**常见论述题**

**索引**

概论

软件产品的特点：

软件就是程序这个观点正确吗？给出原因（2017）。

软件危机、软件危机的特点以及软件危机产生的原因：

什么是软件工程？软件工程的出现主要是为了解决什么？

什么是软件过程？软件过程和软件工程方法学有什么关系（书+真题）？

为什么说分阶段的生命周期模型有助于软件项目管理（书）？

什么是里程碑？它应该有哪些特征（书）？

开发一个软件，功能是把读入的浮点数开平方，所得到的结果应该精确到小数点后4位，一旦实现并测试完以后，该产品将被抛弃。你打算选用哪种软件生命周期模型？为什么？（书）

假如你是一家软件公司的项目经理， 职责是管理该公司已经被广泛使用的图片处理软件新版本的开发。由于市场竞争激烈，公司规定了严格的完成期限并且已经对外公布。你打算采用哪种软件生命周期模型？为什么（书、2017）？

在需求分析和设计阶段建立原型的好处有哪些（2016）？

螺旋模型和Rational统一过程有什么相似之处？差异（书）？

说明敏捷开发的适用范围（书）。

对于螺旋模型的理解？

举例CASE工具

结构化分析与设计

数据流图和数据字典的关系是什么（2017）？

为什么数据流图要分层？画分层DFD要遵循哪些原则（2015）？

情景与描述了所有可能序列的状态图之间有什么关系（书）？

在程序流程图中的每个节点都必须有一条从开始节点到该节点本身的路径，以及一条从该节点到结束节点的路径。为什么数据流图没有关于节点之间可达性的类似规则（书）？

总体设计五准则（设计原理）

模块的作用范围和控制范围是什么（2015）？

衡量模块独立性的两个标准及其含义（2017）？

详细设计的目的和主要任务？

从伪码转变为程序流程图或者从程序流程图转变为伪码是否是唯一的？请说明理由（书）

编码和测试、质量保证

高级语言的优点：

编程时使用的程序设计语言对软件的开发和维护有什么影响（书）？

要想具有好的编程风格，在编写代码时应注意哪些问题

软件测试基本准则（基本原理）是什么

什么是白盒测试？白盒测试的方法有哪些？

黑盒测试和白盒测试的区别（2015）

什么是测试用例？（2016）

测试V模型的理解：

什么是软件可靠性和可用性？软件能不能同时正确又不可靠？

软件测试和调试的目的有什么区别（2015，2017）？

维护

软件维护有哪些类型？

为什么大型软件的维护成本高达开发成本的4倍左右（书）

某软件公司采取下列措施提高他们开发的软件产品的可维护性，请判断哪些措施是正确的，那些不正确？

加入你的任务是对一个已有的软件做重大修改，而且只允许你从下述文档中选取两份...

当一个十几年前开发出的程序还在为用户完成关键的业务工作时，是否有必要对它进行软件再工程？经济上是否划算？

代码重构和正向工程有什么相同和不同之处（书）

面向对象方法学

面向对象分析建模的工具有哪些？面向对象分析时类和对象的静态关系主要有哪几种（2017）？

面向对象范型开发软件时与用结构化范型开发软件时，软件的生命周期有什么不同之处？这种差异带来了什么后果（书）

为什么在开发大型软件时，采用面向对象范型比采用结构化范型较易取得成功（书）？

对象和属性有什么区别（书）？

对象是什么，以及对象和数据实体的区别：

什么是模型？开发软件时为什么要建立模型（书）？

为什么夏利牌小汽车时小汽车类的特化，而发动机类不是？

为什么说面向对象方法和人类习惯的思维解题方法比较一致（书）？

面向对象方法学的优点以及原因：

面向对象方法学中其他的概念：

面向对象中类的封装性是什么？

面向对象方法学的几种模型和其间的关系：

面向对象分析与设计

什么是用例图？（2015）

应该依据什么来评价用例图/脚本/状态图

试比较功能内聚和信息性内聚（书）

继承重用和多态重用的关系

在面向对象设计过程中为什么会调整对目标系统的需求（书）？

为了设计人机交互子系统，为什么要分类用户（书）？

问题空间和解空间有什么区别？

从面向对象分析->面向对象设计阶段，对象模型有什么变化？

面向对象实现

为什么以后刚刚尽量使用面向对象语言来实现面向对象分析与设计的结果（书）？

什么是强类型语言？有哪些优点（书）？

用动态联编实现多态是否会显著降低程序运行效率（书）？

为什么说参数化类有助于提高可重用性？（书）：

把策略方法和实现方法分开后，为什么能够提高可重用性（书）？

面向对象软件的哪些特点使得测试和维护变得容易？哪些特点使测试和维护变得困难？（书）

软件项目管理

关于CMM能力成熟度模型

为什么成本在估算模型中的参数应该根据开发公司的历史数据来确定？（书）

为什么推迟关键路径上的任务会延迟整个项目？（书）

机动时间有什么重要性？（书）

项目负责人，开发应用系统，类似以往做过的系统，规模更大更复杂，需求文档完整，项目组结构？过程模型？（书）

**软件产品的特点：**

软件是逻辑部件而不是物理部件，缺乏可见性是其显著特征，因此，管理和控制软件开发过程相当困难。此外，软件维护通常意味着改正或修改原有的设计，从而使得软件维护困难。

软件的另一个突出特点是规模庞大，程序复杂性会随着程序规模增加以指数速度上升。状态数巨大，难以完全预见每种情况。

**软件就是程序这个观点正确吗？给出原因（2017）。**

不正确，软件等于程序加文档加数据。

文档是软件的一个重要组成部分，在软件的开发过程中起着非常重要的作用。

软件开发的每一个阶段都有响应的文档，它是开发人员与用户以及开发人员与用户管理人员之间交流的媒介。

文档是软件在不同阶段的表现形式。

程序和文档必须一致，软件才有价值。

文档质量直接决定软件质量的高低。

文档是软件测试和维护的依据，在没有文档或文档不全的情况下对大型软件进行测试和维护是困难的。

文档是软件可重用的依据。

**软件危机、软件危机的特点以及软件危机产生的原因：**

软件危机两个方面的问题：

**如何开发软件，以满足对软件日益增长的需求。**

**如何维护数量不断膨胀的已有软件。**

对软件开发成本和进度的估算不准确 – 项目管理

软件成本占计算机系统比例上升 – 项目管理

成果用户不满意 – 需求分析

产品质量靠不住 – 质量保证

软件不可维护 – 维护

没有合适文档 --文档与维护

软件开发生产率提高的速度，远远跟不上计算机应用普及的趋势 – 总结

原因：

客观上，软件产品的特点。

主观上，软件工程师的错误认识。包括：

忽视需求分析的重要性，不能明确用户的需求；认为软件开发就是写程序；轻视软件维护；忽视代码评审和软件测试。

缺乏正确的理论指导，包括缺乏有力的方法学和工具。

**什么是软件工程？软件工程的出现主要是为了解决什么？**

软件工程是研究和应用如何以系统性的、规范化的、可定量的过程化方法去开发和维护软件，以及如何把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来。

软件工程的出现就是为了解决软件危机的两个方面：

如何开发软件，以满足对软件日益增长的需求。

如何维护数量不断膨胀的已有软件。

**什么是软件过程？软件过程和软件工程方法学有什么关系（书+真题）？**

软件过程是为了获得高质量软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作量。

软件过程定义了运用方法的顺序、应该交付的文档资料、为保证软件质量和协调变化所需的管理措施，以及标志软件开发各个阶段任务完成的里程碑。

通常把软件生命周期全过程使用的一整套技术方法的集合称为方法学，也成为范型。软件工程方法学包含三个要素：方法、工具和过程。（也即软件过程是软件工程方法学的一部分）

**为什么说分阶段的生命周期模型有助于软件项目管理（书）？**

软件是计算机的逻辑部件而非物理部件。其显著特点是缺乏可见性。因此，管理和控制软件开发过程相当困难。

分阶段的生命周期模型提高了软件项目的可见性，管理者可以把各个阶段任务的完成作为里程碑来对软件开发过程进行管理，把阶段划分的更细就能更密切的监控软件项目的进展情况。

**什么是里程碑？它应该有哪些特征（书）？**

里程碑是用来说明项目进展情况的事件。通常，把一个开发活动的结束或者一项开发任务的完成定义为一个里程碑。

**假如你是一家软件公司的项目经理， 职责是管理该公司已经被广泛使用的图片处理软件新版本的开发。由于市场竞争激烈，公司规定了严格的完成期限并且已经对外公布。你打算采用哪种软件生命周期模型？为什么（书、2017）？**

增量模型。

原因：

增量模型开发软件时可以并行完成开发工作，因此能够加快开发进度。

旧版本相当于一个原型，通过收集用户对旧版本的反映，较容易确定用户对新版本的需求，没必要专门建立原型来分析用户需求。

公司软件工程师对这种软件很熟悉，有开发该种软件的丰富经验，具有采用增量模型开发需要的技术水平。

今后很可能还要开发新版本，因此需要把体系结构设计成开放式的，有利于今后的改进和扩充。

**在需求分析和设计阶段建立原型的好处有哪些（2016）？**

增进开发人员和用户对系统需求的一致理解，明确软件需求，尤其是功能含糊的需求。

提供了有力的学习手段，能够使用户尽早解除系统。

可以用来识别或者化解风险（螺旋模型）

有的原型可以原封不动地成为产品，有的略加修改就可以最终系统的一个组成部分，有利于系统的最终建成。

**螺旋模型和Rational统一过程有什么相似之处？差异（书）？**

RUP没在真题里面见过，关于RUP的题暂时不整理了，下面还有两道：

试比较Rational统一过程和敏捷过程

试讨论微软过程和RUP以及敏捷过程的关系。

**说明敏捷开发的适用范围（书）。**

敏捷过程具有对变化和不确定性的更快速、更敏捷的反应特性，而且在快速的同时能够保持可持续的开发速度。因此比较适用于开发可用资源以及开发时间都有苛刻约束的小型项目。

**对于螺旋模型的理解？**

螺旋模型是一种风险驱动型过程模型。基本思想是使用原型以及其他方法尽量降低风险。可以看作每个阶段之前都增加了风险分析过程的快速原型模型。主要适用于开发内部大型软件项目。

保留了经典模型中逐步细化的方法，同时纳入迭代思想。

优点：有利于已有软件的重用、有助于把软件质量作为软件开发的一个重要目标、减少了过多测试或者测试不足带来的风险、软件开发和维护没有本质区别。

**结构化分析与设计**

**数据流图和数据字典的关系是什么（2017）？**

DFD是一种图形化技术，描绘信息流和数据从输入、移动到输出的过程中经受的变换。数据流图中没有具体的物理部件，只是描绘在软件中流动和被处理的逻辑过程。数据流图时系统逻辑功能的图形表示，即使不是专业的计算机技术人员也容易理解它。因此是分析员和用户之间极好的通信工具。

数据字典是关于数据的信息的集合，也就是对数据流图中包含的所有元素定义的集合。

数据流图和数据字典共同构成系统的逻辑模型，没有数据字典，数据流图就不严格，没有数据流图，数据字典也难于发挥作用。只有数据流图和对数据流字典中每个元素的精确定义放在一起，才能共同够成系统的规格说明。

共同够成系统的逻辑模型。

**为什么数据流图要分层？画分层DFD要遵循哪些原则（2015）？**

为了控制复杂性，对数据流图分层，就能把复杂的数据简单化。

原则：自顶向下，逐步求精

具体原则：数据守恒和数据封闭原则。

加工分解的原则：父图和子图平衡的原则，合理使用文件的原则。

**情景与描述了所有可能序列的状态图之间有什么关系（书）？**

情景仅仅是通过部分或者全部状态图的一条路径，也就是说，情景仅仅描述了系统的某个典型行为，而状态图描述了系统的所有行为。

**在程序流程图中的每个节点都必须有一条从开始节点到该节点本身的路径，以及一条从该节点到结束节点的路径。为什么数据流图没有关于节点之间可达性的类似规则（书）？**

数据流图不描述控制。因此在同一个数据流图中两个处理之间可能没有通路。如果每个处理都是用不同的输入数据，并且生成不同的输出数据，而且一个处理的输出不用做另一个处理的输入，那么在他们之间没有弧。

**模块化设计五准则（设计原理）**

模块化和模块独立、抽象、信息隐藏、逐步求精、局部化

**模块的作用范围和控制范围是什么（2015）？**

作用范围是指受模块内一个判定影响的所有模块集合。

控制范围是模块本身以及所有下属模块（直接与间接从属于它）的模块集合。

**衡量模块独立性的两个标准及其含义（2017）？**

内聚性：块内联系，模块的功能强度的度量，一个模块内部各个元素彼此结合的紧密程度的度量，模块内元素联系越紧密，内聚性越高。

耦合性：块间联系，软件系统结构中，各模块间相互联系紧密程度的一种度量。模块之间联系越紧密，耦合性越强，模块独立性越差。

**详细设计的目的和主要任务？**

主要目的：确定怎样具体实现所要求的系统。

主要任务是过程设计、接口设计和数据设计

过程设计：设计软件体系结构中所包含的每个模块的实现算法

数据设计：设计软件中所需要的数据结构

接口设计：设计软件内部各个模块之间、软件与协作系统之间以及软件和使用它的人之间的通信方式，这里面包含了用户界面设计。

其中，过程设计是最主要的。

**从伪码转变为程序流程图或者从程序流程图转变为伪码是否是唯一的？请说明理由（书）**

伪码准确的描述了程序的控制流程。由于伪码在描述程序的控制流程时是无二义性的，因此由伪码转变成的程序流程图是唯一的。但是同样的控制流程可以用不同的伪码来描述，因此由程序流程图转变而来的伪码不是唯一的。

**编码和测试、质量保证**

**高级语言的优点：**

高级语言拥有良好的模块化机制、可读性好的控制结构和数据结构。语言的特点使编译程序能够尽可能多的发现程序代码中的错误，此外理想的高级语言还具有良好的独立编译机制。

**编程时使用的程序设计语言对软件的开发和维护有什么影响（书）？**

语法清晰易懂、不容易产生二义性–影响可读性和可理解性

模块化机制是否完善，编译程序差错能力的强弱–影响可靠性

程序设计语言实现设计结果的难易程度、是否提供良好的独立编译机制、可利用的软件开发工具是否丰富有效–影响程序开发效率

编译程序优化能力的强弱、程序设计语言操纵硬件设施的能力–影响程序的运行效率

程序设计语言的标准化程度、所提供的模块封装机制、源程序的可读性和可理解性等–影响可维护性

**要想具有好的编程风格，在编写代码时应注意哪些问题**

注意程序的书写格式，让它的形式反映出内在的意义和结构。

用规范清晰、容易理解的方式编写程序

编程中应该仔细研究编译程序给出的错误信息和警告信息，弄清楚信息的确切根源并且予以解决。

随时注意表达式计算过程和模型

决不去写依赖运算对象的求值顺序的表达式

保证一个函数的定义点和它的所有使用点都能看到同一个完整的函数原型说明

新书中的内容–为了写出好程序，应该遵循下述规则：

程序内部应该有很好的文档，包括恰当的标识符、适当的注解和程序的视觉组织等。

数据说明应该易于理解和查阅。

语句构造应该尽可能简单直观。

输入输出风格遵守人机界面设计准则。

效率满足用户需求即可。

软件测试基本准则（基本原理）是什么

所有的测试都应该追溯到用户的需求

应该远在测试之前就指定测试计划

将Pareto原理（八二）应用到软件测试中

应该从小规模测试开始并逐步进行大规模测试

穷举测试是不可能的

为了最佳测试效果应该由独立担得第三方从事测试工作

**什么是白盒测试？白盒测试的方法有哪些？**

白盒测试是已经知道了程序的内部逻辑结构，根据程序内部逻辑结构通过测试来检验是否按规格说明书的规定正确运行。

方法：逻辑覆盖（点覆盖、边覆盖、路径覆盖、语句覆盖、条件覆盖、判定覆盖、条件/判定覆盖、条件组合覆盖）

控制结构测试：基本路径测试、条件测试、循环测试

**黑盒测试和白盒测试的区别（2015）**

简述：

已知条件：已知产品的功能设计规格，可以进行测试证明每个实现了的功能是否符合要求。

测试对象：测试功能

测试依据：需求规格说明书

方法：等价类划分，边界值测试

优点：能够站在用户的立场上测试

缺点：不能测试程序的内部特定部位，如程序有误，则无法发现

白盒测试：已知产品的内部工作过程，可以进行测试证明每种内部操作是否符合设计规格要求。

测试对象：测试程序接口和结构

测试依据：软件程序

方法：逻辑覆盖

优点：对程序内部特部位进行逻辑测试

缺点：无法检验程序外部特性

黑盒测试把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，针对软件界面和软件功能进行测试，只检查功能是否符合需求规格说明书能正常使用，因此黑盒测试又叫功能测试或者数据驱动测试。

白盒测试对软件的过程细节做细致的检查，这种方法是把测试对象看作一个打开的盒子，它允许测试人员对程序内部的逻辑结构及有关信息，设计或选择测试用例，对程序所有逻辑路径进行测试。通过在不同点检查程序状态，确定实际状态是否与预期的状态一致，因此白盒测试又称为结构测试或者罗技驱动测试。白盒测试是按照程序的内部结构来测试程序，通过测试检验产品内部动作是否按照设计说明书的要求正常进行，检验程序中的每条通道是否按照正常工作。

**什么是测试用例？（2016）**

是为某个特殊目标而编制的一组测试输入、执行条件与预期结果，以便测试某个程序路径或者核实是否满足某个特定需求，它包括测试数据和预期输出结果。

**测试V模型的理解：**

V模型是广为人知的测试模型。

一般来说，单元测试对应的是详细设计环节，也就是说，单元测试的测试用例是和详细设计一起出现的。开发人员进行详细设计的时候，测试人员根据详细设计结果设计对应的测试用例。同理，集成测试对应概要设计，集成测试主要针对模块之间的接口和功能，因此在概要设计的时候根据概要设计结果编写集成测试的测试用例。系统测试主要针对需求分析阶段产生的错误，因此在需求分析阶段，测试人员根据需求规格说明书来设计系统设计的测试用例。

**什么是软件可靠性和可用性？软件能不能同时正确又不可靠？**

所谓软件可靠性，是程序在给定的时间间隔内按照规格说明书的规定正确运行的概率。软件可用性是程序是在给定的时间点，程序按照规格说明书正确运行的概率。

能。通常认为，软件可靠性既包含正确性又包含健壮性，程序不仅在预定的环境下应该能正确地完成预期功能，而且在硬件发生故障、输入数据无效或者用户操作错误等意外环境下，程序应该能作出适当的响应。

如果一个程序在预定环境下能够正确完成预期功能，但是在意外环境下不能做出适当的响应，那么程序就是既正确又不可靠。

**软件测试和调试的目的有什么区别（2015，2017）？**

测试是为了找出软件已经存在的错误，而调试是定位错误、修改程序以修正错误。

测试从已知条件开始，有预知的结局。调试从未知的条件开始，其结局不可预知。

测试可以计划，预先定义测试用例和过程，工作进度可以度量，调试不行。

调试在测试之后，在方法、策略、思路上有所不同。

调试时随机性的，由程序员完成，为了程序可运行。测试时目的性的，由测试人员完成，为了程序可完成指定功能。

目的不同、任务不同、指导原则和方法不同，操作者不同（2015四点）

目的不同：测试的目的是发现错误，而不是找出错误的原因和发生的地方。调试的目的是为了证明程序的正确，因此必须不断地排除错误。前者是挑错，是质量保证活动的一部分，后者是排错，是编码的一部分。

任务不同：测试属于质量保证活动，贯穿整个开发过程，从需求分析开始，就要指定软件测试计划。软件设计时设计 系统、集成测试用例，编码阶段设计单元测试。软件测试要进行集成测试、系统测试等活动，直到产品交付，只要有修改就有软件测试。产品交付后同样。它是比较规律的活动，有系统的方法、原则来指导。调试的方法与经常使用的开发工具有关，只有一些启发式的方法，是一种比较依赖开发人员经验的活动。

指导原则和方法不同。软件测试使用一种有规律的活动，有一系列软件测试的原则，其中主要是制定计划，然后严格执行。其次时一种挑剔性行为，因此它不但要测试软件该做的，还需要测试软件不应该做的事情。调试所遵循的规律主要是一些启发性规则，是一个推理过程，例如归纳法、演绎法、回溯法等。测试的输出是可预知的，其测试用例必须包含预期的结果，而调试的输出大多是不可预见的，需要调试者去解释、发现产生的原因。

操作者不同。由于心理状态障碍，因此为了测试更客观有效，测试人员一般不是开发人员。调试人员一般是开发人员。

**维护**

**软件维护有哪些类型？**

改正性维护（改bug）：诊断和改正用户使用软件时发现的错误

完善性维护（扩充功能、改善性能）：为了使软件和变化了的环境适当配合

适应性维护（硬件换代）：用户在使用软件的过程中提出的增加新功能或改变已有功能的要求，还可能要求进一步提高程序的性能。为了满足这类要求而进行的修改。

预防性维护（重构）：为了提高未来的可维护性或可靠性而主动地修改软件。

完善性维护占比最大。

**为什么大型软件的维护成本高达开发成本的4倍左右（书）**

某软件公司采取下列措施提高他们开发的软件产品的可维护性，请判断哪些措施是正确的，那些不正确？

加入你的任务是对一个已有的软件做重大修改，而且只允许你从下述文档中选取两份…

当一个十几年前开发出的程序还在为用户完成关键的业务工作时，是否有必要对它进行软件再工程？经济上是否划算？

代码重构和正向工程有什么相同和不同之处（书）

代码重构和正向工程都需要重新设计程序的数据结构和算法，而且需要重新编写程序代码，这些是相同之处。

通常，代码重构不修改程序的体系结构，只修改某些模块的设计细节和模块中所使用的局部数据结构，并且重新编写这些模块的代码。如果修改的范围扩展到模块边界之外并且涉及程序的体系结构，代码重构就变成正向工程。

**面向对象方法学**

**面向对象分析建模的工具有哪些？面向对象分析时类和对象的静态关系主要有哪几种（2017）？**

工具：类图、状态图、用例图

对象图、构件图、部署图、状态图、活动图、顺序图、协作图。

静态关系：关联、聚集、泛化、依赖四种

**面向对象范型开发软件时与用结构化范型开发软件时，软件的生命周期有什么不同之处？这种差异带来了什么后果（书）**

结构化生命周期：陈述需求、需求分析、软件设计、实现、维护

面向对象生命周期：需求陈述、面向对象分析、面向对象设计、面向对象实现。

区别： 结构化方法的分析阶段主要任务是确定产品应该做什么，而设计阶段常划分为概要设计、详细设计两个子阶段。在概要设计子阶段，软件工程师把产品分解为若干个模块，在详细设计子阶段再依次设计每个模块的数据结构和实现算法。

面向对象范型的面向对象分析阶段主要工作是确定对象，对象就是面向对象软件的模块，因此，在面向对象分析阶段就开始了结构设计的工作。由此可见，面向对象分析阶段比它在结构化范型中的对应阶段走的更远、工作更深入。

差异的结果： 使用结构化范型开发软件时，再分析阶段和设计阶段有一个很大的转变：分析阶段的目的是确定产品应该“做什么”，而设计阶段的目的是确定产品“怎样做”，这两个阶段的工作性质明显不同。相反，使用面向对象范型开发软件时，“对象”从一开始就进入了软件的生命周期。软件工程师在分析阶段把对象提取出来，在设计阶段对其进行设计，在实现阶段对其进行编码和测试。

由此可见，使用面向对象范型开发软件时，整个开发过程都围绕统一的概念–对象，因此阶段与阶段之间的转变比较平缓，减少了开发软件过程中所犯的错误。

**为什么在开发大型软件时，采用面向对象范型比采用结构化范型较易取得成功（书）？**

结构化技术要么面向处理（面向数据流的设计方法），要么面向数据（面向数据结构的设计方法），但没有既面向处理又面向数据的设计方法，用结构化技术开发出的软件产品的基本成分是产品的行为和这些行为所操作的数据，由于数据和对数据的处理之间是分离的，尽管开发者把程序划分成了许多模块，但是这些模块之间的联系是比较紧密的。因此，使用结构化范型开发出来的软件本质上是一个完整的单元，带来的结果是软件规模越大，用结构化范型开发软件的技术难度和管理难度就越大。

与结构化技术相反，面向对象技术是一种以数据为主线，把数据和操作相结合的方法。面向对象范型把对象作为由数据和操作组成的统一体，用面向对象开发软件时，对象好像一个微型程序，有自己的数据和操作，功能和用途。因此，可以把一个大型软件产品分解成一系列本质上相互独立的小产品来处理，降低了软件开发的技术难度，而且也使得对软件开发工作的管理变得相对容易了。

**对象和属性有什么区别（书）？**

对象时对客观世界实体的抽象，是描述实体静态属性的数据和代表实体动态行为的操作结合在一起所构成的统一体。属性只是对象的一种特性，它是组成对象的一种成分。

**对象是什么，以及对象和数据实体的区别：**

对象是使用面向对象方法学开发软件时对客观世界实体的抽象。对象是由描述实体属性的数据和对这些数据实施的操作封装在一起组成的统一体。

传统的数据使用结构化方法学开发软件时对客观世界实体的抽象，但是这种抽象是不全面的：数据只能描述实体的静态属性，不能描述实体的动态行为，必须从外界对数据施加操作，才能改变数据实现实体应有的行为。

对象和传统数据实体有本质区别，对象不是被动地接受外部实施的操作，相反，他是进行数据处理的主体。必须发消息请求对象主动执行它的某些操作，处理它的私有数据，外界不能直接对它的私有数据进行操作。

**什么是模型？开发软件时为什么要建立模型（书）？**

模型是为了理解事物而对其作出的一种抽象。是对事物一种无歧义的书面描述。通常模型由一组图示符号和组织这些符号的规则组成，利用它们来定义和描述问题域中的术语和概念。更进一步讲，模型是一种思维工具，利用这种思维工具可以把知识规范的表达出来。

为什么：

结局问题前首先要理解问题，为了理解客户要求解决的问题，往往需要建立问题域的模型。

为了开发复杂的软件系统，系统分析员应该从不同角度抽象出目标系统的特性，使用精确的表示方法狗在系统的模型。验证模型是否满足需求，并逐步求精，直至最终实现。

对于过于复杂而不能直接理解的系统，特别需要建立模型。建立模型的主要目的是降低复杂性。通过模型把系统重要部分分解成人脑一次能处理的若干个部分，减少了系统的复杂程度。

**为什么说面向对象方法和人类习惯的思维解题方法比较一致（书）？**

书p139

人类认识世界是一个渐进、多次反复、逐渐深化的过程。

抽象时人在认识和解决问题时有力的思维工具，处理复杂对象时，集中分析与问题有关的实质特性，暂时忽略与问题无关的特性。

面向对象方法学的出发点就是分析、设计和实现的软件系统的过程，尽可能和人类认识世界、解决方法的方法的过程，一致，也就是说描述问题的问题空间和描述解法的解空间在结构上尽可能一致。

也可以说面向对象方法学的基本原则时按照人们习惯的思维方式建立问题域模型，开发出尽可能直观，自然地表现求解方法的软件系统。

对象时客观世界实体的抽象，提供了理想的数据抽象机制，同时又有良好的过程抽象机制。便于人们开发软件时使用习惯的抽象思维工具。继承符合渐进逐渐深化的认识过程。

**面向对象方法学的优点以及原因：**

与人类思维方法一致（模拟客观世界和人类的思维方式，问题空间和解空间一致性强，因此可理解性好）、稳定性好（类的模块独立性好，需求变动不容易引起全局修改，因此易于修改）、可重用性好（类构件和继承机制）、易于测试和调试（对类的测试容易实现），可维护性好（是由其稳定性好、容易理解、容易修改、容易重用、容易测试决定的）

**面向对象方法学中其他的概念：**

类：对具有相同数据和相同操作的一组相似对象的定义。

实例：以某个特定的类为样板而建立的一个具体的对象。

消息：消息是要求某个对象执行在定义它的那个类中所定义的某个操作的规格说明。通常一个消息由三部分组成：接收消息的对象、消息选择符（消息名）、零个或多个变元（参数）。

方法：方法就是对象可能执行的操作，也就是类中所定义的服务。

属性：属性就是类中定义的数据，是对客观世界实体所具有的性质的抽象。

封装：见类的封装性

继承：面向对象软件技术中，继承是子类自动共享基类中所定义的数据和方法的机制。

多态性：在面向对象软件技术中，多态性指子类可以向父类那样使用，同样的消息可以发给父类对象也可以发给子类对象，也就是说，在类等级的不同层次中可以共享一个方法的名字，然而不同层次中的每个类却各自按照自己的需要来实现这个行为。

重载：函数重载和运算符重载。函数重载是同一作用域内的若干参数特征不同的函数可以使用相同的函数名字。运算符重载是指同一个运算符可以施加于不同的操作数。

**面向对象中类的封装性是什么？**

类似传统方法学中的信息隐藏。外界不能访问对象的私有数据和服务，只通过暴露的接口来完成任务。

在面向对象的程序中，把数据和实现操作的代码集中起来放在对象内部，一个对象好像一个黑盒子，具体的数据和操作都被封装在黑盒子里面，从外面看不见，更不能从外面直接访问或者修改这些数据和代码。

使用一个对象的时候，只需要知道其向外界提供的接口形式，而无需知道它的数据结构细节和操作实现方法。

封装也就是信息隐藏，通过封装对外界隐藏了对象的实现细节。

封装性使保证软件部件优良的模块的基础。

面向对象的作用使封装良好的模块，类定义将其说明和实现显式的分开，其内部实现按其具体定义的作用域提供保护。

对象是封装的最基本的单位，封装为了防止程序相互依赖性而带来的变动影响，面向对象的封装比传统的封装更清晰、更有力。

**面向对象方法学的几种模型和其间的关系：**

对象模型–三种模型的基础，表示静态的、结构化的系统的数据性质。

动态模型–描述系统控制性质，规定了对象模型中对象的合法变化序列。

功能模型–描述系统的功能性质，指明系统做什么。

关系：

动态模型描述了类实例的生命周期和运行周期

状态转换促使行为发生，这些行为在数据流图中被映射成处理，在用例中被映射成用例，它们同时与类图中的服务相对应。

功能模型中的处理（或者用例）对应于对象模型中的类所提供的服务。有时一个处理（用例）对应多个服务，也可能一个服务对应多个处理（用例）

数据流图中的数据存储以及数据的源点/终点通常是对象模型中的对象。

数据流图中的数据流往往是对象模型中对象的属性值，也可能是整个对象。

用例图中的行为者可能是对象模型中的对象或者数据流图中的数据源点/终点。

功能模型中的处理（或者用例）可能产生动态模型中的事件。

对象模型描述了数据流图中数据流、数据存储以及数据源点、终点的结构。

总结来说。

对象本身 – 数据流图数据源点、终点、存储、数据流（数据流也可能对应对象的属性）

动态模型描述类实例的生命/运行周期，状态转换对应类图的服务、数据流图的处理、用例图的用例。

动态要素对应动态要素，静态要素对应静态要素。

**面向对象分析与设计**

**什么是用例图？（2015）**

是参与者（actor）、用例（use case）、以及它们之间的关系构成的用于描述系统功能的视图。

用例图是被称为参与者的外部用户能观察到的系统功能的模型图，呈现了一些参与者和一些用例，以及它们之间的关系。主要用于对系统、子系统或者类的功能行为进行建模。

应该依据什么来评价用例图/脚本/状态图

用例图是从用户的观点来描述系统的功能，因此必须包含用户关心的所有关键功能。

脚本必须从用户的观点来描述每个重要的功能序列，因此，脚本应该能说明系统的一类重要功能或者具体的使用方法。

状态图应该描绘所有可能的状态转换，图中每条弧都要有一个引起状态转换的事件，从开始节点（初态）到每个节点（中间状态）以及从每个节点到终态都必须要有一条路径。

试比较功能内聚和信息性内聚（书）

结构化范型-功能分解-功能内聚：如果模块内所有处理元素属于一个整体，完成单一完整的功能。

面向对象范型-对象分解-信息性内聚：一个对象可以完成很多相关的操作，每个操作都有自己的入口点，它们的代码相对独立，而且所有的操作都在相同的数据结构上完成。

也就是说，操作围绕对其数据所需要做的处理来设置，不设置与这些数据无关的操作，该对象具有信息性内聚。

实际上，信息性内聚包含的操作本身应该是功能内聚的。

**继承重用和多态重用的关系**

**实例重用、继承重用、多态重用**

继承重用：当已有的类构件不能通过实例重用方式满足当前系统的需求时，利用继承机制从已有的类派生出符合需要的子类，从而获得可以在当前系统中使用的类构件。

多态重用：设计软件构件时，把可能妨碍重用的、与应用环境密切相关的操作从一般操作中分离出来，作为适配接口，并且把这类操作声明为多态操作。从已有类派生出来的子类只需要重新定义某些多态操作即可满足当前系统的需求。

以上，多态重用实际上是一种特殊的继承重用，是充分利用多态性机制支持的继承重用。一般说来，使用多态重用方式重用已有的类构件时，在子类中需要重新定义的操作较少，因此，这种重用方式的成本比继承重用方式低。

**在面向对象设计过程中为什么会调整对目标系统的需求（书）？**

两种情况导致修改需求：一是客户需求和系统外部环境变化；二是系统分析员对问题域理解不透彻，导致面向对象分析模型不能完整准确反应用户真实需求。

修改需求通常只需要简单修改面向对象分析的结果，然后把这些修改反映到问题域子系统中。

**为了设计人机交互子系统，为什么要分类用户（书）？**

人机界面是提供给用户使用的。

用户对人机界面的评价很大程度上由人的主观因素决定。用户的技能水平或者职务不同，偏好和习惯往往不同。因此，为了设计出符合用户需要的人机界面，应该了解和研究用户。

**问题空间和解空间有什么区别？**

问题空间时现实世界的一部分。它由现实世界的实体组成。解空间实际上就是软件系统，由实现解决方案的软件实体组成。

**从面向对象分析->面向对象设计阶段，对象模型有什么变化？**

面向对象分析阶段，对象模型是对问题空间中实体的抽象。随着软件开发过程进入面向对象设计阶段，这些对象逐渐变成解空间的实体。

**面向对象实现**

**为什么以后刚刚尽量使用面向对象语言来实现面向对象分析与设计的结果（书）？**

面向对象语言支持类、封装、对象、多态、继承、重载等面向对象概念，编译程序能够自动实现上述概念，因此将面向对象设计结果翻译为程序比较容易，降低了编程工作量，减少了编程错误。

2. 面向对象分析到设计到实现，始终使用统一的概念，既可以保证开发各个阶段平滑过渡，又可以提高软件的可重用性和可维护性。

简单说：一致的表示方式（有利于实现）、可重用性（重用面向对象设计结果）、可维护性（源程序可理解性强）

**为什么说参数化类有助于提高可重用性？（书）：**

在实际的应用程序中，往往有这样一些软件元素(函数、类等)从逻辑功能看彼此相通，所不同的主要是处理的对象类型不同。

所谓参数化类，就是使用一个或者多个类型去参数化一个类的机制。有了这种机制，可以先定义一个参数化的类模板，在使用时把数据类型作为参数传递进来，从而把这个类模板在不同的应用程序中反复使用，或者在同一程序的不同部分重复使用。

**把策略方法和实现方法分开后，为什么能够提高可重用性（书）？**

两类不同的方法。

策略方法:负责作出决策，提供变元、管理全局资源。

实现方法：完成具体操作。

策略方法通常紧密依赖于具体应用，应用系统不同，策略方法往往不同。实现方法是自含式算法，相对独立于系统应用，在其他应用系统中也可能重用它们。

为提高可重用性，编程时不要把策略方法 和实现方法放在一个方法中，应该把算法的核心部分放在一个单独的具体实现方法中。当开发不同的应用系统时，可以从已有类派生出新的子类，子类直接从基类继承实现方法，根据需要重新定义策略方法。

**面向对象软件的哪些特点使得测试和维护变得容易？哪些特点使测试和维护变得困难？（书）**

封装性和信息隐藏使维护和测试变得容易。

理解一个对象所需要了解的元素，大部分都在对象内部，因此测试和维护比较容易。

对象彼此之间只能通过发送消息相互作用，不能直接修改私有数据，进一步使测试和维护变得容易。

对象本身的修改不会对对象以外产生影响，减少了回归测试的数量。

继承性和多态性增加了测试和维护的难度。

派生类继承了全部基类的属性和方法，因此为了理解和修改派生类，需要了解整个继承结构。

基类发生的变化会传递给派生类。

多态性和动态联编的存在，维护人员需要研究运行时可能发生的各种绑定，才能判断出大量方法中的哪一个方法会在此处被调用。

**软件项目管理**

**关于CMM能力成熟度模型**

五级：初始级、可重复级、已定义级、已管理级、优化级

目的：为大型软件项目的招标活动提供一种全面而客观的评审依据。

基本思想：由于问题是人们管理软件过程的不当引起的，因此新软件技术的运用不会自动提高软件的生产率和质量。（考过）

其有助于建立一个有规律的、成熟的软件过程。CMM策略力图改进软件过程的管理，而技术上的改进是其必然结果。CMM明确定义了五个不同的成熟度等级，一个组织可以按照一系列效地改良型步骤来向更高更成熟的等级迈进。（定义五个等级的目的，考过）

初始级：软件过程混乱无序

可重复级：建立了基本的软件项目管理过程，能够追踪成本、进度、功能和质量。

已定义级：已经定义了完整的软件过程，该软件过程已经进行了文档化和标准化。

已管理级：定量地管理开发过程和软件产品。

优化级：使用定量的信息来管理并不断地改进软件过程。

**为什么成本在估算模型中的参数应该根据开发公司的历史数据来确定？（书）**

每个公司开发的软件类型不完全相同，此外每个公司都有不同的经验、习惯、标准和策略，也就是说，不同的公司能力成熟度并不相同。因此，不同公司开发软件的生产率也不相同。显然不能用同样的成本估算参数来估算工作量，而应该根据公司开发软件的历史数据确定成本估算模型的参数。

**为什么推迟关键路径上的任务会延迟整个项目？（书）**

关键路径定义为一组任务（关键任务），遮住任务决定了完成项目所需要的最短时间。如果位于关键路径上的一个关键人物的完成时间被推迟了，则关键路径上的下一个关键任务的开始时间就要相应的延迟，依次传递，会波及关键路径上的最后一个任务，从而延迟整个项目。

**机动时间有什么重要性？（书）**

虽然不在关键路径上的任务并不决定完成项目所需要的最短时间，可以适当延迟，但是如果延迟过久，则整个项目的完成时间也会被推迟。机动时间给出了完成这类任务的时间范围。

此外，在制定进度计划时仔细研究并充分利用工程网络中的机动时间，往往能够安排出既节省资源又不影响最终竣工时间的进度表。

**项目负责人，开发应用系统，类似以往做过的系统，规模更大更复杂，需求文档完整，项目组结构？过程模型？（书）**

类似以往做过的系统，开发人员积累了丰富经验，没有多少技术难题需要攻克。为了减少通信开销，充分发挥技术骨干的作用。基于主程序员组的形式为宜。

需求文档完善、又有历史经验，可以选用熟悉的瀑布模型来开发。

**补充论述题**

**1、结合项目经验谈谈软件项目管理**

浅谈软件开发项目管理及过程中的要点和难点

**1 引言**

随着整体市场形式的发展，任务重，关系复杂，时间紧，资源有限已经不是一个超级工程才会遇到的问题，听过或看过各部门领导和项目经理们的年度总结就会发现，哪个项目不是这样的，或者说哪个工作任务不是这样的。

项目经理如若能力超群，能掌握项目全局，遇到突发情况，还能力挽狂澜，平息各方纷争完成项目的，那都是英雄，少之又少。作为半职能半矩阵型组织下的项目经理，多数是技术能人提拔上来的，从技术转管理的角色转变很困难，一般公司会成立项目管理部或者项目内设立一名项目管理专员辅助技术型项目经理的工作，非技术型的项目管理人员的岗位职责和角色定位也多泛化，看所处项目情况，工作职能和权力各有不同，落地程度也相差甚大。**如何保障项目持续成功且尽可能多的项目成功呢？**

一个公司的项目规模达到一定数量时，**规范项目管理**是项目持续成功的必经之路。

**2 成功的项目管理**

一提起项目管理，以前就想的是这样的：项目管理人员向事业部各项目组定期收取包括但不限于：项目计划、周报、月报、阶段评审报告、项目过程成套资料成果……

学习项目管理课程后认为是这样的：项目管理有五大过程组十大知识领域，在项目启动、规划、执行、监控和收尾各阶段中按照范围、进度、成本、质量…… 进行分工、信息收集和整合，信息整合后发送相关方审阅。

偶有感叹，**学习那么多项目管理知识还是管不好项目！**质量不达标，成本持续升高，进度不可控，客户意见大，交付迟迟推，回款难上难。

**那怎样算是成功的项目管理？**

个人认为，首先看项目侧重点方向的关键干系人，关键干系人满意则可视为成功。

在公司本位角度看，就是是否实现**所期望的商业价值**，应至少做到以下三点之一：

一、维护核心业务，赚钱稳脚为发展，聚资巩固核心力。

二、开发创新业务，亏本抢占新领域，陪客打探新市场。

三、促进团队建设，有目标，有成就，有成长，有前途。

综上，项目管理是为公司服务的，公司的目标是盈利，分解到项目管理层面，那就是**降本增效**。

**3 项目管理重点、难点及解决建议**

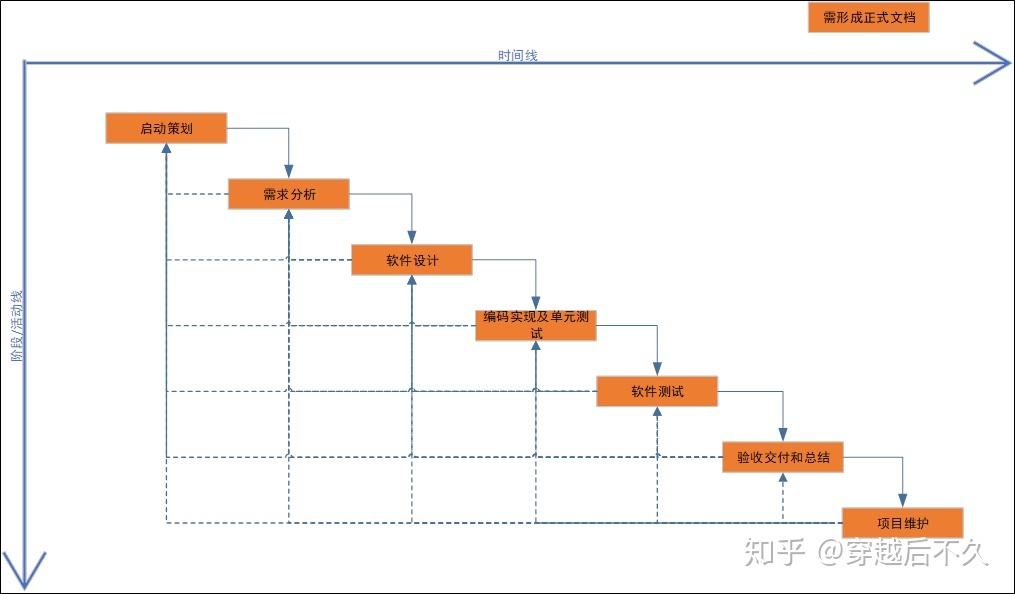
以下就半职能半矩阵型组织架构下的软件开发项目管理做简单经验总结和合理建议。

**3.1 软件项目过程管理简述**

为保障实现项目持续成功，公司一般会建立如下图的组织项目管理流程体系，必须有上下左右环节的配合衔接，相互促进：顶层制度的贯彻，过程流程的执行，知识方法的积累，再推进制度改进，循序提升内部管理水平。



其中产品开发流程中，除公司内部项目外，一般项目周期为：项目机会评估 --》招投标 --》项目开发 --》后期运维。其中售前售后各有团队主导，此处侧重项目开发过程进行细分，项目开发一般存在如下图的过程阶段（以软件开发瀑布模型为例，逻辑顺序可视情况调整或重复过程），项目开发过程中，项目监控及质量控制贯穿始终。



**3.2 项目管理重点**

基于上文组织项目管理流程体系及软件项目开发流程，结合以往项目过程管理及质量管理经验，主要提出以下项目管理重点：

**重点一：统一管理体系。**包括统一管理制度、管理工具、沟通平台、统一模板等，根据产品线制定裁剪指南，重点管理对公司发展有影响的项目，统一底层数据采集，为上层决策提供数据支撑。

**重点二、精准化培训。**培训对象及内容应有针对性，全员通知的大讲堂、感兴趣者参加都是不可取的，充分利用内部资源，注重实操知识培训，针对问题实际，精准化讲落实。

**重点三、坚持有效会议。**避免项目会议多，参加人员多，拖堂时间长；无效会议少开，尽量不开，不搞一言堂，会上讨论要有结果，无果也要确定负责人，会后要有跟进有反馈。

**重点四、提高设计质量。**设计 + 测试是质量两抓手，软件开发项目的设计和测试环节不可裁剪，设计评审是质量把控的第一个监控点，测试则越早进入越能提早发现设计缺陷，但仍要确信质量是设计出来的而非管理出来的。

**重点五、提升专业能力。**提升项目管理人员的专业能力，如何打破部门壁垒，打通沟通渠道，将项目经理、项目组与其他财务、质量、行政、人力、生产、采购等环节协同起来，是项目管理人员中枢作用的关键。推动技术型项目经理转型，项目经理 80% 的工作应该是沟通协调，20% 的精力才可能放在解决具体的开发问题上，加强全局意识，与项目管理人员打好配合。

**重点六：明确角色定位。**部门职责、岗位职责都要明确，如项目管理部门应更多承担服务角色，在掌握多方项目进展情况下，主动服务项目工作，辅助事业部多快好省按时交付项目。

**重点七：维护团队建设。**大团队事业部，小团队项目组，团队成员构成若长期不变，则维护核心人员是关键，培养梯队是重点；团队成员若因项目分分合合，则建立团队机制是关键，加强沟通是重点。

**3.3 项目管理难点及解决建议**

项目管理说简单就五大过程十大知识领域，说复杂就在于没有统一的方法论，我们大多知道应该是怎样的，却不知道如何做到，这里想到一个词 “躬身入局”，很难，这也就出现了很多公司都碰到的难题。

**难点一：制度复杂落地难。**公司多少都会有一些制度要求，根据需要增加的越来越多，可能是特殊要求，也可能是临时要求，很少是长远规划而制定的要求，出于不同部门或是不同立场增加的制度要求无形间造成了制度的复杂无用，没有持续的适应性改进，造成执行层的反感、抱怨，管理层的考核无效。

**建议：**公司确定主管部门、执行部门、监督部门各司其职，站在公司层面疏通流程，必要流程形成制度，阶段流程设定有效期，统一要求，建立平台，保持信息一致，公开透明，相互促进，不适则改。同时结合人事 / 财务部门及时的考核激励措施，有奖有罚比单一的都奖或都罚更能激发人的行动积极性。

**难点二：获取高层重视难。**所有的问题说到底都可以说是管理的问题，项目问题也都可以说是项目管理的问题，所以建立一套适合公司运行管理的项目管理制度及配套管理制度是必要的，而高层领导的重视是关键，不然都是形同虚设，总在关键时刻出岔子后找原因，事后弥补都是运营成本。

**建议：**高层领导的带头落实很重要，一方面体现了严肃性，必要性，也给执行层树立了信心，新建制度或是改进制度一旦发布实施，建议从上到下贯彻落实至少 3 个月，不能搞形式主义也不能半途而废，长期积累数据后才能印证有效性。只有领导重视，真抓实干，持续改进才能真正有所成效。

考虑到领导事务繁忙，主管部门可设立 2-3 个关键时间点，可以是事业部月度会议，或是重大项目总结会，或是项目启动会，邀请高层领导参与并做重要指示，有目的性的强调一二，就可以得到不同于培训宣贯的效果。

**难点三：外部环境压力大。**中国软件行业协会牵头，以促进软件行业高质量发展，推动软件项目管理规范化与标准化建设，去年已经提出了建立软件项目管理标准体系，再而，对于军工企业，这两年软件工程化管理要求更是强制，体系资质更是门槛，对企业和个人的能力都提出了更高的要求。

**建议：**大环境对甲乙双方都是一样的，在与甲方客户合作中，乙方公司如果能抓住利用好这些外部环境，结合自身产品优势，也能帮助公司在一定程度上更好的牵制甲方，共同推进项目成功。其次，大环境既是对公司软件开发管理提出的要求也是公司软件开发管理的方向，公司应该跟进大环境，适当吸收行业内优秀管理方法，助推内部管理创新。

**难点四、如何才为适当？**根据项目具体情况做适当修订或裁剪，适当的力度和粒度怎么衡量一直是管理工作的难题。WBS 中，工作分解得越细致，对工作的规划、管理和控制就越有力，但是，过细的分解会造成管理努力的无效耗费、资源使用效率低下、工作实施效率降低等其他问题。

**建议：**工作安排由近及远，渐近明细；目标分解层层递进，任务落实到人。不可由项目经理或者管理部门单独完成这类工作，必须相关方共同协商确定，管理周期是双周还是月度，受控节点是里程碑还是转阶段，以会议纪要落实最为推荐，重要协商事宜可发送上层领导作为管控，如此更有执行力。

**4 结语**

以一个近期思考的问题结尾：**我们是要管理项目不出问题还是解决项目问题？**

我个人的管理观点是：项目过程中，问题肯定一直存在，不可能没有问题，那就解决问题，坚定解决项目问题而进行的项目管理工作，才是真正的项目管理。

最后还想多提一点，**项目管理解决问题是基本，深层次重点是提升能力**，对个人是提升工作能力，对部门是提升协同能力，对公司是提升盈利能力。

