软件工程第一章作业

1.1 什么是计算机软件？软件的特点是什么？

答：计算机软件指计算机系统中的程序及其文档。

软件的特点是：

A 软件是一种逻辑实体，而不是有形的系统元件，其开发成本和进度难以准确得估算；

B 软件是被开发的或被设计的，没有明显的制造过程，一旦开发成功，只需复制即可，但其维护的工作量大；

C 软件的使用没有硬件那样的机械磨损和老化问题。

1.2 简述软件的分类，并举例说明。

答：在《计算机科学技术百科全书》中，将软件分为系统软件、支撑软件和应用软件 3 类。

A 系统软件：系统软件居于计算机系统中最靠近硬件的一层，其他软件一般都通过系统软件发挥作用。系统软件与具体的应用领域无关。例如：编译程序、操作系统等。

B 支撑软件：支撑软件是支撑软件的开发和维护的软件。例如：数据库管理系统、网络软件、软件工具、软件开发环境等。

C 应用软件：应用软件是特定应用领域专用的软件。例如：工程/科学计算软件、嵌入式软件、产品线软件、Web 应用软件、人工智能软件。

1.4 什么是软件工程？

答：在《计算机科学技术百科全书》中软件工程是应用计算机科学、数学及管理科学等原理，开发软件的工程。

1.5 简述软件工程的基本原则。

答：软件工程原则包括围绕工程设计、工程支持和工程管理提出的以下 4 条基本原则：

第一条：围绕适宜的开发模型；

第二条：采用合适的设计方法；

第三条：提供高质量的工程支撑；

第四条：重视软件工程的管理。

1.6 软件生存周期分哪几个阶段？分别简述各个阶段的任务。

答：软件生存周期有计算机系统工程、需求分析、设计、编码、测试、运行和维护 6 个阶段。

A 计算机系统工程的任务是确定待开发软件的总体要求和范围，以及该软件与其他计算机系统元素之间的关系，进行成本估算，做出进度安排，并进行可行性分析，即从经济、技术、法律等方面分析待开发的软件是否有可行的解决方案，并在若干个可行的解决方案中做出选择。

B 需求分析主要解决待开发软件要“做什么”的问题，确定软件的功

能、性能、数据、界面等要求，生成软件需求规约。

C 软件设计只要解决待开发软件“怎么做”的问题。软件设计通常可分为系统设计和详细设计。系统设计的任务是设计软件系统的体系结构，包括软件系统的组成成分、各成分的功能和接口、成分间的连接和通信，同时设计全局数据结构。详细设计的任务是设计各个组成成分的实现细节，包括局部数据结构和算法等。

D 编码阶段的任务是用某种程序设计语言，将设计的结果转换为可执行的程序代码。

E 测试阶段的任务是发现并纠正软件中的错误和缺陷。测试主要包括单元测试、集成测试、确认测试和系统测试。

F 软件完成各种测试后就可交付使用，在软件运行期间，需对投入运行的软件进行维护，即可发现了软件中潜藏的错误或需要增加新的功能或使软件适应外界环境的变化等情况出现时，对软件进行修改。

1.9 简述各类软件过程模型的特点。

答：典型的软件过程模型有：瀑布模型、演化模型（增量模型、原型模型、螺旋模型）、喷泉模型、基于构件的开发模型和形式方法模型等。

A **瀑布模型**中，上一阶段的活动完成并经过评审后才能开始下一阶段的活动，其特征是：

接受上一阶段活动的结果作为本阶段活动的输入；

依据上一阶段活动的结果实施本阶段应完成的活动；

对本阶段的活动进行评审；

将本阶段活动的结果作为输出。

B 增量模型将软件的开发过程分成若干个日程时间交错的线性序列，每个线性序列产生软件的一个可发布的增量版本，后一个版本是对前一个版本的修改和补充，重复增量发布的过程，直至产生最终的完善产品。

C **原型模型**从软件工程师与客户的交流开始，其目的是定义软件的总体目标，标识需求。然后快速制定原型开发的计划，确定原型的目标和范围，采用快速设计的方式对其建模，并构建模型。被开发的原型应交付给客户使用，并收集客户的反馈意见，这些反馈意见可在下一轮迭代中对原型进行改进。在前一个原型需要改进，或者需要扩展其范围的时候，进入下一轮原型的迭代开发。

D **螺旋模型**将原型模型实现的迭代特征与瀑布模型中控制的和系统化的方面结合起来，不仅体现了这两种模型的优点而且还增加了风险分析。

E 喷泉模型是一种支持面向对象开发的过程模型。类及对象是面向对象方法中的基本成分。在分析阶段，标识类及对象，定义类之间的关系，建立对象-关系模型和对象-行为模型。在设计阶段，从实现的角度对分析模型进行调整和扩充。在编码阶段，用面向对象语言实现类及对象，通过消息机制实现对象之间的通信，完成软件的功能。在面向对象方法中，分析模型和设计模型采用相同的符号表示体系，开发的各个活动没有明显的边界，各个活动经常重复，迭代地交替进行。

F 基于构件的开发模型，基于构件的开发是指利用预先包装的构建来构造应用系统。构件可以是组织内部开发的构建，也可以是商业化的、现存的软件构件。

G 形式化方法是建立在严格数学基础上的一种软件开发方法。软件开发的全过程中，从需求分析、规约、设计、编程、系统集成、测试、文档生成，直至维护等各个阶段，凡是采用严格的数学语言，具有精确的数学语义的方法，都称为形式化方法。形式化方法用严格的数学语言和语义描述功能和设计规约，通过数学的分析和推导，易于发现需求的歧义性、不完整性和不一致性，易于对分析模型、设计模型和程序进行验证。通过数学的演算，使得从形式化功能规约到形式化设计规约，以及从形式化设计规约到程序代码转换成为可能。

1.10 敏捷软件开发的特点是什么？

答：敏捷软件开发的特点有 4 个：

A 个人和交互高于过程和工具；

B 可运行软件高于详尽的文档；

C 与客户协作高于合同谈判；

D 对变更及时做出反应高于遵循计划。

1.12 简述敏捷软件开发的原则。

答：敏捷软件开发必须遵循的 12 条原则如下；

A 最优先的是通过尽早地和不断地提交有价值的软件来使客户满意；

B 欢迎变化的需求，即使该变化出现在开发的后期，为了提升对客户的竞争优势，Agile 过程利用变化作为动力；

C 以几周到几个月为周期，尽快、不断地发布可运行软件；

D 在整个项目过程中，业务人员和开发人员必须天天一起工作；

E 以积极向上的员工为中心建立项目组，给予他们所需要的环境和支持，对他们的工作予以充分的信任；

F 项目组内效率最高、最有效的信息传递方式是面对面的交流；

G 测量项目进展的首要依据是可运行的软件；

H 敏捷过程提倡可持续的开发，项目发起者、开发者和用户应能长期保持恒定的速度；

I 应该时刻关注技术上的精益求精和好的设计，以增强敏捷性；

J 简单化是必不可少的，这是尽可能减少不必要工作的艺术；

K 最好的构架、需求和设计出自于自我组织的团队；

L 团队要定期反思怎样才能更加有效，并据此调整自己的行为。

第 2、3 章

2.1 简述系统工程的任务。

答：系统工程是一个问题求解的活动，其目的是分析基于计算机的系统的功能、性能等要求，并把它们分配到基于计算机系统的各个系统元素中，确定它们的约束条件和接口。主要任务包括：（1）识别用户要求；（2）系统建模与模拟；（3）成本估算及进度安排；（4）可行性分析；（5）生成系统规格说明。

2.2 基于计算机的系统由哪些元素组成？

答：组成基于计算机系统的元素主要有：软件、硬件、人员、数据库、文档和规程。

2.3 简述可行性分析的任务。

答：可行性分析主要从经济、技术、法律等方面分析所给出的解决方案是否可行，能否在规定的资源和时间的约束下完成。

（1）经济可行性主要进行成本效益分析，从经济角度，确定系统是否值得开发。包括成本、效益、货币的时间价值、投资回收期和纯收入。

（2）技术可行性主要根据系统的功能、性能、约束条件等，分析在现有资源和技术条件下系统能否实现。技术可行性分析通常包括风险分析、资源分析和技术分析。

（3）法律可行性主要研究系统开发过程中可能涉及到的合同、侵权、责任以及各种与法律相抵触的问题。

3.1 需求工程的重要性是什么？举出身边由于需求分析失败而造成整个项目失败的例子。

答：重要性是应用已证实有效的技术、方法进行需求分析，确定客户需求、帮助分析人员理解问题，评估可行性，协商合理的解决方案，无歧视地规约方案，确认规约以及将规约转换到可行性的系统时的管

理要求，通过合理的工具和符号系统地描述待开发系统以及其行为特征和相关约束，形成需求文档，并对用户不断变化的需求演进给予支持。因需求分析失败而造成项目失败例子如下:

项目名称：邮政资信管理系统

项目功能：管理邮政方面业务的监督和管理，提高邮政的服务效率。

失败原因：需求分析不足，需求内容不明确，把握不充分。

3.2 需求工程具体包括哪些步骤？每个步骤的具体任务是什么？

答：需求工程具体步骤包括：需求获取、需求分析与协商、系统建模、需求规约、需求验证以及需求管理六个步骤。

（1） 需求获取

在需求获取阶段系统分析人员通过与用户的交流、对现有系统的观察以及对任务进行分析，确定系统或产品范围的限制性描述、与系统或产品有关的人员及特征列表、系统的技术环境的描述、系统功能的列表以及应用于每个需求的领域限制、一组描述不同运行条件下系统或产品使用状况的应用场景以及为更好地定义需求而开发的原型。需求获取的工作产品为进行需求分析提供了基础。

（2） 需求分析与协商

此阶段的任务是对需求进行分类组织，分析每个需求与其他需求的关系以检查需求的一致性、重叠和遗漏的情况，并根据用户的需要对需求进行排序。

（3）系统建模

系统建模是为了在用户和系统分析人员之间建立统一的语言和理解的桥梁，系统分析人员借助建模技术对获取的需求信息进行分析，排除错误和弥补不足，确保需求分析文档正确反映用户的真实意图。

（4）需求规约

软件需求规约是分析任务的最终产物，通过建立完整的信息描述、详细的功能和行为描述、性能和设计约束的说明、合适的验收标准，给出对目标软件的各种需求。需求规约作为用户和开发者之间的一个协议，在之后的软件工程各个阶段发挥重要作用。

（5）需求验证

此阶段的任务是对功能的正确性、完整性和清晰性，以及其他需求给予评价，保证软件需求定义的质量。

（6）需求管理

软件需求管理是对需求工程所有相关活动的规划和控制。换句话说，需求管理就是：一种获取、组织并记录系统需求的系统化方案，以及一个使用户与项目团队对不断变更的系统需求达成并保持一致的过程。

3.3 一个系统分析员应该具备哪些思想素质和基础知识？请说明理由。

答：（1）强烈的责任心和事业心

系统分析师由于必须保证分析的准确性，尤其是需求，所以责任更为重大。一般来说，不应以用户表述不明确，无法得到需求，或者用户需求变化太多太快作为借口，更不能因为下面的研发技术人员技术水平不如自己而责备研发技术人员，同时也需要能够为保证企业利益而说服公司领导做出正确的决策；

（2）钻研精神

IT 行业不同于其他行业，新理念新技术新方法层出不穷，系统分析师需要能够适时适当地引进新理念新技术新方法，为企业提高生产效率，为员工降低劳动强度，为客户提供更具竞争力和更加实用的产品和服务；

（3） 优秀的协商谈判能力

复杂的系统有许多项目相关的人员，他们之间的需求必定会出现冲突，协商的过程就是讨论需求，找出每个人都满意的折衷方案。分析人员是参与这之中的重要一份子，需要协调这之中的各方利益，这就要求分析员必须要有优秀的协商谈判能力。

（4）广泛的知识面：

除了具备基本的 IT 技能、知识外，需要广泛涉猎其他行业其他学科的知识方法，以系统工程的理念，借鉴和利用其他行业的为 IT 行业所用，也可以把 IT 行业的理念应用到其他行业；

（5）精湛的技术能力：

系统分析员往往需要分析可行性和解决研发人员的技术问题，因此必须具备广泛的技术涉猎面和较强的技术能力；

（6）财务能力：

系统分析员往往需要参与项目的招投标分析，为了保证企业的利润和客户的利益，必须进行财务核算，需要具备会计、财务，成本计算等方面的能力；

（7）司法能力：

与其他大部分行业一样，IT 行业也受到法律的约束，任何活动必须合情合理合法，任何违背法律的项目最终都会失败，违背伦理道德的事和人最终都将失去人心。

（8）超强超快的学习能力：

IT 行业新技术层出不穷，同时其他相关必备知识、业务知识也是瞬息万变，系统分析师要能够快速学习并快速掌握和灵活应用；

（9）敏锐的观察力：

由于 IT 行业项目的特殊性，项目复杂多变，系统分析师要能够先于其他人员发现问题、发现隐患，并提前做出规避风险的策略。

3.4 列出在制定需求获取策略时的 3 种主要考虑因素。

答：（1）能否建立起顺畅的通讯途径；（2）是否能够获取用户对系统的功能需求和非功能需求；（3）是否利于在可运行系统时的管理要求。

3.6 举例说明一个系统的 3 个不同类型的非功能需求

答：非功能性需求是指软件产品为满足用户业务需求而必须具有且除

功能需求以外的特性。软件产品的非功能性需求包括系统的性能、可靠性、可维护性、可扩充性和对技术和对业务的适应性等。例如在银行管理系统中，由于银行数据量的庞大以及对银行账户的管理需求，用户对系统的性能、可靠性、可维护性要求很高。安全性是对银行用户个人信息保密的基本要求；在使用系统时，由于用户庞大，要求能快速安全的执行要求，这就对系统的性能有高需求；银行的用户的变动比较大，需求高要求的系统维护。

3.8 软件需求分析的操作性原则和需求工程的指导性原则是什么？

答：（1）必须能够表示和理解问题的信息域；

（2） 必须能够定义软件将完成的功能；

（3） 必须划分描述数据、功能和行为的模型，从而可以分层次地揭示细节；

（4） 分析过程应该从要素信息移向细节信息

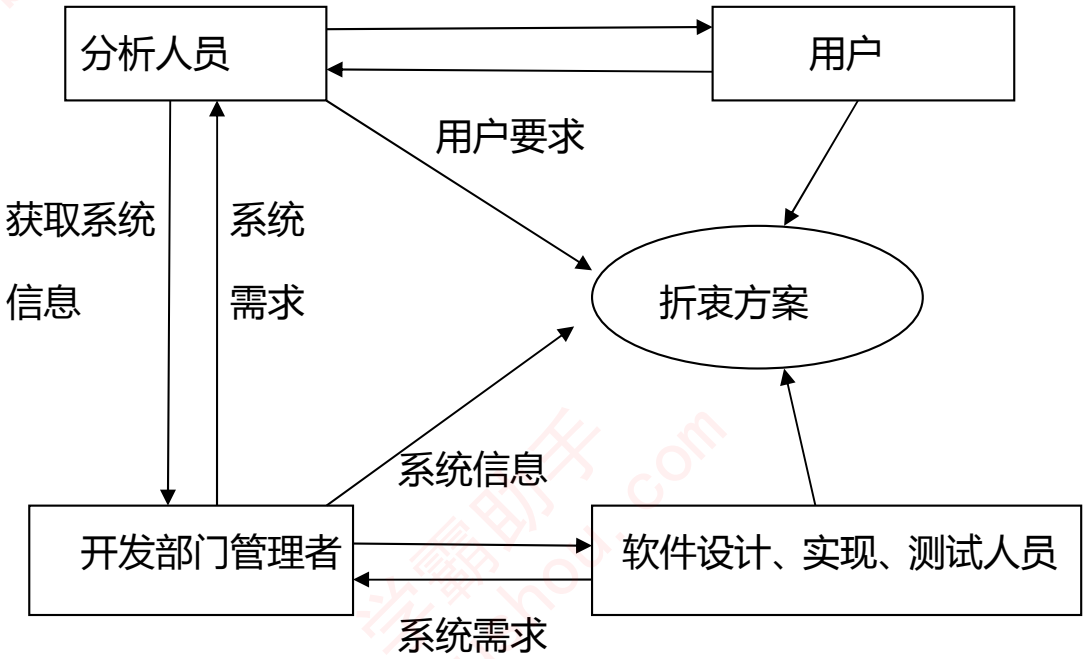
3.9 软件需求规约主要包括哪些内容？自己寻找一个实例，亲自写一个需求规约。

答：软件需求规约包括：引言、信息描述、功能描述、行为描述、检验标准、参考书目、附录。

3.10 需求验证应该有哪些人参加？画出一个过程模型，说明需求评审应该如何组织。

答：参与人员：分析人员，用户，开发部门的管理者，软件设计、实现、测试的人员。

需求分析



第 4 章

4.1 简述软件设计阶段的基本任务

软件设计的输入是软件分析模型。使用一种设计方法，软件分析模型中通过数据、功能和行为模型所展示的软件需求的信息被传送给设计阶段，产生数据/类设计、体系结构设计、接口设计、部件及设计。

✧ 数据/类设计将分析类模型变换成类的实现和软件实现所需要的

数据结构。

- ✧ 体系结构设计定义了软件的整体结构，由软件部件、外部可见的属性和他们之间的关系组成。
- ✧ 接口设计描述了软件内部、软件和协作系统之间以及软件同人之间的通信方式。
- ✧ 部件级设计将软件体系结构的结构性元素变换为对软件部件的过程性描述。

4.2 软件设计与质量的关系是怎么样的？

设计是在软件开发中形成质量的阶段，设计提供了可以用于质量评估的软件表示，是将用户需求准确的转化为完整的软件产品或系统的主要途径。

4.4 简述模块、模块化及模块化设计的概念。

在软件工程中模块是数据说明、可执行语句等程序对象的集合，具有名字、参数、功能等外部特征以及完成模块功能的程序代码和模块内部数据等内部特征。

模块化，即把软件按照规定原则，划分为一个个较小的，相互独立的但又相互关联的部件，实际上是系统分解和抽象的过程。

模块化设计，简单地说就是程序的编写不是开始就逐条录入计算机语句和指令，而是首先用主程序、子程序、子过程等框架把软件的主要结构和流程描述出来，并定义和调试好各个框架之间的输入、输

出链接关系。逐步求精的结果是得到一系列以功能块为单位的算法描述。以功能块为单位进行程序设计，实现其求解算法的方法称为模块化。模块化的目的是为了降低程序复杂度，使程序设计、调试和维护等操作简单化。

4.6 耦合和软件可移植性的概念有何关系？举例说明自己的结论。

所谓“耦合性”是指模块之间联系的紧密程度的一种度量，而软件的“可移植性”是指将一个软件系统从一个计算机系统或环境移植到另一个计算机系统或环境中运行时所需工作量的大小。可移植性是用一组子特性，包括简明性、模块独立性、通用性、可扩充性、硬件独立性和软件系统独立性等，来衡量的。如果一个软件具有可移植性，它必然耦合性低，这样模块独立性要强。例如，有一个图形处理软件，它应具有二维几何图形处理、三维几何图形处理、图形显示、外设控制、数据库管理、用户界面控制、设计分析等模块。如果这些模块之间都是通过参数表来传递信息，那么它们之间的的耦合就是数据耦合或标记耦合等，都是低耦合。将来如果想要把它们移植到另一个外部环境中，这些模块容易修改（功能内聚），且接口清晰，修改可局部化。反言之，如果这些模块都是功能内聚或信息内聚的模块，模块之间的耦合都是低耦合，也对可移植性有促进。但不能讲具有低耦合性模块结构的软件一定具有可移植性，因为是否具有可移植性还有其它因素的影响。

4.7 用自己的话描述信息隐蔽概念,并讨论信息隐藏与模块独立两概

念之间的关系。

信息隐藏指在设计和确定模块时，使得一个模块内包含的特定信息（过程或数据），对于不需要这些信息的其他模块来说，是透明的。“隐藏”的意思是，有效的模块化通过定义一组相互独立的模块来实现，这些独立的模块彼此之间仅仅交换那些为了完成系统功能所必需的信息，而将那些自身的实现细节与数据“隐藏”起来。信息隐蔽为软件系统的修改、测试及以后的维护都带来好处。 通过抽象，可以确定组成软件的过程实体。通过信息隐藏，可以定义和实施对模块的过程细节和局部数据结构的存取限制。模块独立的概念是模块化、抽象、信息隐藏和局部化概念的直接结果。开发具有独立功能而且和其他模块之间没有过多的相互作用的模块，就可以做到模块独立。

4.8 什么是模块独立性？设计中为什么模块要独立？如何度量独立性？模块功能独立有何优点？

（1）模块的独立性是模块化、信息隐藏和局部化等概念的直接结果。

（2）模块的独立性是很重要的：第一，功能被划分，并且接口被简化，所以具有有效模块化的软件易于开发。第二，由于因设计和编码修改引起的副作用受到局限，错误传播被减小，并且模块复用成为可能，所以独立的模块更易于维护和测试。总的来说，模块独立是良好设计的关键，从而又是保证软件质量的关键。

（3）用内聚度与耦合度来度量独立性。内聚度度量同一个模块内部

各个元素彼此结合的紧密程度，耦合度度量不同模块彼此间相互以来的紧密程度。

（4） 模块功能独立的优点：系统容易开发，系统可靠性高，系统易于维护，软件结构清晰。

4.9 软件设计规约主要包括哪些内容？

1.工作范围

2.体系结构设计

3.数据设计

4.接口设计

5.各部件的过程设计

6.运行设计

7.出错处理设计

8.安全保密设计

9.需求/设计交叉索引

10.测试部分

11.特殊注解

12.附录

第 5 章 结构化分析与设计

5.1 简述数据流图的主要思想，概述使用数据流图进行需求分析的过程。

数据流图(DFD)描述输入数据流到输出数据流的变换(即加工),用于对系统的功能建模。

数据流图可以用来抽象地表示系统或软件。它从信息传递和加工的角度，以图形的方式刻画数据流从输入到输出的移动变换过程，同时可以按自顶向下、逐步分解的方法表示内容不断增加的数据流和功能细节。因此，数据流图既提供了功能建模的机制，也提供了信息流建模的机制，从而可以建立起系统或软件的功能模型。

数据流图进行需求分析的过程：

1） 画出系统的输入和输出

2） 画出系统内部

3） 画出加工内部

4）重复第三步，直至每个尚未分解的加工都足够简单（即不必再分解）

5.2 分别采用数据流方法中得哪些技术来完成用户需求的精确化、一致化和完全化的任务。

1）父图和子图平衡

2）数据守恒

3）局部文件

4）一个加工的输入数据流不能与该加工的输入数据流同名

5）每个加工至少有一个输入数据流和一个输出数据流。

6）在整套分层数据流中，每个文件应至少有一个加工读该文件，有另一个加工写该文件。

7）分层数据流图中得每个数据流和文件都必须命名（除了流入或流出文件的数据流），并且与数据字典一致。

8）分层 DFD 中的每个基本加工（即不再分解子图的加工）都应有一个加工规约。

5.4 在数据流图中，可否将两个加工用一个数据流相连？可否将两个源用一个数据流相连？为什么？

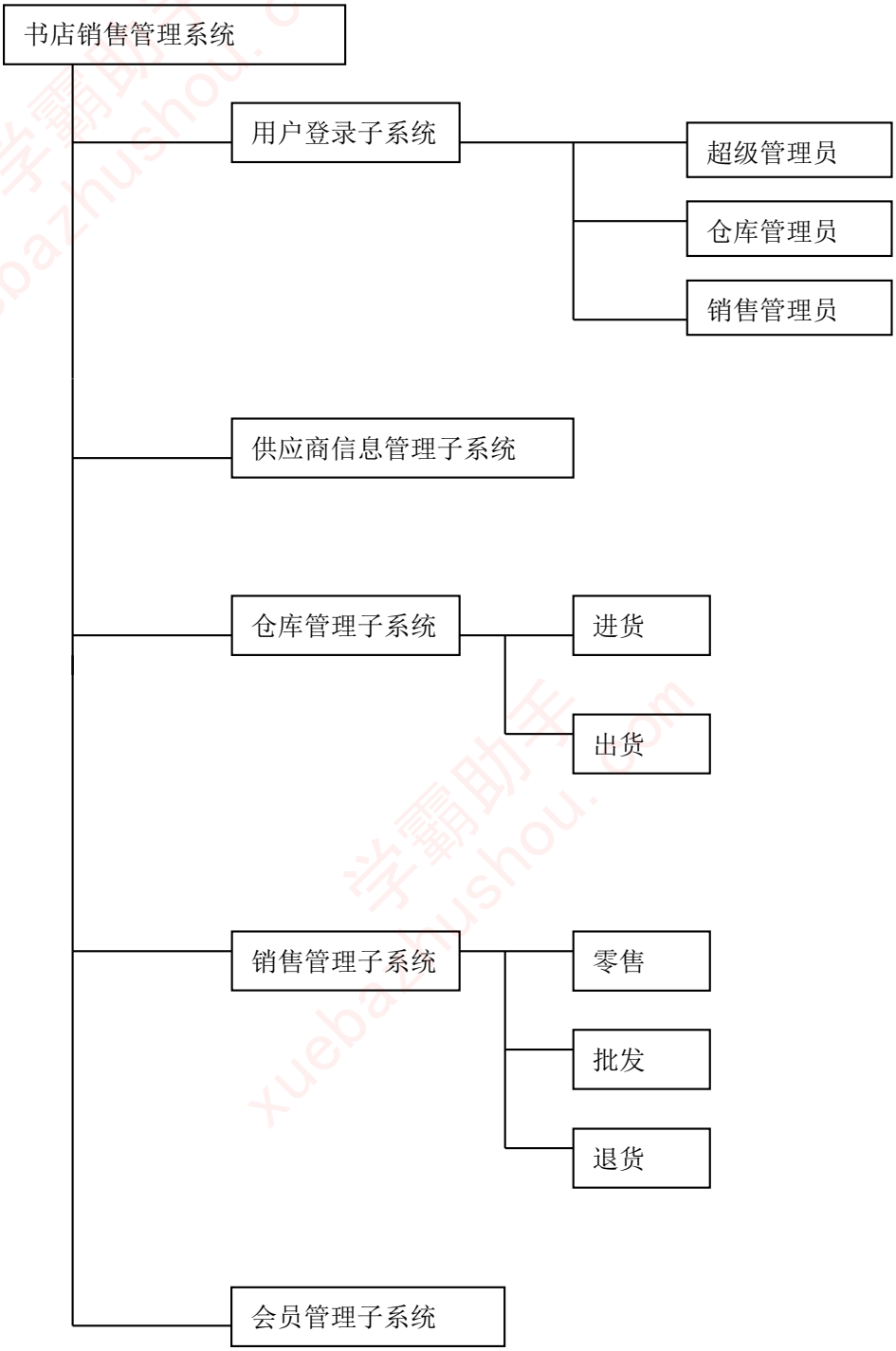
两个加工可以直接用数据流相连，两个源不能直接用数据流相连。因为数据流由一组固定成分的数据组成。在 DFD 中，数据流的流向可以有以下几种：从一个加工流向另一个加工，从加工流向文件（写文件），从文件流向加工（读文件），从源流向加工，从加工流向宿。

5.7

采用结构化分析方法写出书店管理系统的需求文档，包括数据

流图及数据字典。看到这个题目，我想起了以前自己也在手机端Android 平台写了一个书店的管理系统 ,不过那个时候根本没有什么需求分析，只是自己一厢情愿的模拟了一个简单的流程。也没有采用什么结构化分析方法，就仅仅描述了几个对象及其功能。所以我觉得可以对照着新学的软件工程的知识运用到自己的实际项目中去，同时也可以完成这道相识的题目。

下图就是以前的功能结构图：

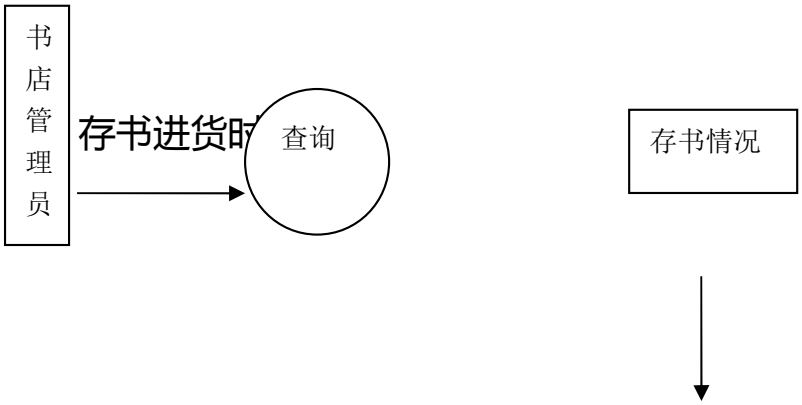


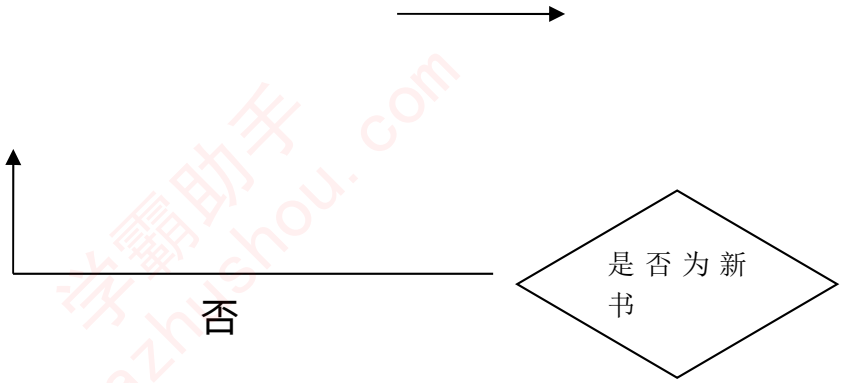
还有一些程序运行的界面：

利用结构化方法分析：

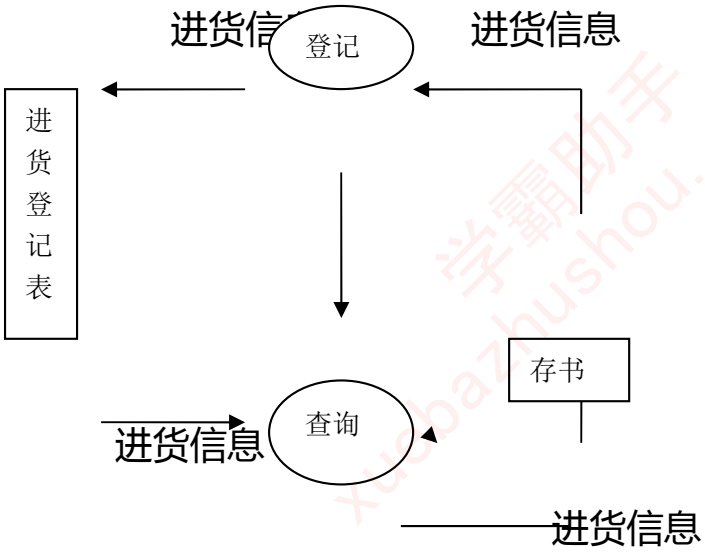
1、 数据流程图

第一， 存书分数据流图：

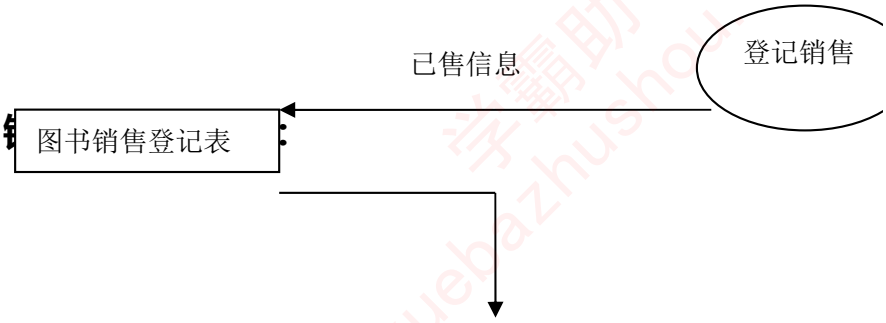


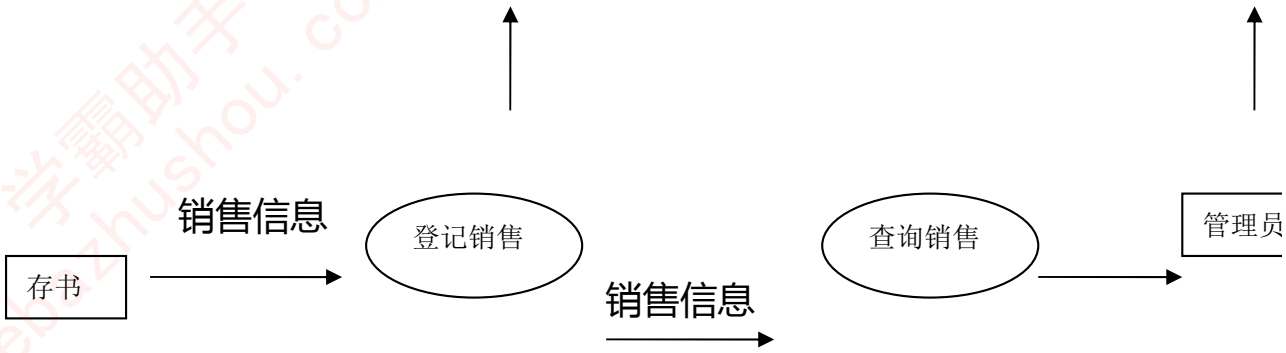


第二， 进货分数据流图：

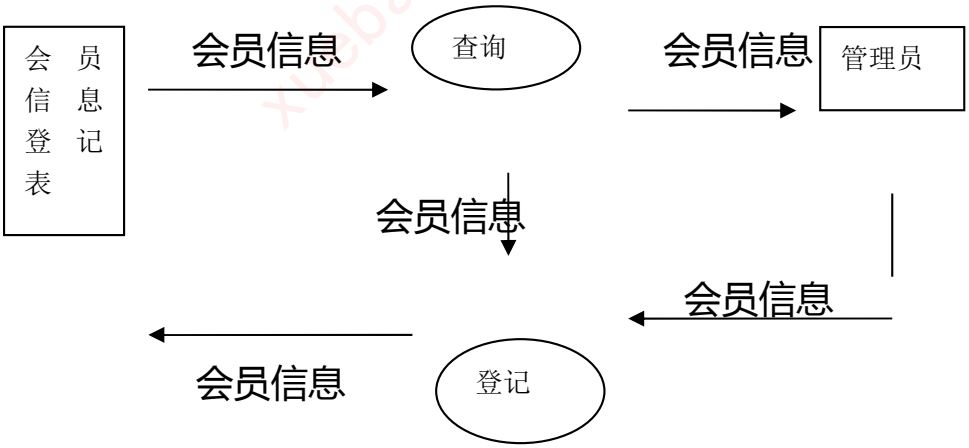


第三，

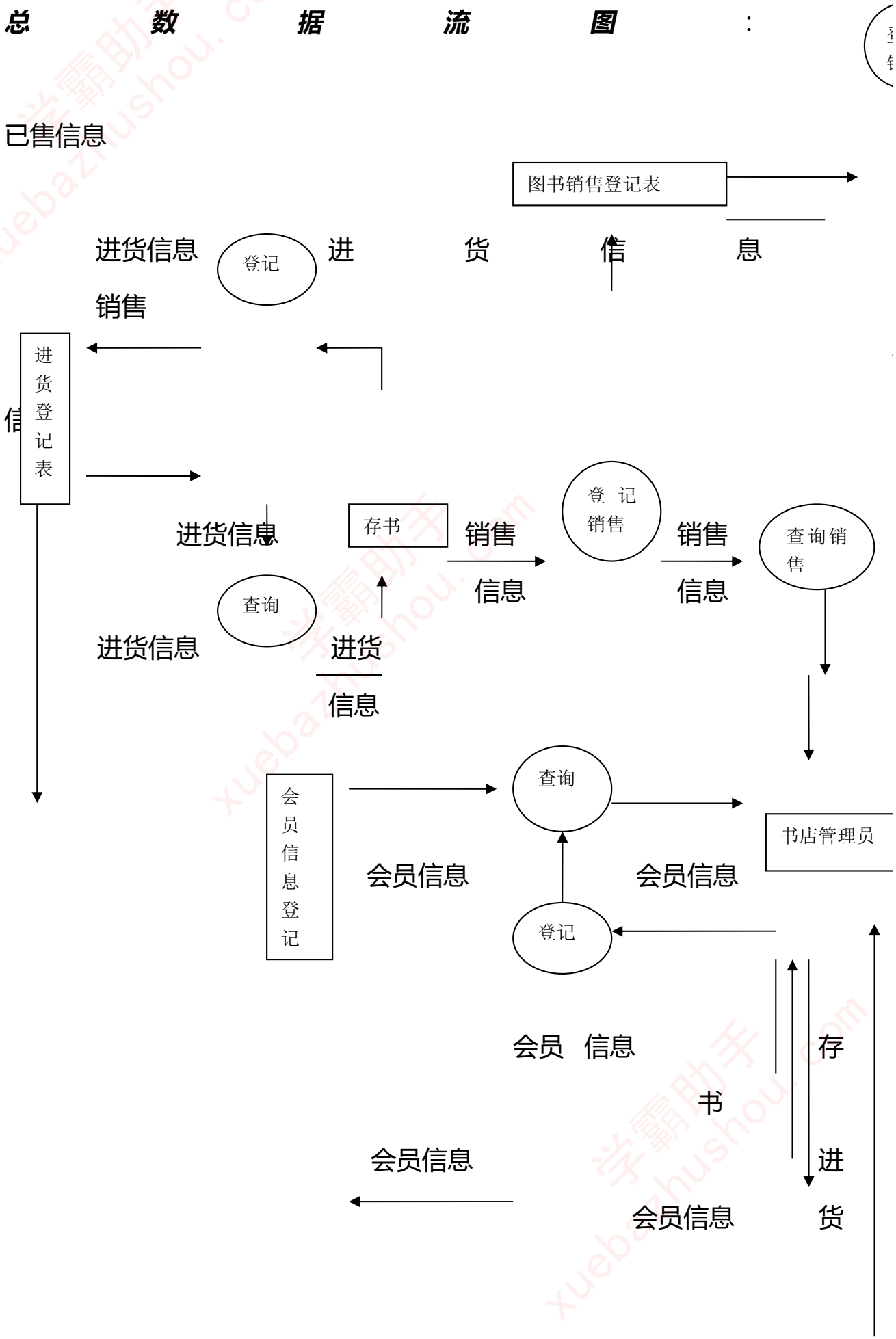


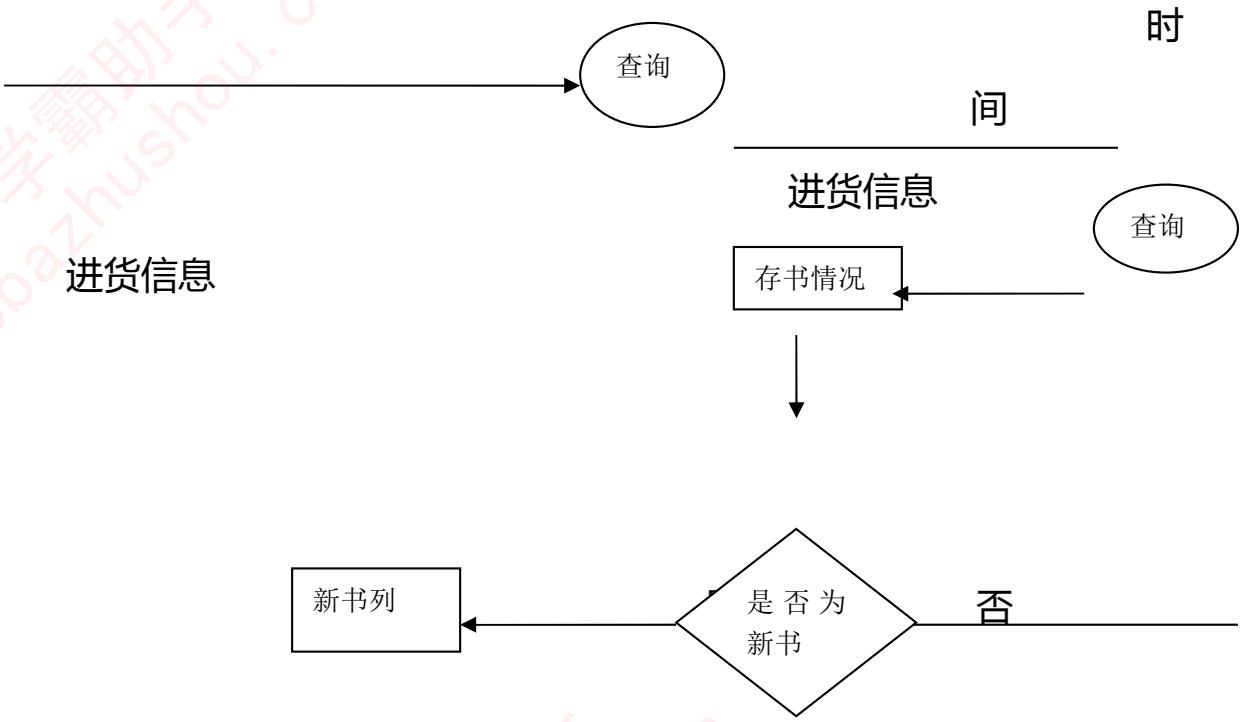


第四，　　会员分数据流图：



总　　数　　据　　流　　图　　：





2、**数据字典**

存书数据字典：

属 性 名	类 型	长 度	备 注
书 号	char	10	书的编号
书 名	char	30	书的名称
作 者	char	20	书的写作者
类 型	char	20	书的类型
简 述	char	150	书的简单介绍

单 价	smallmoney	4	书的价格
现 存 数 量	int	4	店内的现有存书数量
店 内 位 置	char	20	书在店内的具体位置
出 版 社 号	char	10	出版社的编号
特价书	char	2	是否为特价书（缺省值是“否”）

出版社数据字典：

属 性 名	类 型	长 度	备 注
出版社号	char	10	出版社的编号
出版社名称	char	20	出版社的名称
所在城市	char	10	出版社所在的城市
电话	char	15	出版社的联系电话

进货数据字典：

属 性 名	类 型	长 度	备 注
书 号	char	10	书的编号
进 价	smallmoney	4	书的进价
进货数量	int	4	每本书的进货数量

日 期	datetime	8	进货的日期

销售数据字典：

属 性 名	类 型	长 度	备 注
日 期	datetime	8	售货的日期
书 号	char	10	书的编号
销售量	int	4	售出的书本数量
销售金额	money	8	销售金额=（单价*销售量）

会员信息数据字典：

属 性 名	类 型	长 度	备 注
客服号	char	8	会员卡的卡号
姓名	char	20	会员的姓名
地址	char	50	会员的家庭住址
电话	char	15	会员的电话号码
购买书号	char	10	书的编号

店内收出数据字典：

属 性 名	长 度	备 注
月份	6	
水电支出	4	水电费用

员工支出	4	员工的工资
其他支出	4	其他费用支出
结算	8	每月的总结算

2、 数据结构

数据结构名	组成
存书信息	书号，书名，作者，简介，单价，店内位置，类型，现存数量，出版社号，特价书
出版社号	出版社号，出版社名称，电话，所在城市
进货信息	书号，进货数量，进价，进货日期
销售信息	销售日期，书号，销售量，销售金额
会员信息	购买书号，客服号，会员姓名，地址，电话
书店支出信息	月份，水电支出，员工支出，其他支出，结算

3、 数据流

数据流名	数据流来源	数据流去向	组成
登记新书信息	进货信息	存书信息	新书信息
查询新书信息	新书登记信息	管理员	新书信息
登记打折书信息	存书信息	销售信息	存书信息
查询打折书信息	打折书登记信息	管理员	存书信息

.登记会员信息	会员信息	存书信息	会员信息
查询会员信息	会员登记信息	管理员	会员信息

4、 数据存储

数据存储名	输入的数据流	输出的数据流	组成
新书登记表	进货信息，是否为新书	存书信息	进货信息，存书信息
打折书登记表	存书信息，是否为打折书	销售信息	存书信息，销售信息
会员登记表	会员信息，销售信息	会员信息	会员信息，销售信息

第 6\8 章

6.1 简述面向数据结构方法的特点

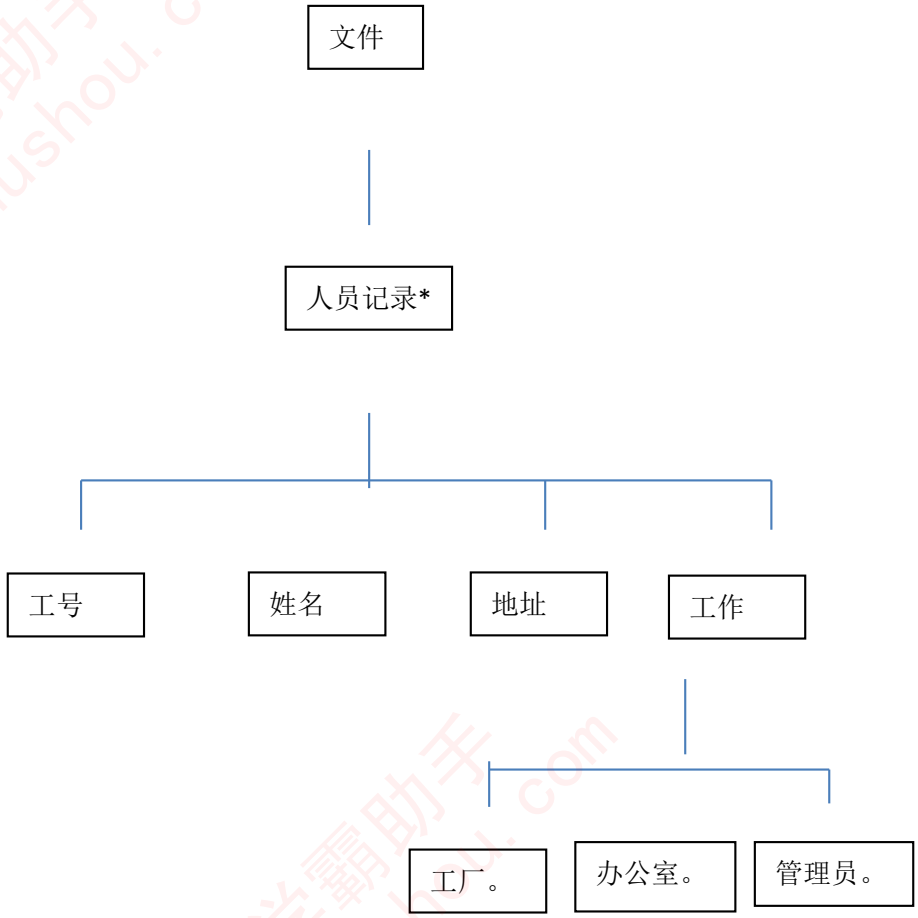
答：特点如下：

1 以信息对象及其操作作为核心进行需求分析；

2 认为复合信息对象具有层次结构，并且可按顺序，选择，重复 3 种结构分解为成员对象信息；

3 提供由层次信息结构映射为程序结构的机制，从而为软件设计奠定良好的基础。

6.2 采用 Jackson 图表示下面的文件结构：



8.1 什么是构件？

答：

根据 pressman 书中的定义

构件是某系统中有价值的、几乎独立的并可替换的一个部分，它在良好定义的体系结构语境内满足某种清晰的功能。

根据 brown 的定义

构件是一个独立发布的功能部分，可以通过其接口访问它的服务。

根据《计算机科学技术百科全书（第二版）》中的定义

软件构件是软件系统中具有相对独立功能，可以明确标识，接口由规约指定，与语境有明显依赖关系，可独立部署，且多由第三方提供的可组装软件实体。

软件构件须承载有用的功能，并遵循某种构件模型。可复用构件是指具有可复用价值的构件。

在基于构件的软件开发中经常会使用到的商用成品构件，是指由第三方开发的满足一定构件标准并且可组装的软件构件。

8.2 简述基于构件的软件开发过程。

基于构件的软件开发过程：

领域工程的步骤：

1 领域分析

2 建立领域特定的基准体系结构模型

3 标识候选构件

4 泛化和可变性分析

5 构件重构

6 构件的测试

7 构件的包装

8 构件入库

应用系统工程的步骤：

1　建立应用系统的体系结构模型；

2　寻找候选构件；

3　评价和选择合适的构件；

4　构件的修改和特化；

5　开发未被复用的不分；

6　构件的组装；

7　集成测试；

8　评价被复用的构件，并推荐可能的新构件。

第 11、13 章

11.1

答：软件测试的目的是发现软件中的错误和缺陷，并加以纠正。

11.2

答：白盒测试又称结构测试，这种方法把测试对象看作一个透明的盒子，测试人员根据程序内部的逻辑结构及有关信息设计测试用例，检查程序中所有逻辑路径是否都按预定的要求正确地工作。白盒测试主要用于对程序模块的测试。包括：

- 程序模块中的所有独立路径至少执行一次。
- 对所有逻辑判定的取值（“真”与“假”）都至少测试一次。
- 在上下边界及可操作范围内运行所有循环。

- 测试内部数据结构的有效性等。

黑盒测试又称行为测试，这种方法把测试对象看作一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符号它的功能需求。黑盒测试可用于各种测试，它试图发现以下类型的错误：

- 不正确或遗漏的功能
- 接口错误，如输入输出参数的个数、类型等。
- 数据结构错误或外部信息（如外部数据库）访问错误。
- 性能错误。
- 初始化和终止错误。

11.3

解：判定覆盖：（1）X=85,Y=85

（2）X=70，Y=95

（3）X=30，Y=95

条件覆盖：（1）X=85，X=85

（2）X=70，Y=75

（3）X=95，Y=50

（4）X=50，Y=95

（5）X=40，Y=40

判定条件覆盖：（1）X=85，X=85

（2）X=70，Y=75

（3）X=95，Y=50

（4）X=50，Y=95

（5）X=40，Y=40

（6）X=20，Y=95

（7）X=95，Y=20

条件组合覆盖：（1）X=85，X=85

（2）X=65，Y=85

（3）X=85，Y=65

（4）X=70，Y=75

（5）X=95，Y=50

（6）X=50，Y=95

（7）X=40，Y=40

路径覆盖：（1）X=85,Y=85

（2）X=70，Y=95

（3）X=30，Y=70

11.5 分别简述单元测试、集成测试、确认测试和系统测试的任务。

单元测试：

又称模块测试 着重对软件设计的最小单元——软件构件或模块进行验证。

单元测试根据设计描述，对重要的控制路径进行测试，已发现构建或

模块内部的错误，通常采用白盒测试，并且多个构件或模块可以并行测试。

单元测试的主要内容：接口、局部数据结构、边界条件、独立路径和错误处理路径。

集成测试：

集成测试，也叫组装测试或联合测试。在单元测试的基础上，将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，进行集成测试。实践表明，一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。程序在某些局部反映不出来的问题，在全局上很可能暴露出来，影响功能的实现。

目的：是确保各单元组合在一起后能够按既定意图协作运行，并确保增量的行为正确。它所测试的内容包括单元间的接口以及集成后的功能。使用黑盒测试方法测试集成的功能。并且对以前的集成进行回归测试。

确认测试：

确认测试的目的是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是确认测试的任务，即软件的功能和性能如同用户所合理期待的那样。

系统测试：

将已经确认的软件、计算机硬件、外设、网络等其他元素结合在一起，进行信息系统的各种组装测试和确认测试，系统测试是针对整个产品系统进行的测试，目的是验证系统是否满足了需求规格的定义，找出与需求规格不符或与之矛盾的地方，从而提出更加完善的方案。系统测试发现问题之后要经过调试找出错误原因和位置，然后进行改正。是基于系统整体需求说明书的黑盒类测试，应覆盖系统所有联合的部件。对象不仅仅包括需测试的软件，还要包含软件所依赖的硬件、外设甚至包括某些数据、某些支持软件及其接口等。

11.6 什么是α测试？什么是β测试？

α测试：

α测试是由一个用户在开发者的场所进行的测试，软件在开发者对用户的“指导下”进行测试。经过α测试后的软件称为β测试。

β测试：

β测试是指软件开发公司组织各方面的典型用户在日常工作中实际使用β版本，并要求用户报告异常情况、提出批评意见，然后软件开发公司再对β版本进行改错和完善。β测试也是黑盒测试。黑盒测试也称功能测试，它是通过测试来检测每个功能是否都能正常使用。

11．7 什么是回归测试？

回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。自动回归测试将大幅降低系统测试、维护升级等阶段的成本。回归测试作为软件生命周期的一个组成部分，在整个软件测试过程中占有很大的工作量比重，软件开发的各个阶段都会进行多次回归测试。在渐进和快速迭代开发中，新版本的连续发布使回归测试进行的更加频繁，而在极端编程方法中，更是要求每天都进行若干次回归测试。因此，通过选择正确的回归测试策略来改进回归测试的效率和有效性是非常有意义的。

- 观念：

1.回归测试是指重复以前的全部或部分的相同测试。

2.新加入测试的模组，可能对其他模组产生副作用，故须进行某些程度的回归测试。

3.回归测试的重心，以关键性模组为核心。

- 测试策略：

对于一个软件开发项目来说，项目的测试组在实施测试的过程中会将所开发的测试用例保存到“测试用例库”中，并对其进行维护和管理。当得到一个软件的基线版本时，用于基线版本测试的所有测试用例就形成了基线测试用例库。在需要进行回归测试的时候，就可以根据所选择的回归测试策略，从基线测试用例库中提取合适的测试用例

组成回归测试包，通过运行回归测试包来实现回归测试。保存在基线测试用例库中的测试用例可能是自动测试脚本，也有可能是测试用例的手工实现过程。

回归测试需要时间、经费和人力来计划、实施和管理。为了在给定的预算和进度下，尽可能有效率和有效力地进行回归测试，需要对测试用例库进行维护并依据一定的策略选择相应的回归测试包。

- 测试过程

有了测试用例库的维护方法和回归测试包的选择策略，回归测试可遵循下述基本过程进行:

(1). 识别出软件中被修改的部分；

(2). 从原基线测试用例库 T 中，排除所有不再适用的测试用例，确定那些对新的软件版本依然有效的测试用例，其结果是建立一个新的基线测试用例库 T0。

(3). 依据一定的策略从 T0 中选择测试用例测试被修改的软件。

(4). 如果必要，生成新的测试用例集 T1，用于测试 T0 无法充分测试的软件部分。

(5). 用 T1 执行修改后的软件。

第(2)和第(3)步测试验证修改是否破坏了现有的功能，第(4)和第(5)步测试验证 修改工作本身。

11．8 简述边界值分析方法的作用

长期的测试工作经验告诉我们，大量的错误是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。

使用边界值分析方法设计测试用例，首先应确定边界情况。通常输入和输出等价类的边界，就是应着重测试的边界情况。应当选取正好等于，刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值作为测试数据。

边界分析是指对输入或输出的边界值进行测试的一种测试方法。所说的边界值是指输入等价类和输出等价类的边界值。

经验证明大量的程序错误是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。

使用边界值分析方法设计测试用例，首先应确定边界情况。通常输入和输出等价类的边界，就是应着重测试的边界情况。应当选取正好等于，刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值作为测试数据。

13.1 请讨论使软件维护成本居高不下的因素。如何尽可能降低这些因素的影响？

软件维护的代价是生产率惊人下降。维护费用只不过是软件及维护最明显的代价，起一些隐性的代价将更为人们关注。

软件维护除费用外的无形代价包括：

1.维护活动占用了其他软件开发可用的资源，使资源的利用率降低

2.一些修复或修改请求得不到及时安排，使得客户满意率下降

3.维护的结果把一些新的潜在的错误引入软件，降低了软件质量

4.将软件人员抽调到维护工作中，使得其它软件开发过程受到干扰

影响维护工作量的因素主要有以下六种：

1.系统的规模：系统规模越大，其功能就越复杂，软件维护的工作量也随之增大

2.程序设计语言：使用强功能的程序设计语言可以控制程序的规模。语言的功能越强，生成程序的模块化和结构化程度越高，所需的指令数就越少，程序的可读性也越好

3.系统年龄：老系统比新系统需要更多的维护工作量。

4.数据库技术的应用：使用数据库，可以简单而有效地管理和存储用户程序中的数据，还可以减少生成用户报表应用软件的维护工作量

5.先进的软件开发技术：在软件开发过程中，如果采用先进的分析设计技术和程序设计技术，如面向对象技术、复用技术等，可减少大量的维护工作量

6.其它一些因素：如应用的类型、数学模型、任务的难度、IF 嵌套深度、索引或下标数等，对维护工作量也有影响

为了有效的进行软件维护，尽可能降低这些因素的影响，应事先就开始做组织工作：

1.首先建立维护的机构

2.申明提出维护申请报告的过程及评价的过程

3.为每一个维护申请规定标准的处理步骤

4.建立维护活动的登记制度以及规定评价和评审的标准

13.3 软件维护的过程是如何进行的？为什么要进行软件可维护性分析？

1.对于非纠错性维护，则首先判断维护类型，对适应性维护，按照评估后得到的优先级放入队列

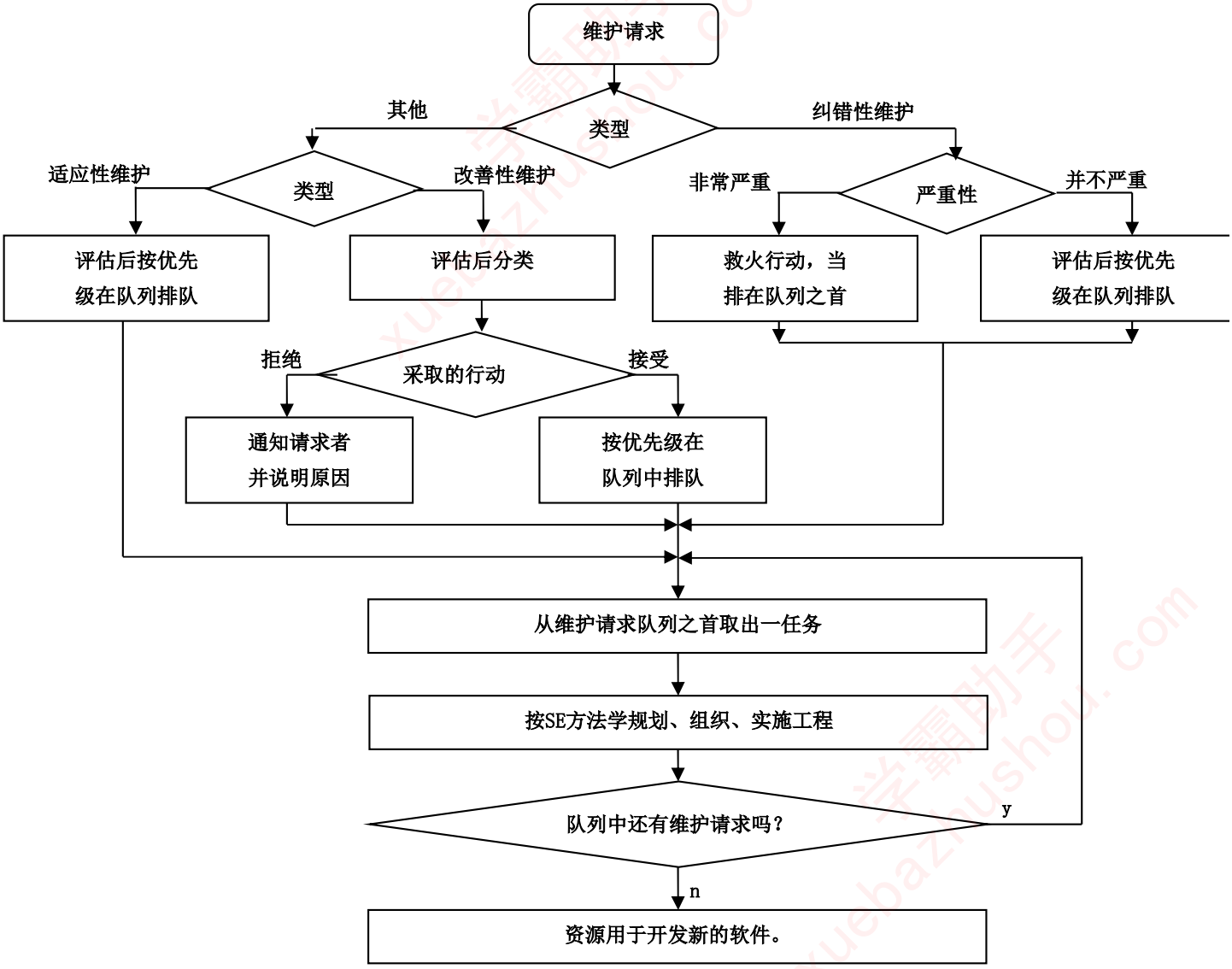
2.对于改善性维护，则还要考虑是否采取行动，如果接受申请，则同样按照评估后得到的优先级放入队列 ,如果拒绝申请 ,则通知请求者，并说明原因

3.对于工作安排队列中的任务，由修改负责人依次从队列中取出任务，按照软件工程方法学规划、组织、实施工程。

4.每种维护请求都要进行同样的一系列技术工作：修改软件需求说明、修改软件设计、设计评审、必要时重新编码、单元测试、集成测试(包括回归测试)、确认测试等

5.维护工作最后一步是复审

维护过程图：



进行可维护性分析的原因：

软件维护是指软件系统交付使用以后，为了改正错误或满足新的需求而修改软件的过程。一个中等规模的软件，如果其开发过程需要一二年时间，则它投入使用以后，其运行时间可能持续 5～10 年之久。在这个维护阶段中，人们需要着手解决开发阶段尚未解决的问题，同时，还解决维护工作本身所产生的问题。做好软件的维护工作不仅能够排除软件中存在的错误，使它能够正常工作，而且还可以使它扩充功能，提高性能，为用户带来新的效益。软件的可维护性是指维护人员为纠正软件系统出现的错误或缺陷，以及为满足新的要求而理解、修改和完善软件系统的难易程度。可维护性是所有软件系统都应具备的特点。在软件工程的每一阶段都应该努力提高系统的可维护性，在每个阶段结束前的审查和复审中，应着重对可维护性进行复审。维护阶段的花费约占整个软件生命期花费的 67%。因此，应充分认

识到维护现有软件的重要意义。

3.7 在重构和正向工程之间存在的细微不同是什么？

当某应用的基本体系结构是坚固的时候发生重构，即使技术的内部细节需要修改。当软件的大部分是有用的，仅仅部分模块和数据需要扩展性修改时，启动重构活动。正向工程是通过到实现语言的映射而把模型转换为代码的过程。正向工程过程应用软件工程的原理、概念、技术和方法来重新开发某个现有的应用系统。