1. 软件  
　　软件是计算机系统中与硬件相互依存的部分，它是包括程序、数据及相关文档的完整集合。  
  
2. 软件危机  
　　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。  
  
3. 软件工程  
　　软件工程是研究和应用如何以系统化的、规范的、可度量的方法去开发、运行和维护软件，即把工程化应用到软件上。  
  
4. 软件生存周期  
　　软件生存周期是指软件产品从考虑其概念开始到该软件产品交付使用，直至最终退役为止的整个过程，一般包括计划、分析、设计、实现、测试、集成、交付、维护等阶段。  
  
5. 软件复用  
　　软件复用就是利用某些已开发的、对建立新系统有用的软件元素来生成新的软件系统。   
  
6. 质量  
　　质量是产品或服务满足明确或隐含需求能力的特性和特征的集合。在合同环境下，需求是明确的；在其他环境下，隐含的需求需要识别和定义。  
  
7. 质量策划  
　　质量策划包括产品策划、管理和作业策划，以及质量计划的编制和质量改进的准备工作。  
  
8. 质量改进  
　　质量改进是以最求最高的效益和效率为目标的持续性活动。  
  
9. 质量控制  
　　质量控制是对流程和产品的符合性的评估，独立分析不足并予以更正使得产品与需求相符。  
  
10. 质量保证  
　　质量保证是有计划的和系统性的活动，它对部件或产品满足确定的技术需求提供足够的信心。   
  
11. 软件质量  
　　软件质量是指明确声明的功能和性能需求、明确文档化的开发标准、以及专业人员开发的软件所具有的所有隐含特征都得到满足。  
  
12. 正式技术复审  
　　正式技术复审是一种由软件开发人员进行的软件质量保证活动，其目的是在软件的任何一种表示形式中发现功能、逻辑或实现的错误，验证经过复审的软件确实满足需求，保证软件符合预定义的标准，使软件按照一致的方式开发，使项目更易于管理。  
  
13. ISO  
　　ISO是一个组织的英语简称，代表International Organization for Standardization，即"国际标准化组织"。  
  
14. ISO9000  
　　ISO9000是由ISO/TC176制定的关于质量管理和质量保证的国际标准。  
  
15. 质量认证  
　　质量认证是由可以充分信任的第三方证实某一经鉴定的产品或服务符合特定标准或规范性文件的活动。  
  
16. 软件过程  
　　软件过程是人们用于开发和维护软件及其相关过程的一系列活动，包括软件工程活动和软件管理活动。  
  
17. 软件过程能力  
　　软件过程能力是描述（开发组织或项目组）遵循其软件过程能够实现预期结果的程度，它既可对整个软件开发组织而言，也可对一个软件项目而言。  
  
18. 软件过程性能  
　　软件过程性能表示（开发组织或项目组）遵循其软件过程所得到的实际结果，软件过程性能描述的是已得到的实际结果，而软件过程能力则描述的是最可能的预期结果，它既可对整个软件开发组织而言，也可对一个特定项目而言。  
  
19. 软件过程成熟度  
　　软件过程成熟度是指一个特定软件过程被明确和有效地定义，管理测量和控制的程度。  
  
20. 软件成熟度等级  
　　软件成熟度等级是指软件开发组织在走向成熟的途中几个具有明确定义的表示软件过程能力成熟度的平台。

21. 关键过程域  
　　每个软件能力成熟度等级包含若干个对该成熟度等级至关重要的过程域，它们的实施对达到该成熟度等级的目标起到保证作用，这些过程域就称为该成熟度等级的关键过程域。  
  
22. 关键实践  
　　关键实践是指对关键过程域的实践起关键作用的方针、规程、措施、活动以及相关基础设施的建立。  
  
23. 软件能力成熟度模型  
　　软件能力成熟度模型是指随着软件组织定义、实施、测量、控制和改进其软件过程，软件组织的能力也伴随着这些阶段逐步前进，完成对软件组织进化阶段的描述模型。  
  
24. 软件需求  
　　软件需求是指  
　　（1）用户解决问题或达到目标所需的条件或能力；  
　　（2）系统或系统部件要满足合同、标准、规范或其它正式规定文档所需具有的条件或能力；  
　　（3）一种反映上面（1）或（2）所描述的条件或能力的文档说明。  
  
25. 业务需求  
　　业务需求（business requirement）反映了组织机构或客户对系统或产品高层次的目标要求，它们在项目视图与范围文档中予以说明。  
  
26. 用户需求  
　　用户需求（user requirement）描述了用户使用产品必须要完成的任务，可以在用例模型或方案脚本中予以说明。  
  
27. 功能需求  
　　功能需求（functional requirement）定义了开发人员必须实现的软件功能，使得用户能完成他们的任务，从而满足了业务需求。   
  
28. 非功能需求  
　　非功能需求（non-functional requirement）是从各个角度对系统的约束和限制，反映了应用对软件系统质量和特性的额外要求。  
  
29. 需求工程  
　　需求工程是应用已证实有效的原理和方法，通过合适的工具和符号，系统地描述出待开发系统及其行为特征和相关约束。  
  
30. 需求分析  
　　需求分析主要是对收集到的需求进行提炼、分析和仔细审查，以确保所有的风险承担者都明白其含义并找出其中的错误、遗漏或其它不足的地方，形成完整的分析模型。  
  
31. 软件需求规格说明  
　　软件需求规格说明是需求开发的最终结果，它精确地阐述一个软件系统必须提供的功能和性能以及它所要考虑的限制条件。软件需求规格说明不仅是系统测试和用户文档的基础，也是所有子系列项目规划、设计和编码的基础。   
  
32. 风险承担人  
　　风险承担人是任何将从新系统或应用的实现中受到实质性影响的人。  
  
33. 软件原型  
　　软件原型是所提出的新产品的部分实现，其目的是为了解决在产品开发的早期阶段需求不确定的问题。  
  
34. 实体关系图  
　　实体关系图描述数据对象及其关系。  
  
35. 数据流图  
　　数据流图是结构化分析的基本工具，它描述了信息流和数据转换。  
  
36. 状态转换图  
　　状态转换图通过描述状态以及导致系统改变状态的事件来表示系统的行为。  
  
37. 数据字典  
　　数据字典描述数据流图的数据存储、数据加工（最底层加工）和数据流。  
  
38. 对象  
　　对象（Object）是系统中用来描述客观事物的一个实体，它是构成系统的一个基本单位，由一组属性和对这组属性进行操作的一组服务组成。  
  
39. 类  
　　类（Class）是具有相同属性和服务的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述，其内部包括属性和服务两个主要部分。  
  
40. 封装  
　　封装（Encapsulation）是把对象的属性和服务结合成一个独立的系统单位，并尽可能隐藏对象的内部细节。

41. 继承  
　　继承（Inheritance）是指子类可以自动拥有父类的全部属性和服务。  
  
42. 消息  
　　消息（Message）是对象发出的服务请求，一般包含提供服务的对象标识、服务标识、输入信息和应答信息等信息。  
  
43. 多态性  
　　多态性（Polymorphism）是指在父类中定义的属性或服务被子类继承后，可以具有不同的数据类型或表现出不同的行为。  
  
44. 主动对象  
　　主动对象（Active Object）是一组属性和一组服务的封装体，其中至少有一个服务不需要接收消息就能主动执行（称为主动服务）。  
  
45. 面向对象分析  
　　面向对象的分析（OOA）就是运用面向对象的方法进行需求分析，其主要任务是分析和理解问题域，找出描述问题域和系统责任所需的类及对象，分析它们的内部构成和外部关系，建立OOA模型。  
  
46. 面向对象设计  
　　面向对象的设计（OOD）就是根据已建立的分析模型，运用面向对象技术进行系统软件设计。它将OOA模型直接变成OOD模型，并且补充与一些实现有关的部分，如人机界面、数据存储、任务管理等。  
  
47. 面向对象编程  
　　面向对象的编程（OOP）就是用一种面向对象的编程语言将OOD模型中的各个成分编写成程序。  
  
48. 面向对象测试  
　　面向对象的测试（OOT）是指对于运用OO技术开发的软件，在测试过程中继续运用OO技术进行以对象概念为中心的软件测试。   
  
49. 统一建模语言UML  
　　统一建模语言（Unified Modeling Language，UML）是一种直观化、明确化、构建和文档化软件系统产物的通用可视化建模语言。  
  
50. 用例图  
　　用例图定义了系统的功能需求，它完全是从系统的外部观看系统功能，并不描述系统内部对功能的具体实现。  
  
51. 类图  
　　类图描述系统的静态结构，表示系统中的类以及类与类之间的关系。  
  
52. 对象图  
　　对象图描述了一组对象以及它们之间的关系，表示类的对象实例。  
  
53. 状态图  
　　状态图表示一个状态机，强调对象行为的事件顺序。  
  
54. 时序图  
　　时序图表示一组对象之间的动态协作关系，反映对象之间发送消息的时间顺序。  
  
55. 协作图  
　　协作图表示一组对象之间的动态协作关系，反映收发消息的对象的结构组织。  
  
56. 活动图  
　　活动图反映系统中从一个活动到另一个活动的流程，强调对象间的控制流程。  
  
57. 组件图  
　　组件图描述组件以及它们之间的关系，表示系统的静态实现视图。  
  
58. 分布图  
　　分布图反映了系统中软件和硬件的物理架构，表示系统运行时的处理节点以及节点中组件的配置。  
  
59. 软件体系结构  
　　软件体系结构包括一组软件部件、软件部件的外部的可见特性及其相互关系，其中软件外部的可见特性是指软件部件提供的服务、性能、特性、错误处理、共享资源使用等。  
  
60. 软件测试  
　　软件测试是以最少的时间和人力系统地找出软件中潜在的各种错误和缺陷。  
  
61. 静态测试  
　　静态测试是指被测试程序不在机器上运行，而是采用人工检测和计算机辅助静态分析的手段对程序进行检测。  
  
62. 动态测试  
　　动态测试是指通过运行程序发现错误，一般意义上的测试主要是指动态测试。  
  
63. 黑盒测试  
　　黑盒测试也称功能测试或数据驱动测试，它是在已知产品所应具有功能的情况下，通过测试来检测每个功能是否都能正常使用。  
  
64. 白盒测试  
　　白盒测试也称结构测试或逻辑驱动测试，它是知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能。  
  
65. 软件调试  
　　软件调试则是在软件测试成功后，根据错误迹象确定错误的原因和准确位置，并加以改正。  
  
66. 软件测试自动化  
　　测试自动化是通过开发和使用一些工具自动测试软件系统，特别适合于测试中重复而繁琐的活动。  
  
67. 软件维护  
　　软件维护是指在软件运行或维护阶段对软件产品所进行的修改。  
  
68. 改正性维护  
　　在软件交付使用后，由于开发时测试得不彻底或不完全，在运行阶段会暴露一些开发时未能测试出来的错误。为了识别和纠正软件错误，改正软件性能上的缺陷，避免实施中的错误使用，应当进行的诊断和改正错误的过程，这就是改正性维护。  
  
69. 适应性维护  
　　随着计算机技术的飞速发展和更新换代，软件系统所需的外部环境或数据环境可能会更新和升级，如操作系统或数据库系统的更换等。为了使软件系统适应这种变化，需要对软件进行相应的修改，这种维护活动称为适应性维护。  
  
70. 完善性维护  
　　在软件的使用过程中，用户往往会对软件提出新的功能与性能要求。为了满足这些要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。这种情况下进行的维护活动叫做完善性维护。  
  
71. 预防性维护  
　　预防性维护是指采用先进的软件工程方法对需要维护的软件或软件中的某一部分重新进行设计、编制和测试，提高软件的可维护性和可靠性等，为以后进一步改进软件打下良好基础。  
  
72. 软件的可维护性  
　　软件的可维护性是指软件能够被理解、纠正、适应和完善以适应新环境的难易程度。  
  
73. 项目  
　　项目就是以一套独特而相互联系的任务为前提，有效地利用资源，为实现一个特定的目标所做的努力。  
  
74. 项目管理  
　　项目管理就是通过合理地组织和利用一切可以利用的资源，按照计划的成本和计划的进度，完成一个计划的目标，它包含团队管理、风险管理、采购管理、流程管理、时间管理、成本管理和质量管理等。