中华人民共和国国家标准

GB/T 11457—1995

软 件 工 程 术 语

代替 GB 11457—89

Software engineering terminology

引言

本标准结构如下：

a. 词条按英文对应词字母顺序排列；

b. 如果一个术语有一个以上的定义，则分别加以说明；

c. 凡必要的地方用例子来说明定义；

d. 为了说明本标准中一个术语与另一些术语的关系，使用了下述词语：

——比较……：指补充性的术语；

——与……相对照：指一个具有相反含义的或本质上不同意义的术语；

——与……同义：指同义的术语；

——参见……：指让读者参见推荐使用的或与之关系密切的术语。

——还可参见……：指一有关术语。

1 主题内容与适用范围

本标准定义软件工程领域中通用的术语，适用于软件开发、使用维护、科研、教学和出版等方面。

2 术语

2.1 夭折，异常终止 abort

在一过程完成之前被迫终止。

2.2 绝对机器代码 absolute machine code

每次使用时必须装入固定存储单元且不能再定位的机器语言代码。与 2.399 条相对照。

2.3 抽象机 abstract machine

a. 过程或机器的一种表示。

b. 一个模块，它象一台机器那样处理输入。

2.4 抽象 abstraction

a. 对某一问题的概括。它抽取与某一特定目标相关的本质的内容而忽略非本质的内容。

b. 形成上述抽象的过程。

2.5 验收准则 acceptance criterion

软件产品要符合某一测试阶段必须满足的准则，或软件产品满足交货要求的准则。

2.6 验收测试 acceptance testing

确定一系统是否符合其验收准则，使客户能确定是否接收此系统的正式测试。参见 2.381 条、2.497 条。

2.7 可接近性 accessibility

使组成软件的各部分便于选择使用或维护的程度。

国家技术监督局 1995-05-04 批准　　1995-12-01 实施

2.8 访问控制机制 access-control mechanism

为使某一计算机系统或计算机系统的某一部分允许被获准者和防止未获准者接触、访问而设计的硬件或软件的特性、操作过程或管理过程。

2.9 准确，准确度 accuracy

a. 无误差的一种品质。

b. 无误差程序的一种定性估计，估计越高，对应的误差越小。

c. 对误差大小的一种度量，最好表示成相对误差的函数，其准确度越高，对应的误差越小。

d. 对无误差程度的一种定量估计。与 2.341 条相对照。

2.10 需方 acquirer

从供方获得或得到一个系统、产品或服务的一个机构。

注：需方可以是买主、客户、拥有者、用户、采购人圆荆。

2.11 获取 acquisition

得到一个系统、一个产品或一项服务的过程。

2.12 活动文件 active file

尚未超过终止时间的文件。

2.13 活动 activity

一个过程的组成元素。

注：对基线的改变要经有关当局的正式批准。

2.14 实参 actual parameter

在调用子程序时用来指定数据或要传输给该子程序的程序元素的数值或表达式。与 2.211 条相对照。

2.15 适应性 adaptability

使不同的系统约束条件和用户需求得到满足的容易程度。

2.16 适应性维护 adaptive maintenance

为使软件产品在改变了的环境下仍能使用而进行的维护。

2.17 地址 address

a. 标识一寄存器、存储器特定部分、或其他一些数据来源或目的地的一个或一组字符。

b. 用来指定一设备或一个数据项。

2.18 地址空间 address space

计算机程序可以有效利用的地址范围。

2.19 算法 algorithm

a. 用有限步数求解某问题的一套明确定义的规则的集合；例如，求 $\sin(x)$ 到给定精度的一系列算术运算的完整的说明。

b. 定义良好的规则的有限集合，它给出完成一特定任务的运算序列。

2.20 算法分析 algorithm analysis

对一算法的检查。目的在于确定与其预期的用途有关的正确性，确定其运行特性，或为了更充分地理解某一算法以便对其进行修改、简化或改进。

2.21 别名 alias

a. 某一项目的另一个名字。

b. 一个替换标号。例如，可以使用一个标号和一个或多个别名来指示计算机程序中同一数据元素或点。

2.22 分析阶段 analysis phase

参见 2.406 条。

2.23 分析模型 analytical model

用一组可解方程来表示一个过程或一个现象。与 2.430 条相对照。

2.24 面向应用的语言 application-oriented language

a. 一种面向计算机的语言，具有用于某种单一应用领域的手段或记号；例如，用于统计分析或机器设计的语言。

b. 一种面向问题的语言，其语句包含或汇集了用户职业的术语。

2.25 应用软件 application software

解决属于专用领域的，非计算机本身问题的软件。

2.26 体系结构 architecture

参见 2.353 条、2.491 条。

2.27 体系结构设计 architectural design

a. 定义一组硬件和软件元素及其接口的过程，其目的是为开发一计算机系统而建立其主体结构。

b. 体系结构设计过程的结果。

2.28 人工语言 artificial language

参见 2.210 条。

2.29 汇编 assemble

把用汇编语言表示的程序翻译成机器语言，有时还要连接子程序。实现汇编的常用方法是用机器语言操作码代替汇编语言操作码，并用绝对地址、中间地址、浮动地址或虚拟地址来代替符号地址。与 2.72 条、2.254 条相对照。

2.30 汇编程序 assemb

用于进行汇编的计算机程序。与 2.73 条、2.255 条相对照。

2.31 汇编语言 assembly language

a. 一种面向计算机的语言，其指令与计算机指令通常是一一对应的，且能提供使用宏指令的便利。与 2.279 条、2.225 条相对照。参见 2.72 条、2.73 条。

b. 一种特定机器语言，其指令通常和计算机指令一一对应。

2.32 断言 assertion

一种逻辑表达式，规定必须存在的一种程序状态，或规定在程序执行过程中某一特定点上程序变量必须满足的条件集合，例如，A 为正且 $A>B$。参见 2.236 条、2.322 条。

2.33 赋值语句 assignment statement

用于表达一系列操作，或用于把操作数赋给指定变量，或符号，或变量和符号两者的指令。

2.34 审计 audit

a. 为评估是否符合软件需求、规格说明、基线、标准、过程、指令、代码以及合同和特殊要求而进行的一种独立的检查。参见 2.63 条。

b. 通过调查研究确定已制定的过程、指令、规格说明、代码和标准或其它的合同及特殊要求是否恰当和被遵守，以及其实现是否有效而进行的活动。

2.35 自动设计工具 automated design tool

帮助进行软件设计的综合、分析、模拟或文档编制的软件工具。自动设计工具的例子如：仿真器、分析工具、设计表示处理器和文件生成器。

2.36 自动测试用例生成器 automated test case generator

参见 2.38 条。

2.37 自动测试数据生成器 automated test data generator

参见 2.38 条。

2.38 自动测试生成器 automated test generator

一种软件工具，它以计算机程序和准则作为输入，产生满足这些准则要求的测试输入数据，有时还确定预期的结果。

2.39 自动验证系统 automated verification system

一种软件工具，以计算机程序及其规格的表示作为输入（可能借助人的帮助），产生该程序的正确与否的证明。参见 2.40 条。

2.40 自动验证工具 automated verification tools

用于评估软件开发过程中的产品的一类软件工具。这些工具有助于验证正确性、完全性、一致性、可跟踪性、可测试性，以及检查是否遵守了标准。软件验证工具包括设计分析器、自动验证系统、静态分析器、动态分析器和标准实施器。

2.41 可用性 availability

a. 软件在投入使用时能实现其指定的系统功能的概率。

b. 系统正常工作时间和总的运行时间之比。

c. 在运行时，某一配置项实现指定功能的能力。

2.42 可用性模型 availability model

用于预测、估计、判定可用性的模型。

2.43 后备，后援 back-up

发生系统失效或灾害时，为恢复数据文件或软件，重新起动处理，使用备份计算机设备而做的准备。

2.44 基线 baseline

a. 业已经过正式审核与同意，可用作下一步开发的基础，并且只有通过正式的修改管理步骤方能加以修改的规格说明或产品。

b. 在配置项目生存周期的某一特定时间内，正式指定或固定下来的配置标识文件和一组这样的文件。基线加上根据这些基线批准同意的改动构成了当前配置标识。对于配置管理，有以下三种基线：

功能基线——最初通过的功能配置；

分配基线——最初通过的分配的配置；

产品基线——最初通过的或有条件地通过的产品配置。

2.45 开始——结束块 begin-end block

由 begin 和 end 分隔符括起来的设计或程序语句序列。其特征是具有单一的入口和单一的出口。

2.46 协约（名），联编，约束，结合 binding

把一个值或指定的对象（referent）赋给某一标识符。例如，把一个值赋给一个参数或把一绝对地址、虚拟地址或设备标识符分配给计算机程序中的符号地址或标号。参见 2.166 条、2.470 条。

2.47 块（名），阻滞（动） block

a. 由某些技术或逻辑原因形成的被当作一个实体看待的一串记录、一串字或一字符串。

b. 作为一个单元而记录下来的一组连续的记录。块与块之间用间隙分隔，每一块可以包含一个或多个记录。

c. 被当作一个单元而加以传送的一组二进制位数或 N 进制位数。通常对这组二进制位数或 N 进制位数采用某种编码步骤以达到出错控制的目的。

d. 作为一个单元来处理的事物，如字、字符或数字的集合。

e. 参见 2.354 条。

f. 系统中的某些操作因某种原因，暂时不能继续执行。

2.48 框图 block diagram

表示某一系统、计算机或设备的图，图中主要部分由加有适当注释的几何图形来表示，用以说明这些主要部分的基本功能及其功能关系。与 2.209 条相对照。

2.49 块结构语言 block-structured language

一种程序设计语言，在这种语言中，语句序列通常是由 begin 和 end 界限符划界。参见 2.354 条。

2.50 引导程序 bootstrap

a. 一段短的计算机程序，常驻计算机或很容易装入计算机。引导程序的执行能把另一个较大的程序，如操作系统或其装入程序引入内存。

b. 一组指令，它能使另外的指令被装入直到全部计算机程序都存入存储器中为止。

c. 借助自身的动作而使其达到所希望的状态的一种技术或设备；例如，一段机器子程序，其前几条指令足以使其余部分指令从输入设备输入到计算机中。

d. 用于建立计算机程序另一版本的部分计算机程序。

e. 使用一引导程序。

2.51 引导装入程序 bootstrap loader

使用预置计算机操作以装入引导程序的一种输入例行程序。

2.52 自底向上 bottom-up

一种方法，这种方法从层次结构的最低层软件组成部分开始，逐级向上直至最高层组成成分为止，例如，自底向上设计、自底向上程序设计、自底向上测试等。与 2.526 条相对照。

2.53 自底向上设计 bottom-up design

从最基本的或原始的部分着手，逐级进入到较高层部分的系统设计方法。与 2.527 条相对照。

2.54 隐错，缺陷 bug

参见 2.198 条。

2.55 隐错撒播 bug seeding

参见 2.201 条。

2.56 构件 build

软件产品的一个工作版本，其中包含最终产品将拥有的能力的一个规定的子集。

2.57 构件块 building block

较高一级程序或模块使用的一个单元或模块。

2.58 (分)情况语句 case

能根据控制表达式的值对有限个程序语句进行选择性执行的多分支条件语句。参见 2.106 条。

2.59 认证 certification

a. 一个系统或计算机程序符合其规定的需求的一种书面保证。

b. 一种书面认可书，说明某计算机系统是可靠的，可以在一确定的环境中工作或产生合理的信息。

c. 为使系统获准投入运行性使用，对系统的可接受性所做的正式演示。

d. 证实一系统、软件子系统或计算机程序在其运行环境中能满足规定的需求的过程。认证通常在实际条件下的现场中进行，不仅用于估价软件本身，而且用于估价作为软件设计依据的规格说明。认证使验证和确认的过程扩充到实际的或模拟的运行环境中。

e. 一正式的权威机构根据可付诸实施的需求以书面形式确定、验证和证明人圆劲处理、过程或条款为合格所采取的步骤和行动。

2.60 链接表 chained list

一种表，在这种表中各个项目可以是分散的，但每项都含有指出下一项位置的标识符。与 2.269 条同义。

2.61 更动管理 change control

提议作一项更动并对其进行估计、同意或拒绝、调度和跟踪的过程。

2.62 代码，编码 code

a. 一组无歧义性的规则，它规定了使数据得以用某种离散形式加以表示的方式。

b. 用处理机可以接受的符号形式表示数据或计算机程序。

c. 书写例行程序。

d. 也可指一个或多个计算机程序，或计算机程序一部分。

e. 为了安全的目的对数据进行的加密表示。

2.63 代码审计 code audit

由某人、某小组、或借助某种工具对源代码进行的独立的审查，以验证其是否符合软件设计文件和程序设计标准。还可能对正确性和有效性进行估计。参见 2.34 条、2.468 条、2.237 条、2.545 条。

2.64 代码生成器 code generator

一个程序或程序功能，常常属于编译程序的一部分，它把计算机程序从某种中间级表示(通常为语法分析程序的输出)变换成较为低级的表示，如汇编代码或机器代码。

2.65 代码审查 code inspection

参见 2.237 条。

2.66 代码走查 code walk-through

参见 2.545 条。

2.67 内聚度 cohesion

单个程序模块所执行的诸任务在功能上的互相关联的程度。与 2.112 条相对照。

2.68 命令语言 command language

一组过程性的操作符及与之有关的语法，用来指明交给操作系统执行的功能。

2.69 注释 comment

a. 在计算机程序、命令语言或数据之间的说明信息，旨在给读者提供澄清性材料，并不影响机器的解释工作。

b. 加到或散置在源语言语句当中的描述、附注或解释，在目标语言中这些是无效的。

2.70 比较器 comparator

用来比较两个计算机程序、文件或数据集合的一种软件工具，目的是找出其共同点或不同的地方。比较的典型对象是源代码、目标(代)码、数据基文件的相似版本或测试结果。

2.71 兼容性 compatibility

a. 两个或两个以上系统运行同一软件可得到同样结果的能力。

b. 两个或两个以上系统处理同样的数据文件可得到同样结果的能力。

比较 2.253 条。

2.72 编译 compile

将高级语言程序变换成与之等价的浮动的或绝对的机器代码。与 2.29 条相对照。

2.73 编译程序 compiler

用于进行编译的一种计算机程序。与 2.30 条、2.255 条对照。

2.74 编译程序的编译程序 compiler compiler

参见 2.75。

2.75 编译程序的生成程序 compiler generator

用来构造编译程序的翻译程序或解释程序。与 2.290 条同义。

2.76 复杂性 complexity

系统或系统组成部分的复杂程度，由下述因素确定，如：接口的数量和错综程度，条件转移的数量

和错综程度，嵌套的深度，数据结构的类型，以及其它一些系统特性。

2.77 部件，组成部分 component

系统或程序的基本部分。

2.78 计算机 computer

a. 能执行大量计算，包括许多算术运算和逻辑运算，而在运行期间无需操作员干预的一种功能装置。

b. 由一台或多台相联的处理机和外围设备组成的一种可编程序的功能装置，这种装置由内部存储的程序控制，可执行大量的计算（许多算术运算和逻辑运算）而无需人的干预。

2.79 计算机数据 computer data

计算机设备和计算机设备之间或计算机设备内部通信用的数据。这种数据可以是外部的（计算机可读形式），也可以是驻留在计算机设备内的，可以是模拟信号，也可以是数字信号。

2.80 计算机网络 computer network

由两个或两个以上按一定的协议互连的计算机组成的复合体。

2.81 计算机程序 computer program

按照具体要求产生的适合于计算机处理的指令序列。

参见 2.352 条。

2.82 计算机程序摘要 computer program abstract

对计算机程序的简短叙述，给用户提供足够的信息，使他们能据此确定该计算机程序是否适合其需要及所拥有的资源。

2.83 计算机程序注释 computer program annotation

参见 2.69 条。

2.84 计算机程序认证 computer program certification

参见 2.59 条。

2.85 计算机程序配置标识 computer program configuration identification

参见 2.96 条。

2.86 计算机程序开发计划 computer program development plan

参见 2.441 条。

2.87 计算机程序确认 computer program validation

参见 2.538 条。

2.88 计算机程序验证 computer program verification

参见 2.539 条。

2.89 计算机系统 computer system

由一台或多台计算机和相关软件组成的一种功能装置。

2.90 并发进程 concurrent processes

可以同时地在多处理机上执行或异步地在单处理机上执行的若干进程。各并发进程可以相互作用，一个进程在接受另一进程的信息之前或一外部事件出现之前可以把执行挂起。与 2.426 相对照。

2.91 条件控制结构 conditional control structure

一种程序设计控制结构，它允许程序中使用根据指定条件的满足情况而加以选择的控制流。例如，按情况、如果……则……否则……。

2.92 配置 configuration

a. 计算机系统或网络按照其功能部件的特点、数量和主要特性而确定的排列。具体地讲，配置一词可以指硬件配置或软件配置。

b. 为确定系统或系统组成部分的特定版本而提出的需求、设计和实现。

c. 在技术文档中制定的并在产品中体现的硬件、软件的功能和(或)物理特性。

2.93 配置审计 configuration audit

证明所要求的全部配置项均已产生出来，当前的配置与规定的需求相符。技术文件说明书完全而准确地描述了各个配置项目，并且曾经提出的所有更动请求均已得到解决的过程。

2.94 配置控制 configuration control

a. 在配置项的配置标识正式确定之后，对配置项的更动情况所做的估价、协调、批准或不批准的过程。

b. 在配置项的配置标识正式确定之后，对配置项所进行的有系统的估价、协调、所表示的批准或不批准。以及配置中被批准的更动的具体实现过程。

2.95 配置控制委员会 configuration control board

对提出的工程上的更动负责进行估价、审批，对核准进行的更动确保其实现的权力机构。

2.96 配置标识 configuration identification

a. 标出系统中的配置项并对其特性进行记录的过程。

b. 经批准同意的确定一配置项的文件说明书。

c. 当前已批准的或有条件地批准的针对一配置项的技术文档说明，如载于规格说明中的图和相关的表及文档说明。

2.97 配置项 configuration item

a. 为了配置管理目的而作为一个单位来看待的硬件和/或软件成分。

b. 满足最终应用功能并被指名用于配置管理的硬件/软件，或它们的集合体。

配置项在复杂性、规模和型号上差异甚大，可从航空、电子或船舶系统到测试仪表甚至一发子弹。在开发和初始生产阶段，配置项就是合同中(或与之相当的内部协定中)直接引用的说明项。在运行和维护期间，被指明要分别获得的任何可维护的项也是配置项。

2.98 配置管理 configuration management

a. 标识和确定系统中配置项的过程，在系统整个生存周期内控制这些项的投放和更动，记录并报告配置的状态和更动要求，验证配置项的完整性和正确性。参见 2.61 条、2.96 条、2.94 条、2.99 条、2.93 条。

b. 对下列工作进行技术和行政指导与监督的一套规范：

——对一配置项的功能和物理特性进行标识和文件编制工作；

——控制这些特性的更动情况；

——记录并报告对这些更动进行的处理和实现的状态。

2.99 配置状态报告 configuration status accounting

记录和报告为有效地管理某一配置所需的信息。包括列出经批准的配置标识表、列出对配置提出更动的状态表和经批准的更动的实现状态。

2.100 监护 confinement

a. 在被核准访问期间，防止对数据做未经核准的改变、使用、破坏和抛弃。参见 2.247 条。

b. 对程序和进程施加的限制，目的是使它们不能访问或影响未经核准的数据、程序或进程。

2.101 连接 connection

a. 程序的某一部分对程序另一部分的标识符(即，在另外地方发现的标识)的引用。参见2.249条。

b. 为了传递信息而在功能部件之间建立的关系。

2.102 合同 contract

通过法律约束当事双方的一个协议，或是在一个机构内部为了提供服务的一个内部协议，该协

议提供的服务适用于一个系统或系统一部分的供应、开发、生产、操作或维护。

2.103 合同所要求的审计 contractually required audit

合同所要求的审核过程。一般由需方或由独立的机构主持进行。此过程对产品或服务提供一个独立的评价，以决定产品或服务是否符合它们的需求。

2.104 控制数据 control data

选择一程序中的操作方式或子方式，给顺序流指向，或者直接影响软件操作的数据。

2.105 控制语句 control statement

影响操作执行顺序的程序设计语言的语句。

2.106 控制结构 control structure

通过计算机程序决定控制流的构造。参见 2.91 条。

2.107 转换 conversion

对现有软件进行修改，使之在不同环境工作时能具有等同的功能，例如，把一个程序从 FORTRAN 变换成 Ada。把在一台计算机上运行的程序变换成能在另一台计算机上运行的程序。

2.108 协同例行程序 co-routines

彼此能调用，但不存在上下级关系的两个或两个以上的模块。

2.109 改正性维护 corrective maintenance

专门为克服现有故障而进行的维护。参见 2.449 条。

2.110 正确性 correctness

a. 软件无设计缺陷和编码缺陷的程度，即无故障。

b. 软件符合规定的需求的程度。

c. 软件满足用户期望的程度。

2.111 正确性证明 correctness proof

参见 2.374 条。

2.112 耦合度 coupling

计算机程序中模块之间相互依赖的量度。与 2.67 条相对照。

2.113 临界的，关键的 critical

系指：

a. 由于设计不当，一个系统或一个软件的某些环节或部分在运行时超出了临界范围，或存在着潜在的、未检测出的错误，会导致死机、人员伤害、任务失败、数据丢失、财经上的损失或灾难性的设备损坏等严重后果。或指：

b. 要使用的软件开发技术的成熟程度和有关的风险。

2.114 关键部分优先 critical piece first

软件开发的一种途径。它首先把注意力集中在软件系统中最关键部分的实现。关键部分可以根据所提供的服务、风险程度、困难程度或其它一些准则来确定。

2.115 关键段，临界段 critical section

将要被执行的一段代码。其执行与另一关键段的代码的执行是互斥的。如果一些代码段竞相使用一计算机资源和数据项时，就要求这些段互斥地执行。

2.116 危急程度 criticality

根据软件错误或故障对系统的开发和运行的影响程度所做的估价进而对这些软件错误或故障进行的分类(通常用来判定是否要对某一故障进行校正，以及何时予以校正)。

2.117 交叉汇编程序 cross assembler

在一台计算机上为另一台不同的计算机产生目标代码的汇编程序。

2.118 交叉编译程序 cross compiler

在一台计算机上为另一台不同计算机产生汇编代码或目标代码的编译程序。

2.119 数据 data

事实、概念或指令的形式化的表现形式，它适于由人或自动装置进行通信、解释或处理。参见2.79条、2.104条、2.179条、2.395条、2.445条。

2.120 数据抽象 data abstraction

通过选择特定的数据类型及其相关的功能特性的办法，仅仅保持或抽取数据的本质特性所得的结果，从而使其与细节部分的表现方式分开或把它们隐藏起来。参见2.235条。

2.121 数据库，数据基 data base

a. 一数据集，或一数据集的部分或全体，它至少包括足够为一给定目的或给定数据处理系统使用的一个文件。

b. 对一系统来说是基本的数据集合。

2.122 数据字典 data dictionary

a. 软件系统中使用的所有数据项的名字及与这些数据项有关的特性(例如，数据项长度、表示等)的集合。

b. 分层数据流图中涉及的数据流、数据元素、文件、数据基和进程之定义的集合。

2.123 数据流图 data flow chart

系统的一种图形表示，其中表示出数据源、数据汇、存储和以结点形式对数据执行的处理，以及在结点间作为连接部分的逻辑数据流。与2.124条、2.125条同义。

2.124 数据流图 data flow diagram

参见2.123条。

2.125 数据流图 data flow graph

参见2.123条。

2.126 数据结构 data structure

数据项之间的次序安排和可访问性的一种形式表示，其中不涉及其实际存储排列方法。

2.127 数据类型 data type

一类数据。用属于该类的元素和可对之施行的操作来表征，例如，整型、实型、逻辑型。

2.128 排错，调试 debugging

查找、分析和纠正错误的过程。

2.129 排错模型 debugging model

参见2.180条。

2.130 判定表 decision table

a. 在叙述一问题中要考虑的所有可能发生的情况及对每一组可能发生的情况将要采取的行动的一张表。

b. 对一组情况及其相应动作以矩阵形式或列表形式所做的表示。

2.131 缺陷 defect

参见2.198条。

2.132 定义阶段 definition phase

参见2.406条。

2.133 交付 delivery

a. 软件研制周期中的一个阶段。在此阶段上将产品提交给计划中的用户供其使用。

b. 软件研制周期中的一个阶段。在此阶段上产品由其预定的用户接受。

2.134 设计 design

a. 为使一软件系统满足规定的需求而确定软件体系结构、部件、模块、接口、测试途径和数据

的过程。

b. 设计过程的结果。

2.135 设计分析 design analysis

a. 对一设计进行估计以确定其相对于预定需求的正确性、符合设计标准的程度、系统效率和是否符合其它一些准则。

b. 对其它替代性设计途径的估计。

2.136 设计分析器 design analyzer

一种自动设计工具。它接收有关程序的设计方面的信息，并产生以下方面的输出，如模块层次图、控制和数据结构的图形表示，以及被访问的数据块的一览表等。

2.137 设计审查 design inspection

参见 2.237 条。

2.138 设计语言 design language

一种具有专门构造，有时还可验证的语言。用以开发、分析设计并为其书写文件。

2.139 设计方法学 design methodology

进行设计的系统途径。由专门选择的工具、技术、准则的有序应用所构成。

2.140 设计阶段 design phase

软件生存周期中的一段时间。在这段时间内，进行体系结构、软件组成部分、接口和数据的设计，为设计编制文件，并对其进行验证，以满足预定需求。

2.141 设计需求 design requirement

影响或限制软件系统或软件系统组成部分的设计的需求：例如，功能需求、物理需求、性能需求，软件开发标准、软件质量保证标准。参见 2.407 条。

2.142 设计评审 design review

a. 在正式会议上，把系统的初步的或详细的设计提交给用户、客户或有关人士供其评审或批准。

b. 对现有的或提出的设计所做的正式评估和审查，其目的是找出可能会影响产品，过程或服务工作的适用性和环境方面的设计缺陷并采取补救措施，以及(或者)找出在性能、安全性和经济方面的可能的改进。

2.143 设计规格说明 design specification

一种描述设计要求的正式文档，按照这种文档对系统或系统组成部分(如，软件配置项)进行设计。典型内容包括系统或系统组成部分算法、控制逻辑、数据结构设定与使用(set-use)信息、输入输出格式和接口描述。参见 2.407 条。

2.144 设计验证 design verification

参见 2.539 条。

2.145 设计走查 design walk-through

参见 2.545 条。

2.146 桌面检查 desk checking

对程序执行情况进行人工模拟，用逐步检查源代码中有无逻辑或语法错误的办法来检测故障。参见 2.468 条。

2.147 详细设计 detailed design

a. 推敲并扩充初步设计，以获得关于处理逻辑、数据结构和数据定义的更加详尽的描述，直到设计完善到足以能实现的地步。

b. 详细设计过程的结果。

2.148 开发者 developer

在软件生存周期中执行开发活动(包括需求分析、设计直至验收)的一个机构。

2.149 开发周期 development cycle

参见 2.438 条。

2.150 开发生存周期 development life cycle

参见 2.438 条。

2.151 开发方法学 development methodology

编制软件的系统方法。它确定开发的各个阶段,规定每一阶段的活动、产品、验证步骤和完成准则。

2.152 开发规格说明 development specification

与 2.407 条同义。

2.153 诊断 diagnostic

a. 计算机程序产生的信息。它用来指示另一系统组成部分中可能的故障。例如,由编译程序标识的语法错误。

b. 涉及故障或失效的探测和隔离。

2.154 有向图 digraph

参见 2.155 条。

2.155 定向图 directed graph

一种图,其中的边均是单方向的。

2.156 文档,文件 document

a. 一种数据媒体和其上所记录的数据。它具有永久性并可以由人或机器阅读。通常仅用于描述人工可读的内容。例如,技术文件、设计文件、版本说明文件。

b. 编制文件。

2.157 文档、文档编制,文档管理 documentation

a. 关于一给定主题的文件集合。参见 2.536 条、2.443 条、2.493 条。

b. 文档管理可能包括下述活动:对文档的识别、获取、处理、存储和发放。

c. 产生一个文档的过程。

d. 为了对活动、需求、过程或结果进行描述、定义、规定、报告或认证的任何书面或图示的信息。

2.158 文档级 documentation level

参见 2.263 条。

2.159 驱动程序 driver

一个程序。它借助模拟较高一级的系统组成部分的办法来履行系统或系统组成部分的作用。参见 2.511 条。

2.160 双份编码 dual coding

一种开发技术。由不同的程序员或不同的程序设计小组,根据同一份规格说明书开发出功能上完全相同的程序的两个版本。所获得的源代码可以采用同一种语言,也可以采用不同的语言。双份编码的目的在于提供错误检测,提高可靠性,提供附加的文件说明,或使系统的程序设计错误或编译程序错误影响最终结果的概率降低。

2.161 虚参数 dummy parameter

参见 2.211 条。

2.162 卸出,转储 dump

a. 已被转储的数据。

b. 为了某一专门目的。如允许存储器另作它用,或作为预防故障和错误的措施;或为了进行

与排除错误有关的工作，将一存储器(通常是内部存储器)的全部或部分内容写到外部媒体上。

2.163 动态分配 dynamic allocation

把可编址的存储器和其它资源分配给正在执行的程序。

2.164 动态分析 dynamic analysis

根据程序的执行情况对程序进行估计的过程。与 2.468 条相对照。

2.165 动态分析器 dynamic analyzer

借助对程序执行情况的监督，帮助对计算机程序进行估计的软件工具。例如探测工具、软件监督器和跟踪器。与 2.469 相对照。

2.166 动态结合，动态联编 dynamic binding

在程序执行期间进行的结合。与 2.470 相对照。

2.167 动态重组 dynamic restructuring

a. 一系统正在运行时，改变软件组成部分或结构的过程。

b. 在程序执行期间重新组合数据库或数据结构的过程。

2.168 编辑程序 editor

可以对计算机中所存储的数据进行有选择性的修正的计算机程序。

2.169 效率 efficiency

软件以最小的计算资源消耗实现其预定功能的程度。

2.170 无效程序设计 egoless programming

在对程序开发采用小组负责制的概念的基础上进行软件开发的一种方式。其目的是防止程序员与其产生的输出的关系过于密切，以免使客观估计受到损害。

2.171 嵌入式计算机系统 embedded computer system

归结在一个其主要目的不是进行计算的较大系统中成为其完整不可分开的部分的计算机系统。例如，在武器、航空、指挥控制、或运输系统中的计算系统。

2.172 嵌入式软件 embedded software

嵌入式计算机系统用的软件。

2.173 仿真 emulation

用一个计算机系统，主要是通过硬件，模仿另一个计算机系统的全部或部分功能，使进行模仿的系统接受的数据、执行的程序和实现的结果均与被模仿的系统所接受的数据，执行的程序和实现的结果相同。

2.174 仿真器 emulator

执行仿真的硬件、软件或固件。

2.175 封装 encapsulation

将系统功能隔离在一个模块中，并为该模块提供精确的规格说明的技术。参见 2.235 条。

2.176 错误，出错，误差 error

a. 计算、观察、测量的值或条件与实际的、规定的或理论上的值或条件不符合。

b. 导致产生含有缺陷的软件的人为行动。例如，遗漏或误解软件说明书中的用户需求，不正确的翻译或遗漏设计规格说明书中的需求。参见 2.192 条、2.198 条。

2.177 出错分析 error analysis

a. 对观察到的软件故障进行调查的过程，调查的目的是跟踪那个故障以找出故障源。

b. 对观察到的软件故障进行调查以找出以下一些信息，例如故障原因。该故障是在开发过程中哪一个阶段发生的，预防或较早地探测出软件故障的方法。

c. 调查软件错误、失效和故障以确定定量速率和趋势的过程。

2.178 出错类别 error category

错误、故障或失效可能归并到其中的一组类别之一，当错误、故障或失效发生或发现后，可根据其原因、危急程度、效果、故障所属的生存周期阶段或其它特性而确定其类别。

2.179 出错数据 error data

出错数据通常(但不是精确地)用于：描述软件的问题、故障、失效及其更动，它们的特性，以及遇到或改正这些问题的条件。

2.180 出错模型 error model

用于描述或估计一软件系统存在的故障数目、可靠性、需要的测试时间或类似特性。参见 2.181 条。

2.181 出错预测 error prediction

对有关软件系统中软件问题、故障或失效的预期目的或性质所作的定量陈述。参见 2.180 条。

2.182 出错预测模型 error prediction model

参见 2.180 条。

2.183 出错恢复 error recovery

参见 2.197 条。

2.184 错误的撒播 error seeding

参见 2.201 条。

2.185 评价 evaluation

决定某产品、项目、活动或服务是否符合它的规定的准则的过程。

2.186 异常 exception

引起正常程序执行挂起的事件。

2.187 执行 execution

由计算机运行计算机程序中一条或多条指令的过程。

2.188 执行时间 execution time

a. 执行一个程序所用的实际时间或中央处理机所用的时间。

b. 程序处于执行过程中的一段时间间隔。参见 2.418 条。

2.189 执行时间理论 execution time theory

采用累计执行时间作为估计软件可靠性基础的一种理论。

2.190 执行程序 executive program

参见 2.485 条。

2.191 退出，终止，出口 exit

a. 计算机程序、例程或子例程中的一条指令。在执行它之后，该计算机程序、例程或子例程就不再具有控制权。

b. 例程不再具有控制权的转折点。

2.192 失效 failure

a. 功能部件执行其功能的能力的丧失。

b. 系统或系统部件丧失了在规定的限度内执行所要求功能的能力。当遇到故障情况时系统就可能失效。

c. 程序操作背离了程序需求。

2.193 失效类别 failure category

参见 2.178 条。

2.194 失效数据 failure data

参见 2.179 条。

2.195 失效率 failure rate

a. 失效数与给定测量单位的比率；例如，每单位时间的失效次数、若干次事务处理中的失效次数，若干次计算机运行中的失效次数。

b. 在可靠性模拟中，给定类别或具有一定严重程度的失效数与给定时间间隔之比率；例如，每秒执行时间的失效次数，每月失效次数。与 2.196 条同义。

2.196 失效比 failure ratio

参见 2.195 条。

2.197 失效恢复 failure recovery

系统失效后又回到可靠的运行状态。

2.198 故障，缺陷 fault

a. 功能部件不能执行所要求的功能。

b. 在软件中表示 2.176b 关于错误的解释。如果遇到，它可能引起失效。与 2.54 条同义。

2.199 故障类别 fault category

参见 2.178 条。

2.200 故障插入 fault insertion

参见 2.201 条。

2.201 故障撒播 fault seeding

为了估计程序中的固有故障数，有意地在计算机程序已有的故障上添加已知数目的故障的过程。与 2.55 条同义。

2.202 容错 fault tolerance

在出现有限数目的硬件或软件故障的情况下，系统仍可连续正确运行的内在能力。

2.203 功能性配置审计 FCA—functional configuration audit

验证一个配置项的实际工作性能是否符合它的需求规格说明的一项审查，以便为软件的设计和编码建立一个基线。

2.204 文件，文卷 file

作为一个单位来看待的一组相关的记录。参见 2.276 条。

2.205 有限状态机 finite state machine

由有限个状态及这些状态之间变迁构成的计算模型。

2.206 固件 firmware

a. 装于某类存储器中的在处理期间不能由计算机动态地修改的计算机程序和数据。参见 2.292 条、2.293 条。

b. 含有在用户环境下不能修改、不会丢失的计算机程序和数据的器件。包含在固件中的计算机程序和数据归类为软件；含有计算机程序和数据的电路归类为硬件。

c. 存储在只读存储器中的程序指令。

d. 由硬件装置和计算机程序集成形成一个功能实体的组件，在正常运行期间该实体配置不能改变。计算机程序存储在集成电路形式的硬件装置中，逻辑配置是固定的，以满足具体应用或工作需求。

2.207 标志 flag

a. 通知出现了某种错误、状态或其它条件的指示符。

b. 用于表示各种指示符中的任何一种。例如，字标。

c. 通知出现了一定条件。如字的结束的字符。

d. 指示程序中的错误、状态，或其它规定条件。

2.208 控制流 flow of control

在执行某一算法时所完成的操作序列。

2.209 流程图 flowchart

问题定义、分析或求解的一种图形表示。在这种表示中，用符号表示操作、、数据、流程和设备。与2.48相对照。

2.210 形式语言 formal language

一种语言，其规则在使用前就已明显地确立。与2.28条同义。例如FORTRAN和Ada等程序设计语言，以及诸如谓词演算之类的数学或逻辑语言。与2.307条对照。

2.211 形参 formal parameter

子程序中使用的变量。用来表示调用例行程序时要传送给子程序的数据或程序元素。与2.161条同义。与2.14条相对照。

2.212 正式规格说明、形式规格说明 formal specification

a. 根据已建立的标准书写并获准的规格说明。

b. 在正确性证明中，对一系统或系统组成部分外部可见行为用形式语言进行的描述。

2.213 正式测试 formal testing

根据已批准的测试计划进行测试活动并报告结果。

2.214 功能，函数 function

a. 一实体或其特征动作能实现特定目的能力。

b. 由自变量的值可得到确定结果的特定子程序。函数通常用函数名来调用，计算函数值的变量以参数的形式提供。

2.215 功能分解 functional decomposition

设计系统的一种方法。这种方法把系统分成若干部分，使其直接与系统功能和子功能对应。参见2.222条。

2.216 功能设计 functional design

制定数据处理系统各部分的功能及相互之间接口的规格说明。参见2.343条。

2.217 功能需求 functional requirement

规定系统或系统组成部分必须能够执行的功能的需求。

2.218 功能规格说明 functional specification

确定系统或系统组成部分必须执行的功能的规格说明。参见2.336条。

2.219 功能部件 functional unit

能实现某一特定目标的硬件、软件或两者兼而有之的实体。

2.220 硬件 hardware

数据处理中使用的物理设备，相对计算机程序、过程、规则和相关的文件而言。与2.433条相对照。

2.221 硬件配置项 HCI—hardware configuration item

整个系统体系结构中的硬件的一个配置项。

2.222 层次结构分解 hierarchical decomposition

设计系统的一种方法。这种方法通过一系列自顶向下逐步求精的办法把系统分成若干部分。参见2.215条、2.298条、2.472条。

2.223 层次结构 hierarchy

一种结构。其组成部分根据一组特定的规则排列成若干层次。

2.224 高级语言 high level language

与2.225同义。

2.225 高级语言 higher order language

一种程序设计语言。它通常包括如下一些特点：嵌套表达式、用户定义的数据类型和通常在低级

语言中没有的参数传递；它不反映任何一台计算机或一类计算机的结构，从而可以用它书写与机器无关的源程序。一个单一的高级语言语句可以表示多个机器操作。与 2.279 条、2.31 条相对照。

2.226 宿主机 host machine

a. 程序或文件所装入的计算机。

b. 用以开发供另一台计算机用的软件的计算机。与 2.502 条相对照。

c. 用以模仿另一台计算机的计算机。与 2.502 条相对照。

d. 在计算机网络中，为该网络的用户提供处理能力的计算机。

2.227 标识符 identifier

a. 用以命名、指示或定位的符号。标识符可以和数据结构、数据项或程序位置相关联。

b. 用以标识一数据项或给一数据项命名，也可能指出该数据某些特性的一个或一组字符。

2.228 不完全的隐错排除 imperfect debugging

在可靠性模拟中，纠正或清除已经发现故障的意图并非总是成功的一种假定。

2.229 实现 implementation

a. 以较为具体的项来体现一抽象的概念；特别是用硬件、软件或两者一起来体现一抽象的概念。

b. 程序的一种机器可执行形式，或者能被自动地翻译成机器可执行的形式的某种形式的程序。

c. 把设计翻译成代码，然后对此代码排除隐错的过程。

2.230 实现阶段 implementation phase

软件生存周期中的一段时间。在这段时间内，根据设计文件制造软件产品并排除其中的隐错。参见 2.238 条、2.513 条。

2.231 实现需求 implementation requirement

对软件设计的实现产生影响或限制的任何需求。例如，设计描述、软件开发标准、程序设计语言需求、软件质量保证标准等。

2.232 独立验证和确认 independent verification and validation

a. 由某机构对软件产品进行的验证和确认，该机构在技术上和行政管理上都与负责开发该软件产品的机构是分开的。

b. 由个人或小组对软件产品进行的验证和确认。这些个人或小组不是软件产品的原始设计人，但可以和后者同属一个机构。独立的程度取决于该软件的重要性。

2.233 原有故障 indigenous fault

计算机程序中存在的一种故障。这种故障不是作为故障撒播过程的一部分而插入的。

2.234 归纳断言法 inductive assertion method

一种正确性证明技术。采用这种技术时要写出描述程序输入、输出和中间条件的断言，推导出当输入条件满足时，使输出条件得到满足的一组定理，并且这些定理被证明是成立的。

2.235 信息隐蔽 information hiding

将模块中的软件设计决策封装起来的技术，使模块内部工作情况尽可能少在模块的接口处暴露。这样，系统中每个模块对其它模块而言是个“黑盒子”。信息隐蔽的原则禁止使用在模块接口中没有说明的信息。参见 2.175 条。

2.236 输入断言 input assertion

逻辑表达式。它规定了程序的输入必须满足的一个或多个条件。

2.237 审查 inspection

a. 一种正式的评定技术。由除作者之外的某人或某一小组仔细检查软件需求、设计或代码，以

找出故障、违反开发标准之处和其它一些问题。与 2.545 条相对照。参见 2.63 条。

b. 质量管理的一个阶段。在此阶段借助检查、观察或测量来确定材料、必须品、零部件、附属品、系统、过程或结构是否符合预定的质量要求。

2.238 安装检验阶段 installation and check-out phase

软件生存周期中的一段时间。在此时间内，软件产品被结合到工作环境中，并在该环境中加以测试，以保证它能按照要求进行工作。

2.239 指令 instruction

a. 使计算机执行一个特定操作或执行一组特定操作的程序语句。

b. 在程序设计语言中，规定某种操作，且如果有操作数则对操作数进行标识的一个有含义的表述。

2.240 指令集合(指令系统) instruction set

计算机的指令集合，程序设计语言指令集合，或程序设计系统中程序设计语言的指令集合。

2.241 指令集合结构 instruction set architecture

用指令集合表征的抽象机。

2.242 指令跟踪 instruction trace

参见 2.530 条。

2.243 探测 instrumentation

参见 2.358 条。

2.244 探测工具 instrumentation tool

一种软件工具。它在被测程序中的适当位置上产生并插入起计数器或其它探头作用的语句，以提供有关程序执行情况的统计数字，如程序中的代码被执行到的覆盖程度。

2.245 集成 integration

把软件、硬件元素或两者合成为一个完整的系统的过程。

2.246 组装测试 integration testing

有序进行的一种测试。这种测试中，把软件元素、硬件元素或两者一并进行测试，直到整个系统成为一体。参见 2.497 条。

2.247 完整性 integrity

在计算机系统中，对软件或数据所受到的未经获准的存取或修改可加以控制的程度。参见2.420条。

2.248 交互系统 interactive system

指这样一个系统。在这种系统中，每一个用户的输入均能得到该系统的响应。

2.249 接口，界面 interface

a. 一个共有的边界。接口可能是连接两个设备的硬件组成部分，也可能是由两个或多个计算机程序所访问的一部分存储器或寄存器。

b. 与另一系统组成部分的交互作用或通信。

2.250 接口需求 interface requirement

规定一个系统或系统组成部分必须与之接口的硬件、软件或数据库元素的需求。或由这样一个接口而引起的对格式、时间关系或其它因素提出的条件。

2.251 接口规格说明 interface specification

规定系统或系统组成部分的接口需求的规格说明。

2.252 接口测试 interface testing

为确保程序或系统组成部分彼此正确地传递信息或控制而进行的测试。

2.253 互操作能力，互操作性 interoperability

a. 两个或多个系统交换信息并相互使用已交换的信息的能力。与 2.71 条相比较。

b. 两个或两个以上系统可互相操作的能力。

2.254 解释 interpret

逐条翻译并立即执行计算机程序的每一源语言语句。与 2.29 条、2.72 条相对照。

2.255 解释程序，解释器 interpreter

a. 用来解释计算机程序的软件、硬件或固件。与 2.30 条、2.73 条相对照。

b. 用于进行解释的计算机程序。

2.256 中断 interrupt

把一进程(如计算机程序)的执行暂停。这一暂停是由该进程之外的事件引起的，中断处理后，被暂停的进程应能恢复。

2.257 迭代 iteration

a. 重复执行给定的程序设计语言语句序列，直到满足给定条件或当给定条件为真时为止的过程。

b. 对循环的一次执行。

2.258 核心，内核 kernel

a. 操作系统的基础，操作系统的最小的不可缺少的部分。

b. 基本功能的封装部分。

c. 在计算机选择研究中用以评价计算机性能的模型。

2.259 关键字 key

数据集合中的一个或多个字符。它含有有关该集合的信息，包括其标识。

2.260 标号 label

a. 数据集合内或附加于数据集合上的一个或多个字符。其中含有有关该集合的信息，包括其标识。

b. 在计算机程序设计中，指令的标识符。

c. 一个带或盘文件的标识记录。

2.261 语言处理程序 language processor

a. 一种计算机程序。它执行这样一些功能，诸如处理指定程序设计语言所需的翻译、解释功能和其它任务。例如 FORTRAN 处理程序、COBOL 处理程序。

b. 一种软件工具。它完成这样一些功能，诸如处理指定的语言(如需求规格说明语言、设计语言或程序设计语言)所需的翻译、解释或其它任务。

2.262 级，层 level

a. 一个项在某一层次排列中下属的级数。

b. 层次结构中的等级。若一项目没有从属项则属最低级，若没有比它高的项则为最高级。

2.263 文档等级 level of documentation

指明文档的范围、内容、格式以及质量。文档等级可根据项目成本、预期用途、作用范围、及其它因素进行选择。

2.264 资料管理员 librarian

参见 2.446 条。

2.265 库 library

参见 2.447 条、2.494 条。

2.266 生存周期 life cycle

参见 2.448 条。

2.267 生存周期模型 life-cycle model

一个框架，它含有从需求定义到使用终止，跨越整个生存期的系统开发、操作和维护中所需实施的过程、活动和任务。

2.268 连接编辑程序 linkage editor

一个计算机程序。它利用一个或多个独立地编译而得到的目标模块或装入模块而建立一个装入模块。为此要在目标模块当中解决交叉引用。也可能需要把一些元素重新定位。注意并不是所有的目标模块在执行之前都需要连接。

2.269 连接表 linked list

参见 2.60 条。

2.270 列表，清单，表 list

a. 数据有序集。

b. 将满足规定准则的数据项进行打印或显示。

c. 参见 2.60 条。

2.271 列表处理 list processing

一种用表的形式来处理数据的方法。通常使用链接表，这样就能改变项的逻辑顺序而无需改变它们的物理位置。

2.272 列表 listing

a. 以人们易读的列表形式给出的计算机输出。

b. 人们易读的、正文形式的计算机输出。

2.273 装入映象表 load map

计算机生成的表，它标识驻留在内存中的计算机程序或驻留在内存中的数据的全部或指定部分的位置或大小。

2.274 装入模块 load module

适合于装入到主存中去等待执行的程序单位。它通常是连接编辑程序的输出。

2.275 装入程序 loader

a. 一种例行程序。它在目标程序执行之前把目标程序读入到主存中去。

b. 一种例行程序。通常是计算机程序。它把数据读入到主存中去。

2.276 逻辑文件 logical file

与物理环境无关的文件。同一逻辑文件的各部分可以放在不同的物理文件中；几个逻辑文件或几个逻辑文件的各部分可以放在一个物理文件中。

2.277 逻辑记录 logical record

与物理环境无关的记录。同一逻辑记录的各部分可以放在不同的物理记录中；几个逻辑记录或几个逻辑记录的各部分可以放在一个物理记录中。

2.278 循环 loop

当某个条件成立时可以反复执行一组指令的程序结构。参见 2.257 条。

2.279 机器语言 machine language

指令和数据的表示。此表示能直接由计算机执行。与 2.31 条、2.225 条相对照。

2.280 宏 macro

a. 一个预先定义好的指令序列。在汇编或编译期间要把该指令序列插入到程序中每一处出现相应宏指令的地方。

b. 与 2.281 条同义。

2.281 宏指令 macroinstruction

源语言中的一条指令。它将用同一源语言书写的预先定义的指令序列所代替。宏指令也可以为将要代替它的指令中的参数指定其值。

2.282 宏处理程序 macroprocessor

某些汇编程序和编辑程序的部分。它允许程序员定义和使用宏。

2.283 可维护性 maintainability

a. 对软件进行维护的容易程度。

b. 按照预定的需要对某一功能部件进行维护的容易程度。

c. 按照规定的使用条件，在给定时间间隔内一个项保持在某一指定状态或恢复到某一指定状态的能力。在此状态下，若在规定的条件下实现维护并使用所指定的过程和资源时，它能实现要求的功能。

2.284 维护者 maintainer

执行维护活动的一个机构。

2.285 维护 maintenance

参见 2.449 条。

2.286 维护阶段 maintenance phase

参见 2.317 条。

2.287 维护计划 maintenance plan

维护软件产品时使用的说明管理方法和技术途径的文档。典型的维护计划内容包括：工具、资源、设施及日程。

2.288 映象程序 map program

编译程序或汇编程序中具有生成装入映象性能的部分。

2.289 主库 master library

存放软件和文件的正式公布版本的软件库。与 2.351 条相对照。

2.290 元编译程序 metacompiler

参见 2.75 条。

2.291 元语言 metalanguage

用来说明一个语言或多个语言的基本语言。

2.292 微码 microcode

a. 微程序的符号表示。

b. 微程序在其存储媒体中的内部表示。参见 2.206 条。

2.293 微程序 microprogram

计算机操作相对应的微指令序列。它被保存在专用存储器中，并且是由计算机指令寄存器中的计算机指令来启动其执行，微程序常常用于代替硬接线逻辑。参见 2.206 条。

2.294 里程碑 milestone

项目有关人员或管理人员负责的在预定时间将发生的事件，用来标志工作进度。例如，正式的复审、规格说明的颁布、产品的交付。

2.295 助记符号 mnemonic symbol

为便于人们记忆而选用的一种符号。例："multiply"的缩写是"mul"。

2.296 模型 model

现实世界中进程、设备或概念的一种表示。参见 2.23 条、2.42 条、2.129 条、2.180 条、2.398 条、2.430 条、2.471 条。

2.297 修改 modification

a. 对软件进行的更改。

b. 更改软件的过程。

2.298 模块分解 modular decomposition

借助于把系统分成若干模块来设计系统的方法。参见 2.222 条。

2.299 模块化程序设计 modular programming

把系统或程序作为一组模块集合来开发的一种技术。

2.300 模块性 modularity

软件由若干离散部分组成的离散程度，即软件模块化的程度(表明改变一个组成部分时对另外的组成部分有多大的影响)。

2.301 模块 module

a. 是离散的程序单位。且对于编译、对于和其它单位相结合，对于装入来说是可识别的，例如，汇编程序、编译程序、连接编辑程序或执行的例行程序的输入或输出。

b. 程序中一个能逻辑地分开的部分。

2.302 模块强度 module strength

参见 2.67 条。

2.303 多级安全性 multilevel security

一种操作方式。当至少有某些用户对系统中包括的全部数据既不清楚也不需要知道时，它允许处于各种安全级上的数据并行地在计算机系统中存储和处理。

2.304 多道程序设计 multiprogramming

a. 一种操作方式。它可以使单处理机交替地执行两个或多个计算机程序。

b. 由一台计算机对两个或多个计算机程序并行执行。

c. 两个或多个功能的并行执行，就好象每个功能单独操作一样。

2.305 变异 mutation

参见 2.360 条。

2.306 N-进制 N-ary

a. 由 n 个不同可能的值或状态的挑选、选取或条件所表征。

b. 具有基数 n 的固定基数数制系统。

2.307 自然语言 natural language

一种语言。其规则是根据当前的习惯用法而不是显式的方法规定的。例如英语、汉语、法语以及斯瓦希利语。与 2.210 条相对照。

2.308 嵌套 nest

a. 把某一类的一个结构或多个结构合并到同一类结构中去。例如，把一个循环(被嵌套的循环)嵌套到另一个循环(嵌套的循环)中去；把一个子例行程序(被嵌套的子例行程序)嵌套到另一个子例行程序(嵌套的子例行程序)中去。

b. 把子例行程序或数据放在处于另一不同层次级别上的另一个子例行程序或数据中，使得该子例行程序可作为递归子例行程序被执行，或该数据能被递归地存取。

2.309 网络 network

a. 一组互连或相关的结点。

b. 涉及约束性或面向问题的修饰词、资料、文件以及人力资源的组合，通过设计把这些组合起来以实现某些目标。例如：社会科学网络、科学信息网络。

2.310 结点，节点 node

a. 网络或图中任何分支的端点，或属于两个或多个分支的公共结点。

b. 在树形结构中，下属数据项由之发源的一点。

c. 在网络中，一个或多个功能部件在此互连传输线的点。

d. 借助于图上的点表示状态或事件。

2.311 不交付项 non-deliverable item

不需要按合同交付的、但在软件的开发中可能用到的硬件和软件。

2.312 目标程序 object program

已完成编译或汇编已准备好装入到计算机中的程序。与 2.460 条相对照。

2.313 现货产品 of-the-shelf product

由供方、需方或第三方提供的、已经开发出来的、可得到、可使用的、现成的或需要加以修改的产品。

2.314 开放系统 open system

一种按标准建立起来的计算机系统。开放系统的主要特征是：

兼容性(compatibility)参见 2.71 条；

可移植性(portability)参见 2.340 条；

互操作性(interoperability)参见 2.253 条；

可扩展性(scaliability)参见 2.419 条。

开放系统可以使用户摆脱特定厂商的控制，并有利于提高软件厂商的效益。

2.315 操作对象，操作数 operand

a. 对之施行操作的一个实体。

b. 施行操作的目标。操作对象通常由指令的地址部分标识。参见 2.320 条。

2.316 操作系统 operating system

控制程序执行的软件。操作系统可以提供以下的服务，例如，资源分配调度、输入/输出控制和数据管理。虽然操作系统目前主要是一种软件，但部分或全部地用硬件来实现是可能的。操作系统是在单一点上提供支持，而不是强制每个程序去关心如何控制硬件。参见 2.496 条。

2.317 运行和维护阶段 operation and maintenance phase

软件生存周期中的一个阶段，在此阶段软件产品在规定的运行环境中进行使用、监视，需要时对软件产品进行修改以改正问题或对变化了的需求作出响应，以获得满意的功能和性能。

2.318 运行可靠性 operational reliability

在实际使用环境中，系统或软件子系统的可靠性。运行可靠性可能与规定环境或测试环境中的可靠性有很大的不同。

2.319 运行测试 operational testing

由最终用户在软件的正常操作环境里对软件执行的测试。

2.320 操作符，操作员 operator

a. 在符号处理中，表示操作中要实现的动作的符号。如＋、－、＊、/。

b. 在描述一进程时，指明对操作对象执行的动作。

c. 操作机器的人或运行系统的一个机构。

参见 2.315 条。

2.321 组织过程 organizational process

为构成自始至终的一个完整过程，由机构指定并以项目为例说明的一套软件工程和管理过程。

2.322 输出断言 output assertion

一个逻辑表达式。它说明为了保证程序是正确的，程序输出必须满足的一个或多个条件。

2.323 覆盖 overlay

a. 在计算机程序中，不永久存储在内存里的一个段。

b. 在程序的不同阶段反复使用内存的同一区域的技术。

c. 在计算机程序的执行过程中，把计算机程序的存储段装到当前不再需要的程序部分所占用的存储区域中。

2.324 参数 parameter

a. 是一种变量，针对每一指定用途，可赋予它一个固定值，并可用这个变量代表该用途。

b. 用来在程序之间传递值的变量。参见 2.14 条、2.210 条。

2.325 语法分析 parse

确定人工语言或自然语言单位的句法结构的过程。方法是把上述单位分解为多个基本子单位并建立子单位之间的关系，例如，块、语句，表达式可分解为语句、表达式及操作符和操作对象。

2.326 部分正确性 partial correctness

在正确性证明中，指出程序的输出断言是它的输入断言和处理步骤的合乎逻辑的结果。与2.529条相对照。

2.327 修补 patch

a. 对目标程序的修改。办法是用修改的机器代码代替已有的部分机器代码。

b. 修改目标程序而无需重新编译源程序。

2.328 路径分析 path analysis

对程序进行的一种分析。以标识通过该程序的所有可能的路径，检测不完全的路径，或发现不处在任何路径上的程序部分。

2.329 路径条件 path condition

为了要执行特定的程序路径所必须满足的一组条件。

2.330 路径表达式 path expression

一个逻辑表达式。它说明为了执行特定程序路径，所必须满足的输入条件。

2.331 物理配置审计 PCA—physical configuration audit

对照设计规格说明检验已建立的某个配置项，其目的是为软件的设计和编码建立一个基线。

2.332 完善性维护 perfective maintenance

为改善性能、可维护性或其它软件属性而进行的维护。参见 2.16 条、2.109 条。

2.333 性能 performance

a. 计算机系统或子系统实现其功能的能力。

b. 对计算机系统或子系统执行其功能的能力的度量。例如，响应时间、吞吐能力、事务处理数。参见 2.335 条。

2.334 性能评价 performance evaluation

对系统或系统部件的技术评价，以确定运行目标达到了何种有效程度。

2.335 性能需求 performance requirement

对系统或系统部件必须具有的性能(例如，速度、精度、频率)作出规定的需求。

2.336 性能规格说明 performance specification

a. 规定系统或系统部件性能需求的规格说明。

b. 与 2.407 条同义。

c. 参见 2.218 条。

2.337 petri 网 petri net

信息流的一个抽象的、形式的模型。指出一系统的静态和动态性质。petri 网通常表示成图。图中有两类用弧彼此相连的结点(称为地点和变换)和指示其动态性能的标记(称为记号)。参见 2.467 条。

2.338 物理需求 physical requirement

规定系统或系统部件必须具有的物理特征的一种需求，例如，材料、形状、大小、重量。

2.339 指针 pointer

a. 指明数据项位置的标识符。

b. 一种数据项。其值是另一数据项的位置。

2.340 可移植性 portability

软件不加改动地从一种运行环境转移到另一种运行环境下运行的能力。

2.341 精度 precision

a. 分辨几乎相等诸值的能力的一种度量。例如，4 位数字在精度上小于 6 位数字；但适当地计算所得出的 4 位数字可以比不适当地计算所得的 6 位数字更精确。

b. 对所提及的数量的可分辨的程度。例如，三位数字可分辨 1000 种可能性。与 2.9 条相对照。

2.342 预编译程序 precompiler

一种计算机程序。它对源代码(其中有一部分可能是编译程序无法接受的)进行预处理，以产生与之等效但可为编译程序接受的代码。例如，把结构化 FORTRAN 转换为 GB 3507FORTRAN 的预处理程序。

2.343 概要设计 preliminary design

a. 分析各种设计方案和定义软件体系结构的过程。典型的概要设计包括计算机程序组成成分和数据的定义及构造、界面的定义，并提出时间和规模方面的估计。

b. 概要设计过程的结果。参见 2.135 条、2.216 条。

2.344 预处理程序 preprocesser

进行某些初步计算或组织的计算机程序。参见 2.342 条。

2.345 特权指令 privileged instruction

只允许管理程序使用的指令。

2.346 过程，规程 procedure

a. 计算机程序的一个部分。它被命名并实现一个特定的任务。比较 2.482 条、2.480 条、2.213 条、2.301 条。

b. 为解决某一问题而采取的动作的经过。

c. 为解决某一问题而采取的动作的经过的描述。

d. 每次完成一任务时要遵循的一组手工的步骤。

2.347 进程，处理 process

a. 在计算机系统中，根据在给定条件下所要达到的目的或效果而定义的若干事件的一个唯一有限的过程。

b. 在进程中对数据进行的操作。

2.348 产品 product(software)

要交付给用户的一套完整的计算机程序、过程以及有关的文档和数据。

2.349 产品认证 product certification

参见 2.59 条。

2.350 产品库 product library

一个软件库。其中含有已被批准供当前运行使用的软件。

2.351 产品规格说明 product specification

与 2.143 条同义。

2.352 程序，计划 program

a. 计算机程序。参见 2.81 条。

b. 规定要采取的动作的一个时间表。

c. 设计、编写并测试计算机程序。

2.353 程序体系结构 program architecture

计算机程序部件之间的结构和关系。程序体系结构也可以包括它和程序运行环境之间的程序界面。

2.354 程序块 program block
在面向问题的语言中，计算机程序的子部分，用于把语句分组、划分例行程序、规定存储分配、确定标号的可使用性、或者为其他一些目的而将计算机程序的分段。

2.355 程序正确性 program correctness
参见 2.110 条。

2.356 程序设计语言 program design language
参见 2.138 条。

2.357 程序扩展 program extension
对现存软件进行增强以扩大程序能力的范围。

2.358 程序探测 program instrumentation
a. 插入到计算机程序中的探头。如指令或断言，以利于执行监督、正确性证明、资源监督或其它活动。
b. 准备探头并把它插入到计算机程序中去的过程。

2.359 程序库 program library
计算机程序的有组织的集合。参见 2.447 条、2.494 条。

2.360 程序变异 program mutation
a. 对预期的程序版本故意进行改变而获得一个程序版本，用以估计程序测试用例的能力，看它能否检测出所做改变。
b. 为了估价程序测试数据的选择是否适当而建立程序变异的过程。

2.361 程序保护 program protection
为预防对计算机程序的任何非授权存取或修改而施行的内部或外部控制。

2.362 程序规格说明 program specification
a. 计算机程序的任何规格说明。参见 2.143 条、2.218 条、2.335 条、2.407 条。
b. 与 2.143 条同义。

2.363 程序支持库 program support library
参见 2.439 条。

2.364 程序综合 program synthesis
借助软件工具把程序规格说明变换为实现那个规格说明的程序。

2.365 程序确认 program validation
与 2.87 条同义。参见 2.538 条。

2.366 编程，程序设计 programming
a. 程序的编写工作。
b. 用源程序语言或某种代码为程序编码之前的一部分工作作设计。

2.367 编程语言，程序设计语言 programming language
用来设计生成或表达程序的人工语言。

2.368 程序设计支持环境 programming support environment
通过单一命令语言来使用的工具的完整集合。用以提供在整个软件生存周期中的程序设计支持能力。典型的环境包括在设计、编辑、编译、装入、测试、配置管理及项目管理中所用的工具。

2.369 项目文件 project file
参见 2.370 条。

2.370 项目簿 project notebook
书面资料(备忘录、计划、技术报告等等)的中心储藏处。与 2.369 条同义。参见 2.440 条。

2.371 项目计划 project plan

描述工程项目所采取的开发方案管理文档。此计划通常包括要做的工作、所需要的资源、使用的方法、配置管理和要遵循的质量保证规程、要求的进度、项目组织等等。

2.372 项目进度表 project schedule

与 2.371 条同义。

2.373 提示 prompt

a. 通知用户系统已为执行下一条命令、下一条消息、或其它用户动作做好了准备。

b. 通知用户，系统已为执行下一条命令、下一个元素、或其它输入做好了准备。

2.374 正确性证明 proof of correctness

a. 数学上证明程序满足它的规格说明的形式技巧。参见 2.326 条、2.529 条。

b. 应用此技巧而得到的程序证明。

2.375 保护 protection

限制对计算机系统进行全部或部分存取或使用的一种安排。

2.376 协议 protocol

a. 一组约定或规则。它控制计算机系统或网络内的进程或应用的相互作用。

b. 一组规则。它控制功能部件的操作以达到通信的目的。

2.377 伪码 pseudo code

在计算机程序的设计工作中使用的程序设计语言和自然语言的组合。

2.378 下推式存储器 pushdown storage

按后进先出(LIFO)的方法处理数据的存储器。在这种方法中下一个要取的项是仍在存储器中的最后存入的那个项。参见 2.465 条。

鉴定 qualification

一个正式的过程，通过这个过程决定产品是否符合它的规格说明，是否可在目标环境中使用。

2.380 鉴定需求 qualification requirement

准则或一组条件，当一个产品符合这些准则或条件时，就确定它符合规格说明并可以在其目标环境中使用。

2.381 合格性测试 qualification testing

正式测试。通常是由开发者(供方)为客户(需方)进行的测试，以显示软件符合规定的需求。参见 2.6 条、2.497 条。

2.382 质量 quality

a. 产品或服务的全部性质和特征，能表明产品满足给定的要求。

b. 参见 2.452 条。

2.383 质量保证 quality assurance

为使某项目或产品符合已建立的技术需求提供足够的置信度，而必须采取的有计划和有系统的全部动作的模式。

2.384 质量度量 quality metric

对软件所具有的，影响其质量的给定属性所进行的定量测量。

2.385 队列 queue

按先进先出方法进行存取的一个列表。与 2.465 条相对照。

2.386 实时 real time

a. 这一术语涉及的是：按照计算机外部进程所提出的时间要求，使用计算机进行与该外部进程相关联的数据处理。也常用这个术语描述会话方式操作的系统，以及在运行中可受人工干预的那些进程。

b. 关于物理进程发生的实际时间；例如，在有关物理进程发生的那一段实际时间内所完成的

计算，以便在引导此物理进程中使用计算的结果。

2.387 记录 record

作为一个单位来处理的有关数据或字的集合。参见 2.277 条。

2.388 递归例行程序 recursive routine

一种例行程序。它可以作为自己的子例程来使用，它直接调用它自己或被它所调用的另一子例程所调用。使用递归例行程序通常要求在某处(例如下推表中)保持其尚未完成的状态的记录。

2.389 冗余 redundancy

以改善运行可靠性而引入重复或代替的系统元素，确保在一旦元素失效时系统能继续运行。

2.390 回归测试 regression testing

选择性重新测试，目的是检测系统或系统部件在修改时所引起的故障，用以验证上述修改未引起不希望的有害效果，或证明修改后的系统或系统部件仍满足规定的需求。

2.391 发行 release

一项配置管理行为，它说明某配置项的一个特定版本已准备好用于特定的目的(例如发行测试产品)。

2.392 可靠性 reliability

a. 在规定时间间隔内和规定条件下，一项配置实现所要求的功能的能力。

b. 参见 2.454 条。

2.393 可靠性评估 reliability assessment

确定现有系统或系统部件可靠性所达到的水平的过程。

2.394 可靠性数据 reliability data

在软件生存周期中在选择的点上评价软件可靠性所需要的信息。例如可靠性模型中使用的错误数据和时间数据，程序属性(如复杂性)，程序设计特性(如使用的开发技术及程序员的经验)。

2.395 可靠性评价 reliability evaluation

参见 2.393 条。

2.396 可靠性增长 reliability growth

从纠正软件故障而得到的软件可靠性的改进。

2.397 可靠性模型 reliability model

预测、估计或评估可靠性所使用的模型。参见 2.393 条。

2.398 数据可靠性 reliability numerical

在规定时间间隔内和规定条件下，一配置项将实现所要求的功能的概率。

2.399 重定位机器代码 relocatable machine code

要求在计算机执行之前把相对地址翻译为绝对地址的机器语言代码。与 2.2 条相对照。

2.400 会合 rendezvous

在两个平行任务之间，当一个任务已调用另一任务的入口，并且后一任务也正在为前者执行一相应的接受语句时所出现的相互作用。

2.401 可重复性 repeatability

参见 2.516 条。

2.402 招标(标书) request for proposal [tender]

需方使用的一份文件，用来向潜在的投标人表示它要获得某特定系统、产品或服务的意图。

2.403 需求 requirement

a. 用户为解决某一问题或达到某个目标所需要的条件或能力。

b. 系统或系统部件为满足或具有的条件或能力以满足合同、标准、规格说明或其它正式的强制性文件。所有需求的集合形成了以后开发系统或系统部件的基础。参见 2.404 条、2.406 条、

2.407 条。

2.404 需求分析 requirements analysis

a. 研究用户要求以得到系统或软件需求的定义的过程。

b. 对系统需求或软件需求的验证。

2.405 需求审查 requirements inspection

参见 2.237 条。

2.406 需求阶段 requirements phase

软件生存周期中的一个阶段。在此期间对软件产品的需求(如功能和性能方面的能力)进行定义并编制出相应的文档。

2.407 需求规格说明 requirements specification

陈述系统或系统部件(例如,软件配置项)的需求的规格说明,通常包括功能需求、性能需求、接口需求、设计需求以及开发标准。

2.408 需求的规格说明语言 requirements specification language

具有特殊构造和验证协议的形式语言,用于规定、验证和编制需求文件。

2.409 需求验证 requirements verification

参见 2.539 条。

2.410 退役 retirement

操作和维护机构撤出现有的支持,全部或部分地由一个新的系统来代替或者安装一个更新的系统。

2.411 退役阶段 retirement phase

软件生存周期中的一个阶段。在此阶段内,对软件产品的支持被终止。

2.412 可重用性,复用率 reusability

一个模块可在多种应用中加以利用的程度。

2.413 评审 review

参见 2.142 条。

2.414 健壮性 robustness

尽管引入了不合理的输入,软件仍能继续正常运行的程度。

2.415 根编译程序 root compiler

一种编译程序。其输出是与机器无关的中间表示。当它与依赖于机器的代码生成程序组合时,就构成了完整的编译程序。

2.416 例行程序,例程 routine

a. 实现特定任务的一个计算机程序段。参见 2.213 条、2.346 条、2.482 条、2.480 条。

b. 广泛或频繁使用的程序段,或由程序调用的指令序列。

2.417 运行态,运行方式 run mode

当计算机正自动地执行着存储在存储单元中的指令,从而被认为是正在履行其功能时的那种状态。

2.418 运行时间 run time

a. 执行一个程序时所花费时间的度量。运行时间包括中央处理机时间、外围处理时间和外围存取时间,例如:5 h 的运行时间。

b. 程序开始执行的瞬间。参见 2.188 条。

2.419 可扩展性 scaliability

是指软件系统可以在不同规模、不同档次的硬件平台上运行的能力。

2.420 安全性,保密性 security

对计算机硬件、软件进行的保护，以防止其受到意外的或蓄意的存取、使用、修改、毁坏或泄密。安全性也涉及对人圆劲数据、通信以及计算机安装的物理保护。

2.421 安全核心 security kernel

与安全性有关的关键性语句的一个小的、自含的集合，作为操作系统的特权部分。为了使用一程序或存取某数据，必须满足核心所规定的全部准则。

2.422 撒播 seeding

参见 2.201 条。

2.423 程序段，存储段，分段 segment

a. 计算机程序中的独立部分，它可在任一时间执行而无需把整个计算机程序都保持在内存中。参见 2.77 条、2.301 条、2.480 条。

b. 在两个相邻转移分支点之间的计算机程序语句的序列。参见 2.328 条。

c. 把计算机程序和数据分为若干段。

2.424 语义 semantics

a. 字符或字符组与其含义之间的关系。这种关系是与它们的解释和使用的方式无关的。

b. 符号和它们的含义之间的关系。

c. 按照元语言表达计算机语言结构的含义的规则。

参见 2.489 条。

2.425 信号量 semaphore

用于同步并行进程的一种共享变量。它用来指明某动作是否已完成或某事件是否已发生。

2.426 顺序进程 sequential processes

以这样一种方式执行的进程，即在下一个动作开始之前本动作必须结束。与 2.90 条相对照。

2.427 服务(软件) service (software)

与软件有关的活动、工作或义务的实施，例如软件的开发、维护和操作等。

2.428 严重性 severity

参见 2.116 条。

2.429 副作用 side effect

进行的处理或活动，或得到的结果，它们与程序、子程序、或操作的主要功能相比是处于第二位的。

2.430 模拟 simulation

由另一系统来表示某实际或抽象系统中选定的行为的特征。在数字计算机系统中，模拟由软件来做，例如，(a)借助于由计算机系统执行的操作表示物理现象。(b)用一计算机系统的操作表示另一个计算机系统的操作。与 2.23 条相对照。

2.431 模拟器 simulator

一个设备、一个数据处理系统、或一个计算机程序，它表示了某实际或抽象系统的行为的某些特征。

2.432 规模估计 sizing

对一个系统或系统部件所需要的源程序的行数或计算机存储的总量的估计。

2.433 软件 software

a. 与计算机系统的操作有关的计算机程序、规程、规则，以及可能有的文件、文档及数据。参见 2.25 条、2.496 条。

b. 与计算机系统的操作有关的程序、规程、规则及任何与之有关的文档。与 2.220 条相对照。

2.434 软部件 software component

一个软件配置项中的一个明确的部分。

注：一个软部件含有软件的多个单元；也可以含有多个较低级的软部件。

2.435 软件配置 software configuration

软件产品在不同时期的组合。该组合随着开发工作的进展而不断变化。

2.436 软件配置管理 software configuration management

参见 2.98 条。

2.437 软件数据库 software data base

存放运行软件系统内部公共数据的数据定义及其当前值的文件。

2.438 软件开发周期 software development cycle

a. 从决定开发一个软件产品开始到产品交付为止的时间间隔。这个周期通常包括需求阶段、设计阶段、实现阶段、测试阶段，有时还包括安装和验收阶段。与 2.448 条相对照。

b. 从决定开发软件产品开始到开发者不再改进产品时为止的时间周期。

c. 有时作为软件生存周期的同义语使用。

2.439 软件开发库 software development library

存放与软件开发工作有关的计算机可读信息和人们可读信息的软件库。

2.440 软件开发簿 software development notebook

有关给定软件模块情况材料的集合。其内容通常包括与给定软件模块有关的需求、设计、技术报告、代码列表清单、测试计划、测试结果、问题报告、进度、注释等等。参见 2.370 条。

2.441 软件开发计划 software development plan

为开发某一软件产品而做的项目计划。与 2.86 条同义。

2.442 软件开发过程 software development process

把用户要求转化为软件需求，把软件需求转化为设计，用代码来实现设计，对代码进行测试，完成文档编制，并确认软件可以投入运行性使用的过程。

2.443 软件文档 software documentation

以人们可读的形式出现的技术数据和信息。包括计算机列表和打印输出，它们描述或规定软件设计或细节，说明软件具备的能力，或为使用软件以便从软件系统得到所期望的结果而提供的操作指令。参见 2.157 条、2.493 条、2.536 条。

2.444 软件工程 software engineering

软件开发、运行、维护和引退的系统方法。

2.445 软件经验数据 software experience data

与软件的开发或使用有关的数据。这在开发软件模型、可靠性预测，或软件的其它定量描述中可能是有用的。

2.446 软件库管理员 software librarian

负责建立、管理和维护软件库的人员。

2.447 软件库 software library

软件和有关的文档说明的一个受控制的集合。目的是有助于软件开发、使用或维护。类型包括软件开发库、主库、产品库、程序库和软件储藏仓。参见 2.494 条。

2.448 软件生存周期 software life cycle

从设计软件产品开始到产品不能再使用时为止的时间周期。软件生存周期通常包括需求阶段、设计阶段、实现阶段、测试阶段、安装和验收阶段、运行和维护阶段，有时还包括引退阶段。与 2.438条相对照。

2.449 软件维护 software maintenance

a. 在一软件产品交付之后对其进行修改，以纠正故障。

b. 在一软件产品交付之后对其进行修改，以纠正故障，改进性能和其它属性，或使产品适应改

变了的环境。参见 2.16 条、2.109 条、2.332 条。

2.450 软件监督程序 software monitor

和另一计算机程序并行执行的软件工具，并对那个程序的执行情况提供详细的信息。

2.451 软件产品 software product

指定交付给用户的软件实体。

2.452 软件质量 software quality

a. 软件产品中能满足给定需要的性质和特性的总体。例如，符合规格说明。

b. 软件具有所期望的各种属性的组合程度。

c. 顾客和用户觉得软件满足其综合期望的程度。

d. 确定软件在使用中将满足顾客预期要求的程度。

2.453 软件质量保证 software quality assurance

参见 2.383 条。

2.454 软件可靠性 software reliability

a. 在规定条件下，在规定的时间内软件不引起系统失效的概率。该概率是系统输入和系统使用的函数，也是软件中存在的缺陷的函数。系统输入将确定是否会遇到已存在的缺陷(如果有缺陷存在的话)。

b. 在规定的时间周期内所述条件下程序执行所要求的功能的能力。

2.455 软件档案库 software repository

一个软件库。它用于存储软件和有关文档的永久性的档案。

2.456 软件潜行分析 software sneak analysis

施用于软件的一种技术。用以识别潜伏的(潜行的)逻辑控制路径或条件。这些路径或条件会禁止所期望进行的操作或引起不希望有的操作出现。

2.457 软件工具 software tool

一种计算机程序。用来帮助开发、测试、分析或维护另一计算机程序或它的文件。例如，自动设计工具、编译程序、测试工具、维护工具。

2.458 软件单元 software unit

一段可分开编译的代码。

2.459 源语言 source language

a. 用来书写源程序的语言。

b. 其语句被翻译的一种语言。与 2.501 条相对照。

2.460 源程序 source program

a. 在计算机执行之前必须被编译、汇编或解释的计算机程序。

b. 用源语言表达的计算机程序。与 2.312 条相对照。

2.461 规格说明，规范 specification

a. 以一种完全的、精确的、可验证的方法规定系统或系统部件的需求、设计、性能或其它特性的文件。参见 2.143 条、2.211 条、2.218 条、2.251 条、2.335 条、2.407 条。

b. 制定规格说明的过程。

c. 对某产品、某种材料或进程将要满足的一组需求的扼要陈述，并在适当的时候，指明一种过程，根据该过程可确定给定需求是否得到满足。

2.462 规格说明语言 specification language

一种语言。常常是机器可处理的自然语言和形式语言的组合。用来规定系统或系统组成成分的需求、设计、性能或其它特性。参见 2.138 条、2.408 条。

2.463 规格说明验证 specification verification

参见 2.539 条。

2.464 稳定性 stability

a. 在有干扰或破坏事件影响下仍能保持不变的能力。

b. 在干扰或破坏性事件之后返回到原始状态的能力。

2.465 栈 stack

按后进先出方法进行存取的一个列表。与 2.385 条相对照。

2.466 标准实施器 standards enforcer

一种软件工具。它确定指定的开发标准是否得到遵循。标准可以包括模块大小、模块结构、注释的约定、某些语句形式的使用以及文件编制约定。

2.467 状态图 state diagram

一种有向图。其中的结点对应于系统的内部状态，也对应于迁移；常常用来通过状态的改变来描述系统。参见 2.337 条。

2.468 静态分析 static analysis

估计程序而无需执行程序的过程。参见 2.146 条、2.63 条、2.237 条、2.545 条、2.164 条。

2.469 静态分析程序 static analyzer

一种软件工具。它有助于分析计算机程序而无需执行该程序，例如语法检验程序、编译程序、交叉引用表生成程序、标准实施器以及流程图。与 2.165 条相对照。

2.470 静态结合 static binding

在程序执行之前实现的，执行期间不加改变的结合。与 2.166 条反义。

2.471 统计测试模型 statistical test model

一种模型。它把程序故障与输入数据集(或多个数据集)联系起来。模型也给出了这些故障将引起程序失效的概率。

2.472 逐步细化(法) stepwise refinement

系统开发方法学，在其中首先概括地决定数据定义和处理步骤，然后逐步增加细节。参见 2.222 条、2.526 条、2.52 条。

2.473 串 string

实体。如字符或物理元素的线性序列。

2.474 强类型 strong typing

一种程序设计语言特性。它要求对每个数据对象的数据类型都作出说明，并排除操作符施用于不适当的数据对象上的情况。因此，防止了不相容类型的数据对象的相互作用。

2.475 结构化设计 structured design

进行软件设计的一种带有约束性的方法。它遵循一组指定的规则。这些规则是建立在诸如自顶向下设计、逐步求精法和数据流分析等原则基础上的。

2.476 结构化程序 structured program

由一组基本的控制结构构造而成的程序。每一个控制结构有一个入口点和一个出口点。控制结构组典型地包括：由两条或多条指令组成的序列；两个或多个指令或指令序列的条件选择；一个指令或指令序列的重复执行。

2.477 结构化程序设计 structured programming

a. 一种定义良好的软件开发技术。它采用自顶向下设计和实现方法，并严格地使用结构化程序的控制构造。

b. 不严格地讲，指组织和编写程序的一种技术。这种技术可简化复杂性，改进明晰度，并便于排除隐错和修改。

2.478 结构化程序设计语言 structured programming language

一种程序设计语言。它提供了结构化程序的构造并有利于结构化程序的开发工作。

2.479 桩模块 stub

a. 在较高级的模块和测试期间所使用的取代低层模块的虚程序模块。

b. 用来代替程序单位并指明该单位是在别处定义或将在别处定义的程序语句。

2.480 分包商 sub-contractor

依据合同向合同当事人的一方提供系统、产品或服务的一个机构。

2.481 子程序 subprogram

可以被一个或多个其它的程序单位所调用的程序单位。例如，过程、函数、子例行程序。

2.482 子例行程序，子例程 subroutine

a. 一组顺序的语句。可在一个或多个计算机程序中以及在一个计算机程序内的一处或多处使用。

b. 可以作为另一例行程序一部分的例行程序。

c. 由调用语句所调用的子程序。它可以接受或不接受输入值，并通过参数名、程序变量或机构而不是子例行程序名自身来返回任何输出值。与 2.213 条相对照。参见 2.346 条。

2.483 子系统 subsystem

一组装配件或部件，或两者的组合，以实现单一的功能。

2.484 管理程序 supervisor

参见 2.485 条。

2.485 管理程序 supervisory program

一种计算机程序，通常是操作系统的一部分。它控制其它计算机程序的执行并且调节数据处理系统中的工作流程。与 2.190 条、2.484 条同义。

2.486 供方 supplier

按照所签的合同向需方提供系统、产品或服务的一个机构(是合同当事人、生产者、卖方、批发商的同义词)。

注：需方可以指定它的机构中的某一部门做为供方。

2.487 支持软件 support software

用于帮助和支持开发的软件。

2.488 符号执行 symbolic execution

一种验证技术。在这种技术中，模拟程序的执行是使用符号而不是真实的值来代表输入数据，而程序输出则表示成包含这些符号的逻辑或数学表达式。

2.489 语法 syntax

a. 字符或字符组之间的关系。这种关系与它们的含义或它们的解释及使用的方法无关。

b. 语言中表达式的结构。

c. 管辖语言结构的规则。参见 2.424 条。

2.490 系统 system

a. 人、机器和方法的集合。用来实现一组规定的功能。

b. 一个完整的整体。它由种类不同的、相互作用的、专门的结构和子功能部件所组成。

c. 由某些相互作用或相互依赖关系联合起来的小组或子系统。它可执行多种职能，但是作为一个单位而发挥其作用。

2.491 系统体系结构 system architecture

系统各部件之间的结构和关系。系统体系结构也可以包括系统和它的运行环境之间的界面。

2.492 系统设计 system design

a. 为系统定义硬件和软件结构、部件、模块、接口及数据，以满足规定的系统需求的过程。

b. 系统设计过程的结果。

2.493 系统文档 system documentation

表达系统的需求、设计思想、设计细节、能力、限制，以及其它特性的文档。与 2.536 条相对照。

2.494 系统库 system library

驻留于系统中的受控软件的集合。可供存取和使用，或通过引用而合并到其它程序中去。例如，在需要时由连接编辑程序合并到某程序中去的一组例行程序。参见 2.447 条。

2.495 系统可靠性 system reliability

包括全部硬件和软件子系统在内的某个系统在规定的环境中及规定的时间里正确执行所要求的任务或使命的概率。参见 2.318 条、2.454 条。

2.496 系统软件 system software

管理、使用和维护计算机系统资源的软件。例如，操作系统、编译程序、实用程序。与 2.25 条相对照。

2.497 系统测试 system testing

测试整个硬件和软件系统的过程。以验证系统是否满足规定的需求。参见 2.6 条、2.381 条。

2.498 系统确认 system validation

参见 2.538 条。

2.499 系统验证 system verification

参见 2.539 条。

2.500 表格 table

a. 数据的一个阵列，其中每一项均可借助于一个或多个变元清楚地予以标识。

b. 数据的一个集合，其中每一项可由标号、位置，或按某些其它方法唯一地标识。

2.501 目标语言 target language

由源语句翻译成的语言。与 2.459 条相对照。

2.502 目标机 target machine

a. 打算在其上运行程序的计算机。

b. 由另一台计算机加以模拟的计算机。与 2.226 条相对照。

2.503 任务 task

构成活动的基本元素，由若干个任务构成一项活动。

2.504 终止性证明 termination proof

正确性证明中的一项内容，它证明在全部规定的输入条件下，程序将终止。

2.505 测试台 test bed

a. 测试环境。其中包括测试系统或系统部件所必须的硬件、探测工具、模拟程序，以及其它支持软件。

b. 在测试系统或系统的部件时所必需的全部测试用例的汇集。

2.506 测试用例 test case

测试数据及与之相关的测试规程的一个特定的集合。它是为了特定目的(如考察特定程序路径或验证是否符合特定的需求)而产生出来的。参见 2.520 条。

2.507 测试用例生成器 test case generator

参见 2.38 条。

2.508 测试范围 test coverage

一个范围，在此范围内测试程序测试系统需求能否满足。

2.509 测试数据 test data

用来测试系统或系统部件的数据。参见 2.506 条。

2.510 测试数据生成器 test data generator

参见 2.38 条。

2.511 测试驱动程序 test driver

一种驱动程序。它调用被测试的对象，它还可以提供测试输入并报告测试结果。

2.512 测试日志 test log

按年月日所做的测试活动的全部有关细节的记录。

2.513 测试阶段 test phase

软件生存周期中的一段时间。在此期间对软件产品的部件进行评价且进行集成。并评价软件产品以确定需求是否已得到满足。

2.514 测试计划 test plan

一个文件，它叙述了对于预定的测试活动将要采取的途径。典型的计划中包括：标识要测试的项目、要完成的测试、测试进度表、人事安排要求、报告要求、评价准则，以及任何临界的要求的临时计划。

2.515 测试规程 test procedure

对给定的测试，就其建立、运行和结果估计所作的详细说明。常常把一组有关的过程组合起来形成测试过程文件。

2.516 测试可重现性 test repeatability

测试的一种属性。指明相同环境、不同时间进行的测试是否产生相同的结果。

2.517 测试报告 test report

描述对系统或系统部件进行的测试行为及结果的文件。

2.518 测试有效性 test validity

完成测试规定目标的程度。

2.519 可测试性 testablility

a. 软件的一种性质。它表明了既便于测试准则的建立又便于就这些准则对软件进行评价的程度。

b. 需求的定义便于对需求进行分析以建立测试准则的程度。

2.520 测试 testing

由人工或自动方法来执行或评价系统或系统部件的过程，以验证它是否满足规定的需求；或识别出期望的结果和实际结果之间有无差别。比较 2.128 条。

2.521 吞吐量 throughput

对计算机系统在一段时间内实现工作总量的度量。例如，每天的作业数。

2.522 分时 time sharing

a. 计算机系统的一种操作技术。它提供了在一台处理机中在时间上交替进行两个或多个进程。

b. 指在一个计算机系统上时间的交替使用，它能使两个或多个用户并发地执行计算机程序。

2.523 计时分析程序 timing analyzer

一种软件工具。用于估计或度量计算机程序的执行时间或部分计算机程序的执行时间。这可通过求每条路径中指令的执行时间之和得到，或由在程序中的规定点处插入探头并度量探头之间的执行时间而得到。

2.524 容错 tolerance

系统在各种异常条件下提供继续操作的能力。

2.525 工具 tool

a. 参见 2.457 条。

b. 用来分析软件或其性能的执行时间而得到。

2.526 自顶向下 top-down

指从层次的最高级部件处开始，逐步推进到较低级的方法。例如，自顶向下设计、自顶向下程序设计、自顶向下测试。与 2.52 条相对照。

2.527 自顶向下设计 top-down design

由整体到局部逐级细化的设计过程，即先标出系统的主要部件，并把它们分解为较低级成分，然后重复进行直到不能(或不必)再分解为止。与 2.53 条相对照。

2.528 自顶向下测试 top-down testing

通过对较低级部件进行模拟的办法来从顶到底逐步地检查按层次方式所构造的程序的过程。

2.529 完全正确性 total correctness

在正确性证明中，指出程序的输出断言是它的输入断言和处理步骤的合乎逻辑的结果，并且在全部规定的输入条件下程序能终止。与 2.326 条相对照。

2.530 踪迹，追踪 trace

a. 计算机程序执行情况的记录；它显示指令执行的顺序。

b. 在计算机程序执行期间出现的全部或某类指令或程序事件的记录。

c. 产生一个踪迹。

2.531 追踪程序 tracer

用于追踪的软件工具。

2.532 翻译程序，转换程序 translator

一种程序。它把一种语言的语句序列变换为另一种语言中的等价的语句序列。参见 2.30 条、2.73 条、2.255 条。

2.533 树 tree

由通过分支互相连接的结点组成的抽象的层次结构。其中：(a)每个分支把一个结点连接到一个直属的下级结点；(b)有唯一称为根的一个结点，它不附属于任何其它结点；(c)根之外的每个结点只直接从属于另一个结点。

2.534 类型 type

参见 2.127 条。

2.535 用户 user

使用可操作的系统完成一项特定的功能的个人或机构。(可以是买主或需方的同义词。)

2.536 用户文档 user documentation

一套文档。它为使用系统以期获得所希望结果的最终用户提供系统指令方面的信息。例如，用户手册。与 2.493 条相对照。

2.537 实用软件 utility software

计算机程序或例行程序。设计这种程序的目的是为其它应用软件、操作系统或系统用户提供他们所要求的某些通用支持功能。

2.538 确认 validation

在软件开发过程结束时对软件进行评价，以确认它和软件需求是否相一致的过程。参见 2.539 条。

2.539 验证 verification

a. 确定软件开发周期中的一个给定阶段的产品是否达到前阶段确立的需求的过程。参见 2.538条。

b. 程序正确性的形式说明。参见 2.374 条。

c. 评审、审查、测试、检查、审计等活动，或对某些项、处理、服务或文件等是否和规定的需求相

一致进行判断和提出报告。

2.540 验证系统 verification system

参见2.39条。

2.541 版本 version

某一配置项的一个可标识的实例。

注：软件某版本的修改产生一个新的版本，但它需要配置管理活动。

2.542 虚拟机 virtual machine

对计算机及其有关设备的功能模拟。

2.543 虚拟空间 virtual space

a. 计算机制图中的坐标空间，空间中的每一单元由用户定义的坐标系表示。

b. 用户程序运行的一个逻辑地址空间。

2.544 虚拟存储器 virtual storage

可以被计算机系统的用户看作可编址主存储器的存储空间。在程序运行时虚地址被映射为实地址。虚拟存储器的大小由计算机系统的编址方式及所能使用的辅助存储器的总容量确定，而不受主存储器的实际容量所限制。

2.545 走查 walk-through

评审过程。在此过程中设计者或程序员引导开发小组中一名或多名其它成员通读已书写的设计或编码，其它成员负责提出问题并对有关技术、风格、可能的错误、是否违背开发标准的地方等进行评论。与2.237条相对照。

中 文 索 引

D

F

G

H

J

K

L

M

N

O

P

T

W

X

Y

Z

英 文 索 引

A

B

C

D

E

F

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

附加说明：

本标准由中华人民共和国电子工业部提出。

本标准由电子工业部标准化研究所归口。

本标准由中国计算机软件与技术服务总公司负责起草。

本标准主要起草人周明德、贾耀良、田云。

本标准于 1989 年 7 月首次发布。