 逻辑 , 对象约束语言 (OCL) , Z, 代数规格说明 , SDL数据
2. 软件需求规格说明 即SRS文档

## 建模与原型化在需求问题上的区别

原型化模型是为弥补瀑布模型的不足而产生的。这种软件需求获取方式是首先快速建立一个能反映用户主要需求的原型系统,实现客户或未来的用户与系统的交互,用户试用后提出意见,开发人员按照用户意见快速地修改原型系统,然后用户再试用。再提意见。开发人员再去修改,如此反复迭代,直到用户满意为止。

原型化方法基本思想：根据用户提出的需求，由用户与开发者共同确定系统的基本要求和主要功能，并在较短时间内简历一个实验性的，简单的小型系统。

原型：优点。符合人们认识事物的规律，开发周期短，费用相对少，应变能力强。

缺点。不符合大型系统，开发难以控制，系统难以维护。范围。小型系统的开发 , 采用原型化的方法会增加成本。

建模

建模有助于我们通过梳理出应该询问什么问题，来透彻地理解需求。模型中的漏洞揭示出未知的或含糊不清的行为；同一个输入的多个、相互冲突的输出揭示出需求中的不一致性。随着模型的开发，我们不了解的问题以及客户不了解的问题会变得越来越清晰。不理解模型的主题，我们就不可能完成一个模型。此外，通过用与客户原始请求完全不同的方式来重新描述需求，可以促使客户为了确认模型的准确性，来仔细地检查我们的模型。