第一部分

零碎的总结

软件系统的可维护、可扩展

高效率、持续、迭代交付可复用的软件产品

前期、需求分析、分析设计、编码开发的主要步骤

迭代开发、需求进化、架构进化，这是一个快速迭代进化的过程

最多的名词  
松耦合、模块化、组件化、分层、分职责、领域模型、概念模型、设计模型、测试驱动、模型视图分离、逻辑架构、包、静态建模、动态建模、对象设计技能、高内聚低耦合、应用逻辑层、领域层

GRASP的9个模式  
创建者、控制器、纯虚构、信息专家、高内聚、间接性、低耦合、多态性、防止变异

1、创建者  
该模式指导我们分配那些与创建对象有关的职责，寻找在任何情况下都与被创建对象具有连接的创建者。保持低耦合。当创建逻辑非常复杂的情况就要考虑到创建职责委托给具体工厂、抽象工厂。

2、信息专家  
在对象设计中，当定义好对象之间的交互后，我们就可以对软件类的职责分配做出选择。把职责分配给信息专家，它具有实现这个职责所必需的信息。信息专家经常用于职责分配，这是对象设计中不断使用的基本指导原则。专家并不意味着模糊或奇特的想法，它表达了一种“直觉”，即对象完成与其所具有的信息相关的职责。“把职责与数据置与一处，知其责，行其事”。

3、低耦合  
考虑怎样降低依赖性，减少变化带来的影响，提高重用性？耦合是对某元素与其它元素之间的连接、感知和依赖程度的度量。分配职责，使耦合性尽可能低。高耦合对于稳定和普遍使用的元素而言并不是问题，例如，j2ee应用能安全地将自己与Java库耦合，因为Java库是稳定、普遍使用的。高耦合本身并不是问题所在，问题是与某些方面不稳定的元素之间的高耦合。

4、控制器  
控制器是UI层之上的第一个对象，它负责接收和处理系统操作消息。控制器应当把需要完成的工作委派给其它对象。控制器只是协调或控制这些活动，本身并不完成大量工作。  
存在边界、控制和实体类的概念，边界对象是接口的抽象，实体对象是与应用无关的领域软件对象，控制对象是此控制器模式描述的用例处理者。控制器模式的重要结果是，UI对象和UI层不应具有实现系统事件的职责。系统操作应当在对象的应用逻辑层或领域层进行处理，而不是在系统的UI层处理。  
设计不良的控制器类内聚很低，即没有重点，并且要处理过多领域的职责，这种控制器叫做臃肿的控制器。解决控制器的臃肿有两个办法：  
1、增加控制器，系统不是只能有一个控制器，应使用用例控制器，而不是外观控制器。  
2、设计控制器，使它把完成每个系统操作的职责委派给其它对象。  
又强调了一遍，控制器模式的重要推论是，UI对象和UI层不应具有处理系统事件的职责。  
将系统操作的职责分配给领域对象控制器，能够更加容易地复用程序逻辑，这些程序逻辑能支持未来应用中相关的业务过程。这也使移出当前的UI层并使用其它UI架构和技术更容易，也能够用一种离线的“批处理”模式来运行系统。  
上面讲的这一点，实在是深有感触，利用分层架构，在领域层引入控制器模式解耦UI层与领域层，这样做的好处：  
1、实现业务逻辑处理可插拔  
2、脱离UI层可更容易的来做单元测试，并且是批处理自动化方式来做。  
3、UI层不紧紧局限在浏览器，直接拉近了UI层与业务需求变化的响应能力。

5、高内聚  
如何保持对象是有重点的、可理解的、可管理的，并且能够支持低耦合？  
从对象设计的角度上说，内聚是对元素职责的相关性和集中度的度量。如果元素具有高度相关的职责，而且没有过多的工作，那么该元素具有高内聚性。  
如何达到高内聚？  
分配职责可保持较高的内聚性。  
内聚性较低的类要做许多互不相关的工作，或需要完成大量的工作。  
内聚性的类通常表示大粒度的抽象，或承担了本应委托给其它对象的职责。

6、其它  
另一个经典：模块化设计  
与耦合和内聚关系紧密的另一原则是模块化设计，模块化是将系统分解成一组内聚的、松散耦合的模块的特性。我们通过创建具有高内聚的方法和类来促进模块化设计方法。在基本的对象级别上，我们为每个方法设计清晰和单独的目标，并且把一组相关关注置于一个类中，以此来实现模块化。  
内聚和耦合：阴和阳  
不良内聚通常会导致不良耦合，反之亦然。内聚和耦合是互相依赖的。  
   
领域模型和设计模型中对象类之间的多重性可能有所不同。

错误的原则：就像为人分配职责一样为软件对象分配职责！对象设计要根据信息专家原则将职责分配给众多对象。

命令—查询分离原则，执行动作的方法通常会改变对象状态，而向调用者返回数据的查询没有副作用。

第二部分

分类的总结

1.什么是软件危机？软件危机出现的原因及解决手段有哪些？

软件危机是指落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。

出现软件危机的原因主要有：

软件本身独有的特点确实给开发和维护带来了困难

软件开发和维护过程不规范

缺乏软件开发的经验和有关软件开发数据的积累

轻视，人们在制定计划时总会有一些天马行空的想法和要求，轻视是一个最大的错误。

忽视软件需求分析的重要性、忽视软件的可理解性、文档不完备、轻视软件的可维护性、过分强调编码技巧等等方面。

用户与开发人员的沟通问题，从而技术人员获取的需求不够全面。

解决手段：

  在软件工程理论的指导下，建立起较为完备的软件工业化生产体系，形成强大的软件生产能力 。软件标准化与可重用性得到了工业界的高度重视，在避免重用劳动，可以缓解软件危机。

2. .软件工程的要素有哪些，各有什么作用？

软件工程的三种基本要素是方法、工具和过程。

软件工程方法（method）为建造软件提供技术上的解决方法（“如何做”）。目前使用得最广泛的方法是传统方法（结构化方法）和面向对象方法。

工具为方法的运用提供自动的或半自动的软件支撑环境。

过程是为了获得高质量的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步骤。

3. 简述软件过程、软件生存周期、软件过程模型（软件生存周期模型）三者之间的概念区别。

软件过程：软件过程为一个为建造高质量软件所需完成的任务的框架，即形成软件产品的一系列步骤，包括中间产品、资源、角色及过程中采取的方法、工具等范畴。

软件生存周期：指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。

软件过程模型：是一种开发策略，该策略针对软件工程的各个阶段提供了一套范形，使工程的进展达到预期的目的。

4. 需求分析的目的是什么，有什么作用？

需求分析的目的：是要求开发人员准确地理解用户需要什么，进行细致地调查分析，将用户的需陈述转化为完整的需求定义，再由需求定义转化为相应的软件需求规格说明。

需求分析的作用：通过需求分析，可以使开发人员深入细致地调研和分析项目，准确理解用户对项目的功能、性能、可靠性等具体要求，从而确定软件开发的方向而少走弯路。

5. 需求分析过程有哪些步骤？

沟通：业务领域的共利益者（如业务管理人员、市场营销人员、产品管理人员）定义业务用例、确定市场的范围、初略的进行可行性分析、确定项目范围的工作说明。

导出需求：该步骤中应该分析和理解三个问题——范围问题、理解问题、易变问题。可采用会谈、调查、场景分析、快速原型法等多种方法来导出目标系统的高层逻辑模型。

与客户和用户协商：通过协商来调节冲突和问题，对需求进行排序，识别和分析每项需求相关的风险，开发工作量、成本和交付时间。

可行性研究：输入一个框架描述和高层逻辑模型，输出一份可行性研究报告。

精化需求：开发一个精确地技术模型，用以说明软件的功能、特征和约束，该模型定义了问题的信息域，功能域和行为域。可以使用结构化分析方法和面型对象分析方法来构建模型。

编写软件需求规格说明：软件需求规格是需求分析任务的最终产品，是客户、管理者、分析工程师、测试工程师、维护工程师交流的标准和依据。软件需求规格说明描述了系统的数据、功能、行为、性能需求、设计约束、验收标准、以及其他需求相关的信息。

验证需求：验证需求对需求分档和制品进行质量评估，确保需求说明准确、完整。包括正确性、一致性、完整性、可行性、必要性、可检验性、需求的可跟踪性，最后确认签字。

管理需求：需求一般会发生变更，可以通过建立需求跟踪矩阵来保存需求和后继工作成果的对应关系，建立和维护“需求-设计-编程-测试”之间的一致性，确保所有的工作成果符合用户需求。

6. 结构化分析的特点是什么？

采用自顶向下、逐层分解的方法求解复杂问题。

方法简单、清晰，易于学习掌握和使用。

结构化分析的实施步骤是先分析当前环境中已存在的人工系统，在此基础上再构思即将开发的目标系统，这符合人们认识世界改造世界的一般规律，从而大大降低了问题的复杂程度。

结构化分析采用了图形描述方式，用数据流图为即将开发的系统描述了一个可见的模型，也为相同的审查和评价提供了有力的条件，才上到下把大问题分解成若干个小问题，然后分别解决问题。

7. 软件设计的目标及主要任务有哪些？

软件设计的目标：软件系统设计是把软件需求“变换”为用于构造软件的蓝图，“输入”是需求分析各种模型元素，“输出”是软件设计模型和表示，软件设计阶段的基本目标是构造系统“怎么做”的模型描述 ，“设计先于编码”，这是软件工程“推迟实现”的基本原则。

主要任务：主要有以下几类设计活动

总体设计：也称为概要设计，软件结构设计，或高层设计。

体系结构设计：定义软件模块（构件）及模块之间的关系。

接口设计：包括用户接口（界面），外部接口，内部接口。

数据设计：软件涉及的数据结构，文件系统结构，数据库的表结构等。

软件详细设计：也称为模块过程设计，或低层设计。

模块内部细节设计：包括模块所需的算法和数据结构等。

8. 软件结构有哪些优化原则？

模块独立性准则：划分模块时，尽量做到高内聚低耦合，保持模块相对独立性，以此原则优化初始的软件结构。

软件结构的形态特征准则：软件结构的深度、宽度、扇入数和扇出数都要适当。

模块的大小准则：按模块的独立性，即根据模块的功能来决定模块的大小，过大的模块应进行分解，过小的模块应适当合并，但不应降低模块的独立性。

控制域和作用域之间的准则：一个模块的作用域应在其控制范围之内，且条件判定所在的模块应与受其影响的模块在层次上尽量靠近。

模块的接口准则：模块的接口要简单、清晰、力求降低模块接口的复杂程度，设计单入口，单出口的模块。

9. 面向对象的分析与设计方法与结构化的分析设计方法有什么不同？

结构化方法首先关心的是功能，强调以模块（即过程）为核心，采用模块化、自顶向下、逐步求精设计过程，系统是事项模块功能的函数和过程的集合，结构清晰，可读性好，是条软件开发质量的一种有效手段。结构化设计从系统的功能入手，按照工程标准和严格规范将系统分解为若干功能模块。然而，由于用户的需求和软、硬件技术的不断发展变化，作为系统基本成分的功能模块很容易受到影响，局部修改甚至会引起系统的根本性变化。开发过程前期入手快而后期频繁改动的现象比较常见。

面向对象方法则从所处理的数据入手，以数据为中心来描述系统，数据相对于功能而言，具有更强的稳定性，这样设计出的系统模型往往能较好地映射问题域模型。对象、类、继承性、多态性、动态定连概念和设施的引入使用，显然另面向对象的设计方法具有一定的优势，能为生产可重用的软件构建和解决软件的复杂性问题提供一条有效的途径。面向对象的设计过程就是指通过建立一些类以及它们之间的关系来解决实际问题，这就需要对问题域中的对象作整体分析，类和类间关系的设计要求较高，否则设计出的并不是真正意义上的面向对象的软件系统，而只是一些类的堆砌而已，不能体现出面向对象设计方法的优势之处。

同时，系统的分析设计是一个注重实践的领域，不仅仅依赖于一整套核心的概念与原理，要想设计出一个成功的系统来，还需要相应的语言、工具和技术的有力支持。在这方面，经过多年的实践和发展，适应结构化方法的技术和开发环境已经相当稳定成熟，而面向对象方法而言，虽然近时期涌现了大量的新工具和新技术，但仍有待于不断的完善和改进，特别是面向对象的数据库技术。