机载系统和设备认证中的软件考虑因素

RTCA / DO - 178 A

由 RTCA SC – 167 / EUROCAE WG – 12 于 1992 年 12 月编制

本文件副本可以从 RTCA 公司获得

通信地址：美国华盛顿特区康涅狄格大道 1140 号

电话：202 – 833 9339

传真：202 – 833 9334

有关价格和订购信息请与 RTCA 公司联系

目录

前言

本文件是 RTCA 公司 167 专门委员会负责编制的，并于 1992 年 12 月经过 RTCA 公司同意。

RTCA 是一个由来自政府和行业人员组成的美国航空组织协会。它致力于推进航空工业的进步，在有关航空电子和通信应用领域寻求有效合理的技术解决方案。其目标就是通过成员和参与组织之间的相互协商解决这类问题。

RTCA 的研究结果理所当然要推荐给所有有关组织。因为 RTCA 不是美国政府的一个官方机构，其推荐的内容不可以被视为政府官方政策的声明，除非经过对有关推荐的问题具有法定管辖权的联邦政府组织或机构正式宣布。

这些指南的编制是由 RTCA SC - 167 和欧洲民航设备组织（EUROCAE）WG - 12 在协商一致的基础上共同完成的。

1 简介

二十世纪八十年代初，用于飞机和发动机的机载系统和设备用软件应用的快速增长促使人们提供满足适航要求可为行业接受的指导性文件。编制文件 DO - 178，即“机载系统和设备认证的软件考虑因素”就是为了满足这一需要。

本文件如今根据经验进行修订，它可以为航空通信提供相应的指导，以确定机载系统和设备用软件符合适航要求。随着软件应用的日益广泛，技术的不断演变，以及在本文件应用过程中经验的不断积累，本文件将会不断经过审核，得到进一步修订。附件 A 包含本文件的版本演变和修订经历。

1.1 目的

编制本文件的目的就是为那些执行预定功能的机载系统和设备用软件的开发提供指南。这些机载系统和设备必须具有一定的保密性、安全性，符合适航要求。这些指南采用下列形式：

- 软件生命周期过程所要达到的目标；
- 为达到这些目标所采取的各项活动和设计考虑因素的说明；
- 显示这些目标已经得到满足的相应证据的描述。

1.2 范围

本文件讨论有关用于飞机和发动机的机载系统和设备用软件形成的适航证的各个方面。在讨论过程中，将描述该系统生命周期及其与软件生命周期的关系，以便于理解其认证过程。这并不意味着要提供该系统生命周期过程的完整描述（包括系统安全评估和验证过程或飞机和发动机认证过程）。

因为本文件只讨论与软件生命周期有关的认证问题，所以，所得软件的运行方面在此就不再讨论。例如，用户可修改数据的认证就超出了本文件的适用范围。

本文件并不提供有关申请人组织结构、申请人与其供应商之间关系或如何划分相关责任的指南。软件开发人员资格认证标准也超出本文件的适用范围。

1.3 与其它文件的关系

除了适航要求外，还提供了各种适用于软件的国家和国际标准。在某些地区，可能要求遵循这些标准。不过，引用特定的国家或国际标准或提出一个用这些标准作为本文件的替代或补充文件已经超出了本文件的适用范围。

当本文件使用术语“标准”时，它应该被解释为适用于这种机载系统和机载设备、发动机或飞机制造商的项目特定标准。这些标准可以来自这些制造商为其活动所制定或采用的总标准。

1.4 如何使用本文件

在使用本文件时必须注意下列几个要点：

- 本文件中包含有解释性文本，以帮助读者理解所讨论的议题。例如，第 2 节提供了理解系统生命周期和软件生命周期之间交护作用所必须的信息。同样，第 3 节是软件生命周期的描述，第 10 节则是飞机及发动机认证的概览。
- 本文件主要用于国际航空界。为了有助于这一用途，文件中尽量减少了特定国家标准和措施参考文献，而采用通用术语。例如，采用术语“认证主管部门”来表示代表负责飞机及发动机认证的国家授予许可证的组织和个人。如果在该认证中涉及到另一个国家或几个国家，那么，在使用本文件时则要附加相关国家之间的双边协议或备忘录。
- 本文件认识到本文件所提供的指南不是法律强制规定，而是代表航空界的一种共识。本文件还认

识到申请人可以采用本文件所述的方法以外的其它替代方法。为此，文件中避免使用诸如“应”和“必须”等词语。

• 本文件说明了划分软件等级的目标，参阅段落 2．2．2 中的规定。附件 A 对这些软件等级目标的变更做出相应规定。如果某一申请人为了认证而采用这一文件，那么为达到这些目标，可以将它作为一整套指南来使用。

• 第 11 节包括一些数据。这些数据一般用来帮助人们在认证过程了解软件特性。该数据的名称用其名称中每个词的首个大写字母来表示。例如源代码。

• 第 12 节将讨论包括使用以前开发的软件、软件工具认证和使用在第 2 节~第 11 节所述的替代方法在内的其它考虑因素等方面的指导意见。第 12 节可能不适用于每一种认证。

•软件等级变动表和术语汇编包含在附件中，可以提供相关信息资料，属于本文件的一个组成部分。其它材料包含在附录中，属于本文件一个资料性提示部分。

• 在采用举例来说明如何使用文件中的指南的情况下，无论是采用图解或通过叙述方式，这些例子都不能被理解为优先采用的方法。

• 所提供的项目列表并不意味着该列表项目全部包括在内。

• 在本文件中采用了附注，以提供解释性材料，强调某一点或者引起人们注意没有完全在文本中叙述的相关的项目。附注中不包括指导性原则。

1.5 文件概略

图 1 – 1 是本文件章节及其相互关系的图示概览。

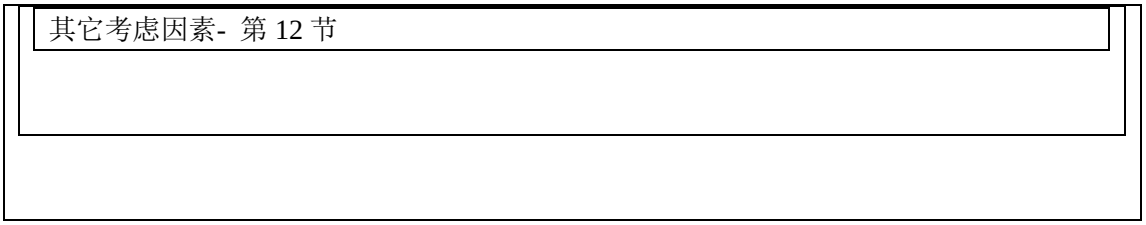

图 1 – 1 文件概览

2 有关软件开发的系统情况

本章节讨论理解软件生命周期所必须的系统生命周期过程方面的一些情况。所讨论的问题是：

- 系统和软件生命周期之间的数据交换（2．1 节）；
- 失效状态的分类、软件等级的定义以及软件等级的确定（2．2 节）；
- 系统结构方面所要考虑的一些因素（2．3 节）；
- 用户可修改软件、根据选项可进行选择软件以及商用现有软件的系统考虑因素（2．**4** 节）；
- 可在现场加载的软件的设计考虑因素（2．5 节）；
- 系统检验中系统要求所要考虑的一些因素（2．6 节）；
- 系统检验中软件所要考虑的一些因素（2．7 节）。

2.1 系统生命周期和软件生命周期之间的信息流

图 2 - 1 系统生命周期过程和软件生命周期过程之间在安全方面的信息流综述示意图。由于系统安全评估过程和系统设计过程存在相互依赖的关系，因此，在这些章节中所描述的信息流存在重复现象。

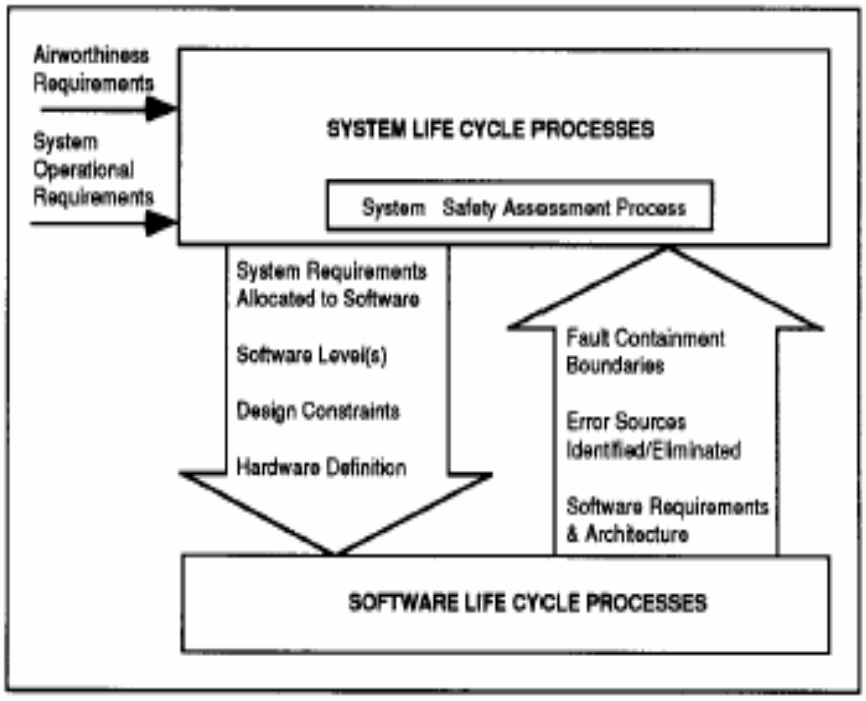

图 2 - 1 系统生命周期过程和软件生命周期过程之间在安全方面的信息流综述示意图

在本文件出版之时，用于系统生命周期过程的认证指南尚处于某一国际委员会的编制当中。在努力保持过程间信息流和定义兼容的情况下，最后出版的文件可能存在某些不尽相同之处。任何不同都将会在未来修订版中予以取消。

2.1.1 从系统过程到软件过程的信息流

该系统安全评估过程确定该系统的失效状态，并对其进行分类。在系统安全评估过程中，对该系统设计进行分析就可以确定有关安全方面的要求，即规定由于这些失效状态所期望产生的抗扰性以及系统对这些失效状态的反应。确定硬件和软件的这些要求就可以预防和限制故障的影响，并可以提供故障检测和故障偏差范围。因为决定是在硬件设计过程和软件开发过程中做的，所以，其系统安全评估过程对所得到的系统设计进行分析，以便检验它是否满足有关安全方面的要求。

这些有关安全方面的要求是系统要求的一个组成部分，是软件生命周期过程的输入信息。为了保证有关安全方面的要求在整个软件生命周期内得当正确的实施，系统要求一般包括或参照下列内容：

- 系统描述和硬件定义。
- 认证要求，包括适用的联邦航空管理条例（美国联邦航空管理条例）、联合航空管理条例（欧洲航空管理条例）和美国航空管理通告等。
- 指定给软件的系统要求，包括功能要求、性能要求和有关安全要求。
- 软件等级和体现其确定过程的数据、失效状况及其分类以及指定给软件的相关功能。
- 安全对策和设计约束条件（包括设计方法），例如分隔、不一致、冗余或安全的检测。
- 如果该系统是另一个系统的一个组成部分，那么，就应该包括有关安全方面的要求和该系统失效状况。

系统生命周期过程可以确定用于软件生命周期过程的要求，这样有助于系统检验的完成。

2.1.2 从软件过程到系统过程的信息流

该系统安全评估过程确定了在使用软件生命周期过程提供的信息时软件设计和实施对系统安全的影响。这一信息包括故障抑制范围、软件要求、软件结构以及可以被检测或通过软件结构通过使用软件工具或软件设计过程中的其它方法加以消除的误差来源。系统要求和软件设计数据之间的可追溯性对于系统安全评估过程来说非常重要。

软件的改动可能会影响系统安全，因此，其修改必须与系统安全评估过程一样看待。

2.2 失效状态和软件等级

在考虑系统失效状态类别、软件等级定义、失效状态类别与软件等级之间的关系以及软件等级是如何确定的之后给出相应的工作指导。确定对飞机及其机上人员产生影响的失效状态的严酷度即可设定某一系统的失效状态类别。软件中的错误可能导致故障的出现，这也是产生失效状态的原因之一。因此，安全运行所必须的软件完整性水平与系统的失效状态密切相关。

2.2.1 失效状态的分类

对于失效状态类别的完整定义，参考适用的规定和指导材料、美国航空管理局 AC 25 -1309 -1 A 和/或联合航空管理局 AMJ 25 – 1309 修订本。所列出的失效状态类别是根据这一指导材料得出的，并包含在本章节中以便于使用本文件。类别划分如下：

a. 灾难性失效状态：将会阻止继续安全飞行和着陆的失效状态。
b. 危险性/严重-重大失效状态：将会降低飞机性能或机组人员处理不良工作状态能力的失效状态，其下降程度可能达到下列水平：
1) 安全边际或功能水平大幅下降；
2) 给机组人员带来身体上的不适或更大的工作负荷，使他们无法赖以准确或完整地完成任务；
3) 对机上人员产生不良影响，包括对其中少数人产生严重的或潜在的致命伤害；
c. 重大失效状态：将会降低飞机性能或机组人员处理不良工作状态能力的失效状态，其下降

程度可能达到一定的水平，例如，安全边际或功能水平明显下降、机组人员的工作负荷明显增大，或者降低了机组人员的工作效率，或者给机组人员带来身体上的不适甚至可能包括身体上的伤害。

d. 轻微的失效状态：对飞机安全降低不十分明显的失效状态。在这种情况下，它可能会影响机组人员的行动但仍在其控制能力范围内。次要的失效状态可能包括安全边际或功能水平小幅下降、机组人员的工作负荷稍微有所增大，例如改变例行的飞行计划，或者给机上人员带来不便。

e. 没有任何影响的失效状态：不会影响飞机的操作能力，也不会增大机组人员的工作负荷的失效状态。

2.2.2 软件等级的定义

软件等级是在系统安全评估过程中确定的，它是根据软件对潜在的失效状态的贡献率来划分的。软件等级表示要达到与认证要求相符所要付出努力的程度，它随着失效状态类别的变化而变化。软件等级的定义如下：

a. A 级：正如它在系统安全评估过程中所显示的那样，这是一种其异常状态将会导致或引起系统功能的失效并给飞机带来灾难性的失效状态的软件；

b. B 级：正如它在系统安全评估过程中所显示的那样，这是一种其异常状态将会导致或引起系统功能的失效并给飞机带来危险性的/严重的-重大的失效状态的软件；

c. C 级：正如它在系统安全评估过程中所显示的那样，这是一种其异常状态将会导致或引起系统功能的失效并给飞机带来重大失效状态的软件；

d. D 级：正如它在系统安全评估过程中所显示的那样，这是一种其异常状态将会导致或引起系统功能的失效并给飞机带来轻微的失效状态的软件；

e. E 级：正如它在系统安全评估过程中所显示的那样，这是一种其异常状态将会导致或引起系统功能的失效并但没有给飞机工作能力或飞行员工作负荷带来任何影响的软件。一旦软件被认证主管部门确定为 E 级，在本文件中无须进一步提供任何指南。

2.2.3 软件等级的确定

起初，系统安全评估过程主要是确定与某一特定系统的软件组成部分（简称组分）相适应的软件等级，而没有考虑到系统设计问题。在做出这一确认时，失效的影响（包括功能的丧失或功能异常现象）已经标出。

附注：*（1）申请人可能在未来开发过程中希望添加计划中的功能，并考虑到有关软件的系统要求中潜在可能导致更为严重的失效状态和更高的软件等级的变更情况。因此期望开发一种等级高于最初应用领域系统安全评估过程中确定的软件，因为替代一个更高级别的软件应用的软件生命周期数据的后期开发可能较为困难。*

（2）对于操作规程中强制要求但不影响飞机的适航性的机载系统和设备（例如飞行事故数据记录器）来说，其软件等级就需要与其指定的功能相适应。在某些情况下，软件等级可以在设备最低性能标准中加以规定。

如果软件组分的异常状态产生一种以上的失效状态，那么，该软件组分最严重的失效状态就为该软件组分确定了其软件等级。目前出现了各种不同类型的体系结构对策，例如 2．3 小节中所述的那些体系结构对策。在系统设计演变过程中，这就可能会导致软件等级需要进行修订。

一个系统功能可能会分配给一个或多个已经分区的软件组分。并行应用就是一个系统功能用多个软件组分来完成，这样，要产生这种失效状态就要求多个软件组分的异常状态。对于并行应用来说，

至少要有一个软件组分具有与该系统功能最严重的失效状态类别有关的软件等级。其它软件组分的软件等级则采用与该功能损失有关的失效状态类别来确定。这种应用实例参阅段落 2．2．3“多版本不相似软件”和段落 2．3．3“安全监测”。

串行应用就是多个软件组分用于一个系统功能，这样，任何软件组分的异常状态都可能产生这种失效状态。在这一应用中，软件组分将拥有与该系统功能最严重的失效状态类别有关的软件等级。某一软件等级的软件开发并不意味着要为该软件分配一种失效等级。因此，软件等级或根据软件等级评定的软件可靠性等级不能被系统安全评估过程用作硬件失效等级。

那些违背本段落 2．2．3 认证指南中的对策需要通过系统安全评估过程来进行验证。

2.3 系统体系结构的考虑因素

如果该系统安全评估过程确定该系统体系结构可以预防软件的异常状态导致某一系统出现最严重的失效状态的话，那么，该软件等级就由剩下的失效状态中最严重的那种类别（即软件的异常状态可能产生不良影响的那一类）来确定。该系统安全评估过程要考虑体系结构设计决策，以确定它们是否影响软件等级或软件的功能。在此，为几个体系结构对策提供指导原则，以便限制这些错误的影响或检测出错误，并提供可以接受的系统响应以包含这些错误。这些体系结构方法并不是被视为最佳解决方案或规定采用的解决方案。

2.3.1 分区

分区是在功能独立的软件各个组成部分之间提供彼此隔离以便包容和/或隔离故障并可以减少软件验证过程所做出的努力的一种方法。如果通过分区提供保护，那么，每一个分区的组分的软件等级就可以用该组分相关的最严重的失效状态类别来确定。

分区指导原则包括下列三项：

a. 在设计分区时，系统的这些方面应该加以考虑，以便确定破坏那种保护措施的潜在可能性：

⑴硬件来源：处理器、存储驱动器、输入/输出器件、中断和记时器；

⑵控制耦合：外部存取的脆弱性；

⑶数据耦合：共享和叠加的数据，包括迭式存储器和处理机寄存器；

⑷与系统保护机制相关的硬件设备的失效模式。

b. 软件生命周期过程应该分区设计考虑因素，包括各个分区的组分之间允许的交互作用的程度和范围，以及这种保护措施是否由硬件或硬件和软件结合体来实施。

c. 如果分区保护涉及到软件，那么，应该为该软件指定与该分区软件组分最高等级相对应的软件等级。

2.3.2 多版本不相似软件

多版本不相似软件是一种涉及产生两个或两个以上软件组分的系统设计方法。通过这一方法可以提供相同的功能，并可避免各个软件组分之间共同的错误来源。多版本不相似软件也可以称为多版本软件、不相似软件、N 版本编程或软件多样性。

在不相似性引入软件开发以前所完成或启用的软件生命周期各个过程依然是潜在的错误来源。系统要求对准备执行多版本不相似软件的硬件配置做出具体规定。

其不相似程度以及由此产生的保护程度通常无法进行测定。系统功能损失的概率将会大大增加，使得与不相似软件版本有关的安全检测程序可以检测出实际误差，或者出现瞬时值超出对比项阈值极限范围的现象。因此，不相似软件版本通常在软件验证过程所确定的适用于该软件等级的目标得到满足之后用来提供一种补充保护措施，参阅第 6 节。如果证据表明，在系统安全评估过程中所确定的系统功能的潜在损失尚处于可以接受的程度，那么，不相似软件的验证方法可能从那些用

来验证单一版本软件的方法中归纳出来。

多版本不相似软件将在段落 12．3．3 中进行讨论。

2.3.3 安全检测

安全检测是防止出现特定失效状态的一种预防措施，它可以直接检测出一种可能会导致失效状态的失效功能。检测功能可以在硬件、软件或一种硬件和软件的结合体中完成。

通过使用这种检测方法，所检测的功能的软件等级可以降到其相关系统功能损失有关的等级。为了达到这一降级之目标，应该确定下列 3 个重要的检测特征：

a．软件等级：安全检测软件指定了与上述功能最严重的失效状态有关的软件等级。

b．系统故障的覆盖范围：一个检测程序的系统故障覆盖范围的评估确保了检测程序的设计和应用应该使得这些故障在所有必要条件下都会被检测出来。

c．功能和检测程序的独立性：检测和保护装置并不被引发这种危险的相同失效状态变得无法发挥作用。

2.4 系统用户可修改软件、选项可选择软件和商用现有软件的系统考虑因素

用户修改的潜在影响通过系统安全评估过程来确定，主要用来编制软件要求，随后是软件验证过程活动。用户可修改软件的设计将在段落 5．2．3 中进一步讨论。影响非可修改软件及其保护措施或可修改软件范围的变更属于一种软件修改，这一修改将在段落 12．1．1 中进一步讨论。对于本文件来说，可修改的软件组分属于该软件的一个组成部分，文件规定它由用户来进行修改，而非可修改软件组分则不能由用户来进行修改。

一些机载系统和设备可能包括一些选项功能，其功能可以由软件编程选项而不是由硬件连接器插针来选择。选项可选择软件功能用来选择特定程序计算机内一个特定的配置。无效代码指南参阅段落 5．4．3。

系统用户可修改软件、选项可选择软件和商用现有软件的系统考虑因素指导原则包括下列内容：

a．用户可修改软件：如果系统要求规定可以进行用户修改时，用户可以在修改约束条件范围内修改软件而无须经过认证审核。

b．系统要求应该规定这种防止用户修改会影响系统安全的装置是否得到正确的实施。为用户修改提供保护的软件应该与其防止可修改软件组分中出现错误的功能处于相同的软件等级。

c．如果系统要求没有包含用户修改条款，那么，该软件不应该被用户修改，除非这种修改已经表明符合本文件的规定。

d．在用户修改时，用户应该承担所有系统用户可修改软件各个方面的责任，例如软件配置管理、软件质量保证以及软件的验证。

e．选项可选择软件：当文件中包含有软件编程选项时，应该提供相应的措施以确保在计算机设施环境内的特定程序计算机中那些未被允许的配置无法被无意中选择。

f．商用现有软件：包含于机载系统或设备的 COTS 软件应该满足本文件规定的各项目标。

g．如果 COTS 软件的软件生命周期数据存在缺陷，那么，该数据应该找到相应的理由，以满足本文件规定的目标。可以参阅段落 12．14“原始资料编制程序的升级”和 12．35“产品使用过程”中的相关说明。

2.5 现场可安装软件的系统设计考虑因素

现场可安装的机载系统软件是指无须从其设施中拆下系统或设备即可安装的软件或数据表。与软件数据加载功能有关的安全方面的要求是系统要求不可缺少的组成部分。如果无意中启用软件数

据加载功能可能引发一种系统失效状态，那么，适用于软件数据加载功能的安全方面的要求就会在其系统要求中加以规定。

有关现场可安装的软件的系统安全考虑因素主要包括下列项目：

- 已经崩溃或局部安装的软件的检测；
- 确定加载不适当软件的影响；
- 硬件/软件的兼容性；
- 软件/软件的兼容性；
- 飞机/软件的兼容性；
- 无意中启用软件数据加载功能；
- 软件配置识别显示器的损坏或崩溃。

现场可安装软件指导原则包括：

a. 除非在系统安全评估过程中另行判定，在安装局部或已经漏洞百出的软件检测装置时，应该为其指定相同的失效状态，或作为最严重失效状态的软件等级，或与使用软件安装功能有关的软件等级。

b. 如果加载不适当的软件或数据，系统会处于一种故障状态的话，那么，系统中每个分区的软件组分应该为这一种模式的运行规定安全方面的要求，这样就可以**提出**潜在的失效状态。

c. 软件加载功能（包括支持系统和程序）应该包括一个检测错误的软件和/或硬件和/或飞机组合的方法，并且应该提供适合于该功能失效状态的保护措施。

d. 如果软件是机载显示装置的组成部分，以确保该飞机符合一种经过认证的配置要求，那么，这种软件就应该编制为需要安装的最高等级的软件，或者该系统安全评估过程应该证明软件配置标识的端对端检查的完整性。

2.6 软件验证的系统要求方面的考虑因素

软件的系统要求是根据该系统的操作要求和安全方面的要求编制的，而其中安全方面的要求则是系统安全评估过程的结果。

其主要考虑因素有下列两个方面：

a. 机载软件的系统要求确定了该软件的两个特性：

⑴ 该软件完成系统要求中规定的功能；

⑵ 该软件并不显示在系统安全评估过程确定的特定异常状态。为此，可以附加一些系统要求以消除这种异常状态。

b. 这些系统要求应该随后编制成软件的高层次要求（可以通过软件检验过程活动来验证）。

2.7 系统验证过程中软件方面的考虑因素

系统验证指导原则超出了本文件的适用范围。不过，软件生命周期过程对系统验证过程有所帮助，并与其产生相互影响。需要提供与该系统功能有关的软件设计详细规范，以帮助系统验证的顺利进行。

系统验证可以提供代码结构的重要覆盖范围。系统验证测试的覆盖分析可以用来实现软件验证下面所述的各种测试活动的覆盖目标。

3 软件生命周期

本章节讨论软件生命周期过程、软件生命周期定义和软件生命周期过程之间的转换条件。本文件指南并不是规定一种最佳软件生命周期，而是说明由大多数生命周期组成的各个独立过程以及它们之间的相互影响。这些过程的独立性并不是特指执行它们的一个组织结构。对于每个软件产品来说，这种生命周期过程在形成时就包括这些过程。

3.1 软件生命周期过程

软件生命周期过程是：

• 确定某一项目的软件开发和整合过程活动的软件规划过程。第 4 节讨论软件规划过程。

• 产生软件产品的软件开发过程。这些过程是软件要求编制过程、软件设计过程、软件编码过程和整合过程。第 5 节讨论软件开发过程。

• 确保软件生命周期过程及其输出信息正确无误、可控和可靠的整合过程。这种整合过程是软件验证过程、软件配置管理过程、软件质量保证过程以及认证联络过程。充分理解这种整合过程在整个软件生命周期过程中与软件开发过程同时进行的重要性。第 4～9 节讨论软件整合过程。

3.2 软件生命周期的定义

一个项目通过选择每个过程的相关活动，并规定这些活动的完成顺序，并为这些活动指定相关责任即可确定一个或多个软件生命周期。

对于一个特定的项目来说，这些过程的完成顺序由该项目的特征来决定，例如系统功能和复杂性、软件大小和复杂性、要求的相对稳定性、以前开发成果的利用、开发战略和软件的有效性。在整个软件开发过程中，其通常的顺序依此为软件要求、软件设计、软件编码和软件整合。

图 3 – 1 表示具有不同软件生命周期的一个单一的软件产品中几个组成部分的软件开发过程运行顺序。组分 W 通过开发软件要求、使用这些要求来确定软件设计、并将这种设计应用于源代码，然后将该组分整合进硬件中即可完成一套系统要求。组分 X 表示用于已经过认证飞机或发动机的以前开发的软件的使用。组分 Y 表示可以根据软件要求直接编码的一个简单的分区功能的使用情况。组分 Z 表示一个样品成型战略的使用。一般来说，样品成型的目标就是要更好地理解软件要求，并降低软件开发和技术风险。其原始软件要求作为完成一个型样的基础。这一型样是在一种代表尚在开发中的系统的某一特定应用环境中评估的。评估结果可以用来改进软件要求。

一个软件生命周期过程可以重复进行，也就是说可以输入和再次输入。由于系统功能的不断开发、错综复杂、软件要求的开发、硬件的有效性、向前面的过程反馈以及该项目的其它特征等因素，其定时和重复程度有所不同。

所选择的软件生命周期的各个部分与增大的整合过程和软件验证过程活动的复合过程捆绑在一起。

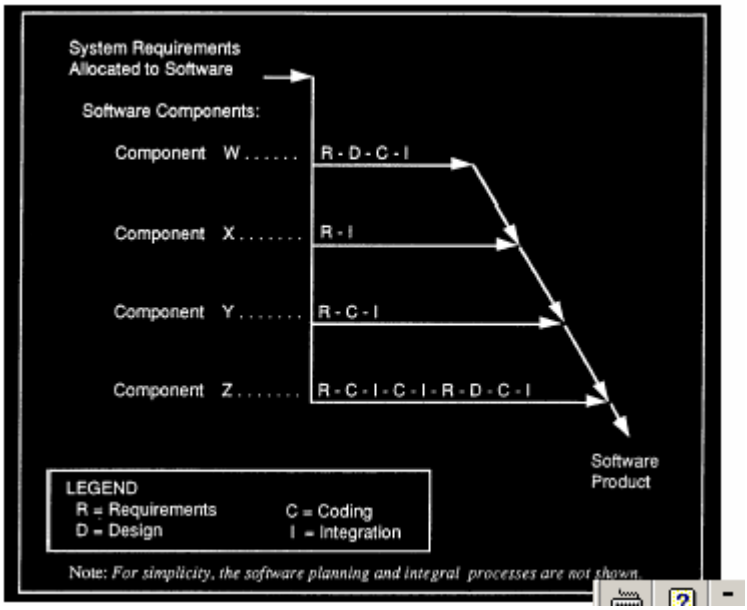

图 3 – 1
采用 4 个不同开发程序的软件项目的例证

3.3 过程之间的转换条件

转换条件用来确定是否可以输入或再次输入某一过程。每一个软件生命周期过程对输入信息进行处理，以形成输出信息。一个过程可以产生提供给其它过程的反馈信息，并接收来自其它过程的反馈信息。反馈的定义包括如何识别信息，并通过接收过程加以控制和解决。反馈定义的例子就是问题报告。

这种转换条件将取决于软件开发过程和整合过程所规划的相关程序。不过，转换条件可能受到软件等级的影响。转换条件（可以进行选择）的例子：完成软件检验过程的审核；输入信息属于一种标识的配置项目；为这种输入信息完成一个可追溯性分析。

如果这些过程所确定的转换条件得到满足的话，那么，该过程的每一条输入信息在该过程启动之前也不必那么完整。

a．如果一个过程作用于局部的输入信息，那么就应该检查随后得到的输入信息，以便确定以前的软件开发和软件检验过程的输出信息依然有效。

4 软件规划过程

本节讨论软件规划过程的目标和活动。这一过程形成指导软件开发过程和整合过程的软件计划和标准。附件 A 中的表 A – 1 就是按软件等级分类的软件规划过程的目标和输出信息。

4.1 软件规划过程目标

软件规划过程的目标就是要规定形成可以满足系统要求并可以确保符合适航要求的软件的措施和方法。软件规划过程目标列述如下：

a. 确定软件生命周期中的软件开发过程和整合过程的活动，以明确系统要求和软件等级，参阅 4．2 小节；
b. 确定软件生命周期（包括各个过程之间的相互关系）及其顺序、反馈机制和转换条件，参阅第 3 节；
c. 已经选择软件生命周期环境（包括用于每个软件生命周期过程的方法和工具），参阅 4．4 小节；
d. 提出其它考虑因素，例如第 12 节中讨论的那些因素；
e. 确定软件开发标准符合要开发软件的系统安全目标，参阅 4．5 小节；
f. 已经编制符合第 4．3 小节和第 11 节要求的软件计划；
g. 协调软件计划的编制和修订之间的关系，参阅 4．3 小节。

4.2 软件规划过程活动

有效的规划是产生符合本文件指南要求的一个决定因素。软件规划过程的指导原则包括：

a. 软件计划应该在软件生命周期中及时编制，以便为完成软件开发过程和整合过程的人员提供及时的指导。另见的 9．1 小节认证指南；
b. 应该确定或选择用于该项目的软件开发标准；
c. 在软件开发过程中应该选择提供错误预防的方法和工具；
d. 软件规划过程应该为软件开发过程和整合过程提供协调机制以便是软件计划中的各个战略前后保持一致，具有连贯性；
e. 软件规划过程应该包括一个随着项目的进展逐步修订软件计划的手段；
f. 如果系统中使用多版本不相似软件，那么，软件规划过程应该选择相应的方法和工具以达到避免出现错误的目标，或进行检测以达到系统的安全目标；
g. 对于要完成的软件规划过程来说，该软件计划和软件开发标准应该处于更改控制和审核范围内，参阅 4．6 小节；
h. 如果将失效代码也规划进去（参阅 2．4 小节），那么，软件规划过程应该说明失效代码（所选择的选项、飞行测试）如何定义、验证和处理，以达到系统的安全目标；
i. 如果将用户可修改代码也规划进去，那么，其过程、工具、环境和替代段落 5．3．2 认证指南中的数据应该在软件计划和标准中加以规定。

如果特定过程活动已经制定了相应的计划和步骤，那么，其它软件生命周期过程可以在软件规划过程完成之前开始。

4.3 软件计划

制定软件计划的目的就是要确定达到本文件规定目标的手段和措施。这些软件计划对完成这些活动的组织进行规定。软件计划具体包括：

• 用于软件认证方面的计划（11．1 小节）作为向认证主管部门交流所申报的开发方法以求得同意

的主要措施，并确定满足本文件要求的那些符合性措施。

- 软件开发计划（11．2 小节）确定软件生命周期和软件开发环境。
- 软件验证计划（11．3 小节）确定满足软件验证过程目标所采取的措施。
- 软件配置管理计划（11．4 小节）确定满足软件配置管理过程目标所采取的措施。
- 软件质量保证计划（11．5 小节）确定满足软件质量保证过程目标所采取的措施。

软件计划的指导原则包括：

a. 软件计划应该符合本文件要求；

b. 软件计划应该通过规定下列事项来确定软件生命周期过程之间过渡的条件：

⑴ 该过程的输入信息，包括来自其它过程的反馈信息；

⑵ 处理这些输入信息可能需要的一切整合过程活动；

⑶ 工具、方法、计划和步骤的有效性。

c. 软件计划应该说明在软件应用于飞机或发动机之前实施软件更改要采用的相应步骤。这种更改可能是其它过程反馈的结果，并可能导致该软件计划的更改。

4.4 软件生命周期环境规划

制定软件生命周期环境规划的目的就是要确定用来开发、验证、控制和形成软件生命周期数据（第 11 节）和软件产品的工具、方法、计划、步骤、程序编辑语言和硬件。所选择的软件环境如何对机载软件产生有利作用的例子包括实施标准，检测错误，并实施错误预防和故障容差方法。软件生命周期环境是导致失效状态的潜在错误来源。这一软件生命周期环境的组成可能受到有关安全方面要求（取决于系统安全评估过程）的影响，例如使用了不相似、冗余软件组分。

错误预防方法的目标就是要避免在软件开发过程中出现可能产生失效状态的错误。其基本原理就是选择可以减少引发错误机会的软件要求编制和设计方法、工具、程序编辑语言，以及确保所引发的错误可以被检测出来的检验方法。采取故障容差方法的目的就是要包括软件设计或源代码中的安全特征，以保证所开发的软件将对输入的数据错误做出正确反应，此外，它还可以预防输出错误，并可以控制这些差错。是否需要采取错误预防和故障容差方法取决于系统要求和系统安全评估过程。

- 上述考虑因素可能会影响：
- 软件要求编制过程和软件设计过程中所采用的方法和符合表示法；
- 软件编码过程中所采用的程序编辑语言和方法；
- 用于软件开发环境的工具；
- 用于软件验证和配置管理的工具；
- 对软件质量检验工具的需要。

4.4.1 软件开发环境

软件开发环境是高质量软件生产的一种重要因素。在某种情况下，这种软件开发环境也可能对航空机载软件产生不利影响。例如一个编译程序可能会由于产生一个不可靠的输出信息而带来相关差错，或者一个连接程序没有能够检测出存储器分配上所存在的差错。软件开发环境方法和工具方面的选择指导原则包括下列各项：

a. 在软件规划过程中，应该选择相应的软件开发环境，以减少最终机载软件潜在的风险；

b. 选择使用高质量的工具或工具与软件开发环境其中组成部分的组合来确保一个部分引发的差错将会被其它部分检测出来。当两个部分始终一起使用时，便形成了一个可以接受的软件开发环境。

c. 应该确定软件验证过程活动或软件开发标准（包括软件等级方面的考虑因素），以减少与软件开发环境有关的差错。

d. 如果所认证的合格证书用于组合工具，那么，这些工具的运行顺序应该相应的计划中加以规定。
e. 如果一个项目手采用软件开发工具的选项特征时，应该检查其选项的作用，并在相应的计划中加以规定。

附注：当这种工具直接产生软件产品时，这一点尤为重要。在本文中，编译程序可能是需要考虑的最重要的工具。

4.4.2 语言和编译程序方面的考虑因素

一旦某种软件产品成功地完成其验证，那么，这种编译程序对该软件产品来说就可以被视为合格。为了使之有效，软件验证过程的各项活动需要考虑到程序编辑语言和编译程序的特殊特征。当选择一种编程语言进行验证时，在软件规划过程中就应该考虑这些特征。其指导原则包括：

a．一些编译程序具有专门用来优化目标代码的特性。如果这种校验数据和条件所给的覆盖范围与软件等级一致的话，该优化的正确性就无须进行验证，否则，这些特征对结构覆盖分析的影响应该根据段落 6．4．4．2 中的指南来确定；

b．为了实施某些特性，用于某些语言的编译程序可以产生目标代码（无法直接追溯到源代码），例如首字母大写、嵌入错误检测或意外情况处置（段落 6．4．4．2 中的 b 项）。软件规划过程应该提供一种手段来检测这一目标代码，确保验证覆盖范围，并在相应的计划中确定这种手段。

c．如果引入一种新的编译程序、链接编辑程序或安装程序版本，或者在软件生命周期内编译程序选项发生了变化，那么，以前的测试和覆盖分析可能不再有效。验证规划应该提供一个与第 6 节和段落 12．1．3 中的指南相一致的手段来重新进行验证。

4.4.3 软件测试环境

编制软件测试环境规划的目的就是要确定可能用来测试整合过程的输出信息的方法、工具、步骤和硬件。测试可以采用特定程序计算机、特定程序计算机仿真机或主计算机模拟装置来完成。其指导原则规定如下：

a．仿真或模拟装置可能需要按照第 12．1 小节中所述的规定进行质量鉴定；

b．应该考虑特定程序计算机和仿真或模拟装置之间的区别，以及这些区别对错误检测能力和检验功能的影响。应该通过其它软件验证过程的各项活动来完成这些错误的检测，并在“软件验证计划”中加以规定。

4.5 软件开发标准

编制软件开发标准的目的就是要确定软件开发过程应该遵循的准则和约束条件。软件开发标准包括软件要求标准、软件设计标准和软件代码标准。软件验证过程使用这些标准作为评估实际输出信息与预定输出信息是否相符的基础。软件开发标准的指导原则包括：

a．软件开发标准应该符合第 11 节中的要求。

b．软件开发标准应该可以使某一给定的软件产品或相关的一组产品的软件组分能够前后设计和执行保持一致；

c．软件开发标准应该不允许使用可能生成那些无法进行验证的输出信息或无法于安全方面的要求兼容的构造或方法。

附注：*在编制标准时，可以考虑以前的经验。可以在标准中包含一些对开发、设计和编码方法的约束条件和规定以控制其复杂性。可以考虑采用防御性的编程方法以提高其可靠性。*

4.6 软件规划过程的审核和保证

实施软件规划过程的审核和保证，以确保软件计划和软件开发标准符合本文件指南，并为这些软件计划和软件开发标准的实施提供保障措施。指导原则如下：

a. 所选择的方法能够实现本文件的目标；
b. 软件生命周期过程可以始终如一地进行应用；
c. 每个过程都可以产生相应的证据，证明其输出信息可以追溯到其相应的活动和输入信息，并显示该活动、环境和使用的方法的独立程度；
d. 软件规划过程的输出信息前后一致，具有连续性，并符合第 11 节要求。

5 软件开发过程

本节讨论软件开发过程的目标和活动。这种软件开发过程根据软件规划过程（第 4 节）和软件开发计划（11．2 小节）规定进行申报。附件 A 和表 A – 2 是按照软件等级分类的软件开发过程的目标和输出信息。软件开发过程包括：

- 软件要求编制过程；
- 软件设计过程；
- 软件编码过程；
- 整合过程；

软件开发过程形成一个或多个层次的软件要求。高层次要求是直接通过系统要求和系统结构的分析形成的。一般来说，这些高层次要求在软件设计过程中进一步，因此，形成了一个或多个连续的、较低层次的要求。不过，如果源代码直接根据高层次要求生成的，那么，这个高层次要求要求也被视为低层次要求，而且低层次要求指南也同样适用。

软件结构的开发涉及到要对软件的结构做出决定。在软件设计过程中，要确定这一软件结构，同时编制出低层次要求。低层次要求是源代码无须进一步提供信息即可直接应用的软件要求。

每个软件开发过程都可以产生衍生要求。衍生要求是那些无法直接追溯到较高层次要求的要求。衍生要求的一个例子就是为那些用于所选出的特定程序计算机开发的处置里信息中断的软件。高层次要求可以包括衍生要求。衍生要求对安全方面的要求的影响由系统安全评估过程来确定。

5.1 软件要求编制过程

软件要求编制过程采用软件生命周期的输出信息来编制软件的高层次要求。这些高层次要求包括功能、性能、接口和有关安全方面的要求。

5.1.1 软件要求编制过程的目标

软件要求编制过程的目标是：

a．编制高层次要求。

b．衍生的高层次要求提供给系统安全评估过程。

5.1.2 软件要求编制过程的各项活动

输入到软件要求编制过程中的信息包括系统要求、来自系统生命周期过程的硬件接口和系统结构（如果在该要求中没有包含的话）以及来自软件规划过程的软件开发计划和软件要求标准。在满足所规划的转换条件之后，这些输入信息用来编制软件高层次要求。

这一过程的主要输出信息就是软件要求数据（参阅 11．9 小节）。

当软件要求编制过程的目标和与此有关的整体目标达到时，软件要求编制过程就完成了。这一过程的指导原则包括：

a．分配给软件的系统功能和接口要求应该进行分析，找出模糊不清、前后不一致和不确定状态；

b．输入到软件要求编制过程中被检测为不合适或不正确的信息应该作为一种反馈信息向输入信息源过程报告，以便于清除或纠正；

c．每个指定给软件的系统要求应该在高层次要求中加以规定；

d．满足指定给软件的系统要求以预防系统出现风险的高层次要求；

e．高层次要求应该符合软件要求标准规定，并且可以进行验证，保持一致；

f．在适用情况下，高层次要求应该用量值术语来说明；

g．高层次要求不应该描述设计或验证详细要求，规定的和已经证明的设计约束条件除外；

h．分配给软件的每一个系统要求都应该可以追溯到一个或多个软件高层次要求；
11．每一个高层次要求应该可以追溯到一个或多个系统要求，所得到的要求除外；
j．衍生的高层次要求应该提供给系统安全评估过程。

5.2 软件设计过程

软件高层次要求通过软件设计过程的一次或多次重复来开发软件的结构，编制可以用来应用源代码的低层次要求。

5.2.1 软件设计过程的目标

软件设计过程的目标是：
a．根据软件的高层次要求来开发软件的结构，编制低层次要求。
b．将所得到的低层次要求提供给系统安全评估过程。

5.2.2 软件设计过程的各项活动

软件设计过程的输入信息包括软件要求数据、软件开发计划和软件设计标准。当满足规划中的转换条件时，高层次要求就可以用于这一设计过程以开发软件结构，编制低层次要求。这可能涉及到一个或多个较低层次的要求。

这一过程的主要输出信息就是设计说明书（参阅 11．10 小节），其中包括软件结构和低层次要求。

当软件设计过程的目标和与此有关的整体目标达到时，软件设计过程就算完成了。这一过程的指导原则包括：
a．低层次要求和软件设计过程开发的软件结构应该符合设计标准，并且可以进行追溯、验证，前后保持一致。
b．衍生要求应该进行确定和分析，以保证较高层次的要求不受影响。
c．软件设计过程活动可能会在该软件中引发失效模式，相反也可能会阻止进入其它软件中。在软件设计中使用分区或其它结构措施可能会改变该软件一些组分的软件等级分配。在这些情况下，应该确定一些补充数据作为衍生要求，并提供给系统安全评估过程。
d．当有关安全方面的要求（例如看门狗定时器、合理性检验和跨通道比较）发出指令时，控制流和数据流应该得到检测。
e．失效状态响应应该符合安全方面的要求。
f. 在软件设计过程中，所检测到的不适当或不正确的输入信息应该作为反馈提供给系统生命周期过程、软件要求编制过程或者软件规划过程，以便于清除或进行纠正。

附注：*软件工程的当前状态不允许负载和安全目标的达到之间存在一种数量相关关系。在无法提供任何目标指南的情况下，软件设计过程应该避免复杂化，因为随着软件的复杂性的增大，要验证设计并显示软件的安全目标是否达到就变得非常困难。*

5.2.3 用户可修改软件

在讨论用户可修改软件软件开发之后便是两条相关的指导原则。一个可修改的软件组分是该软件可以被用户修改的那部分，而一个不可修改的软件组分则是指该软件不可以被用户进行修改的那个部分。

用户可修改软件可以随着复杂程度的不同而有所不同。其应用实例包括一个单存储位（用于在两个设备选项中选择其中之一）、一个电文信息表、一个可以进行编程、编译和链接（用于飞机保

养维护）的存储区。任何等级的软件都可能包括一个可修改的软件组分。
设计用户可修改软件的指导原则包括：

a. 非可修改软件组分应该防止受到用户可修改软件的影响，以防止在非可修改软件组分的安全运行过程中产生干扰。这一保护措施可以通过硬件、软件、用来进行修改的工具来实施，也可以通过这 3 种方式的组合来完成；
b. 申请人提供的手段应该表现为可以更改那些可修改软件组分的唯一手段。

5.3 软件编码过程

在软件编码过程中，源代码来自软件结构和低层次要求。

5.3.1 软件编码过程的目标

软件编码过程的目标是：
a．编制可以追溯、可以验证、前后一致和可正确实施的低层次要求。

5.3.2 软件编码过程的各项活动

软件编码过程输入信息有低层次要求和来自软件设计过程的软件结构要求和软件开发计划和软件代码标准。当满足规划的中的转换条件时，软件编码过程可以输入或再次输入。源代码可以根据软件结构和低层次要求通过这一过程来生成。

这一过程的主要成果就是源代码（11．11 小节）和目标代码。

当软件编码过程的目标和与此有关的整体目标达到时，软件编码过程就完成了。这一过程的指导原则包括：
a．源代码应该应用低层次要求，并符合软件结构要求；
b．源代码应该符合源代码的标准规范；
c．源代码应该可以追溯到设计说明书；
d．在软件编码过程中检测出的不恰当或不正确的输入信息应该作为反馈信息提供给软件要求编制过程、软件设计过程或软件规划过程，以便于进行清除或纠正。

5.4 整合过程

特定程序计算机、源代码和来自软件编码过程的目标代码与整合过程中的链接和加载数据（11．16 小节）一起使用，以开发经过整合的机载系统或设备。

5.4.1 整合过程的目标

整合过程的目标是：
a．可执行目标代码加载目标硬件用于硬件/软件整合。

5.4.2 整合过程的各项活动

整合过程由软件的整合和硬件/软件的整合组成。当满足规划的中的转换条件时，整合过程就可以进行输入或再次输入。整合过程的输入信息是来自软件设计过程的软件结构和源代码。

这一整合过程的输出信息就是可执行目标代码（参阅 11．12 小节的规定）和链接、加载数据。当整合过程的目标和与此有关的整体目标达到时，这一整合过程就完成了。这一过程的指导原则包括：
a．可执行目标代码应该根据源代码、链接程序和加载数据来生成；
b．软件应该加载到特定程序计算机中进行硬件/软件整合；

c．在这一整合过程中所检测出的不恰当或不正确的输入信息应该作为反馈信息提供给软件要求编制过程、软件设计过程或软件规划过程，以便于进行清除或纠正。

5.4.3 整合考虑因素

下面是无效代码和软件补丁的一些考虑因素。一个机载系统和设备在设计中可以包括几种配置，并不是所有配置都适合于每一个应用场合。这可能导致无法执行的无效代码或无用的数据。这与术语词汇中定义的死代码有所不同，在段落 6．4．4．3 中我们将会详细讨论。无效代码和软件补丁指导原则包括：

a．应该提供证据证明无效代码在非指定应用环境下无法使用。由于出现异常系统状态，无效代码的无意中启动与有效代码的无意中启动相同。

b．处理无效代码的方法应该符合软件计划要求。

c．软件补丁不应该用于交付经过认证的飞机或发动机使用的软件中，来实施要求或结构的更改，或者软件验证过程活动结果所需要的更改。这些软件补丁可以根据一种有限原则进行应用，以解决软件开发环境中已知的缺陷，例如已知的一些编译程序问题。

d．在使用软件补丁时，应该确定存在下列情况：

（1）确认软件配置管理过程可以有效地跟踪所使用软件补丁；

（2）通过回归分析来证明，这种软件补丁满足正常情况下开发的这种软件所规定的所有目标；

（3）在“软件完成情况综述”中证明有必要使用一个软件补丁。

5.5 可追溯性

可追溯性指导包括下列内容：

a．应该提供系统要求和软件要求之间的可追溯性，以便于检验系统要求是否全部完成，并可以看到所得出的相关要求。

b．应该提供低层次要求和高层次要求之间的可追溯性，这样可以看到所得出的相关要求和软件设计过程中所做出的体系结构设计决定，并可以对高层次要求的整个完成情况进行检验。

应该提供源代码与低层次要求之间的可追溯性，这样就可以检验是否缺乏未经编制的源代码，并对低层次要求的整个完成情况进行检验。

6 软件验证过程

本节讨论软件验证过程的目标和活动。软件验证是软件开发过程和软件验证过程结果的一次技术评估。软件验证过程根据软件规划过程（第 4 节）和软件验证计划（11．3 小节）规定来进行申报。

验证不是简单的测试。一般来说，测试无法显示软件中不存在差错。我们正在讨论的软件验证过程的目标一般是由审核、分析和测试组成的综合过程，所以，下面的几个小节我们使用“验证”一词而是使用“测试”。

附件 A 中的表 A - 3 ~ A - 7 包含按照软件等级分类的软件验证过程的目标和输出信息的总结。

附注：对于较低的软件等级来说，无须过多强调下列项目：

- 低层次要求的验证；
- 软件结构的验证；
- 测试覆盖程度；
- 验证过程的控制；
- 软件验证过程活动的独立性；
- 软件验证过程活动的重复进行，也就是说存在多次软件验证活动，每一次都可能检测出同类错误；
- 坚固性测试；
- 对错误预防或检测产生间接作用的验证活动，例如符合软件开发标准要求。

6.1 软件验证过程的目标

软件验证过程的目标就是要检测并报告可能在软件开发过程中带来的一些差错。清除这些差错是软件开发过程的一项活动。软件验证过程的总体目标就是要验证：

a．指定给软件的系统要求已经编入满足这些系统要求的软件的高层次要求中；

b．高层次要求已经编入满足高层次要求的软件结构和低层次要求中。如果在高层次要求和低层次要求之间编制一个或多个层次的软件要求，那么就应该编制连续层次的软件要求，使得每个连续的低层次要求满足其高层次要求。如果代码直接根据高层次要求生成的，那么，这一目标并不适用；

c．软件结构和低层次要求已经编入满足这种低层次要求和软件结构的源代码中。

d．可执行目标代码满足这一软件要求；

e．满足这一目标的手段对于该软件等级来说在技术是正确和完整的。

6.2 软件验证过程的活动

软件验证过程的目标通过一个审核、分析、编制校验数据和条件及步骤并在随后执行这些测试步骤这一综合过程来达到。审核和分析可以提供准确性和完整性以及软件要求、软件结构和源代码可验证性的评估。编制校验数据和条件的编制可以提供软件要求内部一致性和完整性的进一步评估。测试步骤的执行则显示符合这一软件要求。

软件验证过程的输入信息包括系统要求、软件要求和软件结构、可追溯性数据、源代码、可执行目标代码和软件验证计划。

软件验证过程的输出信息记录在软件校验数据和条件和步骤（11．3 小节）和软件验证结果（11．14 小节）中。

这些要求一旦在软件中实施的话，其可验证要求就成为追加要求或为软件开发过程制定的约束条件。

这种软件验证过程在软件要求的实施和这些软件要求验证之间提供响应的可追溯性。

- 软件要求和校验数据和条件之间的可追溯性是通过根据要求制定的覆盖分析来完成的；

• 代码结构和校验数据和条件之间的可追溯性则是通过结构覆盖分析来完成的。

软件验证活动的指导原则包括：

a．应该验证高层次要求和这些高层次要求的可追溯性；

b．可追溯性结果分析根据要求制定的结构覆盖分析应该表明每个软件要求都可以追溯到应用这一要求的代码、审核、分析和验证它的校验数据和条件；

c．如果所测试的代码与机载软件代码不同，那么，这些不同之处应该加以规定和验证；

d．如果不可能通过在现实测试环境中应用该软件来验证特定的软件要求的话，应该提供其它手段，其满足软件验证过程目标的验证应在软件验证计划或软件验证结果中加以规定；

e．在软件验证过程中所发现的缺陷和差错应该报告给软件开发过程，以便进行清除和纠正。

6.3 软件的审核与分析

对软件开发过程和软件验证过程所得出的结果进行审核与分析。审核与分析之间的区别在于分析提供可重复的正确性证据，而审核则提供一种正确性量化评估。审核可以由一个过程的检验（辅以审核表等类似清单）组成，而分析则要详细检查功能、完成情况、可追溯性和一个软件组分的安全**含义**以及它在既载系统或设备中与其它软件组分的关系。

6.3.1 高层次要求的审核与分析

这些审核与分析的目标就是要检测和报告可能在软件要求编制过程中带来的一些软件要求方面的错误。这些审核与分析确认高层次要求符合这些目标要求：

a．符合系统要求：这一目标就是要保证软件要完成的系统功能被确定下来，并保证软件的高层次要求满足了该系统的功能、性能和与安全有关的要求，而且衍生要求及其存在的设计基础已经得到适当规定。

b．准确性和一致性：这一目标就是要保证每一个高层次要求都是准确的、清楚的，而且足够详尽，并保证这些要求彼此一致，不存在矛盾冲突的地方。

c．与特定程序计算机兼容：这一目标就是要保证这些高层次要求和特定程序计算机硬件/软件（尤其是系统响应时间和输入/输出硬件）特征之间不存在任何冲突。

d．可验证性：这一目标就是要保证每一个高层次要求都可以进行验证。

e．符合标准要求：这一目标就是要保证在软件要求编制过程中软件要求标准都得到遵循，而且保证与标准之间存在的偏差都找到了相应的依据。

f．可追溯性：这一目标就是要保证指定给该软件的系统的功能、性能和与安全有关的要求已经编入软件的高层次要求。

g．算法方面：这一目标就是要保证所申报的算法的准确性和运行状态，尤其是在断续函数区域。

6.3.2 低层次要求的审核与分析

这些审核与分析的目标就是要检测和报告可能在软件设计过程中带来的一些软件要求方面的错误。这些审核与分析确认低层次要求符合这些目标要求：

a．符合高层次要求：这一目标就是要保证软件的低层次要求满足高层次要求，而且衍生要求及其存在的设计基础已经得到适当规定。

b．准确性和一致性：这一目标就是要保证每一个低层次要求都是准确的、清楚的，并保证这些低层次要求彼此一致，不存在矛盾冲突的地方。

c．与特定程序计算机兼容：这一目标就是要保证这些高层次要求和特定程序计算机硬件/软件（尤其是总线安装、系统响应时间和输入/输出硬件）特征之间不存在任何冲突。

d．可验证性：这一目标就是要保证每一个低层次要求都可以进行验证。

e．符合标准要求：这一目标就是要保证在软件要求编制过程中软件设计标准都得到遵循，而且保证与标准之间存在的偏差都找到了相应的依据。
f．可追溯性：这一目标就是要保证高层次要求和衍生要求都已经编入软件的低层次要求。
g．算法方面：这一目标就是要保证所申报的算法的准确性和运行状态，尤其是在断续函数区域。

6.3.3 软件结构的审核与分析

这些审核与分析的目标就是要检测和报告可能在软件结构开发过程中带来的一些软件要求方面的错误。这些审核与分析确认其软件结构符合这些目标要求：
a．与高层次要求的兼容性：这一目标就是要保证软件结构与高层次要求不存在任何冲突，尤其是那些确保系统完整性方面的功能，例如分区方案。
b．一致性：这一目标就是要保证软件结构每一个组成部分之间存在一种正确的相互关系。这一关系借助于数据流和控制流而存在。
c．与特定程序计算机兼容：这一目标就是要保证这些软件结构和特定程序计算机硬件/软件特征（尤其是初始化、操作异常、同步和中断等）之间不存在任何冲突。
d．可验证性：这一目标就是要保证软件结构可以进行验证，例如不存在任何无限循环算法。
e．符合标准要求：这一目标就是要保证在软件设计过程中软件设计标准都得到遵循，而且保证与标准之间存在的偏差都找到了相应的依据，尤其是可能会与系统安全目标不相符的复杂性限制和设计约束条件。
f．分区的完整性：这一目标就是要确保分区违规现象得到有效预防或隔离。

6.3.4 源代码的审核与分析

这些审核与分析的目标就是要检测和报告可能在软件编码过程中带来的一些软件要求方面的错误。这些审核与分析确认软件编码输出信息的准确、完整，并且可以进行验证。其主要关注的问题包括与软件要求、软件结构有关的代码的正确性，以及与软件代码标准的符合性。这些审核与分析一般仅限于源代码。其审核与分析的问题包括：
a．符合低层次要求：这一目标就是要保证源代码在软件的低层次要求方面的准确性和完整性，并保证任何源代码都不会应用不成文的功能。
b．符合软件结构：这一目标就是要保证源代码与软件结构中规定的控制流和数据流相匹配。
c．可验证性：这一目标就是要保证源代码不含任何无法进行验证的说明和结构，并保证该代码不必为了测试而进行更改。
d．符合标准要求：这一目标就是要保证在软件代码编制过程中软件代码标准都得到遵循，尤其是可能与系统安全目标一致的复杂性限制和代码约束条件。复杂性包括软件组成部分、控制结构的嵌套等级、逻辑或数值表述的复杂性之间的耦合程度。这一分析还保证与标准之间存在的偏差都找到了相应的依据。
e．可追溯性：这一目标就是要保证软件的低层次要求已经编入源代码。
f．准确性和一致性：这一目标就是要确定源代码的正确性和一致性，包括迭式存储器的用法、固定点算法超值和算式解、资源争用、最坏情况实施定时、例外情况的处置、未初始化变量或常数的使用、未使用的变量或常数以及由于任务冲突或信息中断冲突而导致数据漏洞百出。

6.3.5 整合过程输出信息的审核与分析

这种审核与分析的目标就是保证整合过程的结果（即输出信息）完整、正确。这一过程可以通过详细检查链接程序和加载数据以及存储器的映象来完成。其审核与分析的问题包括：
a．硬件地址不正确；

b．存储器重叠；
c．软件各个组分发生丢失。

6.3.6 校验数据和条件、测试步骤和测试结果的审核与分析

这种审核与分析的目标就是保证代码的测试已经编制，并且保证其完成过程的准确而完整。其审核与分析的问题包括：

a．校验数据和条件：校验数据和条件的验证参阅段落 6．4．4；
b．测试步骤：其目标就是验证校验数据和条件已经准确地编入测试步骤和预测结果中；
c．测试结果：其目标就是保证测试结果正确无误，实际结果与预测结果之间的偏差得到解释。

6.4 软件测试过程

机载软件的测试有两个补充目标。一个目标就是要说明该软件满足其要求。另一个目标就是要确信在系统安全评估过程中确定的导致各种无法接受的失效状态的误差已经消除。

图 6 – 1 是软件测试过程的示意图。图中的 3 种测试的目标列述如下：

• 硬件/软件集成测试：其目标就是要检验特定程序计算机环境中的软件运行是否正确。
• 软件整合测试：其目标就是要验证软件要求和各个组成部分之间的相互关系，并验证软件要求和软件各个组成部分在软件体系结构中的实施情况。
• 低层次测试：其目标就是要验证软件低层次要求的实施情况。

附注：*如果校验数据和条件及其相应的测试步骤是为硬件/软件集成测试或软件整合测试编制和进行的，且满足根据要求确定的有效范围和体系结构有效范围，那么就没有必要在低层次测试中重复这一测试。由于所测试的总体函数关系量的减小，从标称意义上来说，用等效的低层次测试取代高层次测试其有效性可能会降低。*

为了满足软件测试目标：

a. 校验数据和条件基本上应根据软件要求来进行；
b. 应该建立校验数据和条件以检验其功能是否正确，并建立揭示潜在误差的相关条件；
c. 软件要求有效范围分析应确定没有测试的软件要求；
d. 体系结构有效范围分析应确定没有实际运行的软件体系结构。

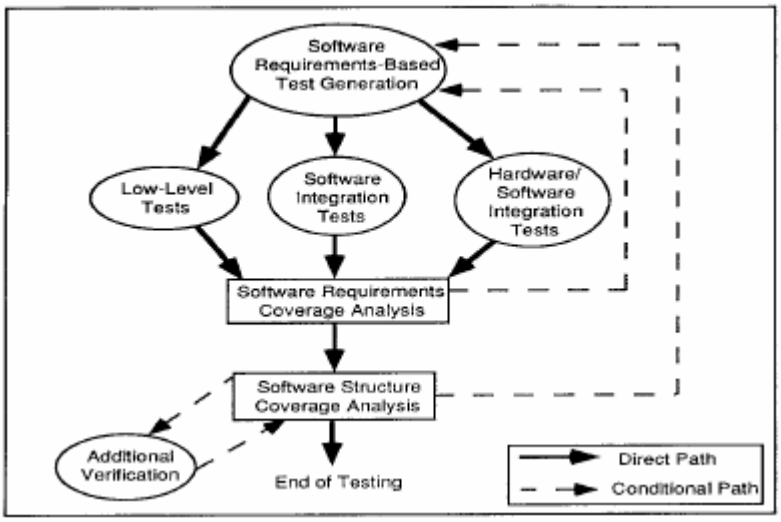

图 6 – 1 软件测试过程

6.4.1 测试环境

需要建立一个以上的测试环境来满足软件测试目标。一个优越的测试环境包含有加载于特定程序计算机中并在特定程序计算机那高清晰度的模拟中进行测试的软件。

附注：*在许多情况下，根据要求确定的有效范围和体系结构有效范围只有在对测试输入信息和代码比在一个完全一体化的环境下更为精确控制和监测的情况下才能达到。对于一个在功能上与其它软件组成部分彼此独立的较小的软件组成部分来说，更需要完成这一测试。*

如果测试采用一个特定程序计算机仿真程序或主计算机模拟装置来完成的话，那么也可以授予相应的认证证书。测试环境的指导原则包括：

a．所选择的测试应该集成的特定程序计算机环境中完成的，因为一些误差只有在这种环境中才能被检测出来。

6.4.2 根据要求确定的校验数据和条件的选择

这里强调的是根据要求所进行的测试，因为人们已经发现这一战略在寻找错误方面最为有效。根据要求确定的校验数据和条件的选择指导原则包括：

a．为了实现软件测试目标，应该包括两种校验数据和条件：正常范围的校验数据和条件和异常范围的校验数据和条件；

b．应该根据软件要求和软件开发过程中固有的错误源编制特定的校验数据和条件。

6.4.2.1 正常范围的校验数据和条件

正常范围的校验数据和条件的目标就是要表明该软件对异常输入信息和条件的反应能力。正常范围的校验数据和条件包括：

a．实际和整数变量应该用有效的等效类别选择和区间值来运算；

b．对于与时间有关的功能（例如滤波器、积分电路和延时）来说，应该完成代码的多次反复，以便检查文本中的功能特性；

c．对于状态转换来说，应该编制相应的校验数据和条件，以便尽可能在正常情况下实现状态转换；

d．对于用逻辑公式表示的软件要求来说，应采用正常范围的校验数据和条件来验证变量的用法以及布尔运算子。

附注：*一个方法就是测试这些变量的所有组合。对于复杂表达式，这一方法并不适用，因为它要求大量的校验数据和条件。采用一个不同的对策来保证所要求的覆盖范围。例如，对等级 A 来说，可以通过分析或审核来验证布尔运算子，为弥补这一不足，可以确定校验数据和条件来提供经过修改的条件/判定覆盖范围。*

6.4.2.2 坚固性校验数据和条件

坚固性校验数据和条件的目标就是要表明该软件对异常输入信息和条件的反应能力。坚固性校验数据和条件包括：

a．实际和整数变量应该用无效值的等效类别选择来运算；

b．系统初始化应该在异常状态发生过程中进行；

c．应该确定外来数据可能的失效模式，尤其是那些来自外部系统的复杂的、数字式数据串；

d．对于其环数量为一个计算得出的值的环来说，应该编制其校验数据和条件，以便计算出超范围环的数值，并由此说明与环有关的代码的坚固性；

e．应该完成一次检查，以确保在帧象周期内其超限保护装置反应适当；
f．对于与时间有关的功能（例如滤波器、积分电路和延时）来说，应该为算法超值保护装置编制相应的校验数据和条件；
g．对于状态转换来说，应该编制相应的校验数据和条件，以便于促进为软件要求所不允许的状态转换。

6.4.3 根据要求确定的测试方法

根据要求确定的测试方法由三种方法组成：即根据要求确定的硬件/软件整合测试、根据要求确定的软件整合测试和根据要求确定的低层次测试。除了根据要求确定的硬件/软件整合测试外，这些方法并不要规定特定的测试环境或对策。其指导原则如下：

a．根据要求确定的硬件/软件整合测试：这一测试方法应该重点放在与在特定程序计算机环境内运行的软件有关的错误源以及高层次功能。根据要求确定的硬件/软件整合测试的目标就是要保证特定程序计算机中的软件将会满足高层次要求。通过测试方法所显示的典型错误包括：
- 中断处置不正确；
- 无法满足执行时间的要求；
- 软件对硬件瞬时值或硬件失效的响应不当，例如启动顺序、瞬时输入负荷和输入功率瞬时值；
- 数据总线和其它资源争用问题，例如存储器映象；
- 内部测试无法检测出失效；
- 硬件/软件接口误差
- 反馈环运行状态不正确；
- 存储器管理硬件或处于软件控制的其它硬件设备控制不当；
- 迭式存储器出现超值现象；
- 用于确认现场安装软件正确性和兼容性的结构的运行不正确；
- 违反软件分区规定。

b．根据要求确定的软件整合测试：这一测试方法应该重点放在软件要求之间的相互关系，以及通过软件结构来应用这些要求。根据要求确定的软件整合测试的目标就是要保证软件的各个组分彼此相互作用正确无误并符合软件要求和软件结构。随着校验数据和条件范围的相应扩展，代码组成部分的连续整合，其软件要求的范围也随之扩展，这样就可以完成这一整合测试。通过测试方法所显示的典型错误包括：
- 变量和常量的初始化不正确；
- 参数偶然出错；
- 数据不可靠，尤其是全局数据；
- 端对端数值消解不当；
- 过程发生的顺序和运行不当。

c．根据要求确定的低层次测试：这一测试方法应该重点说明每个软件组分符合其低层次要求。根据要求确定的低层次测试的目标就是要保证这些软件组分满足其低层次要求。
通过测试方法所显示的典型错误包括：
- 算法无法满足软件要求；
- 循环运算不当；
- 逻辑判定不正确；
- 无法正确地处理输入条件的合理组合；

- 丢失的或不可靠的输入数据反应不正确；
- 例外情况（例如算法错误或违反阵列极限范围）处理不当；
- 计算程序不正确；
- 算法精确度不够或运算不当。

6.4.4 测试覆盖分析

测试覆盖分析过程分两个步骤来完成，它涉及到根据软件要求确定的覆盖分析和结构覆盖分析。第一步就是分析与软件要求有关的校验数据和条件，以确认所选择的校验数据和条件是否符合规定的规范条件；第二步就是确认根据软件要求确定的测试步骤运用了代码结构。结构覆盖分析可能不符合规定的条件。本文为作为死区的种种情况提供了一些补充认证指南（参阅段落 6．4．4．3）。

6.4.4.1 根据软件要求规定的测试覆盖分析

这一分析要达到的目标就是要确定根据软件要求规定所进行的测试如何验证软件要求的应用情况。这一分析可以揭示需要根据软件要求添加校验数据和条件。根据要求规定的测试覆盖分析应该显示：

a．每一个软件要求都编制有校验数据和条件；

b．校验数据和条件符合正常测试和坚固性测试的规范条件（参阅 6．4．2 中的规定）。

6.4.4.2 结构覆盖的分析

这一分析要达到的目标就是要确定哪一个代码结构没有根据软件要求所确定的测试步骤来实施。根据软件要求所确定的校验数据和条件可能没有完全运用代码结构，因此，应完成结构覆盖分析，并形成追加验证来提供结构覆盖。其指导原则包括：

a．这一分析应该确认合乎软件等级的结构覆盖程度；

b．这种结构覆盖分析可以在源代码上进行，除非软件等级为 A，编译程序生成一个无法直接追溯到源代码语句的目标代码。然后应该在这种目标代码上完成追加验证，以确认这一生成的代码顺序的正确性。目标代码中由编译程序生成的阵列上下限检查是无法直接追溯到源代码的目标代码分析的一个实例；

c．这一分析应该确认代码各个组成部分之间的数据耦合和控制耦合。

6.4.4.3 体系结构有效范围的分析解决

体系结构有效范围分析可以揭示代码体系结构在测试过程中没有开始运行。其数值的消解将要求完成补充软件的验证过程。这一没有运行的代码体系结构可能是由于下列因素导致的结果：

a. 根据要求确定的校验数据和条件或测试步骤中存在不足之处：该校验数据和条件应该得到进一步补充或改变测试步骤，以提供缺失的有效范围。用来完成根据要求确定的有效范围分析的方法可能需要进行相应的审核。
b. 软件要求不合适：应该修改软件要求，编制补充的校验数据和条件并运行相应的测试步骤。
c. 死代码：这种代码应该去除，并进行分析以评估其影响，再评估是否需要重新进行验证。
d. 代码无效：对于不是指定运行于飞机或发动机内任何配置中的无效代码来说，将分析和测试结合起来就可以显示可能使这一代码无意中运行的那些方法或措施是可以预防、隔离或消除的。对于那种之内在特定程序计算机环境的某些配置下运行的无效代码来说，应该建立这一代码正常运行所必须的工作配置条件，并编制补充校验数据和条件和测试步骤，以达到所要求的有效范围目标。

7 软件配置管理过程

本节讨论软件配置管理过程（SCM 过程）的目标和活动。软件配置管理过程按照软件规划过程（第 4 节）和软件配置管理计划（11．4 小节）规定进行申报。软件配置管理过程的输出信息记录在“软件配置管理记录”（ 11．18 小节）或其它软件生命周期数据中。
附件 A 中的表 A－8 是软件配置管理过程目标和活动的总结。

7.1 软件配置管理过程目标

软件配置管理过程随同其它软件生命周期过程一起可以帮助实现下列总体目标：
a．在整个软件生命周期内提供一个确定和受控的软件配置；
b．使软件制造商可以连续地复制可执行目标代码，或者在需要调查或修改时重新生成；
c．在整个软件生命周期内提供过程输入和输出信息控制，以保证过程活动的一致性和可重复性；
d．借助于配置项目的控制和建立的原始资料为审核、评估状态和更改控制提供一个已知点；
e．提供相应的控制手段，以保证问题都受到关注，更改已经记录、批准和实施；
f．通过软件生命周期输出信息的控制提供软件得到批准的证据；
g．有助于评估软件产品符合要求；
h．确保安全物理存档，并保证配置项目处于恢复和受控状态。
软件配置管理（SCM）目标独立于软件等级。不过，根据应用于数据的软件配置管理控制（参阅 7．3 小节）可能会出现两类软件生命周期数据。

7.2 软件配置管理过程活动

软件配置管理过程活动包括配置识别、更改控制、原始资料的建立、软件产品（包括相关的软件生命周期数据）的存档。下列指南确定每个软件配置管理过程活动要达到的目标。当软件产品为认证主管部门接受时，这一软件配置管理过程并没有停止，而是在机载系统或设备的整个寿命周期内继续下去。

7.2.1 软件配置的标识

软件配置标识活动的目标就是要清楚地标出配置项目（及其连续版本），这样就为配置项目的控制和参照建立一个必要的基础。其指导原则包括：
a．应该为软件生命周期数据建立软件配置识别；
b．应该为每一个软件配置项目、为配置项目的每一个独立控制的组成部分以及构成一个软件产品的配置项目组合建立软件配置识别；
c．在更改控制和可追溯性数据记录实施之前，配置项目应该做到配置可以识别；
d．在该配置项目用于其它软件生命周期过程，被其它软件生命周期数据参照，或用于软件制造商进行软件制造或软件安装公司之前，其配置项目应该做到配置可以识别；
e．如果该软件产品标识无法通过物理检查（例如件号铭牌检查）来确定，那么，其可执行目标代码应该包含可以通过机载系统或设备的其它部分访问的配置识别。这同样也适用于现场安装软件（参阅 2．5 小节）。

7.2.2 原始资料和可追溯性

建立原始资料的目标就是要为详细了解软件生命周期过程活动，参照配置项目控制以及配置项目之间的可追溯性确定一个基础信息库。其指导原则包括：
a．应该为用于认证的软件产品建立原始资料。（可以建立中间阶段的原始资料，以帮助控制软件生

命周期过程活动）；
b．应该为软件产品建立软件产品原始资料，并在“软件配置索引”中加以规定（参阅 11．16 小节）。

附注：*软件产品原始资料中并没有包含用户可修改软件，其相关的保护和上下限组分除外。因此，可以对用户可修改软件进行修改，而不影响软件产品原始资料的配置标识。*
c．应该在受控软件库中建立原始资料（物理、电子或其它类），以保证其完整性。一旦建立原始资料，它就应该尽量保持不变；
d．随后应该是更改控制活动，以便根据已经建立的原始资料来编制一个衍生原始资料；
e．如果软件生命周期过程活动或与前一个原始资料的编制有关的数据要获得合格证的话，那么，一个原始资料应该可以追溯到形成这一原始资料的上游原始资料；
f．如果软件生命周期过程活动或与前一个配置项目的编制有关的数据要获得合格证的话，那么，一个配置项目应该可以追溯到形成这一配置项目的上游配置项目；
g．一个原始资料或配置项目应该可以追溯到它标识的输出信息，或者与它有关的过程。

7.2.3 问题报告、跟踪和纠正措施

编制问题报告、跟踪和纠正措施的目标就是要记录过程与软件计划和标准的非符合性项目，并记录软件生命周期过程的输出信息中缺陷，记录软件产品的异常特性，并确保这些问题的解决。
附注：*软件生命周期过程和软件产品问题可以记录在独立的问题报告系统中。*
其指导原则包括：
a．问题报告应该编制描述软件生命周期过程与软件计划的不符合性项目、输出信息中的缺陷活软件产品的异常特性以及所采取的纠正措施，参阅 11．17 小节；
b．问题报告应该提供受影响的配置项目的配置标识或受影响的过程活动、问题报告的状态报告和问题报告的批准和关闭；
c．要求软件产品纠正措施或软件生命周期过程输出信息的问题报告应该援引相应的更改控制活动。

附注：*问题报告系统和更改控制密切相关。*

7.2.4 更改控制

更改控制活动的目标就是在整个软件生命周期过程中提供更改的记录、评估、解决和批准各项活动。其指导原则包括：
a．更改控制应该通过提供预防更改措施保持配置项目和原始资料的完整性；
b．更改控制应该保证对配置项目的任何更改都要求对其配置标识进行更改；
c．在实施更改控制之下，原始资料的更改和配置项目的更改都应该进行记录、批准和跟踪。将问题报告与更改控制联系起来，因为所报告问题的解决可以导致配置项目和原始资料的更改；

附注：*人们普遍认为，更改控制的尽早实施有助于控制和管理软件生命周期过程活动。*
d．软件更改应该追溯到其源头以及从更改影响其输出信息时点起的软件生命周期过程。例如在硬件/软件整合时发现的错误（源于设计错误）应该带来设计纠正、代码纠正以及相关的集成过程活动的重复进行；
e．在整个更改活动中，受到更改影响的软件生命周期数据应该升级更新，其记录应该保留，用于更改控制活动。
采用更改审核活动将有助于这一更改控制活动的实施。

7.2.5 更改审核

更改审核活动的目标就是保证未经批准的问题和更改得到评估、批准，经过批准的问题和更改得到有效实施，并通过在软件规划过程中确定的问题报告和更改控制方法将反馈提供给受到影响的各个过程。这种更改审核活动应该包括：

a．确认受到影响的配置项目是已经经过标识的配置项目；

b．用发送到系统安全评估过程的反馈来评估它对安全方面要求的影响；

c．问题或更改的评估，决定要采取的措施；

d．问题报告或更改的影响以及对受影响过程的判定。

7.2.6 配置状态统计

配置状态统计活动的目标就是要在配置标识、原始资料、问题报告和更改控制方面为软件生命周期过程的配置管理提供数据。配置状态统计活动应该包括：

a．报告配置项目的标识、原始资料标识、问题报告状态、更改历史和发布情况；

b．要保存的数据的定义和这一数据记录和报告状态手段的定义。

7.2.7 存档、检索和发布

存档、检索和发布的目标就是要保证软件产品有关的软件生命周期数据可以在需要复制、再生、重新测试或修改软件产品时进行检索。这一发布活动的目标除了保证存档和可以检索之外，就是保证只使用经过授权的软件，尤其是对软件制造来说。其指导原则包括：

a．与软件产品有关的软件生命周期数据应该可以从许可的信息来源（例如开发组织或公司）处进行检索；

b．应该通过下列措施建立相应的步骤，以保证所存储的数据的完整性（不管采用什么样的存储媒介）；

（1）保证未作任何未经授权的更改；

（2）选择存储媒介，以减少再生过程中出现错误或退化；

（3）使用和/或更新存档的数据，其频率与媒介的保存受命相一致；

（4）以独立的实体档案形式保存副件，以避免在出现灾难性事件时减少丢失的风险；

c．其复制过程应该进行验证，以保证生成准确的副件，而且制定了复制的步骤，以保证可执行目标代码被正确无误地复制；

d．配置项目在用于软件制造和授权之前，应该进行标识和发布。起码来说，加装在机载系统或设备的软件产品的各个组成部分（包括可执行代码以及加载软件的相关媒介）应该发布；

附注：*对于确定那些已经批准的用于机载系统或设备的软件来说，其发布一般也在要求之列。该数据的定义已经超出本文件的适用范围，不过，可以包括软件配置指标。*

e．应该建立数据保存程序，以满足适航性要求，以便于软件修改。

附注：*其它数据保存考虑因素可能包括诸如业务需求和未来认证主管部门的审核等项目，这已经超出了本文件的适用范围。*

7.2.8 软件加载控制

软件加载控制活动的目标就是要保证可执行目标代码用适当的措施加载到机载系统或设备中。软件加载控制是指编程的说明书和数据从一个主存储器设备转换到机载系统或设备中的过程。例如，这些方法可能包括（要取决认证主管部门的同意）工厂存储器设备的安装或者使用一个现场加载装置在现场将机载系统或设备重新编程。无论采用哪一种方法，软件加载控制都应该包括：

a．零件编号步骤和识别软件配置（指定安装在机载系统或设备中）的媒介标识；

b．无论这种软件是否作为一种最终项目交付使用，还是安装在机载系统或设备中交付使用，其记录都应该保存，以便确认软件与机载系统或设备的兼容性。
附注：*有关软件加载控制的其它指导原则参阅2．5小节*。

7.2.9 软件生命周期环境控制

软件生命周期环境控制的目标就是保证用来生产软件的工具得到标识、控制，并且可以检索。软件生命周期环境工具通过软件规划过程来确定，并在“软件生命周期环境配置指标”（11．15小节）中加以规定。其指导原则包括：
a．应该为用来开发、控制、制造、验证和加载软件的工具的可执行目标代码提供配置标识；
b．控制合格工具的软件配置管理过程（SCM）应该符合与第1控制类别（CC 1）或第2控制类别（CC 2）数据（7．3小节）有关的目标，参阅段落12．2．3中b项规定。
c．除非适用段落7．2．9中b项规定，用来控制工具（用来制造和加载软件）可执行目标代码的软件配置管理过程至少应该符合与第2种控制类别（CC 2）数据有关的目标。

7.3 数据控制类别

软件生命周期数据可以划分为两类：控制类别1（CC1）和控制类别2（CC2）。这些类别与置于数据上的配置管理控制符密切相关。表7 - 1规定了与每个控制类别有关的一系列SCM过程目标。其中“•”表示适用于该类型中的软件生命周期目标。
附件A中的那些表按照软件等级规定了每个软件生命周期数据项目的控制类别。数据控制类别的指导包括下列两种：
列为CC类的软件生命周期数据的软件配置管理过程目标应该按照表7 – 1规定进行申请。
列为C类的软件生命周期数据的软件配置管理过程目标至少应该按照表7 – 1规定进行申请。

表7 – 1 与CC 1和CC 2 有关的软件配置管理过程目标

SCM过程目标	参照章节	CC 1	CC 2
配置标识	7．2．1	•	•
原始资料	7．2．2 a、b、c、d、e	•	
可追溯性	7．2．2 f、g	•	•
问题报告的编制	7．2．3	•	
设计更改的控制-整合与标识	7．2．4 a、b	•	•
设计更改的控制-追溯	7．2．4 c、d、e	•	
设计更改的审核	7．2．5	•	
配置状态统计报表	7．2．6	•	
检索	7．2．7 a	•	•
未经授权的更改预防措施	7．2．5 b (1)	•	•
媒介选择、刷新和复制	7．2．7 b (2)、(3)、(4)、c	•	
发布	7．2．7 d	•	
数据保留	7．2．7 e	•	•

8 软件质量保证过程

本节讨论软件质量保证（SQA）过程的目标和活动。软件质量保证（SQA）过程按照软件规划过程（第 4 节）和软件质量保证计划（11．5 小节）规定进行申报。软件质量保证（SQA）过程活动的输出信息记录在软件质量保证记录（11．19 小节）或其它软件生命周期数据中。
软件质量保证（SQA）过程对软件生命周期过程及其输出信息进行评估，以确保其相应的目标得以实现，并保证所存在的缺陷被检测、评估、跟踪和解决，保证软件产品和软件生命周期数据符合认证要求。

8.1 软件质量保证过程的目标

软件质量保证过程的目标可以通过保证这些过程是在符合经过批准的软件计划和标准下完成的，这就使认证主管部门确信软件生命周期过程生成符合其要求的软件。
软件质量保证过程的目标就是要保证完成下列各项任务：
a．软件开发和整合过程符合经过批准的软件计划和标准。
b．软件生命周期过程的转换条件得到满足。
c．完成软件产品的符合性审核。
以软件等级划分的目标适用性参阅附件 A 表 A - 9 中的详细规定。

8.2 软件质量保证过程的具体活动

为了达到软件质量保证过程的目标：
a．软件质量保证过程应该在软件生命周期过程中发挥积极作用，并行使其权限、责任和独立性完成软件质量保证过程，以确保软件质量保证过程的目标得到实现。
b．软件质量保证过程应该保证软件计划和标准在编制和审核过程中前后一致，具有连贯性。
c．软件质量保证过程应该保证软件生命周期过程符合已经批准的软件计划和标准要求。
d．在软件生命周期过程中，软件质量保证过程应该包括软件开发和整合过程的中审核，以保证得到下列各项目标：
（1）提供符合 4．2 小节中规定的软件计划。
（2）检测并记录与软件计划和标准所存在的差距，然后进行评估、跟踪并消除这些偏差。
附注：*过程偏差的早期检测是一个行之有效的办法。它有助于人们有效地达到软件生命周期过程所确定的目标。*
（3）记录得到允许的偏差。
（4）已经提供符合软件计划中规定的软件开发环境。
（5）问题报告、跟踪和纠正措施过程符合“软件配置管理计划”要求。
（6）通过正在进行的系统安全评估过程提供给软件生命周期过程的输入信息已经经过相应的处理。

附注：*可以对软件生命周期过程活动进行监测，以保证这些活动处于受控状态。*

软件质量保证（SQA）过程应该保证软件生命周期过程的转换条件在符合所批准的软件计划方面的要求得到满足。
软件质量保证（SQA）过程应该保证软件生命周期数据根据 7．3 小节和附件 A 的表中确定的控制类别进行控制。
g．作为认证申请一个部分所提交的软件产品送交之前，应该完成软件的符合性审核。
h．软件质量保证（SQA）应该生成相应的软件质量保证过程活动记录（11．19 小节），包括审

核结果以及作为认证申请一个部分所提交的每个软件产品的软件符合性审核完成情况的证据。

8.3 软件符合性审核

软件符合性审核的目标就是要得到相应的质量保证，对于一个作为认证申请的一部分提交的软件产品来说，就是要保证整个软件生命周期过程的完整性，并保证可执行目标的代码处于受控状态并且可以再生。

这种审核应确定：

a. 为了取得适航证而已经计划的软件生命周期过程活动（包括软件生命周期数据的生成）已经完成，并保存其完成的相关记录。
b. 根据特定系统要求、有关安全方面的要求或软件要求编制的软件生命周期数据可以追溯到这些要求。
c. 软件生命周期数据符合软件计划和标准规范，并根据 SCM 计划进行控制。
d. 问题报告符合 SCM 计划要求，并且已经进行评估，并将其状态记录在案。
e. 记录软件要求偏差并得到相应批准。
f. 可执行目标的代码可以根据存档的源代码进行再生。
g. 经过批准的软件可以通过使用发布的说明书成功地加载。
h. 根据上一个软件符合性审核编制的问题报告应重新评估，以确定其符合性情况。
i. 如果要谋求使用以前开发的软件的合格证书，那么，现有软件产品原始资料就可以追溯到以前的原始资料和对那个原始资料的更改批复文件。

附注：*对于认证后软件的更改，可以完成相应的软件符合性审核，这可以根据更改的重要性来进行判断。*

9 认证联络过程

认证联络过程要达到的目标就是要在整个软件生命周期内在申请人和认证主管部门之间建立沟通和理解渠道，以帮助完成这一认证过程。

认证联络过程的申请可以根据软件规划过程（第 4 节）以及软件认证方面的计划（11．1 小节）来确定。附件 A 中的表 A – 10 是这一过程的目标和输出信息的总结。

9.1 符合性措施及其规划

申请人提出一个确定机载系统或设备的开发如何满足认证基础的符合性措施。而软件认证方面的计划（11．1 小节）则在所提出的符合性措施范围内确定机载系统或设备方面的软件。这一计划还要说明在系统安全评估过程中确定的软件等级。申请人应该：

a．向认证主管部门提交软件认证方面的计划和其它所要求提供的数据，以便于在变更影响程度最低时（即可以在项目约束条件范围可以进行控制时）及时进行审核。

b．解决由认证主管部门标注的软件认证计划的问题。

c．在软件认证计划方面与认证主管部门达成一致意见。

9.2 符合性证据

申请人应提供软件生命周期过程符合软件计划的证据。认证主管部门可以在申请人的业务场所或申请人的供应商经营场所内进行审核。这可能涉及到与申请人或申请人的供应商进行讨论。申请人负责安排这些软件生命周期过程的审核活动，并根据需要提供软件生命周期数据。申请人应该完成下列事项：

a．解决由认证主管部门提出的审核结果发现的问题。

b．向认证主管部门提交软件完成总结报告（11．20 小节）和软件配置指标（11．16 小节）。

c．根据认证主管部门要求提交或准备其它符合性数据或证据。

9.3 提交给认证主管部门有关软件生命周期数据的最低要求

提交给认证主管部门的软件生命周期数据的最低要求如下：

- 软件认证方面的计划；
- 软件配置指标；
- 软件完成总结报告。

9.4 与典型设计有关的软件生命周期数据

除非与认证主管部门另行签署协议，与典型设计有关的软件生命周期数据检索和批准有关的规定适用于下列内容：

- 软件要求数据；
- 设计说明；
- 源代码；
- 可执行的目标代码；
- 软件配置指标；
- 软件完成总结。

10 飞机和发动机认证概述

本章节是飞机和发动机在机载系统和设备软件方面认证过程的一个概括性描述，在信息方面仅供参考。认证主管部门把软件视为安装在飞机和发动机上的机载系统或设备的一部分。这就是说，认证主管部门并不把该软件作为一个唯一和独立的产品来批准。

10.1 认证基础

认证主管部门经过与申请人协商后建立飞机和发动机认证基础。这种认证基础连同那些可以补充公布的相关规定的一切特殊条件一起确定其专项认证规定。

对于改型的飞机和发动机来说，认证主管部门考虑到这种改型对原来为飞机和发动机制定的认证基础的影响。在某些情况下，这种改型后的认证基础可能与原来的认证基础没有任何变化。然而，原来的符合性措施可能无法再表明这种改型仍然符合规定的认证基础，所以，可能需要进行相应的修改。

10.2 软件方面的认证

认证主管部门要对软件方面的认证计划进行评估，以确定它是否完整，是否与那些为满足认证基础而制定的符合性措施保持一致。认证主管部门要确信申请人申报的软件等级与系统安全评估过程的输出信息和其它系统生命周期数据一致。认证主管部门会将有关申报的软件计划中需要在认证主管部门批准之前满足规定要求的问题通知申请人。

10.3 符合性确认

在认证之前，认证主管部门确定飞机和发动机（包括其系统和设备软件）符合规定的认证基础。对于这些软件来说，这种符合性确认通过审核“软件完成情况综述”和符合性证据来完成。认证主管部门采用“软件完成情况综述”作为软件方面认证的一个概述。

认证主管部门可以在软件生命周期随意审核软件生命周期过程及其输出信息，参阅 9．2 小节。

11 软件生命周期数据

软件生命周期数据是在软件生命周期内生成的，其作用就是用来计划、指导、解释、确定、记录或提供活动证据。这一数据可以启用软件生命周期过程、系统或设备认证，以及软件产品的认证后修改。本节讨论软件生命周期数据的特征、形式、配置管理控制和内容。

软件生命周期数据的特征：

a．清楚而明白：如果信息采用只有一种解释的术语写成，必要时辅以定义说明，其信息自然不会含糊不清。

b．完整；当信息包括必要的、相关的要求和/或说明性材料，响应确定了有效输入数据的范围，所有的图形作了标识，而且术语和计量单位都已经确定，那么，其信息自然就非常完整。

c．可验证：如果信息可以通过人员或工具来检查其正确性，那么，它就可以进行验证。

d．一致性：如果信息内部没有任何冲突，那么说明这一信息前后一致。

e．可修改：如果信息在构成时具有一种格式使之可以完全、连续和正确地进行修改而又保留其结构，说明这一信息具有可修改性。

f．可追溯：如果可以确定其各个组成部分的来源，说明这一信息具有可追溯性。

此外，这一指导原则还适用于：

g．形式：在机载系统和设备的整个使用寿命期间，这种形式应该提供软件生命周期数据的有效检索和审核。这种数据和特定数据形式应该在“软件认证计划”中加以规定。

附注：*（1）软件生命周期数据可能呈现许多种形式。例如，它可以表现为书面形式作为计算机文件夹存储在磁性媒介中，也可以在远程终端上显示。它可以包装作为单个文件，结合成更大的文件。或者在几个文件中分配。*

（2）一些认证主管部门可能要求提供“软件认证计划”和“软件完成情况的总结”作为单独的打印文件。

软件生命周期数据可以放置于与软件配置管理控制有关的两类控制方式的其中之一：第 1 种控制（CC 1）和第 2 种控制（CC 2）（7．3 小节）。这一区别为控制开发成本提供一个手段，它可以放宽某些控制而不至于是使其安全等级降低。

下面的数据说明对那些一般用于软件认证方面的数据加以规定。这些说明并不是专门用来说明开发软件产品所必须的所有数据，也不是用来暗示某一特定的数据包装方法或某一包装内的数据编排方式。

除了在这些小节中规定的数据外，也可以形成其它数据，作为对认证过程有帮助的证据。

h．控制：如果专门用于这一用途，那么，这一数据应该在软件计划中加以规定。这一数据的特性和内容可以会发生变化，至少，应该采用第 2 种控制（CC 2）。其例子包括备忘录和会议纪要。

11.1 软件认证计划

“软件认证计划”是认证主管部门用来确定某一申请人申报的某一种软件生命周期是否所开发的软件等级所规定的严格要求的主要手段。这一计划应该包括下列内容：

a．系统综述：本节提供该系统的一个综述，包括其功能的描述，及其对硬件和软件、体系结构、所用的处理器、硬件/软件接口和安全特性方面的规定。

b．软件综述：本节简单描述软件功能，重点放在所申报的安全和分区概念，例如资源分享、冗余度、多版本不相似软件、故障容差范围、定时和进度策略等。

c．认证要考虑的因素：本节提供认证基本要求的总结，包括有关软件认证方面的符合性措施。本节还说明所申报的软件等级，并总结通过系统安全评估过程所提供的相关证据（包括导致失效状态

的潜在软件）。

d．软件生命周期：本节确定要采用的软件生命周期，包括一份每个软件生命周期及其过程（在这一过程中要在其相应的软件计划中确定其详细信息）的总结。这一总结解释每一个软件生命周期过程的目标是如何实现的，规定了所涉及的组织机构、各个机构的责任以及系统寿命周期过程和认证联络过程的责任。

e．软件生命周期数据：本节对将在软件生命周期过程中生成并控制的软件生命周期数据进行规范。本节还描述数据之间的彼此关系、与确定本系统的其它数据之间的关系、提交给软件认证主管部门的软件生命周期、数据的形式，以及软件生命周期数据提交给认证主管部门所借助的手段和方法。

f．进度表：本节描述申请人准备为软件认证当局提供使之可以直观看到软件生命周期过程各项活动以便认证当局可以计划进行审核的相应措施。

g．其它考虑因素：本节描述可能影响认证过程的特定属性，例如符合性替代方法、工具质量检验、以前开发的软件、选项可选择软件、用户可修改软件、COTS 软件、现场安装软件、多版本不相似软件和产品使用情况记载。

11.2 软件开发计划

软件开发计划包括目标、标准和软件开发过程中所采用的软件生命周期。它可以包含于软件认证计划。这一计划应该包括：

a．标准：软件要求标准的标识、软件设计标准和该项目的软件代码。此外，参照以前开发的软件的相关标准，包括 COTS 软件，如果这些软件不同的话。

b．软件生命周期：描述使用软件生命周期过程以形成用于这种项目的特定软件生命周期的情况，包括软件开发过程的转换条件。这一描述与软件认证计划中所提供的有所不同，因为它提供的是详细描述，以确保软件生命周期过程的正确实施。

c．软件开发环境：所选择的软件开发环境在硬件和软件方面的陈述，包括：

（1）所选择的软件要求编制方法和所采用的工具；

（2）所选择的设计方法所采用的工具；

（3）所采用的编程语言、编码工具、编译程序、链接编辑程序和安装程序；

（4）所采用的工具硬件平台。

11.3 软件验证计划

软件验证计划主要是描述各个验证步骤符合软件验证过程目标的情况。这些步骤可能随着软件等级的不同（参阅附件 A 中各表的规定）。这一计划应该包括：

a．组织机构：软件验证过程中的组织责任，以及与其它软件生命周期过程的相互关系。

b．独立性：描述创建验证独立性的方法（如果要求的话）。

c．验证方法：描述用于软件验证过程每个活动的验证方法。

（1）对这些方法进行审核，包括审核表或其它辅助手段；

（2）对这些方法进行分析，包括可追溯性分析和覆盖分析；

（3）对这些方法进行测试，包括建立校验数据和条件选择过程的认证指南、所采用的测试步骤和将要生成的测试数据。

d．软件验证环境：软件验证环境是对测试设备、测试和分析工具描述以及应用这些工具和硬件测试设备（另见段落 4．4．3 中 b 项参阅特定程序计算机与模拟或仿真装置之间的区别）的一些认证指南。

e．转换条件：本计划中规定的进入软件验证过程的转换条件。

f．分区考虑因素：如果采用分区的话，则要考虑用来验证分区完整性的方法。

g．编译程序假定条件：申请人所作的有关编译程序、链接编辑程序或安装程序正确性的假定条件的描述（参阅段落 4．4．2）。
h．重新验证指南：对于软件修改来说，描述用来识别软件受影响区域以及可执行目标代码被变更的部分的方法。重新验证应该保证以前报告的错误或者若干类错误已经清除。
11．以前开发的软件：对于以前开发的软件来说，如果用于验证过程的初次符合性资料不适用于本文件，那么就应该描述满足本文件目标要求的方法。
j．多版本不相似软件：如果使用多版本不相似软件的话，描述软件验证过程的各项活动（参阅段落 12．3．3）。

11.4 软件配置管理计划

软件配置管理计划为在整个软件生命周期范围内达到软件配置管理（SCM）过程的各项目标而提供相应的方法。这一计划应该包括：
a．环境：描述所采用的软件配置管理环境，包括步骤、工具、方法、标准、组织责任和相互关系。
b．活动：描述软件生命周期中满足下列目标所进行的软件配置管理活动：
（1）配置标识：要标识的项目、什么时候标识、软件生命周期数据的标识方法（例如零件的编号），以及软件标识与机载系统或设备之间的关系。
（2）原始资料和可追溯性：创建原始资料的方法、要建立什么样的原始资料、什么时候建立这些原始资料、软件库的控制、配置项目和原始资料的可追溯性。
（3）问题报告：用于软件产品和软件生命周期过程的问题报告的内容和标识、什么时候书写、关闭问题报告的方法，以及与更改控制活动的关系。
（4）更改控制：要控制的配置项目和原始资料、什么时候进行控制、对它们实施控制的问题/更改控制活动、认证前的控制、认证后的控制以及保持原始资料和配置项目完整性的措施。
（5）更改审核：处理来自和发送到软件生命周期过程反馈的方法；评估、排定优先考虑的问题、批准更改、处理其问题的解决或更改的实施；问题报告方法和更改控制活动之间的关系。
（6）配置状态统计：需要记录的用来启用报告配置管理状态的数据、该数据保存位置的确定，以及如何检索这些数据，什么时候得到这些数据。
（7）存档、检索和发布：完整性控制、发布方法和授权以及数据保存。
（8）软件加载控制：描述软件加载控制安全措施和记录。
（9）软件生命周期环境控制：用来开发、创建、验证和加载软件的工具的控制，参阅上述第 1~第 7 项。这包括需要进行检验的工具的控制。
（10）软件生命周期数据控制：第 1 种控制（CC 1）和第 2 种控制（CC 2）（7．3 小节）数据的控制。
c．转换条件：本计划中规定的进入软件配置管理过程的转换条件。
d．软件配置管理数据：由软件配置管理过程中产生的软件生命周期数据的定义，包括软件配置管理记录、软件配置指标和软件生命周期环境配置指标。
e．供应商控制：将软件配置管理过程要求应用到分供应商的措施。

11.5 软件质量保证计划

软件质量保证计划为达到软件质量保证（SQA）过程中的各项目标而提供相应的方法。软件质量保证计划可能包括过程改进、公制计量以及渐进式管理方法的说明。这一计划应该包括：
a．环境：描述所采用的软件质量保证环境，包括范围、组织责任和相互关系、标准、步骤、工具和方法。
b．权限：软件质量保证权限、责任和独立性的陈述，包括软件产品的批准权限。
c．活动：每个软件生命周期过程和整个软件生命周期内要完成的各项软件质量保证活动，具体内

容包括：
（1）软件质量保证方法，例如软件生命周期过程的审核、审查、报告、检验和监测；
（2）与问题报告、跟踪和纠正措施体系有关的活动；
（3）软件符合性审核活动的描述。
d．转换条件：本计划中规定的进入软件质量保证过程的转换条件。
e．定时：与软件生命周期过程活动有关的软件质量保证过程各项活动的规定时间。
f．软件质量保证记录：由软件质量保证过程产生的相关记录的定义。
g．供应商控制：描述保证分供应商认证过程和输出信息符合软件质量保证计划的措施。

11.6 软件要求标准规范

“软件要求标准规范”的目的就是确定用来编制高层次要求的方法、条例和工具。这些标准应该包括：
a．编制软件要求的方法，例如结构化方法。
b．用来表述要求的符号表达法，例如数据流示意图和正式的规范语言。
c．使用软件要求开发工具的约束条件。
d．向系统过程提供衍生要求所使用的方法。

11.7 软件设计标准

“软件设计标准”的目的就是确定用来编制低层次要求的方法、条例和工具。这些标准应该包括：
a．要采用的设计说明的方法。
b．要采用的命名规范。
c．对允许的设计方法施加的条件，例如进度、中断的使用、事件驱动构造、动态任务执行、重新进入、全局数据、例外处置，以及它们使用的基本原理。
d．使用设计工具的约束条件。
e．有关设计的约束条件，例如排除循环数、动态目标、数据别名和紧凑表达法。
f．复杂性约束条件，例如最高层次嵌套的调用或条件结构、无条件转移的使用、代码各个组成部分进/出点。

11.8 源代码标准

“源代码标准”的目的就是确定用来给软件编码的编程语言、方法、条例和工具。这些标准应该包括：
a．所采用的编程语言和/或确定的子集。对于编程语言来说，参照明确规定语法的数据、控制特性和这种语言的数据特征和副作用。这可能要求限制使用某一语言的某些特征。
b．源代码表示的标准，例如行长度的限制、空行的使用和源代码文件标准，例如作者名称、版本修订记载、输入和输出信息以及受影响的数据。
c．源代码各个组成部分、子程序、变量和常量的命名习惯。
d．对允许采用的编码规范的条件和限制条件，例如软件各个组成部分之间的耦合程度、逻辑或数值表达法的复杂性，以及它们使用的基本原理。
e．使用编码工具的限制条件。

11.9 软件需求文档

“软件需求文档”就是软件高层次要求（包括衍生要求）的一个定义。这一文档应该包括：

a．系统对软件要求的配置说明，重点在于有关安全上的要求和潜在的失效状态。
b．每种操作模式下的功能和操作要求。
c．性能标准规范，例如精确度和准确性。
d．定时要求和约束条件。
e．存储器尺寸方面的约束条件。
f．硬件和软件接口，例如文件协议、格式、输入信息的频率和输出信息的频率。
g．失效检测和安全监测要求。
h．分配给软件的分区要求、分区的各个软件组成部分彼此是如何相互影响的以及每个分区的软件等级。

11.10 设计说明

“设计说明”就是软件结构和满足软件高层次要求的软件低层次要求一个定义。这一数据应该包括：
a．软件如何满足规定的软件高层次要求的详细描述，包括算法、数据结构，以及软件要求是如何分配给处理器和任务的。
b．确定执行这些要求的软件结构的软件体系结构的描述。
c．输入/输出说明，例如数据词典、内部和外部软件体系结构。
d．这种设计的数据流和控制流。
e．资源限制、管理每种资源的战略及其限制规定、边际、测量这些边际的方法，例如定时和存储器。
f．计划完成的步骤、处理器之间、任务之间的沟通机制，包括对时间要求严格的排序和优先安排计划。
g．设计方法及其应用的详细描述，例如软件数据加载、用户可修改软件或多版本不相似软件。
h．分区方法和预防违背分区的相应措施。
11．软件各个组成部分的描述，它们是属于新开发的，还是以前开发的，如果是以前开发的软件组分，参照其原始资料。
j．来自软件设计过程的软件衍生要求。
k．如果系统包含无效代码，应说明所采取的措施，以保证该代码在特定程序计算机中无法被启用。
l．可以追溯到有关安全方面系统要求的那些设计决定的基本原理。

11.11 源代码

这一数据由用源语言写成的代码、根据源代码生成目标代码的编译程序说明以及链接程序和加载数据组成。这一数据应该包括软件标识，包括软件名称、修订日期和/或版本（如果有的话）。

11.12 可执行的目标代码

可执行的目标代码由一种形式的源代码组成，它可以通过特定程序计算机中的中央处理器直接取用。因此，可执行的目标代码是一种加载到硬件或系统中的软件。

11.13 软件验证校验数据和条件

软件验证校验数据、条件和步骤详细说明软件验证过程活动是如何完成的。这一数据应该包括下列活动或活动的描述：
a．审核分析步骤：作为补充软件验证计划中说明的详细规定，它描述了所采取的审核或分析方法的范围和深度。

b．校验数据和条件：每个校验数据和条件的目的、输入信息组、条件、要达到期望的覆盖范围标准的预期结果，以及合格/不合格判据。
c．测试步骤：如何设置每个校验数据和条件以及如何实施、测试结果如何评估的逐条描述、要采用的测试环境。

11.14 软件验证结果

软件验证结果是通过软件验证过程活动产生的。软件验证结果应该：
a．对于每个审核、分析和测试来说，软件验证结果应该显示在这些活动中通过或未通过验证的每个步骤和合格/不合格的最终结果。
b．识别所审核、分析或测试的配置项目或软件版本。
c．包括测试、审核和分析（包括覆盖分析和可追溯性分析）的结果。

11.15 软件生命周期环境配置指标

软件生命周期环境配置指标（SECI）标识软件生命周期环境的配置。写出这一指标有助于硬件和软件生命周期环境的复制，以便于软件再生、重新验证或软件修改。软件生命周期环境配置指标应该：
a．标识软件生命周期环境硬件及其操作系统软件。
b．标识软件开发工具，例如编译程序、链接编辑程序和安装程序以及数据完整性工具（例如计算和嵌入检验和或循环冗余检验的工具）。
c．标识用来验证软件产品的测试环境，例如软件验证工具。
d．标识合格工具及其相关的工具检验数据。
附注：*这一数据可以包含在软件配置指标中*。

11.16 软件配置指标

软件配置指标（SCI）识别软件产品的配置。

附注：*软件配置指标（SCI）可以包含下列项目，也可能参照其它软件配置指标或其它规定单个项目及其版本的配置标识数据*。

软件配置指标（SCI）应该标识：
a．软件产品。
b．可执行目标代码。
c．每个源代码的组成部分。
d．软件产品中以前开发的软件（如果使用的话）。
e．软件生命周期数据。
f．存档和发布媒介。
g．创建可执行目标代码的说明，包括用于编译和链接程序的说明和数据、用来恢复软件以便进行再生、测试或修改的步骤。
h．如果是单独封装的话，参照软件生命周期环境配置指标（SCI）（11．15 小节）。
11．可执行目标代码的数据完整性检查（如果使用的话）。
附注：*可以为一个软件产品版本制定软件生命周期环境配置指标，或者将它扩展到包含可以用于几个替代或连续软件产品版本的数据*。

11.17 问题报告的编制

问题报告是标识和记录软件产品异常行为、过程与软件计划和软件标准的不符合性以及软件生

命周期数据中的缺陷的解决方案。问题报告的编制应该包括：

a．配置项目和/或发现问题的软件生命周期过程活动的标识。

b．要修改的配置项目或要更改的过程的描述。

c．可以使问题得到理解并解决的问题描述。

d．为解决上报问题所采取措施的描述。

11.18 软件配置管理记录

软件配置管理（SCM）过程活动的结果在软件配置管理记录中加以记载。其实例包括配置标识表、原始资料或软件库记录、更改历史报告、存档记录和发布记录。这些实例并不表示需要生成这些特定类型的记录。

附注：*由于软件配置管理（SCM）过程的整体特性，其输出信息通常作为其它软件生命周期数据的一部分被包含在内。*

11.19 软件质量保证记录

软件质量保证（SQA）过程活动的结果在软件质量保证记录中加以记载。这些结果可以包括软件质量保证审核或审查报告、会议纪要、经过批准的过程偏差的记录或软件符合性审核记录。

11.20 软件完成情况综述

软件完成情况综述是显示符合软件认证计划的主要数据项目。这一软件完成情况综述应该包括：

a．系统综述：本节提供该系统的一个综述，包括其功能的描述，及其对硬件和软件、体系结构、所用的处理器、硬件/软件接口和安全特性方面的规定。本节还说明了与软件认证计划中系统综述之间的不同之处。

b．软件综述：本节简单描述软件功能，重点放在所申报的安全和分区概念，并解释了与软件认证计划中申报的软件综述之间的不同之处。

c．认证要考虑的因素：本节再次重申软件认证计划中所描述的认证考虑的因素，并说明其中存在的不同之处。

d．软件特征：本节说明可执行目标代码大小、定时和存储边际、资源限制和测量每个特征的手段和方法。

e．软件生命周期：本节总结实际软件生命周期，并解释与“软件认证计划”中所申报的软件生命周期和软件生命周期过程之间的区别。

f．软件生命周期数据：本节参照软件开发和软件整合过程活动所生成的软件生命周期数据。它说明数据之间的彼此关系、与确定本系统的其它数据之间的关系，以及将软件生命周期数据提交给软件认证主管部门所借助的手段和方法。本节还说明了与“软件认证计划”中所申报的软件生命周期数据之间的区别。

g．其它考虑因素：本节总结可能引起管理当局注意的认证问题，并参照适用于这些问题的数据项目，例如问题报告或特殊条件。

h．软件的标识：本节用件号和版本号来标识软件的配置。

ll．软件更改的历史记录：在适用情况下，本章节包括一份软件更改的总结报告，注意那些由于影响系统安全的失效而做出的更改，并将上一次认证以来的更改与软件生命周期过程区分开来。

j．软件状态：本节包括一个在认证时尚未解决的问题报告的总结，包括功能局限性的相关申明。

k．符合性陈述：本节包括一个符合本文件要求的陈述以及用来说明符合软件开发计划中规定条件的方法的 1996 - 07 – 02 总结。

12 一些补充考虑因素

本文件的前几节为满足认证要求提供一些指导原则，申请人可以根据这些要求提交软件生命周期过程的证据。本节介绍在以前开发软件的使用、软件工具的质量检验以及为达到本文件前几节确定目标而采用的替代方法方面有关软件认证的补充考虑因素。使用以前开发的软件的目的要在“软件认证计划”加以陈述。

12.1 以前开发软件的使用

本节认证指南讨论有关以前开发软件的使用问题，包括修改的评估、更改飞机计算机设施的影响、应用或开发环境的改变、软件配置管理和软件质量保证方面的考虑因素。

12.1.1 以前开发软件的修改

本指导原则讨论在以前的软件生命周期过程的输出信息符合本文件要求的情况下对以前开发软件的修改。修改可以是软件要求更改、错误检测和/或软件增强的结果。所申报修改的分析活动包括：

a．该系统安全评估过程修改的输出信息应该根据所申报的修改内容进行审核。

b．如果软件登记修改了，那么，应该考虑段落 12．1．4 中的原始资料编制过程的升级指南。

c．应该分析软件要求更改的影响和软件体系结构更改的影响，包括软件要求更改对其它要求和几个软件组成部分之间的耦合的重要影响，从而可能导致除了受影响区域之外的其它部分也需要进行重新验证。

d．应该确定受到更改影响的部分。这可以通过数据流分析、控制流分析、定时分析和可追溯性分析来完成。

e．受这种更改影响的各个部分应该根据第 6 节的认证指南重新进行验证。

12.1.2 飞机计算机设施的改变

如果机载系统或设备所包含的软件以前已经在某一方面软件等级并在一个特定的认证基础下经过认证的话，那么，这种机载系统或设备就可以用于一个新的飞机计算机设施中。如果使用以前开发的软件，这一指导原则应该用于新的飞机计算机设施中。

a．该系统安全评估过程评估新的飞机计算机设施，并确定软件等级和认证基础。如果新的计算机设施的软件等级相同的话，那么就无须再做评估，因为它们依然处于以前的计算机设施中。

b．如果在新的计算机设施中要求进行功能修改，那么，段落 12．1．1 中的“以前开发软件的修改”的相关要求应该得到满足。

c．以前的开发活动没有产生用来证明新的应用中的安全目标所需要的输出信息的话，那么，段落 12．1．4 中的“原始资料编制过程的升级”指南应该得到满足。

12.1.3 应用或开发环境的改变

a．如果一个新的开发环境使用软件开发工具，那么，12．2 小节中的“工具质量检验”指南可以适用。

b．一项应用更改的评估严格程度应该考虑编程语言的复杂性和严密度。例如，ADA 语言类属评估的严格程度就比在新的应用中生成参数不同这种情况下的严格程度更大一些。

c．如果使用不同的编译程序或不同组的编译程序选项，产生不同的目标代码，那么，使用这种目标代码的以前软件验证过程活动所得出的结果可能会无效，而且不能用于新的应用领域。在这种情况下，以前测试的结果对于新的应用的结构覆盖条件来说可能就不再有效。同样，编译程序有关最

优化的假定可能就不再有效。
d．如果使用一种不同的处理器，那么：
（1）在新的应用中不应该使用以前软件验证过程活动（在硬件/软件接口处进行验证）所得到的结果。
（2）如果是新的应用，以前硬件/软件的整合应该进行测试。
（3）硬件/软件的兼容性审核应该重复进行。
（4）硬件/软件的整合可能需要进行追加测试和审核。

e．以前开发的软件采用不同的接口软件时，就应该进行软件接口的验证。

12.1.4 原始资料编制过程的升级

原始资料编制过程的升级
- 商用存货软件；
- 按照其它指南开发的机载软件；
- 在本文件存在之前开发的机载软件；
- 按照本文件要求以前开发的较低等级的软件。

原始资料编制过程的升级指导原则包括：
a．在利用以前开发的符合新应用的目标的软件生命周期数据时，本文件的各项目标应该得到满足。
b．软件方面的验证应该根据系统安全评估过程所确定的失效状态和软件等级来进行。然后将它与以前应用中的失效状态进行比较即可确定可能需要升级的区域。
c．来自以前开发的软件生命周期数据应该进行评估，以保证在新的应用中该软件等级的软件验证过程的各项目标同样也得到满足。
d．可以采用反向工程来再生在满足本文件各项目标中不适合或丢失的软件生命周期数据。除了产生软件产品外，可能还需要完成附加活动，以满足软件验证过程的目标。
e．如果计划使用产品使用史来满足本文件在原始资料编制过程升级的各项目标，那么应该考虑采用段落 12．3．5 中的相关指南。
f．申请人应该在“软件认证计划”中对完成符合本文件要求的策略方针加以规定。

12.1.5 软件配置管理方面的考虑因素

如果使用以前开发的软件，那么除了第 7 节中的指南外还应该包括用于新应用的软件配置管理过程：
a．从以前应用的软件产品和软件生命周期数据到软件产品和软件生命周期数据的可追溯性；
b．可以使问题报告、问题解决和软件组成部分的更改跟踪用于一个以上的应用场合的更改控制。

12.1.6 软件质量保证方面的考虑因素

如果使用以前开发的软件，那么除了第 8 节中的指南外还应该包括用于新应用的软件质量保证过程：
a．保证软件组成部分满足或超过适用于新应用软件等级的软件生命周期标准要求；
b．保证软件生命周期过程的更改都在软件计划中加以陈述。

12.2 软件工具的质量检验

当使用一种软件工具而无须按照第 6 节规定对其输出信息进行验证从而使本文件的各个过程取消、减少或自动进行时，工具就需要进行质量检验。使用软件工具使软件生命周期过程的活动自动

进行有助于实现系统安全目标，在这种情况下，它们可以强制性符合软件开发标准并使用自动检验。工具质量检验过程所规定的目标就是要确保该工具至少等效于要取消、减少或自动完成软件生命周期过程活动的那个（些）过程。如果可以演示工具功能的分区，那么，只有用来取消、减少或自动完成软件生命周期过程活动的那些功能以及其输出信息才不用进行验证，而需要进行质量检验。
只有确定的工具可以进行质量检验，也就是说，在相同的运行环境下，只有为相同输入数据产生相同输出信息的工具才进行质量检验。工具质量检验过程可以适用于单一工具，也可以适用于一组工具。

软件工具可以划分为下列两种类型：

• 软件开发工具：其输出信息属于机载软件组成部分的工具，因此可能带来错误。例如如果生成的源代码不按照第 6 节中的规定进行验证的话，那么，直接根据低层次要求生成源代码的工具必须进行质量检验。

• 软件验证工具：工具不能带来错误，但可能无法检测出错误来。例如，一个可以实现软件验证过程自动化的静电分析仪，如果其功能没有通过其它活动检验的话，那么就应该进行质量检验。等级检测装置、分析工具和测试工具则属于其它例子。

工具质量检验指导原则包括：

a．工具应该按照上述规定的类型进行质量检验。

b．综合的软件开发工具和软件验证工具应该按照段落 12．2．1 指南中的规定进行质量检验，除非可以对这两个功能之间的分区进行示范。

c．机载系统或设备的软件配置管理过程和软件质量保证过程的目标应该适用于需要进行质量检验的工具。

软件开发工具的软件验证过程目标的具体描述参阅段落 12．2．1 中的 d 项。

一个工具的质量检验只能对“软件认证计划”中的规定用途有效，如果这种工具用于其它系统中则可能需要进一步进行质量检验。

12.2.1 软件开发工具的质量检验标准

软件开发工具的质量检验标准包括：

a．如果一种软件开发工具要进行质量检验，那么，该工具的软件开发过程应该满足机载软件的软件开发过程所确定的各项目标要求。

b．分配给该工具的软件等级应该与它产生的机载软件的软件等级相同，除非申请人能够向认证主管部门证明工具的软件等级已经降低。

附注：*工具的软件等级的降低取决于要取消、减少或自动进行的软件验证过程活动的重要性。这一重要性取决于下列因素*：

• 要取消、减少或自动进行的软件验证过程活动的类型。例如源代码与软件标准的符合性验证活动在重要性方面低于可执行目标代码与软件高层次要求的符合性验证活动。

• 其它验证活动已经检测出相同错误的可能性。

c．申请人应该通过演示证明这种工具符合“工具操作要求”（参阅段落 12．2．3．2）。这一演示可能涉及到一个试运行阶段。在这一阶段，工具输出信息已经完成，相关的工具问题已经得到分析、记录和纠正。

d．软件开发工具应该进行验证，以检查“工具操作要求”的正确性、一致性和完整性，以验证该工具是否符合这些要求。该工具软件验证的目标与机载软件的目标不同，因为工具的高层次要求对应于“工具操作要求”而不是系统要求。软件开发工具的验证可以通过下列途径达到：

（1）按照段落 6．3．1 a 和 b 项规定审核工具操作要求；

（2）通过演示证明这种工具在正常运行条件下符合其“工具操作要求”；
（3）通过演示证明这种工具在异常运行条件下（包括外部干扰和所选择的失效适用于工具及其环境）执行时符合其“工具操作要求”；
（4）根据要求确定的覆盖分析和完善该要求覆盖范围的追加测试；
（5）适合于该工具软件等级的结构覆盖分析；
（6）按照段落 6．3．2．2 的规定，用一个适合该工具软件等级的复杂数据流来测试工具的坚固性；
（7）分析由该工具产生的潜在错误，以确认工具质量检验计划的有效性。

12.2.2 软件验证工具的质量检验标准

通过演示证明这种工具符合其正常运行条件下的“工具操作要求”，以达到满足软件验证工具的质量检验标准要求的目的。

12.2.3 软件工具质量检验数据

工具质量检验数据指导原则包括：

a．在检验工具时，相关机载系统或设备的“软件认证计划”应该对要检验的工具进行规范，并参照工具质量检验数据。

b．工具质量检验数据应该控制为用于软件开发工具的第 1 种控制（CC 1）和用于软件验证工具的第 2 种控制（CC 2）。

c．对于软件开发工具来说，其工具质量检验数据应该与第 11 节中的数据一致，并与用于机载系统或设备软件的数据具有相同的特性和内容，所考虑的因素包括：

（1）工具质量检验计划符合机载系统或设备的“软件认证计划”的相同目标；
（2）工具操作要求符合机载系统或设备的软件要求数据的相同目标；
（3）工具完成情况综述符合机载系统或设备软件的“软件完成情况综述”的相同目标。

12.2.3.1 工具质量检验计划

对于要进行质量检验的软件开发工具来说，工具质量检验计划描述工具质量检验过程。这一计划应该包括：

a．工具的配置标识；
b．合格证书所谋求的具体目标，即要取消、减少或自动完成的软件验证过程活动；
c．为该工具申报的软件等级；
d．该工具体系结构的描述；
e．要完成的工具质量检验活动。

12.2.3.2 软件工具的使用要求

工具的使用要求描述该工具的使用功能。这些数据包括：

a．该工具的使用功能和技术特性的描述。对于软件开发工具来说，它包括由这种工具完成的软件开发过程；
b．用户信息，例如安装指南和用户手册；
c．这种工具使用环境的描述；
d．对于软件开发工具来说，这些数据还包括该工具在异常运行状态下的预期反应。

12.2.4 工具质量检验批准

认证主管部门通过下列两个步骤批准使用某一工具：

对于软件开发工具来说，按照工具质量检验计划要求进行审批。对于软件验证工具来说，按照机载系统或设备的“软件认证计划”进行审批。
对于软件开发工具来说，按照工具完成情况综述要求进行审批。对于软件验证工具来说，按照软件完成情况综述要求进行审批。

12.3 替代方法

由于一些方法在本文件编写时尚不成熟或限制在机载软件上的应用，所以，在本文件的前面几节没有进行讨论。本文件并不是有意要限制任何现有或未来的方法的应用。本小节中所讨论的任何单独一个替代方法并不能被视为要替代本文件所推荐的那些方法，而是可以用来实现本文件中规定的一个或多个目标。

替代方法可以用来彼此印证。例如形式方法可以用来支持工具质量检验，而一个经过检验的工具也有助于形式方法的使用。
一个替代方法不可以被认为独立于软件开发过程。一个替代方法要获得认证证书取决于软件等级以及这种替代方法对软件生命周期过程的影响。使用替代方法的指导原则包括：
a．一个替代方法应该符合本文件规定的目标；
b．申请人应该在“软件认证计划”中加以规定，并在下列方面得到认证主管部门的批准：
（1）所申报的方法对软件开发过程的影响；
（2）所申报的方法软件生命周期数据的影响；
（3）使用替代方法的基本原理，即说明该替代方法如何符合该系统的安全目标。
c．这种基本原理应该通过软件计划、过程、预期结果来证明，并提供使用这种方法的证据。

12.3.1 形式方法

形式方法涉及到使用形式逻辑、离散数学可计算语言来改进软件的规范和验证。这些方法可以产生一个应用，结果表明其工作特性完全处于规定的范围内。在其非常严格的应用中，形式方法可以等效于一个系统在其要求方面的完备性分析。这一分析可以提供：
• 系统已经完善，其要求已经合理的证据。
• 确定哪一个代码、软件要求或软件体系结构符合下一个较高层次的软件要求。

应用形式方法的目标就是要在整个软件开发过程中防止和消除要求错误、设计错误和代码错误。因此，形式方法是对测试的补充。测试表明功能要求得到了满足，错误得到检测，形式方法可以用来提高其可信度：即异常行为不会发生（对于输入信息来说就是不会超出范围），也不可能发生。

形式方法可以适用于软件开发过程，并应考虑下列一些因素：
• 设计改进的层次：用一种正式的规范语言规范软件的高层次要求并用合理的证据验证它们符合系统要求（尤其是可接受的软件运行方面的约束条件），那么，形式方法就开始使用了。随后便显示下一个较低层次要求符合其高层次要求。向下完成这一过程直到源代码，即可提供证据证明该软件符合系统要求。形式方法可以按照连续层次的设计改进启用和终止使用，从而提供证据证明那些层次的软件要求已经得到正确规定。
• 软件要求和软件体系结构的覆盖范围：形式方法可以适用于下列软件要求：
  • 与安全有关；
• 可以通过离散数学来确定；
• 涉及复杂行为，例如同时发生、中断处理、冗余管理和同步。
这些条件可以用来确定这种形式方法所应用的那种设计改进层次上的一组软件要求。
• 严格程度：形式方法包括下列严格程度逐渐递增的标准规范：
• 没有经过任何验证的正式规范；

• 经过人工验证的正式规范；
• 经过自动验证的正式规范。

单独使用正式规范使得这些软件要求不会出现歧义。人工验证是一个很好理解的过程，在不怎么具体时就可以采用。自动验证可以帮助人工验证过程，并可以提供一个较高程度的可靠性，尤其是对于更为复杂的验证。

12.3.2 完备的输入信息测试

有时候往往会存在这样的情况：某一机载系统或设备较为简单而且处于封闭状态，这样就可以包含这组输入和输出信息。如果属于这种情况，那就可以说明这一输入空间的完备测试经过一次软件验证过程活动即可得到验证。要采用这一替代方法，申请人应该具备：

a．一组软件的输入和输出信息的完整的定义；
b．一份确认软件的输入信息的封闭性的分析报告；
c．这种完备的输入信息校验数据和条件和步骤；
d．校验数据和条件、测试步骤和测试结果。

12.3.3 多版本不相似软件验证的考虑因素

在考虑软件验证过程之后便提供相应的认证指南，因为它适用于多版本不相似软件。如果由于使用多版本不相似软件而修改软件验证过程，那么就应该提供证据证明该软件验证过程的目标已经符合，而且每一个软件版本都完成了等效的错误检测。

采用下列综合方法可以开发出多版本的不相似软件。

• 源代码是用两个或多个不同编程语言来完成的。
• 目标代码是用两个或多个不同编译程序来生成的。
• 可执行目标代码的每个软件版本在一个封闭的不同处理器上验证，或者在一个单独的处理器上验证时采取相应措施在软件版本之间提供分区。
• 软件要求、软件设计和/或源代码由两个或多个相互影响受到控制的开发团队开发的。
• 软件要求、软件设计和/或源代码在两个或多个软件开发环境中开发的，每个版本都采用单独的测试环境来进行验证。
• 可执行目标代码采用两个或多个不同链接编辑程序和两个或多个不同的安装程序来链接和安装。
• 软件要求、软件设计和/或源代码是分别按照两个或多个不同的软件要求标准、软件设计标准和/或源代码标准开发的。

当采用多版本软件时，软件的验证方法可能与验证那些单一版本的软件有所不同，应进行相应的修改。它们将适用于多头软件开发过程活动，例如单一开发团队和多组开发团队。这种软件验证过程取决于综合的硬件和软件体系结构，因为这影响到多重软件版本的不相似性。要达到的其它软件验证过程目标是：

a．满足不同版本之间的兼容性要求，包括在正常操作和异常操作过程中以及在状态转换过程中的兼容性；
b．证明已经达到等效的错误检测。

软件验证过程活动的其它更改可能要经过认证主管部门的同意，以证明这些更改是否经过基本原理的验证具有相同的软件验证覆盖范围。

12.3.3.1 多版本不相似软件的独立性

如果多版本不相似软件的版本是采用受控方法单独开发的，其开发过程就可能会发现错误的某

种类别，使得每个软件版本的验证与软件开发过程的独立验证效果相同。为了发挥这一潜能，其独立性指导原则包括：

a．申请人应该表明相互间作用有限的不同团队编制出每种软件版本的软件要求、软件设计和源代码；

b．对于单一版本来说，仍然应该完成独立测试覆盖分析。

12.3.3.2 有关多处理器的验证

当每个版本的不相似软件在一个不同类型的处理器上运行时，该代码与处理器（参阅段落6．4．3）在某些方面兼容性的验证可以替换为确保多种类型处理器生成正确的输出信息的验证。这一验证由整合测试组成，通过这一测试多种版本的输出信息可以在根据要求确定的校验数据和条件中进行横向比较。申请人应该证明：

a．达到了等效错误检测；

b．每种处理器都由一个不同的开发商设计的；

c．多版本的输出信息效果相同。

12.3.3.3 多版本源代码的验证

结构覆盖分析认证指南（参阅段落6．4．4．2）可以使用多版本不相似软件进行修改，以便适用机载系统或设备的需要。结构覆盖分析可以在源代码层次上进行，假定申请人提供下列证明，即使其目标代码无法直接追溯到源代码说明也可以完成。

a．每种软件版本采用一种不同的编程语言进行编码；

b．所使用的每个编译程序来自不同的开发商。

12.3.3.4 多版本不相似软件验证的工具质量检验

如果采用多版本不相似软件，而且有证据表明多项软件开发工具不同的话，那么，这种工具质量检验过程可以进行相应的修改。这取决于等效软件验证过程活动采用多种开发工具在多软件版本开发中的示范效果。申请人应该证明：

a．每种工具都来自不同的开发商；

b．每种工具都有一个不同的设计。

12.3.3.5 多模拟装置和验证

如果采用独立的不同模拟装置来验证多版本不相似软件，那么，该模拟装置的工具质量检验可以进行修改。这取决于等效软件验证过程活动通过多个模拟装置在多软件版本模拟中的示范效果。除非可以证明没有必要，否则，对于不同的多个模拟装置来说，应该提供下列证据：

a．每种模拟装置由不同的团队开发的；

b．每种模拟装置有不同的要求、不同的设计和不同的编程语言；

c．每种模拟装置在不同的处理器上运行。

附注：*当使用多个不同版本的软件的多个处理器系统在相同的处理器上运行时，可以很难示范模拟装置的不相似性，因为这要取决于从一个普通来源、处理器制造商那里获得的信息。*

12.3.4 软件可靠性模型

在本文件的编制过程中，我们对验证后软件存在错误的概率的评估方法进行了检测。其目标就是编制电脑操作的机载系统或设备的这种数值要求。不过，所得出的结论是，现有方法并没有提供令人信服的结果。因此，本文件不提供软件错误率方面的指导。如果申请人申报使用软件可靠性模

型来获得认证，应该在“软件认证计划”中提供这种模型的基本原理，并获得认证主管部门的许可。

12.3.5 产品使用史

如果用软件产品使用情况记载可以证明该软件具有等效安全保障，那么就可以获得某种合格证书。这一方法的可接受程度取决于下列各个方面：

- 该软件的配置管理；
- 问题报告活动的有效性；
- 该软件的稳定性和成熟度；
- 产品使用的历史环境；
- 实际误差率和产品使用史；
- 修改的影响。

产品使用史使用的指导原则包括：

a．申请人应该表明软件和用来说明达到系统安全目标的相关证据在整个产品使用过程中已经处于配置管理之下。

b．申请人应该表明产品使用过程中问题报告机制为确保提交的是代表性数据，并保证使用过程中的问题都已经报告、进行记录并且可以进行检索。

c．在产品使用过程中，配置更改可以识别，其影响已经经过分析，确保软件的稳定性和成熟。在产品使用过程中那些可执行目标代码中未受控制的更改可能使得产品使用史料的使用无效。

d．对于软件的预定用途应该加以分析，以标明产品使用史的关联性。

e．如果现行和申报的应用工作环境不同的话，追加的软件验证应该确认符合系统安全目标。

f. 配置更改分析和产品使用史环境可能要求使用软件要求和设计数据来确认使用产品史环境的适用性。

g．如果该软件是一个在使用期限内可以运行的软件子集，那么，通过分析确认新环境与以前环境完全等效，并确定这些软件组成部分在正常运行过程中没有执行。

附注：*可能需要进行追加验证以确认符合这些软件组成部分的系统安全目标*。

h．应该对问题报告史进行分析，以确定与安全有关的问题是如何发生的以及那些问题得到了纠正。

i．那些显示相关过程实施不当的问题（例如软件设计或代码问题）应该单独标识出来，以便与那些超出本文件适用范围的问题（例如硬件或系统要求错误）区别开来。

j．上述数据和这些项目应该在“软件认证计划”中明确加以规定。

（1）产品使用史环境的关联性分析；

（2）使用期限的长度、计算使用小时数的基本原理，包括诸如工作模式、在安装和使用中独立使用副本的数量等参数，以及“正常运行”和“正常运行时间”的定义。

（3）定义什么可以算作一个错误，以及这种定义的基本原理；

（4）所申报的可接受的错误率以及与系统安全和所申报的错误率有关的产品使用期限基本原理。

k．如果错误率大于计划中标定的标准，应该对这些错误进行分析，并将这些分析提交认证主管部门审核。

附件 A 按照软件等级划分的软件编制过程目标和输出信息

本附件按照软件等级为本文件中所述的软件生命周期目标和输出信息提供相关指南。这些表参照本文件中上述软件生命周期各个过程的目标和输出信息。

这些表中包括了用于下列场合的指南：

a．适用于每个软件等级的该设计过程的目标。对于 E 级软件来说，可以参阅 2．2．2 节；

b．按照软件等级不同适合于满足过程目标的软件生命周期目标设计过程活动的独立性；

c．由软件生命周期目标设计过程活动产生的按照软件等级不同适用于软件生命周期数据的控制类别。

表 A - 1 软件规划程序

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	确定软件开发和一体化程序活动	4．1 a 4．3	○	○	○	○	认证方面的软件计划 软件开发计划 软件验证计划	11．1 11．2 11．3	① ① ①	① ① ①	① ① ②	① ① ②
2	确定各个过程之间的过渡标准、相互关系和运行次序	4．1 b 4．3	○	○	○	○	SCM 计划 SQA 计划	11．4 11．5	① ①	① ①	② ②	② ②
3	确定软件生命周期的运行环境	4．1 c	○	○	○	○						
4	处理其它应该考虑的因素	4．1 d	○	○	○	○						
5	确定软件开发标准	4．1 e	○	○	○	○	软件要求标准规范 软件设计标准 源代码标准	11．6 11．7 11．8	① ① ①	① ① ①	② ② ②	
6	软件计划符合本文件的要求	4．1 f 4．6	○	○	○	○	SQA（软件质量保证）记录 软件验证结果	11．19 11．14	② ②	② ②	② ②	
7	软件计划协调一致	4．1 g 4．3	○	○	○	○	SQA（软件质量保证）记录 软件验证结果	11．19 11．14	② ②	② ②	② ②	

图表说明：
该目标应该满足独立性；

○ 该目标应该得到满足；
空白：目标的满意度取决于申请人的判断；
①数据应满足控制类别 1（CC 1）的要求；
②数据应满足控制类别 2（CC 2）的要求。

表 A - 2 软件开发程序

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	编制相关的高层次要求	5. 1. 1a	○	○	○	○	软件要求数据	11．9	①	①	①	①
2	确定衍生的高层次要求	5. 1. 1b	○	○	○	○	软件要求数据	11．9	①	①	①	①
3	编制软件体系结构	5. 2. 1a	○	○	○	○	有关设计的说明	11．10	①	①	②	②
4	编制相关的低层次要求	5. 2. 1b	○	○	○	○	有关设计的说明	11．10	①	①	②	②
5	确定所得到的低层次要求	5. 3. 1a	○	○	○	○	有关设计的说明	11．10	①	①	②	②
6	编制源代码	5. 3. 1b	○	○	○	○	源代码	11．11	①	①	①	①
7	编制可执行的目标代码，并在特定程序计算机中进行集成	5. 4. 1a	○	○	○	○	可执行的目标代码	11．12	①	①	①	①

图表说明：
● 该目标应该满足独立性；
○ 该目标应该得到满足；
空白：目标的满意度取决于申请人的判断；
①数据应满足控制类别 1（CC 1）的要求；
②数据应满足控制类别 2（CC 2）的要求。

表 A - 3 软件要求程序输出信息的验证

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	软件高层次要求符合系统要求	6. 3. 1a	●	●	○	○	软件验证结果	11．14	②	②	②	②
2	软件高层次要求精确而且前后一致	6. 3. 1b	●	●	○	○	软件验证结果	11．14	②	②	①	①
3	高层次要求与特定程序计算机兼容	6. 3. 1c	○	○			软件验证结果	11．14	②	②		
4	高层次要求具有可验证性	6. 3. 1d	○	○	○		软件验证结果	11．14	②	②	②	
5	高层次要求符合相关标准	6. 3. 1e	○	○	○		软件验证结果	11．14	②	②	②	
6	软件高层次要求可追溯到系统要求	6. 3. 1f	○	○	○	○	软件验证结果	11．14	②	②	②	②
7	算法非常精确	6. 3. 1g	●	●	○		软件验证结果	11．14	②	②	②	

图表说明：

● 该目标应该满足独立性；

○ 该目标应该得到满足；

空白：目标的满意度取决于申请人的判断；

①数据应满足控制类别 1（CC 1）的要求；

②数据应满足控制类别 2（CC 2）的要求。

表 A - 4 软件要求程序输出信息的验证

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	软件低层次要求应符合高层次要求	6．3．2a	●	●	○		软件验证结果	11．14	②	②	②	
2	软件低层次要求精确而且前后一致	6．3．2b	●	●	○		软件验证结果	11．14	②	②	②	
3	低层次要求与特定程序计算机兼容	6．3．2c	○	○			软件验证结果	11．14	②	②		
4	低层次要求具有可验证性	6．3．2d	○	○			软件验证结果	11．14	②	②		
5	低层次要求符合相关标准	6．3．2e	○	○	○		软件验证结果	11．14	②	②	②	
6	软件低层次要求可追溯到高层次要求	6．3．2f	○	○	○		软件验证结果	11．14	②	②	②	
7	算法非常精确	6．3．2g	●	●	○		软件验证结果	11．14	②	②	②	
8	软件体系结构与高层次要求兼容	6．3．2a	●	○	○		软件验证结果	11．14	②	②	②	
9	软件体系结构前后一致	6．3．2b	●	○	○		软件验证结果	11．14	②	②	②	
10	软件体系结构与特定程序计算机兼容	6．3．3c	○	○			软件验证结果	11．14	②	②		
11	软件体系结构具有可验证性	6．3．3d	○	○			软件验证结果	11．14	②	②		
12	软件体系结构符合相关标准	6．3．3e	○	○	○		软件验证结果	11．14	②	②	②	
13	软件分区的完整性得到确认	6．3．3f	●	○	○	○	软件验证结果	11．14	②	②	②	②

图表说明：

● 该目标应该满足独立性；

○ 该目标应该得到满足；

空白：目标的满意度取决于申请人的判断；

①数据应满足控制类别 1（CC 1）的要求；
②数据应满足控制类别 2（CC 2）的要求。

表 A - 5 软件要求程序输出信息的验证

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	源代码符合低层次要求	6. 3. 4a	●	●	○		软件验证结果	11．14	②	②	②	
2	源代码符合软件的体系结构要求	6. 3. 4b	●	○	○		软件验证结果	11．14	②	②	②	
3	源代码具有可验证性	6. 3. 4c	○	○			软件验证结果	11．14	②	②		
4	源代码符合相关标准	6. 3. 4d	○	○	○		软件验证结果	11．14	②	②	②	
5	源代码可追溯到低层次要求	6. 3. 4e	○	○	○		软件验证结果	11．14	②	②	②	
6	源代码准确且前后一致	6. 3. 4f	●	○	○		软件验证结果	11．14	②	②	②	
7	软件整合过程的输出信息完整而且正确无误	6. 3. 5g	○	○	○		软件验证结果	11．14	②	②	②	

图表说明：
● 该目标应该满足独立性；
○ 该目标应该得到满足；
空白：目标的满意度取决于申请人的判断；
①数据应满足控制类别 1（CC 1）的要求；
②数据应满足控制类别 2（CC 2）的要求。

表 A - 6 软件要求程序输出信息的验证

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	可执行的目标代码符合高层次要求	6.4.2.1 6．4．3	○	○	○	○	软件验证情况和步骤 软件验证结果	11．13 11．14	① ②	① ②	① ②	① ②
2	可执行的目标代码非常健全，满足高层次要求	6.4.2.2 6．4．3	○	○	○	○	软件验证情况和步骤 软件验证结果	11．13 11．14	① ②	① ②	② ②	② ②
3	可执行的目标代码符合低层次要求	6.4.2.1 6．4．3	●	●	○		软件验证情况和步骤 软件验证结果	11．13 11．14	① ②	① ②	② ②	
4	可执行的目标代码非常健全，满足低层次要求	6.4.2.2 6．4．3	●	○	○		软件验证情况和步骤 软件验证结果	11．13 11．14	① ②	① ②	② ②	
5	可执行的目标代码与特定程序计算机兼容	6．4．3a	○	○	○	○	软件验证情况和步骤 软件验证结果	11．13 11．14	① ②	① ②	② ②	② ②

图表说明：

● 该目标应该满足独立性；

○ 该目标应该得到满足；

空白：目标的满意度取决于申请人的判断；

①数据应满足控制类别 1（CC 1）的要求；

②数据应满足控制类别 2（CC 2）的要求。

表 A - 7 软件验证过程结果的验证

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	测试步骤正确无误	6．3．6 b	●	○	○		软件验证情况和步骤	11．13	②	②	②	
2	测试结果正确无误，对不相符部分应加以解释	6．3．6c	●	○	○	○	软件验证结果	11．14	②	②	②	
3	达到高层次要求的测试范围	6．4．4．1	●	○	○		软件验证结果	11．14	②	②	②	②
4	达到低层次要求的测试范围	6．4．4．1	●	○	○		软件验证结果	11．14	②	②	②	
5	软件体系结构的测试范围（修改的条件/决定）	6．4．4．2	●				软件验证结果	11．14	②			
6	软件体系结构的测试范围（确定的覆盖范围）	6．4．4．2a 6．4．4．2b	●	●			软件验证结果	11．14	②	②		
7	软件体系结构的测试范围（报告的覆盖范围）	6．4．4．2a 6．4．4．2b	●	●	○		软件验证结果	11．14	②	②	②	
8	软件体系结构的测试范围（数据耦合和控制耦合）	6．4．4．2c	●	●	○		软件验证结果	11．14	②	②	②	

表 A - 8 软件配置管理过程

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	识别配置项目	7．2．1	○	○	○	○	SCM 记录	11．18	②	②	②	②
2	建立原始材料和可追溯性	7．2．2	○	○	○	○	软件配置指标 SCM 记录	11．16 11．18	① ②	① ②	① ②	① ②
3	问题报告的编制、变更控制、变更审核，并建立配置状态记录	7．2．3 7．2．4 7．2．5 7．2．6	○	○	○	○	问题报告 SCM 记录	11．17 11．18	② ②	② ②	② ②	② ②
4	建立相关的数据的存档、检索和发布	7．2．7	○	○	○	○	SCM 记录	11．18	② ②	② ②	② ②	② ②
5	对文件加载编制相应的控制规定	7．2．8	○	○	○	○	SCM 记录	11．18	② ②	② ②	② ②	② ②
6	编制软件生命周期环境控制条例	7．2．9	○	○	○	○	软件生命周期环境配置指标 SCM 记录	11．15 11．18	① ②	① ②	① ②	② ②

图表说明：

● 该目标应该满足独立性；

○ 该目标应该得到满足；

空白：目标的满意度取决于申请人的判断；

①数据应满足控制类别 1（CC 1）的要求；

②数据应满足控制类别 2（CC 2）的要求。

附注：（1）虽然第 7 节中的软件配置管理目标并不随着软件等级的改变而改变，但是，指定给软件生命周期数据的控制类别可能会发生变化；

（2）第 7 节中的软件配置管理目标可以在 SCM 过程中的活动提供一个充分的完整性标准，而无须提供独立性标准。

表 A - 9 软件软件质量保证

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	质量保证应做到软件开发和整合过程符合已经批准的软件计划和标准要求	9．0	○	○	○	○	软件质量保证（SQA）记录	11．19	②	②	②	②
2	质量保证应使得那些用于软件生命周期过程的过渡标准得以满足	9．1	○	○	○	○	SCM 记录	11．19	②	②		
3	完成软件的符合性审核	9．2	○	○	○	○	SCM 记录	11．19	②	②	②	②

图表说明：

● 该目标应该满足独立性；

○ 该目标应该得到满足；

空白：目标的满意度取决于申请人的判断；

①数据应满足控制类别 1（CC 1）的要求；

②数据应满足控制类别 2（CC 2）的要求。

表 A - 10 软件质量认证联络过程

	目标		按照软件等级的适用性				输出信息		按照软件等级的控制类别			
	名称	参考文件号	A	B	C	D	名称	参考文件号	A	B	C	D
1	申请人与认证主管部门之间建立相应的沟通与理解渠道	8．1 a	○	○	○	○	软件认证方面的计划	11．1	①	①	①	①
2	提出符合性措施和方案，并与“软件认证方面的计划”达成一致	8．1 b	○	○	○	○	软件认证方面的计划	11．1	①	①	①	①
3	提供符合性证明材料	8．1 c 8．3	○	○	○	○	软件认证方面的计划 软件配置指标	11．20 11．16	① ①	① ①	① ①	① ①

图表说明：

● 该目标应该满足独立性；

○ 该目标应该得到满足；

空白：目标的满意度取决于申请人的判断；

①数据应满足控制类别 1（CC 1）的要求；

②数据应满足控制类别 2（CC 2）的要求。

附注：“软件认证方面的计划”和“软件配置指标”是其它过程的输出信息。它们包含于本表中，用来表示认证联络过程目标的完成。

附件 B 首字母缩略词和术语词汇

首字母缩略词

AC：通报
AMJ：通报材料 - 联合报告
CC 1：控制类别 1
CC 2：控制类别 2
COTS：商用现有软件
EUROCAE：欧洲民航设备组织
FAA：联邦航空管理局
FAR：联邦航空管理条例
IC：集成电路
I/O：输入和/或输出
JAA：联合航空管理局
JAR：联合航空要求
RTCA：RTCA 公司（非赢利性公司）
SCI：软件配置指标
SCM：软件配置管理
SECI：软件生命周期环境配置指标
SQA：软件质量保证

术语词汇

下面是为本文件所用的术语提供的定义。如果一个术语在这一附件中没有加以定义，那么，它可以在此加以定义，而不是在本文件的文本中进行定义。参阅《美国传统词典》查阅普通术语的定义。

算法- 是在有限步骤内求解某一问题所使用的一组定义明确的规则。
异常行为- 不符合规定要求的行为。
申请人-谋求获得认证主管部门批准的个人或组织。
批准-表达赞成意见或给出正式的行动或事例。
保证-使人相信或提供证据表明产品或过程满足给定的要求所必须的计划和系统的行动。
审核-软件生命周期过程及其输出信息的一种独立检查，以确认所要求的属性。
原始资料- 一个或多个配置项目批准、记录的配置，这一配置以后可以作为进一步开发的基础，而且只能通过更改控制程序来进行更改。
认证- 一个产品、服务、组织或个人符合本要求时由认证主管部门作出的合法认可。这一认证包括对产品、服务、组织或个人进行技术检验，并通过发放一个其它国家法律和程序要求的合格证、许可证、批复或文件形式正式承认符合适用的要求。尤其是产品的合格证包括：（1）评估一个产品设计确保其符合适用于该类产品的一系列标准，以表明一个合格的安全等级的过程；（2）评估一个单

独的产品以确保它符合认证过的型样设计的过程；（3）根据国家法律要求，申明已经确认符合上述（1）和（2）项标准的合格证的发放。

认证主管部门- 在州或联邦政府范围内负责有关符合该要求认证的组织或个人。

附注：*有关飞机、发动机或推进器类型认证或有关设备批准的事项一般由该认证主管部门处理；有关连续适航性的事项则由适航性主管部门处理。*

认证证书- 认证主管部门对满足某一认证要求的一个过程、产品或演示的承认。

更改控制-（1）记录、评估、批准或不批准以及在正式建立其配置标识后协同更改配置项目或原始资料的过程。（2）系统评估、协调、批准或不批准以及在正式建立其配置标识后配置项目的配置中批准更改的实施。

附注：*这一术语在其它行业可能称为配置控制。*

代码-特殊数据的应用或以一种符号形式出现的一个特殊计算机程序，例如源代码、目标代码和机器代码。

商用现有软件（COTS）-由卖主通过公共目录方式销售的商业现货交易。商用现有软件（COTS）不采用专门的定制方式。开发用于某一特定应用的合同协商软件不属于商用现有软件。

编译程序-将一种高级语言的源代码表述（例如 FORTRAN 或 Pascal）编译成目标代码的程序。

组成部分（简称组分）- 一个独立的部分，即构成一个整体的元件、组件或器件，它可以完成一个系统的一个独特的功能。

条件-不含有布尔运算子的布尔表达式。

条件/判定覆盖范围-程序中的每个入口点/出口端至少被启用一次，该程序中一个判定中的每一个条件至少已经将所有可能的成果利用了一次，该程序中每一个判定至少已经将所有可能的成果利用了一次。

配置标识-（1）在一个系统中命名其配置项目并记录其特征的过程；（2）确定一个配置项目的批准文书。

配置项目-（1）为了达到配置管理目的被视作一个单位的一个或多个硬件或软件组成部分。（2）为了达到配置管理目的被视作单位的软件生命周期数据。

配置管理-（1）标识和确定一个系统的配置管理目，在整个软件生命周期内控制这些项目的发布和更改，记录和报告配置管理目的状态和更改请求，并验证配置管理目的完整性和正确性的过程。（2）应用技术和管理指导和监督以①标识和记录一个配置管理目的功能和物理特征，②控制这些特征更改，以及③记录和报告更改控制处理和实施状态的规定。

配置状态统计-管理一项配置所需信息的记录和报告，包括批准的配置标识的列举、所申报的配置的更改以及已经获得批准的更改的实施情况。

控制耦合- 一个软件组成部分影响另一个软件组成部分的方式和程度。

控制程序-用来促进和监督一个计算机系统中的程序执行的一条计算机程序。

覆盖分析-确定一个申报的软件验证过程活动符合其目标的程度的过程。

数据库-构成另组数据的一部分或整体的一组数据，至少由一个足以使用某一给定用途或给定数据处理系统的文件组成。

数据耦合-一个软件组成部分在这种软件组分的控制下对数据的相关性。

数据词典-用于该系统的数据、参数、变量和常量的详细描述。

数据类型-由该类数据各个组成部分以及可以应用于这些组分的运算式来进行特征描述的一类数据。例如字符类型和计算类型。

无效代码-其设计（1）既不是用来执行（代码）也不是用来使用（数据）的可执行目标代码（或数据），例如以前开发的软件组分的一部分；或（2）只是在特定程序计算机的某一配置中执行（代码）或使用（数据）的可执行目标代码（或数据），例如通过一个硬件插针选择或软件编程选项启用的代码。

死代码-一个设计错误无法在特定程序计算机的某一配置中执行（代码）或使用（数据），并且无法追溯到一个系统或软件要求的可执行目标代码（或数据）。嵌入标识符属于一个例外。
判定-一个由条件和零或更多的布尔运算子组成的布尔表达式。没有一个布尔运算子的判定就是一个条件。如果在一个判定中一个条件出现一次以上，那么每一次发生的事件就是一个独特的条件。
判定覆盖范围-程序中每个入口点和出口端至少被激发一次，程序中的每一个判定使用所有可能的成果至少一次。
衍生要求- 来自软件开发过程的一些补充要求，它可能无法直接追溯到较高层次的软件要求。
仿真装置- 一个与具有相同目标代码的给定的系统一样可以接受相同输入信息并产生相同输出信息的装置设备、计算机程序或系统。
等效类别-一个程序输入域的分区，使得该类别中一个代表性值的测试等效于该类别其它值的测试。
错误-对于软件来说，是指要求、设计或代码中出现的差错。
失效-一个系统或系统组成部分在规定的范围内无法完成一个所要求的功能。一次失效可能是在出现一次故障时产生的。
失效状态-考虑到相关的不利的运行条件和环境条件，由一个或多个失效造成的给飞机或机上人员所带来的直接和间接的影响。
故障-软件中一种错误的表现。在出现故障时可能会导致失效。
故障容差-一个系统在出现数量有限的硬件或软件故障时可以提供连续正确执行命令的固有能力
形式方法-用来创建、开发和推理系统特性的数学模型的描述性符号和分析方法。
硬件/软件整合-将软件结合到特定程序计算机中的过程。
高层次要求-根据系统要求、有关安全方面的要求和系统体系结构的分析编制而来的软件要求。
主计算机-开发该软件所用的计算机。
独立性-保证完成目标评估的责任的分割。（1）对于软件验证过程来说，当这种验证活动由被验证项目开发商以外的个人完成，而且一个工具使用时可以达到与人工验证活动相同效果时，其独立性就体现出来了。（2）对于软件质量保证过程来说，独立性还包括确保纠正措施的根据。
集成过程-有助于软件开发过程和其它集成过程的一个过程，因此，在整个软件生命周期始终有效。集成过程包括软件验证过程、软件质量保证过程、软件配置管理过程以及认证联络过程。
中断-一个任务（例如一个计算机程序的执行）由于一个外部事件的发生被暂时搁置，外部事件之后，该任务又会恢复。
低层次要求-从高层次要求、衍生要求和设计约束条件（即源代码可以直接应用无须更多的信息）衍生而来的软件要求。
符合性措施-专门用于申请人满足飞机或发动机认证的基础条件中规定要求的指定方法。例如陈述、设计图纸、分析、计算、测试、模拟、检验和环境质量检测。在适当的情况下，可以使用认证主管部门下发的咨询材料。
媒介-作为软件的转移或存储的一种手段的设备或材料，例如可编程序只读存储器、磁带或磁盘和纸。
存储器件-一件可以存储通过机器识读的计算机程序和相关数据的硬件。
检测-（1）[安全方面]在一个系统中，用来检测该系统异常行为的功能。（2）[质量保证方面]证明或检验所选择的测试、检验或其它活动例证（或者这些活动的记录），以保证该活动处于受控之下，而且所报告的结果代表预期的结果的行动。检测一般与（在 100%检验无法实施或没有必要情况下）在延期时间完成的活动有关。检测可以证明所要求的活动都是按计划进行。
多版本不相似软件-单独开发用来满足相同功能要求的一组或多组程序。其中一个版本的特定错误可以通过多项输出信息的比较检测出来。
目标代码-一般不为特定程序计算机直接使用而是除了处理器说明书信息之外还包括重新定位信息的计算机程序的一种低层次表示法。
件号-用来标识一个配置项目的一组数字、字母或其他字符。

过程-在软件生命周期内完成的用来产生一个确定的输出信息或产品的一系列活动。
产品使用史-一个软件在一个已知的环境中运行并记录连续失效的一段相近的时间周期。
正确性证据-一个程序满足其要求的听起来合乎逻辑的论据。
发布-使一个可检索的配置项目公开面世并授权使用的行动。
反向工程-从源代码中提取软件设计信息的方法。
坚固性-在出现无效输入信息的情况下软件可以继续正确运行的程度。
模拟装置-一个使用由原目标代码衍生的目标代码在软件验证过程中可以接受相同输入信息并产生相同输出信息的装置设备、计算机程序或系统。
软件-计算机程序、和可能相关的文档资料和维持计算机系统运行的数据。
软件体系结构-选择用来实施软件要求的硬件和软件的结构。
软件更改-源代码、目标代码、可执行代码或来自其原始资料与其相关的文档资料的一个修改。
软件整合-将代码组成部分结合起来的过程。
软件库-包含一批用来帮助软件开发、使用或修改的受控软件和相关数据和文件库。例如软件开发库总库、生产库、程序库和软件库。
软件生命周期-（1）由某一组织确定的足以产生一个软件产品的一系列过程；（2）从决定生成或修改一个软件开始并到该产品退出使用时结束的时间周期。
软件分区-将软件分隔成一个或更多的软件属性以防止特定的相互影响和交互耦合干扰的过程。
软件产品-指定交付给用户的计算机程序组、相关的文件材料和数据。在本文件中，这一术语是用于机载应用的软件和相关的软件生命周期数据。
软件要求-所给定的软件要产生什么样的输入和输出信息的描述。软件要求包括高层次要求和低层次要求。
软件工具-一种用来帮助开发、测试、分析、产生或修改另一个程序或其文档资料的计算机程序。
源代码-使用源语言（例如汇编语言和/或高级语言）写成的代码，它以机器可识读的形式将输入信息变成一个汇编程序或一个编译程序。
标准-用来为某一给定活动或特定数据项目性能提供指导和评估的一个规则或可供参照的基准。
静态分析程序-一个用来不执行程序而能发现程序的某些特性的软件工具。
结构-形成一个整体的各个组成部分的特定排列或相互关联性。
系统-组织在一起完成一个特定的功能或一组功能的一批硬件和软件。
系统体系结构-选择用来实施系统要求的硬件和软件的结构。
系统安全评估-所申报的系统一个正在进行的显示相关的安全要求得到满足的系统性和综合性的评估。
系统安全评估过程 – 这些活动主要用来表明其符合适航要求及相关的指导材料，例如 JAA AMJ / FAA AC 25. 1309。这一过程的主要活动包括功能危险性评估、系统安全前期评估和系统安全评估。这些活动的严酷程度取决于相关系统的临界状态、复杂性以及相应的使用场合。
任务 – 从某一控制程序的观点来说的一个基本工作单位。
测试情况 – 一系列测试输入信息、执行情况以及为某一特定目标所带来的预期结果，例如要实施一项特殊程序路径或验证是否符合某一方面特定的要求。
测试 – 执行某一系统或系统组成部分以验证它是否满足特定要求以及检测其中误差的过程。
测试步骤 – 准备并实施某一系列特定测试的详细说明，以及评估完成这种测试所得出结果的说明。
工具的质量检定 – 获得某一特定机载系统范围内的软件工具所必须经历的过程。
可追溯性 – 项目之间（例如过程的输出信息之间、输出信息与其产生过程之间或者某一要求与其实施之间）某一相关性的证据。
转换条件 – 由软件的规划过程确定的可以得到满足以输入一个过程的最低条件。
验证 – 确定软件要求是否是正确要求以及确定其是否完整的过程。该系统生命周期过程可以使用

软件要求和系统验证过程中所得到的要求。

鉴定 – 对保证其正确性和为这种过程提供的相关输入信息和标准相符的某一过程结果的评估。

附件

附录 A

文件 DO - 178 的编制背景

1．0 以前文件的版本演变

1980 年 5 月，航空无线电技术委员会（即现在的 RTCA 公司）建立了一个 145 专门委员会（SC - 145）——“数字航空软件”来开发和编制支持基于软件运行的机载系统和设备的开发的软件习惯操作。此前，欧洲民航电子工业组织，即现在的欧洲民航设备组织（EUROCAE），已经建立 12 工作组（WG - 12）来编制类似的文件，并于 1980 年 10 月准备出版文件 ED – 35 “软件操作建议和机载系统文件编制”。此外，EUROCAE 决定不出版其文件，而是与 RTCA 一起开发一套普通的认证指南。145 专门委员会（SC - 145）编制了 RTCA 文件 DO – 178，“机载系统和设备认证的软件考虑因素”。这一文件得到了 RTCA 执行委员会的同意，由 RTCA 在 1982 年出版。随后不久，EUROCAE 推出了 ED - 12。

早在 1983 年，RTCA 执行委员会决定，应该对 DO - 178A 进行修订，以反映在飞机和发动机（包含有软件的系统和设备）认证过程中所积累的经验。为此，设立了一个 152 专门委员会（SC - 152）。

经过该专门委员会的辛勤工作，终于在 1985 年出版了一个经过修订的 RTCA 文件 DO - 178A “机载系统和设备认证过程中的软件考虑因素”。随后不久，EUROCAE 推出了 ED - 12A。在技术内容方面，它与 DO - 178A 完全相同。

2．0 在本文件形成过程中的 RTCA / EUROCAE 委员会活动情况

1989 年初，美国联邦航空管理局（FAA）正式要求 RTCA 建立一个专门的委员会来审核和修订 DO - 178 A 文件。自从 1985 年推出以来，全世界飞机制造商、航空工业和认证主管部门一直采用 DO - 178 A 或等效的 EUROCAE WG - 12 作为主要指导性文件，以确定系统和设备（包含软件）的可接受程度。

然而，随着软件技术的快速进步（这并非 SC – 152 所能预见的）以及不同的译码（适用于某些关键的领域），这就需要对这些指南进行必要的修订。所以，来自 ARINC、航空飞行员协会、全国商业飞机协会、空中运输协会和 FAA 的代表形成一个 RTCA 特设小组来考虑 FAA 的要求处理的问题。该小组审核了这些问题，总结了 DO - 178 A 应用中所得出的经验，最后得出结论：应该授权一个专门委员会来修订 DO - 178 A。1989 年秋，RTCA 执行委员会设立了 167 专门委员会（SC - 167）来完成这一项任务，并对这些参考术语方面达成协议：

“167 专门委员会将根据需要审核并修订 RTCA 文件 DO - 178 A，即《机载系统和设备认证的软件考虑因素》”。

指导方针：

167 专门委员会应该认清软件要求、软件设计、代码代别、测试和文件编制的动态、演变环境，并形成一个可以适应这一环境并能适当推荐使用严格的技术方法的修订文件。167 专门委员会还应该认识到这一文件的国际意义，因此，它应该与 EUROCAE 建立一个紧密的工作关系。这在 RTCA 委员中已经形成了一种惯例。为了完成这一修订，该专门委员会应该考虑包含于 DO - 178 A 和 DO - 178 B 文件中的指导材料在现场应用过程中所总结的经验以及软件工程近期研究的最新成果。该专门委员会应该下列一些领域关注这一审核：

1. 检查行业标准和政府标准，并对相关内容考虑采用或提供参照；
2. 评估现有软件的层次以及相关分析、验证、测试和确认活动的性质和程度。应该对修改的过程编制相关的标准判据，以支持目标的符合性描述；
3. 检查所采用的工具标准，看是否合格，例如软件开发工具、软件配置管理和软件验证工具；

4. 检查以前开发的软件、现有库存软件、数据库和用于系统认证的用户可修改软件的认证标准；
5. 检查用来降低软件等级或用来提供验证覆盖范围的体系结构和系统对策的认证标准，例如分区和不相似软件；
6. 检查配置控制指南、软件质量指南和认证规章制度；检查其与现有认证主管部门对类型认证、使用过程中的变更和设备批准等要求的兼容性；
7. 考虑新技术的影响，例如，模化体系结构、数据加载、封装和存储器技术；
8. 检查所有文件的需要、内容和发送要求，特别强调“软件编制总结”。
9. 确定并考虑软件和系统生命周期间的接口；
10. 审查与某一系统认证前后进行变更相关的标准判据；
11. 应考虑技术演化的影响，并考虑其它替代的生命周期对 DO - 178 A 文件所指定的型号的影响；

在此，可以再次设立 EUROCAE WG - 12，以便与 SC - 167 一起完成这一项任务，并形成 5 个 RTCA / EUROCAE 联合工作组来处理这些问题：

1. 文件编制的综合与形成；
2. 系统问题；
3. 软件开发；
4. 软件验证；
5. 软件配置管理和软件质量保证。

通过 SC - 167 和 WG - 12 的共同努力，终于出版发行了 RTCA 文件 DO - 178 A 和 EUROCAE 文件 ED - 12 B。

3．0 DO - 178 B 和 DO - 178 A 文件之间的差别的总结

它是 DO - 178 A 文件的整体复制。

建议在孤立使用任何章节或图表之前阅读整个文件。

DO - 178 B 是一种主要针对各个过程的文件。对每个认证过程来说，就是要确定相应的目标，并描述满足这些目标的措施。此外，应提供一个软件生命周期数据描述，以表明这些目标已经得到满足。每个认证过程的目标总结见表，软件等级对其适用性的影响以及软件等级是否与这些目标无关都在该表中加以明确规定。配置管理要求中适用于软件生命周期数据的软件等级的变动也在该表中加以明确规定。

DO - 178 B 认识到有许多不同的软件生命周期可以用来开发机载系统和设备用软件。DO - 178 B 强调，所选择的软件生命周期应该在某一项目的计划过程中加以确定。某一软件开发项目所包含的各个过程应加以具体描述，不管选用哪一种软件生命周期。这些过程可以划分为 3 类：软件规划过程、软件开发过程（包括软件要求、软件设计、软件编码和整合）和整合过程（包括软件验证、软件质量保证、软件配置管理和认证的联络）。其整合过程在整个软件生命周期都在有效进行。DO - 178 B 要求在软件规划过程中应该建立控制软件生命周期过程之间转换的标准规范。

系统生命周期过程和软件生命周期之间的关系要进一步加以确定。在系统安全评估过程中，应该考虑到与软件中要实施的功能相关的失效状况。这是确定软件等级的基础。软件等级目前有 A、B、C、D、E 五个等级。

软件验证部分强调根据软件要求所进行的测试，这样就可以补充体系结构有效区。

要发送的项目和其它数据项目要加以进一步确定，所要求的配置管理也同样需要加以确定。

此外还添加了一个包括其它问题的章节，以便为以前 DO – 178A 中没有涉及到的领域提供指导。这些议题包括以前开发的软件的使用、工具的质量认证以及替代方法（包括形式方法）的使用、

详细的输入信息的测试、多版本不相似软件的验证、软件可靠性模型和产品使用的历史记录。

DO - 178 B 还对术语词汇进行审核找出其中多余的或相互冲突的术语，并在可能的情况下采用一些行业接受的定义。

本文件的这些指南与国际标准 ISO 9000 – 3（1991）“ISO 9001 在软件开发、供应和保养维护中的应用指南”和 IEC 65 A（IEC 秘书处）122（1991-年 11 月的草案）“应用于工业安全系统中的计算机软件”。一般来说，这些指南都符合这些标准的宗旨要求。

本文件还提供一个综合的索引，以便于本文件的推广应