第一章

（1.1软件危机）

1、软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。

2、软件危机包含下述两方面的问题：（1）如何开发软件，以满足对软件日益增长的需求；

（2）如何维护数量不断膨胀的已有软件。

3、软件危机主要有以下一些典型表现：

（1）对软件开发成本和进度的估计常常很不准确。

（2）用户对“已完成的”软件系统不满意的常常很不准确、

（3）软件产品的质量往往靠不住。

（4）软件常常是不可维护的。

（5）软件通常没有适当的文档资料。

（6）软件成本在计算机系统总成本中的比例逐年上升。

（7）软件开发生产率提高的速度，远远跟不上计算机应用迅速普及深入的趋势。

4、在软件开发和维护的过程中存在这么多严重问题，（1）一方面与软件本身的特点有关，（2）另一方面也和软件开发与维护的方法不正确有关。

5、软件是程序、数据及相关文档的完整集合。其中，（1）程序时能够完成预定功能和性能的可执行的指令序列；（2）数据是使程序能恰当地处理信息的数据结构；（3）文档是开发、使用和维护程序所需要的图文资料。

（1.2软件工程）

7、概括地说，软件工程是指导计算机软件开发和维护的一门工程学科。采用工程的概念、原理、技术和方法来开发和维护软件，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，以经济地开发出高质量的软件并有效地维护它，这就是软件工程。

8、软件工程的7条基本原理

（1）用分阶段的生命周期计划严格管理

（2）坚持进行阶段评审

（3）实行严格的产品控制

（4）采用现代程序设计技术

（5）结果应能清楚地审查

（6）开发小组的人员应该少而精

（7）承认不断改进软件工程实践的必要性

9、管理就是通过计划、组织和控制等一系列活动，合理地配置和使用各种资源，以达到既定的目标过程

10、软件工程方法学包括3个要素：方法、工具和过程。其中，（1）方法是完成软件开发的各项任务的技术方法，回答“怎么做”的问题；（2）工具是为运用方法而提供的自动的或半自动的软件工程支撑环境；（3）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。

（1.3软件生命周期）

10、软件生命周期由软件定义、软件开发和运行维护（也称为软件维护）3个时期组成，每个时期又进一步划分成若干个阶段。

（1）软件定义时期的任务是：确定软件开发工程必须完成的总目标；确定工程的可行性；导出实现工程目标应该采用的策略及系统必须完成的功能；估计完成该项工程需要的资本和成本，并且制定工程进度表。这个时期的工作通常又称为系统分析，由系统分析员负责完成。

（2）开发时期具体设计和实现在前一个时期定义的软件。

（3）维护时期的主要任务是使软件持久地满足用户的需求。

11、生命周期每个阶段的基本任务。

（1）问题定义

（2）可行性研究

（3）需求分析

（4）总体设计

（5）详细设计

（6）编码和单元测试

（7）综合测试

（8）软件维护

（1.4软件过程）

12、各种模型的特征、优缺点、使用范围。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 优点 | 缺点 | 使用范围 |
| 瀑布模型 | （1）有利于大型软件开发过程中人员的组织、管理，  （2）有利于软件开发方法和工具的研究，从而提高了大型软件项目开发的质量和效率。 | （1）开发过程一般不能逆转，否则代价太大；  （2）实际的项目开发很难严格按该模型进行；  （3）客户往往很难清楚地给出所有的需求，而该模型却要求如此。  （4）软件的实际情况必须到项目开发的后期客户才能看到，这要求客户有足够的耐心。 | （1）用户的需求非常清楚全面，且在开发过程中没有或很少变化；  （2）开发人员对软件的应用领域很熟悉；  （3）用户的使用环境非常稳定；  （4）开发工作对用户参与的要求很低。 |
| 快速原型模型 | （1）可以得到比较良好的需求定义，容易适应需求的变化；  （2）有利于开发与培训的同步；  （3）开发费用低、开发周期短且对用户更友好。 | （1）客户与开发者对原型理解不同；  （2）准确的原型设计比较困难；  （3）不利于开发人员的创新。 | （1）对所开发的领域比较熟悉而且有快速的原型开发工具；  （2）项目招投标时，可以以原型模型作为软件的开发模型；  （3）进行产品移植或升级时，或对已有产品原型进行客户化工作时，原型模型是非常适合的。 |
| 增量模型 | （1）采用增量模型的优点是人员分配灵活，刚开始不用投入大量人力资源；  （2）如果核心产品很受欢迎，则可增加人力实现下一个增量；  （3）可先发布部分功能给客户，对客户起到镇静剂的作用。 | （1）并行开发构件有可能遇到不能集成的风险，软件必须具备开放式的体系结构；  （2）增量模型的灵活性可以使其适应这种变化的能力大大优于瀑布模型和快速原型模型，但也很容易退化为边做边改模型，从而是软件过程的控制失去整体性。 | （1）进行已有产品升级或新版本开发，增量模型是非常适合的；  （2）对完成期限严格要求的产品，可以使用增量模型；  （3）对所开发的领域比较熟悉而且已有原型系统，增量模型也是非常适合的。 |

第二章

13、可行性研究的目的，就是用最小的代价在尽可能短的时间内确定问题是否能解决。

（2.4数据流图）

14、数据流图（DFD）是一种图形化技术，它描绘信息流和数据从输入移动到输出的过程中所经受的变换。

15、数据流图有4种成分：源点或终点，处理，数据存储和数据流。

！16、画图

（2.5数据字典）

17、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的定义的集合。

第三章

18、在需求分析阶段结束之前，系统分析员应该写出软件需求规格说明书，以书面形式准确地描述软件需求。

（3.2与用户沟通获取需求的方法）

19、访谈有两种形式，分别是正式和非正式的访谈。（1）正式访谈时，系统分析员将提出一些事先准备好的具体问题（2）在非正式访谈中，分析员将提出一些用户可以自由回答的开放性问题，以鼓励被访问人员说出自己的想法。

（3.4实体-联系图）

！20、画图

21、实体-联系图（entity-relationship diagram）来建立数据模型。

22、ER图中包含了实体（即数据对象）、关系和属性3种基本成分。（范式要基本了解）

（3.6状态转换图）

23、状态转换图（简称为状态图）通过描绘系统的状态及引起系统状态转换的事件，来表示系统的行为。

24、状态图提供了行为建模机制。

25、在状态图中定义的状态主要有：初态（即初始状态）、终态（即最终状态）和中间状态。在一张状态图仲只能有一个初态，而终态则可以有0个至多个。

（3.7其他图形工具）

26、IPO图式输入、处理、输出图的简称。

27、它的基本形式是在左边的框中列出有关的输入数据，在中间的框内列出主要的处理，在右边的框内列出产生的输出数据。

第五章

28、总体设计又称为概要设计或初步设计。（1）通过这个阶段的工作将划分出组成系统的物理元素——程序、文件、数据库、人工过程和文档等。（2）总体设计阶段的另一项重要的任务是设计软件的结果，也就是要确定系统中每个程序是由哪些模块组成的，以及这些模块相互间的关系。

29、总体设计过程

（1）寻找实现目标系统的各种不同的方案，需求分析阶段得到的数据流图是设想各种可能方案的基础。

（2）然后分析员从这些供选择的方案中选取若干个合理的方案。

（3）分析员应该综合分析比较这些合理的方案，从中选出一个最佳方案向用户和使用部门负责人推荐。

（4）进一步为这个最佳方案设计软件结构，通常，设计出初步的软件结构后还要多方改进。

（5.1设计过程）

30、总体设计过程通常由两个主要阶段完成：（1）系统设计阶段，确定系统的具体实现方案；（2）结构设计阶段，确定软件结构。

（5.2设计原理）

31、模块是由边界元素限定的相邻程序元素（例如，数据说明，可执行的语句）的序列，而且有一个总体标示符表示它。

32、模块化就是把程序划分成独立命名且可独立访问的模块，每个模块完成一个子功能。

33、在现实世界中一定事物、状态或过程之间总存在着某些相似的方面（共性）。把这些相似的方面集中和概括起来，暂时忽略它们之间的差异，这就是抽象。或者说抽象就是抽出食物的本质特性而暂时不考虑它们的细节。

34、信息隐藏原理指出：应该这样设计和确定模块，使得一个模块内包含的信息（过程和数据）对于不需要这些信息的模块来说，是不能访问的。

35、耦合式对一个软件结构内不同模块之间互连成都的度量。

36、在软件设计中应该追求尽可能松散耦合的系统。

37、总之，耦合式影响软件复杂程度的一个重要因素。应该采取下述设计原则：尽量使用数据耦合，少用控制耦合和特征耦合，限制公共环境耦合的范围，完全不用内容耦合。

38、内聚标志着一个模块内各个元素彼此结合的紧密程度。

39、设计时应该力求做到高内聚。

（5.3启发规则）

40、深度表示软件结果中控制的层数。

42、宽度是软件结构内同一个层次上的模块总数的最大值。

43、扇出是一个模块直接控制（调用）的模块数目。

44、一个模块的扇入表明有多少个上级模块直接调用它。

45、设计得好的软件结构通常顶层扇出比较高，中层扇出较少，底层扇入到公共的实用模块中去（底层模块有高扇入）。

第六章

（6.1结构程序设计）

46、结构程序设计的经典定义如下所述：“如果一个程序的代码仅仅通过顺序、选择和循环这3种基本控制台结构进行连接，并且每个代码只有一个入口和一个出口，则称这个程序是结构化的。”

（6.3过程设计的工具）

47、坚持使用盒图作为详细设计的工具，可以使程序员逐步养成结构化的方式思考问题和解决问题的习惯。

48、判定表能够清晰地表示复杂的条件组合与应做的动作之间的对应关系。

49、一张判定表由4部分组成，左上部列出所有条件，左下部是所有可能做的动作，右上部是表示各种条件组合的一个矩阵，右下部是和每种条件组合相对应的动作。

！50、画判定表判定树

（6.4面向数据结构的设计方法）

51、面向数据结构的设计方法的最终目标是得出对程序处理过程的描述。

52、Jackson图表现的是组成关系，是层次结构图。

53、Jackson结构程序设计方法基本上由下述5个步骤组成。

（1）分析并确定输入数据和输出数据的逻辑结构，并用Jackson图描绘这些数据结构。

（2）找出输入数据结构和输出数据结构中有对应关系的数据单元。

（3）综合输入数据结构和输出数据结构的层次关系到处Jackson图。

（4）列出所有操作和条件，并分配到程序结构图的适当位置。

（5）用伪码表示程序。

（6.5程序复杂程度的定量度量）

54、程序复杂度定量度量方法是其中比较成熟的一种。

55、定量度量的结果可以用来比较两个不同的设计或两个不同算法的优劣。

第七章

56、把编码和测试统称为实现。

（7.2软件测试基础）

57、G.Myers给出了关于测试的一些规则，这些规则也可以看作是测试的目标或定义。

（1）测试是为了发现程序中的错误而执行程序的过程。

（2）好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案。

（3）成功的测试是发现了至今为止尚未发现的错误的测试。

58、软件测试准则

（1）所有测试都应该能追溯到用户需求。

（2）应该远在测试开始之前就制定出测试计划。

（3）把Pareto原理应用到软件测试中。

（4）应该从“小规模”测试开始，并逐步进行“大规模”测试。

（5）穷举测试是不可能的。

（6）为了达到最佳的测试效果，应该由独立的第三方从事测试工作。

39、黑盒测试

（1）在程序接口进行的测试（2）又称为功能测试

40、白盒测试

（1）按照程序内部的逻辑测试程序，检测程序中的主要执行通路是否都能按预定的要求正确工作（2）又称为结构测试。

41、测试步骤

（1）模块测试

（2）子系统测试

（3）系统测试

（4）验收测试

（5）平行运行

（7.4集成测试）

42、一般说来，纯粹自顶向下或纯粹自底向上的策略可能都不实用，人们在实践中创造出许多混合策略。

（1）改进的自顶向下测试方法。基本上使用自顶向下的测试方法，但是在早期使用自底向上的方法测试软件中的少数关键模块。

（2）混合法。对软件结构中较上层使用自顶向下方法与对软件结果中较下层使用的自底向上方法相结合。

43、回归测试就是用于保证由于调试或其他原因引起的变化，不会导致非预期的软件行为或额外错误的测试活动。

（7.5确认测试）

44、Alpha测试由用户在开发者的场所进行，并且在开发者对用户的“指导”下进行测试。

45、Beta测试由软件的最终用户们在一个或多个客户场所进行。与Alpha测试不用，开发者通常不在Beta测试的现场。

（7.6白盒测试技术）

46、语句覆盖：选择足够多的测试数据，使被测程序中每个语句至少被执行一次。

47、判定覆盖（分支覆盖）：不仅每个语句必须至少执行一次，而且每个判定的每种可能的结果都应该至少执行一次。

48、条件覆盖：不仅每个语句至少执行一次，而且使判定表达式中的每个条件都取到各种可能的结果。

49、判定/条件覆盖：选取足够多的测试数据，使得判定表达式仲的每个条件都取到各种可能的值，而且每个判定表达式也都去到各种可能的结果。

50、条件组合覆盖是更强的逻辑覆盖标准，它要求选取足够多的测试数据，使得每个判定表达式中的条件的各种可能组合都至少出现一次。

51、路径覆盖：选取足够多测试数据，是程序的每条可能的路径都至少执行一次（如果程序图中有环，则要求每个环至少经过一次）。

（7.7黑盒测试）

52、黑盒测试力图发现下述类型的错误：

（1）功能不正确或遗漏了功能。

（2）界面错误。

（3）数据结构错误或外部数据库访问错误。

（4）性能错误。

（5）初始化和终止错误。

（6）划分出等价类以后，根据等价类设计测试方案时主要使用下面两个步骤。

设计一个新的测试方案以尽可能多地覆盖尚未被覆盖的有效等价类，重复这一步骤直到所有有效等价类都被覆盖为止。

设计一个新的测试方案，使它覆盖一个而且只覆盖一个尚未被覆盖的无效等价类，重复这一步骤直到所有有效等价类都被覆盖为止。

（7.8调试）

53、调试（又称为纠错）作为成功测试的后果出现，也就是说，调试是在测试发现错误之后排除错误的过程。

（7.9软件可靠性）

54、软件可靠性的定义：程序在给定的时间间隔内，按照规格说明书的规定成功地运行的概率。

55、软件可用性的定义：程序给定的时间点，按照规格说明书的规定，成功地运行的概率。

56、可靠性和可用性之间的主要差别：可靠性意味着0到t这段时间间隔内系统没有失效，而可用性只意味着在时刻t，系统是正常运行的。

57、估计错误总数的方法

（1）植入错误法

（2）分别测试法

第八章

（8.1软件维护的定义）

58、软件维护就是在软件已经交付使用之后，为了改正错误或满足新的需要而修改软件的过程。

（8.4软件的可维护性）

59、决定软件可维护性的因素主要有5个。

（1）可理解性

（2）可测试性

（3）可修改性

（4）可移植性

（5）可重用性

习题

1-7 什么是软件生命周期？试比较瀑布模型、快速原型模型、增量模型和螺旋模型的优缺点，说明每种模型的适用范围。

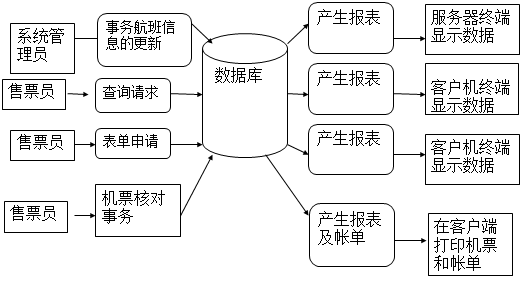
软件生命周期是指从软件定义、开发、使用、维护到淘汰的全过程。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 优点 | 缺点 | 使用范围 |
| 瀑布模型 | （1）有利于大型软件开发过程中人员的组织、管理，  （2）有利于软件开发方法和工具的研究，从而提高了大型软件项目开发的质量和效率。 | （1）开发过程一般不能逆转，否则代价太大；  （2）实际的项目开发很难严格按该模型进行；  （3）客户往往很难清楚地给出所有的需求，而该模型却要求如此。  （4）软件的实际情况必须到项目开发的后期客户才能看到，这要求客户有足够的耐心。 | （1）用户的需求非常清楚全面，且在开发过程中没有或很少变化；  （2）开发人员对软件的应用领域很熟悉；  （3）用户的使用环境非常稳定；  （4）开发工作对用户参与的要求很低。 |
| 快速原型模型 | （1）可以得到比较良好的需求定义，容易适应需求的变化；  （2）有利于开发与培训的同步；  （3）开发费用低、开发周期短且对用户更友好。 | （1）客户与开发者对原型理解不同；  （2）准确的原型设计比较困难；  （3）不利于开发人员的创新。 | （1）对所开发的领域比较熟悉而且有快速的原型开发工具；  （2）项目招投标时，可以以原型模型作为软件的开发模型；  （3）进行产品移植或升级时，或对已有产品原型进行客户化工作时，原型模型是非常适合的。 |
| 增量模型 | （1）采用增量模型的优点是人员分配灵活，刚开始不用投入大量人力资源；  （2）如果核心产品很受欢迎，则可增加人力实现下一个增量；  （3）可先发布部分功能给客户，对客户起到镇静剂的作用。 | （1）并行开发构件有可能遇到不能集成的风险，软件必须具备开放式的体系结构；  （2）增量模型的灵活性可以使其适应这种变化的能力大大优于瀑布模型和快速原型模型，但也很容易退化为边做边改模型，从而是软件过程的控制失去整体性。 | （1）进行已有产品升级或新版本开发，增量模型是非常适合的；  （2）对完成期限严格要求的产品，可以使用增量模型；  （3）对所开发的领域比较熟悉而且已有原型系统，增量模型也是非常适合的。 |

2.3 为方便旅客，某航空公司拟开发一个机票预订系统。旅行社把预订机票的旅客信息（姓名、性别、工作单位、身份证号码、旅行时间、旅行目的地等）输入进该系统，系统为旅客安排航班，印出取票通知和账单，旅客在飞机起飞的前一天凭取票通知和账单交款取票，系统校对无误即印出机票给旅客。

写出问题定义并分析系统的可行性。

答：



处理流程和数据流程

2-4 目前住院病人主要由护士护理，这样做不仅需要大量护士，而且由于不能随时观察危重病人的病情变化，还会延误抢救时机。某医院打算开发一个以计算机为中心的患者监护系统，请分层次地画出描述本系统功能的数据流图。