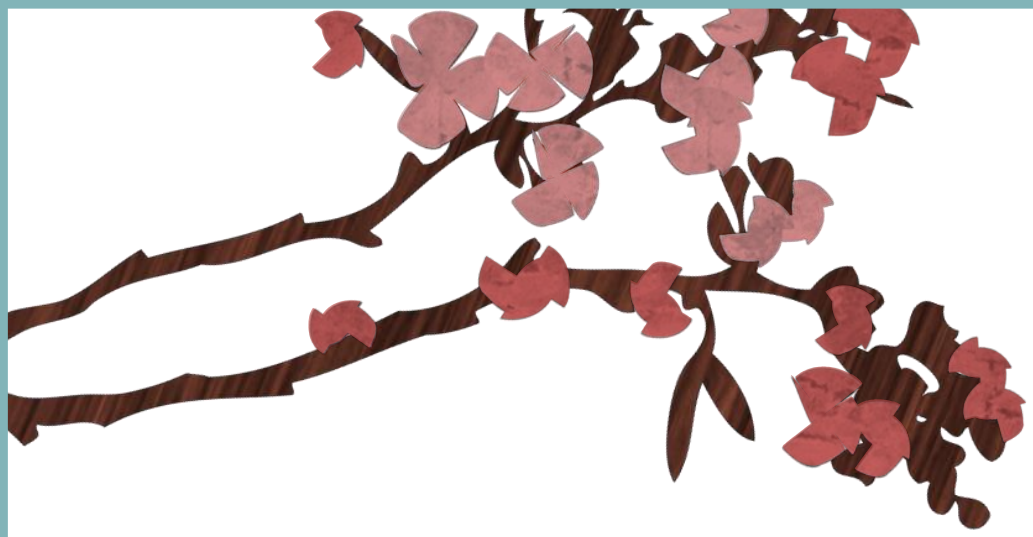


STPA HANDBOOK

NANCY G. LEVESON
JOHN P. THOMAS

MARCH 2018

中文版 V1.0

2019 年 4 月

本手册的核心不是介绍 STPA 的基础理论，而是针对有兴趣在真实系统上使用 STPA 的用户而编制。我们的目标是为那些开始在实际项目中使用 STPA 的人提供指导，或作为 STPA 教学课程中的补充材料。

TRANSLATION TEAM LED BY DR.WANG PENG
COLLEGE OF AIRWORTHINESS, CIVIL AVIATION UNIVERSITY OF CHINA
KEY LABORATORY OF CIVIL AIRCRAFT AIRWORTHINESS TECHNOLOGY, CAAC

译者序

随着技术的发展，现代民机系统向着综合化、智能化和模块化方向发展，系统的复杂程度成级数增长，系统交联、信息融合、人机结合、软硬件结合等方面引起的系统性风险愈加突出，许多新型系统的事故并不来源于单一的系统组件故障，而是因为非故障组件之间的不安全相互作用所致，而表现为系统交联不良、系统组件之间非期望相互作用、系统能力不及等非故障因素，因此需要新的安全性评估和分析技术来解复杂民机系统的安全性问题，需要充分考虑组件故障、组建不良交联、软件需求分析不充分、组件性能变化共振、系统内部（外部）协调、垂直（横向）协调以及人为差错等带来的影响或后果。

麻省理工学院的 NANCY G. LEVESON 教授于 2000 年前后以公开的学术论文形式提出 STAMP 模型和 STPA 方法，基于系统理论和控制理论，将系统的安全性问题看作系统的涌现性，而对这种涌现特性的控制方法是对系统部件行为和交互进行约束，即认为系统的安全状态是由系统对部件行为和部件间交互施加安全约束而得以保持或加强的。STAMP 提出了三个关键的概念即安全约束、控制反馈结构和过程模型，在解决复杂系统诸如核工业安全、食品安全、飞行事故分析等领域取得了很好的效果，目前 STPA 方法也在作为民机安全性分析可接受的一种分析方法被 SAE S-18 委员会讨论，有可能作为 SAE ARP 4761A 附录被提出。

在国际民机安全性分析技术发展的大背景下，我们团队希望将国际先进的安全性评估理论和方法引入到国内，解决我国民机安全性分析的实际问题，在保持对国际安全性相关领域持续追踪的基础上，提出我们对民机安全性问题的中国办法，我们团队与 NANCY G. LEVESON 教授团队建立了合作关系，以推动 STPA 技术在国内的应用和发展。

我们很荣幸能够和 NANCY G. LEVESON 教授、JOHN THOMAS 博士合作在中国推进 STPA 方法的落地和应用，感谢教授的信任。

同时感谢阎芳、赵长啸、李浩、邢培培、董磊、肖女娥等各位同仁、感谢胡剑波教授团队对本翻译稿做出的贡献。

王鹏 博士

中国民航大学 适航学院

民航航空器适航审定技术重点实验室

前言

本手册旨在帮助想要开始使用 STPA 或想要尝试使用 STPA 进行非简单危险源分析的人员。因此，我们设法提供示例，并将更多的示例包含入附录中。我们还提供了一个附录，为未经过工程培训的人员解释一些理解和使用本手册所需的基本工程概念。其他的新概念将在本手册的主要章节中进行介绍。

引言简要介绍了 STAMP——STPA 下的事故因果关系模型。引言还展示了一些事故的例子，并解释了 STPA 是当今复杂的软件密集型系统所必需的原因。

下一章是关于如何进行 STPA 的深入教程，这对初学者以及尝试过 STPA 并希望改善其结果的人员非常有用。

本手册的其余部分介绍了 STPA 的用途，包括如何将 STPA 结合到标准系统工程过程中、在工作场所安全中的作用、使用 STPA 进行组织分析以及安全以外的应急系统属性、使用 STPA 提供风险不断上升的领先指标、设计有效的安全管理系统和网络信息安全。

最后一章介绍了我们在将 STPA 结合到一个大型组织中，以及如何构建 STPA 流程，以使其最有效但对企业的破坏性最小。

我们编写本手册的目的是为那些实际使用 STPA 而不是撰写学术入门书的人员提供指导。因此，我们省略了对其他工作的参考。许多其他来源也提供这类信息，但很少能关注说明和使用 STPA 的方法。查找更多示例、已发表的论文等，请参阅 http://psas.scripts.mit.edu/home/

我们鼓励本手册的用户在阅读手册时能处理与其行业相关的示例。

目录

第 1 章：引言

南希·莱文森

STPA（基于系统理论的过程分析）是一种相对较新的基于事故因果扩展模型的危险分析技术。除了组件故障之外，STPA 假设事故也可能是由系统组件（其中没有一个组件可能发生故障）的不安全交互引起的。与传统的危险/风险分析技术相比，STPA 的优势如下：

- 可分析非常复杂的系统。之前仅在运行中才能发现的“未知的未知”可在开发早期识别出来，并且可以得到消除或缓解。预期的和非预期的功能都得到了处理。
- 与传统的危险分析方法不同，STPA 可以在早期概念分析中启动，以帮助确定安全要求和制约因素。然后，这些可以用于增加系统架构和设计的安全性，从而消除在开发后期或运行期间识别出设计缺陷时的昂贵返工。随着设计的完善和更详细的设计决策，STPA 分析也得到了改进，以帮助做出越来越详细的设计决策。从需求到所有系统工件的完整可追溯性可很容易得到维护，从而增强系统的可维护性和演化能力。
- STPA 在分析中包括软件和人工操作员，从而确保危险分析包括所有潜在的损失因果因素。
- STPA 提供了在大型复杂系统中经常缺失或难以找到的系统功能的文档。
- STPA 可以轻松融入系统工程过程和基于模型的系统工程中。

STPA 与较传统的危险分析方法（例如故障树分析（FTA）、故障模式及其影响关键性分析（FMECA）、事件树分析（ETA）以及危险性和可操作性分析（HAZOP））进行了评估和比较[1]。在所有这些评估中，STPA 发现了较传统的分析发现的所有因果方案，但它也发现了传统方法没有发现的软件相关和非故障的场景。在某些情况下，如果发生了分析师没有被告知的事故，只有 STPA 能找到事故的原因。此外，与传统方法相比，STPA 在时间和资源方面的费用要低得多。

1. 理论基础

使用 STPA 几乎不需要搞明白基础理论和数学基础，但直观的理解是有帮助的。如果您对此主题不感兴趣，我们建议您跳到下一章节。

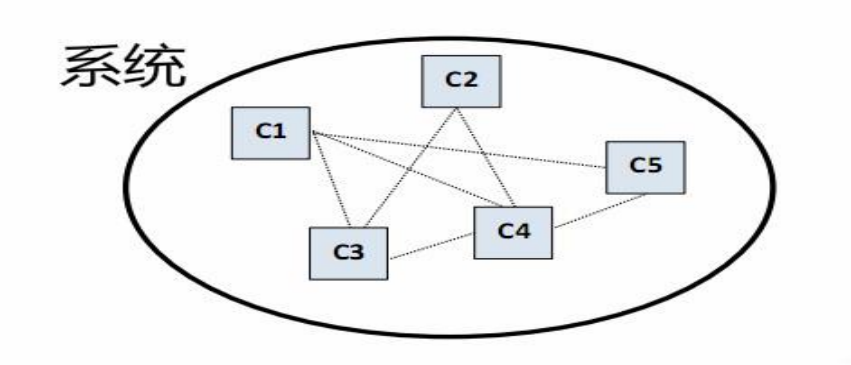
STPA 是基于系统理论的。系统理论是在第二次世界大战后发展起来的，以应对先进技术所创造的日益复杂的系统。首先，了解它所取代的内容是非常重要的。

分析分解（传统方法）

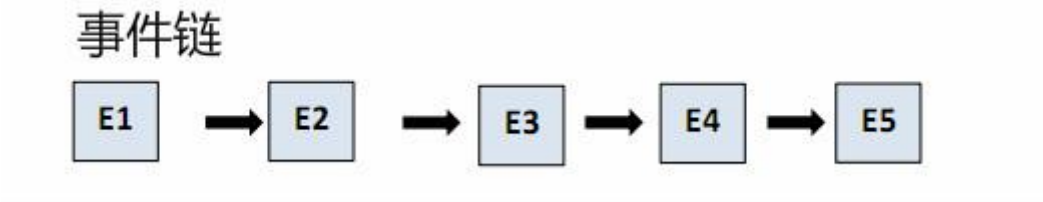
[1] 其中一些信息在麻省理工学院 STAMP/STPA 研讨会上出现过。可在 http://psas.scripts.mit.edu/home/找到演示文稿

几个世纪以来，通过将系统分解为更小的组件，单独检查和分析每个组件，然后组合结果以便理解组合组件的行为来处理复杂性。

物理或功能组件被分解成不同的组件，并假设它们以直接和已知的方式相互作用。例如，飞机的功能组件可以是推进装置、导航系统、姿态控制系统、制动系统、机舱环境控制系统等。飞机也可以分解成不同的物理组件，例如机身、发动机、机翼、安定面、喷嘴等。请注意，可在功能组件和物理组件之间进行映射，但假设映射是直接且已知的。



相反，传统上，行为被建模为随时间推移的独立事件，其中每个事件是前一事件的直接结果。通过使用 AND/OR 逻辑可能会有多个先前或后续事件，但这也会导致多个链（具用 AND/OR 逻辑，用于减少独立链的数量，需要将这些逻辑符号指定到树中）。



这些事件可出现在飞机不同的运行阶段，如推出、滑行、起飞、爬升、巡航、进近和着陆。

一旦系统被分解成组件（件），组件就会分别得到分析，并将单独分析的结果组合在一起以创建系统分析。例如，如果复杂对象的权重是分析目标，则可以对单独的件进行加权并且将结果组合以获得系统权重。系统可靠性作为另一个常见示例，通常通过评估各个组件的可靠性来进行评估，然后以数学的方式组合组件可靠性以评估系统可靠性。

这种分解或简化方法的成功依赖于这样的假设，即分开和个体分析不会扭曲感兴趣的现象或属性。更具体地说，该方法适用于：

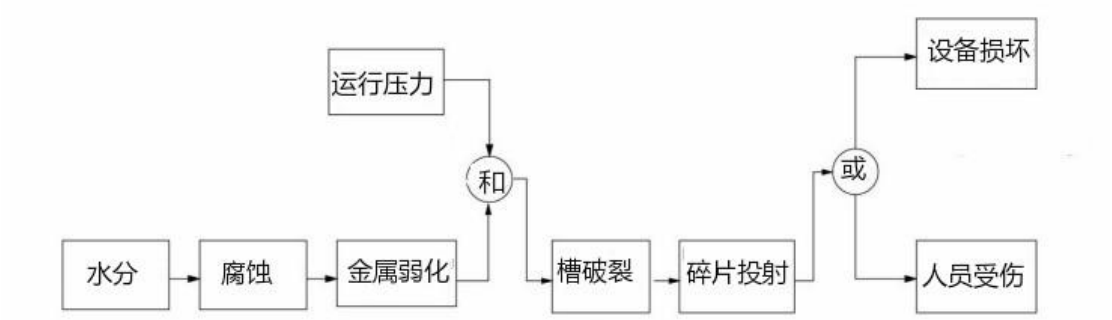
- 每个组件或子系统独立运行。如果对事件建模，除了直接的前后事件，事件是独立的。
- 单个（单独）检查时，组件的行为与它们在整体中发挥作用时的行为相同。

- 组件和事件不受反馈回路和其他间接交互的影响。
- 可以成对检查组件或事件之间的交互，并将其组合成复合值。

如果这些假设不成立，那么这些单独分析的简单组合将无法准确反映系统价值。

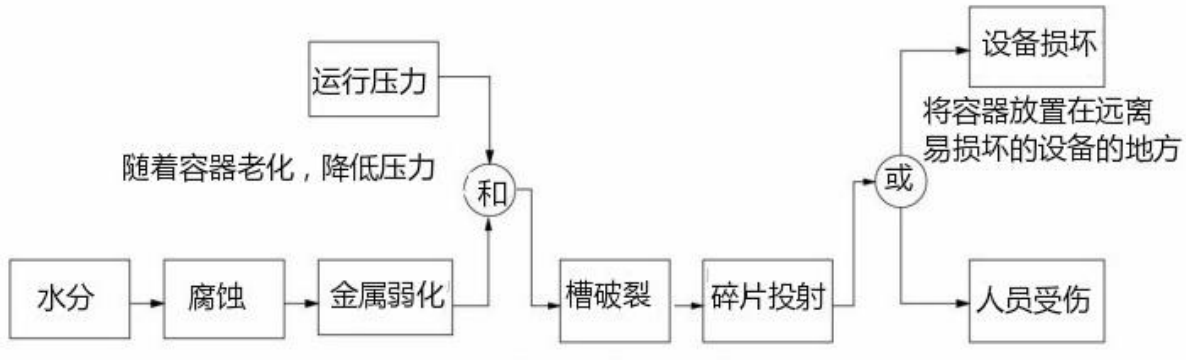
为什么这一切都很重要？因为传统的危险分析是基于分解，因此也是基于这些假设。所涉及的基本方法是将系统划分为组件，假设事故是由组件故障引起的，分别计算每个组件的故障概率，然后将分析结果（基于组件之间相互作用类型的假设）融入系统可靠性框图中，其数值被假定为安全性或风险的度量。示例包括了故障模式及其影响分析（FMEA）以及故障模式和影响关键性分析。后者仅考虑可能导致严重损失的故障，而非所有故障。

或者，识别可导致损失的直接相关的物理或逻辑（功能）故障事件链，并且将每个的概率组合成事件链发生的概率。以下是容器（化学罐）爆炸的一系列事件示例。



在这个例子中，水分进入容器，导致腐蚀。腐蚀导致金属弱化，这与特定的操作压力相结合会导致容器破裂。结果，碎片投出，设备损坏或人员受伤。

如果假设事故是由故障事件引起的，那么预防事故的方法是消除事件或在事件之间设置障碍以使一次故障不会导致另一事件发生。容器（槽）破裂事件模型在下面有注释，包括所使用的事故预防设计技术的典型类型，包括冗余、障碍、组件高完整性和过度设计、故障安全设计、以及对于人类方面的操作程序、检查表的使用和培训。这些设计特征用于降低故障事件发生或传播的概率。



使用干燥剂　　使用不锈　　保险设　　在容器破裂　　使用筛　　增压时，

使水分不能进入容器。	钢或碳钢板涂层以防止接触水分。	计金属厚度，这样腐蚀将不会将强度降低到故障点。	之前，使用爆破隔膜破裂，防止更大的破坏和更多的碎片。	网控制可能出现的碎片。	人员远离容器。

当然，必须假定故障事件为概率或可能性的随机性[2]。不幸的是，软件和人员不满足这种假设。其次，事件链危险分析方法是故障树分析（FTA）、事件树分析（ETA）、故障危险源分析（FHA）以及危险性和可操作性分析（HAZOP，它使用偏差而不是故障作为事件或考虑的条件））的基础。

对于我们过去构建的机电系统类型，传统危险分析的分解方法所依据的假设是真实的（或足够接近），并且对于新的高科技、软件密集型系统中的某些特性，它们仍然是正确的。然而，当今系统的复杂性大大增加（这主要是通过使用软件实现的），所创建的系统中此方法不再有效。在运行我们的系统之前，预测、理解、计划和防范所有潜在的系统行为要困难得多。复杂性正在创造无法通过将系统行为分解为事件链来识别的“未知”。此外，复杂性导致重要的系统属性（例如安全性）与单个系统组件的行为无关，而与组件之间的交互有关。事故可能是由未发生故障但实际上满足了要求的组件之间的不安全交互而引起的。

[2] 如果可以通过概率分布或模式来描述某些东西是随机的，这个概率分布或模式可以通过统计分析来理解平均行为或行为的预期范围，但不能精确地预测。

一些例子可能会有所帮助:

一些海军飞机从一个点到另一个点运送导弹。一名飞行员通过瞄准前面的飞机（正如他被告知要做的那样）并发射一枚虚拟导弹来执行计划的测试。显然没有人知道软件被设计成：如果被命令发射的导弹未处于良好状态，将替换发射另一种导弹。在这种情况下，虚拟导弹和目标之间存在一个触发机制，致使软件决定发射位于不同（更好）位置的实弹。什么飞机部件在这里出现故障了呢？

这次损失涉及火星极地登陆者。使航天器减速以至安全着陆是很有必要的。这样做的方法包括使用火星大气层、降落伞和下降发动机（由软件控制）。一旦航天器着陆，软件必须立即关闭下降发动机，以避免损坏航天器。着陆支架上的一些非常敏感的传感器提供了这些信息。但事实证明，当支架展开时，会产生噪音（假的传感器信号）。这种预期的行为并没体现在软件要求。也许它不包括在内，因为此时软件不应该运行，但是软件工程师决定尽早启动以平衡处理器上的负载。该软件认为航天器着陆并关闭了下降引擎，此时航天器仍在行星表面 40 米处。哪个航天器组件在这里故障了？

当飞机仍然在空中时，激活飞机的反推装置（在飞机接地后使用，使飞机减速）是很危险的。软件设计了保护装置，以防止飞行员在飞机不在地面时错误地启动反推装置。缺少探究细节，软件确定飞机着陆的一些线索是轮重和轮速，而由于各种原因，这种情况在此并不成立。例如，跑道非常潮湿，机轮打滑。结果，飞行员无法启动反推装置，飞机从跑道末端跑到一个小山丘上。什么飞机组件失效了？

这起事故涉及 2012 年在莫斯科着陆的 Tupelov 飞机。缓慢接地使跑道接触时刻比往常晚一点。当时伴有侧风，轮重开关没有启动，反推系统也不会展开。而飞行员认为反推装置会像往常一样展开。因此，在有限的跑道空间的情况下，他们很快就会使用高发动机功率来使其更快地停止。相反，这加速了飞机前进，最终与高速公路堤防相撞。什么飞机组件故障了？

华盛顿州的沿海地区拥有大量的水域和岛屿。汽车渡轮是人们出行的必需品。有一天，当租赁车到达港口后无法从渡轮上驶下时，渡轮系统已经瘫痪。原来，当地一家租车公司安装了一种安全装置，如果汽车在轮渡引擎停止时移动，该装置就会使汽车失效，从而防止盗窃。当渡轮移动并且汽车没有运行时，汽车全部失效，渡轮系统停止运行，直到找到足够的拖车将汽车从渡轮上拖下来。什么汽车组件在这里故障了？

参与此事件的锂离子电池制造商确定，电池单元通风将在 1000 万飞行小时内发生一次，但 2013 年仅两周就有两个电池发生了故障。虽然涉及的因素很多，但其中一个因素是这里有趣的例子。在电池发生故障的情况下，环境控制系统用于通过致动某些阀门来冷却管道风扇以去除烟雾。然而，由于电池故障，向环境控制系统供电的单元同时关闭。结果，阀门不能被致动，并且 APU 电池产生的烟雾不能有效地引导到客舱和电池室外。

航空电子系统中一个更复杂的例子可在附录 A 中找到。此外，格拉茨大学（奥地利）的 Ioana Koglbauer 博士提供了一个强调系统方法有用性的例子。在他们大学，经过认证的固定基地的飞行模拟器具有与认证飞机相同的生产驾驶舱组件（例如方向舵——踏板），但模拟器中的踏板经常断裂并且必须更换。如果您试图了解原因及在组件级别解决问题，您可以责怪并惩罚造成损坏的飞行员，培训飞行员如何处理模拟器，或投资改进踏板。如果您看一下组件（人员、硬件、软件）之间的相互作用，并试图理解为什么飞行员在模拟器中使用的踏板不同于真实飞机的踏板，你会得到一个不同的问题定义和解决方案。事实证明，当飞行员在模拟器中制动时，模拟器软件没有被编程来显示飞机停止时典型的小俯仰运动。停车时也能感受到这种运动。在飞机模拟器没有这种反馈的情况下，飞行员踩刹车的时间比需要的长，力度也比需要的大。出于这个原因，踏板在模拟器中断裂，但在真正的飞机中并没有出现这种情况。最好的解决方案是改进软件并为飞行员提供所需的反馈。

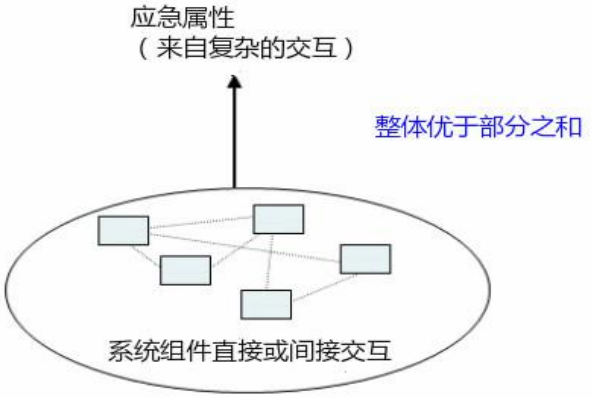
在所有这些情况下（以及其他数百个案例中），问题不是源于单个组件故障，而是源于系统工程过程中导致系统设计有缺陷的缺陷。分解分析无法识别这些事故原因，包括人为错误、软件需求错误和系统设计缺陷。我们需要不同的东西。作为 STPA 基础的这一不同理论基础被称为系统理论。

2. 系统理论

如上所述，工程中使用的系统理论是在第二次世界大战之后创建的，用于处理战后建立的系统的复杂性[3]。它也是为生物学而创建的，以便成功地理解生物系统的复杂性。在这些系统中，分离和分析单独的交互组件（子系统）会使系统整体的结果失真，因为组件行为以非显而易见的方式耦合。这些新想法的第一次工程用途是在 20 世纪 50 年代和 60 年代的导弹和早期预警系统中。

系统理论的一些独特方面如下：

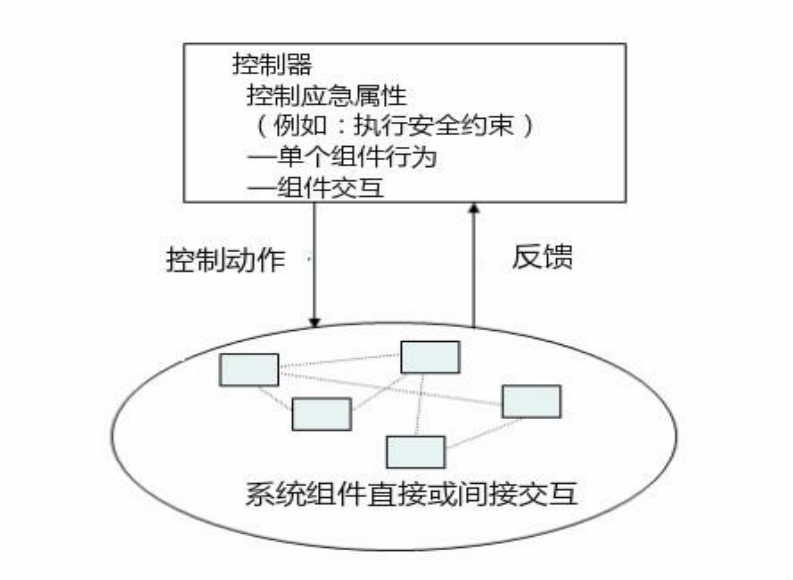
- 系统被视为一个整体，而不是其各部分的总和。你可能听过这样的公用语句：“整体不仅仅是它各部分的总和。”
- 应急属性是主要关注，这些属性不是各个组件的总和，而是在组件交互时“出现”。只有考虑到所有技术和社会方面，才能充分对待应急属性。
- 应急属性源于系统各部分之间的关系，即它们如何相互作用和组合在一起。



如果应急属性来自单个组件行为和组件之间的交互，那么控制应急属性（例如安全性、可维护性和可操作性）就需要控制各个组件的行为以及组件之间的交互。我们可以在图中添加一个控制器。这个控制器在系统上提供控制行为并获得反馈以确定控制行为的影响。如果这看起来像一个标准的反馈控制循环，你是绝对正确的。事实就是这样。

此控制器对系统行为进行限制。示例安全限制可能是飞机或汽车必须保持最小距离，深水井中的压力必须保持在安全水平以下，飞机必须保持足够的升力以保持空中飞行，除非着陆，必须防止意外引爆武器，以及有毒物质绝不能从工厂中释放。

[3] 如果您有兴趣了解更多信息，我们推荐 Peter Checkland，《系统思考，系统实践》，John Wiley＆Sons（1981），Gerald Weinberg，《一般系统思维导论》，John Wiley＆Sons（1975），也许还有历史兴趣，W. Ross Ashby，《控制论导论》，Chapman 和 Hall（1956）。



为使用一个简单的例子来解释上述模型，请考虑国家或国际空域。如果允许每家航空公司在任何时间飞行和使用任何航线去优化其航班时刻表，如果每个人都试图在下午 5 点在芝加哥、纽约或希思罗机场着陆，可能会导致混乱。为了避免这种冲突，引入了空中交通管制（ATC）来控制两个系统级应急属性：吞吐量和安全性。ATC 负责优化系统的整体吞吐量（可能无法优化每架飞机的航线）并保持飞机之间的充分间隔。

该模型包括我们现在在安全工程中所做的一切。控制被广义地解释。例如，可以通过设计（例如使用冗余、互锁、屏障或故障安全设计）来控制组件故障和不安全的交互。安全性也可以通过过程来控制，例如开发过程、制造过程和程序、维护过程和一般系统运行过程。最后，可以使用社会控制来控制安全，包括政府监管、文化、保险、法律和法院、或个人自身利益。通过设计社会或组织激励结构可以部分控制人类行为。

3. 什么是 STAMP?

STAMP[4]（基于系统-理论事故模型和过程）是基于系统理论的新事故因果关系模型的名称[5]，如前一章节所述，它为 STPA 提供了理论基础。它将传统的因果关系模型扩展到一系列直接相关的故障事件或组件故障之外，以包括更复杂的流程和系统组件之间的不安全交互，并且它是 STPA 和其他工具的基础。

在 STAMP 中，安全被视为动态控制问题而不是故障预防问题。STAMP 模型中没有遗漏任何原因，但包括更多原因，重点从防止故障变为对系统行为进行限制。

使用 STAMP 的一些优点：

- 它适用于非常复杂的系统，因为它自上而下而不是自下而上地工作。
- 它包括软件、人类、组织、安全文化等作为事故和其他类型损失的因果因素，而不必以不同方式或单独处理它们。
- 它允许创建更强大的工具，例如 STPA、事故分析（CAST）、识别和管理风险不断增加的领先指标、组织风险分析等。

由于 STAMP 适用于任何应急属性，因此 STPA 可用于任何系统属性，包括网络安全。

STAMP 不是一种分析方法。相反，它是关于事故如何发生的模型或一组假设。STAMP 是传统安全分析技术（如故障树分析、事件树分析、HAZOP，FMECA 及 HFACS）的故障链事件（或多米诺骨牌或瑞士奶酪切片，所有这些都基本相同）的替代。正如传统的分析方法是基于对故障链事件模型中事故发生原因的假设构建的，可以使用 STAMP 作为基础构建新的分析方法。请注意，由于故障链事件模型是 STAMP 的子集，因此基于 STAMP 构建的工具可以包含使用旧安全分析技术得出的所有结果的子集。

目前最广泛使用的两种基于 STAMP 的工具是 STPA（系统理论过程分析）和 CAST（基于系统理论的致因分析）。STPA 是一种主动分析方法，可分析开发过程中事故的潜在原因，从而消除或控制危险源。

CAST 是一种追溯分析方法，用于检查已发生的事故/事故征候，并确定所涉及的因果因素。本手册专注于 STPA 的使用。未来的类似手册计划用于 CAST。

---

[4] Nancy G. Leveson，《建造更安全的世界》，麻省理工学院出版社（2012）

[5] 作为 STAMP 基础的基本系统理论不应与复杂性理论相混淆，复杂性理论来源于基本系统理论，以解释自然界和社会组织中常见的自适应性非线性行为。对于工程，甚至组织的工程或设计，基本系统理论就足够了。更多信息，请参见附录 F。

4. 本手册的组织构架

本手册的第 2 章介绍了 STPA 的基础知识以及如何进行 STPA 分析。使用航空航天的例子，但其他行业的例子包括在附录中。

第 3 章到第 7 章介绍了如何在各种常见类型的工程活动中使用 STPA。由于 STPA 可以应用于任何应急（系统）属性，因此第 5 章将 STPA 应用于安全以外的其他方面，在这种情况下，即系统工程的有效性。虽然 STPA 在产业中的大多数应用到目前为止都涉及安全性，但公司现在开始将其用于其他系统属性，如质量、安全性和生产工程。

最后，第 8 章提供了有关如何将 STPA 集成到大型组织中的一些建议。附录提供了其他示例、分析组织安全管理系统的指南、一些关于为什么分解不适用于当今软件密集型复杂系统的附加说明（包括示例），以及对非工程师的一些基本概念的介绍。

第 2 章：如何进行基本的 STPA 分析

约翰 托马斯

1. STPA 方法概览

图 2.1 给出了基本 STPA 的步骤及其图解。

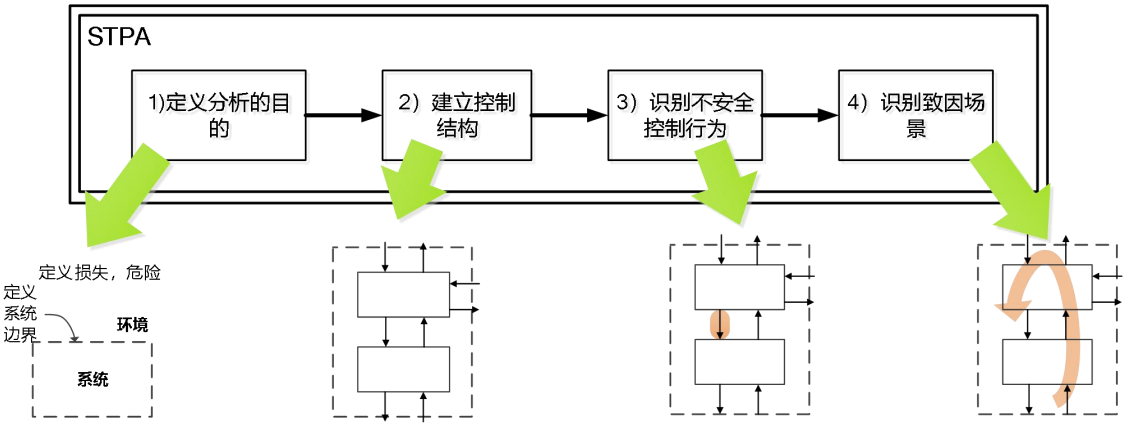


图 2.1： 基本 STPA 方法概览

任何一种分析方法的第一步都是定义分析目的。该分析所要避免何种损失？是否仅为达到传统的安全目标诸如防止人身伤害而应用 STPA，或是需要将 STPA 应用到更广泛的领域，如安保、隐私、性能或是其他系统特性上？被分析的系统是什么系统，系统边界在哪？这些问题以及其他基础性问题将在本步骤中得到答案。

第二步是要建立系统模型，称为控制结构。控制结构通过一套反馈控制回路为系统建模捕捉功能性关系及相互作用。控制结构通常起始于较为抽象的层级并通过迭代调整以捕捉更多系统细节。无论 STPA 应用于包括安全、安保、隐私或其他特性在内的任一方面，本步骤都相同。

第三步旨在分析控制结构中的控制行为，以验证此类行为如何导致第一步中所提到的损失。此类不安全控制行为用于生成系统的功能性要求与限制。无论 STPA 应用于包括安全、安保、隐私或其他特性在内的任一方面，本步骤也相同。

第四步主要是识别系统中可能出现不安全控制的原因。构建适当情境以解释以下内容：

1. 不正确反馈、不充分要求、设计错误、组件失效以及其他因素如何导致不安全控制行为并最终导致损失。
2. 没有恰当地遵守或执行所提供的安全控制行为如何导致损失。

一旦识别了这样的场景，即可用于生成其他需求、识别缓解措施、改进系统架构、提出设计建议以及新的设计决策（如在研发阶段使用 STPA）、评估/回顾现有的设计决策及识别差距（如在设计完成后使用 STPA）、定义测试用例并生成测试计划、形成风险领先指标以及本手册中后面章节提到的其他应用。

2. 定义分析目的

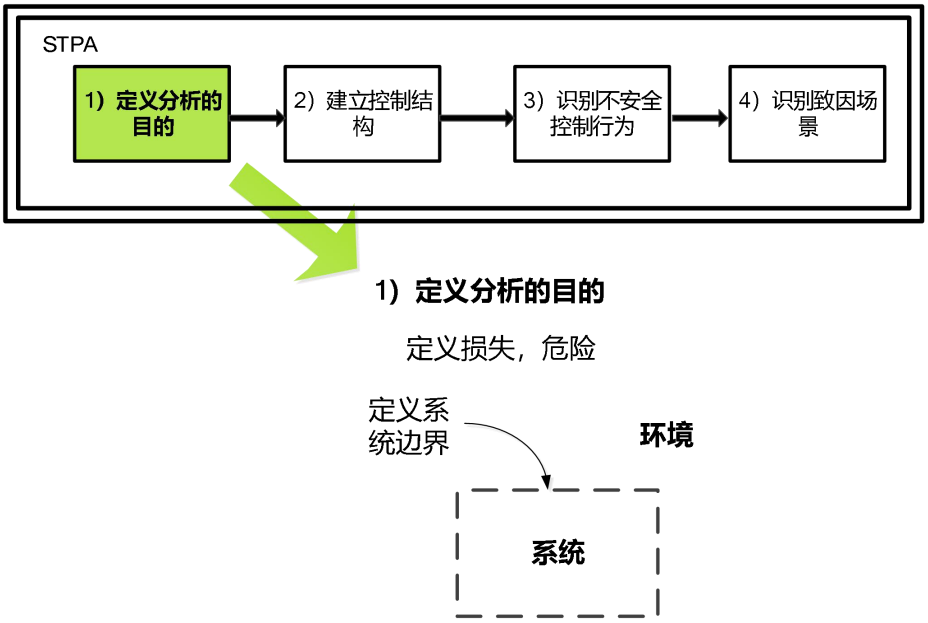


图 2.2：定义分析目的

如*图 2.2* 所示，应用 STPA 的第一步就是定义分析目的。定义分析目的分为四部分：

1. 定义损失（事故）
2. 识别系统层级危险
3. 确定系统层级安全约束
4. 提炼危险（可选）

图 2.3 总结了本节中提到的四个部分。

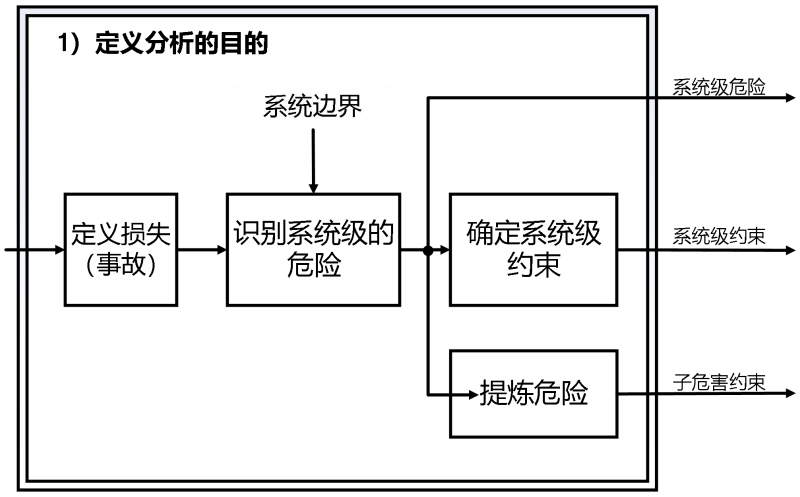


图 2.3：定义分析目的概览

a) 定义损失

定义：损失涉及对利益相关者具有价值的东西。损失可能包括失去生命或人身伤害、财产损失、环境污染、任务失败、声誉受损、敏感信息丢失或泄露或利益相

关者无法接受的其他损失。

不同行业中人们采用不同的词语来定义危险分析的目标，比如，防止事故发生、避免闪失或者预防不良事件。为消除不同行业从业者采用的多种词语所产生的误会，本章节即一律采用“损失”一词，STPA 的目标也是避免损失。

STPA 可针对利益相关者所不能接受的任何损失。如涉及多项损失，则可以根据轻重缓急进行排序。由于每个 STPA 结果都可追溯到一项或多项损失上，所以可以简单地根据所涉及的损失对分析结果进行主次排序。

在进行任何分析之前，利益相关者必须识别出他们需要着重分析的损失。所考虑的损失可能由管理、政府规章或是客户来定义。定义损失的通常方法包括以下方面：

1. 识别利益相关者，如用户、生产商、客户、运营方等。
2. 利益相关者识别他们在系统中的“利益”。即他们重视什么？例如，生命安全、在役机队、发电、交通等。他们的目标是什么？例如，保持可用机群、提供交通、提供医疗服务、提供电力等。
3. 将每项价值或目标转化成一项损失，如生命损失、飞机损失、电力损失、交通损失等。

以下清单给出了分析用户经常有意避免的一些损失：

L-1: 失去生命或遭受人身伤害

L-2: 交通工具丢失或受损

L-3: 交通工具外部物品丢失或受损

L-4: 任务失败（如交通任务、监控任务、科学任务、防卫任务等）

L-5: 客户满意度受损

L-6: 敏感信息受损

L-7: 环境受损

L-8: 电力供应受损

避免识别损失时常见错误的一些建议：

1. 损失包含利益相关者所不能接受的任何损失
2. 损失不应指向单一组件或特定原因诸如“人为因素”或“刹车失效”
3. 损失可能涉及系统设计者无法直接控制的环境方面内容
4. 记录任何特殊考虑或假设，如明确得以排除的损失。

损失定义结束后，下一步即为定义与这些损失相关的风险。

b) 识别系统级危险

定义：危险指的是一种系统状态或是一系列条件，在特定的最不利环境条件下，会导致事故（损失）。

定义：系统是能够共同作用以达成某种共同目的、目标或状态的一系列组件。系统可能包含子系统，也可能本身就是更加庞大系统的一个部分。

为了识别系统层级危险，首先要识别准备分析的系统及其系统边界。系统是分析员构想的一个抽象概念。必须明确系统中有什么，系统边界在哪里。从工程角度来说，用于分析目的来定义系统边界的最行之有效的方法就是将系统设计者控制下的系统部分包含进去。这也是区分风险与损失的主要原因——即损失涉及的方面可能是系统设计者或运营者只有部分控制权或根本没有控制权的方面。安全工程的目标在于消除或缓和分析下被分析系统中的危险的影响，因此保持一定层面的控制是必要的。图 2.4 将系统边界阐释为把系统与其环境相分离的某种抽象。

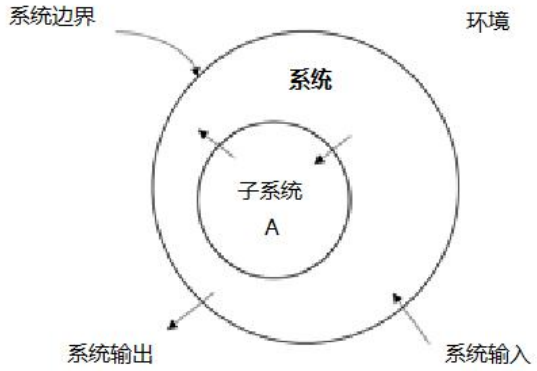


图 2.4：系统与系统边界、环境的关系

举例来说，试想一下核电厂。辐射物泄露、工厂与附近人口或城市的距离、风向都可能成为导致生命安全损失的重要因素。然而，工程师不能控制风向，也不能左右城市选址，但可以控制工厂内外的辐射物泄露情况（即系统级危险）。

一旦识别了系统及系统边界，下一步即为通过识别将会在最坏环境条件下导致损失的系统状况或状态来定义系统级危险。以下清单给出了一些系统级危险清单：

H-1: 飞机在飞行过程中违反最低间隔标准 [L-1, L-2, L-4, L-5]

H-2: 飞机机体完整性受损 [L-1, L-2, L-4, L-5]

H-3: 飞机脱离特定滑行道、跑道或地面机坪 [L-1, L-2, L-5]

H-4: 飞机与地面其他物体距离过近 [L-1, L-2, L-5]

H-5: 卫星无法收集科学数据 [L-4]

H-6: 车辆未能与地形或其他障碍物保持安全距离 [L-1, L-2, L-3, L-4]

H-7: 无人机（UAV）无法完成监视任务[L-4]

H-8: 核电厂危险品泄露 [L-1, L-4, L-7, L-8]

总的来说，一种危险可能导致一项或多项损失，对于每项危险均应追溯其可能导致的损失结果。这种追溯在危险描述后应妥善存档归类。以上实例显示的即为上一页损失实例的追溯过程。

定义系统级危险有三个基本标准：

- 危险指的是系统状态或条件（而不是组件级的原因或环境状态）
- 危险会在一些最坏情况下导致损失
- 危险必须描述出需要预防的状态或状况

首先，系统级危险指的是系统状况或状态，而不是设计者控制范围之外的外部环境状况。另外，定义系统级危险不应描述具体的组件原因如特定组件失效（例如液压泄露、刹车油不足等）。在本步骤中指向组件级原因会过度限制分析，导致在后续步骤中容易忽视其他不太明显的原因。因此，识别需要预防的系统级状态或条件（即危险），允许后续 STPA 步骤对组件级危险原因进行系统化识别。

其次，一定存在危险在最坏情况下导致损失的情况。该要求未必保证一种危险一定会导致损失。例如，化工厂可能会泄露有毒物质，但风向及天气条件也可能会使附近人员及人口密集区域免受有毒物质的侵害。然而，在最坏情况环境下，有毒物质可能会被带到人口密集区从而造成损失。

最后，危险即为需要预防的状态或条件。“飞行中飞机”是一种在最坏情况环境下可能会导致损失的系统状态，但它不是一个需要消除或阻止的情况（否则我们也不会去造飞机了）。危险应为一种需要预防的状态，一种不想让系统陷入的状态——不是系统为达成其目标而必须正常的那种状态。

c) 识别系统级危险时的常见错误

混淆危险与危险原因

识别系统级危险时的常见错误之一就是混淆危险与危险原因。例如，“刹车失效”、“刹车失效但未警告”、“操作人员分心”、“发动机失效”及“液压泄露”均为潜在危险原因而非系统级危险。为避免这类错误，需确保所识别的危险不涉及系统单一组件如刹车、发动机、液压管路等。反之，危险指的是整个系统和系统状况。换句话说，需检查每项危险是否包含以下信息：

<危险说明> = <系统> & <不安全条件> & <导致的损失>

例：H-1: 飞机在飞行过程中违反最低间隔标准 [L-1, L-2, L-4, L-5]

重点不在于顺序准确——也可以简单地表示成“违反飞机间最低间隔标准 [L-1, L-2, L-4, L-5]”。重要的是系统级风险包含这些要素。

很多危险涵盖了过多非必要细节

与损失一样，对所涉及的系统级危险数量没有硬性限制。根据经验，如存在七项到十项以上的系统级风险，可以考虑将危险分类组合，以列出清单，方便管理。但可能会输入过多非必要细节，造成清单管理不便，评审困难，为识别遗漏项制造障碍。因此，需从较为抽象且便于管理的系统级危险集开始，在后续需要的情况下，

不断提炼为多个子危险（参见下面提炼危险部分）。

表意不清或循环用词

系统级危险就是在系统级准确定义什么是“不安全”。常见的错误就是在危险本身中使用“不安全”一词。这样做只是生成了一个循环定义并没有为分析增加任何信息或价值。例如， 可能很容易写出“H-1：飞机正经历不安全飞行 [L-1]”。之所以容易是因为用词听起来确实很危险——不安全飞行的定义不就是有风险吗？问题在于这类用词过于模糊无法具体化不安全的实际情况到底是什么。掉进同一个陷阱的类似说法还有“H-1：飞机正经历不安全状况 [L-1]”，单纯执着于剩下的分析而忽视了重要情况。解决方法很简单，就是避免在危险本身中使用“不安全”一词，而要将“不安全”准确具体化——即何种系统状况或状态导致不安全？例如，失控的飞机或与其他飞机距离过近的飞机是不安全的。所以，以这种方式将实际情况具体化对后续的 STPA 步骤是非常有帮助的。

混淆危险与失效

擅长其他危险分析方法的专业人员撰写 STPA 危险时偶尔也会掉入陷阱，即根据特定技术功能描述潜在偏离或是描述组件失效。你可能对传统的技术比较熟悉，即从寻找技术系统中的一系列偏离、错误或功能失效起始。为了在 STPA 中定义更加广泛的集原因，我们不能假设所定义的及所规定的功能就是安全正确的、不能假设操作人员能完全按照预期执行、不能假设自动化行为不会导致人为错误或误会、不能假设不会出现非正常情况、也不能假设技术设计，规格以及要求均准确无误。例如，危险“控制飞机撞向地形”可纳入 STPA，但在单纯检查技术功能性失效时也可能被忽略。

STPA 中的危险识别是针对固有的不安全系统状况和状态——与其触发原因无关。事实上，系统危险应在一个较高的层级上进行规定，而在这个层级上无需区分与技术失效、设计错误、要求漏洞或人为程序或交流不当相关的各种原因。

在评审危险时应评审哪些内容？

识别危险时常见错误的提示：

- 危险不应涉及系统单一组件
- 所有危险应参考整个系统和系统状况
- 危险应参考可控或可由系统设计者和操作者管理的因素
- 所有危险应描述需要预防的系统级状况
- 危险数量应相对较小，通常情况下不多于 7 到 10 项
- 危险中不应含有表意不清或导致循环效应的词语，如“不安全”、“意外”、“偶然”等。

注意 STPA 是一种迭代方法，在此阶段危险无须固定。后续 STPA 步骤会暴露新的危险，危险清单可根据需要重新进行评审并修改。

d) 定义系统级约束

定义：系统级约束规定了需要满足以预防危险（最终防止损失）的系统条件或行为

一旦识别了系统级危险，需直接识别必须强制实施的系统级约束：只需简单地反转条件。

<危险> = <系统> & <不安全条件> & <导致的损失>

<安全约束> = <系统> & <执行条件> & <与危险的关系>

H-1:飞机违反最低间隔标准 [L-1, L-2, L-4, L-5]

SC-1:飞机必须满足与其他飞机及物体的最低间隔标准 [H-1]

H-2:飞机机体完整性受损 [L-1, L-2, L-4, L-5]

SC-2:飞在最坏情况下，必须保持飞机完整性 [H-2]

每项约束可追溯到一项或多项危险上，每项危险可追溯到一项或多项损失上。总的来说，追溯过程无需一对一；单一约束也可用于预防多项危险，多项约束也可能只与一项危险有关，每项危险可导致一项或多项损失。

约束也可定义一旦确定发生了危险，如何使系统损失最小。例如，如果飞机确实违反了最低间隔标准，那么必须检测到违反行为随即采取措施防止飞机碰撞。如果一座化工厂真的释放了有毒化学物质，必须检测有毒环境并采取相应措施。此类约束可大致表示为：

<安全约束> = 如果 <危险> 发生， 那么<为防止或最小化损失所采取的措施>

SC-3: 如果飞机违反最低间隔标准，那么必须检测到违反行为随即采取措施防止飞机碰撞

安全约束不应对特定解决方法和实施策略进行规定。例如，SC-3 仅仅声明必须检测违反行为且提出某种方法以防止碰撞，而不是规定具体解决方法比如安全防撞系统。在早期阶段规定特定解决方法通常为时尚早，会导致后期忽略更加有效的解决方法。

STPA 的其他分析将系统化地识别可能违反此类约束且导致系统级危险和致因场景。

e) 提炼系统级危险（可选）

一旦系统级危险清单完成识别及评审，在适当情况下，可对这些危险进行提炼，分化为多个子危险。对于很多 STPA 应用，子危险未必存在[6]，但子危险对于大型分析和复杂应用有着很大帮助，因为它们能够引导接下来的步骤，如建立控制结构建模（参见下一节控制结构部分）。

提炼系统级危险的第一步就是识别需要控制以预防系统危险的基本系统程序或活动。例如，假设我们已经为商用航空识别了系统级危险：

H-4: 飞机与地面其他物体距离过近 [L-1, L-2, L-5]

导出子危险的一种方法就是询问：为避免这一危险我们需要控制的是什么？为控制地面飞机，我们需要控制飞机的减速、加速以及转向。如果以上方面没有得到充分控制（比如未充分减速），就可能导致系统层级危险。

以下子危险可由 H-4 导出：

H-4: 飞机与地面其他物体距离过近 [L-1, L-2, L-5]

减速

H-4.1: 在降落、起飞中断或滑行期间未充分减速

H-4.2: 不对称减速使飞机转向其他物体

H-4.3: 起飞速度达到 V1 后出现减速

加速

H-4.4: 滑行期间过度加速

H-4.5: 不对称加速使飞机转向其他物体

H-4.6: 起飞期间未充分加速

H-4.7: 降落或停靠期间出现加速

H-4.8: 中断起飞期间持续加速

转向

H-4.9: 沿滑行道、跑道或机坪路径转向不充分

H-4.10: 转向不充分使飞机脱离滑行道、跑道或机坪路径

以上每项子危险均可用于生成更加具体的约束。表 2.1 显示的是基于以上子危

[6] 刚刚接触 STPA 的人可能会觉得子危险在无需此步骤的较小应用的起始阶段很有帮助。

险所生成的减速相关约束。

表 2.1：从子危险中生成特定程序约束

H-4 导出的子危险	**安全约束实例**
H-4.1: 在降落、中断起飞或滑行期间未充分减速	SC-6.1: 减速必须在降落或起飞中断后 x 秒内以至少 x m/s^2 的加速度进行
H-4.2: 不对称减速使飞机转向其他物体	SC-6.2: 采用不对称减速不得导致方向失控或飞机偏离滑行道、跑道或机坪。
H-4.3: 起飞速度达到 V1 后出现减速	SC-6.3: 起飞速度达到 V1 后不得减速

3. 构建控制结构

如*图 2.5* 所示，STPA 的第二步就是建立分层控制结构模型。

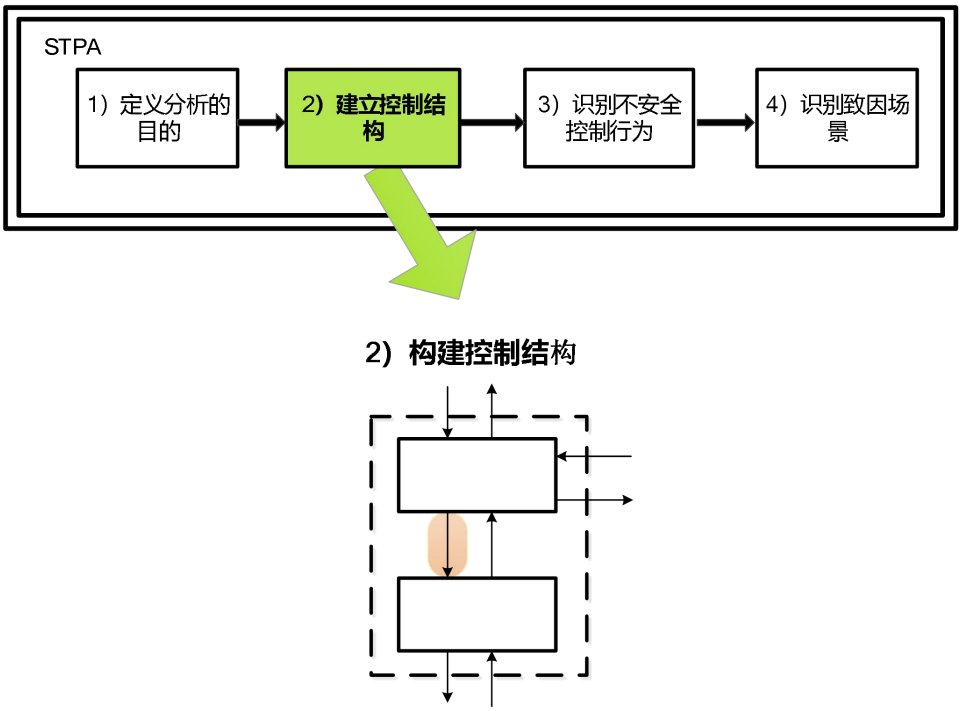


图 2.5：构建控制结构

a) 何为控制结构?

定义：分层控制结构指的是由反馈控制回路组成的系统模型。有效的控制结构可以将约束施加到整个系统的行为上。

分层控制结构由*图 2.6* 中所示的反馈控制回路组成。总的来说，控制者可通过提供控制行为来控制某个程序并将约束施加在受控过程的行为上。控制算法代表了控制者的决策过程——即决定提供哪些控制行为。控制者同时有其过程模型，代表

着其以往决策时的内部想法。过程模型可包含有关受控过程或系统/环境的其他相关方面的想法。过程模型可通过用于观察受控过程的反馈进行部分更新。

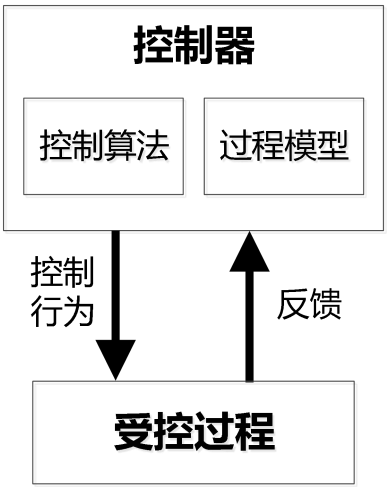


图 2.6：一般控制回路

图 2.6 所示的任何一点均可能发生问题。例如，过程模型可能与实际情况不符（比如管制员以为飞机正在下降，但实际上飞机在爬升，或是以为车辆已经停好但实际上在倒车，又或是以为机场跑道空闲等等）可能导致不安全的控制行为。传感器失效也可能导致错误反馈并导致不安全行为。设计有时可能会忽略必要的反馈或出现延迟反馈的情况，这会导致过程模型缺陷，并且导致非安全行为。接下来，STPA 将提供一种能够系统化识别这些和其他可能导致损失的场景的方法。

图 2.6 中的一般控制回路可用于解释并预测可能导致损失的复杂人机交互——这也是现代工程学的两大挑战。就人而言，人的过程模型一般称作心智模型，而控制算法则可称为操作步骤或决策规则，但基本概念是相同的。

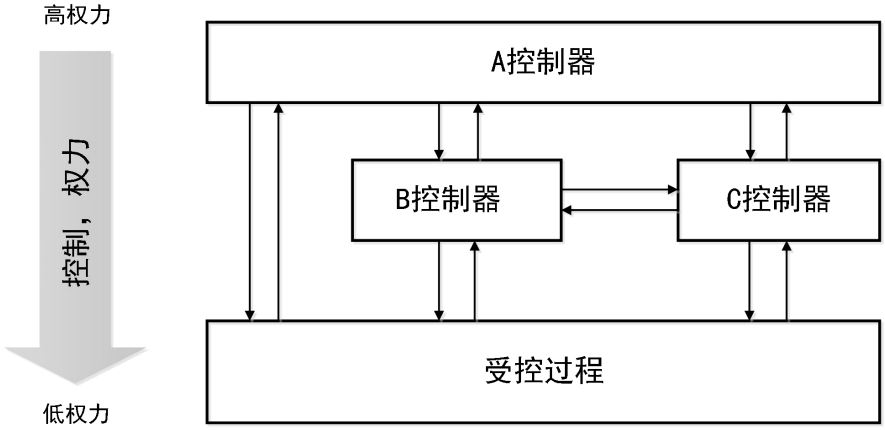


图 2.7：一般层级控制结构

当然，一般情况下大多系统都有几个重叠且交互控制回路。如*图 2.7* 所示，多个交互控制回路可通过一个分层控制结构进行建模。*图 2.8* 为应用在航空领域中的一个简单实例。

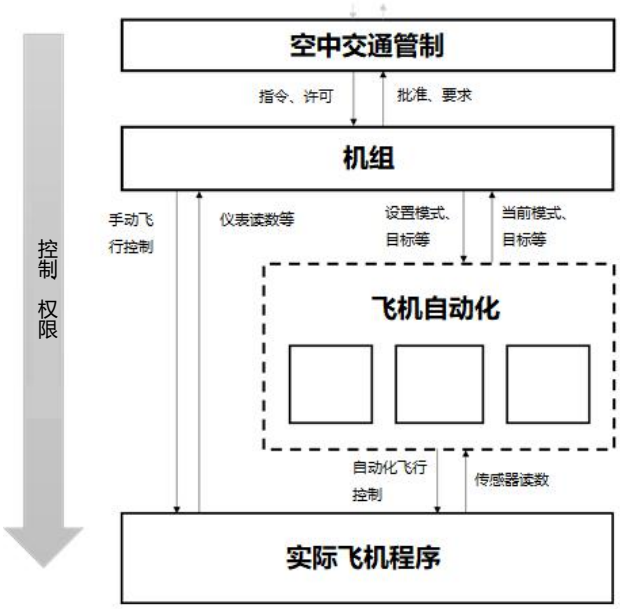


图 2.8：层级控制结构在航空领域中的简单实例

总的来说，一个分层控制结构包括至少以下五类要素[7]：

- 控制器
- 控制行为
- 反馈
- 来自组件（除控制器和反馈以外的）的其他输入或输出指令
- 受控过程

分层控制结构的纵向具有一定意义：象征系统内的控制与权威。纵向分布代表着控制层级，由顶端的高级控制器到底部的最低级个体。每个个体对其下方直接个体都有完全控制权，同样，每个个体都必须服从其上方直接个体的控制与权威。举例来说，*图 2.8* 中的飞机自动化可以作为控制器给飞机物理系统发送控制行为并监测反馈。同时，飞机自动化驾驶还是一个受控过程，必须接收并执行来自机组的控制行为并给机组发送反馈。

换句话说，所有下向箭头代表控制行为（命令），而上向箭头代表反馈。这些规则可以帮助把握复杂性并使得控制关系及反馈回路更加容易辨识[8]。在一些情况下，简单地绘制一个控制层级图解可以使之前没有发现的一些漏洞展露无遗。比如，一些能够提供控制行为的个体并不具备选择安全控制行为的必要反馈，而反馈可能会被发送至没有能力应对反馈的个体；多个控制器可能会给同一个组件提供互相矛盾的命令，而无法检测或解决矛盾等。如果在此阶段无法发现漏洞，不必担心，后续 STPA 方法将对上述及其他问题进行系统化识别。

---

[7] 控制结构也可包括执行器和传感器，但通常在后续的情境鉴定步骤添加

[8] 控制结构也可以按照需要由左到右进行绘制，此处为简化讨论仅在注释中提及。

b) 普遍疑问

控制结构不是实体模型

STPA 中使用的分层控制结构是功能模型，而不是诸如物理框图、原理图或是管线及仪表图之类的实体模型。连接线表示个体间可发送诸如命令及反馈的信息——不一定需要对应实体连接。比如，机组和空中交通管制间的交互并非物理上的交互，但存在于功能性控制结构模型中。

控制结构不是可执行的模型

控制结构并不是可执行的或模拟的模型。实际上，控制结构通常包括的一些并不存在的可执行模型组件（如人员）。反之，STPA 可用于谨慎生成必要的行为限制、要求以及执行特定系统特性所需的规格。比如，图 2.8 中的控制结构没有假设空中交通管制在机组需要的情况下总会提供指令信息，或总是有能力发送指令（例如，无线电总是处于运行状态），也没有假设空中交通管制发出的指令总是正确的，或者机组人员总能够遵照所接收的指令执行。仅仅表示这样一个允许空中交通管制向机组发送指令的系统已经/将要被创造出来。STPA 的下一步骤将严格检验不安全行为发生的方式，包括已发送的指令未被接收、所发送指令存在非安全因素等。尽管控制结构本身不是一个可执行的模型，但 STPA 方法将产生能够用于生成可执行模型及规格的精准的要求及其他输出信息。

控制结构没有假设服从性

不要将控制器及控制行为与服从性混淆。就算控制器发送了控制行为，也不意味着在实践中控制行为会被遵照执行。同样，就算反馈路径包括在控制结构中，也不意味着在实践中总会发送反馈或反馈内容总是精确无误。控制结构中的控制行为及反馈仅仅表示这样一个用于发送此类信息的机制将被建立（也就是该机制将囊括在系统设计中），并没有暗示或假设控制器及程序在实践中将如何表现。事实上，STPA 的一个主要目标就是分析控制结构并预测每个要素在非安全及可能出现的意外情况下会如何表现。

运用抽象概念把握复杂性

在任何危险分析中，最大的挑战之一莫过于把握系统复杂性。控制结构运用抽象概念以多种方式帮助把握复杂性。比如，对于商业飞行来说，机组成员一般由两个或三个飞行员组成。因此，我们可以将机组视为一个群体，能够共同发送控制行为并接收反馈，而不是在一开始就将三个飞行员独立分隔开来，避免控制结构的杂乱。同理，还可以从一个较为抽象的层级开始，将机组所控制的飞机自动化及实体

程序作为控制层级的两级，而不是将每个独立的飞机子系统详尽罗列出来。

抽象的原则也可以应用到控制结构的命令与反馈路径上。可以从较为宽泛的活动如爬升机动开始，而不是将驾驶舱中每个独立的按钮、开关、操纵杆列出来。随后，我们可以将宽泛的活动提炼细化到俯仰、推进或适合的其他命令。该原则在早期开发阶段即未涉及单个命令和传感器时尤为有效。

控制行为路径可能包括控制器作用于受控过程（称为执行器）所通过的机制以及控制器从受控过程（称为执行器）感知的反馈所通过的机制。这些细节通常在开始创建控制结构时就被抽象化了，但在后续的情境创建步骤中，将改进控制结构以涵盖执行器与传感器。

还有一种抽象方式也能用于控制结构。试想一下由机组发送到飞机自动化装置的飞行指令。在遥控 UAV（无人机）应用中，这些命令需要通过很多不同的组件——从命令控制台上的一个实体按钮，通过一个进行数字编码的嵌入系统，再到网络交换机、无线电广播发射机、卫星最后到无人机上的无线电接收器。其实没有必要在初始控制结构中控制结构中体现所有步骤——真正重要的是遥控飞行员将通过何种方式向无人机发送飞行命令。

实际上，应用 STPA 的最高效方式是在设计决策完成前，涉及此类细节前开始进行。上述抽象控制结构可用于开始 STPA 及识别通讯路径及系统其他部分的要求与限制。随后，STPA 结果可用于导出体系构架、进行初始准备及细节设计，进行实施决策并完善控制结构。即使已经知道了细节，并且已经做出了设计决策，在分析更详细的控制结构模型之前，首先在更高的抽象级别上应用 STPA 可以帮助更快地提供结果并识别更广泛的问题。

c) 构建控制结构

控制结构建模一般起始于抽象控制结构并反复添加细节。在很多情况下，系统内的控制结构与控制回路可能较为明显或可从以往应用中进行再利用。如果控制结构不明显，以下指南将提供一种方法来为功能性控制结构建模并添加细节。附录 B 中涵盖了不同行业所使用的控制结构的实例。

方法之一，着手识别用来执行约束并防范已经识别出的危险所需的基本子系统。比如，我们已经生成了与不充分减速有关的危险及约束。这些约束可以通过应用刹车子系统、反推等其他子系统执行。*图 2.9* 中显示的是带有这些子系统的初始控制结构。

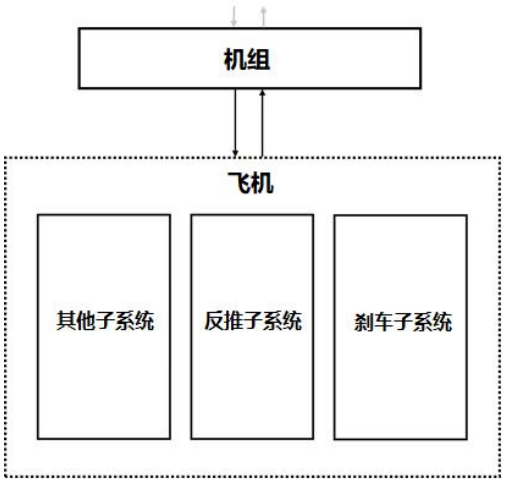


图 2.9：调整后带有子系统的控制结构实例

一旦子系统识别完成，我们就可以通过定义子系统的受控方式来改进控制结构。在此例中，即为完善刹车子系统。那么刹车子系统仅仅直接且手动受控于机组人员吗？控制单元、自动化控制器或其他在场人员也能控制刹车吗？如果将 STPA 应用到后续开发阶段，这些问题的答案就自然水落石出了。而如果 STPA 仅被应用于早期概念开发，那么上述危险和约束也可用于指导此类决策。例如，我们可将自动化刹车控制器加入刹车子系统以实现着陆或中断起飞期间自动刹车。

图 2.10 显示的是在刹车子系统中加入刹车系统控制单元（BSCU）改进后的控制结构。请注意在为控制结构加入更多细节时我们需要谨慎“放大细节”。

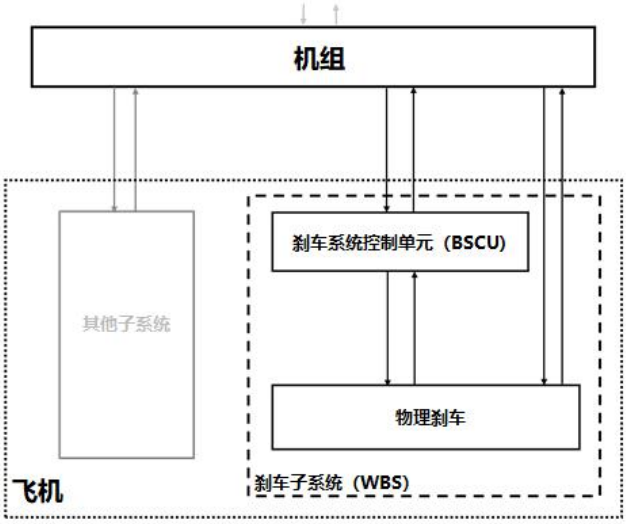


图 2.10：添加了子系统控制器的已调整控制结构

一旦控制器经过识别，系统工程师即可分配职责。这些职责也是安全约束的改进版本——即个体需要共同完成哪些职责以保证安全约束的执行？例如，刹车系统控制单元（BSCU）负责在刹车导致打滑时自动暂停刹车（防打滑功能）而机组则负责决策何时使用刹车。刹车相关职责实例如下：

物理刹车

o R-1:接收到刹车系统控制单元（BSCU）或机组指令时刹车减速[SC-6.1]

刹车系统控制单元（BSCU）

o R-2:应机组要求刹车[SC-.1]

o R-3:在打滑情况下暂停刹车（防打滑）[SC-6.2]

o R-4:在着陆或中断起飞时自动刹车（Autobrake） [SC-6.1]

机组

o R-5:决策何时需要刹车[SC-6.1, SC-6.3]

o R-6:决策如何刹车：自动刹车、常规刹车或手动刹车 [SC-6.1]

o R-7:配置刹车系统控制单元（BSCU）及 Autobrake 以准备刹车[SC-6.1]

o R-8:监控刹车并关闭刹车系统控制单元（BSCU），在出现故障时手动刹车[SC-6.1, SC-6.2]

接下来，每个控制器的控制行为可根据这些职责进行定义。例如，机组需要有能力发送手动刹车控制行为以满足 R-5 和 R-6。他们需要一种方法来启动并设置刹车系统控制单元（BSCU）以满足 R-6 和 R-7。也可能需要关闭刹车系统控制单元（BSCU）以满足 R-8。*图 2.11* 显示的是根据职责标注控制行为进行修正后的控制结构。

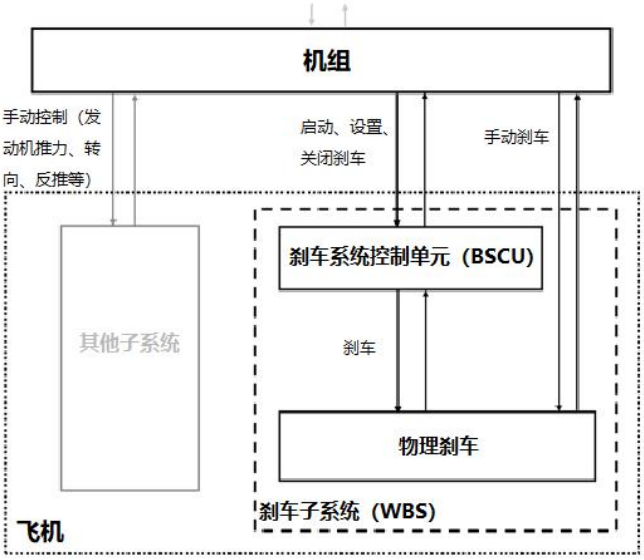


图 2.11：程序分配至子系统后的已调整控制结构

那么此时，控制器和控制行为已经得以识别并标注，但如何反馈呢？反馈可根据控制行为及职责导出，首先需要识别控制器决策所需的过程模型。其次，识别形成精确的过程模型所需的反馈及其他信息。例如，R-3 中规定刹车系统控制单元（BSCU）需要在打滑时暂停刹车。因此，刹车系统控制单元（BSCU）就需要知道是否出现打滑（该信息应包含在刹车系统控制单元（BSCU）过程模型中）。那么检测打滑需要什么反馈呢？可参考机轮转速反馈。同理，R-8 规定机组在出现故障时

可能需要关闭刹车系统控制单元（BSCU）。因此，机组就需要知道刹车系统控制单元（BSCU）是否出现故障（同样，该信息应包含在过程模型中）。那么机组检测故障需要什么反馈呢？可参考刹车系统控制单元（BSCU）反馈。为了在着陆或中断起飞（R-4）时实现自动刹车，刹车系统控制单元（BSCU）需要知道飞机何时着陆或何时出现中断起飞（该信息应包含在过程模型中）。可参考机轮负重开关及其他输入信息来检测着陆及中断起飞情况。下面*表 2.2* 显示的是由职责生成反馈的方式。

表 2.2：由职责生成反馈的方式实例

刹车系统控制单元（BSCU）职责	**过程模型**	**反馈**
应机组要求急刹车[SC-6.1]	机组要求下的刹车	踩刹车踏板
在打滑情况下暂停刹车（防打滑）[SC-6.2]	飞机正在打滑	机轮转速 惯性参考单元
在着陆或起飞中断（RTO）时自动刹车（Autobrake） [SC6.1]	飞机着陆 起飞中断	机轮负重 油门杆角度

另外，还可通过运用职责再次“放大”以添加更多细节进而进一步调整控制结构。比如，物理刹车负责收到指令后的机轮减速（R-1）。这个步骤也可以通过液压完成。R-6 显示刹车系统控制单元（BSCU）需要执行常规和自动化刹车（Autobrake）两种命令。刹车系统控制单元（BSCU）中的两个控制器可用于控制这两个行为：自动刹车控制器和液压控制器。

图 2.12 显示的是带有已标注反馈及已识别刹车系统控制单元（BSCU）内部控制器的控制结构。

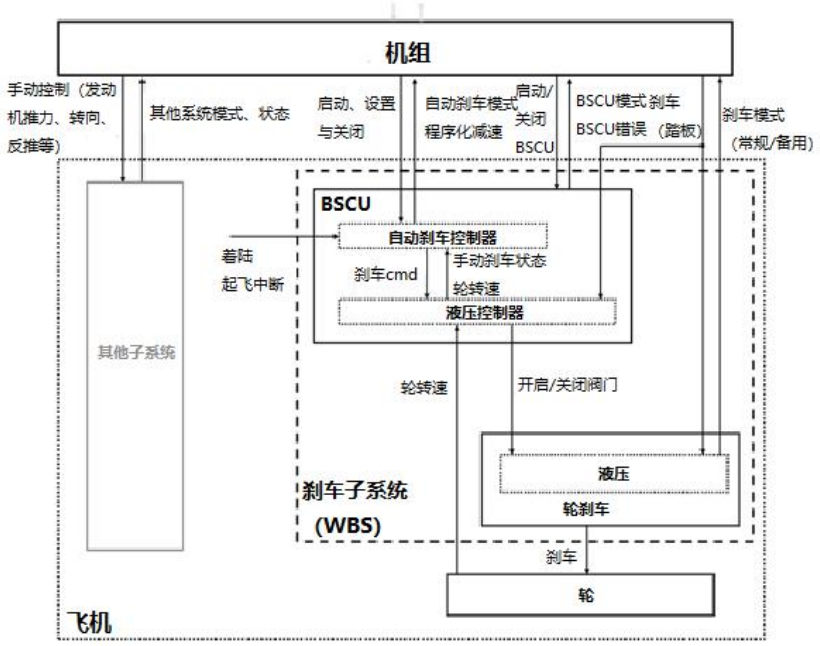


图 2.12：根据职责调整后的控制结构

d) 构建控制结构时的常见问题

控制结构需要在在进程开始前完成吗?

如果在开发完成前早期应用 STPA，可能会丢失某些信息并且控制结构也会不完整。分析人员可以将不完整的控制结构和 STPA 作为分析的第一步，STPA 将帮助识别可能丢失的反馈、控制及其他差距，这样即可调整控制结构使其达到与开发相一致的状态。进行下一步所需的最低限度包含至少一个控制器、控制行为及受控过程。然而，如果相关信息不是从控制结构中故意丢失，总体来说 STPA 将会更加简便更加高效。

如果在与控制器、控制行为及反馈相关的后续开发中应用 STPA，那么不存在从控制结构中故意忽略高级信息的理由。STPA 还可应用于设计，因为 STPA 可以识别如关键反馈此类被忽略的潜在漏洞。

标记说明应具体到什么程度?

在控制结构中，避免使用模糊或容易导致歧义的标记，如把所有控制行为都简单标记为“命令”，或把反馈都简单标记为“反馈”或“状态”，把控制器都简单标记为“计算机”或“控制器”。控制行为标记应指示出命令类型（在了解的情况下），如“开/关阀门”，反馈标记应指示出发送信息类型（在了解的情况下），如“机轮转速”。此时与用于发送控制行为及反馈的特定实体媒介没有关系——重要的是能够被发送的功能性信息。比如，使用“刹车系统控制单元（BSCU）开/关”而不是“刹车系统控制单元（BSCU）按钮”、“共享总线”、“编码数字包”。同样，控制器标记应指示出行为的功能性类型或控制器的角色，而不是具体操作。比如，可以使用的控制器标注有“自动刹车控制器”而不是“单板机”，无论飞行功能是在空中或地面进行，都要使用“机组”或“飞行员”标记。

控制结构应该包括所有执行器和传感器吗?

启动 STPA 时，无需特定执行器与传感器，所以暂时还不需要包括进来（会在后续步骤中加入）。为了把控复杂性，需要等到 STPA 后续步骤中情境创建需要考虑这些细节时再添加进来。例如，*图 2.12* 中的刹车子系统控制行为与反馈需要配合不同种类的电机阀门或通过使用其他完全不同的方式来进行操作。这一阶段重要的是所提供的命令及反馈的类型，而不是特定操作（可了解也可不了解）。

物理过程和交互如何纳入控制结构?

与任何模型一样，控制结构模型在抽象化或去强调其他方面时也在强调现实世界的特定方面。控制结构会强调功能性关系及功能性交互，这在识别诸如设计漏洞、

要求漏洞、人为错误、软件错误甚至传统实体组件失效等问题时是非常有效果的。控制结构模型一般不会捕捉单纯的物理或几何关系，如组件间或火焰传播的物理距离。

受控物理程序通常是在控制结构的最低层级进行说明规定，而该层级以上的所有层级则规定说明用于决策和直接或间接控制物理程序的功能性控制器。

控制结构需要按照线性层级展开吗?

不需要。对于一些系统来说，它们的控制层级可能类似梯形的带有清晰纵向线性层级。而其他系统的控制层级可能在同一纵向层级包括多个互不相控的控制器。控制器可能都控制同一个物理程序或者也可能控制不同程序。

同一层级的控制器也可能在控制/反馈关系外互相交流。这样的交流可在控制器图示中以横向箭头表示。总的来说，控制结构中的交互（箭头）可分为三类：控制行为、反馈以及其他信息。

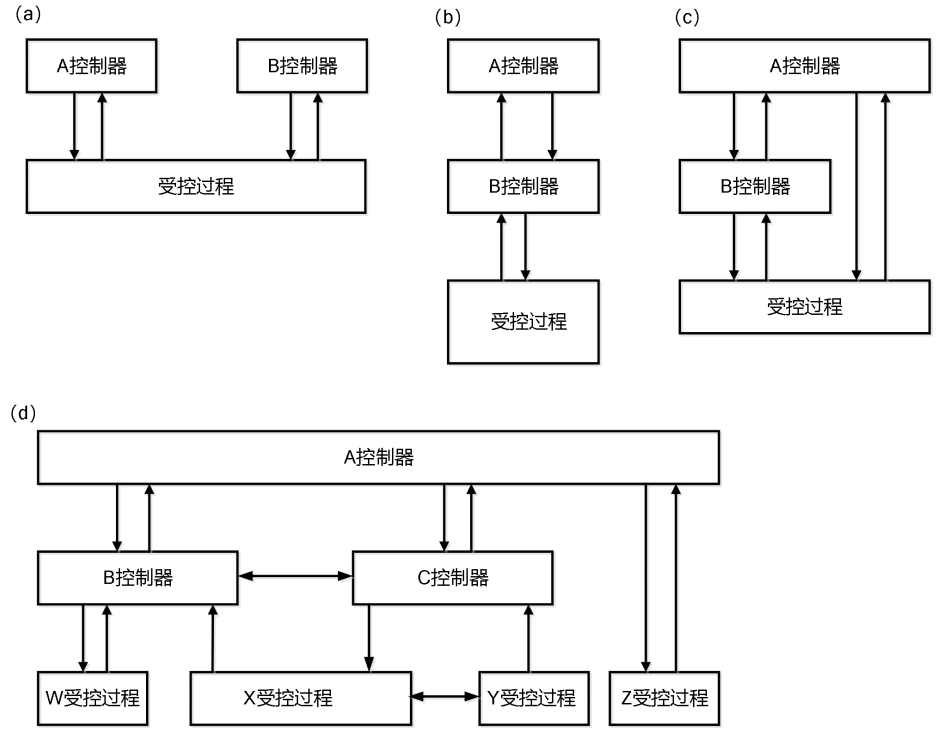


图 2.13：不同控制结构类型

图 2.13 显示的是几种不同类型的控制结构模型，每种都具备在设计或分析一个系统时需要考虑的不同优势及所需权衡的方面。（a）中的控制结构显示的是两个控制器并行运行，控制并监控同一个过程。控制器之间不存在交互且不存在控制关系（可能导致 STPA 将要识别的情境）。在控制器需要快速行动或独立于其他控制器的情况下，这种结构具有优势——例如，可以为若干名操作员提供在发现不安全条件下直接终止操作的能力，无需经由命令链审批。然而，该结构可能会给协调带来挑战，如混淆与其他控制器活动的责任与职责（后续步骤将对此及其他问题进行详细

分析）

（b）中的控制结构显示的是一个线性的控制层级，类似一个清晰的命令链。该结构能够帮助解决协调问题并澄清每个控制器的职责区别，但可能会给控制器 A 的反馈及控制行为反应时间带来挑战，也更加依赖于控制器 B 的运行。

控制行为及反馈可能会“僭越”——因为没有防止控制器与低层或高层进行交互的模型限制。例如，高层控制器可以绕过其他控制器并直接控制或监控低级程序，如（c）所示。控制器 A 可代表飞行员，而 A 常使用控制器 B 中的自动化来作动刹车。然而，在紧急情况下，飞行员可以绕过控制器 B 并直接（手动）作动刹车。

如（d）中的更复杂的结构也是可能出现的。控制器与过程间并不存在一对一的控制要求。一个控制器可控制一个或多个过程，同时一个过程也可被零个或多个控制器控制。一般情况下，每个控制行为路径会与平行的反馈路径匹配，但也有例外。例如，控制器 C 可能不可以直接控制过程 Y。但是它可以控制过程 X 并监控过程 X 对于过程 Y 的影响。

如何分辨谁控制谁？

在很多情况下，控制关系简单明了，就像一个主管对下属下达指令或电脑通过发送命令来开启/关闭阀门一样。控制关系也可以不这么直接，类似经理分配优先事项、航空公司提供标准操作程序等。这都是不同形式的控制。

如之前所言，控制不同于服从。就算控制关系确实存在，也并不意味着控制行为能够被充分执行。实际上，我们也不总是希望所有控制行为总是都能被服从；在意外情况下，为了避免损失，也可以出于正确的理由而忽视控制行为。STPA 并不假设服从性，此类问题会在后续步骤中得到仔细检验。

同样，引起另一组件反应或影响其行为的能力也并不意味着控制。开启阀门的控制行为可能会导致阀门开启，但相反，指示高温的反馈信号也可能促使控制器开启风扇。因此，引起另一组件反应或影响其行为的能力并不足以区分控制行为和反馈。

控制涉及的是做出有目的的决策以达成目标。提供反馈时，可以不了解（或不负责）特定高级目标，但控制行为的出现一定是为了实现某个目标。控制一般也涉及监督——监控低层个体，形成更好的能力或收集更多信息来监督它们的行为或影响，引导它们或干涉它们以确保高级目标的达成。

控制层级与目标和责任的层级息息相关。当检测到着陆时，上面的自动化刹车系统控制单元（BSCU）可能只能在有限职责范围内触发刹车。而飞行机组具有高级职责如决定何时着陆，还有高级目标如避免硬着陆。在实现此类高层机组目标时，刹车系统控制单元（BSCU）可能是一个必要但不充分的元素——机组必须监督刹车系统控制单元（BSCU）及很多其他组件以达成高级目标。同样，空中交通管制

也有高级职责比如确保多个飞机间的最低间隔，还有高级目标如最大化航班间的吞吐量。单个飞行机组指示空中交通管制用于达成这些目标的一个要素，甚至机组自己都没有意识到他们如何促进了高级目标的达成或空中交通管制是否达成了这些高级目标。

最后，控制与权力紧密相关。设想一下空中交通管制与飞行员的关系。空中交通管制对飞行员具有权威性，并向飞行员发送指令及许可，而飞行员必须服从。但飞行员对空中交通管制没有权威性，他们也不能命令空中交通管制服从。飞行员可以宣布紧急状态且空中交通管制需要对该反馈进行适当回应——就算飞行员能够触发反应，也不意味着飞行员能够控制空中交通管制。在必要情况下，为了确保飞行安全，飞行员也可以不服从空中交通管制的指令，但服从同样区别于控制。但飞行员对其飞机具有最终和直接控制权，但对空中交通管制没有最终权威，也不能管理整个空域。空中交通管制通过发布指令和许可间接控制飞机，而不是飞机控制空中交通管制。由于该问题涉及组织和社会分析，第五章中涵盖了有关控制结构和控制关系的详细讨论。

尽管初学者有时会无法区分，但通过经验的积累，这些内容会变得简单明了。有时，在控制结构中弄错了控制关系，如谁控制谁，也不会对分析结果造成较大影响。例如，假设控制行为 X 被错当成了反馈 X。因为是反馈，负责识别不安全控制行为的步骤将不会考虑丢失的或延迟的控制行为 X 是否会导致危险。然而，下一步检查潜在反馈危险时，会识别同样的情境，即考虑丢失的或延迟的反馈 X 如何会导致危险。

除控制结构图外是否需要存档其他文件？

存档任何其他有利于理解控制结构的文件是非常好的做法。为保证控制结构说明的完整性，对于与控制器有关的其他信息如基本描述、目的、特殊功能、控制器职责、过程模型等都应进行存档。尤其是在无法从简短的标注中获取重要方面信息时，与控制行为及反馈有关的说明信息也存在一定价值。对于受控过程也是如此。总的来说，任何重要的说明信息或假设都应与控制结构图一同进行明确存档。

在评审控制结构时应评审哪些内容?

以下建议有助于查找控制结构内的常见错误：

控制结构内常见错误的避免建议：

- 确保标注描述的是已发送的功能性信息而不是特定实际操作。
- 在了解信息类型的情况下，避免使用模糊且容易引发歧义的简单词语如“命令”、“反馈”进行标注。
- 查看是否每个受控物理过程都是由一个或多个控制器控制（不做常态化要求，但通常情况下能暴露出问题）。
- 审核职责（包括追溯过程），查看有无矛盾及差距。
- 检查用于满足职责的控制活动是否被纳入。
- 检查用于满足职责的反馈是否被纳入。（如在早期概念开发阶段进行应用，当反馈不知道时，该项为可选项；后续步骤可以识别丢失反馈）

4. 识别不安全控制行为

一旦控制结构建模完成，下一步（如*图 2.14* 所示）就是识别不安全控制行为。

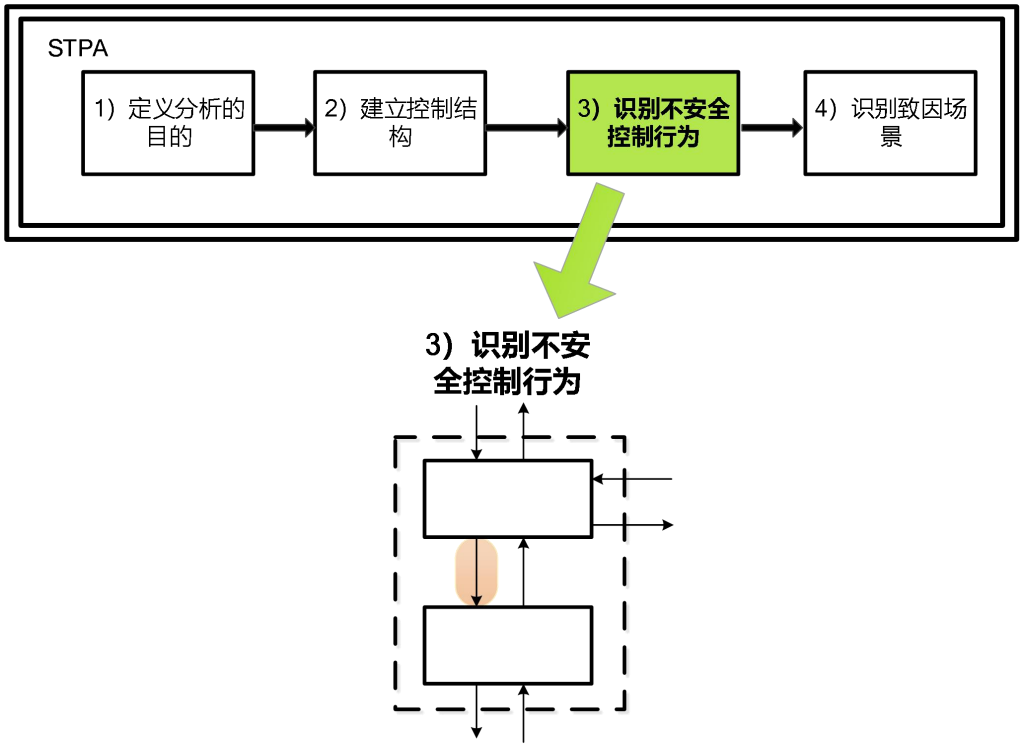


图 2.14：识别不安全控制行为

定义：不安全控制行为（UCA）指的是在特定情境及最坏环境下可能导致危险

的控制行为[9]。

下方*表 2.3* 给出了刹车系统控制单元（BSCU）控制器的不安全控制行为实例。更多刹车子系统不安全控制行为内容参见附录 C。

表 2.3：刹车系统控制单元（BSCU）不安全控制行为实例（部分内容）

控制行为	**由于“未提供”引起的危险**	**由于“提供”引起的危险**	**提供太早、太晚或顺序颠倒**	**停止太早、应用时间过长**
刹车	UCA-1：刹车系统控制单元（BSCU）Autobrake 在刹车系统控制单元（BSCU）启动状态下未提供刹车控制行为[H-4.1]	UCA-2：刹车系统控制单元（BSCU）Autobrake 在正常起飞时提供了刹车控制行为[H-4.3, H-4.6] UCA-5：刹车系统控制单元（BSCU）Autobrake 在着陆滑跑时所提供的刹车控制行为不充分[H-4.1] UCA-6：刹车系统控制单元（BSCU）Autobrake 在着陆滑跑时所提供的刹车控制行为引起偏航或不匀称[H-4.1, H-4.2]	UCA-3：刹车系统控制单元（BSCU）Autobrake 在着陆之后提供的刹车控制行为过晚（>x 秒）[H-4.1]	UCA-4：刹车系统控制单元（BSCU）Autobrake 在飞机着陆时停止提供刹车控制行为过早（在达到特定滑行速度前）[H-4.1]

出现以下四种情况，代表控制行为可能存在安全隐患（以上面栏目中的内容给出）

1. 未提供控制行为导致危险。
2. 提供控制行为后导致危险。
3. 提供可能安全的控制行为但提供节点过早、过晚或顺序错误
4. 控制行为持续太久或停止过早（仅针对持续性控制行为而非离散行为）。

[9] “不安全”一词指的是 STPA 中所识别出来的危险。如之前所言，危险可能包括与人身伤害或生命威胁（传统意义上的安全）相关的问题，但它的定义也可以比较广义，包括其他损失如任务失败、性能失效、环境损失等。

考虑 UCA-2：

UCA-2：刹车系统控制单元（BSCU）在正常起飞时提供了刹车控制行为 [H-4.3, H-4.5]

该 UCA 不安全是因为它可能导致 H-4.3：起飞速度达到 V1 后出现减速以及 H-4.5：起飞期间未充分加速。每个不安全控制行为（UCA）都能追溯到一个或多个危险（或子危险）上，因此在每个不安全控制行为结束的括号内归档其追溯是非常好的做法。

不安全控制行为应说明在何种情境下控制行为是不安全的。情境信息十分重要。假设你知道飞机上的刹车系统控制单元（BSCU）可以提供刹车命令。那么这条命令是不安全的吗？在不考虑情境的情况下是无法进行评估的。UCA-2 包含了“在正常起飞时”这个情境，因此正是这个情境使得控制行为处于不安全状态。

如果一个控制行为一直处于不安全状态，那么工程师就不会将其纳入系统设计。所以每个不安全都必须说明控制行为在什么情况下（哪种情境下）是不安全的。随后我们就可以将那些条件从系统设计中剔除或是找出缓解的方法。在不安全控制行为中可以提出任何相关情境，包括环境条件、受控过程状态、控制器状态、以往活动及参数（如正被编程的特定下降率）。在构建不安全控制行为时使用诸如“当...时”、或“在...过程中”等表达方式有利于情境的描述。

每个不安全控制行为都包含五个部分：

UCA-2：BSCUAutobrake 在正常起飞时 提供 刹车命令 [H-4.3]

<来源> <情境> <类型> <控制行为> <相关危险>

第一部分是可以提供控制行为的控制器。第二部分是上面讨论到的情境。第三部分是不安全控制行为的类型（提供、未提供、过早或过晚、停止太早或持续太久）。第四部分是控制行为或命令本身（来自控制结构），最后一部分关联所可能导致的危险（或子危险）。

不安全控制行为通常情况下按照上述顺序进行描述，但在某些情况下调换顺序可以显得更加清晰自然。顺序无关紧要。重要的是每个不安全控制行为都要包含这四个部分。

a) 不安全控制行为（UCAs）常见问题

不安全控制行为一定会导致危险吗？

不一定。在最佳情境下，UCA-2 可能会发生在起飞初始阶段，飞行员意识到 UCA-2 出现后，立即关闭刹车系统控制单元（BSCU），如此也就不会出现事故。而在最坏情境下，飞行员可能无法及时作出反应，无法有效关闭刹车系统控制单元（BSCU），而且出现了严重顺风，UCA-2 刚好在达到 V1 决策速度后发生，或者出现了其他因素导致飞机脱离跑道（H-4）。STPA 的目标不是争论最佳或最坏情况哪

个更可能出现或是对飞行员的能力和反应做出假设。在设计刹车系统控制单元（BSCU）时，我们仍然希望能够防止它在正常起飞时发送刹车命令，无论出现的是最佳情境还是最坏情境。该步骤的目标仅为识别应该避免的行为。STPA 是一个最坏情境分析方法而不是最佳情境、一般情境或是最可能发生情境分析方法。

在已经采取保护措施的情况下还可以识别不安全控制行为吗？

系统本身可能包括诸如保护特性、冗余及备用系统等保护性措施，特别用于防止不安全控制行为导致危险。例如，一些人可能认为 UCA-1：“刹车系统控制单元（BSCU）Autobrake 在刹车系统控制单元（BSCU）启动状态下未提供刹车控制行为”不会导致危险，因为设计中所包含的独立备用刹车系统能够有效绕过刹车系统控制单元（BSCU）并允许飞行员随时进行手动刹车。在最佳情境下，这些保护措施可以按部就班地正常运行，且其效果可以满足要求，那么就可以避免危险。但在最坏情境下，保护措施也可能无法正常运行或者其效果不充分，不能解决棘手的问题。与上述理由一样，STPA 是一个最坏情境分析方法，即使存在保护措施我们也不能忽略不安全控制行为。

另外一种理解方式是明确即使存在保护措施，我们也要预防不安全控制行为。例如，即使设计中可能包含备用刹车系统，我们也不会故意设计刹车系统控制单元（BSCU）来制造不安全控制行为。我们希望确保刹车系统控制单元（BSCU）Autobrake 能够提供合适的刹车控制行为（如防止 UCA-1 等），即使存在如备用刹车系统这样的保护措施。

实际上，理想状态下，STPA 的应用节点应在保护措施存在并纳入设计之前。STPA 将识别必须预防的不安全控制行为，随后这些不安全控制行为将用于生成功能性要求并进行设计决策以预防或缓解不安全控制行为。潜在不安全行为经过识别后，将创建具体设计特性，添加保护措施（如果设计尚未成形）或决定现有设计决策和保护措施的充分性（如果设计已经成形）。

不安全控制行为的最后两类都与时间有关，二者区别在哪？

不安全控制行为的第三类指的是控制行为的提供时间错误——即过早、过晚或顺序颠倒。而第四类则仅适用于具有一定持续时长的控制行为，即持续性或连续性控制行为。

首先，假设刹车系统控制单元（BSCU）Autobrake 所提供的刹车命令作为一个持续性控制行为已被执行。换句话说，在 Autobrake 开始提供刹车控制行为时，刹车就已经执行了，直到 Autobrake 停止提供刹车控制行为，刹车才停止。刹车控制行为的初始时间可能会导致危险：

UCA-3：刹车系统控制单元（BSCU）Autobrake 在着陆之后提供的刹车控

制行为过晚（>x 秒）[H-4.1]

在着陆期间刹车控制行为可以正确准时无延迟（在 x 秒内）地进行提供，但 Autobrake 随即停止提供控制行为，刹车立即停止。刹车停止时间过早，造成无效刹车。即使原来控制行为在适当情况下准时提供，该行为也可导致 H-4.1，如 UCA-4 所述：

UCA-4：刹车系统控制单元（BSCU）Autobrake 在飞机着陆时停止提供刹车控制行为过早（在达到特定滑行速度前）[H-4.1]

现在我们来考虑以下备用解决方案：刹车系统控制单元（BSCU）Autobrake 提供了两个独立的非连续控制行为以启动和停止刹车。因此独立的控制行为的持续时长没有意义，所以“停止过早/持续过久”并不适用于该情境。所以，我们考虑其中一个控制行为的提供节点可能过早或过晚：

UCA-3：刹车系统控制单元（BSCU）Autobrake 在着陆之后提供的开启刹车控制行为过晚（>x 秒）[H-4.1] UCA-4：刹车系统控制单元（BSCU）Autobrake 在飞机着陆时停止提供停止刹车控制行为过早（在达到特定滑行速度前）[H-4.1]

注意在对控制行为是持续的还是非持续的分析中我们处理过同样的问题。

我需要识别每个不安全控制行为的确切类型吗？

不需要。虽然所有四个不安全控制行为类型都需要考虑，但不是适用于任何情况。也可能需要识别属于同一种类别的多个不安全控制行为，如上面 UCA-2、UCA-5 和 UCA-6。总的来说，每个类型都可能包含 0 个、1 个、2 个或者更多不安全控制行为。

有多于四种类型的不安全控制吗？是否需要另做分类？

经验证，这四种类型较为全面——不存在其他类型的不安全控制行为。但存在子分类。例如，第二类不安全控制行为就包括三种表现方式：

2.已提供一项不安全控制行为

a 考虑控制行为在任何情况下都是不安全的情境

b 考虑控制行为在不充分或者过量情况下是不安全的情境

c 考虑控制行为的方向是不安全的情境

后两个情况仅针对包含一个或多个参数的控制行为。例如, 刹车系统控制单元（BSCU）刹车控制行为一般不是一个简单的非黑即白的开/关一样的控制行为。刹车系统控制单元（BSCU）通过命令应用一定程度的刹车来控制刹车。UCA-2 规定在正常起飞时提供刹车命令（无论刹车系统控制单元（BSCU）所要求的刹车程度如何）是不安全的。UCA-5 之所以存在，是因为即使刹车系统控制单元（BSCU）

可能可以在着陆滑跑时正确提供刹车命令，但如果刹车系统控制单元（BSCU）所提供的刹车不充分，也可能是不安全的。UCA-6 存在的原因是刹车命令可能会被不对称发送，从而导致飞机偏航。任何时候当控制行为可以规定一个或多个参数时，在给定情境下考虑参数可能如何不充分、过量或方向错误是非常重要的。

我刚刚识别出一项重要的不安全控制行为，但与任何系统级危险都无关。我该怎么办？

每个不安全控制行为都一定可以追溯到一个或多个系统级危险。如果你识别出的不安全控制行为与任何已识别危险均无关联，你可能遗漏了一项危险。识别由不安全控制行为导致的系统级状态或条件并考虑添加新危险或修改现有的危险列表，纳入新状态或新条件。STPA 是一个不断迭代的进程，无需严格按照线性方式执行——随着分析进行，更多信息的获取，可以对早期结果进行更新。

所描述的不安全控制行为的后果或结果在哪？

每个不安全控制行为结果应参考其可能导致的危险或子危险，这些危险捕捉了不安全控制行为的系统级影响及后果。对于很多系统来说，不安全控制行为和危险之间的联系显而易见。例如，在着陆滑跑期间刹车系统控制单元（BSCU）提供的不充分刹车明显可以联系到 H4.1：着陆时不充分减速。

然而，在一些情况下不安全控制行为和危险之间的联系可能比较复杂或不直观。那么归档不安全控制行为后的特殊原因和它们与危险的关系，尤其是在极为复杂的应用下二者关系不那么明朗时是非常好的做法。在一些情况下可通过向不安全控制行为添加描述来归档原因，但如果原因较为复杂，这种方法实施起来比较困难。一般的解决方法就是根据需要归档每个不安全控制行为的描述评价。不安全控制行为评价可以描述其工作原理、如何导致危险、所基于的假设、影响与结果或其他用于了解不安全控制行为与其如何导致危险的信息。

注意不要将不安全控制行为情境与其后果、结果或其他原因混淆。常见错误之一就是忽略不安全控制行为情境而纳入其结果。例如：

> *正确的不安全控制行为：刹车系统控制单元（BSCU）Autobrake 在正常起飞时提供刹车命令[H-4.3]*
>
> *错误的不安全控制行为：刹车系统控制单元（BSCU）Autobrake 在发生碰撞时提供刹车命令*

如果不安全控制行为情境（当前状态或条件）未经识别，下一步即生成要求及识别情境将很难甚至无法进行。每个不安全控制行为都必须包含不安全控制行为情境。为了更加明确，也可能包含其结果。

我应该在不安全控制行为情境中说明过程模型漏洞吗？

注意不要将不安全控制行为与其原因混淆。不安全控制行为的内容应说明导致控制行为不安全的实际状况或条件，而不是特定控制器过程模型或想法（这样描述有可能是对的，也有可能是错的）。STPA 的下一步将识别不安全控制行为的原因，如过程模型漏洞及其他因素。例如：

正确的不安全控制行为：刹车系统控制单元（BSCU）Autobrake 在正常起飞时提供刹车命令

错误的不安全控制行为：刹车系统控制单元（BSCU）Autobrake 在错误认为飞机处于着陆状态时提供刹车命令

b) 识别人为性质的不安全控制行为

在识别人为控制的不安全控制行为时也可采用同样的方法。例如,考虑机组控制行为关闭刹车系统控制单元（BSCU）。*表 2.4* 显示的是该命令所对应的机组不安全控制行为的一些实例。更多与轮刹车相关的机组不安全控制行为实例参见附录 C。

表 2.3：机组不安全控制行为实例（部分实例）

控制行为	由于“未提供”引起的危险	由于“提供”引起的危险	提供太早、太晚或顺序颠倒	停止太早、应用时间过长
关闭刹车系统控制单元（BSCU）	UCA-1：当刹车系统行为异常时，机组未能提供刹车系统控制单元（BSCU）关闭命令 [H-4.1, H-4.4, H-7]	UCA-2：当需要防打滑功能且刹车系统功能正常运行时，机组提供了刹车系统控制单元（BSCU）关闭命令 [H-4.1, H-7]	机组在需要自动刹车或防打滑功能时，在其完成之前过早地关闭了刹车系统控制单元（BSCU）[H4.1, H-7]	N/A

最后一栏为 N/A（即不适用）是因为在该情况下关闭控制行为不存在持续时间。机组仅仅是提供开启和关闭的命令来控制刹车系统控制单元（BSCU）的状态。关于刹车系统控制单元（BSCU）维持电源关闭时间过长的问题将在考虑未提供上电命令或提供上电过晚时进行捕获。注意，人类行为与系统的其他类型组件一样，都是系统的一部分，可直接整合到整个分析中。

c) 避免常见错误的一些建议：

以下为识别不安全控制行为时避免常见错误的建议：

识别不安全控制行为时常见错误的提示：

- 确保每个不安全控制行为都对使控制安全活动处于不安全状态的环境进行说明。
- 确保不安全控制行为情境说明导致控制行为不安全的实际状况或条件，而不是有关实际状态的潜在想法。
- 确保不安全控制行为情境定义清晰。
- 确保涵盖不安全控制行为情境而不会被后续影响或后果取代。
- 确保追溯过程妥善存档以将每个不安全控制行为与一个或多个危险联系起来。
- 重新评审每个被假定为不适用的控制作用类型，并验证其不适用性。
- 对于带有参数的任何持续性控制行为，确保考虑到参数的过量、不足、方向错误等情况。
- 确保不安全控制行为背后的任何假设或特殊原因妥善存档。

d) 识别控制器约束

定义：控制器约束规定了用于预防不安全控制行为所需满足的控制器行为

一旦不安全控制行为被识别出来，可以转化为每个控制器的行为约束。例如，在分析刹车系统控制单元（BSCU）控制行为时，我们认为在正常起飞时刹车系统控制单元(BSCU)提供刹车控制行为可能导致危险。因此，刹车系统控制单元(BSCU)在该情境下务必不可提供刹车控制行为。总的来说，每个不安全控制行为都可以经过转化来定义每个控制器的约束。*表 2.5* 给出了从*表 2.4* 中的不安全控制行为导出的刹车系统控制单元（BSCU）行为约束。

表 2.5：机组不安全控制行为实例（非完整版）

不安全控制行为	控制器限度
UCA-1：刹车系统控制单元（BSCU）Autobrake 在刹车系统控制单元（BSCU）启动状态下且飞机着陆滑跑时未提供刹车控制行为[H-4.1]	C-1:刹车系统控制单元（BSCU）Autobrake 在刹车系统控制单元（BSCU）启动状态下且飞机着陆滑跑时必须提供刹车控制行为[UCA-1]
UCA-2：刹车系统控制单元（BSCU）在正常起飞时提供刹车控制行为[H-4.3, H-4.5]	C-2:刹车系统控制单元（BSCU）在正常起飞时不得刹车控制行为[UCA-2]
UCA-3：刹车系统控制单元（BSCU）Autobrake 在着陆之后提供的刹车控制行为过晚（>x 秒）[H-4.1]	C-3:刹车系统控制单元（BSCU）Autobrake 在着陆之后的 x 秒内必须提供刹车控制行为[UCA-3]
UCA-4：刹车系统控制单元（BSCU）Autobrake 在飞机着陆滑跑时停止提供刹车控制行为过早（在达到特定滑行速度前）[H-4.1]	C-4:刹车系统控制单元（BSCU）Autobrake 在飞机着陆滑跑时且在达到特定滑行速度前不得停止提供刹车控制行为[UCA-4]
UCA-5：刹车系统控制单元（BSCU）Autobrake 在着陆滑跑时所提供的刹车控制行为不充分[H-4.1]	C-5:刹车系统控制单元（BSCU）Autobrake 在着陆滑跑时不得提供少于 x 等级的刹车[UCA-5]
UCA-6：刹车系统控制单元（BSCU）Autobrake 在着陆滑跑时所提供的刹车控制行为引起偏航或不匀称[H-4.1, H-4.2]	C-6:刹车系统控制单元（BSCU）Autobrake 在着陆滑跑时不得提供引起偏航或不匀称的刹车[UCA-6]

e) 控制行为分析的输入与输出

图 2.15 总结了控制行为分析的输入与输出。

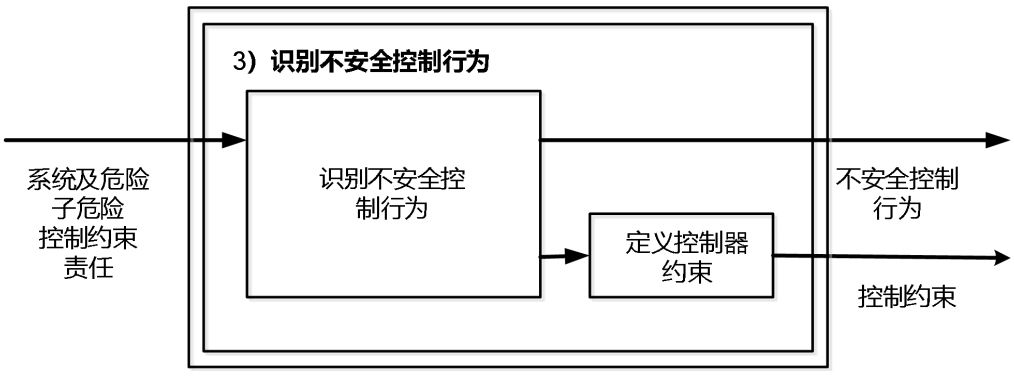


图 2.15：控制行为分析概览

5. 识别致因场景

一旦不安全控制行为识别完成，下一步（如*图 2.16* 所示）就是识别致因场景。

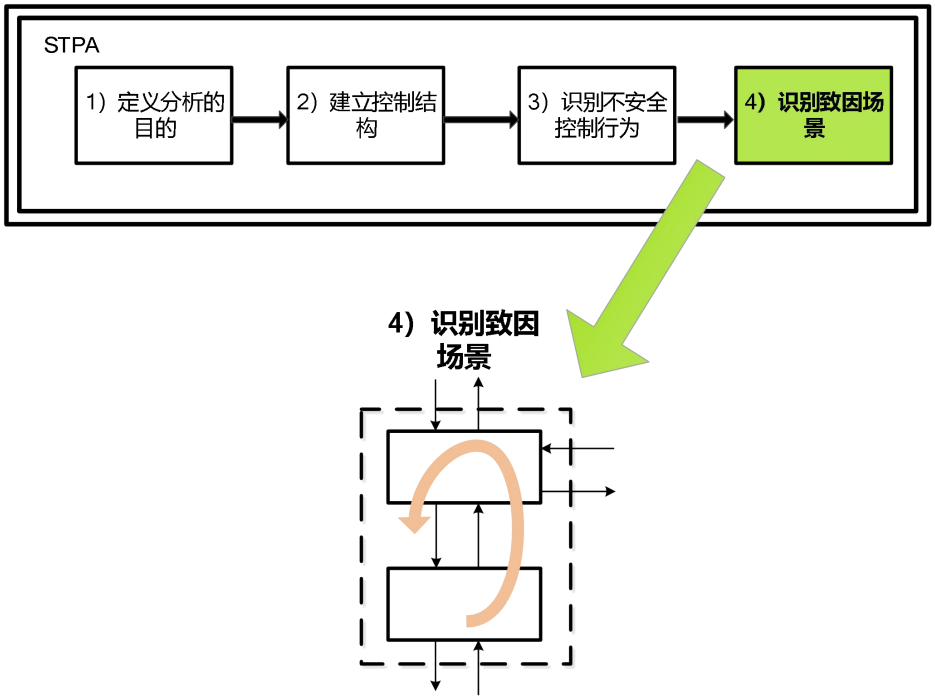


图 2.16：识别致因场景

定义：致因场景描述的是可能导致不安全控制行为以及危险的诱发因素。

如*图 2.17* 所示，必须考虑两类致因场景：

a) 为何会出现不安全控制行为？

b) 控制行为为何会出现执行不利、未执行的情况而导致危险？

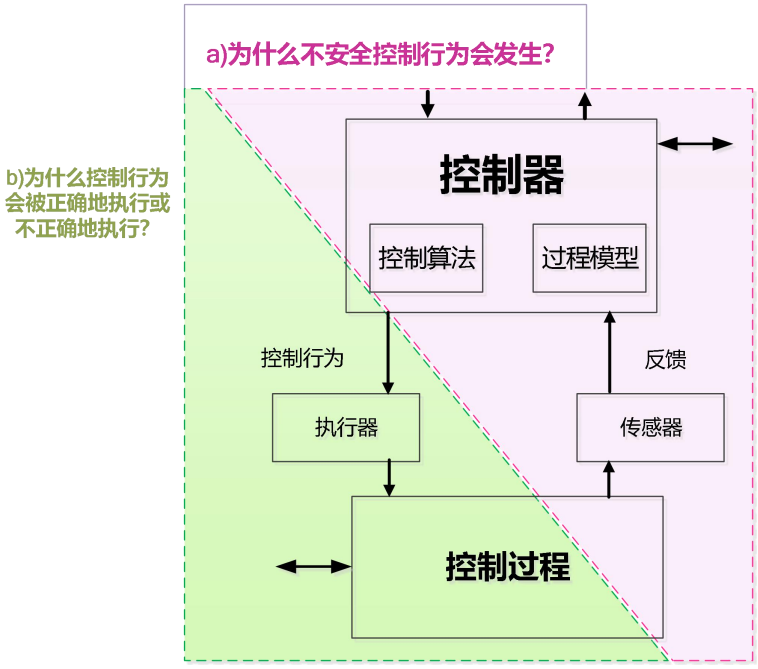


图 2.17：必须考虑的两种情境

请注意*图 2.17* 中包含传感器和执行器。到了分析中的这一步，我们已经考虑到控制行为与反馈存在的可能，但还没有验证如何测量或检测反馈（如，利用传感器）或是如何执行控制行为（如，利用执行器）。由于情境可以识别不安全控制及反馈的具体原因，这就有利于完善控制结构以加入传感器和执行器。

a) 识别导致不安全控制行为的致因场景

此类致因场景可通过以不安全控制行为作为起始点，反向解释是什么导致控制器提供（或不提供）控制行为进行创建。总的来说，可导致不安全控制行为的致因场景包括：

- 与控制器相关的故障（仅针对实际控制器）
  - 控制器自身物理性故障
  - 电源故障
  - 等
- 不充分的控制算法
  - 特定控制算法运行漏洞
  - 特定控制算法存在漏洞
  - 特定控制算法由于变更或退化逐渐体现出不充分性。
- 不安全的控制输入
  - 从其他控制器接收到的不安全控制行为（在考虑来自于其他控制器的不安全控制行为时已得到解决）
- 不充分过程模型
  - 控制器接收错误的反馈/信息
  - 控制器接收正确的反馈/信息但未做正确处理或未处理
  - 控制器未收到所需的反馈/信息（延迟收到或从未收到）
  - 不存在必要的控制器反馈/信息

为创建涉及不安全控制行为的致因场景，我们必须考虑*图 2.18* 中显示的因素，从导致 UCA 的控制器行为开始分析。

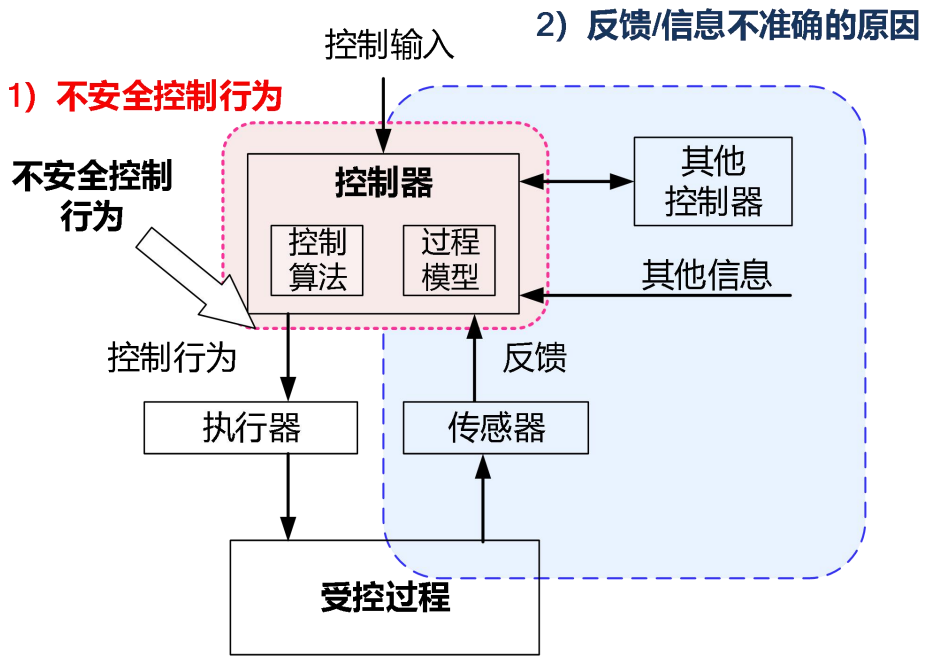


图 2.18：不安全控制行为可由（1）不安全控制器行为以及（2）不充分反馈及其他输入导致

1）不安全控制器行为

控制器可能提供（或不提供）不安全控制行为的原因包括以下四点：

- 与控制器相关的故障（仅针对实际控制器）
- 不适当的控制算法
- 不安全的控制输入（来自其他控制器）
- 不适当过程模型

对于实际控制器来说，UCA 可由与控制器有关的故障导致。例如，由于刹车系统控制单元（BSCU）电源故障等因素导致控制器故障，从而刹车系统控制单元（BSCU）可能无法提供刹车命令。为识别这些致因场景，需从 UCA 开始，随后确定控制器、识别控制器相关的且能够解释不安全控制行为的物理故障。例如：

UCA-1：刹车系统控制单元（BSCU）Autobrake 在刹车系统控制单元（BSCU）启动状态下未提供刹车控制行为[H-4.1]

UCA-1 致因场景 1：刹车系统控制单元（BSCU）Autobrake 在刹车系统控制单元（BSCU）启动状态下且飞机着陆滑跑时出现故障，导致无法提供刹车控制行为[UCA-1]。因此，飞机着陆时可能出现不充分减速。

不适当的控制算法也可能导致 UCA。控制算法可规定如何根据控制器的过程模型、以往控制输入与输出以及其他因素来选择控制行为。对于人员控制器，控制算法有时候也被称为决策，可由多种因素如培训、程序以及以往经验决定。

为了识别此类场景，需从不安全控制行为开始，随后识别控制算法如何能够导致不安全控制行为。例如：

刹车系统控制单元（BSCU）Autobrake 实例：

UCA-3：刹车系统控制单元（BSCU）Autobrake 在着陆之后提供的刹车控制行为过晚（>x 秒）[H-4.1]

UCA-3 致因场景 1：飞机着陆，但刹车系统控制单元（BSCU）内的进程延迟导致了刹车控制行为提供时间过晚[UCA-3]。因此，飞机着陆时可能出现不充分减速。

人类机组实例：

机组-UCA-1：当刹车系统行为异常时，机组未能提供刹车系统控制单元（BSCU）关闭命令 [H4.1, H-4.4]

机组-UCA-1 致因场景 1：刹车系统行为异常且机组收到刹车系统控制单元（BSCU）错误指示。机组没有关闭刹车系统控制单元(BSCU)[Crew-UCA-1] 因为操作程序没有规定机组在收到刹车系统控制单元（BSCU）错误指示时必须关闭 BSCU。

总的来说，控制算法漏洞主要来源于以下方面：

- 特定控制算法运行漏洞
- 特定控制算法存在漏洞
- 特定控制算法由于变更或退化逐渐体现出不适用性。

产生这些漏洞背后的具体原因取决于所研究的应用对象。

当控制算法假设以往控制行为都已被妥善执行时，就会出现常见的控制算法漏洞。而在没有反馈能够显示控制行为是否成功的情况下，该漏洞的相关度极高。例如，BSCU 可能无法提供刹车控制行为，由于刹车系统控制单元（BSCU）Autobrake 错误地假设以往刹车控制行为已经生效，飞机已经在刹车。

为了添加安全相关致因场景，这里需要另外考虑一种可能性：识别控制算法是否有漏洞以及控制算法缺陷是如何通过对立方引入。

来自于其他控制器的不安全控制输入也可能导致不安全控制行为。这在识别其他控制器的不安全控制行为时即可发现。

最后一点，不完整的过程模型也会导致不安全控制行为。如上所释，过程模型代表着控制器的内部想法，可被控制算法用来决定控制行为。当控制器过程模型不符合实际情况时就会出现过程模型漏洞。过程模型漏洞可由以下原因导致：

- 控制器接收错误的反馈/信息
- 控制器接收正确的反馈/信息但未做正确处理或未处理
- 控制器未收到所需的反馈/信息（延迟收到或从未收到）
- 不存在必要的控制器反馈/信息

此类问题的表现形式多样，根据应用的不同，有所区别。控制器可能会接收到错误的反馈/信息，包括无法解决或无法正确解决的矛盾信息和矛盾。控制器可能会

接收到正确的反馈/信息但忽略处理，原因可能是控制器出现故障、被关闭、忙于其他任务、丢失带有必要更新所需条件的过程模型或是其他原因。如果过程模型未正确更新、更新了错的过程模型、反馈/信息代表了其他项目或出现其他错误控制器可能出现处理问题。

如果控制器从未接收到反馈，那么在需要时可能也无法接收反馈/信息——特别是如果控制器假设了一个默认值代替反馈——或是反馈延迟，包括所接收到的信息出现错乱。最后，必要的反馈/信息可能不存在于控制结构或设计中，将导致过程模型不充分。

为识别此类场景，需从 UCA 开始，随后识别控制器过程模型如何能够导致 UCA。要求考虑有关当前状态或模式、以往状态、能力、动态行为[10]、以往行为或活动、未来状态或行为（预测）的观念、有关当前受控过程、其他受控过程、系统中其他控制器（特别是协调所需的想法）、执行器、传感器或系统或环境的其他相关方面的想法。

一旦可能导致不安全控制行为的相关过程模型被识别出来，接下来就要确定过程模型会在已接收的反馈或其他信息（或缺乏此类信息）的影响下以何种形式出现。

例如：

UCA-2：刹车系统控制单元（BSCU）Autobrake 在刹车系统控制单元（BSCU）启动状态下未提供刹车控制行为[H-4.1]

可能导致不安全控制行为的控制器过程模型（想法）：控制器认为飞机已经停稳

控制器接收正确的反馈但未做正确处理：在防打滑功能运行期间，轮速信号在某一刻可能归零，导致过程模型漏洞

UCA-2 致因场景 1：刹车系统控制单元（BSCU）已经启动且飞机开始着陆滑跑。刹车系统控制单元（BSCU）未能提供刹车控制行为[UCA-2]因为刹车系统控制单元（BSCU）错误地以为飞机已经停稳。如果已接收的反馈在着陆滑跑期间的某一刻显示速度为 0，该过程模型漏洞将会出现。虽然飞机并没有停下，但已接收的反馈在防打滑功能运行期间的某一刻显示速度为 0。

可能导致不安全控制行为的控制器过程模型（想法）：飞机处于飞行中，控制器在需要时未接收到信息：未接收到着陆指示

UCA-2 致因场景 2：刹车系统控制单元（BSCU）已经启动且飞机开始着陆滑跑。刹车系统控制单元（BSCU）未能提供刹车控制行为[UCA-2]因为刹

[10] 在这里可以捕捉控制理论中的调试相关问题。过程模型可能包括控制器关于一个受控过程的动态特性的想法，当想法出现错误时，可能会出现调试问题。例如，PID 控制器中的比例项、积分项、导数项代表着控制器关于受控过程的动态特性的想法。如果这些想法出现错误，就会出现稳定性或其他问题，进而控制算法会生成不安全控制行为。

车系统控制单元（BSCU）错误地以为飞机正在空中还没有着陆。如果在着陆时未接收到着陆指示，该过程模型漏洞将会出现。<为什么呢？该致因场景有待完善。>

任何涉及不适当的反馈/信息的致因场景必须经过完善以解释其不适当的原因。在不了解不适当的反馈和信息原因的情况下，无法进行预防。

2）不适当的反馈和信息原因

当一个致因场景能够识别导致不安全控制行为的反馈或信息（或缺乏此类信息）时，我们需要查验反馈/信息的来源以找出导致这些问题的罪魁祸首。反馈来自于受控过程（一般通过传感器发送），其他信息可能来源于其他程序、控制器或系统或环境中的其他源头。

总的来说，与不适当的反馈和信息相关的致因场景可能包括以下方面：

- 未接收到反馈或信息
  - 控制器未接收到通过传感器发送的反馈/信息
  - 传感器未发送反馈/信息但反馈/信息已被传感器接收或应用
  - 反馈/信息未被接收或应用于传感器
  - 控制结构中不存在反馈/信息或不存在传感器
- 接收到不适当的反馈
  - 传感器合理地响应但控制器接收到的是不适当的反馈/信息
  - 传感器对已被接收或应用于传感器的反馈/信息进行了适当的响应
  - 传感器无法提供必要反馈/信息或不具备提供必要反馈/信息的功能

涉及反馈/信息的致因场景已被发送但未被接收或未被适当接收可能是由于传输错误、通信丢失、通信延迟（包括反馈/信息已被发送但接收错乱）以及其他问题所导致。不适当的传感器响应或无响应的致因场景可能是由于传感器失效、电源供应故障、传感器运行或测量失准、传感器错误或失误、响应延迟、配置错误、老化退化、传感器环境出现意外情况或其他问题导致。另外，传感器可能由于设计错误、规格漏洞、对受控过程的错误假设、测量条件错误、报告正确但误导性信息（如当机轮被锁但飞机还在移动时报告 0 轮速）或其他问题导致无法提供必要反馈。

我们可以通过在实际系统状态下（参考不安全控制行为致因场景）确定反馈/信息可能被接收的原因来完善这些致因场景。

例如，让我们来完善上面的 UCA-2 致因场景 2：

<u>不安全控制行为致因场景下的真实状态：</u> 飞机正在着陆滑跑（参见上面致因场景 2）

<u>接收到的信息：</u> 着陆时未接收到着陆指示（参见上面致因场景 2）

<u>在实际状态下此类情况如何发生：</u> 报告机轮速不足、报告轮载重不足、轮速或轮载重指示延迟等。

UCA-2 致因场景 2：刹车系统控制单元（BSCU）已经启动且飞机开始着陆滑跑。刹车系统控制单元（BSCU）未能提供刹车控制行为[UCA-2]因为刹车系统控制单元（BSCU）错误地以为飞机正在空中还没有着陆。如果在着陆时未接收到着陆指示，该过程模型漏洞将会出现。

如出现以下任何一种情况，可能无法接收着陆指示：

- 由于跑道湿滑，机轮像水上飞机一样滑行（轮速不足）
- 由于采用过滤技术，轮速反馈延迟
- 由于侧风着陆，出现矛盾的空中/地面指示
- 轮速传感器失效
- 空中/地面开关失效
- 等

因此，飞机着陆时可能出现不适当的减速。

为了添加安全相关致因场景，这里仅需要另外考虑一种可能性：识别特定反馈与其他信息怎样能够被对立方所影响。具体来说：特定反馈与其他信息是如何加入、顶替、篡改、截获或泄露给对立方的？例如，如果添加安全因素，以下原因可加入上述致因场景 2：

- 对立方顶替反馈，指示轮速不足
- 由于对立方执行 DoS 攻击，轮速反馈延迟
- 正确的轮速反馈被对立方截获并屏蔽
- 对立方关闭了轮速传感器的电源

b) 识别控制行为执行不当或未被执行的致因场景

不安全控制行为可能导致危险，但如果控制行为执行不当或未被执行，没有不安全控制行为的情况下也可能导致危险。为了生成此类致因场景，我们必须考虑会影响控制路径和受控过程的因素，如*图 2.19* 所示。

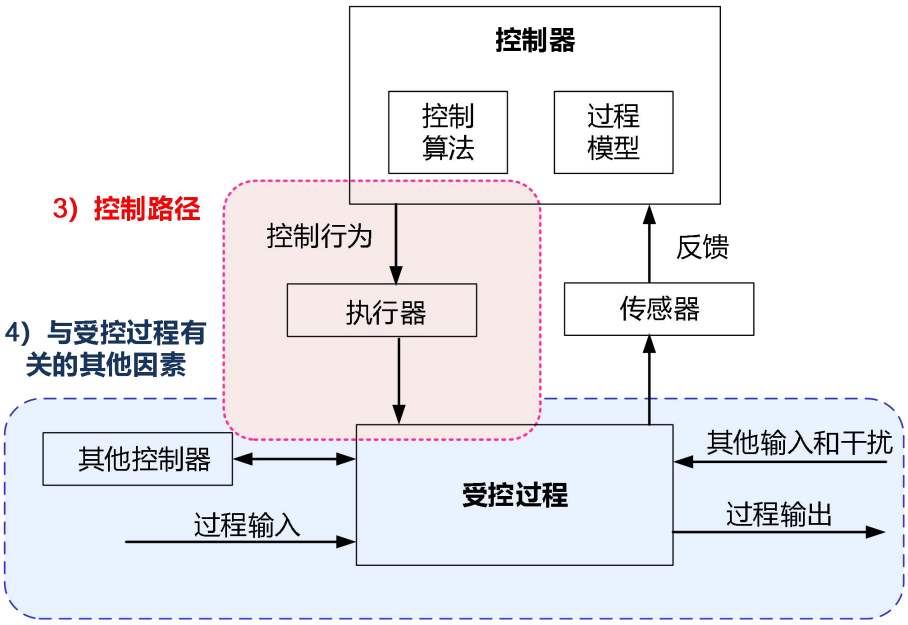


图 2.19：一般控制回路阐释 1）控制路径及 2）其他可以影响受控过程的因素

1）涉及控制路径的致因场景

控制路径负责将控制行为转移到受控过程。控制路径可能由一个简单的执行器组成，可能涉及一系列执行器，或可通过带有开关、路由器、卫星或其他设备的复杂网络转化控制行为。无论如何操作，我们必须识别路径中的问题如何能够导致控制行为执行不当或未被执行。

总的来说，涉及控制路径的致因场景包括：

- 控制行为未执行
  - 控制行为已通过控制器发送但未被执行器接收
  - 控制行为已被执行器接收但执行器未响应
  - 执行器做出响应但控制行为未被应用到受控过程或未被受控过程接收
- 控制行为执行不当
  - 控制行为已通过控制器发送但执行器接收不当
  - 控制行为已被执行器正常接收但执行器响应不适当
  - 执行器做出适当的响应但控制行为未被应用到受控过程或未被受控过程正确接收
  - 控制器未发送控制行为但执行器或其他元件仍做出了响应

已被发送的控制行为，没有被正确接收或未接收，可能由于通信延迟（包括控制行为已被发送但接收顺序错乱）、传输错误、通信丢失以及其他问题导致。

与执行器响应错误或无响应，可能是由于执行器失效、电源供应故障、执行器运行失准、执行器错误或失误、响应延迟、所接受到的其他命令（包括可能与其他

控制器矛盾的命令）、执行器错误的优先级方案、配置错误、老化退化、执行器环境出现意外情况或其他问题导致。

为了生成此类致因场景，需要从一个控制行为开始，识别执行不当或未执行对于你的应用来说有什么后果，确定控制路径如何导致该行为。

例如：

控制行为：刹车系统控制单元（BSCU）发送刹车命令

未执行：刹车未应用

执行不当：刹车应用不充分

致因场景 1：刹车系统控制单元（BSCU）在着陆时发送刹车命令但由于执行器故障刹车未应用。因此，飞机着陆时可能出现不充分减速[H4.1]

致因场景 2：刹车系统控制单元（BSCU）在着陆时发送刹车命令但由于执行器响应较慢导致刹车不充分。因此，飞机着陆时可能出现不充分减速[H-4.1]

致因场景 3：刹车系统控制单元（BSCU）在着陆时发送刹车命令但由于线路错误未被执行器接收。因此，飞机着陆时可能出现不充分减速[H-4.1]

同时也要考虑控制行为可能未被发送但执行器或其他元件仍做出响应的原因。这些情节与所提供控制行为的不安全控制行为相似。

例如：

控制行为：刹车系统控制单元（BSCU）未发送刹车命令

执行不当：在正常起飞时应用刹车（类似 UCA-2）

致因场景 4：刹车系统控制单元（BSCU）未发送刹车命令但由于液压阀门故障导致刹车应用。因此，飞机起飞时可能出现加速不充分[H-4.6]

为了添加安全相关致因场景，这里需要另外考虑一种可能性：识别特定控制行为怎样能够被对立方所影响。具体来说：特定控制行为是如何加入、顶替、篡改、截获或泄露给对立方的？例如：

致因场景 5：刹车系统控制单元（BSCU）发送了刹车命令但由于对立方否认了服务攻击，屏蔽了刹车命令导致刹车未应用。因此，飞机着陆时可能出现不充分减速[H-4.1]

2）受控过程相关致因场景

即使控制行为被转移或应用于受控过程，控制行为也可能失效或被其他控制器重写。

总的来说，与受控过程相关的致因场景可能包括：

- 控制行为未执行
    - 控制行为被应用或被受控过程接收但受控过程未响应
- 控制互动执行不当

- 控制行为被应用或被受控过程接收但受控过程响应不当
- 控制行为未被应用或被受控过程接收但受控过程仍做出了响应

此类情况可能是由缺失或不充分程序输入（如不充分液压等）、外部或环境干扰、组件故障、响应被程序延迟、程序中的错误或失误、从其他控制器接受到的可能矛盾的命令、以往接受到的或应用于受控过程的控制行为、程序中错误的优先方案、配置错误、老化退化、程序环境中出现意外或无法应对的情况或其他问题导致。

为了生成此类致因场景，需选择一个控制行为并确定何种因素能够影响受控过程使得控制行为失效。

例如：

控制行为：刹车系统控制单元（BSCU）发送刹车命令

致因场景 6：刹车系统控制单元（BSCU）发送了刹车命令但由于刹车系统此前被命令进入备用刹车模式（绕过刹车系统控制单元（BSCU））导致刹车未应用。因此，飞机着陆时可能出现不充分减速[H-4.1]

致因场景 7：刹车系统控制单元（BSCU）发送了刹车命令但由于不充分的液压（液压泵故障、液体泄漏等）导致刹车未应用。因此，飞机着陆时可能出现不充分减速[H-4.1]

致因场景 8：刹车系统控制单元（BSCU）发送了刹车命令但由于跑道湿滑导致飞机无法减速。（水上飞机机轮情况）。因此，飞机着陆时可能出现不充分减速[H-4.1]

为了添加安全相关致因场景，这里需要另外考虑同一种可能性：识别对立方如何与受控过程交互以导致同样的问题。例如：

致因场景 9：刹车系统控制单元（BSCU）发送了刹车命令但由于对立方加入了一个命令，促使刹车系统进入备用刹车模式导致刹车未应用。因此，飞机着陆时可能出现不充分减速[H-4.1]

c) 避免常见错误的一些建议：

最常见的错误就是识别单独的原因而不是从广义致因场景考虑。例如，你可能会倾向于列出一个因素清单，如“轮速传感器故障”、“轮速反馈延迟”、“断电”等等。但在不考虑整体致因场景的情况下，列举单独因素，这样做可能会忽略这些因素之间的作用关系，可能会遗漏那些间接导致不安全控制行为甚至危险的重要的且不明显的因素，还可能忽略考虑这些因素的组合如何导致危险。只考虑单独因素实质上就下降到了故障模式与理象分析（FMEA）的层面，即只考虑单独组建的故障。

d) 鉴定致因场景的输入与输出

图 2.20 总结了致因场景识别步骤的输入与输出。

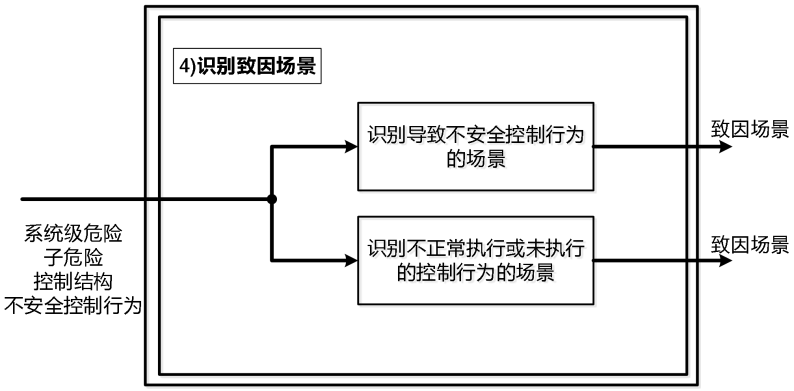


图 2.20：致因场景识别概览

6. STPA 输出与追溯过程

图 2.21 给出了多个 STPA 输出间所维系的追溯过程。

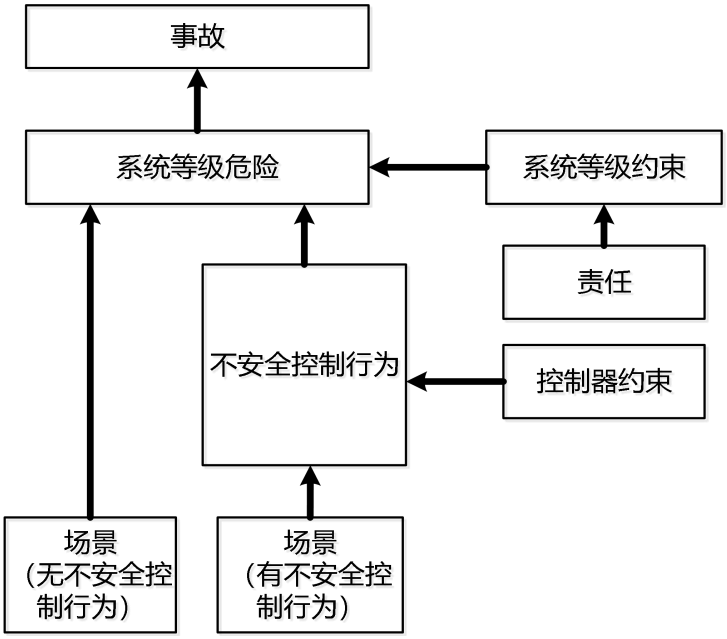


图 2.21：STPA 输出间的追溯过程

控制结构与每个 STPA 输出都紧密相连，因此为了简单明了，*图 2.21* 显示了它们之间的复杂关系。这些 STPA 输出用处多样，包括：

- 导出系统架构
- 生成可执行要求
- 识别设计建议
- 识别所需的缓和与保护措施
- 定义测试案例并生成测试计划

- 导出新的设计决策（如果 STPA 用于发展阶段）
- 评估现有设计决策并识别差距与所需变更（如果 STPA 用于设计完成后期）
- 生成领先风险指标
- 设计更加有效的安全管理系统
- 等

现在你应该对 STPA 的执行有了基本的了解。我们建议在你自己的工程问题上试着应用该分析方法。附录中所提供的其他实例可能会有所帮助。

7. 总结与展望

本章描述了 STPA 中使用的基本方法。下一章将阐释如何将 STPA 应用于系统工程的多个阶段与活动。第四章提供了在工作场地安全中使用 STPA 的额外信息与实例。到目前为止，所述实例均涉及技术系统但 STPA 也可用于任何社会技术系统。第五章将展示如何在组织分析中运用 STPA 以及如何将 STPA 应用于除安全外的突发的、整个系统的性能。第六章将介绍如何运用 STPA 识别风险领先指标。第七章则将描述如何运用 STAMP 和 STPA 来设计更加有效的安全管理系统。最后，第八章将说明目前我们所了解的有关于如何将 STPA 集成到大型组织或项目的方法。

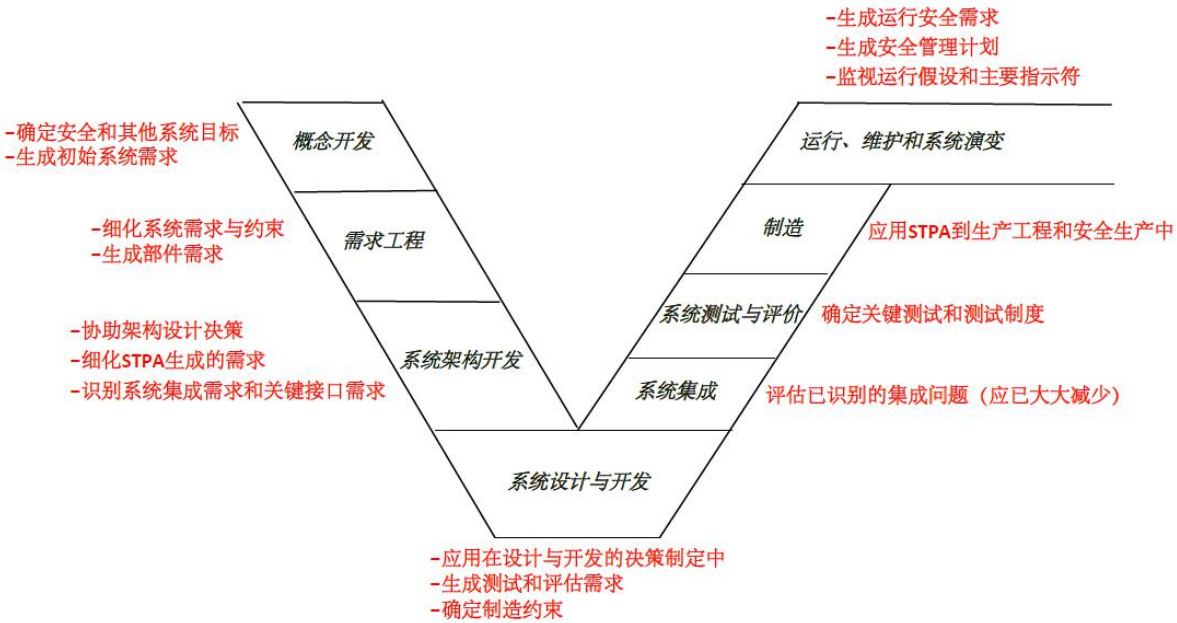
第 3 章：将 STPA 集成到系统工程过程中

南希·莱文森

通常，系统安全在某种程度上与系统工程过程隔绝或分离。最常见的结果是，安全被视为事后保障活动。由于无法保障系统安全，但又必须在系统中设计安全，因此当安全相关的设计缺陷无法修复时，往往很晚才发现。这时，工作的重点变成了试图论证已识别的缺陷无需修复。当这些论证无法得到支撑时，处理安全缺陷的努力往往会转移到制定昂贵却又不太有效的解决方案上，例如冗余或期望系统操作员通过不理想的程序解决方案来检测并修复问题。

本章描述了如何将系统安全分析紧密地集成到整个系统工程过程中（有关系统工程的基本介绍，请参阅附录 F）。这一行为将使安全工程的成本显著降低，有效性大大提高，并且有望减少损失。同时还可以减少返工，从而减少成本和工作计划。

下图展示了简化版的标准系统工程 V 模型（或带折线的瀑布模型）。为了使图表整洁，省略了反馈环。此图用于说明如何将 STPA 集成到标准系统工程过程中。STPA 的潜在作用以红色显示。如果您使用 V 模型的变体或不同的开发模型，大多数其他模型都不难从标准 V 模型转换过来，除了那些省略了早期过程（V 字左边部分上面的两个活动）的模型之外。在这种情况下，您不应该构建安全关键系统。



从最早的概念开发阶段开始，STPA 可应用于整个标准系统工程过程中。本质上，STPA 可以在概念开发阶段早期生成高层安全需求，在系统需求开发阶段对其进行细化。系统需求和约束可以帮助系统架构的设计和更详细的系统设计与开发。STPA 的结果过程继而与设计和开发密切相关，因为这些分析可用于制定决策。STPA 通过保障与制造继续发挥作用，并为运行期间的使用提供重要信息。

由于 STPA 是在系统模型（随着设计决策的制定而细化）的基础上工作的，因此 STPA 适合基于模型的工程过程，尽管这一模型与当前为基于模型的系统工程通常所提出的架构模型不同。在整个开发过程中，STPA 促进了可追溯性，因此在改变决策和设计时，对重做先前分析的需求可降至最低。

最后，正如本手册中多次提到的那样，除安全性之外，STPA 还可应用于系统工程和产品生命周期的任何系统特性。

整体过程

STAMP 和 STPA 可用于系统工程的所有活动。本章其余部分介绍了这些分析及其结果在以下活动中的使用：

1. 开发和运行期间应处理损失的定义
2. 识别系统设计的外部约束（包括市场和监管需求）
3. 系统级危险的识别及其对系统行为的相关需求和约束
4. 系统控制结构的建模
5. 危险与约束的细化以及对系统部件的功能分配
6. 协助架构，设计与实施决策的制定
7. 系统集成帮助
8. 系统测试需求的生成
9. 制造控制（生产工程、安全生产）
10. 运行安全需求的生成（包括不断增长风险的领先指标）与安全管理计划
11. 包括监测领先指标在内的运行安全管理

1. 关于应考虑损失的决策

虽然早期的概念开发可能会与 V 模型的特定变体有所不同，但这一阶段通常包括利益相关者和用户分析（需求分析）、客户需求生成、监管需求评审、可行性研究、概念和权衡空间探索以及建立项目演变及最终设计的评估标准等活动。在这一步的最后，可能会创建一个运行概念。

绝大多数情况下，系统工程的这一早期阶段没有得到应有的重视和努力，开发几乎立即进行系统架构规范和高层设计。然而，概念开发不合适可能导致系统无法使用、系统只能部分满足利益相关者的需求，或系统难以保障、维护和运行。虽然可以在开发后期进行更改，以弥补早期概念开发阶段的遗漏，但随着开发的进行，这些后期的更改成本越来越高，破坏性也越来越大。

在早期概念探索中，特别是安全性和保障问题往往未得到充分考虑。图 3.1 展示了处理这些涌现系统特性的典型方法（Young，2017）。工作重点往往是在运行中出现损失后才对其做出响应。此外，工作中心可能是在设计和系统工程主体完成之后添加“改进措施”（例如，保护系统或入侵检测）。在开发过程的后期进行这样的更改不仅成本更高，而且修复效果通常比从一开始就在系统中构建安全性和保障要差得多。

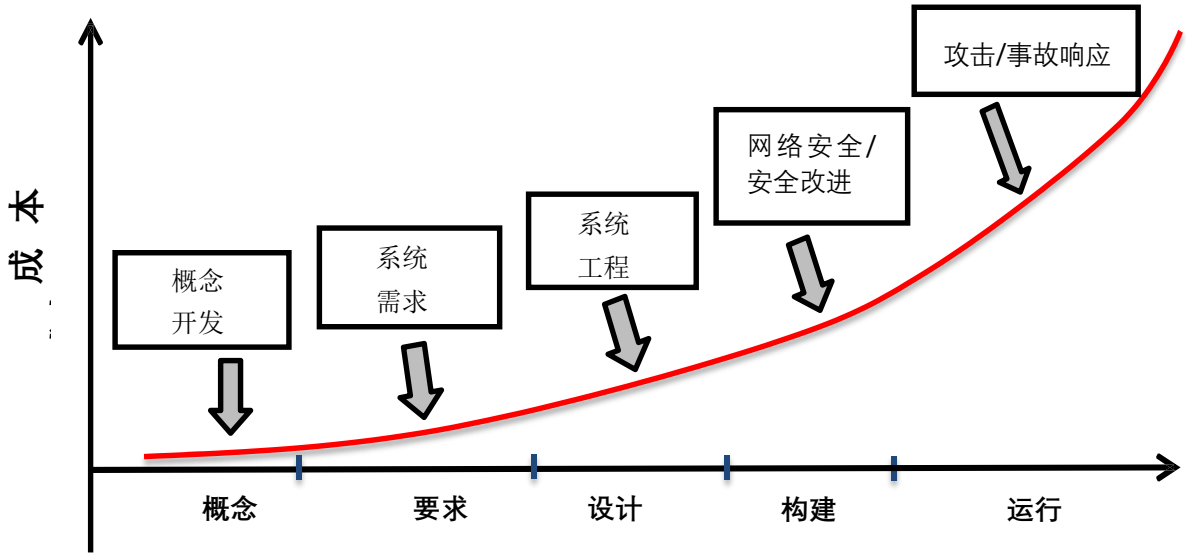


图 3.1：在开发初始阶段就需要在系统中构建安全性和保障[11]

相反地，在概念开发阶段就应当开始考虑安全性和保障，并将结果用于生成整个系统及其部件的安全性与保障需求。Frola 和 Miller（1984）[12]声称 70-90%的与安全性相关的设计决策是在概念开发阶段做出的，之后再进行更改也许是不可行的，也许是成本高昂的。对于保障亦是如此。

在许多系统工程过程中，可能在这一阶段考虑安全性，但通常是以“预先危险性分析”的形式进行的。该分析会提供一个危险评估，以确定在开发和运行中，对于已识别的危险应投

[11] William Young，保障复杂运行不受网络干扰的系统理论安全分析法，麻省理工学院博士论文，2017 年。

[12] F. Ronald Frola 和 C.O. Miller，飞机采购中的系统安全，技术报告，物流管理机构，华盛顿特区，1984 年一月。

入多少精力，以及一些关于如何消除或控制这些危险的初步建议。

表 2.1. 典型的 PHA 格式[13]

项　目： 工程师：				日期： 页码：		
项目	危险情况	起因	影响	RAC	评估	建议
指定编号	列出情况的性质	描述导致所述情况存在的原因	如果允许不进行纠正，危险情况的影响将是什么	指定危险等级	发生的概率与可能性： -可能性 -暴露率 -严重性	消除或控制危险的建议措施

[Vincoli, 2005]

虽然表中大部分信息在早期概念开发阶段都可用，但是在详细设计完成之前，危险情况的概率或可能性无法预知；而对于软件密集型系统，甚至在完成之后依然不可知。当系统与过去的系统存在显著差异时，历史信息不再有用。我们发现，使用 STPA 时，在系统开发过程的早期，那些具有完全合理的致因场景的实际项目中的特定危害通常被错误地排除为“临界”或极不可能。心理学家的研究表明，人类非常不擅长估计不寻常事件的概率或可能性[14]。

最大的问题是，这类 PHA 没有为系统提供识别功能安全性和保障需求的必要信息，而这是早期概念分析的主要目标。理论上可用于这一目标的传统危险分析技术需要相当详细的设计，因此这些技术在开发过程后期才适用，而此时为设计工作提供功能安全性需求已为时过晚。

航空界常用的另一种方法是生成概率需求，以满足设计系统的要求【SAE ARP 4761】。示例包括“每次飞行中，着陆或中止起飞过程中的全部机轮刹车损失应小于 5E-7”以及“每次飞行中，起飞过程中一个机轮未锁定的情况下未检测到的意外机轮刹车应小于 5E-9”【SAE ARP 4761】。虽然在刹车系统几乎完全由故障概率已知的硬件部件组成的时代，这类要求可能是合理的，但是，由于软件几乎所有系统中的广泛应用，无法确保这些需求在当今正在设计的软件密集型系统中得到满足

STAMP 和 STPA 能够生成概念开发阶段所需的信息——确定利益相关者和用户需求（需求分析），生成客户目标和需求，评审监管要求，评估可行性，探索系统概念和概念性权衡空间，并建立面向项目演变和最终设计的评估标准。许多失败或有问题的系统工程工作中涉及到的主要问题源于对利益相关者需求和适当的系统目标的理解不足。

[13] Jeffrey Vincoli，系统安全基本指南，约翰威立出版公司，2005 年。

[14] 例如，参见 Tversky 和 Kahneman 关于启发式认知偏差的研究。

2. 识别系统设计的外部约束（包括市场和监管要求）

社会系统安全控制结构可以用来理解系统的外部（环境）社会约束。图 3.2 展示了一个社会技术控制结构的示例，该示例类似于许多管制行业的常见结构。控制结构适用于开发（左侧）和运行（右侧）。每一个行业都可能有自己的社会技术控制结构，而且可能因国家而异。这里只展示了一个国家结构的例子，但在某些行业（如航空业和核工业）中，存在叠加的国际控制要素。

一旦对此结构进行建模，对着手的每个开发项目可能都会有一些更改，除了可能存在一些差别。该结构可用于识别和归档正在开发的系统的监管要求和其他外部要求。

3. 应考虑的损失的确定

在概念开发的早期阶段，利益相关者需要指定不可接受的损失。虽然许多危险分析和系统安全技术只涉及人身伤亡和财产损失事件，但是 STPA 可以处理任何损失，包括环境污染、任务损失、金钱损失，甚至是公司声誉损害。换句话说，损失代表了利益相关者想要避免的任何涌现系统特性。正规来讲，事故（或军方使用的术语——意外事故）是指使利益相关者认定为损失的任何意外或计划外的事件或情况。本手册中使用的飞机示例从以下事故（损失）开始：

A1. 乘客或飞机区域内人员的死亡或重伤

A2. 飞机或飞机外部物体的不可接受的损坏

须由利益相关者来定义什么被视为“不可接受”。如果认为有必要努力避免额外损失，则可对其进行详述指定，例如：

A3: 延误运行（航班延误）带来的财产损失

A4: 因飞机或航空公司声誉受损而减少的销售额。

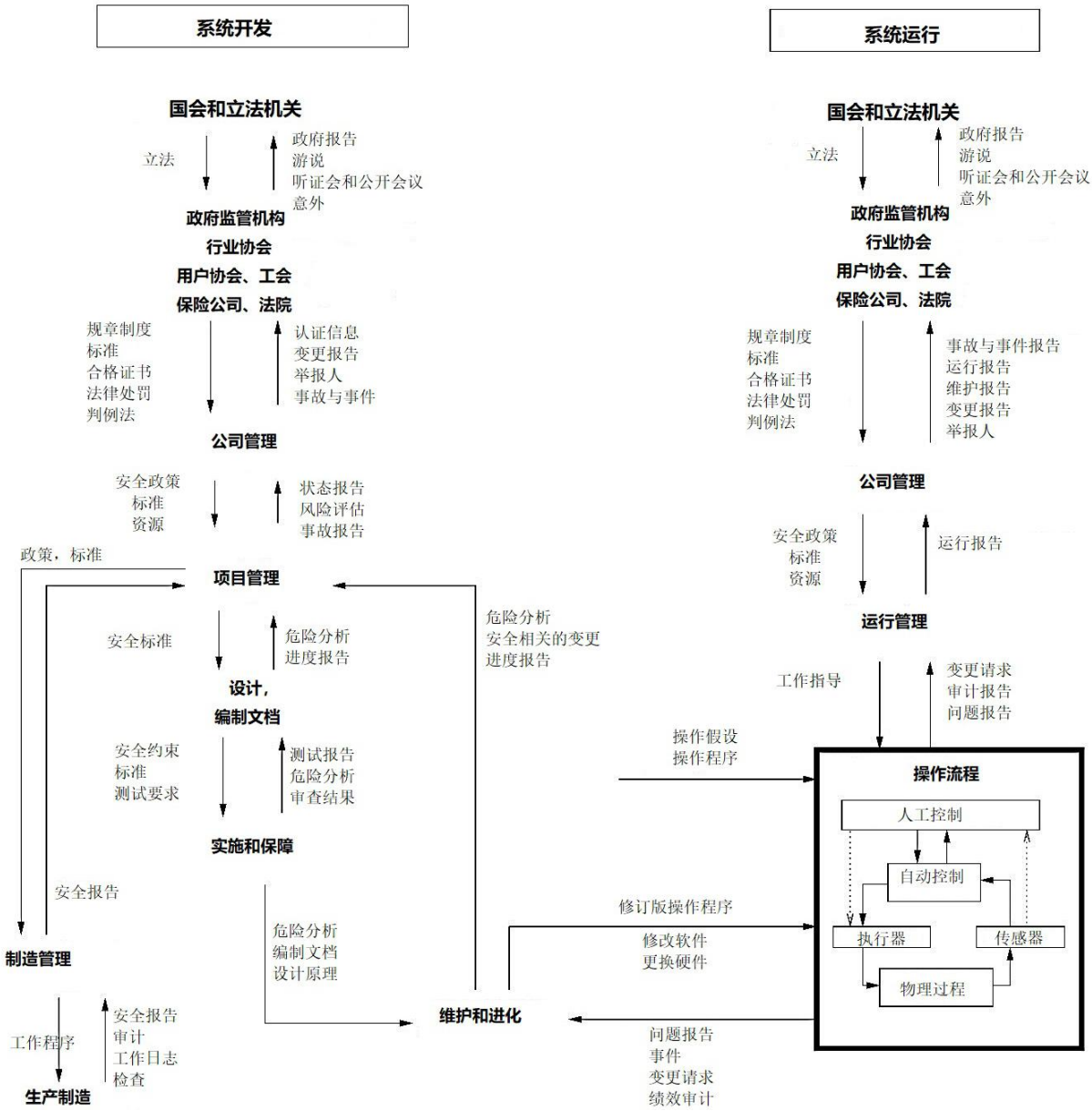


图 3.2：安全控制结构示例

4. 系统级危险的确定及其对系统行为的相关约束

一旦利益相关者确定了不可接受的损失，工程师就可以定义与这些损失相关的危险。最终目标是在系统设计期间消除、控制（若不可能消除的话）或减少已确定的危险。

第一个问题是如何定义危险。尽管“危险”这一术语有时用于指系统边界以外的事物，例如航空中的恶劣天气，但 STPA 中的“危险”仅限于系统设计者控制范围内的系统状态。例如，“危险”不是指恶劣天气，而是指飞机所受到的恶劣天气的负面影响。约束或控制可能包括远离恶劣天气或把飞机设计为可以承受恶劣天气的影响。由于系统工程与系统的设计有关，因此它只能够影响系统本身的行为，而不能改变系统边界以外的事物（即在系统操作环境中）。

有时，危险被定义为可能导致事故的条件或事故的先决条件。但这个定义几乎涵盖一切事物，如果系统要运行的话，其中大部分都不能被消除或控制。例如，一架正在着陆的飞机可能会在之后引起事故，但我们不能建立一个飞机永不着陆的有效系统。另一个例子是，危险不是着陆时的减速（这是安全操作所必需的行为），而是着陆后无法减速。

因此，出于实际原因，危害被定义为系统设计者不希望发生的事故的前兆状态，因此应该在设计过程中消除或控制。正式定义在前一章已出。

显然，为了在系统设计中消除危险，需要在系统开发过程的初始阶段对危险进行识别确定。指定的危险用于为包括软件和操作人员在内的各种系统部件创建行为（功能性，而非概率性）安全需求。

对于上述的事故 A1 和 A2，危险包括以下几点：

H1：推力不足，无法维持受控飞行[A1, A2]

H2：机身完整性损失[A1, A2]

H3：违反飞机与固定或移动物体之间的最小间隔[A1, A2]

H4：地面上的飞机离移动或静止的物体太近，或无意中离开滑行道[A1, A2]

H5：飞机不能按计划起飞[A3, A4]

H6：等等…

这些危险会导致高层的系统约束[15]，例如：

SC1：必须有足够的推力来维持受控飞行[H1]

SC2：在最坏的情况下，也必须保持机身完整性[H2]

SC3：必须满足飞机与固定或移动物体之间的最小间隔标准[H3]

SC4：地面上的飞机必须始终与移动或静止物体保持安全距离，并保持在如滑

[15] 由于通常情况下不允许负需求，且在不丢失重要信息的情况下，有时无法将安全约束改为正面陈述，因此这里要区分要求/需求和约束。约束使用“必须”或“禁止”，而非“应当”。另一个好处是对达成目标的约束和需求（系统目标或任务）。当需要做出权衡决策时，差异化是有用的。

行道等的安全区域内。

SC5：飞机必须能在预定起飞时间的 TBD（待定）分钟内起飞[H6]

SC6：等等…

5. 构建功能系统控制结构

虽然第一个开发的控制结构模型可以包括系统运行的环境（在系统边界之外），但本步骤中的系统控制结构提供了关于系统设计空间内所含内容的更多详细信息。模型的范围，即要考虑的系统边界的定义，将取决于系统工程过程的目标。随着概念设计过程的进行，提供越来越多的细节，多个不同的模型可能都会有用。在上一章介绍了构建控制结构模型的详细内容。在本章节中会有一些重复内容，以便可以独立阅读并理解每个章节。

从图 3.2 中的控制结构来看，要设计的系统可能是图片右下角标注“操作流程”的方框中的功能系统控制结构。通常首先在这个系统控制结构中执行 STPA，以确认高层的系统需求和约束。在概念设计过程中，将细化并改进控制结构，并使用 STPA 来细化相关的系统需求和约束。

例如，图 3.3 展示了一个十分高层的飞机功能控制模型，而该模型只有三个部件：飞行员、自动控制系统（可能由多台计算机组成，但该决策将留给之后的设计阶段）以及物理飞机部件。对于飞机等复杂系统，抽象级别可用于放大当前正在考虑的控制结构的各个部分。这种自上而下的改进也有助于理解飞机的整体运行和识别部件之间的相互作用。

请注意，这个模型中没有任何设计决策，即该系统将是一架飞机，且该飞机将包括任何飞机的一般功能和设计。因此，它适用于早期概念开发阶段。对于系统工程这一阶段的某个特定应用来说，如果该模型的任意部分过于详细，则可以删除掉。请注意，飞机可能是飞行员操作（机上或地面）或自动操作（无人驾驶飞机），且飞机甚至可以是固定翼、直升机或 VTOL 飞机（垂直起降）。在概念开发的这一阶段，无需加以区分。这种通用性允许将决策推迟至有更多的信息可用之时，并允许重复使用模型和分析。如今正在设计的飞机系统可能包括几种或所有这些类型的驾驶配置。

如图 3.3 的控制结构所示，飞行员的作用是管理飞机的自动操作，并根据飞机的设计，直接或间接地控制飞机的起飞、飞行、降落以及地面操纵。飞行员和自动控制器拥有一个他们所控制的系统或子系统的模型。自动化装置控制着飞机，因此它必须拥有当前飞机状态的模型。飞行员也需要一个飞机状态模型，但除此之外，他们还需要一个自动化装置状态的模型和一个飞机正在运行的机场或空域的模型。许多飞行员错误可以追溯到他们对自动化装置工作原理的理解错误，或他们对自动

化装置当前状态的理解错误。

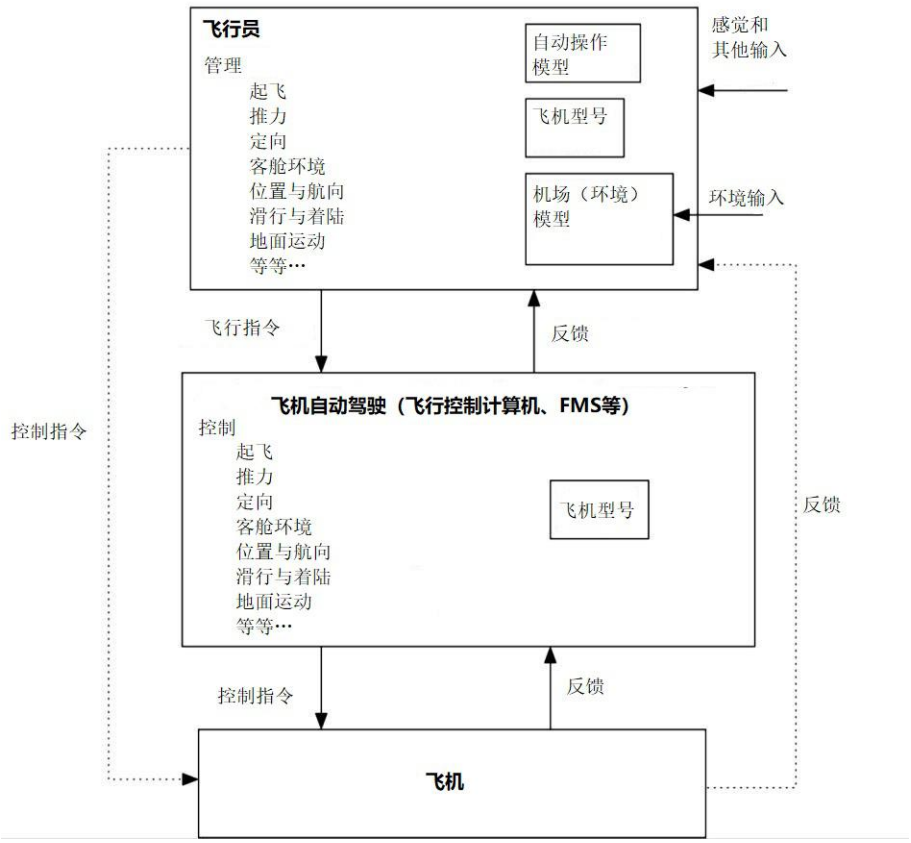


图 3.3：飞机层面的高级控制结构

飞行员向自动化装置提供飞行指令，并接收有关自动化装置和飞机状态的反馈。在一些设计中，飞行员可以直接控制飞机硬件（即不通过自动化装置），并接收直接反馈。虚线表示这种直接的控制和反馈。随着设计的改进和更详细的设计决策的制定，这些虚线链接可能会被删除或用特定的内容将其实例化。飞行员通过多种感觉通道，总是对环境和飞机的状态（除非飞行员在地面上，如 UAS（无人机系统））有一些直接的感觉反馈。同时，飞行员对自动化装置和正在进行自动化控制的物理部件状态的直接感觉反馈通常比较有限；他们可获得的信息将仅限于工程师认为对他们来说重要的那些项目。STPA 帮助确定这些关键信息。现如今，有些先进的飞行控制系统中包括一些根本不向飞行员提供反馈的状态和输入，其原因是工程师认为飞行员不需要反馈，或由于其他原因忽略了反馈。

STPA 使用这种控制结构来完善从系统危险中开发出来的非常笼统的系统级安全需求和约束（如上所示）。这些新的完善的安全需求将详细说明高层控制结构中三个部件的职责、权限和责任，以及每个部件的过程模型的内容和反馈要求，以保持飞机的安全运行。例如，一个需求可能是飞行员命令了合适的推力，以确保飞机在每个运行阶段的安全操纵[SC1]，而另一个需求是飞行员应提供安全指令，以控制

地面运动[SC4]。请注意，系统级安全需求现在开始分配给安全控制结构中的各个部件。

6. 危险与安全约束的细化以及对系统部件的分配

随着概念设计过程的继续，控制结构和需求将被不断细化。该过程源于不安全控制行为的识别及其与当前设计细节水平相适应的致因场景。结果将是一组可被细化并用于开发过程其余部分的与安全相关的需求。

系统和安全需求应先于用于满足这些需求的系统架构的创建。当然，对任何系统来说，安全都不是唯一目标，在架构的开发中需要考虑其他需求。但是，根据我们的经验，架构通常是在确定安全（通常是系统）需求之前开发的。这种颠倒的顺序使得架构不是最优的可能性增加，有时架构甚至不适合系统目标。

架构开发将包括将系统需求分配给系统部件。对于 STPA，该过程将包括生成在系统级产生的不安全控制行为的致因场景。

减速是细化过程的一个示例。这里的相关高层系统危险是

H4：地面上的飞机离移动或静止的物体太近，或无意中离开滑行道[A1, A2]

与 H4 相关的特定事故发生在飞机近地或在地面运行的时候，可能涉及飞机离开跑道或者飞机撞击跑道上或跑道附近的物体。此类事故可能包括以一定的速度撞击障碍物、其他飞机或其他位于跑道末端或跑道以外的物体，结果会造成不可接受的损害或人身伤亡。

H4 可细化为下列减速相关的危险：

H4-1：着陆、中止起飞或滑行时，飞机减速不足

H4-2：起飞时在 V1[16]点后减速

H4-3：停放时，飞机发生运动

H4-4：无意的飞机方向控制（差动刹车）

H4-5：飞机在安全区域外机动（滑行道、跑道、登机口、停机坪等）

H4-6：起落架收起时主起落架机轮未停止转动（或在收起后继续转动）

与这些危险相关的高层系统安全约束是对危险的简单重述，如同设计约束。

SC1：在着陆、中止起飞或滑行时，前向运动必须在刹车指令的 TBD 秒内减速（H4-1）。

SC2：在没有飞行员直接行动的情况下，飞机不得在 V1 后减速（H4-2）。

SC3：飞机停放时不得发生非命令移动（H4-3）。

SC4：差动刹车意外的飞机方向控制或失去控制（H4-4）

[16] V1 点是指在起飞顺序中到达这一点后，停止起飞比继续起飞更加危险。在一些非常罕见的情况下，飞行员在 V1 点后减速比继续起飞更安全。在这个示例分析中，我们忽略了这些情况（以及其他一些情况），以使结果可以符合这几页的内容。

SC5：飞机不得无意中驶出安全区域（滑行道、跑道、登机口和停机坪等）（H4-5）

SC6：起落架收起时，主起落架必须停止转动（H4-6）

请注意，所有这些都是非常高级别的，即他们将飞机作为整体来考虑。如前一章所述，它们将通过迭代过程被细化为一组更详细的功能安全需求，这些需求与包括机组人员、软件和组件交互在内的特定系统部件相关。之后生成系统安全需求，以预防 STPA 确认的致因场景的出现。该过程一直持续到在设计中充分分析和处理危险之后。重要的一点是，首先考虑整体系统行为，以便在流程开始时识别出部件之间潜在的不安全交互，而无需在之后考虑所有交互，并确定它们是否有害。

图 3.4 放大了发生减速行为的地面控制功能的控制结构模型。有三个基本的受控物理部件：反推器、扰流板和机轮刹车系统。同样，通过包含更大的功能控制结构而不是简单的其中一个，STPA 可以考虑与被分析的危险相关的制动部件之间的所有交互（预期和非预期）。显然，这里表明了在这一刹车系统中的一些设计决策，即决意包括反推器、扰流板和机轮刹车。然而，这一级别的设计细节仍然非常普遍，且属于高级别，可以满足大部分飞机。

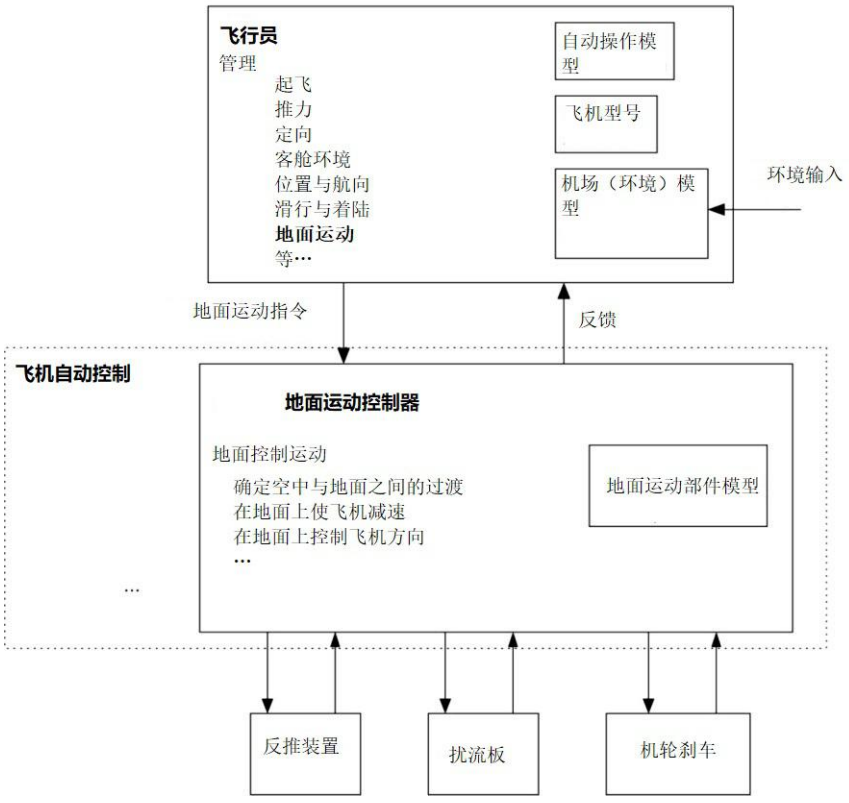


图 3.4：减速控制结构

与开发过程这一阶段可由 STPA 生成的制动系统的部件间交互作用相关的安全约束示例包括：

SC-BS-1：当高于 TBD 速度，机轮刹车手动或自动开启时，扰流板必须展开。

SC-BS-2：起落架收回时，必须启动机轮刹车。

SC-BS-3：地面扰流板的启动必须激活自动刹车系统。

SC-BS-4：在施加向前推力时，自动刹车系统不得启动机轮刹车。

SC-BS-5：在施加向前推力时，自动扰流板系统必须收回扰流板。

请注意，由于 STPA 包括操作人员、维护人员甚至管理人员，因此 STPA 提供了一种结构化方法来确认受系统设计影响的可能导致危险的人为差错，例如模式混淆和情景意识丧失。这些人为误差可能是由于自动操作的实际状态和操作员的心智模型之间的一致性丢失（缺乏同步）而引起的。

图 3.5 通过重点介绍机轮刹车系统的功能性结构，从而进一步放大了减速控制结构。同样，尽管图 3.5 中只考虑了制动系统的一个部件，但对于物理设计和实施没有做任何假设。STPA 在开始时没有针对潜在问题的具体设计解决方案。相反，它从所需的基本功能行为开始，并确定该行为在哪种条件下可能具有危险性。设计人员可以稍后决定特定的设计解决方案，例如冗余或重新设计功能，以满足通过该分析得出的安全需求。

图 3.5 确实包括一个重要的设计决策，以便将现代飞机常见的自动刹车功能包含在内。飞行员可以在着陆前预设一个自动刹车，以确保持续的刹车动作。在阵风或结冰的情况下，自动刹车可以减少飞行员的工作量，并大大减少刹车磨损。如果设置了自动刹车，则自动刹车将在指定时间接合机轮刹车。

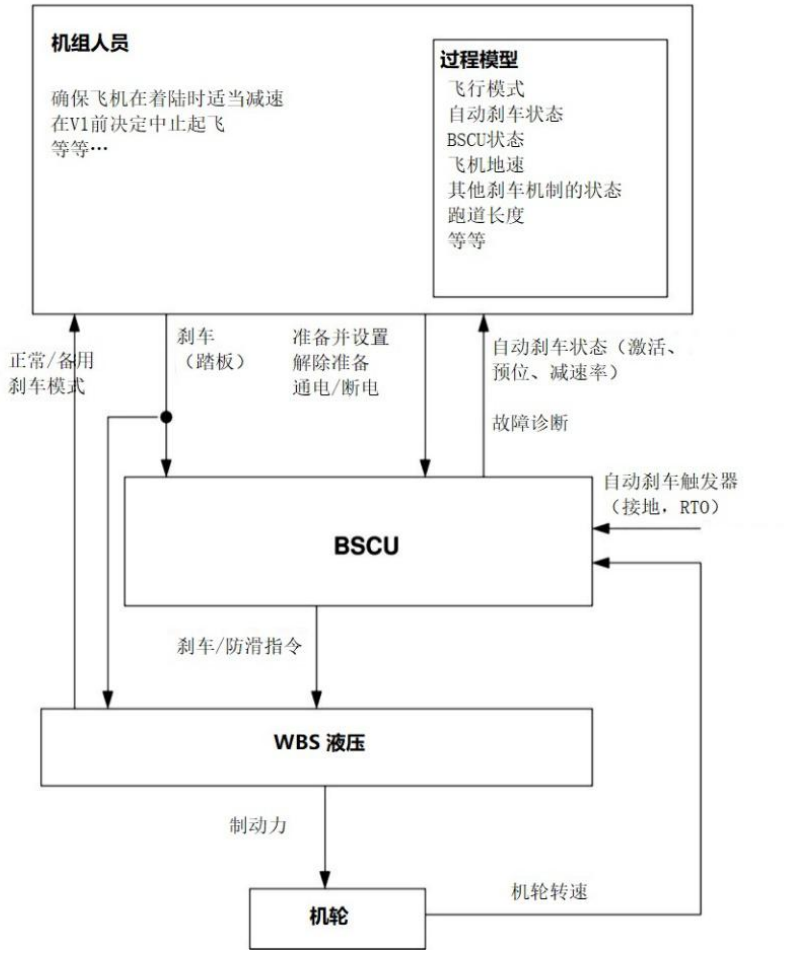


图 3.5：机轮刹车系统的控制结构

表 3.1 展示了 STPA 为图 2.6 中的飞行机组（FC）和刹车系统控制单元（BSCU）生成的一些安全约束。这些约束是根据本手册之前章节中所述的不安全控制措施背景表生成的。表 3.1 中包含了理论基础和可追溯性信息，以便在之后变更设计决策时，无需再介绍不安全特性。

表 3.1：STPA 生成的系统级安全约束示例

不安全的控制措施	描述	理论基础
FC-R1	着陆前，机组不得提供手动刹车[CREW.1c1]	可能导致机轮抱死、失控或轮胎爆裂
FC-R2	在达到安全滑行速度之前，机组不得停止手动刹车超过 TBD 秒[CREW.1d1]	可能导致超速或冲出跑道
FC-R3	自动刹车过程中，机组不得关闭 BSCU 电源[CREW.4b1]	会解除自动刹车
BSCU-R1	在 RTO 期间，必须始终提供刹车命令[BSCU.1a1]	可能导致无法在可用跑道长度内停止

可以使用系统需求和开发过程中此阶段的这些安全约束来定义高级别系统架构。STPA 生成的违反这些高级安全约束的致因场景将提供更详细的设计约束和其他信息，以帮助详细的架构和系统设计过程。

在做出设计决策时，STPA 分析会不断重复和细化，以帮助设计师做出权衡和有效的设计决策。对于刹车这一示例，图 3.6 展示了一个更详细的，可以体现出一些设计决策（包括液压控制器使用的阀门）的控制结构。涉及与液压控制器相关的额外阀门和指令的设计决策，至少部分是解决已确定的危险场景的结果。

在更详细的设计层上继续进行 STPA 分析的这一行为，在控制改进设计中包括的三个独立阀门时，涉及到不安全控制措施的识别以及与刹车系统控制单元液压控制器相关的安全约束（表 3.2）：

表 3.2：STPA 生成的关于机轮刹车系统控制结构的系统级安全约束示例

不安全的控制措施	**描述**	**理论基础**
HC-R1	当出现需要备用刹车的故障时，液压控制器不得打开绿色液压切断阀[HC.1b1]	正常刹车和备用刹车都会被禁用。
HC-R2	在发生打滑时，液压控制器必须触发防滑阀[HC.2a1]	需要防滑能力，以避免打滑，并在潮湿或结冰条件下实现完全停止。
HC-R3	当没有收到制动指令时，液压控制器不得提供打开绿色仪表阀的位置指令。[HC.3b1]	机组人员将不能意识到正在使用未指令的刹车。
HC-R4	在低速滑行时，液压控制器防滑装置不得释放刹车。	防滑系统通过感应机轮转速过低或机轮转速快速变化来感应打滑。这可能会导致低速滑行时刹车的意外

再次，可追溯性是通过不安全的控制行为和场景来保持的，而这些功能设计约束就是从中衍生出来的。那些控制行为和场景反过来又可以追溯到更高级别的设计和分析。其结果是一个复杂的树形结构，这个结构描绘出了贯穿整个设计过程的 STPA 分析，它将有助于进行更改或了解某些约束的起源。

此外，可以生成与不安全控制行为相关的更详细的致因场景，以及关于如何消除或缓解新场景的更多设计决策。可以通过分析备选设计来了解如何权衡备选设计方案的信息。该过程在整个系统开发过程中持续进行。

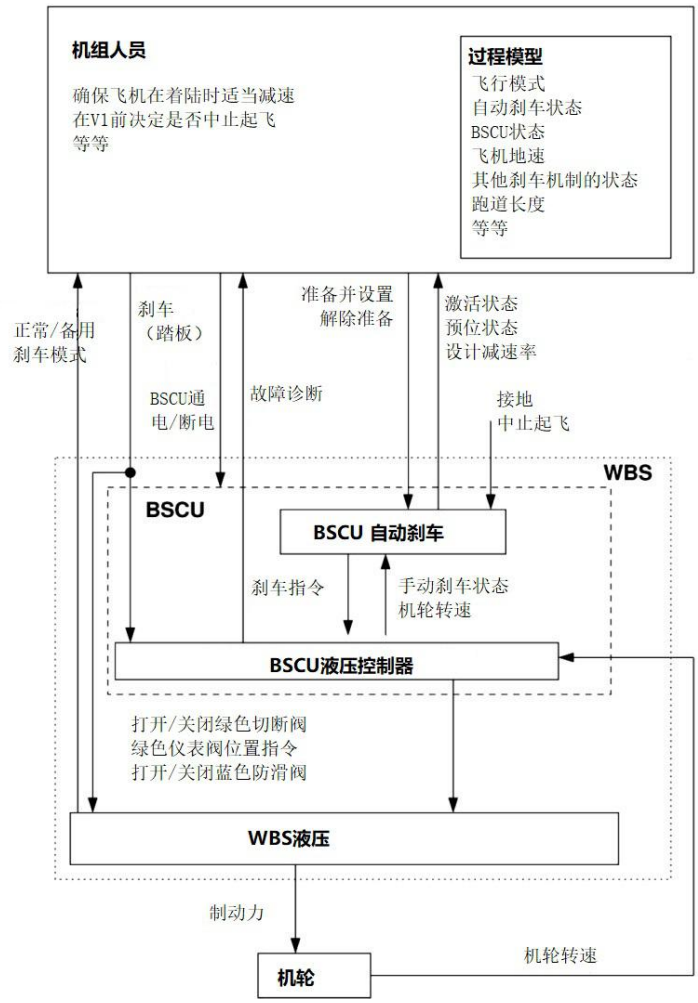


图 3.6：机轮刹车系统控制结构的更详细版本

7. 系统集成

如果在开发过程的早期就使用 STPA 来确定集成系统的安全需求，那么系统集成便会简单许多（至少从安全性角度来说）。部件之间的不安全交互应当在后期开发阶段之前就被排除在系统之外，而不是在这时被发现。此时发现的所有与安全性相关的集成系统的缺陷都应追溯到开发过程中的缺陷（如上所述），并从未来的开发项目中消除。当然，所发现的缺陷需要纳入 STPA 分析中，以确定消除缺陷的约束条件。

8. 系统测试与评估需求的生成

在 V 模型左侧活动的 STPA 分析中生成的致因场景和其他信息可用于生成测试需求和评估程序。测试需求和规划将取决于系统类型以及正在进行的测试。

例如，美国空军试飞员 Dan Montes 在其麻省理工学院博士论文中描述了如何将 STPA 应用于飞机试飞。图 3.7 所示的安全控制结构示例版本是在图 3.2 的基础上

增加了一个在开发和运行控制结构之间内嵌的，独立的测试控制结构。请注意，在试飞项目中可能存在新的危险、不安全的控制措施和致因场景。例如，对于在正常运行环境中可能不存在的试飞环境，也许需要考虑新的背景（例如，测试飞机上异常应力的结果或使用极端控制措施以测试可恢复性）。STPA 可应用于该系统测试控制结构，和用在任何控制结构上一样，其结果可用于试飞安全规划和生成试飞计划。

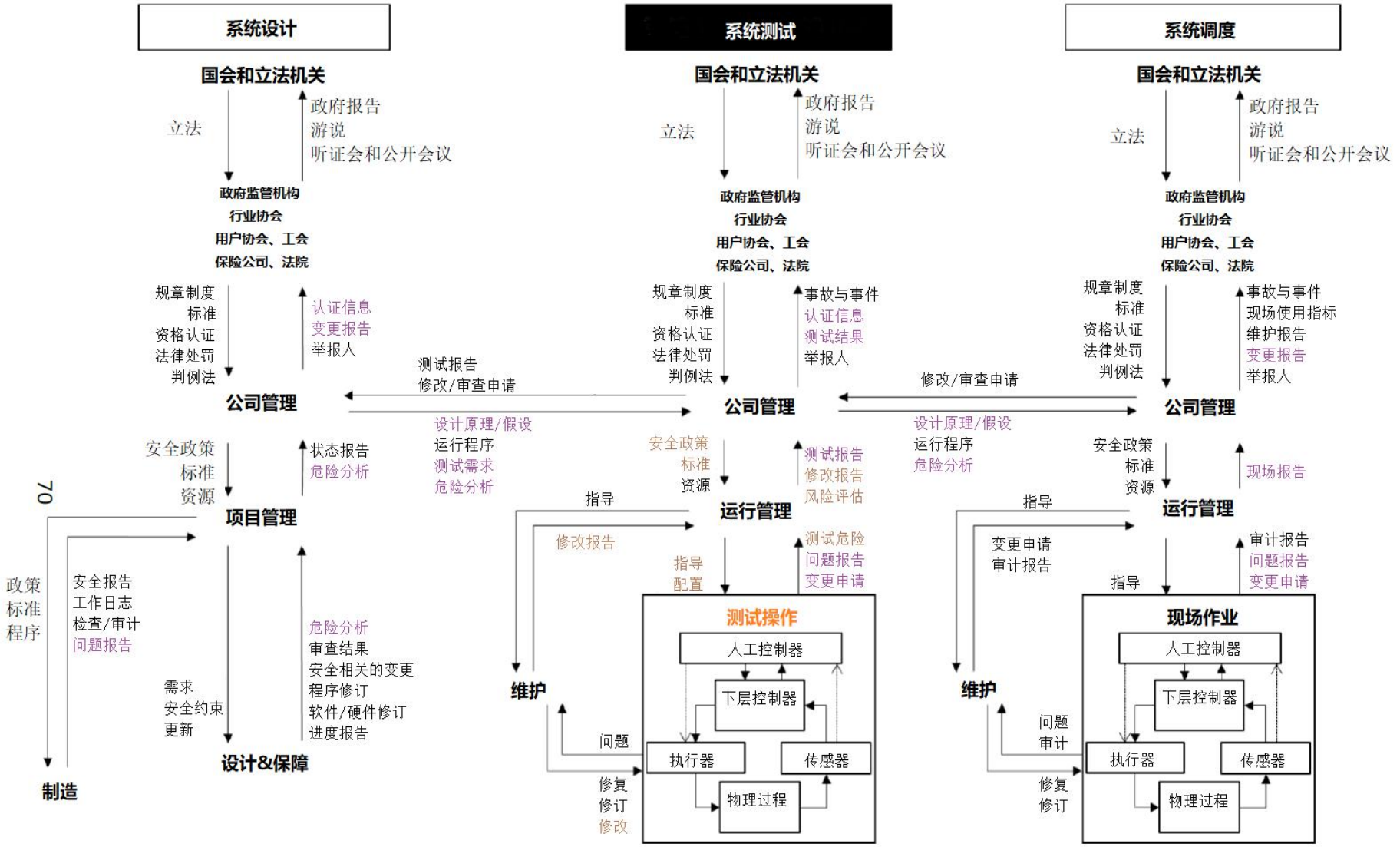
9. 制造控制（生产工程、安全生产）

STPA 已成功应用于生产工程和安全生产领域。如果加工制造和可制造性设计对正在设计中的特定系统很重要，则原始的 STPA 分析中应考虑可生产性（尤其是与制造安全有关的方面）。本手册在不同章节分别描述了关于 STPA 在安全生产领域以及在组织和管理分析中的应用。

10. 运行安全需求的生成（包括不断增加风险的领先指标和安全管理计划）

在系统设计中不能消除或充分控制的致因场景必须告知系统操作员，以便他们实施操作控制。本手册第 6 章所描述的不断增加风险的领先指标可从开发过程或运行安全分析期间生成的 STPA 分析中得出。这些主要指标的目标是在设计过程中（希望在发生严重损失之前）确定有关使用环境的有缺陷假设，并确定何时系统行为会随时间而变化并违反有关使用的原始设计假设。

对系统操作人员和维护人员的要求可以直接从开发时执行的 STPA 分析中得到，因为两者都是系统设计的组成部分。致因场景将为操作需求提供输入。


系统设计
国会和立法机关
立法
政府报告
游说
听证会和公开会议
政府监管机构
行业协会
用户协会、工会
保险公司、法院
规章制度
标准
资格认证
法律处罚
判例法
认证信息
变更报告
举报人
公司管理
测试报告
修改/审查申请
设计原理/假设
运行程序
测试需求
危险分析
安全政策
标准
资源
状态报告
危险分析
项目管理
70
政策
标准
程序
安全报告
工作日志
检查/审计
问题报告
需求
安全约束
更新
危险分析
审查结果
安全相关的变更
程序修订
软件/硬件修订
进度报告
设计&保障
制造
系统测试
国会和立法机关
立法
政府报告
游说
听证会和公开会议
政府监管机构
行业协会
用户协会、工会
保险公司、法院
规章制度
标准
资格认证
法律处罚
判例法
事故与事件
认证信息
测试结果
举报人
公司管理
修改/审查申请
设计原理/假设
运行程序
危险分析
安全政策
标准
资源
测试报告
修改报告
风险评估
运行管理
指导
修改报告
指导
配置
测试危险
问题报告
变更申请
测试操作
人工控制器
下层控制器
执行器
传感器
物理过程
维护
问题
修复
修订
修改
系统调度
国会和立法机关
立法
政府报告
游说
听证会和公开会议
政府监管机构
行业协会
用户协会、工会
保险公司、法院
规章制度
标准
资格认证
法律处罚
判例法
事故与事件
现场使用指标
维护报告
变更报告
举报人
公司管理
设计原理/假设
运行程序
危险分析
安全政策
标准
资源
现场报告
运行管理
指导
变更申请
审计报告
指导
审计报告
问题报告
变更申请
现场作业
人工控制器
下层控制器
执行器
传感器
物理过程
维护
问题
审计
修复
修订

11. 包括监测领先指标在内的运行安全管理

在每个运行环境中都应该有运行风险管理或安全管理系统。STPA 的结果可用于创建运行安全计划和安全需求。此外，它还可以在识别领先指标和监测系统运行情况（以检测向高风险状态的迁移）方面发挥重要作用（请参阅本手册中关于此主题的章节）。

STPA 还可以在事件与事故分析以及变更管理中发挥作用。诸如系统设计变更（升级或修改），背景变更（在不同的环境中或出于不同的目的使用系统）和运行支持变更（物流供应链或维护操作）之类的变更也同样需要考虑。

计划内的变更必须始终进行危险分析。在 STPA 中植入可追溯性应该可以使这一过程更加容易，以此，变更影响可以追溯到特定的系统危险，或表明不影响这些系统危险。计划外的变更通过 STPA 分析生成的领先指标进行处理。

第 4 章：使用 STPA 的安全生产

南希·莱文森

工厂车间或职业安全经常关注工人的行为，而不是超越这一点。安全生产的系统方法是假定人类行为受其所嵌入的系统的影响。目标是确定如何设计工作环境以减少人为差错。STPA 在这种环境下的应用与其在更加技术性的系统中的标准应用是一样的，但公开文献中的例子并不多，因此我们将本章节纳入了本手册。

包括两个示例：一个例子使用先进的自动化系统，另一个使用锁定标定（LOTO），而较少涉及技术性工作场合危险。在这两个例子中，STPA 与当今安全生产的标准方法的主要区别在于，STPA 关注的不仅仅是为工人创建可遵循的程序，而是着眼于整个系统来识别危险场景并考虑如何消除或减少危险场景。

由于下面的第一个示例描述了 STPA 在制造环境中的实际实验应用，因此可以提供所使用的实际评估程序和成本的详细信息。

1. 半自动化制造流程的 STPA 示例

这个例子直接选自 Nathaniel Peper 的硕士论文[17]，在该论文中，他将 STPA 应用到了一个真实的飞机装配过程中，在这里，工厂引入了先进的自动化操作（机器人）。他将过程描述如下：

“…过程的开始是一名工人操作着一辆车，载着飞机的子部件，进入工厂。在工厂内，另外两名工人分别负责通过手持控制的方式，各自驾驶一辆用于子部件进货的自动导引运输车（AGV），将其从工厂内的 AGV 停靠和充电位置开到产品运输车辆（PTV）上的连接点。连接到 PTV 后，两名驾驶员必须协同驾驶联合产品运输系统（PTS）围绕工厂行驶到前面提到的第一辆车上的子部件的位置。此时，另一组工人一起将子部件从车上移到 PTS 上。然后，AGV 的操作员将驾驶 PTS 穿过工厂到达另一个位置，在该位置，子部件可以准备进入自动化制造间。在制造间内有多个机器人和相关的支持设备，这些机器人和设备将执行在子部件中钻孔，以及用其他部件填充钻孔的自动化过程。当子部件、PTS、制造间和操作员都准备就绪时，制造间操作员和控制员将承担 PTS 的控制权，并通过自动化流程执行其工作内容。在此期间，对于不在自动制造间范围内的子部件部分也将同时进行其他工作。一旦整个自动化过程完成，工人将重新承担 PTS 的控制权，并将其向下移动到生产线的下一个位置，以便进行更多的人工工作流程。”

[17] Nathaniel Peper，应用于自动化和安全生产的系统思想，MIT 硕士论文（全球运营计划负责人），2017 年 6 月。

此处的事故或损失是：

A：人员伤亡或疾病

这一工作场合的一般危险：

H1：暴露于不受控制的能量（或可能导致损失的能量水平）中

H2：可能导致受伤的人体运动（或对身体的压力）

H3：暴露在安全水平以上的有毒物质中

H4：暴露在可能影响听力的噪音水平下

H5：长期暴露在不提供基本人类健康要求的环境中

该团队将这组危险细化为与特定应用相关的危险：

H1：暴露于不受控制的能量（或可能导致损失的能量水平）中

H1.1：违反 AGV 与外部物体之间的最小间隔

H1.2：违反 PTV 与外部物体之间的最小间隔

H1.3：违反 AGV 和 PTV 的组合以及与外部物体之间的最小间隔

H1.4：违反机器人与外部物体之间的最小间隔

为了 H1，必须控制当前设计中存在的其他能源，包括电能、热能、气动、液压、重力、机械和非电离辐射（激光）。

H2 和 **H3** 可被细化为：

H2：可能导致受伤的人体运动（或对身体的压力）

H2.1：在常规操作过程中可能导致受伤的人体运动（或对身体的压力）

H2.2：在维护、维修或故障排除过程中可能导致受伤的人体运动（或对身体的压力）

H2.3：在安装、修理或大修过程中可能导致受伤的人体运动（或对身体的压力）

H3：暴露在安全水平以上的有毒物质中

H3.1：在操作系统时，工人暴露在安全水平以上的有毒化学品中

H3.2：在维修设备时，工人暴露在安全水平以上的有毒化学品中

H3.3：在制造过程中，工人暴露在安全水平以上的有毒化学品中

将这些危险改为安全约束是没有价值的，这里不再赘述。

执行实际分析的八人跨职能团队由来自集成、工程、运营和维护部门的成员组成。所有相关人员均被视为系统责任范围内的领导或主题专家。推动人是麻省理工学院的硕士生，曾接受过 STPA 培训，但对引入工厂的新制造工艺知之甚少。

分析工作在三周内完成。在这一过程中，成立的评估小组只用到了两次，每次不到三个小时，而其余的评估则是由推动人和系统各部分的专家进行一对一会谈完成。

由于 STAMP 和 STPA 对于公司来说相对较新，而对团队来说则完全是新的，所以第一次小组会议作为指导会议，以提供所用方法的概述，然后定义团队想要避免的事故（或损失）、危险和系统边界，以及此应用的安全控制结构的初步草案。在制定了安全控制结构的初步草案之后，安排了与各专家的后续会议，以向控制结构的特定部分增加细节。由于 STPA 采用控制结构模型，因此在建立完整的模型后，可以将评估重点放在模型内的特定控制回路上。这样就能够划分建模及分析过程，可以方便地安排会议时间，并在组内每个人的可用时间框架内安排许多会议。在这些会议中，添加了控制行为、过程模型、反馈等细节，并与受影响方一起解决了冲突或问题。

一旦为系统安全控制结构制定了适当的详细级别，一对一或小组讨论将继续进行 STPA 的步骤 1 和 2。然后在另一组中讨论这部分分析的结果，以确认该组的单个结果并讨论系统的潜在缓解措施。这一过程历时三周，在小组、一对一会议和推动人的工作中花费的时间总计约为 300 小时。

在确认危险之后，针对系统的四种不同功能、可能的配置以及每个控制器的 RAA（职责、责任和权限）开发了控制结构。STAMP 和 STPA 中的 RAA 被定义为：

职责：基本的控制职责、过程模型和过程模型变量

权限：控制器可提供的可能的控制行为，以及控制行为的发送方式、发送时间和发送位置。

责任：反馈的内容、提供的时间、方式以及反馈在控制结构中的位置。

由于这个团队初次接触 STPA，因此他们在开始时以团队的方式开发控制结构，以确保抓住所有主要细节。然后，由在系统不同部分拥有相应技术专长的个人和小组对模型进行改进。

四个基于场景的模型如图 4.1、4.2、4.3 和 4.4 所示。

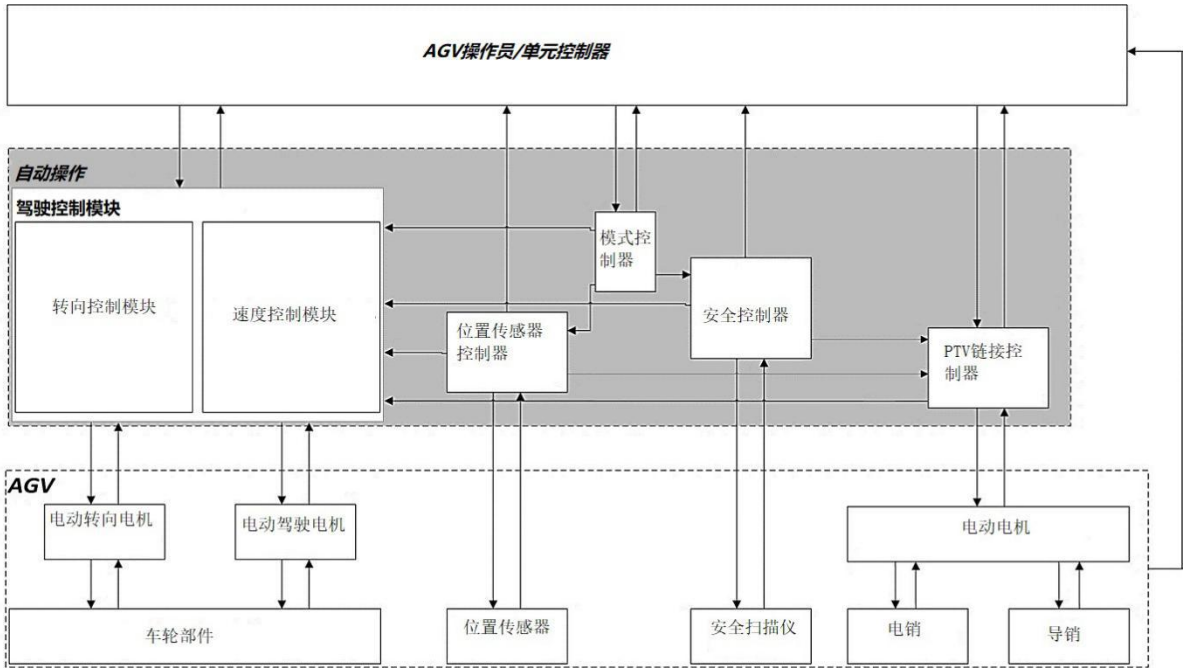


图 4.1：自动导引运输车（AGV）安全控制结构

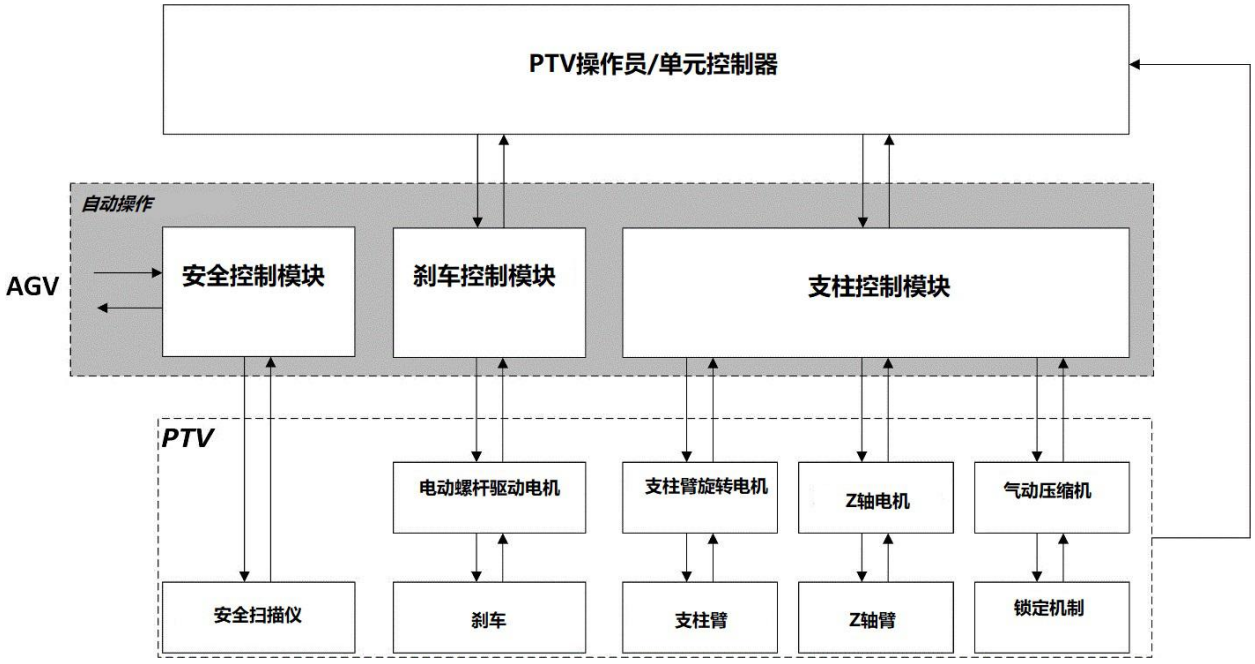


图 4.2：产品运输车辆（PTV）控制结构

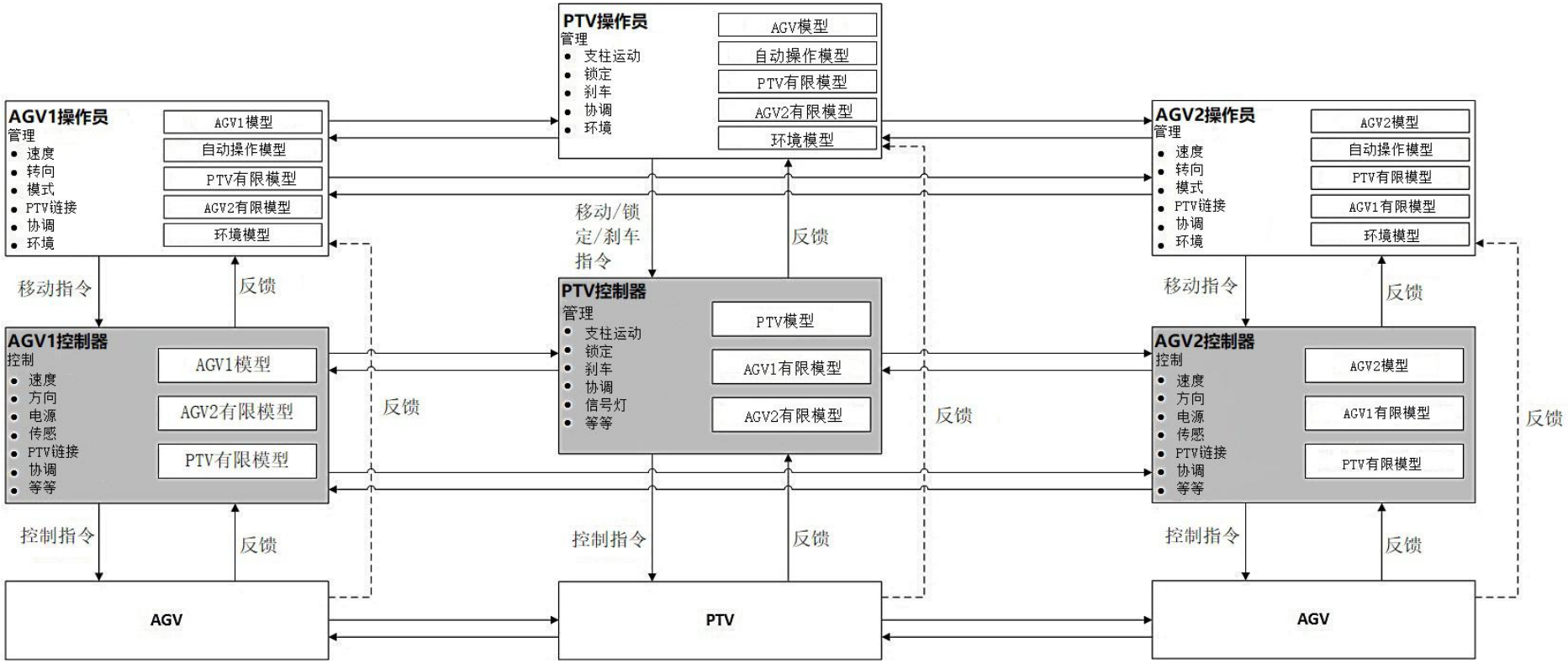


图 4.3：联合产品运输系统安全控制结构

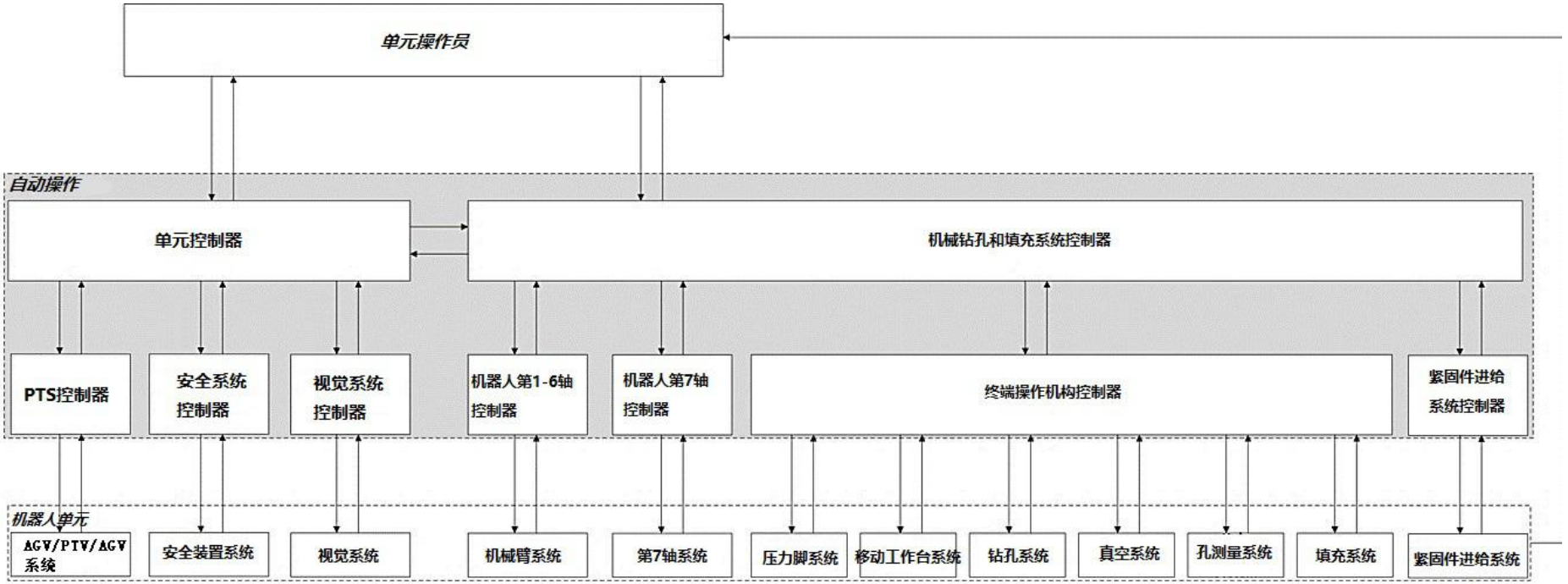


图 4.4：机器人系统安全控制结构

这里只显示了剩余 STPA 分析的一个样本。示例中使用的 AGV 安全控制结构的更详细部分如图 4.5 所示。

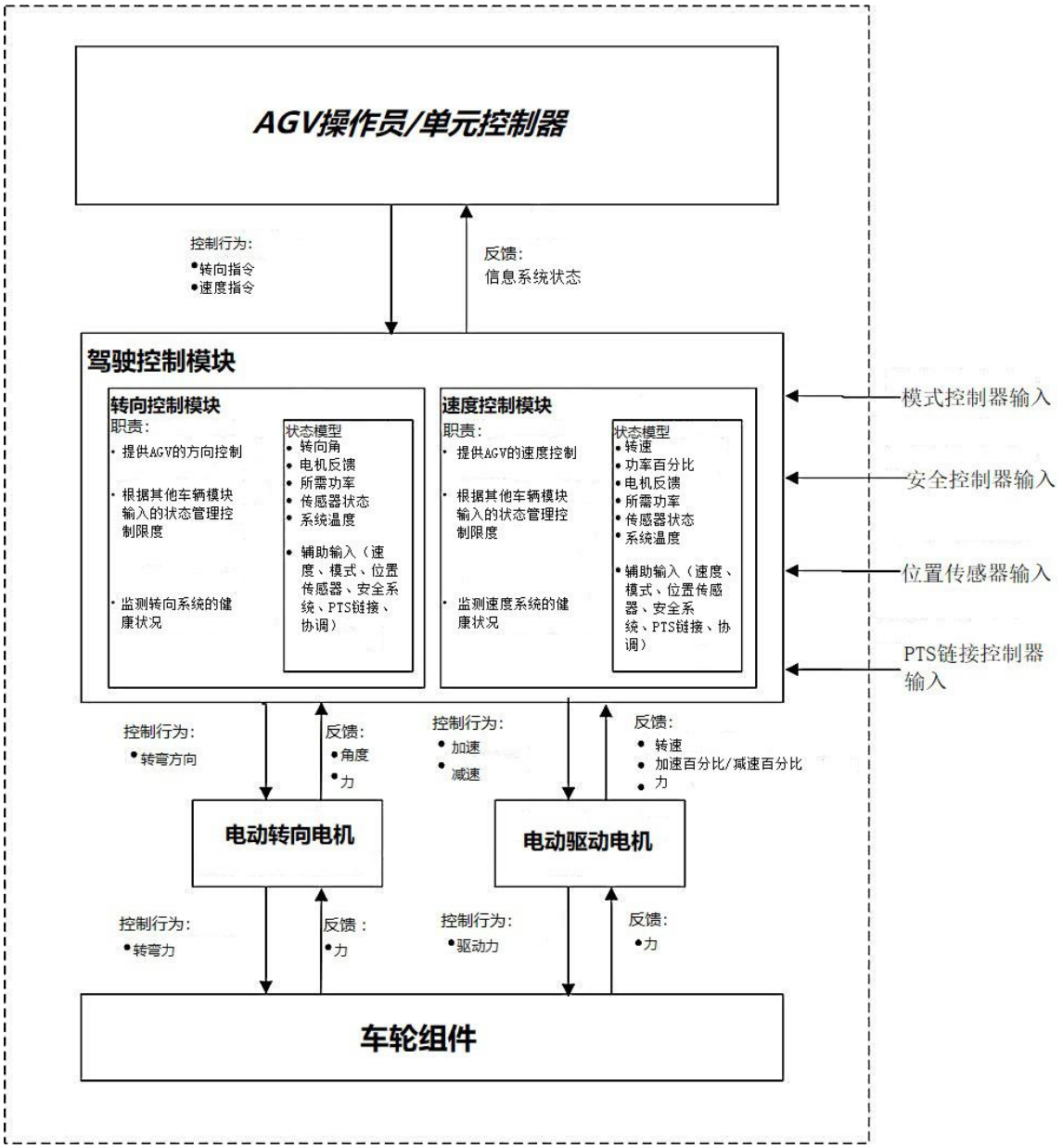


图 4.5：AGV 安全控制结构部分

以下是一个 AGV 的 UCA 表示例：

控制行为	**提供引起的危险**	**不提供引起的危险**	**时间/顺序不正确**	**停止太快/使用太久**
操作员提供驱动命令以驾驶控制模块	**UCA1：**当运动违反物体间最小间隔时，驾驶控制模块命令驱动[H1.1]	**UCA3：**当运动不会违反物体间最小间隔时，驾驶控制模块未没命令驱动[H1.1]	**UCA4：**在通过安全路径方向之前或之后，驾驶控制模块命令驱动[H1.1]	**UCA7：** 当运动已违反物体间最小间隔时，驾驶控制模块命令驱动的时间太久[H1.1]
	UCA2：当工人在操作将要移动的部件时，驾驶控制模块命令驱动[H1]		**UCA5：**在工人停止操作将要移动的部件之前，驾驶控制模块命令驱动[H1]	
			UCA6：在工人开始操作将要移动的部件之后，驾驶控制模块命令驱动[H1]	

UCA1（在运动将违反物体间最小间隔时，驾驶控制模块命令驱动[H1]）的致因场景包括以下示例：

致因场景 1：由于操作人员不熟悉 AGV 的操作，因此导致 AGV 驾驶不当。

- 新操作员未经培训，或培训不足

致因场景 2：由于操作人员未看到物体，或错误地判断了安全路径，因此导致 AGV 驶向外部物体。

- 由于超额工作、环境变化或其他外部因素，导致操作员的注意力不集中
- 在杂乱/限制区域作业
- 其他工人、车辆本身或 AGV/PTV/AGV 组合上的翼梁负荷阻挡了操作员的视线，导致未看到物体

致因场景 3：由于 AGV 的速度/转向设置已更改，导致 AGV 驶向外部物体。

- 操作员不知道或不熟悉对控制器的修改，且依然按照之前的认知继续操作
- 操作员不知道或不熟悉控制器性能下降，且依然按照之前的认知继续操作
- 操作员不知道或不熟悉对 AGV 系统的修改，且依然按照之前的认知继续操作

致因场景 4：由于控制器硬件故障，导致 AGV 驶向外部物体。

- 输入被阻塞，因此即使操作员更改命令，但控制器依然继续之前命令

致因场景 5：由于车辆的启动命令延迟，使得操作员保持驱动命令的时间过长，进而导致 AGV 驶向外部物体。

- 处理延迟和计算过度
- AGV 控制输入超时

致因场景 6：由于提供给操作员的信息不正确或没有提供关于导航所依赖的关键系统态势信息（如安全扫描仪和位置传感器），导致 AGV 驶向障碍物。

- 由于小型 HMI 在 AGV 的侧面且靠近地面，不易获取相关信息，因此操作员错过了反馈。
- 扫描仪开着没状态信息或关着
- 位置传感器开着但没有状态信息或关着

致因场景 7：由于障碍物未被扫描到，因此操作员驾驶 AGV 驶向障碍物。原因可能有以下几种：

- 物体位于安全扫描仪的视野范围之外。
- 工厂内的障碍物位于 PTV 或翼梁的高度。
- PTV 的导轨在 AGV 之上
- AGV 被用作非预期的用途，例如携带超过扫描仪 FOV 的物体。
- 物体在安全扫描仪的视野范围内，但低于检测阈值
- 物体进入视野，且受 AGV 影响，其速度快过车辆扫描和反应的速度
- 随着时间的推移，扫描仪性能下降。
- 扫描仪出现故障，进入不安全状态。

致因场景 8：障碍物虽然被扫描到，但 AGV 并未停止，因此操作员驾驶 AGV 驶向障碍物。原因可能有以下几种：

- 软件编码错误，未命令车辆按预期停止
- 由于 CPU 过载，导致信号处理有延迟
- 由于与 PTV 扫描仪的交互出现不足，导致 PTV 扫描仪不能向 AGV 安全控制模块传输信号

致因场景 9：在扫描仪被故意关闭时，操作员驾驶 AGV 驶向外部物体。

- 扫描仪被禁用，操作员不知情，但仍依靠扫描仪来停止 AGV
- PTV 上的扫描仪被禁用，没有反馈给 AGV 操作员
- 操作员没有看到物体
- 操作员依靠位置感应模块来避免可能出现的障碍物
- 操作员认为车辆处于不同的模式，并期望车辆做出不同的响应
- AGV 已被修改，但操作员不知道或不了解相关更新。
- 命令被延迟，操作员提供命令的时间过长

总的来说，场景包括硬件故障、不正确的过程模型（由反馈缺失或错误引起）、系统组件交互、管理和程序的作用以及工厂中的进度压力这些因素。在团队完成 STPA 的步骤 1 和步骤 2 之后，他们使用这些结果为系统开发新的控制，以消除或防止进入危险状态。下面举几个例子。

一个例子是，系统严重依赖安全扫描仪，以防止 AGV 和 PTV 与周围物体（包括工

人）相撞。针对物体处于系统运动路径中，且操作员或扫描仪未检测到该物体的情形，则系统中没有设计其他控制装置来防止危险或减轻事故影响。由于系统庞大的体积、长度、重量以及 AGV 驱动电机的作用力，任何不安全的控制行为都可能在某些情况下导致严重的事故。虽然该区域的操作员和工人对周围的 AGV 移动保持警惕，且 AGV 安全扫描仪是当前的设计控制手段，但有许多致因场景表明扫描仪不够充分。

首先，正如公司其他部门已经证实的那样，如果安全扫描仪出现故障、未正确校准或设计不当，工人会绕过安全扫描仪，以避免完不成任务和落后于生产计划。安全扫描仪的集成方式不会在扫描仪不使用时阻止 AGV 运行，也不会通知操作员或周围的工人 AGV 运行时，安全系统是禁用的。虽然对用户来说关闭安全系统以完成工作可能不算问题，但如果用户不重新打开安全系统的话，下一个用户将在不知道扫描仪已禁用的情况下操作 AGV。

另一种情况是，AGV 遇到扫描仪无法防护的障碍物。车辆的驱动电机不会向驱动控制器提供关于实际碰撞所遭受的力的反馈。这将导致 AGV 继续向障碍物全力撞击，直到 AGV 驱动控制器收到操作员发出的不同驱动命令。比如物体超过了扫描仪的扫描范围，例如障碍物位于车辆上方，但仍在运输系统顶部的产品路径中；扫描仪经调整后，使车辆周围的覆盖范围出现间隙；扫描仪清除障碍物或故障的控制逻辑，当障碍物仍在扫描范围内时；多个控制器管理集成产品运输系统中的多个扫描系统等等。

AGV 是一种强大的大型移动系统，人们不能靠简单地设置围栏来保护工人的安全。考虑到安全扫描仪可以并且已经被绕过或未按预期工作，团队讨论了意欲实施的多个新控制手段。一个是设计变更，使工人能够关闭扫描仪，并向周围区域的每个人提供扫描仪已被禁用的反馈，例如使用警示灯或声音提示。此外，团队重点关注如何向操作员和周围工人提供 AGV 的反馈。AGV 的原始设计似乎主要是假设 AGV 将会完全按照初始设计和操作员预期的方式来运行，没有提供有效便利的可视反馈来帮助了解系统实际在做什么，而不仅仅知道操作员确信命令做什么事或周围的工作人员认为机器人应当做什么事。

另一个讨论涉及到为 AGV 的驱动电机编程电机反馈逻辑，这在许多应用协作机器人的情况中都可以看到。由于安全扫描器是唯一的控制方案，而且扫描器可能无法保持正常控制的方式很多，因此考虑了对系统的移动功能进行辅助控制的方法。虽然这种控制可能仍然允许违反危险中定义的最小间隔，但 AGV 将停止移动，并避免发生事故或降低事故的严重性。

虽然这些只是 STPA 的过程和结果引起的讨论的几个示例，但系统其余部分的结果非常相似。为了避免定义的系统级危险，建议对需要协调和控制的系统之间的过程模型变量通信、各种运动部件的系统电机反馈逻辑以及从物理系统到操作员和周围工人的反

馈增加进行设计变更。小组的重点是确定系统如何违反现场的安全控制措施，有可能导致事故，然后确定需要如何改变现场控制行为，或需要实施哪些新的控制行为。

2. 使用锁定标定程序的 STPA 示例

第二个简单的例子用于表明 STPA 也适用于更传统的安全生产危险。这里举的例子为传统的锁定标定（LOTO），这是一种安全程序，用于确保危险机器正确关闭，并且在维护或维修工作完成之前无法再次启动。它要求在相关设备开始工作之前，隔离能源并使其失效。然后，将隔离电源锁定，并在锁上放置一个标签，以标识放置该标签的工人。然后由该名工人持有锁钥匙，确保只有他或她才能启动机器。目的是防止机器在处于危险状态或工人直接接触机器时被意外启动。

虽然这一过程似乎相对保险，但事实上，全美汽车工人联合会进行的一项研究发现，1973 年至 1995 年间，其成员中发生的死亡人数的 20%（414 人中的 83 人）是由于锁定标定程序没有适当实施。

事故：

A1：人员受伤

A2：产品受损

A3：制造设备损坏

A4：交付延误

危险：

H1：人员暴露在危险能量中[A1]

H2：产品暴露在过量或不适当的危险能量中[A2]

H3：制造设备暴露在过量或不适当的危险能量中[A3]

H4：工作在关键时刻延误[A4]

安全约束：

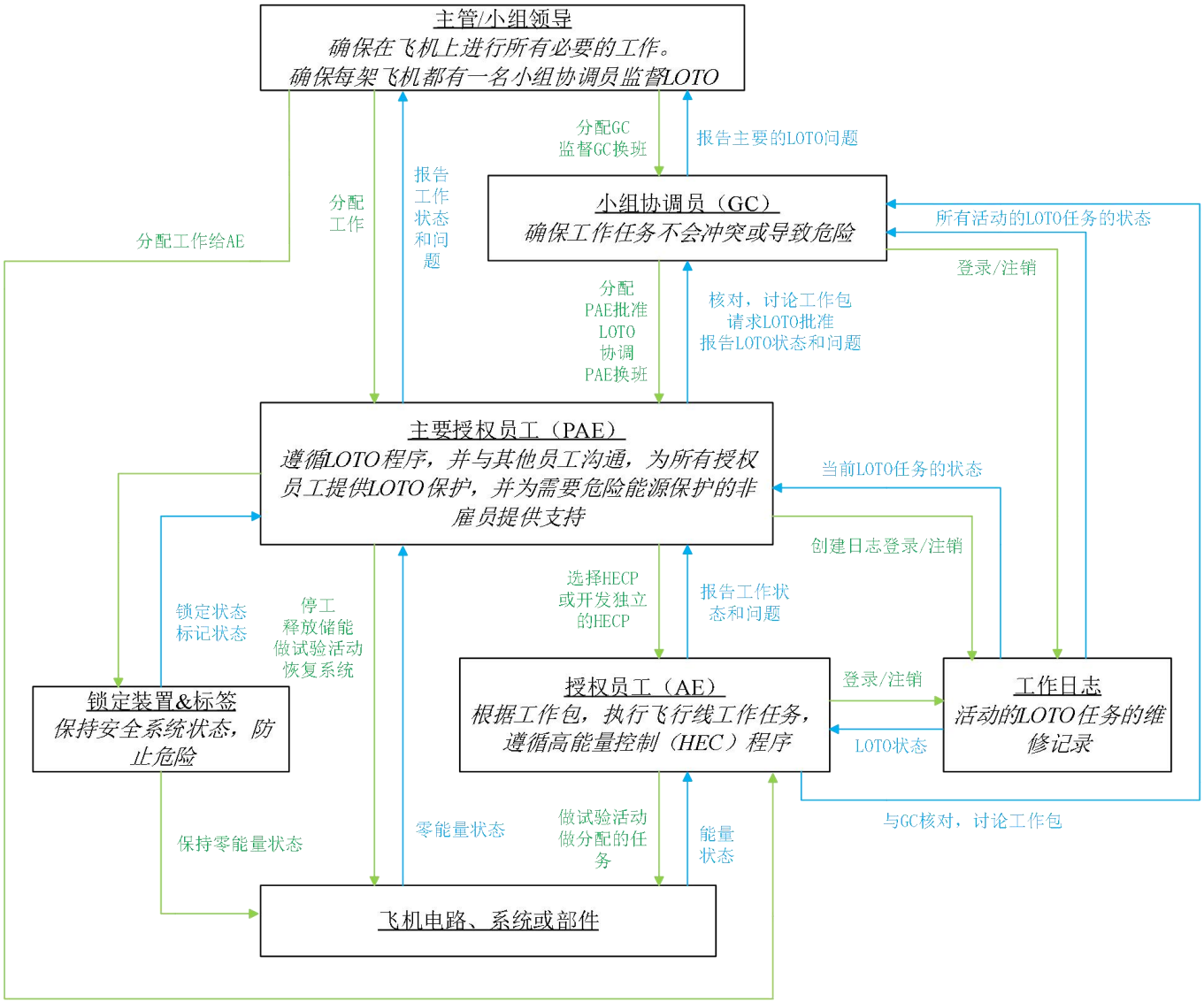
SC1：工人不得在可能暴露于过量或不适当形式的危险能量的环境中工作。[H1]

SC2：危险能量只能在适当和已建立的临界条件下激活。[H2]

SC3：制造设备不得在可能暴露于过量或不适当的危险能量的环境中使用。[H3]

SC4：不得在流程的关键节点（签核、上锁/除锁等）延误工作。[H4]

下图展示了此示例的通用控制结构。



下表展示了两个不安全控制行为的一些致因场景：PAE（主要授权员工）没有切断正在工作的部件的电源以及 PAE 在工作开始后和工作完成前且工人尚未安全离开该区域时，切断正在工作的部件的电源。其中一些被详细指定，以确认修复手段。如果没有的话，那就需要在致因场景中添加更多细节。

PAE 没有切断正在工作的部件的电源	由于 HECP 的特异性不足，她认为她正在切断正确部件的电源。 由于缺乏培训，她认为她正在切断正确部件的电源。 由于高能量控制程序的特异性不足，她认为她正在切断正确部件的电源。 由于缺乏培训，她认为没有必要切断部件电源。 由于缺乏经验，她认为没有必要切断部件电源。 由于她认为其他人已经切断了部件电源，因此没有必要再次切断电源。 由于她认为自己已经切断了部件电源，因此没有必要再次切断电源。
PAE 在工作开始后和完成前切断了正在工作的部件的电源	由于工作不是计划好的，因此她认为工作还没开始。 由于她认为其他人已经切断了部件电源，因此没有必要再次切断电源。 由于她认为自己已经切断了部件电源，因此没有必要再次切断电源（错误的过程模型）。

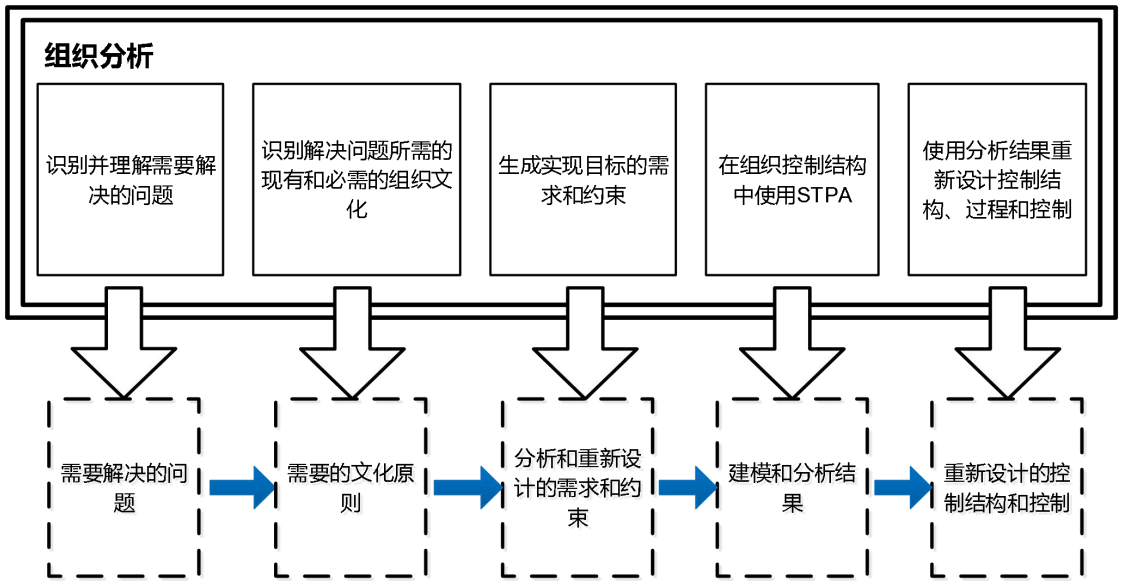
第 5 章 组织和社会分析

南希·莱文森

将 STPA 应用于组织或社会体系本质上是一种系统工程或重组系统工程的工作，但需要进行工程化的产品是组织本身。此类系统工程流程的目标是确定系统当前的进度、发展的方向以及实现的方法。该分析的产品和副产品包括：

1. 需要解决的工程和业务问题，即分析的目标
2. 实现目标所需的组织文化
3. 组织结构的要求和限制
4. 对组织结构的 STPA 分析
5. 分析结果用于：
   a) 设计或改进组织结构
   b) 识别风险和风险的领先指标（leading indicators）
   c) 设计（或重新设计）风险管理流程。

此章将分条介绍上述内容，整个过程情况如下。



STPA 在管理和社会系统中的应用与工程产品虽有相似之处，但在很多方面也有很大不同。最重要的差异就是对待解决问题的初始理解。虽然任何系统工程或重组工程的流程都应从理解问题开始，但在组织中这种理解过程更为复杂。飞机、汽车或核电厂的目标和限制都已得到很好的理解，因此分析此类系统的第一步是让利益相关者共同选择他们想要实现的具体目标和限制。

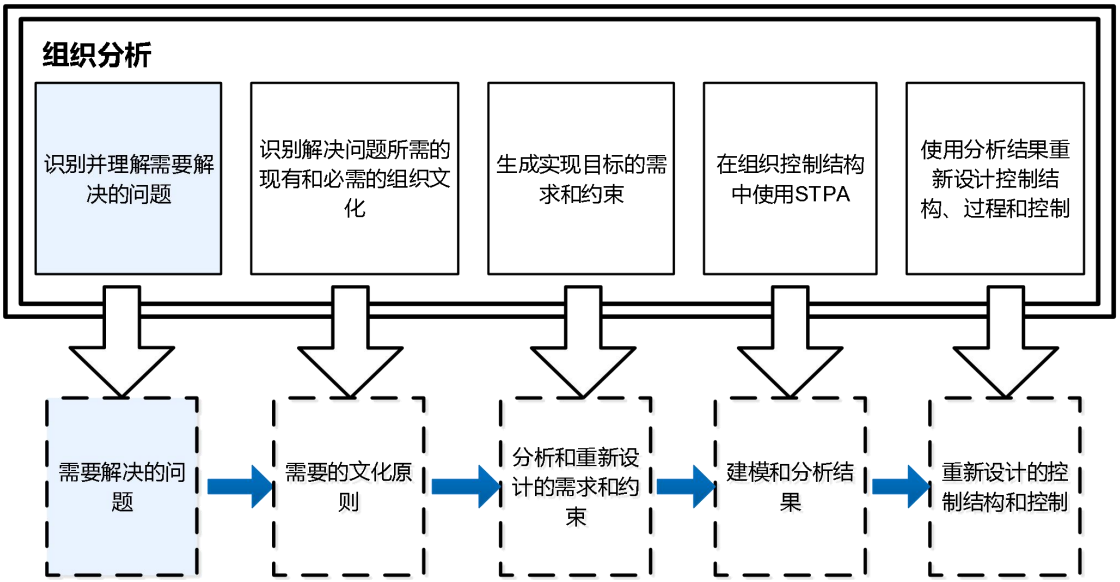
相反，在对社会或管理系统（通常已经存在）进行分析时，第一步是确定并理解为什么目前没有实现目标或如何更好地实现目标。由于本手册中的大多数示例都

侧重于安全性，此处示例将考查其他属性，以说明如何将 STPA 用于安全或安保之外的其他涌现属性。

此处使用的示例是系统工程的有效性，即在预算范围内按时交付产品并满足客户的需求。假设该公司为外部客户（包括政府和商业实体）生产复杂的工程产品。

由于我们在真实公司执行此流程时获得的专有信息无法披露，因此此处仅提供了流程中非隐私的结果及其在真实公司中的使用。

1. 确定待解决的工程和业务问题



虽然并非时时如此，但成功解决问题的第一步总是要理解待解决的问题。在花时间理解要解决的问题之前就确定解决方案并不可能实现成功。

与利益相关者和关键人员的访谈是一种有用的方法，可以通过简单的观察来明确待解决的问题以及感知到的问题原因。其后，将结果信息用于分析当前的系统并促进可能的改进。有关如何进行此类访谈的信息包含在社会科学文献中，我们只是能够使用简单的访谈流程以获取所需信息。首先，描述我们的目标、承诺匿名、探讨简单的问题并给予受访者足够的答题空间。

出于本示例的目的，我们假设可通过访谈环节询问系统工程在其公司中失败的原因。受访者至少应包括管理人员、工程师、可能的话还应包括客户。鉴于对系统工程的关注，结果可能会出现以下对问题的解释以及问题产生的原因：

- 项目在需要时并不总能具备具有专业知识和技能的人员。技能发展低于项目要求。
- 撰写的提案和谈判得到的合同会导致不切实际的预算和/或进度表。因此，项目一开始就过时了，导致产品开发偷工减料，最终只会使结果变得更糟。

- 在此环节中，明确重组工程需要解决的问题非常重要；同样，为满足客户需求、避免后续停滞甚至返工，在确定解决方案之前先明确待解决的问题也是非常重要且必要的。但在本示例公司中，前期规划和开发的早期阶段付出的努力太少。这包括在了解客户需求、产品应用环境上花费的时间太少。工程师将时间浪费在解决由不良计划引起的问题上，以及在开发早期阶段由于没有确定目标产品的关键假设和要求引起的问题上。草率的决定浪费了大量时间和金钱，并且经常导致产品不如预期。
- 缺乏早期规划导致交付延迟和不断重新规划，既浪费时间又会使项目“波折”，同时如果大量时间浪费在会议和重新规划上，进度会被严重拖慢，导致预算和进度表也出现问题。草率的改变会加剧问题，原因是这些改变会影响分析，从而导致变更管理开始降级。
- 项目风险分析和缓解是得过且过的，成了“例行公事”的活动。分配给识别和减轻风险的资源很少。通常，风险分析的优先级低于其他活动，经常委托给没有足够经验的新工程师。此外，风险缓解通常没有专门资金，实现风险分析的步骤也不包含在主进度表中。
- 创建系统工程门和里程碑，以确保为后续步骤做好准备。虽然大多数人都了解对门的需求，但为保证进度依然存在很大压力。通过将短期关注放在预先确定的进度表上，可以将长期进度表和质量问题纳入其中。项目经理通常会跳过里程碑（“门”），假设它们将在下一阶段涵盖。这些项目从未赶上进度并最终需要付出极大代价解决问题。在项目后期需要更改时，预算和进度表会被进一步分解。更糟糕的是，许多（如果不是大多数）关键决策都是在开发早期阶段做出的，后续如果不付出巨大成本，通常不太可能撤消，也很难实现重新设计。此时，唯一可行的解决方案通常是创建变通方法，而这会导致产品出现缺陷。草率做出的决定和变更往往会造成比需要解决的问题更严重的问题。
- 返工涉及到的问题是对设计意图和理由的记录不够充分。缺乏此类文档可能会导致之前被丢弃的解决方案会被再次建议和尝试。
- 所有这一切都导致英雄主义和牛仔工程成为标准，并得到管理层的高度回报。通常，有人需要对满足进度和实现成本目标负责任，但是对于首期质量却很少有问责制或高回报，这意味着对早期开发阶段的不足仅能进行很少的更改。
- 无论成功失败，反馈不足都会导致学习不充分，难以提供持续改进。并非所有问题都可以靠严格遵循流程解决。管理人员认为，只要制定了正确的流程，所有系统工程问题都将得到解决，这忽视了涉及人员。想要利用流程解决问题、或发展和支持发展流程中的人员，管理层需要更加智慧的方

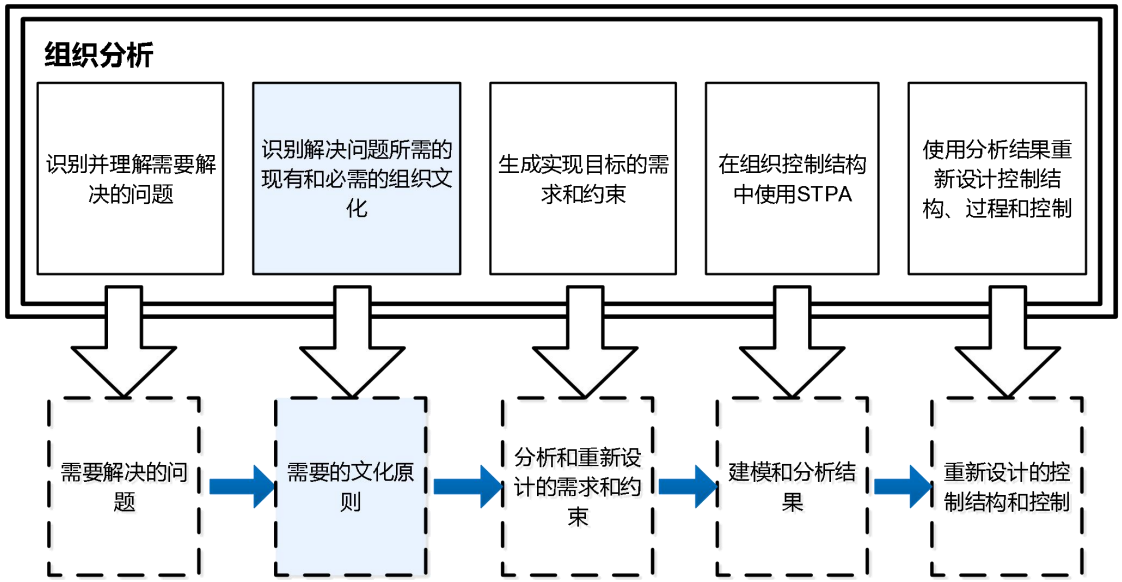
法。该公司擅长“观察”课程（收集数据），但不汲取“学到的”课程。

- 学习需要创新、承担风险和保持灵活性（与严格执行特定流程相反）。但是，在工作中，对促进创新的因素没有给予足够的授权和支持。在访谈中反映出的一个共同观点是，员工害怕承担风险和创新，因为很少有创新会得到奖励，事实上，很可能还会受到惩罚。因此，他们不会主动做其工作本身之外的事情。对失败的恐惧和由此产生的惩罚可能会扼杀创新。目前的工程结构被描述为无法承担颠覆性创新，且几乎没有能力处理变更。
- 访谈中发现缺乏创新的另一个原因是，大量的工程创新和资源都用来解决问题了而非创造更好的产品，而这些问题产生的原因是实践不当或风险管理不足。
- 在可生产性和保障性（可维护性）方面的设计不足，导致产品更迭时不可避免的产生大量成本和质量问题。
- 在已完成的设计或架构中过分强调安全和安保，但在产品设计初期并没有对安全和安保给予足够重视。
- 对于分包了部分产品的公司，我们经常听到供应商监督无效，进而导致进度延后甚至需要返工的情况。
- 当进度与质量冲突时，在项目和业务战略的考虑中缺乏技术决策的独立性。与之相伴的问题可能还包括缺乏公开、诚实的沟通以及缺乏降低技术风险和问题的独立途径。
- 工程师和管理人员无法获取做出正确决策所需的信息和反馈。有时他们忽略了已经掌握的信息。反馈回路不存在或无效。有人抱怨经常收到反馈但却被忽略了。有一种众所周知的现象称为确认性偏差，这意味着人们倾向于相信支持其假设的信息，而忽略那些不支持其假设的信息。确认偏差与防御性回避有关，意思是人们会怀疑或忽视其不想听到的信息。
- 一些常见问题，特别是制造业或客户的问题，与设计者和产品装配者之间的关系、或设计者与系统操作者之间的关系有关。常见问题还与缺乏工作指导、缺乏机械师和设计工程师等人员的反馈有关。

虽然上述问题的数量可能令人惊讶，但事实上，它们在工业系统工程组织中相当普遍。

在访谈中，我们通常还会尝试了解当前的组织文化。

2. 实现目标所需的组织文化



要讨论组织文化，我们需要从一个有效的定义开始。Edgar Shein 的定义是：

组织文化是一个团体或组织所共同持有的价值观和深层文化预设，这是决策的基础。

为了在组织中进行最有效的重要变革，有必要将文化确立为更能促进变革的文化。

在访谈期间，我们定义了组织文化的含义，并向受访者询问他们认为组织中需要哪些文化上的更新。答案反映了受访者认为文化应该促进实践的想法。我们在采访中获得反馈的一些例子涉及系统工程的有效性以及提高有效性所需的组织文化，包括：

- 减少对进度表的关注，不再将其作为决策的主要驱动力
- 能够向客户和公司内部说“不”。提供营销和销售答案而不是工程答案。
- 思维应超越立即要做的项目和客户
- 更有效，更清晰的沟通
- 从失败和成功中汲取更多经验
- 初期花费更多时间做好准备，以便后续节省时间
- 更多合作
- 重视工程师并尊重他们
- 认为只有当客户使用产品并感到满意时，工作才算完成
- 不要害怕寻求帮助
- 对产品保持开放性和诚信；不要对传信人抱有偏见
- 考虑更大的公司目标而不仅仅是本地目标
- 不是“推迟痛苦”，而是让短期优先于长期。

- 确保风险缓解计划得到合适的资源且落地，而不仅仅停留在 沟通层面。

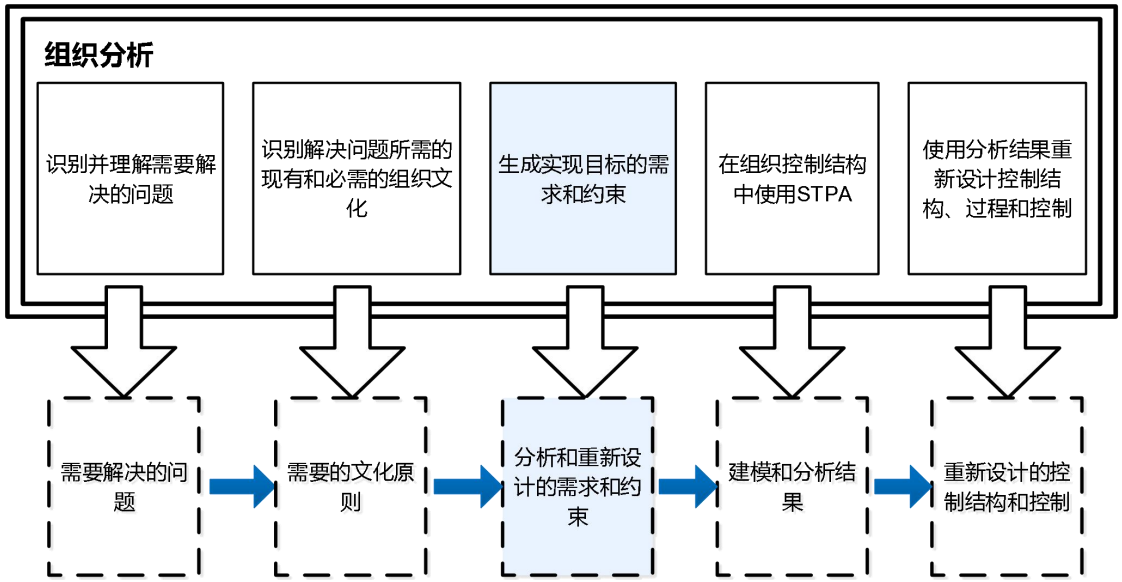
显然，这些观点与访谈中发现的问题有关。通常我们流程的下一步是实施这些建议，同时实施在上一节中列出的、和访谈中发现的问题，并将它们重新表述为营造目标组织文化所需的文化原则（共享组织价值系统）。结果是：如果在组织设计中实施这一系列人们普遍认同的信念或价值观，将大大有助于解决已发现的问题。以下是一组示例文化原则，来源于目前为止提供的示例访谈结果：

- 质量和安全：提高质量和安全性可以降低成本同时加快进度，从长远来看，还可以提高利润。安全是生产力和盈利能力的关键决定因素。
- 规划：预先进行良好的系统工程可以节省昂贵的返工工作。良好的工程设计不应该需要额外的工作甚至返工。
- 劳动力和技能开发：有效地利用劳动力、促进技能开发能够提高士气、降低成本、且有助于培养公司未来的人才基础。
- 技术独立性：如果能够将技术决策独立于程序执行决策之外，技术决策将更好。超越眼前项目的思考将为整个企业带来长期利益。
- 沟通：有效、清晰、一致、及时的沟通和信息共享对实现工程目标至关重要。
- 学习和改进：有效的工程需要不断学习和改进，包括从成功和失败中学习。增加知识与创建产品同样重要。
- 创新：对新知识和技术（创新）的投资可促进增长、提高生产力。
- 协作：成功的工程需要跨越领域和结构边界，应包括所有适用的工程学科。
- 风险管理：风险评估和管理是实现绩效目标的关键。

生成一系列所需的文化原则（价值观和信念）只是一个起点; 它还需要领导团队和工程师的评审、编辑和（完成后的）支持。目标应该是启动讨论并开始寻找改变公司文化的方法。

一旦就有助于实现目标的工程文化形成共识，那么推行该文化的前进路径就变得更加可行。

3. 在有效的组织文化中生成要推行的要求和限制



在确定系统工程工作的目标之后，关键一步就是创建需求。需求是由问题识别过程的结果和所需的工程文化共同产生的。以下是本手册本节中使用样本分析结果生成的一组示例要求。它们根据每个商定的文化目标进行具体重述。

质量和安全：提高质量和安全性可降低成本和加快进度。安全是生产力和盈利能力的关键决定因素。

- 每个项目的人员都应具备成功实现项目目标所需的技能和专业知识。
- 安全和安保分析应从概念分析阶段开始，并在整个开发过程中持续使用安全和安保分析，目的是在制定设计决策时将安全和安保分析纳入其中。
- 门（里程碑）控制应包括质量和安全分析。
- 对项目经理进行第一次质量奖励[18]。
- 所有产品开发都应强调可生产性和可支持性。
- 应在每个开发阶段都强调验证。必须避免试图绕过或减少验证步骤的行为。
- 公司应监督分包商的产品质量和安全。

规划：预先做好系统工程可以节省昂贵的返工。良好的工程设计不应该需要额外的工作甚至返工。

- 在提案中预测未来性能时，应（至少有一部分）基于执行能力和工程可行性，而不仅仅基于外部期望，如客户目标。

[18] “安全管理系统”一章讨论了激励“正确”行为的重要性。例如，因满足会议时间表奖励经理人可能会导致质量下降。

- 详细的项目规划和系统分析应在项目开始时进行，包括详细的风险管理和应急计划; 全面了解客户需求、产品需要解决的问题以及产品使用环境; 早期活动还应包括明确和减轻风险，可通过模拟，原型设计等方式。
- 需要更改时，应实施严格的变更管理。
- 分包决策应使用适当的工程输入来进行。
- 工程计划应包括进度表和成本利润和/或重新分配资源的途径，以便处理不可避免的“技术上的未知难题”。
- 计划应适应不断变化的客户需求。

劳动力和技能开发：有效的劳动力和技能开发可以提高士气、降低成本，并有助于培养我们未来的人才基础。

- 应对技能需求进行预测，从而提供适当的培训、指导、传授成长经验。特别是系统工程人员应该接受培训，成为每个项目的核心部分。
- 在可能的情况下，应根据特定领域的经验选择领导者。应为未来的领导者开发这种经验。
- 管理层应重视员工的贡献，并采取措施确保员工得到适当的回报。

技术独立性：如果能够将技术决策独立于程序执行决策之外，技术决策会更好。

- 独立机构（非项目经理）应负责确定是否已通过门。
- 当进度与质量冲突时，技术决策应独立于进度和业务战略等考虑因素。独立的技术机构应负责确保这种独立性。
- 比项目经理权力更高的的主管负责解决项目经理与独立技术机构之间的冲突。
- 应有独立的途径来降低技术风险和其他工程问题。

沟通：有效、清晰、一致、及时的沟通和信息共享对实现工程目标至关重要。

- 应在工程和业务收购之间建立更好的沟通，包括对正准备的提案的可执行性进行反馈，以及已完成（或取消）的合同的成功原因或严重问题进行反馈。
- 应建立反馈和信息渠道，以确保决策者获得做出适当决策所需的全部信息。

学习和改进：有效的工程需要不断学习和改进，包括从成功和失败中学习。增加知识与创建产品同样重要。

- 项目经理应撰写检视报告，以便从成功和失败中学习经验。项目经理在检

视报告中表现出的诚实和彻底应该得到相应的奖励[19]。

- 应建立项目管理信息系统，不仅包括过去项目的数据，还包括从成功和失败中吸取的经验教训。

创新：投资新知识和技术（创新）能够推动增长、提高生产力。

- 应对创新、灵活性和适当的承担风险进行鼓励和嘉奖。
- 组织应提供资源，为新技术挑战、机遇以及大多数不可避免的技术问题提供创新的解决方案。
- 应识别并消除妨碍工程师承担创新风险的文化障碍。

协作：成功的工程需要跨越领域和结构性界限，应包括所有适用的工程学科。

- 应促进和优化跨领域和跨结构性界限的工作，以促进协作。
- 应跨领域和跨界建立工作组，工作组应包括所有适用领域。

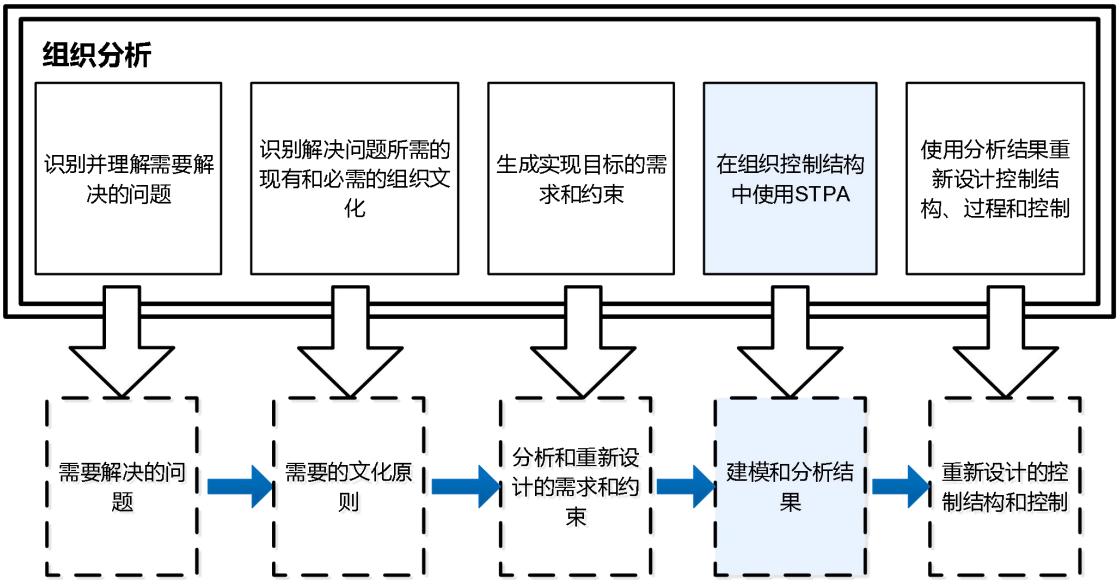
风险管理：风险评估和管理是实现绩效目标的关键。

- 风险识别和评估应由知识最渊博的专家确定，同时明确传达给所有相关人员，并根据项目目标和客户需求制定一套明确的有计划有资源的缓解措施。
- 应从项目一开始就基于早期风险评估和经验规划风险缓解措施。风险缓解措施应得到资助并纳入主计划。
- 应向所有相关人员清楚地传达风险。
- 风险管理计划应包括充足的成本和进度余量，以应对“未知的未知数”。
- 风险管理应有适当的控制措施，促使管理者不能忽视现实风险。
- 风险管理应得到适当的资源调配。

这些只是部分示例，是以文化原则作为组织基础衍生而来的一些要求，其他原则也是可能的。只要要求达成一致，就可以开始实际的建模和分析。

[19] 残酷但诚实的项目检视被认为是微软成功的原因之一。会花时间详细分析项目结果，并且领导者应该诚实地了解成功和失败的原因。军方有类似的概念叫做 After Action Reviews。从自己（以及其他人）的经验中学习是未来成功的必要条件。没有奖励人们诚实地评估他们自己的表现并分享经验教训可能会导致新人被提升，这可能会导致同样的错误一犯再犯，从而阻碍积累学习经验。

4. 组织架构的 STPA 分析



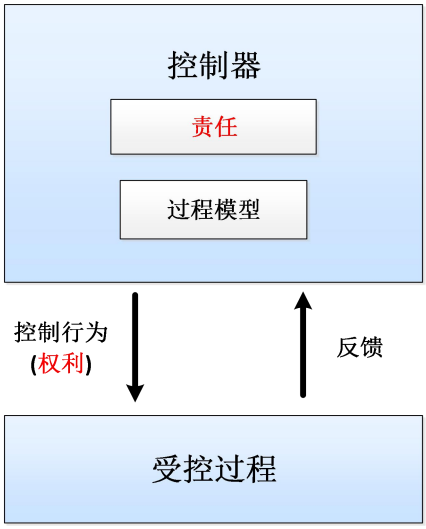
流程的下一步是对当前的系统工程控制结构进行建模，检查已确立的要求是否已经在当前系统中得以执行，这些活动是否都指定了责任人，并判别阶段一中明确的问题是否已充分解决，做出的要求是否已认真执行。检查结果可为下一步更改提供参考，以更好地实现组织设计中的要求。

当建模完成时或至少完全理解了整体控制架构时，可以开始对结构进行危险性分析。危险性分析包括生成领先指标，以便尽早明确组织中与系统工程相关的不正确假设，以及随着组织和环境因素变化不再成立的假设。这些领先指标还应提供一种方法，帮助确定解决问题的完善程度和目标完成程度。

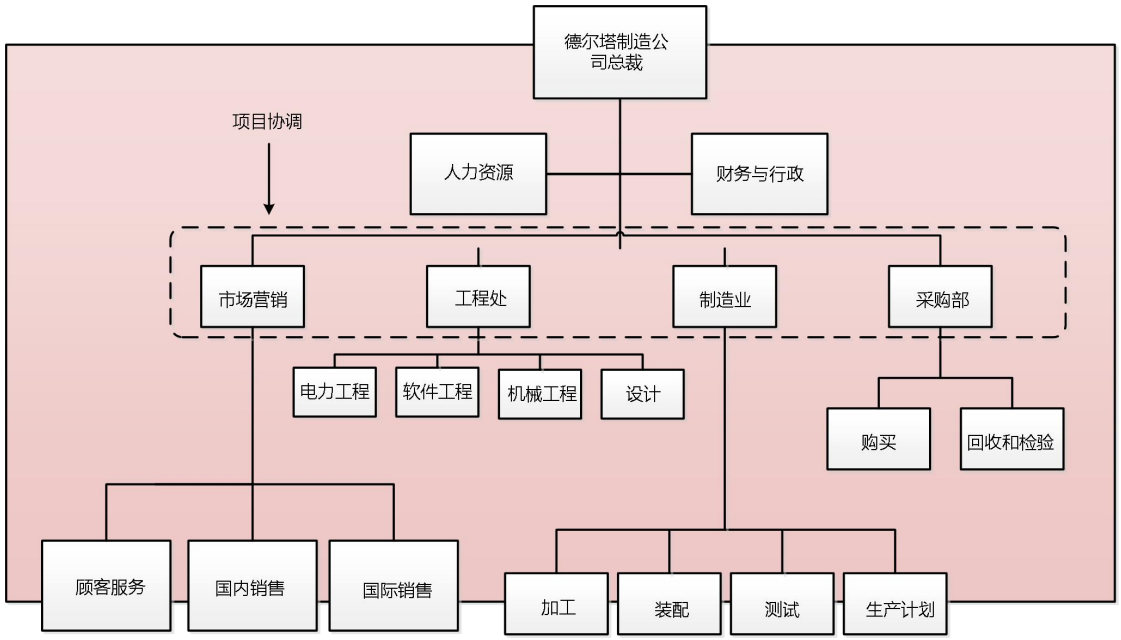
我们在建模和分析过程中发现人们经常知道他们问题的解决方案，但他们对已确定的进度表和问题感到不知所措，因为他们没有充足时间实施解决方案。我们还发现，分层控制结构模型的使用有助于解决方案确定小组组内的沟通，还有助于对重新设计的解决方案进行改进和评估。要找到问题的解决方案，确立一个能够指导讨论和解决问题的通用模型也是必不可少的。例如，有时在没有任何复杂分析的情况下，可以观察到某个实体试图控制另一个实体，而没有任何反馈来指导其控制（管理）行为。增加有效反馈可能会对解决当前问题大有帮助。

下图显示了本手册前文所示的通用循环结构，但增加了责任、权限和问责制的标准管理要求。

(责任制)



下图是建模工作生成的系统工程架构示例。



请注意，简单地假设项目协调发生在此示例中 4 个相同级别的管理人员中而不对任何个人分配具体责任，这不太可能是公司总裁的合理做法。

当然，该模型包括结构中职责、权限和责任的规范。例如，产品开发公司的一般管理人员所分配的职责可能包括：

人力资源：

培训、招聘、劳动力和技能发展

项目管理

项目计划和概念开发

技术决策

整合项目中技术工作和交流

风险管理和意外情况管理

里程碑和进度管理

预算/资源管理

分包管理

团队动力/沟通

系统工程

执行技术决策

专业工程师

在组件设计中实施技术决策和项目要求

业务开发

引进新业务

为项目竞标

计划管理

计划/项目风险管理

监督项目进度和成功，劳动力管理

管理分包商关系，供应链

行政管理

实施技术/流程改进

业务风险管理

预估未来所需的技能和工程

确保当前和未来的技能和工程能够满足需求

劳动力“管理”

研究和工程改进

创新、新科技/流程改进

明确未来业务和技术趋势和所需技能

质量保证和系统安全

监督产品质量

评估产品安全

适当的后续步骤是：（1）非正式评审会议，以确保控制结构的正确性并识别缺陷;（2）利用从需求到控制结构责任的可追溯性以分析控制结构模型；（3）（如有必要）应用正式的 STPA 分析。目标是确定组织控制结构中每个要求（前面指定的）的实现情况和方式，以及控制结构中与这些要求相关的所有缺陷。下面将提供实例。

第一步是通过专家审核当前组织设计来验证模型。可能需要多次审核，因为每个受访者可能只熟悉结构的一部分。

一旦整体控制结构得到验证，下一步就是检查在访谈中确定的控制结构，并分析控制结构设计，以确定改进决策所需的整体变化。通常有些东西会立即显现出来，例如结构的复杂性; 多人控制同一流程或责任重叠; 缺乏对决策的适当且独立的监督; 或者缺少反馈路径（即，人们可能在没有所需的全部信息或信息不准确的情况下做出决定）。

为了进一步找到缺陷，完成从需求到控制结构的跟踪。下表显示了根据上述要求进行潜在追踪的结果示例。

要求	**控制结构内容**	**识别的缺陷**
风险管理	项目管理、流程管理	成本和计划边际通常不足以应对突发事件 风险识别有时不完整，重要风险被忽略
技术独立性	项目管理、流程管理	技术决策不能独立于程序和业务战略之外，且受项目经理影响
质量和安全	质量保证、项目管理、流程管理、行政管理	安全是以保证为导向的，不是由开发团队执行或与开发团队密切合作。 许多质量保证是“清单”和敷衍。 门（里程碑）控制不包括质量和安全性分析 项目经理在进度和预算绩效方面得到的回报高于在质量和安全性方面得到的回报。 不强调生产能力和保障能力 承包商在质量和安全方面受到最低限的

		监督。

确定的缺陷将为重新设计提供指导，旨在提高需求的满足度。有些改进将需要很少的结构变化。例如，承包商对质量和安全的监督不足，只需要向承包商的经理简单分配责任，并提供可能需要的质量保证和安全方面的协助。其他调整则可能需要更大范围的结构更改。例如，如果关于承包商监督的政策不足或缺失，则需要在政策制定和变更方面做出一些努力。

一个较为复杂的更改示例是技术独立性需要新的管理方法，例如建立独立的权威技术机构[20]以及新的联系沟通和冲突解决的决策流程。独立性往往随着时间的推移而逐渐消失，因此领先指标也适用于此（参见下一个关于领先指标的部分）。

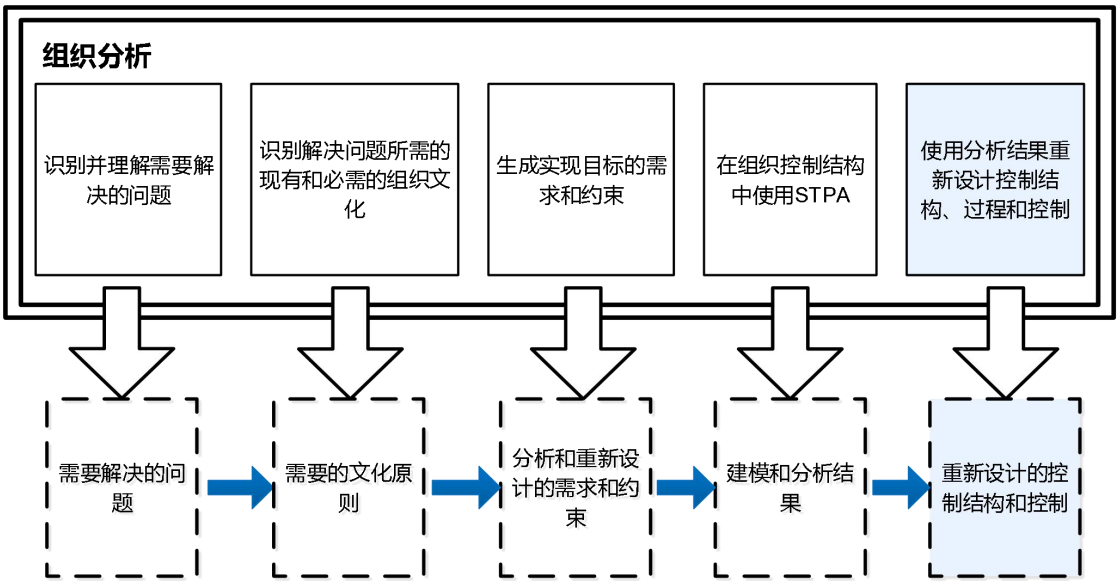
最后一个示例是，将安全性与工程决策分开，使其成为事后保证或合规性工作，这几乎不会对设计工作产生影响。后期发现安全或质量问题要付出极大代价，因此将安全责任与基础工程工作分开是一个错误，尽管这一错误十分常见。确定优化交互的最佳策略可能需要在职责、沟通和结构方面进行大量更改。例如，在项目工程中可能有一个安全工程组件和一个单独的质量保证组件，每个组件都有非常不同的职责和执行不同的活动。

在某些情况下，应用完整的 STPA 对控制结构或其部分进行致因分析可能是适当且有用的。例如，由于大多数公司都在努力执行这项重要任务，因此可能并不清楚为什么风险管理不够充分。确定所有潜在原因可能是有用的，例如缺乏关于实际风险的信息（有缺陷的心智模型），强调积极性而非潜在风险的文化，风险评估培训不足，有缺陷的程序，管理层施加压力降低重要项目的风险等。在探究所有潜在原因后，可以启动一项研究确定该组织的运行原因以及哪些变化可能有助于消除或控制上述问题。虽然某些假设可以采用临时方式制定，但使用 STPA 中的步骤流程可以提高完整性并降低错过重要因果因素的可能性。组织中每种风险管理控制的潜在缺陷可能有不同的原因。例如，没有充分强调识别某些类型的风险可能涉及不同

[20] 美国海军 SUBSAFE 项目中的独立技术管理局在过去 50 多年来一直非常有效地消除了该项目所涉及的事故。您可以在 Nancy Leveson，Engineering a Safer World，2012，第 14 章以及下一章安全管理系统中阅读更多相关信息。

的因果因素，或没有提供足够的应急计划。我们发现使用 STPA 识别风险比有经验的员工进行头脑风暴更加有效。

5. 在再次设计的组织流程中应用研究结果



流程的最后一步是将分析结果用于整个控制结构的重新设计和风险管理流程。控制设计中的部分风险管理涉及领先指标计划。将 STPA 应用于控制结构或其中的一部分有助于确定风险不断增加的领先指标，这将在下一章中讨论。

第 6 章：使用 STPA 识别领先指标[21]

南希·莱韦森

风险管理在任何项目中都是重要一环，人们需要仔细识别并管理计划的和组织的风险。风险评估通常是即时评估，以一组专家集思广益，思考项目中可能存在的风险，随后评估风险等级，通常是随机的。STPA 则为风险管理计划提供了一种结构化程序。

当领先指标用于风险管理时，它们通常也是临时创建的。正如本手册前面所述的产品安全和其他涌现性质一样，使用更正式，结构化的系统思考过程可能更有效。使用 STPA 创建领先指标还具有以下优势：由此产生的指标可直接追溯到相关危险和事故。本章描述了一种识别领先指标的结构化方法，包括使用 STPA 的结果和整个系统工程过程的其他结果。本章内容有：

- 什么是领先指标？
- 目前领先指标的用途是什么？
- 什么是基于假设的领先指标？
- 如何
  - ✓ 使用 STAMP 和 STPA 识别合适高效的领先指标
  - ✓ 生成领先指标的依据的假设
  - ✓ 使用假设创建基于假设的领先指标项目
  - ✓ 将领先指标融入风险管理项目
- 可行性和最终想法

1. 什么是领先指标?

领先指标用于在事故发生之前预测事故发生的可能性，并由此采取措施防止事故发生。它们基于这样的假设：重大事故不是由一组独特的随机近端事件引起的。相反，事故产生原因有

随着时间的推移，组织会向风险不断增加的状态迁移

放松了安保和管控

相互冲突的目标和权衡以及对风险的认知减少导致更具风险的行为。

这一假设意味着重大事故会随着时间推移而发生，因此，人们可以检测这一危

[21] 这一章和本手册中的其他章节一样，编写了实用性质的“方法”指南，包括一些介绍性的解释性信息。更完整详尽的技术过程，参见南希·G·莱韦森，通过领先的安全指标进行危险管理的系统方法，《可靠性工程和系统安全》，埃尔塞维尔出版社，2014。

险路径并加以干预。领先指标的出现则是需要人为干预的信号。

对于组织来说，领先指标可以在产品、服务或组织行为的任何一方面开始偏离正轨时，提供早期提示，避免其对组织的目标造成深远影响。

2. 目前领先指标的用途是什么？

人们已经在识别通用领先指标方面付出了大量努力，尤其是在石化行业。人们试图找到一些普遍适用的指标，这些指标可以在任何企业甚至行业中引起注意。这其中包括维护积压、小事故、设备故障率以及员工文化(信仰)调查结果。员工文化（信仰）则假设所有或大多数事故都是人工引起的。这种确定领先指标的尝试目前看来效果不好。它们主要也是针对职业安全而非系统安全。

使用事件和事故报告鉴定领先指标也是一种尝试。该尝试存在的一个缺点是只能考虑先前发生过的事情。这些先前发生的事情已经被管控或消除了。另一种方法是使用危险分析来预测未来情景。这里存在的问题是传统危险分析技术预测的场景十分有限，几乎都集中在硬件故障上。

导致事故的社会，组织和管理因素已被假设为有用的领先指标，但大多数情况下，只有少数（5 或 6）被确定，关于这些一般指标的有效性的实际数据很少。问题在于，这些假设并非基于事故发生的方式和原因的任何模型或框架，因此非常不完整。此外，这些领先指标仅包括大量组织共享的最常见因素，这些因素可能会忽略特定组织中最重要的因素。

本章描述的方案不是尝试确定所有事故的一般领先指标，而是根据特定公司甚至特定产品或服务中的安全控制设计确定领先指标。我们称它们为基于假设的领先指标。

3. 什么是基于假设的领先指标？

将假设作为确定领先指标的风险管理计划基础的想法，在最初提出时不包括工程项目。兰德（RAND）在 20 世纪 80 年代后期开发了基于假设的规划（ABP）方法，以协助美国陆军客户的中长期国防计划，减少不确定因素性并管理陆军任务中的风险。

我们采用了这一基本思想，并创建了一种工程风险管理方法。该方法的基本前提为：通过特定组织，产品或操作的安全设计程序中的假设，可以确定不断增加风险的有效领先指标。所有工程都涉及对操作系统行为的假设，即使该系统是组织或管理结构。这些假设可包括，例如，硬件组件随时间进程而增加的故障率，产品的使用方式以及产品的使用环境或产品提供服务的环境，对人员使用的工具的基本培

训，决策所需信息和信息渠道操作有效率的假设。其他外部或环境假设可能来自客户需求，这一点会随着时间变化，整体市场的改变而变动。

因此，基于假设的领先指标的正式定义为：

> *基于假设的领先指标是警告标志，可用于监控整体流程，以检测某一假设的破坏或薄弱得可以产生危害、或者假设的有效性发生改变。*

领先指标计划的目标应该是监控设计系统的设计者所做出的社会，管理和产品/服务假设，以发现最初不正确的假设——例如关于人或其他系统组件的信念在特定的环境中的表现——以及那些因为人们优化了局部目标或环境变化，导致最初正确但随着时间的推移变得不正确的假设。我们找到的领先指标不是试图适用于所有组织和系统，而是根据与特定设计相关的安全假设，为特定组织或系统设计生成的。

虽然组织结构可能无法以其设计方式运作的原因可能有很多，但一个重要原因是组织文化，即组织的目标和价值观以及组织结构内部[22]，不符合实现计划目标所需的组织结构目标和价值观。或者，文化可能会随着时间的推移而退化，可能是由竞争，财务或其他压力和人事变动影响的，或者由于退化到了熟悉的习惯中。作为有效的领先指标计划的一部分，人们需要检测并纠正这种文化偏差。

基于假设的领先指标来自何处？从最基本的意义上讲，基于假设的领先指标是系统目标如何实现的假设。一般来说，涉及三种类型的假设：

- 工程系统初始设计中使用的模型和假设（包括组织结构）是正确的。
- 系统的构建和操作将以设计者假定的方式进行。
- 系统中的原始模型和假设不会随着时间（也许人们试图改进或优化程序）或环境变化而更改。

在基于假设的领先指标程序中使用这些类型的假设来决定应该检查什么，如何检查，何时检查，如果检查确定了假设不再正确（或者可能一直都是错的），应该采取何种行动，何时行动以及采取何种行动。此程序包涵三个方面：

1. 确定适当和有效的领先指标，
2. 创建领先指标监控程序投入使用，以及
3. 将此监控系统嵌入精心设计的风险管理计划中。仅以前两个方面的形式进行检测是不够的——必须有一个管理流程，以便在领先指标出现需要采取行动时，立刻行动。

现在依次阐述这三个方面。

[22] 关于设计安全管理系统一章包含了对组织文化更详细的定义和讨论。

4. 使用 STAMP 和 STPA 确定适当和有效的领先指标

尽管人们为避免事故产生付出了大量努力，但仍无法完全避免。从理论上讲，如果我们设计一个安全系统，即消除或充分控制或减轻所有危险并且这一状态不会改变，那么就不应该发生事故。问题是这些情况在实践中通常都不正确：没有哪个工程的过程是完美的，人类行为也是如此。此外，每个系统及其环境都会随着时间的推移而发生变化。寻求更有效的领先指标的出发点是要考虑事故的一般原因。

A. 事故原因的一般分类

事故原因可能来自技术系统开发，运营和管理。通常，在事故情景中可以找到几种或所有这些类型的原因。下面描述了这三个方面各自发生事故的原因：

设计与制造

- 危险分析不充分：有关系统危害或用于识别系统危害过程的假设不成立
    - HA 未执行或未完成
    - 由于危险分析过程中的不足或执行方式的原因，未发现某些危险。忽略了重要原因，因此未进行处理
    - 识别了危害，但由于认为危害“不太可能”发生，所以未对其进行处理。
- 对已识别的危害的控制和缓解措施设计不足，可能是由于工程知识不足或对操作的不恰当假设
- 。
- 控制和缓解措施的构建不足。

运行

- 设计人员假设在操作期间存在的控制实际上不存在，未使用或结果无效。
- 控制存在，得到了使用，并且最初是有效的，但随着时间的推移，带来的变化违反了控制原始设计的基本假设。
    - 随着条件变化而出现新的危害，这些危害在开发过程中没有预料到，或者被判定为不太可能发生
    - 物理控制和缓解措施随着时间的推移会以分析和设计过程中没有考虑的方式退化
    - 组件（包括人类）随时间推移而行为不同（违反设计和分析期间的假设）
    - 系统环境随时间而变化（违反设计和分析过程中的假设）

管理

- 安全管理系统设计存在缺陷（参见本手册中设计有效安全管理系统方法相关章节）。

- 安全管理系统可能是有效的，但并没完全按其设计（和假设）的方式运行。虽然原因包括许多不当行为，其中一个一般原因是安全文化，即组织在安全方面的目标和价值，随着时间的推移而降低或从一开始就是无效的。此外，做出安全相关决策的人的行为可能会受到竞争，财务或其他压力的影响。
- 为防止事故发生，我们必须消除或减少这些原因的产生。可在事故发生之前使用领先指标检测它们。让我们逐个看看这三类事故原因。

设计和制造过程所使用的危险分析过程可能存在不充分或被忽略的现象。有时，由于进度压力，人们会跳过危险分析过程或仅表面上执行危险分析过程。目前事故频繁发生，原因是危险分析技术无法识别当今复杂的软件密集型系统中存在的新原因。在系统开发和实施过程中消除与事故因果关系相关的这些因素在表面上似乎相对简单，但实际上或政治上，在某些组织中可能相当困难。本手册提出了有关将 STPA 整合到组织中的建议，但更多内容则不在本手册范围内。

造成事故的一个非常常见的原因是在开发过程中识别出来了技术因素，但是因为它们被计算为不太可能发生而被排除在外。对此最好的解决方案是不使用概率风险评估或非正式可能性评估作为忽略某些危害或其致因场景的理由。这种解决方案在政治上可能不可行，但可以创建一个领先指标，用于识别何时发生的事件被错误地判断为不可能或太遥远可忽略。在我最近分析的化学工厂爆炸和火灾中，催化剂在 20 世纪 70 年代被确定为惰性，因此它导致事故的可能性被判定为零。然而，多年来催化剂的组成和制造方法发生了变化。事实上，在同一家公司拥有的化工厂中，有两次爆炸与这种催化剂有关。然而，当该公司的工程师设计一家新工厂（在两次爆炸之后）时，设计师仍然认为催化剂是惰性的并且在风险评估中忽略它。导致该工厂后来发生了与使用相同催化剂有关的严重事故。随着时间推移，原始假设明显变得不正确。在前两次爆炸中的第一次爆炸之后就应该触发一个领先指标，更不用说在其中两次爆炸之后，并且在爆炸发生后设计假设应随之改变。

另一种情况，在危险分析中可以识别出原因，但可能很难设计和实施控制和缓解措施。一个例子是物理控制机制的设计不合理。可能涉及简单的计算或知识错误，但不正确的假设也可以发挥重要作用。当利用冗余防止故障时，独立性的相关通用工程设计假设就是一个例子。考虑一下马孔多（深水地平线）防喷器。控制井喷风险的方法存在冗余，但冗余单元包含共同原因故障模式。虽然事实上无效的防喷器曾导致几起严重事故，但人们仍然普遍接受了防喷器从未失败这一理念。有明确的领先指标表明假设存在错误，但被人们忽略了。另一个共同原因失败的例子是挑战者号，只是这次，在挑战者号飞行失败前多年，人们便科学地检查了 O 形环独立性的假设并使其无效。但是，马歇尔太空中心数据库中没有记录这一变化，挑战者号却在这里进行了发射。鉴于大量事故都涉及共模/原因失败，所以这似乎是一个重要的工程设计假设，需要像其他导致许多事故的假设一样重新审视。挑战者号这种情

况，还涉及一个有缺陷的安全信息系统。

有时，假定在运行期间使用的危险控制设计不当且效果不佳，可能是存在与其他控制或激励措施未识别的冲突。它们也可能没有得到充分实施或使用。如前所述，迁移到违反支持系统安全的运营假设的可能性需要反映在一系列领先指标中。例如，运营商可能会开始采取捷径或关闭安全设备，以便更有效地运营;或者，在空中交通管制系统中，空域可能比最初系统设计期间考虑的更加拥挤。必须在此处检查需要检查的假设并告知给操作员。

最后，安全管理系统的设计和运行所涉及的假设（见第 6 章）可能是错误的或者变得错误了。参与行业或组织的人的目标和价值观，即安全文化，是一个重要的假设，即错误可能是事故的一个主要因素，必须在一系列领先指标中反映出来。例如，安全政策是每个公司或组织传达期望的个人安全文化和行为的基本要求。必须有一种方法来衡量人们遵守政策的程度以及情况是否随着时间的推移而减弱。在事故发生后，通常会发现人们违反了管理行为和决策的假设，必须监控这些假设。

5. 如何生成基于假设的领先指标

使用 STPA，可以至少部分地确定适当和有效的领先指标。目标是使用分层控制结构模型和组织中内置的控制来识别关于涌现属性的限制所基于的假设，并确定如何检测旨在实现系统目标的控制的一切弱化的有效性。如本节所述，可以在系统设计过程中确定其他假设。

下表显示了假设的可能来源。

假设可能来自：

- 在概念开发过程中产生的高级系统目标，尤其是包括数量在内的目标
- 系统目标产生的系统级要求
- 关于系统运行的外部环境假设
- 与安全相关的环境要求和约束（包括对系统使用的限制）所强加的系统行为要求
- STPA 产生的危害，分层控制结构，不安全控制措施和致因场景
- 设计用于管理致因场景的功能
- 为管理无法通过系统设计功能完全处理的致因场景而创建的操作要求
- 安全相关控制设计的局限性，包括操作控制
    - 与实现功能要求相关的限制
    - 与环境假设相关的限制
    - 与设计或操作程序中无法完全消除或控制的危险和因果关系相关的限制
    - 在系统设计期间进行权衡所产生的限制

一种名为 TCAS（空中防撞系统）的飞机碰撞系统可用于说明领先指标可以基于关键假设产生。我和我的学生在 20 世纪 90 年代初协助完成了 TCAS 的官方认证，后来我们两个人创建了一个 TCAS 意图规范[23]。

意图规范的一个关键部分是记录系统安全性所依据的假设。这里的例子来自该规范，几乎所有例子都与用于 TCAS 认证的定性危险分析[24]产生的情景有关。一些假设不是来自危险分析方案，而是来自设计目标和预期 TCAS 运行的环境。虽然在设计和认证 TCAS 时不存在 STPA，但我们已经证明 STPA 会产生使用故障树所产生的所有危险情况以及故障树分析中遗漏的许多故障树。因此，下面的所有示例假设都可以从 STPA 结果生成。

假设的产生应该从早期概念发展期间开始。当高级系统的目标和要求包含数字时，可能会找到一些假设迹象，但这不是必然的。在 TCAS 中，TCAS 系统的两个目标是：

> ***G1***：*为广泛的国家空域系统（NAS）用户提供价格合理且兼容的防撞系统选项。*
>
> ***G2***：*在所有气象条件下检测与其他飞机可能发生的潜在空中碰撞；整个可航行空域，包括未被 ATC 一级或二级雷达系统覆盖的空域，以及没有地面设备的空域。*

系统目标用于创建系统要求。同时可以生成这些要求的假设。例如，

> ***1.18***：*TCAS 应为任何两架水平接近的飞机提供防撞保护，飞机速度最高可达 1200 节，垂直速度可达每分钟 10,000 英尺[**G1**]。*
>
> *<u>假设：</u>这一要求源于这样的假设：商用飞机在垂直爬升或下降期间可以飞行高达 600 节和 5000 英尺/分钟，因此两架飞机的水平接近速度可以高达 1200 节，垂直接近速度可高达 10,000 英尺/分钟。*

这个假设是未来需要检查的一个例子，以确保技术变化不与它相矛盾而使基于它的所有技术设计决策变得脆弱(这可以通过意图或其他规范中的可追溯性指标来识别)。

另一个系统要求的例子是：

> ***1.19.1***：*TCAS 应在航路和终端区域运行，其空域密度最高为每平方海里 0.3 飞机（即 5 海里内 24 架飞机）[**G2**]。*
>
> *<u>假设：</u>到 1990 年，空域密度可能会增加到这个水平，这将是未来 20 年的*

---

[23] 《意图规范》的基础是系统理论、系统工程原理、人类问题解决的心理学研究以及如何增强人类问题解决。目的是帮助人类处理特定复杂系统的复杂性。更多内容见《工程建设一个更安全的世界》第 10 章。

[24] 在 TCAS 开发和认证工作中使用了非常广泛的故障树分析——包括详细的人员和软件行为。因为目标是完全性而不是量化，所以没有尝试将故障树框限制为可以量化的事物。由此产生的故障树是我在 35 年的安全工程中看到的最完整的故障树。它不包括 STPA 可以找到的所有情景，但它也没有排除当目标是为情景产生概率时不可能进行量化的情景。

最大密度。

同样，未来的飞机性能限制可能会发生变化，或者空域管理可能会发生重大变化，例如垂直间隔减少或处理空中交通的方式变得非常不同。TCAS 中的大量计算基于基本要求 1.19.1 的假设，如果安全参数发生变化，则需要对其进行监控以重新评估。

可以识别（和指定）另一种类型的假设以解释决策或记录设计所基于的基本信息。例如，设计可以基于关于系统其运行环境的假设。TCAS 的例子有：

EA1：*飞机之间存在高度完整的通信*

EA2：*配备 TCAS 的飞机上装有 Mode-S 空中交通管制应答机*[25]*。*

EA3：*所有飞机都有工作的应答机*

EA4：*所有飞机都有合法的识别代码*

EA5：*高度信息可从侵入目标获得，最小精度为 100 英尺。*

EA6：*为 TCAS 设备提供飞机气压高度的高度测量系统将满足 RTCA 标准要求*

EA7：*威胁飞机不会采取突然机动来阻止 TCAS 的逃生机动。*

集成到受控空域的新技术和新型飞机可能违反这些假设。**EA4** 是非技术假设的一个例子。识别代码通常由每个国家的航空当局提供。这一假设需要由国际协议保证，并由一些国际机构监督。飞机具有运行应答机（**EA3**）的示例假设可以由特定国家的空域规则强制执行，并且必须由某些组确保执行。这一假设的真实性至关重要，因为 TCAS 在没有运行应答机的情况下不会显示任何飞机，也不会提供解决决断咨询（RA）[26]。**EA7** 是关于飞行员和空中交通管制系统行为的假设的一个例子，也可能因无人驾驶或其他类型的新飞机进入空域而受到侵犯。

一些假设可能是由环境要求和限制强加给系统的。这些假设可能导致新系统的使用受到限制(这将需要进行假设检查)，或者可能表明需要系统安全和其他分析来确定必须对正在创建的系统或更大范围的系统施加的限制，以确保安全。TCAS 的例子包括:

E1：*非 TCAS 设备与 TCAS 的行为或相互作用不得降低 TCAS 设备的性能或 TCAS 相互作用的设备的性能。*

E2：*在飞机环境警报中，层次结构应为：风切变优先，然后是近地警告系统（GPWS），然后是 TCAS。*

E3：*TCAS 警报和建议必须独立于使用主告警和警报系统的飞机。*

只有在飞机中引入主要设计变更时才需要检查这些假设。

STPA 将为假设和领先指标过程提供重要贡献，首先从系统危险规范开始，然

[25] 飞机应答器发送信息，帮助空中交通管制维持飞机间隔。

[26] 决断咨询基本上是一种避开入侵者的行为。

后进行整体建模、分析过程。TCAS 中系统危险的例子有：

H1：*TCAS 造成或引起中空碰撞（NMAC），即为一对受控飞机违反了最低间隔标准。*

H2：*TCAS 造成或引起飞机太靠近固定障碍或自然地形。*

H3：*TCAS 造成或引起飞行员失去对飞机的控制。*

H4：*TCAS 干扰其他与安全相关的飞机系统（例如，近地警告）。*

H5：*TCAS 干扰地面空中交通管制系统（例如，应答机到地面或雷达或无线电服务的传输）。*

H6：*TCAS 干扰与安全相关的 ATC 建议（例如，避开限制空域或恶劣天气）。*

安全关键假设的第一组基本假设是，在设计和操作得当的系统中不会发生危险。发生了这些危险任何一种(即使没有导致事故)都应触发对安全工程过程的全面评审，在这种情况下，安全工程过程是用于消除或减轻 TCAS 危险的过程。然而，在危险发生后检查假设可能为时过晚，无法避免损失，危险的识别是一个起点，通过使用 STPA 识别可能导致危险的场景，通过这个起点可以实现早期的检查。

即使在这高层也可以推断出其他假设，例如，有一个基于地面的空中交通管制系统（将来可能会改变），而 TCAS 不会干扰其运作。虽然危险很少发生变化，但在对系统进行更改时可能会引入新的危险，并且可能会破坏用于处理危险的程序。

检查危险的发生和不安全的控制措施也提供了有关危险分析过程本身充分性的重要信息。危险分析和安全工程的目标是识别危险，然后消除或预防危害。如果无法阻止它们，则需要减轻它们。当然，工程师认为被消除或阻止的危险应该永远不会发生。如果他们这样做，则此事件表明工程过程中存在缺陷，或者可能是对操作系统的假设存在缺陷，飞行员或空中交通管制员行为的假设就是一个例子。它不仅足以修复技术流程。开发过程中可能导致危险行为的漏洞也需要修复。

理想情况下，在实际危险状态发生之前，领先指标将确定工程实践或操作行为假设中的缺陷。通过 STPA 确定的危险情景的基本假设，可以实现这一目标。分层控制结构设计，UCA 和致因场景是有用的。

控制结构本身包含可用于创建领先指标的假设。2002 年德国 Uberlingen 的空中撞机证明了控制结构在安全运行中的假设作用。飞行员对潜在 NMAC 的响应负有潜在责任涉及了三个小组：TCAS，地面空中交通管制员和航空公司运营中心。后者提供了航空公司程序，用于响应 TCAS 警报并训练飞行员。其他两组的相关性是显而易见的。显然，这三个控制器之间的任何潜在冲突和协调问题，都需要在空中交通管理系统的整体设计和运行中得到解决。就 TCAS 而言，在有冲突的咨询的情况下，假定始终遵循 TCAS 提供的决断咨询（RA）。设计人员认为，由于当时没有切实可行的方法给地面管制员下载有关可能已发给机组人员的任何 TCAS 建议的

信息，飞行员应立即实施 TCAS 咨询，副驾驶将通过无线电将 TCAS 警报信息发送到地面 ATC，以便地面空中交通管制员知道空域状态和给出的建议。航空公司应提供适当的程序和培训来实施该协议。

Uberlinger 空中撞机违反了关于如何处理相互矛盾的建议的几个重要假设。例如，地面 ATC 塔台应该有两个空中交通管制员，当地面 ATC 系统提供的咨询与 TCAS 之间存在冲突时，飞行员应该遵循 TCAS 机动，并假设航空公司培训飞行员在这种冲突情况下遵循 TCAS 警报。在悲剧发生时，当晚处理这两架飞机的瑞士空中交通管制中心违反了第一个假设。目前尚不清楚第二个是否曾被侵犯，因为并未检查过该信息。第三个假设，即所涉及的航空公司培训飞行员在提出相互矛盾的建议时始终遵循 TCAS 也很长时间没有实现，但显然没有人负责确保此类培训的开展或他们没有一直履行这一责任。关于控制结构运行的这些不正确假设，如果检查出来了，就可能成为设计的控制结构降级的领先指标。

不安全的控制行为，安全约束和致因场景也可以提供关键假设的相关信息。致因场景用于设计控件，从而形成控件创建的假设。例如，**H5** 引起以下系统安全约束：

> ***SC.2***：*TCAS 不得干扰地面 ATC 系统或其他飞机信息传输到地面 ATC 系统（**H5**）。*

STPA 可用于识别违反 SC.2 的致因场景。然后可以将此信息细化为更详细的安全约束 SC2.1。

> ***SC2.1***：*系统设计不得干扰地面二次监视雷达，距离测量设备信道以及工作在 1030/1090 MHz 频段（2.5.1）的其他无线电业务。*

一个 TCAS 安全设计的依据的假设是永远不会发生这种干扰的。如果确实发生了，这则是系统设计或操作存在缺陷的领先指标。

人类往往倾向于随着时间的推移而改变其行为，并以不同于设计师最初预期的方式使用自动化程序。因此，关于操作员行为的假设为识别领先指标提供了另一个重要来源。例如，H3 是 TCAS 导致或促成飞行员失去对飞机的控制。安全约束 SC.6 由 STPA 从 H3 导出，内容如下:

> ***SC.6***：*TCAS 不得在飞行的关键阶段扰乱飞行员和空中交通管制作业，也不得扰乱飞机运行（**H3,2.2.3,2.19,2.24.2**）。*

除了识别产生这种安全约束的相关危险(在这种情况下为 H3)，这里的规范还指出了用于控制该危险的设计特征(在我的 TCAS 意图规范中标记为 2.2.3、2.19 和 2.24.2)，即执行 SC.6。这些控制也包含需要检查的重要假设。最基本的假设是，这些控制措施将有效地防止危险情况，并且能够正确实施。例如，在 STPA 分析中，确定的可能导致违反 SC.6 的场景之一是 TCAS 在飞行员在地面或起飞过程中提供分散了注意力的解决建议。人们设计了一个控制装置来防止这种情况，抑制飞行员

在起飞和着陆的关键阶段产生可能令人分心的决断咨询：

***SC6.1** 配备 TCAS 飞机的飞行员必须可以选择切换到仅交通决断模式（Traffic-AdvisoryMode-Only），仅显示交通咨询，但禁止显示决断咨询（**2.2.3**）。*

假设：此功能仅在起飞或平行跑道最后进近时使用，当两架飞机预计相互靠近时，TCAS 将要求进行规避机动。

增加控制，即飞行员切换到仅交通决断模式来抑制 TCAS 解决方案建议的能力时，便创造了另一种危险情景，必须通过飞行员程序，培训等加以控制，并导致了另一个假设：应在系统运行期间进行检查，以确保飞行员不违反 SC6.1 相关假设。

为消除或控制危险情景而创建的操作要求的其他示例有：

***OP4**：威胁解决后，飞行员应迅速顺利返回其先前指定的飞行路径。*

***OP9**：飞行员不得在仅交通决断模式的基础上操纵飞行*

由于这些程序是为了应对被确定为导致危险的特定情景而创建的，因此它们代表了一部分假设，应该检查这些假设以识别可能导致事故的危险行为。请注意，违反假设的相关信息可能会由 FOQA[27]系统自动收集。

另一个例子，在 Uberlingen 事故中，还有其他未提及的因果因素。其中之一是撞机时 ATC 设备正在维护，这使得空中交通管制员的听觉警报失效，而管制员并不知道这一故障。如果空中交通管制员知道警报被禁用，他可以及时调整反映。这种类型的因果因素可以在操作程序中进行控制，在这种情况下，这一程序是在 ATC 塔台正常运行的同时进行维护。当然，另一个重要的假设是遵守了该程序，并且这一假设仍需检查。

可用于确定领先指标的假设的最终来源是安全相关控制设计的限制。应记录这些限制，因为它们是决定是否以及如何部署系统的重要信息。一些 TCAS 限制与基本功能要求有关，例如：

***L2**：TCAS 目前不提示水平方向逃生机动，因此不会（也不打算）增加水平间隔。*

其他限制与环境假设有关，例如：

***L1.** TCAS 不对没有应答机或装载运行应答机（**EA3**）的飞机提供保护。*

***L6**：飞机性能限制限制了飞行机组人员响应决断咨询时可安全执行的逃生机动的程度。这些限制可能会妨碍冲突决断（**H3,2.38,2.39**）*

***L4**：TCAS 取决于存在威胁的飞机报告的高度准确性。威胁飞机的应答机（**EA5**）报告的飞机气压高度误差可能会降低间隔的安全裕度。*

假设：此限制适用于初始 TCAS 部署时存在的空域，其中许多飞机使用气

[27] FOQA 代表飞行运营质量保证，有时也称为飞行数据监控（FDM）。它指的是收集和监控飞机数据的程序，以提高飞行安全性和效率。其他行业有类似的监控程序，但名称不同。

压高度表而不是 GPS。随着越来越多的飞机安装了比气压高度表精度更高的 GPS 系统时，这种限制将会减少甚至消除。

与 L1 相关的一个示例假设是，没有应答机的飞机操作将在运行中被排除。

限制可能与设计中无法完全消除或控制的危险或危险因素有关：因此它们代表可接受的风险。

L3：*如果 TCAS 在冲突中启用或可以发布决断咨询，则不会发出咨询。*

这里隐含的假设是，除非在特殊情况下，飞行员将在起飞前打开 TCAS，这可以在性能审核中进行检查[28]。

最后，限制可能与系统设计期间遇到的问题或权衡有关。例如，TCAS 具有高级别的性能监控要求，使其在系统设计中包含自检功能，以确定 TCAS 是否正常运行。以下系统限制与此自测设施有关：

L9：*飞行中飞行员使用自测功能时将禁止 TCAS 操作，时间根据被检测目标的数量而定，最高不超过 20 秒。ATC 应答机在自检程序的部分时间不起作用。*

与安全相关的假设是，这种行为很少出现，因此不会导致 TCAS 经常处于非运作状态而增加 NMAC 的风险。

6. 使用假设创建基于假设的领先指标程序

上一节描述了生成基于假设的领先指标的方法。必须对系统设计中的假设实施控制，并在系统运行期间检查领先指标来维护这些假设。如何执行已确定的安全假设？一些新术语有助于区分不同的情况。

首先，在最初的系统设计中，维护假设的行为，称为*塑造行为（shaping actions）*，可以用来防止假设被违反。更具体地说，它们用来维持假设，防止危险，并控制假设向高风险的状态迁移。塑造行为提供了对突显特质的前馈控制[29]，并嵌入到分级控制结构中。比如，防止进入危险状态的物理互锁；使用干燥剂防止腐蚀；将人工操作设计的更加便捷且不易省略；建立特殊的管理结构，如独立的技术权威，以做出关键安全问题的决定。

*对冲(应急)行为（Hedging (contingency) actions）*用来为假设失败的可能性做准备，并相应地采取行动。在 STAMP 术语中，对冲行为使用反馈来控制突显系统属性。本质上，它们监控系统行为，并当系统控制所依据的假设发生的特定变化时做出反应。这种监测包括监测塑造行为的有效性。比如，故障安全设计(例如，保护和关闭系统)，该设计预计在某些情况下，塑造行为在消除假设被违反方面可能不成功。

[28] 这种限制背后的原因是由危害分析产生的情景引起的，但是超出了本手册的范围，。

[29] 未受过工程师培训的读者可能不熟悉的基本工程术语，详见附录 F 中的定义。

用于创建塑造和对冲行为的信息来自 STPA 致因场景和这些情景产生的假设。

不需要一直检查假设。一些假设只在系统或环境发生特定变化时才被违反。*标志杆（signposts）*是未来发展中的点，在这些点上，可能必须改变或应该改变当前的控制措施(塑造和对冲行动)。标志杆是触发假设检查的时间点，特定的未来事件或变化，包括对控制持续有效性的假设。管理变更政策通常包括各种类型的标志杆。举例来说，空域的变化，例如减少间隔或增加交通密度，可以被识别为 TCAS 的标志杆。

将塑造和对冲行为设计成安全控制结构，包括技术和组织两方面。可在系统设计和开发过程中识别标志杆，并创建和指定具体的响应行为。系统运行期间更一般的*假设检查（assumption checking）*包括特定检查，用来确定系统设计的基本假设是否仍然有效。在假设检查中，风险管理和控制人员在系统运行期间监控系统，以确定原始假设的有效性。这种监测可能包括监测对标志杆的反应，或者假设的变化和失效，人们已知这些假设没有通过塑造和对冲行为得到充分或完全的处理，或者根本没有通过塑造和对冲行为来执行。检查可包括性能审计、调查和自动收集的 FOQA 或其他数据。

执行基于假设的领先指标的方法

- 塑造行为防止违反假设
- 对冲行为防止假设失败
- 操作期间的假设检查
  - 使用标志杆进行特定检查
  - 在系统运行期间检查（定期或连续）
    - 性能审计
    - 调查
    - 自动收集数据

在前面章节有关组织分析的内容中，示例分析的结论之一是，技术独立是组织的理想目标。在建立这样一个独立的技术权威时，需要为新的控制设计塑造行为以及反馈渠道。此外，如果项目经理无视独立的权威，自己做出不明智的决定，可能会产生对冲行为。第三种可能性，也是最常见的一种可能性，即独立性在一开始运作良好，但随着时间的推移，它会慢慢被一系列小决定或环境变化所破坏。哥伦比亚号航天飞机失事后，美国宇航局独立技术管理局的总审计局记录下了第三种情况。为了尽早发现这些问题，避免产生未遂事件或更严重的事件，需记录并检查建立独立机构(在这种情况下)的假设，如果可行，需要持续记录和检查，如果不可行，则需要定期检查，以确定系统何时会由于独立决策结构的退化而陷入高风险状态。当然，事件和事故的调查是另一个可以识别失效假设的地方，尽管在遭受损失后进行

补救已为时过晚。

STPA 和其他方式确定的每个领先指标应如下归档：

- 相关假设
- 如何检查
- 检查时间
- 指标为真时应采取的行动（假设被违反）

该文档的实际结构将取决于组织及其风险管理系统。

7. 将领先指标融入风险管理项目

在确定领先指标的同时，还需要设计一个风险管理结构，当领先指标显示有必要采取行动时，该结构能够也将会采取行动。领先指标为风险管理提供了基础。整个风险管理过程包括确保不仅最初在新系统设计中处理风险，而且要能确定一个或多个假设和约束失败的主要指标，以便采取行动。

一般管理原则适用于设计使用领先指标的风险管理系统，但一些具体特征尤为重要。即使系统中高风险的信息十分清楚并且可得，许多组织在处理这些信号、做出反应仍然很慢。如果领先指标出现了却未采取适当的行动，那么创建和监控这些指标毫无用处。启发式偏差干预的次数太多，尤其是防御性回避，明确的领先指标就会被人们忽略，以至于事态不断发展导致严重的负面后果[30]。除了心理偏见之外，组织文化和政治也会在设计和应对领先指标方面造成问题。

成功创建和使用领先指标需要采取方法控制这些心理和文化偏见。例如，为了鼓励有效的行动，必须将认真仔细地领先指标纳入总体风险管理计划:不仅须将这些指标传达给相应的决策者，还应该为关键场景制定详细的行动计划，并为实施这些行动计划指定触发点。分配责任来检查领先指标的存在，并在发现问题时跟进。

如前一节所述，每一个领先指标都应该指出何时以及如何检查，并且必须配有与之相关的行动。在发现假设无效之前，应该制定必要的行动计划，以减少否认行为和回避行为，克服组织和文化上的盲点。监测和行动的责任可能需要分配给一个独立的组织，而不是分配给项目经理以及那些承受冲突压力的人。需要定期重新评审领先指标清单，必要时进行更新。应建立一个持续的改进过程，根据经验，随着时间的推移，重新评估当前的指标，并诊断出任何发现的有效性不足。例如，如果出现了领先指标没有显示的负面后果，就应该分析为什么领先指标没有及时发现问

[30] 心理学家们已经写了大量关于人类在评估危险时表现出的偏见的文章。这些偏差会影响我们设计领先指标的方式，以及我们如何应对这些偏差的发生。尽管所有这些偏差都很重要，但就对领先指标的反应而言，最相关的偏差之一被称为“防御性回避”这种偏见可能反映在对领先指标的感知准确性的降低上，或者反映在人们认真对待这些指标并接受危险可能会增加的能力上。防御性回避是基于一种共同的心理倾向，即避免考虑一个压力大或与其他紧迫目标冲突的话题。

题以防止事件发生，或者如果发现了问题，为什么没有采取有效行动。然后，应该利用这些信息来评估和改进整个领先指标计划。是否已确定领先指标，但未检查？还是从未被发现？这些问题的答案不仅应该用来改进领先指标本身，而且应该确定遗漏或检查不足的原因。这一原因可能涉及流程中的一个缺陷，并产生识别其他未被检查的重要编程假设。

8. 可行性和最后思考

假设所有这些看起来都不错，仍然存在创建和检查领先指标的可行性问题。虽然 STPA 过程中似乎有大量潜在的领先指标，但并非所有这些都需要检查或经常检查。对冲行为和应急措施通常内置于安全控制结构中，以控制大多数已确定的因果因素。许多假设可以通过适当的设计来实施，这应该是处理危险分析结果的最高优先级，并且在早期概念和设计阶段进行危险分析时最可行，如本章将 STPA 集成到系统工程所述。

标志杆将把检查限制在某些触发事件的发生上。关键假设的文档应该是任何系统开发和规范活动的一部分。记录假设不仅对识别领先指标和提高安全性很重要，而且对系统的正常维护和发展也很重要。即使在最初的开发过程中，人们也会忘记为什么他们会做出某些决定，除非将其记录在案。在我们使用这种方法来引导关于实际问题的指标的情况下，例如 TCAS，我们发现，为了减少开发和测试时间以及维护系统并保证过程中没有危险，有关键假设的记录十分重要。请注意，我在几个月的“业余时间”中创建了 TCAS 意图规范，其中包括关键安全（和其他）假设。

还会有很多假设吗？当然可能有。处理大量安全因果因素的通常方法是使用概率来减少它们。问题是没有完全准确的方式可以预测未来事件的发生概率。与数字相比，人类(工程和管理)专业知识和知识的使用往往被低估，尤其是计算机产生大量数字的时候。但是，虽然数字看起来更准确，但它们通常只是基于很少的真实数据的胡乱猜测。心理学家发现，事实上，人类很难估计未来事件的概率[31]。当然，你可以随意指定领先指标的概率或可能性，但我们发现基于工程、管理、心理和社会学知识的风险级别产生了更佳的结果。另一个考虑是，对非数字风险评估的独立评审比数字风险评估可能更加“深入”。人们倾向于相信数字是有根据的，且不太可能挑战它们，除非数字很离谱(有时也许不太离谱)。

[31] 一个典型是特维斯基和卡尼曼关于启发式偏差的研究。

第 7 章：安全管理系统

南希·莱文森

继第六章描述了如何使用 STAMP 和 STPA 分析社会技术系统，本章将着重描述安全管理系统（SMS）这一特定社会技术系统的设计和分析。正如所有创建和运行安全攸关系统的组织都有安全文化，此类组织皆设有 SMS，尽管 SMS 和安全文化对实现社会安全目标和组织安全目标的作用大小不一（甚至有时不利于实现安全目标）。

本章讲述如何使用 STAMP 和 STPA 创建新的 SMS 或分析改进现有 SMS。然而，本章只是间接提及 STPA，基于 STPA 的安全设计项目只有在保障管理结构中使用才有效。本章的目标是协助设计此类结构。

1. 什么是 SMS?

SMS 应以能够积极主动地控制（管理）组织每一方面的安全为目标。对于生产组织，SMS 管理产品在研发过程和产品自身的安全。对于服务行业，在工作场所和提供的服务中必须对安全进行设计和管理。因为社会技术系统在初期不会十分完美，并且随着时间的发展，世界也在不断的变化，所以必须留有有效学习的过程，从经验中不断学习，提高工作场所、产品、服务和管理系统自身的安全性。

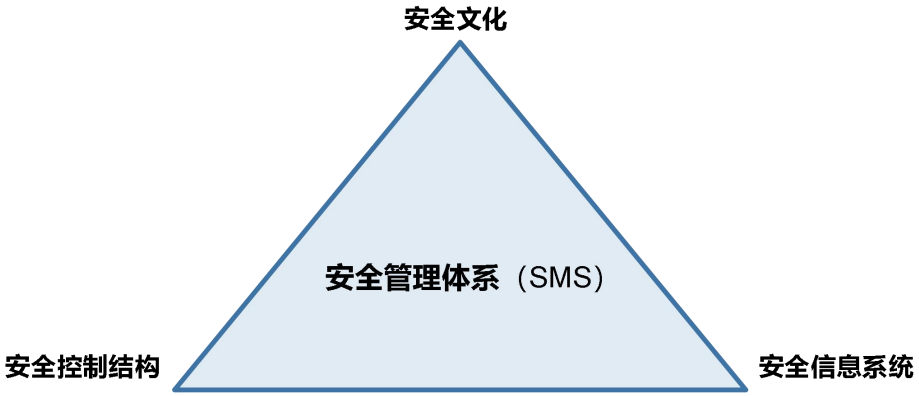
不存在一种有效保障安全管理系统的单一设计。设计目标取决于组织类型(产品研发或服务提供方)、组织本身的社会性文化、组织的结构和文化、产品或服务固有的安全性、环境因素以及组织预期达成的其他目标以及法律环境。包括 FAA 在内的一些组织会指定一种风险管理的方法，但是这种风险管理方法并不适用于所有组织，并且风险管理系统只是安全管理系统的子系统。除此，特定的安全管理系统规则所强调的安全价值可能并不完全适用于某一组织或组织，无法达到其较高的安全等级。

然而，成功的 SMS 是有共通特征的。本章描述了系统设计 SMS 的思维方法，是作者数十年在安全工程领域、阅读和撰写事故报告、识别涉事因素以及帮助各组织预防事故发生的经验汇集而成。一些行业和组织事故频发，而一些行业或组织却鲜有事故，还有一些根本不会发生事故（例如 SUBSAFE，美国海军核潜艇安全项目[32]）。一个组织乃至一个行业属于这三类中的哪一类，某些因素至关重要。许多最有效的安全控制方法并不适用于所有组织或行业。设计 SMS 是一项系统设计工程的问题：

[32] SUBSAFE 关注两个安全目标：（1）潜艇外壳的水密完整性；（2）用于控制洪水紧急情况并恢复的关键系统的可操作性和的完整性

目标应该是设计和运行在组织中有效且切实可行的 SMS。

组织安全管理中最重要的因素包括：文化、安全控制结构（管理结构）和安全信息系统，但是这三个因素并不相互独立。系统方法主张文化、安全控制结构（管理结构）和安全信息系统必须连贯一致才能发挥最大效用。文化决定可取的和可接受的行为与如何做出决策。管理或控制结构决定如何在组织中的践行文化。最后，安全信息系统提供使管理结构成功实现预期安全文化的必要信息，即使最好的意图没有合适的信息匹配也是难以执行。



2. 安全文化

安全文化的定义多种多样。我个人喜欢埃德加·施恩对安全文化的定义：

安全文化是做出安全相关决策的价值观和假设

图 6.1 是施恩的三层组织文化。最底层是文化的本质，价值观和深层的文化假设，构成了创建组织规则、政策和实践的基石。规则、政策和实践反过来描述了表层文化产出物（例如，危险分析、事故调查报告等）

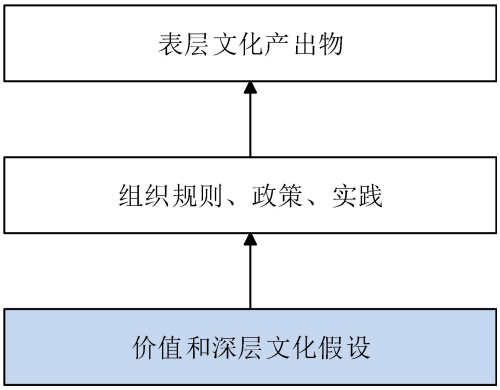


图 6.1：施恩的三层组织文化

中间层和顶层不属于文化范畴；仅仅反应组织文化。尽管试图仅改变上两层可能短暂地改变行为和甚至降低短期风险，但无法影响底层共享价值和社会规范的这两个层面上的表面修复有可能会被撤销，且随时间的推移而失效。

虽然人们有时探讨“创建”安全文化，但是通常情况下，是存在着某种安全文化的，尽管现有的安全文化并不可以十分有效的提高安全，并不存在无文化的组织、行业或者社会。

通常，历史和环境因素是创建现有安全文化的关键。例如，威廉姆·波音在构建商业航空领域时面临的现状是，为了出售飞机和推广飞机出行，需要提高领域的安全性。在 1955 年，只有 20%的美国公民愿意选择飞机作为其出行方式——当时空难与现今相比较为普遍。与此相反，1957 年普莱斯·安德森法案通过，提出发展核能领域，同意政府加强监管以承担事故中的有限责任。在一些领域，安全文化是强有力的个人领导力推行的结果，诸如核动力海军之父海军上将海曼·里科弗。结果就是，各领域管理安全的方法各异。

然而评定安全文化的“好”“坏”是很难的。有一些行业相比其他行业会发生更多的事故或较大损失。事故率相对较高的组织的安全文化更易表现出以下一个或多个特征：

- 接受风险的文化：接受风险的文化以事故的无法避免这一假设作为基础的，只有告诫所有人小心行事才可以预防事故发生，事故被视为生产力的代价。同时，所有人应为自己和他人的安全负责，事故是由个人缺乏责任感的行为导致的。普及的观点是：只有所有人安全且负责任的行事，便会减少事故的发生。
- 否认的文化：在否认文化中，风险评估通常不切实际的低，未对切实的风险和警告进行适当调查便将其解除：管理层只想听到好消息，因此他们也只会收到好的消息。否认文化的关注重点在于系统是在可接受的安全范围内，而不是识别不安全的情况。
- 合规文化：合规文化的重点在于遵守政府部门的法规。基本原则在于遵守规章达成可接受的目标。由于制定规章的部门倾向于将重点放在为安全的产品或服务颁发合格证上，因此强调事后保证，大量的“安全案例”论据对实际产品或过程安全的影响很小或没有影响。
- 流于形式的文化：流于形式文化依据原则是：生成书面文件和分析性的文案工作从而获得产品和服务的安全。虽然生成了大量文件，但是文件对设计和运行的实际影响很小。大部分安全相关的文案工作是由一个独立的团队完成，与设计、运行产品，过程实施和服务提供的人员没有交流。
- 追求风险的文化：懦弱的人才求安全，真正的勇士视风险为机会

一个好的安全文化的特点是较难界定的。但是通常情况下，好的文化包括以下特点：对安全和安全目标的开放性、愿意收到负面消息、重视采取必要措施提高安全而不是一味遵守政府法规或生成大量书面文件以及相信安全对于实现组织目标的重要性。在有效的安全文化中，员工相信领导层愿意倾听自己安全方面的担忧，

并会采取一定措施予以解决，领导者也认为员工的意见值得倾听，值得尊重，员工可以放心地报告其忧虑，且自己的意见会得到珍视。安全是所有人共同的责任，员工不仅要考虑问题也是解决问题的一环。与此同时，这不仅是员工保证自己和他人安全的责任。

组织的安全文化是如何建立和改变的？所有组织的安全文化（用于决策的价值观）是由高层管理者设定的。管理层对安全的切实承诺是实现组织安全的最重要因素。在员工表达出合理的担忧时和将安全放在其他诸如进度和费用前时给予支持。

管理者在日常与员工交流时对安全问题的开放态度深刻地影响着对问题的接受。有一则经典的故事就说明这一问题。保罗奥尼尔在 1989 年受美铝外聘，任 CEO 时宣布首要目标是将美铝公司打造成美国最安全的公司，争取零伤亡。这一目标一经宣布，大多数人的反应都是美铝的董事会让“一个疯疯癫癫的嬉皮士管理公司”，他会毁掉公司的。而事实是，在不到一年期间，美铝利润达到历史新高，直到 2000 年奥尼尔退休，美铝一直坚持这一目标。在快速发展的同时，成为了世界上最安全的公司之一。奥尼尔深知安全和生产力并不互相矛盾，实际上更是相互助益的。

在设计任何系统时，设定目标以及确定为实现目标所需达成的要求是成功的第一步：未知方向，寸步难行。目标对指引成功的设计过程来说必不可缺。管理层确立价值观，并在价值观之上做决定。因此，管理层设计或提高安全文化的第一步便是建立和沟通预期的安全决策和行为。

管理层一旦决定了可取的价值观，下一步便是领导间通过精悍成文的安全理念和用以执行理念的、详尽的安全政策就基本价值观相互交流。高层管理者的职责在于确保管理层和员工全面接受成文的理念和政策。

深化安全文化的第一步是领导者就其想要建立的安全文化进行讨论。大多数公司都有大量的、详述如何进行安全建设的安全政策文件。安全政策文件故而重要，借助更为简短的哲学原理和价值观阐述组织预期的安全原理也较为有效。这种哲理性的描述能够界定安全与组织其他目标的关系。

一些普及性的原理适用于所有组织，另一些是否适用取决于组织是否设立、运行安全攸关系统，或提供安全相关的服务。通用原理可能也是安全哲理声明的一部分：

1. 所有损伤和事故都是可以预防的
2. 质量和安全的提高会降低成本，减缓进度，而长远来看可以提高利益。从防止事故发生的角度来说是合算的。
3. 安全和生产率手拉手。增强安全管理有利于提高其它质量和性能因素。要达成最高的经营绩效需要安全保驾护航。
4. 安全需要融入到产品或服务的设计中。后期加入会降低有效性而且代价更高。事后担保无法保证设计的安全性，因为设计中并未呈现出安全性。设

计中考虑到安全性比事后保证更有效。

5. 事故/事故征候伤亡分析的目标是确定 损失（或几近损失）的原因，以便做出适当的修正而不是责备相关负责人。
6. 事故和事故征候是观测不安全运行系统的重要窗口触发全面的原因分析并采取改善措施。
7. 毫无畏惧地面对安全信息。应大胆的进行安全分析。
8. 组织重视安全承诺，开放性和诚实。
9. 有效沟通和信息分享对于防止损失十分重要。
10. 根据所有员工的表现和帮助我们推进安全做出的贡献来评估。

第 4 条的附加解释如下：第 4 条与当下各组织乃至行业处理安全问题的方式相悖。将重点放在事后衡量或评估风险比自一开始便将安全性融入产品及工作场所更为常见。设计出的产品和服务或是安全的或是不安全的，都是争论和评估无法改变的事实。除此之外。事后评估或验证不但比一开始便进行安全设计更为昂贵，而且有效性也会降低，设计完成后进行提高是非常有限的。

事后评估只能说明存在安全性或者是有证据显示安全性不存在。因此，强调事后评估，诸如，只强调积极因素或生成正面数据的因素（称为确认偏误）、只考虑可测量的因素及只提供一份数据但是忽略其他因素（例如管理和设计缺陷）、编造数据或为了实现数据目标篡改数据等等，都会无意识的造成数据不准确。这并不意味着测量和评估不重要，但是至多只能确保创建安全系统的措施是成功的，如果系统已经是不安全的，那么这不能使系统安全。

事后评估的替代性方法是带着证明设计不成功的心态进行评估，而不是争辩系统是安全的。这一方法可能会更为有效，然而，一旦评估开始，修复可能发现的问题也是十分昂贵的，且比一开始便进行安全性设计的花费更多，效果更差，设计之初进行安全分析，及设计过程中进行安全分析效果是最好（见本手册关于系统设计过程添加 STPA 相关章节）。此替代性方法不是证明安全性，而是强调识别和消除或控制危险源。强调事后评估是不太可能达到较高程度的安全性的。

此外，强调评估中的可能性会产生过度自信和不切实际的评估，并忽略非随机或可能性无法获得的重要因素。在十分成功的 SUBSAFE 计划中，用于认证安全系统的所有证据必须是客观质量证据（OQE）。OQE 的定义是：“OQE 是基于可验证的观察、测量或测试的、关于产品或服务质量的定量或定性的事实陈述。”概率风险评估（甚至包括识别事故发生危险源可能性的定性风险评估）是对后期发展进行预测，且只有多年后才能验证预测是否有效。有效的预测几乎不存在。评估可能性只有在过去与未来相同或相近时才有效，即借助过去的经验做出推断。但是创建新产品是为了改进过去的产品，而不是进行复制，并且是系统（及其环境）也会随时间变化。即使产品本身没有变化或降级，操作或使用系统的人也会开始改变他们的

行为，人们对系统的使用在改变，环境也在发生变化。

关于安全理念和更为详细的政策声明的书面声明是一个起点，但它们还不够。员工应快速识别书面政策何时与管理者的实际行为有出入。要取得成功，高层管理者应投入其中，而不仅仅是通过口号和动议。

承诺如何展示？确定优先事项并贯彻执行：通过个人参与（例如，安全决策的高级管理层领导小组）；建立适当的组织结构；任命指定的高层领导者承担安全相关的责任，并提供足够的资源以使其有效;指定最优秀的员工参与安全相关的活动并对员工的努力进行奖励；回应他人动议。沟通过程中将责罚最小化。领导者需表明最为优先的事项是确定产生损失的系统性因素，而不仅仅是找到责任人（通常是组织中最低级别的人）之后不了了之。最后，领导者需要在组织中设计激励结构，鼓励所需行为。

本节主要思想总结如下：

管理层如何提高安全文化的建议

- 设定决策指定中的目标和价值; 建立并沟通与安全相关的决策和行为的预期。
- 支持合理关注工作安全的员工
- 制定短小的书面安全理念和更详细的安全政策
- 确保安全理念和政策得到领导者和员工的广泛认可
- 遵循决策中的安全理念，并期待普及到每个人
- 强调产品或服务设计之初的安全性，而不是保证或事后评估
- 进行评估的目的是提供设计不安全的证据，而不提供安全的论据
- 需要客观的质量证据进行认证或保证
- 表明对安全的承诺
  - 个人参与
  - 确定优先事项并贯彻执行
  - 建立适当的组织结构
  - 任命高级，受人尊敬的领导者担任安全相关的角色和责任
  - 为安全工作提供足够的资源以使其有效
  - 分配最优秀的员工参与安全相关的活动，而不是那些不重要的或无关紧要的员工
  - 奖励员工安全工作
  - 回应他人的动议
- 责罚最小化; 专注于“为什么”不是“谁”
- 设计激励结构以鼓励可取的安全行为

3. 安全控制结构

如前所述，设计安全控制结构只是系统工程的另一种形式。首先是对工作目的的陈述。SMS 是：

是什么？：一种控制结构，有助于创建和维护组织的安全性

为什么？：确保消除危险源，或者如果不可能，控制（减轻）危险源并推动有效安全文化

如何做？：通过建立管理控制和责任（RAA）来管理危险源和全面可用的安全信息系统。

总体目标是设计一种消除或减少损失的控制结构。实现这一目标需要在控制结构的各个层面明确安全任务的预期、职责、权限和责任。此外，为进行有效运作，结构需得到适当的反馈和各部分之间的协调，结构还应包括用于显示由于内部或外部变化导致控制无效的领先指标。整个控制结构必须通过物理设计、定义流程及程序和社交互动及文化来强制对系统行为施加安全约束。

用于开发的安全控制结构和用于操作的安全控制结构之间可能存在显着差异。对于受监管的行业，需将政府监管组织包含在安全控制结构模型中。法院、保险公司、用户群等其他外部团体可能也需包含其中，等等。

a) 通用的安全控制结构设计考虑因素

设计评估安全控制结构时需考虑如何分配安全责任、组织中安全相关活动的适当位置、信息交流和活动协调、变更管理和控制、设计和反馈鼓励，并确定风险管理程序的设计和作用。此外，还须考虑所需的教育和培训工作，以及保证控制结构学习和持续改进的方法。更多内容，请参阅附录 D“设计和评估 SMS 的指南”。

责任分配：

领导力是实现较高安全水平的关键。这意味着组织安全职能的领导不应仅是为培训未来领导者和管理者的轮岗任务。损失较少的组织所任命的安全职能部门的领导者对自身在安全及防止损失方面有极大工作热情。在安全管理组织中为致力于损失预防的人提供了明确的职业路线。

与任何有效的管理系统一样，必须分配职责、权限和责任。“每个人都对自己和他人的安全以及产品的安全负责”的原则会导致过多的事故。人人都对安全负责等同于没人对安全负责。

安全责任存在于组织结构的各个层面，尽管各层面的职责都有所不同。在事故频发的组织组织中，人们普遍认为安全责任应在控制结构低端。因为他们认为在底端系统各部分实际运行的信息更多。这一观点的问题在于低层还缺少视野：尽管具有关于系统特定部分的详细信息，却没有系统其他部分的信息，因此无法预测或防

止系统内部各部分不安全交互。低层关注期也短，而较高的组织层对系统目标关注点更为广泛而不局限于子组件的目标。较高的组织层需要控制低层组件之间的交互，并确保低层执行安全约束。

每个层级必须通过使用适当的程序、正确地执行程序及其有效运行来监督低层。通常还需要一个管理协调中心（或多个）保证安全管理系统在管理系统的每个层级的设计和正常运行。

责任分配的建议
– 任命对安全工作充满热情的领导者
– 为致力于损失预防的人提供职业发展道路
– 在安全控制结构的各个层面提出明确的预期、职责、权限和责任
– 不要让所有人都对安全负责
– 在整个控制结构的各个层面分配安全责任
– 指派某人负责保证 SMS 的设计和正常运行

组织中的位置：

在决定在组织何处设置系统安全活动以及如何设计和管理反馈和控制变换时，需要考虑的基本原则如下：

第一，成立企业级职责的高级别小组，企业级职责包括确保所需活动的有效进行。该小组还提供领导和协调。虽然安全活动将渗透到大型组织发展和运作的各个部分，但是普遍性的方式方法也会增强个人活动。该小组的领导者需要具备与高层管理直接沟通的能力，并为所有类型的管理决策提供输入。这意味着担任此职位的人必须向有影响力的人报告并受到高级管理层的支持。

在高层安全管理之下，各个层级可能也会有活动，层级越高，职责越广泛，各层级接连的较低层次都有与其运行层级相当的、关注点更集中的职责，例如，处于开发组织的业务单位，计划和项目级别。常见的错误将安全视为对生产线运行几乎没有影响员工功能。

不存在针对安全控制结构的一个甚至多个正确的设计。有效的设计将取决于行业、组织管理风格等。本手册的附录中列出了应包含在开发和运营安全控制结构中的一些职责。

第二，正如上文所述，将系统安全工程与系统工程分离是产品开发组织的一个错误; 此举可能导致实现安全目标的成本增加，原因在于必须在开发早期解决安全问题。安全工程不应该集中在事后保证上。虽然服务组织的担忧各有不同，但所有做出安全决策的部门或团体中应配备安全工程岗位。与此同时，仍有必要在主要决策小组之外设立独立的监督和决策部门。对于所有类型的组织，安全性需要设计到

工作场所中，并且需就组织中此职责的适当位置做出决策。在组织结构中分配安全责任的一般原则是：

- 决策者需要与能够提供安全信息的人直接联系。如果关键信息是一系列命令，就容易因为进度或预算压力而出现故意或无意丢失或修改的现象。直接沟通渠道提供及时接受信息的更多机会，且不会因有潜在利益冲突的小组的影响而被过滤。许多决策者可能还需要快速访问信息。
- 需要与组织大部分部门建立直接沟通渠道。安全问题几乎渗透到所有组织活动中
- 安全必须对决策产生影响，这意味着决策者在需要做出与安全相关的决策时获得必要的 安全信息。

SUBSAFE 美国核潜艇安全计划采用了一种满足以上全部要求的独特设计。他们将自己的结构描述为权力分割或“三脚凳”（图 6.2）。管理人员只能从独立技术管理部门（ITA）推导出的一组可接受的选项（安全性）中进行选择。技术权威是建立并确保遵守技术标准和政策的过程。ITA 提供一系列包括风险[33]和价值评估在内的技术上可接受的替代性方案。职责（和责任）包括：

- 制定和执行技术标准
- 保持主题专业知识
- 确保安全可靠的运行
- 确保有效和高效的系统工程
- 指定无偏见的、独立的技术决策
- 提供技术和工程能力的职位及职责管理.

“三角凳”的第三条凳腿是合规性验证组织。它与计划管理者和 ITA 在权限上是平等的。

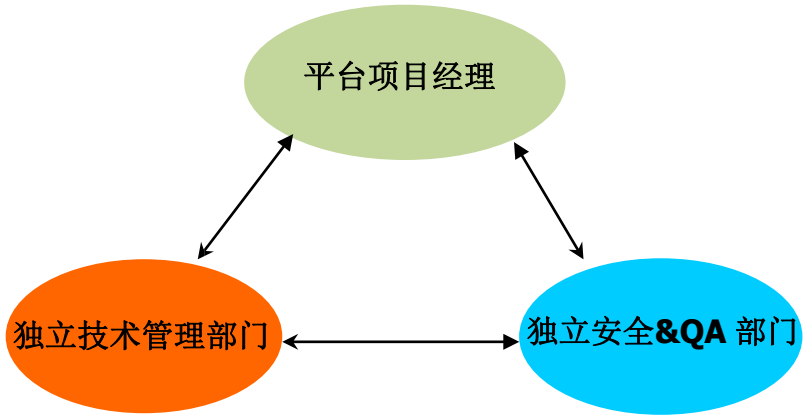


图 6.2. SUBSAFE 三角凳理念

[33] 风险可以定量或定性指定，但如本章其他部分所述，SUBSAFE 必须使用客观质量证据，因此不允许进行无法测试或验证的概率风险评估或可能性估计。

在控制结构何处安排活动的建议

- 创建一个具有企业级职责的高级别小组，例如确保所需的活动正在进行并且有效并提供领导和协调。
  - 确保每个人都使用通用的方式方法
  - 他们必须与直接最高管理层直接交流，并投入管理决策入
  - 他们必须向有影响力的人报告，并有高级管理层的支持
- 职责分配。安全不仅仅是员工职能，而且对生产运营有直接作用和影响
- 不要在产品开发组织中分离系统安全和系统工程。将安全功能融合到工程功能中。
- 将安全工程岗位涵盖在制定安全决策的所有部门或小组中。
- 除集成功能外，还提供独立监督
- 确保决策者与提供安全分析和信息的人员建立直接联系。
- 在组织中对需要进行安全活动和安全决策的任何部分之间建立直接的沟通渠道。
- 考虑实施 SUBSAFE 权力结构分离。至少确保技术决策独立于程序化决策。

沟通与协调：

向需要其执行安全管理职责的人员反馈和传播信息是安全控制结构设计中需要考虑的重要因素。这不单是收集许多（有时是大量）的信息，其实收集过多的信息可能会降低信息系统的性能，因为过多的信息不利于个人识别出有效做出决策的内容。此外，改进安全决策所需的信息必须可随时取用。它在减少损失方面非常重要，因此本章稍后将单独讨论安全信息系统。

沟通在协调活动和对事件的反应也很重要。有职责重叠的人需要沟通渠道和方法来协调他们的活动，以确保安全约束的执行。例如，一个子系统中由安全性引起的变化可能会影响另一个子系统和整个系统。安全活动最终不得以散碎和不协调而告终。交互不仅发生在层次结构组件之间，而且还必须发生在同一级别的不同部分或系统类型之间，例如在开发和制造之间或开发和运行之间。图 6.3 是系统开发人员和系统操作人员之间所沟通的各类安全信息。

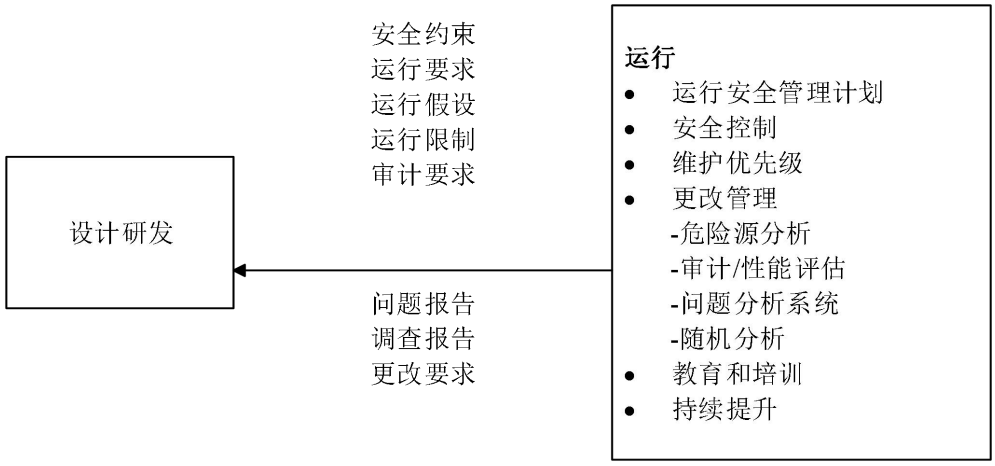


图 6.3.为支持研发和运行之间安全活动的必要信息渠道

对安全控制结构的所有组件之间需要沟通的信息的相近描述进行识别和记录。

国防部设计的一种非常有效的沟通和协调手段是*工作组*。国防部项目可以持续多年，庞大且复杂，并且通常涉及大量不同地域和不同组织的参与者。协调和沟通可能成为此类项目的主要问题。分层结构可以解决部分问题，其中项目的高层管理可能位于国防部，但由一总承包商负责系统工程和分包商之间的协调工作。沟通对于此结构至关重要。工作组在协调和沟通上已取得一定成效，此方法可以适用于不太复杂的项目。

工作组是安全控制结构中两个层级之间或在同一级别的两个或多个组件之间的交流纽带。工作组的成员负责协调工作，报告未解决的安全问题，并分享独立安全工作的信息。控制结构的各个层级都可能有工作组，成员和职责取决于运行级别。公司工作组可由不同部门或项目的安全管理人员组成，低级别的工作组可从产品各部分的设计小组中选择代表组成。当出现需要全新解决方案的特殊安全问题时，可创建诸如软件安全工作组的安全组分享经验和方法。政府组织可为由代理部门和主承包商的代表组成的大型项目建立安全工作组。主承包商可以为所有分包商创建一个由安全管理人员组成的安全工作组。

设计交流和协调的建议

- 确保决策者可以获得安全决策所需的信息。
- 提供沟通渠道和协调安全职责有重叠人员之间活动的方式
- 确保安全活动完整且协调。
- 确定所需的交互，并确保各层级必要信息流的明确使用
- 在安全控制结构的所有组件中识别并记录必要的安全相关沟通通道。
- 确保存在反馈和协调渠道并且有效
- 考虑建立安全工作组

管理和控制变更

大多数事故发生在某种变化之后[34]。这一事实并不令人惊讶，因为在行为或环境都没有发生变化的情况下，事情的结果理论上应该是相同的。适应和变化是所有系统固有的特征，对组织蓬勃发展来说必不可少。问题也就不在于改变，而是不安全的改变。因此，SMS 必须具备精心设计的控制措施，以防止不安全的变化，以及除阻止不安全变化的努力，还检测是否发生了不安全变化。目标是在不违反安全限制的前提下允许变更。需考虑的两种变更是：计划变更和计划外变更。

计划变更：SMS 必须在计划变更时持续有效，计划变更包括组织变更、人员行为和流程、运营（包括维护阶段变更和运营阶段变更）或环境变更。大多数公司都有（如果没有，他们应该设置）变更管理（MOC）政策，要求评估所有计划变更的安全影响。评估成本取决于变更的范围、文件的质量以及最初危险源分析的执行方式。STPA 将已识别的危险源的完整追溯性与其致因场景及危险源预防的设计特征相结合。如果没有可追溯性，变更后的无法分析成本，因此会略过计划变更。

事实上，在实践中经常忽略 MOC 程序，致使相应的损失永远不会发生。安全控制结构应指定 MOC 程序和反馈渠道的职责，确定是否遵循，如果不是，为什么。如果忽略的原因在于 MOC 变更程序本身（过于繁复、难以实行、太过耗时等），则可能需要更改 MOC 程序。

实施 MOC 程序的成本取决于系统文件的质量以及最初危险源分析的执行方式。STPA 的设计目标是将评估变更的成本最小化。

计划外的变更：计划外变更带来的挑战更加困难。需要有（1）识别潜在不安全变化的方法和（2）应对这些变化的方法。如前一章所述，应从根据系统（包括 SMS）设计和安全分析过程中的假设创建领先指标。确定领先指标需要在组织结构、SMS 以及产品和工作场所的设计和开发过程中记录假设。STPA 控制结构、UCA 和致因场景为创建有效领先指标提供所需信息。如果无法消除致因场景，则可以使用 STPA 分析的信息创建塑造和对冲措施，以及识别标示和假设以检查审计和性能评估。

风险再评估是发生不安全变化的常见原因。一段时间没有损失时，人们会开始再次评估风险是否下降。没有任何变化的情况下风险是不会真正下降的，但人们可能相信风险已经降低。受这方面压力影响，人们开始违反自己的规则，并通过争论来证明安全没有受到影响。安全控制结构的过程需要在安全裕度被削弱之前中断风险再评估。但是要求沟通实际风险水平的适当信息。管理层需要了解所控制的流程中的风险状况。这需要提供适当的所设计的安全控制的状态反馈。

除此之外，在行为与真实的风险水平相悖时，需要对负责人发出警报。防止这

[34] 大多数事故发生在某种变化之后。这句话的注释是：参阅见 Nancy G. Leveson 于 Addison-Wesley 出版社出版的“Software”。

种现象的关键是允许实现安全目标的方法有一定的灵活性，并提供便于决策者进行准确风险评估的信息。

对于有效的 SMS，其特征在于其持续的警惕性，系统不会随时间退化。以 SUBSAFE 项目为例，该项目为解决以下三大难题付出很多的精力：

1. 无知：不知道;
2. 傲慢：骄傲、自我为中心、自负、或自觉具有智力优势和未遵照实施自行推理
3. 自满：对自己的成就感到满意，并缺乏对实际危险或缺陷的认识。

SUBSAFE 计划的大部分设计和重点旨在解决这三个挑战，SMS 的设计都应包括面对这三大挑战提供警惕性的措施。

管理和控制变更的建议

- 设计控制和 MOC，以防止不安全变化并检测变化是否发生
- 评估所有计划的变更，包括临时变更，以确定它们对安全的潜在影响
- 分配职责，确保 MOC 程序得到执行且遵守。如若无法实现，请找出原因并解决问题。
- 创建最大限度地降低执行 MOC 程序的成本的文件和程序
- 创建识别不安全的计划外变更和应对更改的方法
- 设计基于假设的领先指标并创建风险管理计划，以便在发现潜在的不安全变更时有效监控和响应。
- 在组织结构、SMS、产品和工作场所的设计和开发过程中记录假设。
- 创建塑造和对冲措施以及领先指标的检查程序，包括审计、性能检查以及标示
- 实施领先指标，以指示控制措施何时无效
- 确保决策者掌握当前风险水平和所设计安全控制状态的信息
- 分配职责解决反馈中提到的行为与真实的风险水平不相符的情况
- 随着时间的推移保持警惕，防止安全控制结构和安全文化的退化以及任何自满情绪

<u>设计和鼓励反馈</u>

准确的风险评估需要信息流和合适的反馈渠道。文化问题是反馈问题的常见原因。反馈渠道有三种：审计和性能评估，事故/事故征候致因分析和报告系统。

*审计和性能评估：*审计和性能评估类型多样，但评估安全状态的组织是从安全限制和对安全控制设计所作假设开始的。审计不仅应涉及产品和流程，还应包括安

全管理系统本身以及旨在确保防止损失的控制措施的有效性。至少安全审核和性能评估的某些部分应该集中在安全控制结构是否按照其设计和理应运行的方式运行上。

不仅审核低层，整个安全控制结构都应进行审核。在 SUBSAFE 项目中，即使是顶级海军上将也需要接受 SUBSAFE 审核。这不仅确保了计划按预期进行，而且还为审计提供了正面的文化信息，因为所有员工都看到即使是领导者也应遵循 SUBSAFE 的规则，并且高层管理人员愿意与参与 SUBSAFE 的任何其他成员一样接受并解决审核结果。SUBSAFE 安全控制结构中低层的人员参与其上级人员的绩效评估，明确的表现出整个项目对安全和防止损失的重要性的重视。接受审计并进行改进，以身作则，领导者强有力的传达了他们对安全和增进安全的承诺。

参与式和非惩罚性审核可能会非常有效。这种审计的目标是将这一过程视为一种建设性的学习经历，而不是一种批判的过程，是提高安全性的机会，而不是评估员工的方式。与外部审计公司只观察审计程序不同，建立由未直接接受审计的组织的其他部门的专家组成的审计小组。各种利益相关方可能在审计中发挥作用，甚至被审计小组的个人也可能在审计对象中。目标应该是树立一种态度：这是一个改善我们实践的机会，为包括审计员在内的所有人提供学习机会的活动。

审计本身应包括与接受审计人员的持续沟通，以便充分了解任何已发现的问题和可能的解决方案。与外部审计规则不同，参与式审计允许即时的反馈并讨论解决方案，以加强人们的理解，提高安全性，而不是惩罚或评估相关人员。由于知识储备丰富的团队就在现场而不是在审计完成后几个月才提交书面报告，审计也是解决问题的机会。

安全审核和性能评估的重要目标是衡量实际存在的安全知识和培训水平，而不是管理者认为存在的或培训计划和用户手册中存在的内容。审计可以就培训和教育活动潜在改进之处提出重要反馈。为了与非惩罚性审计理念保持一致，不应消极的将知识评估视为惩罚。相反，应是改进培训和教育工作。

事故和事故征候调查：事故和事故征候调查明确表明 SMS 未按设计或预期工作的证据。调查程序必须嵌入允许使用结果的组织结构中。必须确定涉及的所有系统因素，而不仅仅是征兆或技术因素。分配责任不应该是目标；相反，目标应该是找出安全控制结构无法有效防止损失或接近损失的原因。这就意味着必须调查整个安全控制结构，以确定各组件对不良事件的潜在影响。

管理者不应负责调查其指挥链中发生的事故：调查员和原因分析师必须在管理上和财务上独立于所涉及直接管理结构中的人员。因此应考虑使用预算独立和高层独立管理的训练有素的团队。

调查不单单是需要撰写报告。还必须分配职责，以确保采取适当措施强化导致事件发展的安全控制结构的各个方面。随后采取后续措施确保修复有效。通常情况

下，在进行修复后，缺少确定修复是否成功地改进安全管理。最后，调查结果应作为未来审计和性能评估的参照。如果出现有相同因素造成的与之前相同事故和事故征候，则需要调查这些因素从未得到纠正的原因，或者为什么即使因素得到移除，一段时间后仍会再次发生。如果修复无法成功移除事故发生的原因，则需要对提出的建议和对建议的实行进行调查以确保识别组织在以上过程中的任何弱点并进行改进。也就是说，不但必须纠正事故中的涉事因素，而且还必须纠正在先前的事故或事故征候之后实施不足的修复过程。

报告系统：报告系统至关重要。通常，在事故后，同一事件在过去多次发生但从未报告过，或者已经报告过，但从未得到纠正。这些事件可能是侥幸脱险。例如，若干飞机可能无法正确进近，但由于不会发生事故即使存在报告系统，也可能就不会就此进行报告，直到发生事故才会采取行动。

事故报告中普遍调查结果是虽然存在报告系统但未使用。报告内容通常包括建议培训员工使用报告系统并要求使用报告系统。该建议假设问题出在潜在的报告者身上，而非报告系统本身的设计。

检查报告系统的自身检查可能发现系统本身使用困难或不方便，而且报告的信息似乎陷入了一个只进不出的境地。人们可能认为使用官方的报告系统意义不大，因为组织不会对报告的因素采取任何措施。通常，发现问题的人会绕过报告系统，直接找到他们认为可以解决问题的个人或团体。虽然这在短期内可能是有效且高效的，但它可能会导致今后出现同样的问题，因为系统性原因并未消除。在某些情况下，报告者由于担心报告的信息会对自身产生不利的影响，报告会被抑制。此时匿名报告系统就十分有用。

另一个需要考虑的因素是，一线运营人不会报告事件，因为他们在感知到风险之前就已经确定了问题，或者他们可能认为这只是他们自己的错误。风险作为工作流程的一部分，大多数人并未接受过识别风险的培训。人们将设计中的缺陷视为“正常现象”，并且可能已经对缺陷有所了解，工作中习惯了缺陷的存在，而不是去报告。一个相关因素是，当这些事件不符合所要求的报告标准时，人们通常不会报告危险事件。侥幸脱险的事件可能属于后一类。

一般而言，需要鼓励报告。通过可访问性最大化、焦虑最小化和对获得的信息采取行动来鼓励员工报告。报告表格或渠道应该容易获取，并且便于填写或使用。为了尽量减少焦虑，应该编制一份书面表格、表明报告流程、报告结果、报告和跟进报告人员的权利、特权、保护和义务。如果不存在书面政策，人们会因报告信息模糊减少报告。或者，不清楚谁负责报告跟进可能导致所有人都会认为其他人会关心报告跟进。

最后，鼓励报告还包括提供反馈。报告创建后，应立即通知提供信息的人员已收到报告，保证对报告的调查，并感谢报告者的意见。第二个关键的步骤是过后提

供反馈，反馈内容包括所有调查结果和为防止事故再次发生的调查步骤。报告者应觉得自己的担忧受到了重视。

设计与鼓励反馈的建议

- 设计并确保审计，性能评估和报告系统的持续有效性。
- 审核安全控制结构本身（包括所有级别）和所设计控制措施的有效性。
- 设计审核，使审核是建设性的学习经历，而不是一个批判的过程。
  - 参与者可以是审核小组的成员。
  - 将审核用作提高安全性和学习活动的方式，而不是评估员工。
- 利用审计来评估培训和教育活动的有效性，并提出反馈（知识评估），用于改进培训活动。
- 创建有效的系统级事故/事故征候致因分析程序，将关注点放在什么原因而不是是谁。
- 根据 STAMP 和系统思维（如 CAST）创建事件和调查程序，以识别系统因素，而不只是深层问题的表征。
- 将调查程序嵌入允许利用结果的组织结构中。指定责任或找到“根本原因”不应该是目标。
- 事故调查应在管理上和财务上与所涉及的直接管理结构相分离。考虑使用具有独立预算和高级管理的训练有素的团队。跟进建议，以确定它们是否有效，如果没有，那么为什么。
- 确保报告系统易于使用和可用，并且存在匿名报告渠道。
  - 鼓励报告并培训人们知道何时应该使用。
  - 提供书面政策
  - 最大限度地提高可访问性，减少焦虑，并根据获得的信息采取行动。
  - 向使用报告渠道的人提供反馈。报告者需要感觉他们的担忧不会被忽视。

风险管理

有时风险管理系统被称为“安全管理系统”，但它们并不相同。SMS 是一个更大的概念，可能（通常确实）涉及风险管理活动。风险管理是与识别危险，分析危险并使用此信息通过产品、流程、服务和工作场所的设计减少损失相关的一系列活动。这些活动将嵌入安全控制结构的某个地方。虽然风险管理包括预防危险所涉及的技术活动，但 SMS 还包含安全的组织，管理和文化方面。

STAMP 意味着更广泛或至少不一样的风险定义。传统上，风险被定义为发生危险或事故的严重性和可能性。相反，在 STAMP 中

风险是根据用于实施安全系统行为的控制措施的有效性来定义的，即安全控制结构的设计和运行。

注意该定义不要求确定事件发生的可能性，而是要求评估用于防止事件发生的控制措施的有效性。

建立风险管理系统涉及设计用于执行技术风险管理活动的程序，并将实施这些程序的责任分配给安全控制的组成。风险管理还可能涉及创建领先指标，以确定风险何时增加。

事实上，风险管理系统有效性的一个重要因素在于能够确定在发生重大损失之前风险何时增加。所有安全措施都涉及对系统组件的行为以及它们运行环境的假设。违反这些假设将破坏最初的风险识别和管理假设。事实上，所有事故都有未被认知的征兆，或者对有些征兆，其响应不存在或无效。事故的前兆是没发生损失但在其他情况下可能发生的意外事件或情况。领先指标是组织或组织运营的特征，表明安全管理系统未按设计预期来运行。

看待这种情况的另一种方法是，领先指标只是在损失发生之前的证据，即确保安全得以保证的假设存在缺陷或不再为真且风险可能会增加。例如，在壳牌 Moerdijk 化学工厂在荷兰发生爆炸之前，有两次情况导致其他壳牌工厂爆炸。这些爆炸从未被彻底探索过，或者至少在后来的设计活动中没有使用分析结果，例如 Shell Moerdijk 工厂的设计，其中假设这些情形不可能并且不会导致爆炸。令人惊讶的是，即使在从系统的实际使用中收集的具体证据表明分析中的假设不正确之后，概率风险评估通常也不会被重新评估。显然，计算出的数字的力量通常胜过这个数字是不真实的证据。生成有关风险评估真相的证据及其背后的假设应成为任何安全管理体系的重要组成部分。

虽然期望人们认识到只能眼见为实的前体是不现实的，但往往可以而且应该已经确定前体，并导致进一步调查。有效的 SMS 会具有持续警惕性，防止因时间推移安全控制结构退化。

一个常见的悖论是，一个组织发生的事故越少，未来发生事故的可能性就越大。如果安全管理系统有效，则损失就会很少。但是，人们倾向于以当前事故率判断的固有风险量。事故很少，证明安全管理系统非常有效，一段时间后会判断固有风险较低，安全管理水平下降，有效性降低。为使决策者的过程模型随时与实际风险水平保持同步，就需要反馈和信息。

风险管理的建议

- 创建识别危险，分析危险，然后在产品开发和运营中使用结果信息的程序。分配责任以确保这一过程。
- 强调定性程序，提供改进所需的信息，而不仅仅是定量评估。不要对不可知的可能性进行定量评估。
- 在获得数据以及随时间发生变化时重新评估风险评估结果。
- 提供方法来确定风险何时大于最初假设。
- 如果不遵循程序，请不要只是尝试强制执行。反而评估为何不遵守的原因并重新设计。尝试弄清规定内容与正在发生的内容之间的差距。

教育和培训

安全控制结构中的每个人，不仅仅是物理系统的低层控制人员，都必须了解他们在安全方面的角色和责任，以及为什么系统（包括安全控制结构的组织方面）的设计方式。如果员工了解 SMS 的意图并服从，他们更有可能遵守该意图，而不是在方便的时候遵循规则。训练是不够的; 需要教育。

教育不仅必须包括有关安全和由控制措施所实施和安全限制的信息，还必须包括优先事项以及如何做出决策。前面讨论过的安全理念陈述提供了有关决策中使用安全价值观的信息。此外，每个人都需要了解他们在做出决策时所承担的风险。通常，由于对假设的风险进行了错误的评估，导致决策失误。使用上面提供的非概率性风险定义，这意味着决策者必须知道他们的决策将如何影响安全控制结构中所设计的控制措施。

告诉管理者和员工“注意微弱信号”（高可靠性组织或人力资源管理文献中的一个常见建议）只是在发生损失事件后创造了一个责备的借口，并且后见之明提供了将弱信号（噪音）变为一个强烈信号的声明。相反，每个人都必须接受与系统运行相关的危险的培训，以及如果我们期望他们认识到事故前兆，如何识别它们。人们需要知道要寻找什么，而不仅仅是被告知寻找未定义的东西。

培训还应该包括“为什么”以及“什么”。了解他们被要求遵循的安全规则背后的基本原理将有助于减少自满情绪，看似鲁莽的行为（但对于完全有意义的人），以及意外的变化导致危险。这些基本原理包括理解为什么先前发生的事故以及为防止再次发生而做出的改变。随着自动化程度的提高和人们与之交互，了解所涉及的受控危险可能需要比过去更多的培训，过去危险较明显和直观。与复杂系统交互的人需要学习的不仅仅是要遵循的程序：他们还必须深入了解受控物理过程以及他们可能正在监督或进行交互的任何自动控制器（软件）中使用的逻辑。安全控制结构各级需要知道的控制器清单包含在“更安全的工程世界”中，并在此处重复：

- 系统危险以及安全关键程序和操作规则背后的原因。
- 删除或覆盖控制，改变规定程序的潜在结果；不注意安全关键功能和操作：应评审和理解过去的事故及其原因。
- 如何解释反馈：培训需要包括警报和事件序列的不同组合，而不仅仅是单个事件。
- 如何在解决问题时灵活思考：控制人员需要有机会实践涉及安全的问题解决。
- 一般策略而非具体响应：控制者需要培养处理意外事件的技能。
- 如何以适当的方式测试假设：为了更新心智模型，人工控制器通常使用假设检验来了解当前的系统状态并更新其过程模型。这种假设检验在计算机和自动化系统中很常见，其中文档通常很差且难以使用，因此实验通常是理解自动化设计和行为的唯一方法。然而，假设检验可能导致损失。设计人员需要为操作员提供安全测试假设的能力，并且必须教育控制人员如何操作。
- 紧急程序必须反复学习并不断实践。需要向控制人员讲授操作限制以及在超出限制时要采取的特定操作。要求操作人员在压力和没有完整信息之下做出正确的决策，只确保他们在发生损失后才受到指责。

培训不应该是员工的单次事件，而应该在整个就业期间持续进行，只是为了提醒他们的责任和系统危险。了解最近的事件和趋势应该是这一培训的一部分。通过定期审计和绩效评估来评估培训的有效性，可以有助于实施有效的改进和学习过程。意外事件和事故调查结果是培训效果的重要信息来源。

一些公司创建了一种管理人员提供安全培训的实践。培训专家帮助管理小组动态和课程开发，但培训由项目或小组负责人提供。通过学习教材，主管和经理更有可能吸收和实践关键原则。此外，已经发现，员工更多地关注由他们的老板传递的信息而不是教练或安全专家。

教育和培训的建议

- 教育和训练兼具
- 每个人都应该了解他们的角色和责任，系统为什么这样设计，控制所执行的危险和安全约束的信息，以及他们在决策中所承担的风险。
- 培训应包括“为什么”而不仅仅是“什么”
- 每个人都应该学习：
  - 系统危险以及与其工作相关的安全关键程序和操作规则背后的原因。
  - 删除或覆盖控制，更改规定程序以及注意安全关键功能和操作的潜在结果。
- 确保过去事故的原因得到充分传播和理解。
- 让人们有机会练习涉及安全的问题解决
- 教授一般策略而不是具体的反应，致使控制者可以提高处理意外事件的技能。
- 教导操作员如何以适当的方式测试假设，并为他们提供以这种方式学习的方法。
- 紧急程序应该反复学习并不断实践。
- 应在任何安全关键活动之前评审应急程序。
- 培训应该是持续的，而不仅仅是被雇用时的单次事件。
- 讲授最近的事件和趋势
- 考虑让经理提供安全培训。

学习和持续改进：

因为 SMS 设计从一开始就很难完美，并且世界会随着时间而变化，所以必须有一个有效的流程来确保组织不断学习安全事件并改进 SMS 本身以及工作场所，产品和 服务。

实验是学习过程的重要组成部分，应该鼓励尝试新的想法和方法来提高安全性，同时也要仔细评估以确保实际改进。

4. 安全信息系统（SIS）

全面可用的安全信息系统是 SMS 成功的关键。SIS 是控制结构有效性的信息来源。在 STAMP 术语中，这是风险的定义。

控制者和决策者的过程模型必须保持精准和协调。实质上，SIS 充当着共享过程模型，是更新单个过程模型的来源。因此，准确及时的反馈和数据非常重要。SIS 可以提供检测事故趋势、变化和其他前兆所需的信息；以评估安全控制的有效性；将模型和风险评估与实际行为进行比较；并从事件中汲取教训并改进 SMS。在发生重大事故之后，经常发现可防止损失的信息已存在但是未经使用或无法被相关人员

使用。一般而言，收集大量信息只是因为政府报告需要这些信息，而不一定是有效SMS 运行的必要条件。

为了确定 SIS 中应该包含的内容，可以使用安全控制结构中的职责以及履行这些职责所需的反馈，即便于做出适当的决策的过程/心智模型中所需的信息。通常，SIS 至少包含：

- 安全管理计划（用于开发和运营），
- 系统所有安全活动的状况，
- 安全约束和设计假设，包括运行限制，
- 危险源分析的结果（危险源日志）以及所有已知危险源的跟踪和状态信息，
- 绩效审计和评估的结果，
- 事故和事故征候调查报告以及采取的纠正措施，
- 经验教训和过去的信息，以及
- 趋势分析结果。

使用安全控制结构设计来识别其中每个人进行安全决策所需的信息，安全控制结构有助于设计便于所有人能够找到所需信息的、可用的 SIS，

信息可以由公司或行业收集。无论收集信息的方式如何，了解其局限性都是很重要。最有用的信息必须准确和及时，并且必须以有用的形式传播给适当的人。涉及因素有三：收集、分析和传播。

*收集：*数据可能因收集方式而失真。常见问题是系统性过滤、抑制和不可靠。为事故报告收集的数据往往侧重于近端事件和参与者，而不是所涉及的系统性因素，如管理问题或组织缺陷。研究表明[35]，运营人报告的数据过滤了侥幸负职的数据，事故原因主要指向运营人失误。管理层的报告同样经滤，指出事故原因是运营人失误。即使是一般的安全检查也倾向于识别有限的类别，尤其是使用检查单时。

对于与过去相似的事故，数据收集往往更可靠，而对于新系统，过去的危险经历和因果因素更为有限。由于缺乏对此类错误的知识，缺少可接受的和一致性的分类，或者根本没有将其作为因果因素考虑，软件错误和计算机问题经常受到忽略或描述不充分。

经验表明，很难在长时间内保持可靠的事件报告，因此这类数据对于估计事件的可能性或其潜在后果并不十分有效。根据数据显示，在某些行业中，未报告的事故多达四分之三。有时，信息涉及公司希望保密或者报告者可能会担心的人在工作绩效评估中对自己有不利的影响。也可能涉及潜在的法律责任。

对已发生的运行问题所进行的讨论应记录在 SIS 中，并分析其潜在危害。虽然如前所述，如果将问题直接提交给能够解决问题的人，绕过在 SIS 中进行归档是很

[35] 如管理问题或组织缺陷。研究表明，这句话的注释：本节的许多科学参考资料可以在 Nancy G. Leveson，Safeware，Addison Wesley Publishers，1995，第 11 章中找到。我们尽可能完善手册的阅读体验。

常见的，但此类讨论或电子邮件通常可能包含参与者未承认的安全问题。此外，因为讨论中包含有价值的公司信息，而人们离职或退休或由于电子邮件保存方式容易丢失，所以将这些讨论记录下来可能是有益的。要求保留和记录讨论内容，或者公司开通沟通渠道自动保留此类信息。

有些措施可用于提高数据收集的全面性和可靠性，例如数据收集人员的特殊培训，数据收集人员的结果反馈，固定例程和匿名报告。对 CAST 等技术的培训可以促进易忽略信息的留存。自动监控系统现在很是常见，但问题在于它们可以收集并且通常收集的信息过多，以至于很难分析结果和检测问题。STPA 可能是一种帮助确定需要收集有用信息类型的方法，并确定所收集的大量数据中的领先指标。

*分析：*将大量数据系统化并整合成对学习有用的形式中是很困难的。特别是如果不包括统计显着性，原始定量数据可能会造成误导：数据与信息不同。同样，STPA 结果不仅可用于识别有哪些数据需要收集，还可用于为正在发生的事件的重要性提出指导。此外，用于验证 STPA 致因分析结果和识别理应被消除或缓解的因素的数据应包括在是 SIS 收集和分析过程中。

分析中最大的问题是虽然设计数据收集渠道很容易，但找到分析所有结果数据的时间和人力可能很困难或不切实际。像 STPA 这样的工具可以帮助识别需收集的最重要数据和最有用的分析过程。

传播：以有用的形式传播信息（而非数据）可能是 SIS 最困难的部分。信息需要以人们可以学习的形式呈现，应用于人们的日常工作中，并在整个项目进行期间中使用，而不仅停留在概念设计阶段。并且必须及时更新：事故是由于运行期间的变化或由于在进行设计改进时对危险源分析更新不充分造成的。STPA 强调包括基本原理和意图以及源头等可用文档的，以协助变更过程。此外，安全控制结构中给个人和团体分担的安全责任将提供必要的信息，以便根据接收者的需要定制信息。向个人提供过多信息与提供太少信息一样，是有危险的。

呈现信息的方法应该适合用户的认知风格，并且融入到安全决策的环境中。同样，安全控制结构设计过程将就需要哪些信息，何时需要信息，以及控制结构中各控制器如何使用等提供大量有用输入。

设计安全信息系统的建议

- 准确及时的反馈和数据非常重要
- SIS 应提供检测事故趋势，变化和其他前兆所需的信息；评估安全控制的有效性；将模型和风险评估与实际行为进行比较；并从事件中汲取教训并改进 SMS。
- 使用安全控制结构中定义的职责来确定他们所需的信息，以保持其过程模型足够准确，以便做出正确的决策。
- 了解收集数据的局限性。
- 为了最有用，信息必须准确及时，并且必须以有效的形式传播给适当的人。
- 找到收集数据的方法，最大限度地减少数据失真（过滤，抑制和不真实数据）。
- 保存实际安全相关事件的详细信息，并将信息提供给需要的人员。
- 创建提高数据收集全面性和可靠性的方法。
- 将统计显着性包含在数值数据中。
- 使用 STPA 识别需要收集的数据，并对正在发生的事件的重要性提供指导。
- 收集数据以验证 STPA 致因分析结果并确定原以为得到消除或减轻的因素。
- 确保数据得到了分析而不仅仅是收集数据。
- 以人们可以从中汲取教训并将其应用于日常工作的方式呈现数据。
- 保持数据更新。
- 记录设计的基本原理和意图，并提供可追溯性，协助更变过程。
- 根据收件人的需要定制所提供的信息。
- 提供过多的数据，特别是原始数据，可能与提供过少的数据一样，都是危险的。
- 使信息的呈现适合用户的认知风格，并将其集成到与安全相关的决策环境中。
- 使用安全控制设计过程来确定需要哪些信息，何时需要以及如何使用。

5. 小结

一般而言，有效的安全管理要求：

- 各级的安全承诺和领导
- 强大的企业安全文化
- 利益相关者共享清晰明确的安全愿景、价值观和程序方法
- 适当分配了责任、权限和责任的安全控制结构
- 提供安全控制结构各级安全状态的准确视图的反馈通道
- 将安全纳入开发和生产线运营（不仅仅是一个独立的部分或单独的亚文化）
- 具有适当知识，技能和能力的个人
- 拥有合作伙伴角色和责任的利益相关者
- 解决安全优先事项与其他优先事项之间紧张关系的指定程序
- 传播安全信息的风险意识和沟通渠道
- 对朝着高风险变迁的系统进行控制
- 有效和可用的安全信息系统
- 持续改进和学习
- 教育、培训和能力发展

有关更多信息，请参阅“安全软件工程更安全世界”第 13 章和“安全软件”第 11 章和第 12 章。本手册附录 D 还列出了有效安全控制结构中需涵盖的责任清单。

第 8 章：将 STPA 引入您的组织机构

约翰·托马斯、南希·莱文森

对于相对较小的组织机构，引入 STPA 可能只涉及培训，或者示范项目，以确定使用 STPA 的实用性和优势是值得的。对于大型组织机构而言，可能涉及其他挑战，例如改变文化、改变所使用的标准流程、以及培训大量人员去做一些与过去不同的新事情。本章探讨了当组织机构决定评估或采用 STPA 时出现的一些问题，以在整个组织机构中广泛使用 STPA。

1. 培训

学习 STPA 需要练习和“边做边学”的方法。正式的面对面培训非常有效，我们发现专业人员可以在短短几天内接受培训并准备开始尝试 STPA。最有效的 STPA 培训是交互式的动手练习，用于加强这一过程。令人惊讶的是，使用传统危险分析技术的人员可能在学习 STPA 方面遇到最大困难——似乎需要相当多的学习，并且可能需要额外的培训。在我们的经验中，系统工程师往往学得最快。

权衡取决于培训课程的规模。小班（10 人以下）可提供对 STPA 的深入和详细的了解，而且对于想要开展小型试点研究来评估 STPA 的小型组织机构来说，这是一个很好的选择。大型课程（超过 50 人）需要更长的时间并限制可能的交互量，但对于需要向大型团队提供 STPA 基本认知的组织来说，这是一个很好的选择。中等班级（约 20-30 人）倾向于提供合理的折中。

对于需要培训数百甚至数千人的大组织机构，可能需要“培训培训师”的方法。培养可以作为未来培训师和协调人（见下文）的 STPA 专家的最有效的方法是让候选人沉浸在积极使用 STPA 的实际项目中。参加短期培训课程不足以培养 STPA 专家，但那些沉浸在少量大型 STPA 项目中的人员是未来培训师和协调人的理想选择。一种行之有效的方法是，允许一名或多名接受培训的协调人学习从事实际项目的 STPA 协调人。接受培训的协调人通过亲眼目睹不同项目中遇到的挑战和提出的问题，学到了很多东西。一旦他们展示了提供有效指导、解决挑战并回答其他参与者的关于 STPA 的详细问题的能力，他们就可以随时为自己提供便利。

本手册应提供更多在之前没有的培训选项。此外，我们正在制作与手册一起使用的视频。

2. 协调人

STPA 在集成到设计工作中时非常有用，它不是由单独的团队作为独立工作执行的。但是，并非所有人都希望或需要接受 STPA 专家的培训。在 HAZOP 中使用协调人的想法与我们为 STPA 鼓励它的原因相同，即系统设计专家成为最好的 STPA 团队参与者，但如果他们不熟悉 STPA，可能他们需要实际应用该技术的指导。

STPA 协调人的工作是在整个过程中提供 STPA 专业知识和指导，并帮助避免可能遇到的常见陷阱。协调人可以在项目选择和范围界定期间提供指导；可以帮助确定最初的团队成员和所需的专业知识；可以根据以前类似的项目制定一套损失、危险和潜在的初始控制结构；并在整个过程中提供全面协调。对于大型项目而言，协调人把过程分解为可由个人或小组专家成功完成的较小组件。协调人可以识别和安排可由不同组并行完成的任务，并且最终协调人可以将这些组件整合到整体分析中。协调人还可以领导参与者的会议并管理团队成员之间的互动。

协调人可以在项目初始提供一些初步的 STPA 培训，他们应该能够回答项目期间出现的 STPA 相关问题。协调人通常会在流程期间和流程结束时审核结果，以确保正确遵循方法，并且不会忽略任何差距。在项目结束时，协调人可将结果汇总成最终报告，并确保将结果提供给适当的人员。

3. 团队组成

STPA 最好由一个包括不同类型专家的小团队执行。初始控制结构可以帮助确定所需的专业领域。跨学科团队通常包括来自不同领域（如软件专家、运营专家、维护专家等）的专业知识。在某些情况下，可能无法将所有相关专家都包括在 STPA 团队中（例如，当专家需求量很大时）。在这种情况下，您可能希望确保团队至少可以询问具有适当技术知识的人员，并且您可以获得分析过程中出现的问题的答案。

个性很重要。系统的工程师和设计师可能拥有最多的专业知识，但他们有时可能会采取防御措施，并将 STPA 工作视为对其工作质量的挑战。他们可能不想贡献潜在的 UCAs，因为他们认为他们已经在设计中得到了缓解，或者他们可能认为已确定的 UCAs 被认为是不可能的，应该被移除或不用分析。另一方面，没有参与此特定设计的局外人或其他设计师可能对该系统知之甚少，但他们可能不具备防御性，并且可能乐于提出潜在的缺陷。这可能有助于向工程师和设计师解释 STPA 是一种最坏情况的分析方法，它可以通过避免返工而不是使其更难以减轻他们的工作。在设计完成之前和已经做出决定之前，尽早开始应用 STPA 也很有帮助。当项目工程师自己应用 STPA 来帮助指导决策和做出明智的选择时，这是最有利的。在任何情况下，个性都很重要，理想情况下，您应该尝试找到知识渊博的专家，因为他们对

新方法持开放态度，并且会毫不犹豫地发现以前可能忽略的问题。

我们经常发现，深入参与项目设计的人员的最初怀疑态度在他们找到以前未发现过的 UCAs 或方案时很快就会被克服。然后他们很快就成为 STPA 的协调人。

具有测试背景或经验的人可以成为 STPA 团队的绝佳人选。测试人员可以非常了解可能出现的问题，通常他们不仅愿意而且渴望找到可能被忽视的问题。测试人员很少认为需求、规范或设计决策是正确的，他们的工作涉及不断质疑已经做出的假设和声明。这些都是 STPA 团队成员的优秀特征。

4. 成本和投资回报

STPA 非常有效。图 8.1 显示了所有 STPA 团队成员在最近的行业项目上花费的相对时间。对于大多数行业的 STPA 项目，大部分时间实际上并不花在学习 STPA 或执行 STPA 上，而是学习如何分析系统（对于任何识别危险原因的方法都需要这样做）。大约四分之一的时间用于寻找关于系统的“假设”技术问题的答案，这些问题以前没有被考虑过。通常花费相对少量的时间来学习 STPA 并遵循该过程。

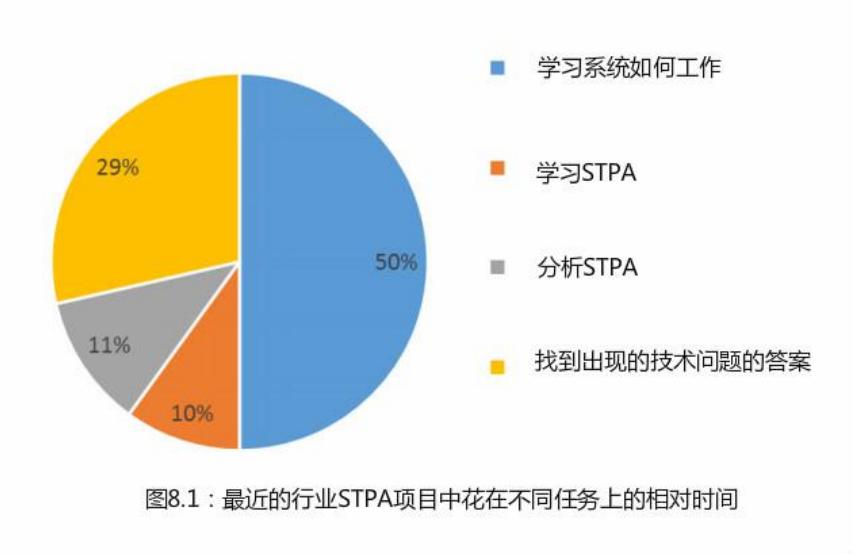


图8.1：最近的行业STPA项目中花在不同任务上的相对时间

STPA 不是推文练习。大部分时间都花在探索系统中的真正问题上，这种探索可以立即用于推动决策并提供有关如何改进的见解。

虽然我们没有大量精心收集的数据比较所需的时间，但有人报告说 STPA 比传统的危险分析技术需要的时间和资源少 2 或 3 个数量级，并能产生更完整的结果。通常，由于学习成本的原因，第一次使用成本最高，但在重复项目上的成本很快就会降低。

5. 在大型组织机构中使用 STPA 的示例

第 4 章描述了在一个大型组织机构中使用 STPA 来构建复杂的高度自动化的系统。我们在这里重复一遍。在这种情况下，STPA 被一个八人组成的跨职能团队应用，该团队由集成、工程、运营和维护成员组成。所有相关人员都是其系统责任范围内的主管或项目问题的专家。协调人是麻省理工学院硕士生，曾接受 STPA 培训，但对引入工厂的新自动化制造工艺知之甚少。团队的其他成员在项目开始时并不了解 STPA。

分析在三周的时间内完成。设立的评估小组在该过程中仅使用两次，每次不超过三小时，而其余评估是在协调人和所讨论部分的相应专家之间的一对一会议中完成。

由于 STAMP 和 STPA 对于公司来说相对较新，并且对于团队而言是全新的，因此第一次小组会议作为指导期，以提供所使用的方法概述，然后定义团队想要避免的事故（或损失）、危险和系统边界，以及本应用的安全控制结构初稿。

初始控制结构由整个集团开发，以确保捕获所有主要组件。在制定控制结构的初稿后，安排后续会议，由具有相应技术专长的个人专家或小组在系统的不同部分添加细节。

一旦制定了适当级别的控制结构细节，就会继续进行一对一或小组讨论，确定不安全控制措施和方案。然后在另一组中讨论该部分分析的结果，以确认该组的各个发现并讨论该系统的潜在缓解措施。这个过程在三个星期内发生，在小组、一对一会议和单个协调人工作中花费的时间总计约 300 个小时。

一般而言，所确定的方案包括诸如硬件故障、不正确的过程模型（源于缺失或不正确的反馈）、系统组件交互以及管理和程序对安全的作用以及工厂中的进度压力等因果因素。在团队确定 STPA 方案后，他们使用结果来为系统开发新的控制措施，以消除或防止进入危险状态。示例包含在本手册的第 4 章中。

由于 STPA 使用控制结构模型，因此在构建完整模型后，评估可以关注模型中的特定控制回路。这种划分建模和分析过程的能力可以在组内每个人可用的时间范围内轻松安排会议时间和大量会议。在这些会议期间，增加了诸如控制措施、过程模型、反馈等细节，并且与受影响的各方一道解决了冲突或问题。

6. 其他建议

我们打算在我们获得 STPA 时将更多信息纳入大型组织机构中，包括投资回报数据。事实上，许多组织机构在对几个项目进行尝试并将其与现在的工作进行比较后，采用 STPA 是令人鼓舞的。如果您有任何意见，请告诉我们。

附录 A：危险源的例子

行业内的危险源通常非常相似。一旦确定了适用于您的行业、产品或服务的危险源，您就可以重复使用该列表，可能只需要很小的改动。本附录包含一些已在实际项目中使用的危险源列表。当然，利益相关者是考虑哪类损失和危险源的最终决策者。对特定项目的利益相关者而言，重要的损失显然会影响所发现的危险源。

核电站

损失：

L1：人员受伤或死亡

L2：环境污染

L3：设备损坏（经济损失）

L4：发电量损失

危险源

H1：释放放射性物质[L1，L2，L3，L4]

H2：反应器温度过高[L1，L2，L3，L4]

H3：超出限制运行的设备[L3，L4]

H4：反应堆关闭[L4]

飞机

损失：

L1：失去生命或受到严重伤害

L2：飞机或飞机外物体的损坏

危险源

H-1：飞机违反飞行中的最低间隔标准[L1，L2]

H-2：飞机进入地形的受控飞行[L1，L2]

H-3：飞机失控[L1，L2]

H-4：飞机机身完整性丢失[L1，L2]

H-5：飞机环境对人体健康有害[L1，L2]

H-6：飞机离开指定的滑行道、跑道或地面的停机坪[L1，L2]

H-7：飞机太靠近地面上的其他物体[L1，L2]

放射治疗

损失：

L1：患者因过度暴露或治疗不足而受伤或死亡。

L2：非患者因辐射而受伤或死亡。

L3：设备损坏或丢失。

L4：治疗期间对患者或非患者的身体伤害。

危险源：

H1：错误的剂量：输送给患者的剂量在数量、位置或时间上是错误的[L1]。

H1a：正确的患者，正确的剂量，错误的位置。

H1b：正确的患者，错误的剂量，正确的位置。

H1c：正确的患者，错误的剂量，错误的位置。

H1d：病人不对。

H2：非患者不必要地暴露于辐射[L2]。

H3：设备受到不必要的压力[L3]。

H4：人受到非放射性损伤[L4]。

军用航空

小灾难

M-1：飞机或机上设损失或损坏

M-2：对人员造成严重伤害或死亡

M-3：无法完成任务

危险源

H-1：违反固定或移动物体的最低间隔标准[M-1，M-2，M-3]

H-2：无法控制飞机[M-1，M-2，M-3]

H-3：机身完整性丧失[M-1，M-2，M-3]

根据飞机的任务，其他危险可能是相关的。例如，对于飞机上的武器系统：

H-4：没有指令爆炸[M-1，M-2，M-3]

H-5：没有指令发射[M-1，M-2，M-3]

H-6：协同损害或向友军开火[M-1，M-2，M-3]

H-7：指令时没有部署（爆炸和/或发射）法令[M-3]

汽车

损失

L1：失去生命或受到严重伤害

L2：车辆或车辆外部物体的损坏

危险源

H1：车辆与附近物体未保持安全距离[L1，L2]

H2：车辆进入危险区域[L1，L2]

H3：车辆超过环境安全运行范围（速度、横向/纵向力）[L1，L2]

H4：车辆乘员受到有害影响和/或健康危险源[L1，L2]

（例如火灾、温度过高、无法逃生、乘客关上车门等）

附录 B：示例分层控制结构

本附录展示了我们在自己的项目中使用的一些控制结构。

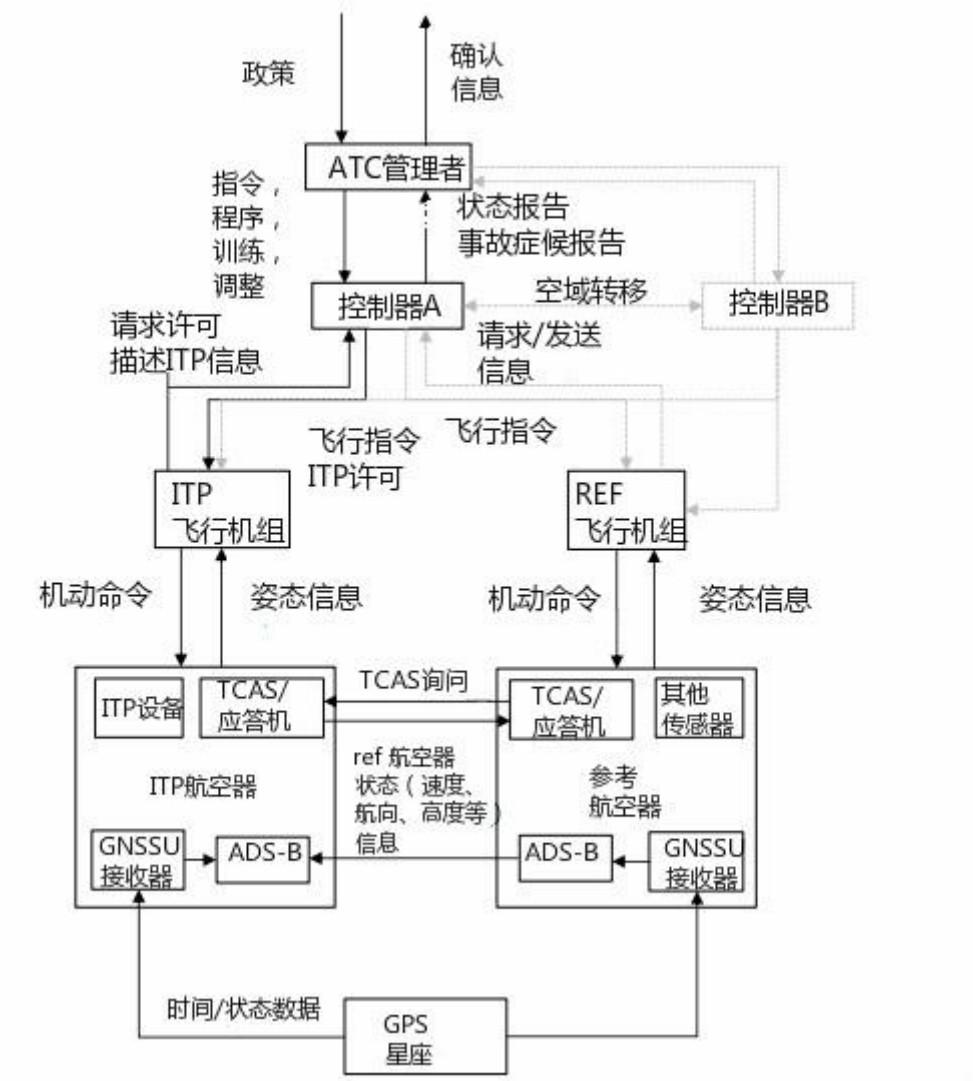
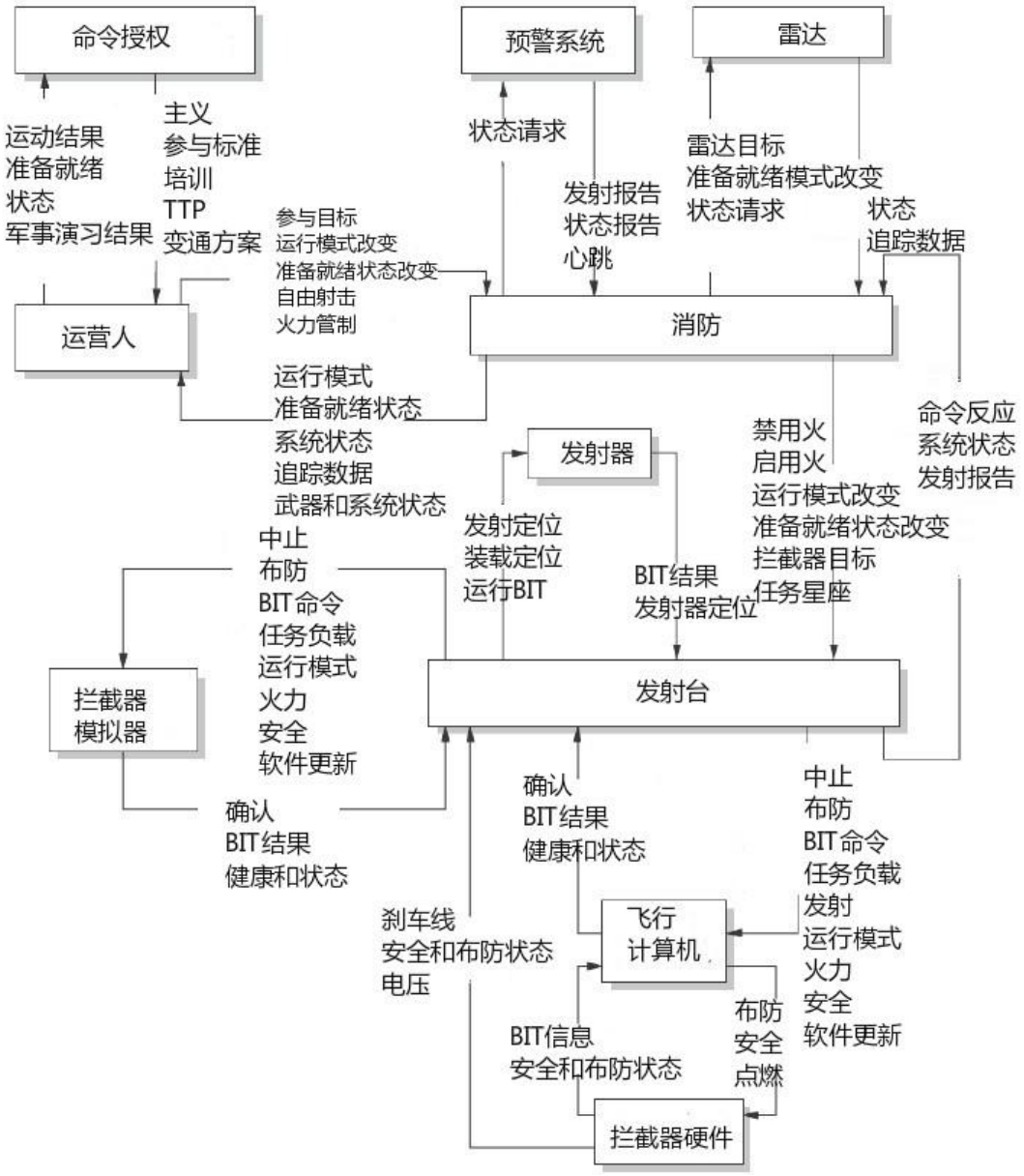


图 B.1：使用新设备（航空）的 NextGen In-Trail 程序的控制结构



图B.2：虚构导弹拦截系统的控制结构

图 B.3：质子治疗机的高级（抽象）控制结构

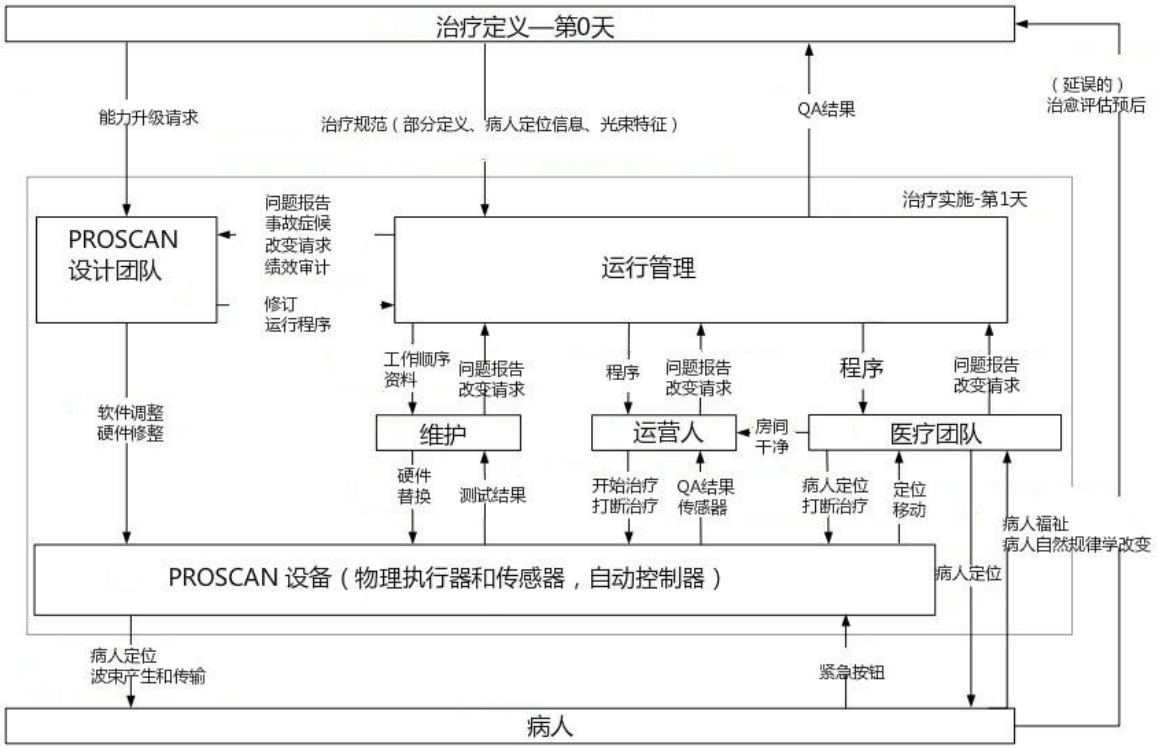
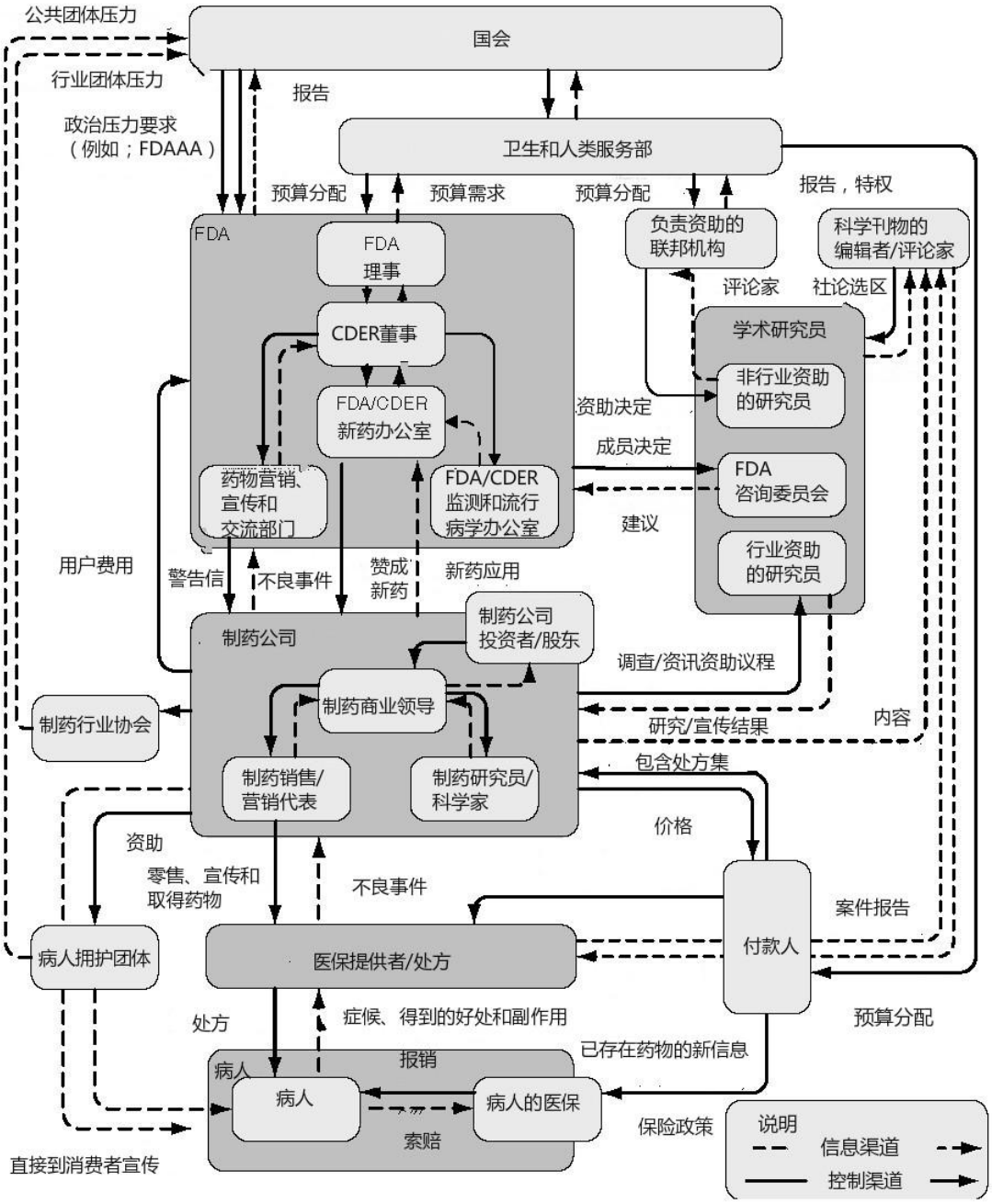


图 B.4：放大图 B.3 的治疗实施部分，显示治疗定义和患者组件，以提供背景信息



图B.5：美国药品批准安全控制结构

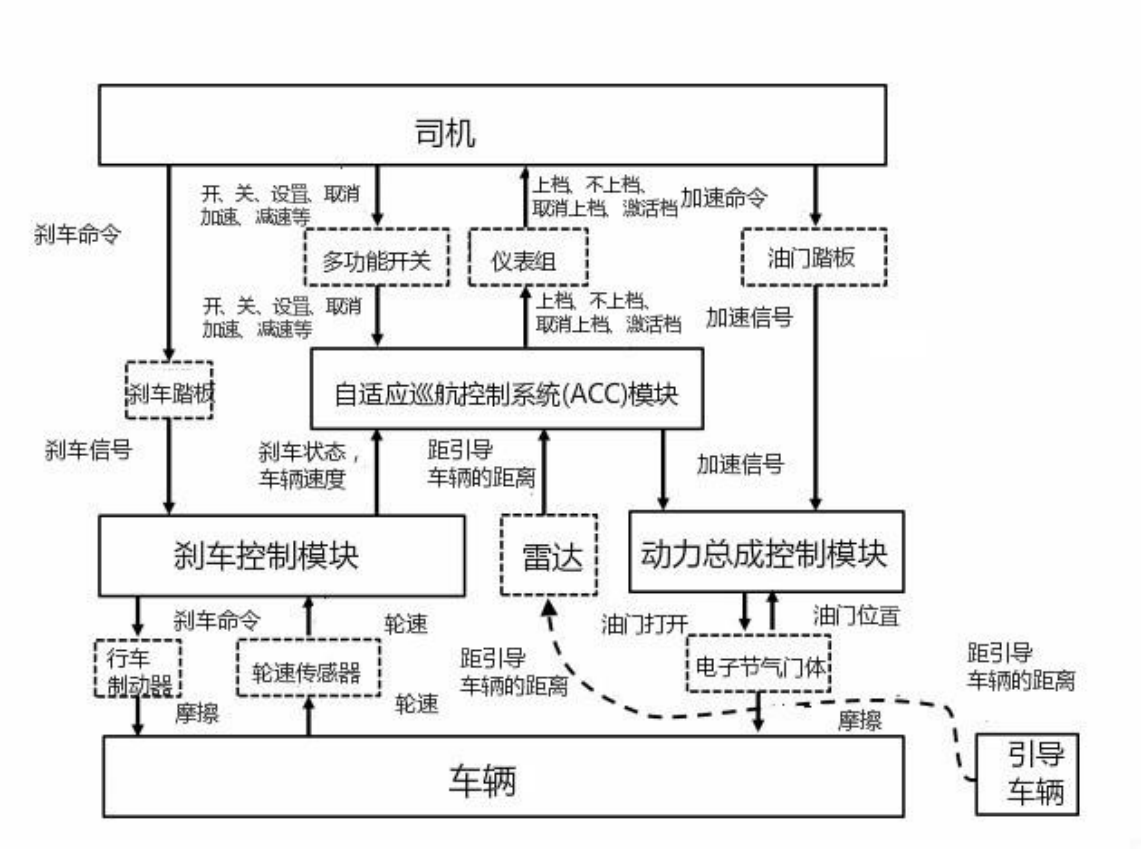


图 B.6：汽车自适应巡航控制系统

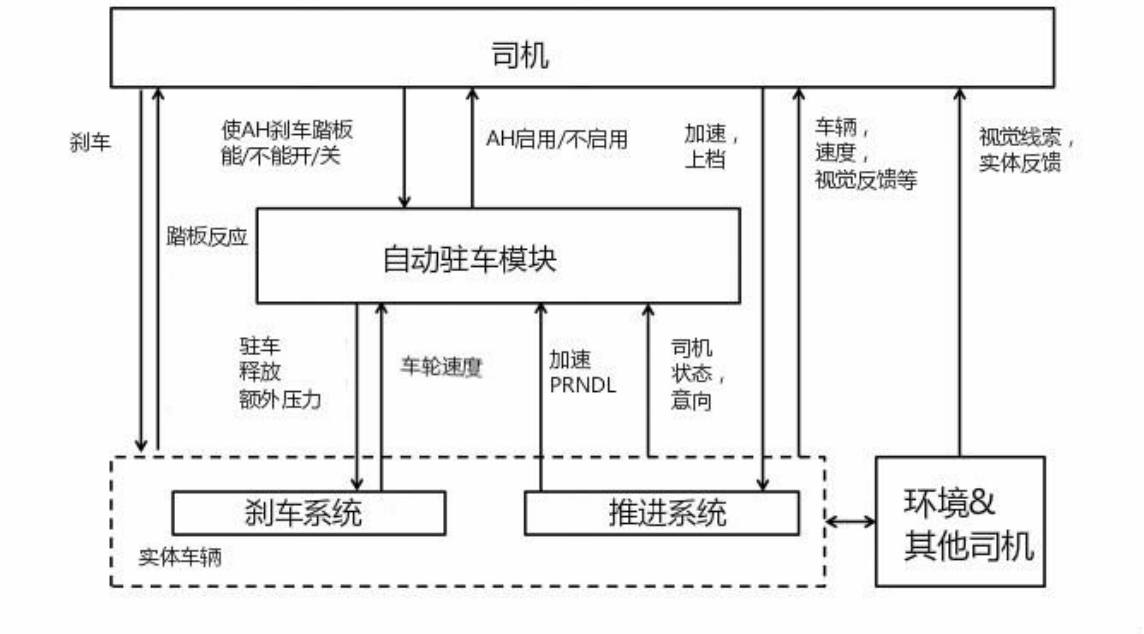


图 B.7：汽车自动驻车系统的控制结构

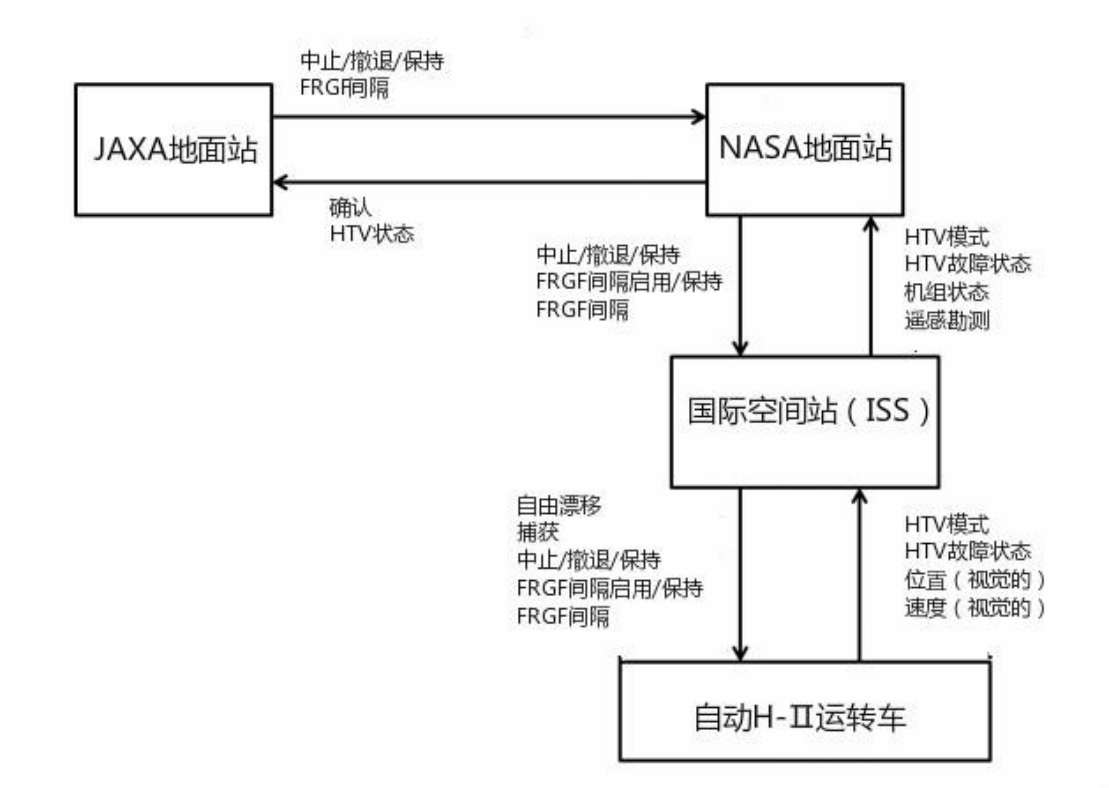


图 B.8：控制结构，例如自主空间飞行器运行

附录 C：UCA 表的更多示例

图 C.1：汽车自动驻车系统的 UCAs

控制行为（自动驻车）	不提供导致危险的原因	提供导致危险的原因	时间/顺序不正确	停止太快/应用太久
驻车命令	UCA-AH-1：车辆停止和制动踏板释放时，AH 不提供 HOLD[H-1,2]	UCA-AH-2：当驾驶员使用加速器时，AH 提供 HOLD[H-1] UCA-AH-3：当 AH 被禁用时，AH 提供 HOLD[H-1] UCA-AH-4：在车辆行驶时，AH 提供 HOLD[H-1] UCA-AH-5：在驾驶员未施加制动时，AH 提供 HOLD[H-1,2]	UCA-AH-6：在未满足静止所需时间之前，AH 过早提供 HOLD[H-1] UCA-AH-7：在车辆停止后开始滚动时，AH 太晚提供 HOLD[H-1]	N/A
释放命令	UCA-AH-10：当驾驶员使用油门踏板时，AH 不提供 RELEASE [H-1,2]	UCA-AH-12：当驾驶员没有使用加速器时，AH 提供 RELEASE [H-1,2]	UCA-AH-13：AH 在有足够的车轮扭矩之前提供 RELEASE [H-1,2] UCA-AH-13：加速器施加后，发动机扭矩超过车轮扭矩，AH 提供 RELEASE 过晚[H-1,2]	N/A
额外压力命令	UCA-AH-14：当车辆打滑且 AH 有效时，AH 不提供额外压力 [H-1,2]	UCA-AH-15：AH 在 AH 不活动时提供额外压力[H1,2] UCA-AH-16：超出制动系统规格时，AH 提供额外压力[H-1,2]	UCA-AH-16：AH 在车辆打滑后提供额外压力[H-1,2]	N/A

表 C.2：用于自主 H-II 转运车辆（HTV）运行的 UCA

控制行为（来自国际空间站的宇航员）	不提供导致危险的原因	提供导致危险的原因	顺序太早/太晚	停止太快/应用太久
中止	当紧急情况存在时，ISS 工作人员不提供中止命令 [H1]	SS 工作人员在捕获 HTV 时提供中止命令 [H-1] 当 ISS 处于中止路径时，ISS 工作人员提供中止命令 [H-1]	ISS 工作人员提供中止命令为时已晚，以至于发生碰撞[H-1] ISS 工作人员在捕获发布之前提供中止命令[H-1]	N/A
自由漂移	当 HTV 在捕获箱中停止时，ISS 工作人员不提供自由漂移命令 [H-1]	当 HTV 接近 ISS 时，ISS 工作人员提供自由漂移命令 [H-1]	ISS 工作人员提供自由漂移命令太晚，超过 HTV 停止后 X 分钟[H-1] ISS 工作人员在 HTV 停止前过早提供自由漂移命令[H1]	N/A
捕获	当 HTV 处于自由漂移的捕获箱中时，ISS 工作人员不执行捕获[H-1]	当 HTV 没有自由漂移时，ISS 工作人员执行捕获[H-1] 当 HTV 中止时，ISS 工作人员执行捕获[H-1] ISS 工作人员以过度/不足的运动进行捕获（可能会影响 HTV，导致碰撞过程）[H-1]	ISS 工作人员执行捕获太晚，超过 HTV 停用后 X 分钟[H-1] ISS 工作人员在 HTV 停用前过早执行捕获 [H-1]	紧急情况存在后，ISS 工作人员继续执行捕获[H-1]

表 C.3：与车轮制动系统相关的飞机飞行机组的 UCA

控制行为（来自飞行机组）	不提供导致危险的原因	提供导致危险的原因	顺序太早/太晚/不按顺序	停止太快/应用太久
CREW.1 通过制动踏板手动制动	CREW.1a1： 当自动制动没有提供制动或提供的制动不足时，在着陆、RTO 或滑行期间机组不提供手动制动[H4.1]	CREW.1b1： 踏板压力不足时，机组提供手动制动[H4.1] CREW.1b2： 踏板压力过大时（导致失控、乘客/机组人员受伤、制动器过热，制动褪色或着陆时轮胎爆裂），机组提供手动制动[H4-1，H4-5] CREW.1b3： 机组在正常起飞时提供手动制动[H4-2，H4-5]	CREW.1c1： 在接地之前（导致车轮失锁、失控、轮胎爆裂），机组提供手动制动[H4.1] CREW.1.c2： 机组提供手动制动太晚（TBD）以至于不能避免与其他物体发生碰撞或冲突（在飞机重量、速度、物体距离（冲突）和停机坪条件下可能超载制动能力）[H4-1，H45]	CREW.1d1： 在达到安全滑行速度（TBD）之前，机组停止提供手动制动命令[H4.1，H4.4] CREW.1d2： 机组提供手动制动时间过长（导致飞机停在跑道或主动滑行道上）[H4-1]
CREW.2 布防自动制动	CREW.2a1： 在着陆时，机组不会对自动制动进行布防（导致扰流板展开时失去自动制动操作。机组反应时间可能导致过冲。）[H4-1，H4-5]	CREW.2b1： 在起飞过程中，机组不会将自动制动设置为最高水平。（假设拒绝起飞需要最大制动力）[H4-2] CREW.2b2： 对于跑道状况具有过高的减速率，机组布防自动制动（导致失控和乘客或机组人员受	CREW.2c1： 机组提供过时的布防命令（TBD）（导致 BSCU 没有足够的时间进行制动）[H4-1，H4-5]	

	Crew.2a2： 在起飞前，机组不会对自动制动进行布防（导致在拒绝起飞期间制动不足，假设自动制动负责在机组人员减速后的 RTO 期间制动）[H4-2]	伤）。[H4-1，H45]		
CREW.3 解除自动制动	CREW.3a1： 在 TOGA 期间，机组不会解除自动制动（导致（重新）起飞时失去加速度）[H4-1，H4-2，H4-5]	CREW.3b1： 在着陆或 RTO 期间，机组解除了自动制动（当扰流板展开时导致自动制动操作失效。机组反应时间可能导致过冲）[H4-1，H4-5]	CREW.3c1： 在（a）飞机下降超过 TBD fps，（b）能见度小于 TBD ft，（c）等.....（导致飞机失控或（重新）起飞失去加速度）后，机组人员解除自动制动超过 TBD 秒的时间 [H4-1，H4-2，H4-5]	
CREW.4 关闭 BSCU	CREW.4a1： 如果 WBS 行为异常（需要启用备用制动模式），机组不会关闭 BSCU 电源[H4-1，H4-2，H4-5]	CREW.4b1： 当需要自动制作并且 WBS 正常运行时，机组关闭 BSCU [H4-1，H4-5] CREW.4b2： 需要（或即将使用）自动制动的 WBS 时，如果正常使用，机组会关闭 BSCU [H4-1，H4-5]	CREW.4c1： 机组太晚关闭 BSCU（TBD）以在 WBS 行为异常的情况下启用备用制动模式 [H4-1，H4-5] CREW.4c2： 在需要完成自动制动或防滑行为之前，机组过早关闭 BSCU [H4-1，H4-5]	N/A

		CREW.4b3： 当需要（或将需要）防滑功能并且 WBS 正常运行时，机组关闭 BSCU [H4-1，H4-5]		
CREW.5 打开 BSCU	CREW.5a1： 当使用正常制动模式、自动制动或防滑并且 WBS 正常运行时，机组不会启动 BSCU [H4-1，H4-5]		CREW.5c1： 在正常制动模式、自动制动或防滑之后，机组启动 BSCU 太晚[H4-1，H4-5]	N/A

表 C.4：飞机制动系统控制单元（BSCU）自动制动的 UCA

控制行为（BSCU）	**不提供导致危险的原因**	**提供导致危险的原因**	**顺序太早/太晚/不按顺序**	**停止太快/应用太久**
BSCU.1 自动制动的制动命令	BSCU.1a1： 自动制动在 RTO 到 V1 期间不提供制动命令（导致无法在可用的跑道长度内停止）[H4-1，H4-5] BSCU.1a2 当 BSCU 布防时，自动制动不会在着陆时提供制动命令（导致减速不足和潜在的过冲）[H4-1，H4-5] BCSU.1a4 自动制动在起飞后不提供制动命令（需要锁轮，导致在飞行中起落架收回或机轮旋转期间潜在的设备破坏）[H-6]	BSCU.1b1： 自动制动在着陆时提供过多的制动命令[H4-1，H45] BSCU.1b2 在起飞过程中，自动制动不恰当地提供制动命令（导致加速不足）[H4-1，H4-2，H4-5] BSCU1b3 自动制动提供的制动指令水平不足（导致着陆时减速不足）[H4-1，H45]	BSCU.1c1 自动制动在接地前提供制动命令（导致轮胎爆裂、失控、受伤、其他伤害）[H4-1，H4-5] BSCU.1c2 自动制动在接地后提供超过 TBD 秒的制动命令（导致减速不足和潜在的失控、过冲）[H4-1，H4-5] BSCU1c3 自动制动在机轮离开地面并且没有请求 RTO 之前的任何时候提供制动命令（在齿轮缩回之前可以对制动器施加制动）[H4-1，H42，H4-5] BCSU.1c4 自动制动在拒绝起飞期间，在 V1 后提供制动超过 TBD 秒（假设自动制动负责机组减速后 RTO 期间的制动）[H4-2]	BSCU.1d1 在 TBD 滑行速度达到之前，自动制动在着陆时停止提供制动（导致减速度降低）[H41，H4-5] BSCU.1d2 自动制动在着陆时提供制动命令太长（超过 TBD 秒）（导致停在跑道上）[H4-1] BSCU.1d3 自动制动提供轮胎锁定制动命令，直到接近前接地时间小于 TBD 秒（导致失控、设备损坏）[H4-1，H4-5]

附录 D：安全控制结构中包含的责任

下面的列表（来自《建设一个更安全的世界》，第 13 章）可用于创建新的安全控制结构，作为创建现有安全控制结构模型或启动活动以改进安全控制结构的人员的清单，以及可用于确定执行事故和事故分析时失当的控制和控制结构。它并不意味着详尽无遗，需要针对特定行业和安全计划进行补充。

此列表仅包含一般职责，但不表示应如何分配。将责任适当分配给组织机构中的特定人员和地点将取决于每个组织的管理结构。每项一般责任可分为多个个人责任，并在整个安全控制结构中分配，其中一组实际履行职责，其他组负责监督、领导或指导或监督活动。当然，每项责任都需要相关的权力和责任，以及实施责任所需的控制、反馈和沟通渠道。

通用管理

•在组织机构的各个层面提供领导、监督和安全管理。

•制定公司或组织机构的安全政策。建立评估安全关键决策和实施安全控制的标准。建立政策的分销渠道。建立反馈渠道，以确定员工是否理解、遵循以及是否有效。根据需要更新策略。

•建立公司或组织机构的安全标准，然后实施、更新和实行。设定开发和运营中安全工程的最低要求，并监督这些要求的实施，包括任何承包商活动。为危险运行设定最低物理和运行标准。

•制定事故征候和事故调查标准，确保建议得到落实和有效。使用反馈来改进标准。

•建立变更要求管理，以评估变更对安全的影响，包括安全控制结构的变化。审核安全控制结构，以了解计划外的变化和向更高风险状态的迁移。

•创建和监控组织机构的安全控制结构。为安全分配责任和权力。

•建立工作组。

•建立健全可靠的沟通渠道，以确保开发系统设计和运行流程的状态的准确管理风险意识。这些渠道应包括承包商活动。

•为安全相关的活动提供物质和人力资源。确保执行安全关键活动的人员具备适当的技能、知识和物质资源。

•创建一个易于使用的问题报告系统，然后进行监控以进行必要的更改和改进。

•为所有员工建立安全教育和培训，并建立反馈渠道，以确定其是否与持续改进流程一起奏效。教育应包括过去事故和原因的提醒以及经验教训和麻烦报告的输入。有效性评估可能包括从审计期间的知识评估中获得的信息。

•建立组织机构和管理结构，以确保与安全相关的技术决策独立于计划考虑因素，包括成本和进度。

•为安全相关的技术决策与计划考虑之间的冲突建立明确、透明和明确的解决程序。确保冲突解决程序正在使用并且有效。

•确保做出与安全相关的决策的经理充分了解并熟练掌握。建立机制，以允许并鼓励所有员工（包括一线运营商）和承包商为安全相关的决策做出贡献。

•建立安全相关的决策的评估和改进流程。

•创建和更新组织机构的安全信息系统。

•创建和更新安全管理计划。

•为员工和承包商建立沟通渠道、解决流程和裁决程序，以表达对系统或安全控制结构部分安全性不正常的投诉和担忧。评估报告问题时匿名的必要性。

开发

•为开发人员和开发经理提供安全导向设计和其他必要技能的特殊培训。随着事件的发生更新此培训，并从经验中学习更多。为培训创建反馈、评估和改进流程。

•创建并维护危险源日志。建立并维护文件和跟踪危险源及其状态。

•建立工作组。

•使用系统危险源和安全约束来设计系统的安全性。随着设计过程的进行，迭代并改进设计和安全约束。确保系统设计包括如何消除或减少造成或导致不安全运营人行为的背景因素的考虑，从而导致了系统危险源。分心、疲劳等是由设计引起的风险因素，这些因素取决于人类以设计师想象的方式表现，而不是像在正常情况下人类那样的表现。

•记录运行性假设、安全约束、安全相关的设计特征、运行假设、安全相关的运行限制、培训和运行说明、审核和性能评估要求、运行程序以及安全验证和分析结果。记录内容和原因，包括在安全约束和设计功能之间进行跟踪以强制执行它们。

•在需要做出安全相关的决策时，从早期决策开始到持续整个系统的生命周期，执行高质量和全面的危险源分析。确保将危险源分析结果及时传达给需要它们的人。建立允许向下、向上和侧向（即，在那些建筑子系统中）通信的通信结构。确保在设计发展和获得测试经验时更新危险源分析。

•培训工程师和管理人员在决策过程中使用危险源分析的结果。

•维护和使用危险源日志和从经验中获得的危险源分析。确保向参与开发的每个人传达与安全相关的要求和限制。

•收集运行中的经验教训（包括事故和事故征候报告），并使用它们来改进开发过程。利用运行经验来识别开发安全控制中的缺陷并实施改进。

运行

•制定运行安全管理计划

•为运营人和运行管理层开发特殊培训，以创造所需技能，并在事件发生时更新此培训，并从经验中学习更多。为此培训创建反馈、评估和改进流程。培训员工安全地完成工作，了解如何正确使用安全设备，并在应急情况下做出适当的响应。

•建立工作组。

•在运行期间维护和使用危险源日志和危险源分析，并获得经验。

•在危险运行期间，确保所有应急设备和安全装置始终可操作。在开始安全关键、非常规、潜在危险的运行之前，检查所有安全设备以确保其正常运行，包括警报测试。

•对任何运行异常情况进行深入调查，包括危险情况（例如水箱中含有对水有反应的化学物质的水）或事件。确定在启动或重新启动任何潜在危险运行之前它们发生的原因。提供进行此类调查所需的培训和适当的给管理层的反馈渠道。

•创建变更程序的管理并确保遵循这些程序。这些程序应包括对所有拟议变更和批准与安全关键运行相关的所有变更的危险源分析。制定并实施有关禁用安全关键设备的政策。

•使用危险源分析结果作为运行和维护的先决条件，执行安全审核、性能评估和检查。收集数据以确保遵守安全政策和程序的正常，并确保有关安全的教育和培训有效。为风险增加的领先指标建立反馈渠道。

•使用在开发过程中创建的和传递给运行的危险源分析和文档，以确定向高风险状态迁移的领先指标。建立反馈渠道以检测领先指标并作出适当回应。

•建立从运行到开发的沟通渠道，以传递有关运行经验的信息。

•进行深入的事故征候和事故调查，包括所有的系统因素。分配实施所有建议的责任。跟进以确定建议是否得到充分实施和有效。

•对安全关键的活动进行独立检查，以确保其正确完成。

•确定已确定的安全关键项目的维护优先级。实施维护计划。

•创建和实施有关禁用安全关键设备和更改物理系统的策略。

•创建并执行特殊过程，以便在先前关闭的单元中或在维护活动之后启动。

•调查并减少虚假警报的频率。

•清楚地标记故障的报警和仪表。通常，建立将所有当前故障设备的信息传达给运行人的程序并确保遵守这些程序。消除报告故障设备的所有障碍。

•为所有安全关键设备和报警程序定义并传达安全运行限制。确保运营人了解这

些限制。确保运营人在遵守限制和应急程序时获得奖励，即使事实证明不存在应急情况。根据需要随时调整运行限制和警报程序。

•确保备用的安全关键物品有库存或可以快速获得。

•为工厂管理层建立与安全相关的事件和活动的沟通渠道。确保管理层具备做出安全运行决策所需的信息和风险意识。

•确保应急设备和响应可用并可用于治疗受伤工人。

•建立与社区的沟通渠道，以提供有关危险源和必要的应急行动和应急响应要求的信息。

附录 E：软件密集型系统的分析分解的局限性（航空电子学示例)

确保系统中的属性的经典方法应用分析分解，即，分析各个组件并将结果组合以确保该属性作为一个整体存在于系统中。这种方法适用于组件之间的耦合和相互作用相对简单的系统，这种假设主要适用于机电系统.[36]对于软件密集型的系统，这种方法的独立性和可还原性的假设（例如，组件不对彼此产生间接或意外的影响）不再成立。飞机制造业正开始发现这一事实。Bartley 和 Lingberg[37]的一篇论文描述了这个问题并提供了飞行中飞机出现这种问题的例子。他们在集成模块化航空电子设备（IMA）系统的背景下讨论该问题，该系统是一组飞机航空电子设备功能，其可由不同的制造商创建并随时间单独更新。

为了允许 IMA 系统中共享信息，通常在接口控制文档（ICD）中指定接口以用于电交互、机械或物理交互以及软件模块交互。典型的软件接口规范包括诸如电子信号特征之类的东西；数据传输格式、编码、时序和更新要求；数据和数据元素定义、消息结构和流程、事件的运行顺序；和错误检测和恢复程序。ICD 没有明确规定实际生成这些变量应该返回到接口。

使用 ICD 的假设是组件之间的数据和控制耦合被最小化，并且如果对一个功能的改变不需要改变 ICD，那么使用这些参数的功能不会受到这一改变的影响。这个假设是错误的，如同 Bartley 和 Lingberg 提供的涉及襟翼控制的例子所表明的那样。

图 E.1 显示了襟翼系统控制器之间的接口，其中“襟翼扩展”的离散变量在所谓独立的航空电子功能之间传递。如果没有进一步襟翼扩展的确切技术含义的定义，襟翼扩展离散变量的真实状态可能意味着以下任何一种：

•在 1 或更大的襟翼制动装置位置时，检测到左右后缘（TE）襟翼表面[38]。

•在“收起（向上）”襟翼制动装置的过程中，未检测到左右后缘襟翼表面。

•在 1 或更大的襟翼手柄制动装置位置时，检测到襟翼杆手柄。

•在“收起（向上）”襟翼手柄制动装置的过程中，未检测到襟翼杆手柄。

所有这四种可能性具有彼此不同的特征和不同的接收功能，其可以以不同的方式使用襟翼扩展离散量。例如，一旦后缘襟翼开始移动，确定“襟翼不在向上位置”的逻辑几乎会立即得到满足。相反，襟翼可能需要 5 到 10 秒才能完全到达“在‘1’制动器中检测到的襟翼”位置。此外，如果由于液压系统故障而使襟翼表面不响应

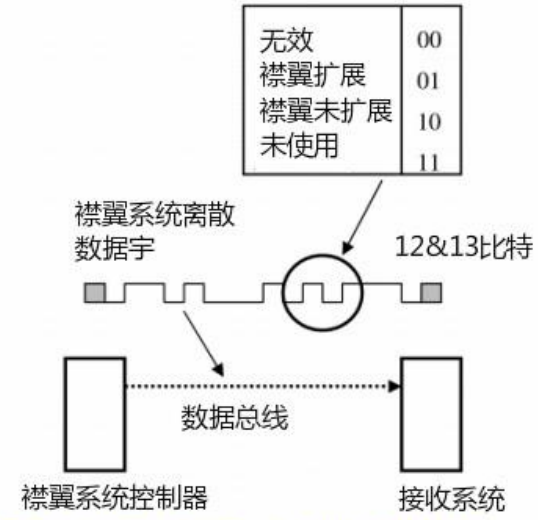
---

[36] 即使对于这些系统，当组件之间的细微间接交互扭曲了组合的组件交互时，组合组件分析的结果也可能是不正确的。

[37] Gregg Bartley 和 Barbara Lingberg，集成模块化航空电子设备（IMA）系统认证问题，联邦航空管理局，华盛顿特区。

[38] 制动器是用于机械地抵抗或阻止车轮，车轴或主轴的旋转的装置。

有效命令，则一旦杆从上制动器中移出，襟翼杆位置将不再反映襟翼表面的真实位置。显然，任何使用该特定信号的功能的设计者都需要理解如何计算信号。诸如所描述的那些差异对于使用该系统可能是至关重要的。



图E.1：襟翼扩展（Ext）离散数据传输[3]。

如果仅通过数据总线传递图 E.1 中的信息，则襟翼系统控制器逻辑的改变可能会对使用襟翼扩展变量的功能行为产生负面影响。因此，该信号接收系统开发人员将无法得知变更，也不会理解系统级影响。例如，在各个应用程序的设计已经“最终确定”之后，稍后在开发周期中发现的问题可能导致襟翼系统控制器的逻辑发生变化。

使用襟翼警报逻辑解决此问题的一种可能方法是使用襟翼杆位置而不是实际的襟翼表面位置来计算襟翼扩展变量。襟翼扩展变量的物理意义已经改变。

根据 Bartley 和 Lingberg 的说法：

所示的一点是，它不能留给正在更新的功能（在这个例子中，襟翼系统）的设计者和系统专家来确定该改变对下游用户的影响。评估变更影响所需的专家是使用系统，而不是源系统的设计者和分析师。即使源功能的 ICD 没有改变，此示例显然也需要跨越功能边界的更改影响分析。此外，这个例子说明了为什么健壮的分区不能完全将驻留在一个分区中的功能与对另一个分区中的功能的更改隔离开来。

传统的危险分析技术（如 FMEA 和故障树分析）侧重于单个组件故障或故障模式。这种对组件可靠性的关注假设鲁棒分割和接口控制是有效的，并且组件不直接或间接地交互。但是，如示例所示，这些假设通常不适用于复杂系统。

可以使用 STPA 在理论上解释和解决该问题。

附录 F：基本工程和系统工程概念（适用于非工程师）

在审阅这本手册的初稿时，我们发现我们假设了一些并非所有读者都具备的工程背景：STPA 的用户不一定受过工程方面的培训。本附录介绍了一些使用本手册的用户可能不熟悉的基本工程概念。

我决定将本附录进行独立撰写，以便读者能够选择自己感兴趣的部分。在撰写文章内容时，没有假设读者有任何特定的教育背景，本文内容包括：

- 什么是系统？
- 系统工程的基本概念
- 工程中“control”的概念
- 系统理论和复杂度理论

什么是系统?

这本手册的所有内容中的最基本的概念是一个系统。

系统： 一组事物(称为系统组件)，作为一个整体共同作用，以达到某种共同的目标、或目的

系统的一些定义忽略了“目标”或“目的”，本质上说，系统是一组相互关联的事物或组合形成的一个整体。该定义没有使用术语“系统”常用的表达方式。一只鞋、一颗星和一台机车是系统的一组事物或潜在部分，但大多数人通常不会认为这是一个系统。可以把这些看作是一个系统，只要设想出一个目标将这些单独的事物放在一起考虑，目标是这一概念的基础。

系统的其他定义指出系统的组件必须相互依赖、连接或相互作用，但这些条件没有一个是拥有某一系统必不可少的。并以排除通常被认为是系统的事物的方式这一约束定义。系统组件可以直接或间接地相互连接，后者包括只涉及系统用途的连接和各种非线性相互依赖关系。

该定义表明系统主要以目标或目的为基础。但是定义和查看系统的人可能有不同目的。以机场为例，对旅客来说，使用机场的目的可能是提供到其他地方的航空运输服务。对地方或州政府来说，机场可能是增加机场地区政府收入和经济活动的一种手段。对航空公司而言，机场的目的可能是接收和卸载乘客和货物。对机场的企业来说，目的是吸引他们可以销售产品和提供服务的客户。因此，我们在谈论系统时，始终需要指定正在考虑的系统的目的。

系统是一种抽象的，即观察者构思的模型。

观察者可能会看到与设计者不同的系统目的，或者关注不同的相关属性。系统的规范，包括系统的目的在系统工程中至关重要。它们确保了在设计，使用或查看

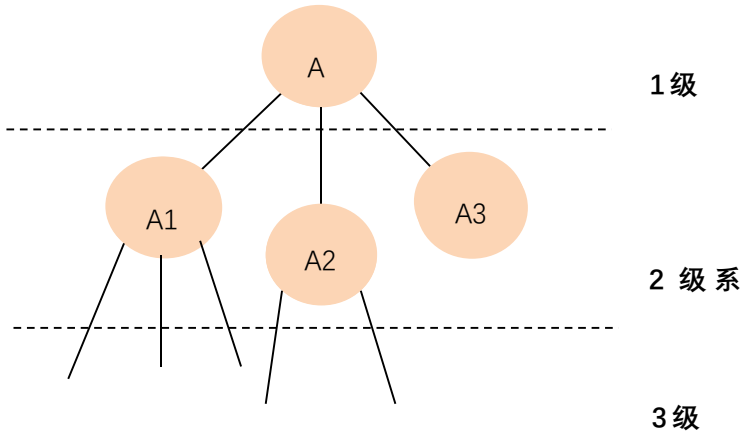
系统的过程中心智模型的一致性，并增强了沟通。请注意，机场的不同组件可能包含在特定的“机场”系统中，例如乘客登机柜台，通往飞机的廊桥以及航空公司视图中的机场滑行道与商业视图中的商店和客户。需要注意的是，这些是人类思想在实际物理世界中的模型或抽象表现。在任何“机场系统”或子系统中考虑的组件以及它们在整个系统中扮演的角色对于机场系统或机场子系统的每个概念（模型）可能是不同的。有关此主题的更多信息，请参阅下一节有关系统工程中规范重要性的部分。这里的基本想法是，所考虑的系统的目的或目标必须由建模和分析特定系统的人指定和同意，并且系统的这些方面是观察者施加给现实世界对象的抽象或模型。

工程（人造）系统的目的通常在创建系统之前指定，尽管系统的观察者可能对目的的解释不同于设计者的最初预期。自然系统中的行为可能被系统的观察者认为是有目的的。

系统的概念有一些基本假设：

- 可以定义系统目标
- 系统是可以再分的，也就是说，它们可以被分成具有交互行为的组件或组件之间的关系，无论这些交互是直接的还是间接的。

系统本身可以是更大系统的一部分，也可以分成子系统（被视为整个系统的组件）。请注意，这里用的定义方式是数学和计算机科学中所谓的“递归”。也就是说，定义是根据其自身来做的。系统通常是较大系统的一部分，可以分成子系统。图 F.1 显示了标记为 A 的系统（例如“机场”系统）和三个子系统的典型抽象层次结构，这三个子系统本身又可以被视为系统。



图F.1：抽象系统A 被视为由三个子系统组成。每个子系统本身也是一个系统。

图 F.1 不是连接图。相反，它是一个分层抽象，显示了不同抽象或建模层次的系统的不同视图。图 F.2 给出了 A 本身被视为较大系统 AB 中的一部分的一个抽象。

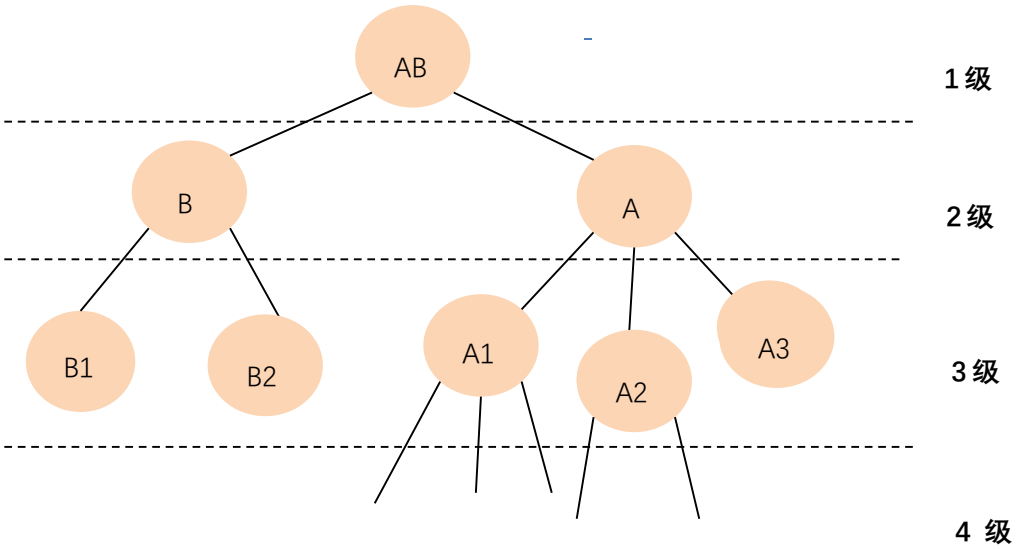
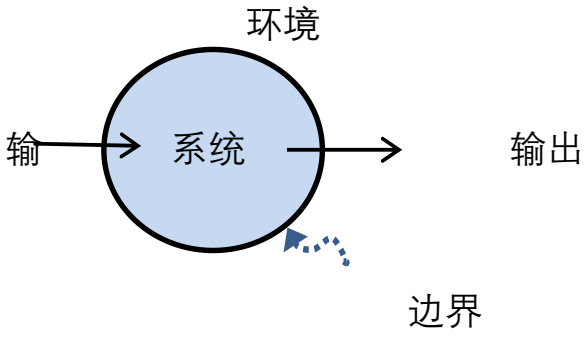


图 F.2：系统 A 可以被视为较大系统 AB 的组件（子系统）

系统定义中的递归性质很重要，因为许多人认为“系统的系统”必须与系统区别对待。实际上，相同的通用系统工程和分析方法和技术适用于所有系统。根据系统理论中的系统定义“系统的系统”也只是一个“系统”，这一概念也适用于工程中。

系统是有状态的。状态是一组可以随时用来描述系统的相关属性。航空运输系统中机场状态的一些属性可以是特定登机口的乘客数量，飞机所在的位置以及他们正在做的事情（装载乘客，滑行，起飞或着陆）。相关的状态组件取决于系统与其环境之间的边界具体是怎样的。

环境通常定义为不属于系统但其行为可能影响系统状态的组件集（及其属性）。因此，系统在特定时间具有状态，并且环境也具有状态。环境的概念表明系统与其环境之间存在界线。同样，这个概念是由系统的观察者创建的抽象概念，不需要是物理边界。什么是系统的一部分或什么是环境的一部分将取决于当时的特定系统及其使用方式。



系统输入和输出跨越系统边界。该模型通常称为*开放*系统。还有一种封闭系统的概念（没有输入或输出），但这个概念与我们在系统安全中最关心的工程系统没多大关系。

什么是系统工程?

系统工程是指将结构放入系统的设计和构造中以改进工程工作结果的努力。对于相对简单的系统，可能不需要系统工程。系统工程在第二次世界大战后开始形成，当时我们开始构建更复杂的系统，特别是导弹和其他防御系统，并发现非正式的设计和施工过程经常导致无法满足系统设想目标的系统和对其潜在行为的限制，即限制它实现目标。缺乏结构化流程也导致项目的推迟及超出预算。

系统工程是一门大学科，但理解本手册只需要两个概念。第一是系统工程阶段和过程的概念。这些阶段的最基本模型如图 F.3 所示。

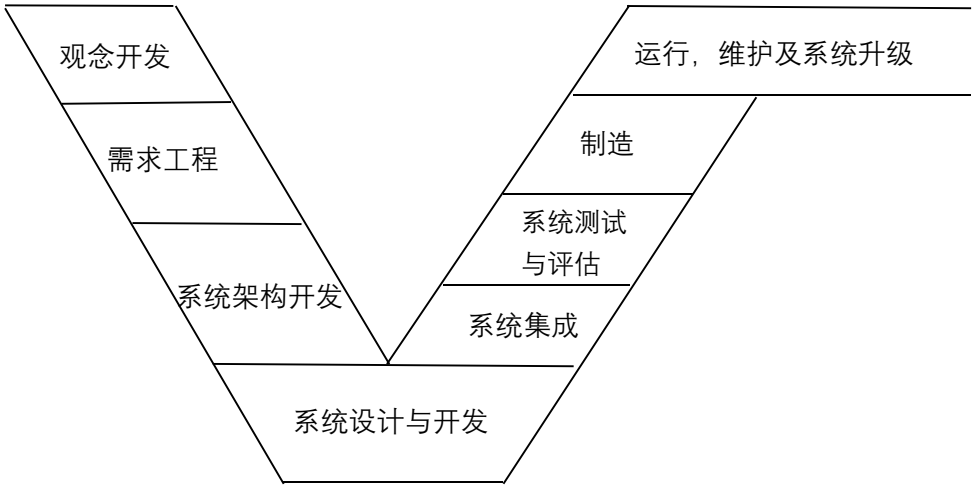


图 F.3：基本系统工程"V 模型"

该模型被称为 V 模型，尽管可以在各种组件上放置不同的标签。有时阶段是直线绘制的（如果线条向下倾斜，则称为"瀑布"模型），但除了绘制方式之外没有区别。其基本想法为如果在首先开发了系统概念，并定义了基本目标和约束的场合中使用结构化流程，则会产生最佳结果。在开发的早期阶段可能会进行各种可行性和其他分析，以阻止该过程沿着需要以后进行重新回折的道路走。需求工程通常是该过程的下一步，其中考虑到先前商定的基本目标和约束，为系统开发提供详细要求。系统架构开发意味着在详细设计和开发之前创建系统的基本架构或高级设计。"V"的左侧代表基本设计过程。

V 的右侧显示了评估和制造发生的后期发展阶段。有时包括系统的运行（后来称为生命周期模型）——如本手册的图 F.3 和第 3 章所示，因为安全分析需要在系统的整个生命周期中持续进行。在 V 模型过程的右侧，集成了设计和构造的各种组件并经历了测试和评估阶段。然后对之进行制造和使用。

有目的地简化上图，以反映整个过程的组件以减少混乱。这些阶段周围总是有

很多弹性因素，而不仅仅是一个直线过程。例如，开始时并没有完整地指定要求，因为设计过程本身将更好地理解要解决的问题，从而添加或更改原始要求和约束。开发过程中的严重问题有时可能需要项目完全回到早期阶段。但通常遵循这种循序渐进的过程。

此外，这个模型不应该太按字面意思理解。阶段标记并不意味着不存在并行性，事实上某些并行性是很常见的。例如，被动等待所有测试的进行，可能到最后会给项目带来严重风险。各种测试可以从最早的概念阶段开始（可能以模拟或原型设计的形式），并且通常持续到每个阶段。而且，根据所设计的系统类型，我们可以将各种标签应用于各个阶段。目标只是定义一个结构化流程，该流程将根据特定项目的各种需求进行定制。第 3 章介绍了 STPA 通常如何应用于工程化某一复杂化系统所涉及的各个阶段和活动。

V 模型或瀑布模型的另一个方面是文档驱动。根据我的经验，成功设计复杂系统的关键在于规范化。通常有大量人员参与这样的系统（有时会有成千上万），并且开发和运行可能发生在很长的时间框架内。规范是很必要的，这样可以确保结果满足用户的需求，并且决策不会因为沟通存在问题而陷入混乱。

规范从最早的开发阶段开始。本附录的前一部分将系统定义为抽象的，其目的是由系统的查看者强加的。为了确保每个人都同意所创建系统的基本目标，规范必须至少指定系统目的（目标），与兴趣或期望的相关系统属性，以及对如何实现目标的任何限制。此外，必须识别系统边界（即什么是在设计者的边界和控制范围内的，什么是设计者所假设的而不在其控制之下的部分环境）。初始规范的其他方面包括输入和输出，物理或逻辑组件，结构，组件之间的相关交互以及它们的行为如何影响整个系统状态。随着开发的进行，初始规范将得到完善并且细节得到开发。

可追溯性是系统工程中另一个重要的概念。非常庞大和复杂的系统在完成时可能会有数千页的详细规格。表明系统各部分之间的关系以及特定决策是基于开发过程的哪些假设或先前结果是创建和运行系统的要求。例如，在 STPA 中，不安全的控制操作可追溯到系统级危险和并且可以查到它们进行不安全控制操作会有什么样的情景。当需要进行更改时，可以确定需要更改的具体内容而无需从头开始。可追溯性也是记录决策制定基础的一种方式。

工程中的“控制”概念

控制的概念在系统的概念，设计和运行中十分重要。可以想象，系统可以在不对其行为进行任何类型的控制的情况下运行，但是难以确保系统满足它的目标，而同时限制其实现那些目标的方式，反而不会产生不利的，不想要的后果。

控制是工程学的基础，尤其是在复杂的系统中。工程师们不是在处理有生命的

东西，而是处理人造物品——汽车，水坝，飞机，化工厂，航天器，洗衣机，机器人——这些都是为了实现某些目的而建造的。如果它们不是一动不动的（如桥），则需要操作或控制它们的运行，即它们进入的状态。

非工程师有时会对“控制”一词给出与工程中使用的方式不同的解释。控制不是通过某种类型的军国主义权力来强制个人遵守指定的角色和程序。相反，控制在这里得到了广泛意义上的使用，可以通过设计控制，如联锁和故障安全设计，通过过程控制，如维护和制造程序，甚至社会结构，如文化，激励结构和个人自我利益来实现。如果不实施某种形式的控制，就无法确保实现系统目标和产生约束。

控制器一般设计较为简单，这样可以提高安全性并且不会增加复杂性和成本，但这通常需要在设计过程的早期考虑安全性因素。通过设计提高安全性的最有效方法是使用消除危险状态的设计。其次，如果它们确实会有危险，那么设计要将危险控制在较低的状态并且易于处理。通过最小化由此造成的损害来对危险发生做出反应的设计显然是我们最不希望的危害控制形式，但通常必须包括这种控制。

因此，控制回路仅仅是一种控制形式。《设计一个更安全的世界》(“*Engineering a Safer World*”)中强调了这一点，因为我之前的书《*Safeware*》中广泛涵盖了基本安全设计的其他方面。此外，系统理论强调控制回路或主动控制的概念。本附录中不可能教授用于设计安全的技术，但是会介绍一些基本概念，例如被动控制和主动控制之间的区别，可能有助于理解一般安全设计技术。

有三种类型的设计技术可以“控制”安全性。第一种是使用本质或固有的安全设计。本质安全设计是指不能产生所担心的危险的设计。例如，系统设计可能没有足够的能量来引起爆炸。如果所担心的是使用潜在的有毒物质，可能会用无毒的替代品。如果存在通过通信网络丢失信息的可能性，则可以使用更加直接的通信。

如果无法实现本质安全设计，则可以使用被动控制机制。被动控制不需要积极的行动而生效，而是依赖于基本的物理原理，例如重力。在没有任何明显控制行为的情况下防止系统进入危险状态的物理联锁就是被动控制机制。例如：

- 压力敏感垫或光幕：当有人靠近时，它会关闭机器人或有危险机器的电源。
- 一种确保阀门关闭和开启的正确顺序，或确保两个阀门不能同时打开或关闭的物理设备。
- 汽车发动机冷却系统中的冷冻塞：如果冷却系统中的水冻结，膨胀将使塞子脱落而不是使汽缸破裂。
- 锅炉中的易熔塞：如果有过热现象并且水量下降到预定水位以下则会暴露在外。在这种情况下，塞子熔化并且形成开口允许蒸汽逸出，这降低了锅炉中的压力并防止爆炸发生。
- 速度调节器不允许超过特定限制的速度，或安全阀会将压力保持在危险水以下。

通常设计被动控制以使系统基本上进入防故障状态。实际上，飞机安全的基本方法是将系统构建为故障安全的。例如：

- 旧铁路信号量使用重量来确保如果电缆（控制信号量）断开，控制杆将自动落入 STOP 位置
- 通过气压将空气制动器保持在关闭位置。如果制动管路断裂，则会失去气压并自动施加制动。
- 死人开关（一种安全开关）
- 通过磁铁将压载物固定到特定位置的深海望远镜。如果电力丢失，镇流器将被释放，并且深海望远镜会上升到水平面位置。
- 故障影响的设计（例如当飞机发动机发生故障时风扇叶片被弹出），以便相邻部件不受影响。

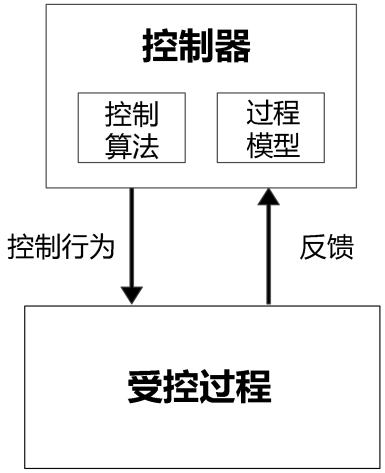
相反，主动控制需要检测和纠正危险情况。今天的主动控制通常涉及使用计算机和本手册中描述和使用的基本控制回路。请注意，主动控件可能出现更多问题，例如，可能无法检测到故障或危险状态，或者可能检测到故障或危险状态但未进行纠正，或者可能无法及时纠正以防止损失。被动控制更加可靠，因为它们通常依赖于物理原理和更简单的设计，而主动控制依赖于不太可靠的检测和恢复机制以及通常更为复杂的设计。然而，被动控制在设计自由度方面往往更具限制性，并且在复杂系统中实施并不总是可行的。

主动控制通常涉及控制回路。

什么是控制回路?

控制系统使用控制回路命令，指导或调节其他设备的行为或过程。这种控制系统的范围从使用恒温器的单个家用加热控制器到控制锅炉到用于控制复杂过程和机械的大型工业控制系统。

一个非常基本的控制回路如下所示（本手册的其他部分也有这种控制回路）：



为了控制过程，控制器必须具有一个或多个目标，可以包括对受控过程的行为的持续约束。另外，控制器必须具有某种方式来影响受控过程的行为，即受控过程或系统的状态。 这些是图中标记的控制行为。在工程中，控制行为由执行器实现。为了知道必要的控制行为，控制器必须能够确定系统的当前状态。在工程术语中，有关系统状态的信息由传感器提供，称为反馈。最后，控制器必须包含一些受控过程状态模型，如手册中的其他地方所述。

例如，简单的恒温器以将温度保持在特定范围内为目标。反馈向控制器提供关于受控过程的当前状态的信息，在这种情况下是指房间中的空气温度。反馈用于更新控制器的过程模型。其他信息也可以是过程模型的一部分，例如环境信息（外部空气温度）或关于自然温度波动的基本信息。控制器使用过程模型来决定进行什么控制行为，例如，施加热空气或冷空气，以便改变室温（受控过程）状态，以使之保持在所需范围内。

控制回路有两种通用运行模式：反馈控制和前馈控制。反馈控制是上面所述的内容。驾驶时，驾驶员可以读取车速表（反馈）来决定制动或踩油门以使汽车的速度保持在所需的水平。

当驾驶员接近山坡时，则会产生前馈控制的情况。驾驶员可能等到汽车开始减速然后踩下加速器，但更有可能的情况是经验丰富的驾驶员预计在上坡时需要加速以保持恒定速度并且会在汽车开始减速前增加加速度。因此，在前馈控制中，控制器使用过程的当前状态（在这种情况下是车速）和未来情况（在倾斜上操作）的模型，然后提供控制行为，无需特定的反馈来识别这要需要。

反馈和前馈控制经常会串联使用。在我们的所举的例子中，司机可能预计需要在接收反馈之前使用前馈控制踩油门，但反馈可用于调整前馈控制行为，以便如果前馈计算不正确，则在斜坡上保持恒定速度。

在本手册中关于领先指标的章节中，塑造行为被描述为通过设计控制以维持假设情况，防止危害（受控过程中的不安全状态）以及控制过程向高风险状态迁移来维持安全状态的尝试。塑造行为提供了一种前馈控制。使用的控制类型可以是被动的或是主动的。对冲或应急行动提供了一种反馈控制，因为它们涉及监控系统风险增加的迹象并给出相应的响应。套期保值行为包括监控塑造行为的有效性，从而既提供前馈控制又提供反馈控制。

控制的工程概念（基于系统理论）也可以应用于社会和组织系统，并已广泛用于管理理论和社会科学。

系统理论与复杂性理论

STAMP 基于系统理论，而不是复杂性理论。许多人已经对系统理论进行了大量研究，但奥地利生物学家路德维希·冯·贝塔朗菲被认为是他称之为一般系统理论的创始人之一。诺伯特·韦纳探讨了数学和工程学的含义。维纳称他的理论是控制论，但随着时间的推移，这个术语逐渐消失，而贝塔朗菲被的术语在今天被广泛使用。

正如本手册中以及其他地方所见，系统理论是一套原则，可用于理解复杂系统的行为，无论它们是自然系统还是人造系统。系统思维是一个术语，通常用于描述人们在应用系统理论原理时所做的事情。本手册第 1 章就简要介绍了这些原则。

复杂性理论源于 20 世纪 60 年代的系统理论和其他概念，通常与圣菲研究所和在那里工作的研究人员有关。系统理论和复杂性理论之间存在一些共性，两者都包括涌现和从整体上关注复杂系统行为，而不是减少或分解成组件。系统理论的基本组成部分是涌现，层次，沟通和控制，这些也包含在复杂性理论中。系统理论和复杂性理论都包括反馈和前馈控制，适应性，非线性相互作用和约束的概念。拒绝简化论或分解是理解系统行为的原则。

但是，二者也存在显着差异。创建复杂性理论来描述自然系统，其中看似独立的机构使用我们尚未完全理解的自然法则自发地将自己排序并重新排序成一个连贯的系统。相反，系统理论更适合人造和设计的系统，其中系统是由人类使用某些工程或设计过程故意创建的，并且基本设计是已知的并且变化是受控的。

另一个主要区别是系统理论将所有系统视为显示涌现行为的，而复杂性理论将系统划分为四种类型：简单，复杂，错杂和混乱，每种都伴随有不同程度或类型的涌现。

因此，系统理论似乎最适合工程化或设计的系统，而复杂性理论最适合不知道设计的自然系统，如天气，以及社区，如社区，非设计和 缺乏秩序，很难或不可能预测涌现行为。 复杂性理论处理的是无法设计的行为，反而必须在它出现时进行体验和研究。

系统理论，至少对我来说，更容易理解和使用。 复杂性理论框架可能难以应用和理解，并且对于与提高安全性和其他涌现系统属性相关的工程目标而言可能过于复杂。 因此，通常会选择系统理论作为 STAMP 的基础。

附录 G：辅助致因场景生成的控制模型

STPA 的目标是确定可能导致危险状态的致因场景。《设计一个更安全的世界》中提供了一个模型（见下图 G.1），该模型显示了可能产生危险状态的一些一般原因。提供此模型的目的是帮助生成方案。然而，该模型非常抽象，并且遗漏了在场景生成中可能有用的细节。该模型还将自动控制器和人工控制器组合在一个标有“控制器”的通用框中，这是另一个省略重要差异的抽象表现。之后，在本书的第 9 章中，创建了另一个模型来描述设计技术。这种模式也过于笼统。自该书出版以来，还生成了其他更详细的人为控制器模型。在本附录中，我们描述了一个更完整的模型，并希望对那些确定致因场景更有用。

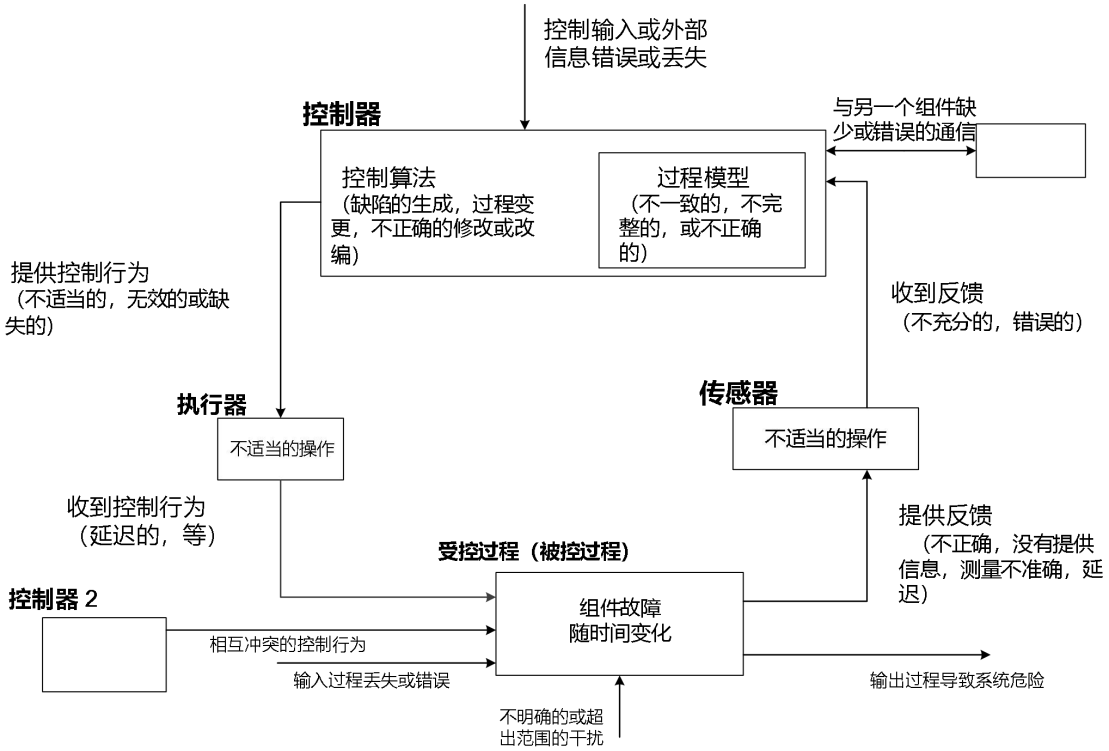


图 G.1：之前协助致因场景生成的通用模型

致因场景生成的抽象模型

新模型如图 G.2 所示。然后我会介绍模型的各部分以及它们如何用于创建场景。有太多潜在的致因因素，将它们全部包含在一般模型图本身中，因此将对它们进行单独描述。使用此模型作为指南来生成致因场景时，应从应用程序（系统）的特定控制模型开始。下面的通用组件将对应模型的各个部分。通用指南可用于帮助您为系统生成特定的致因场景。

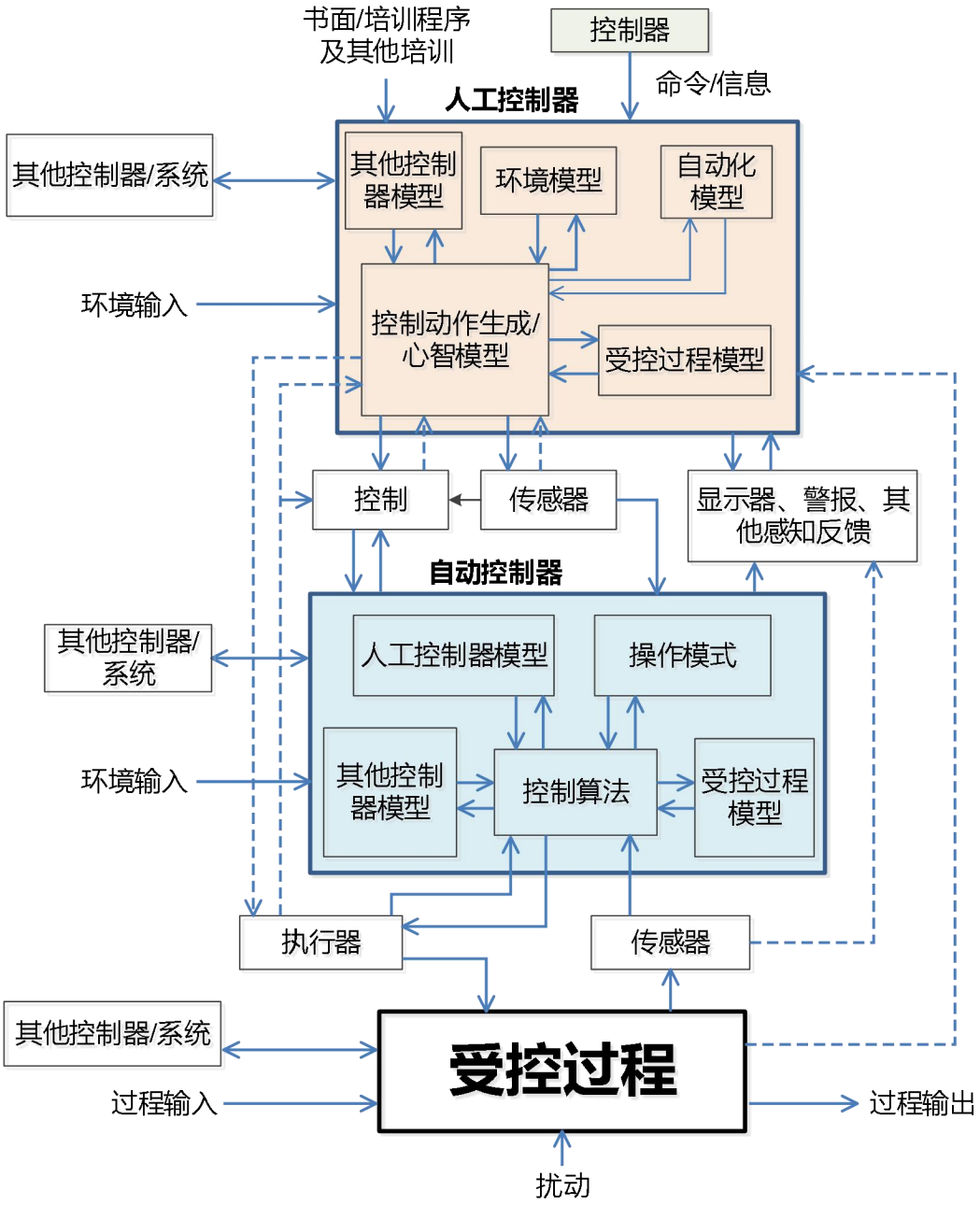


图 G-2：STPA 致因分析情景生成的新通用控制模型

受控过程

危险由图片底部的受控过程的状态来定义，例如，飞机的姿态、汽车的速度、机器人的位置。状态由组件或变量组成。由于 STPA 的目标是确认受控过程进入危险状态的过程，因此我们需要了解受控过程改变状态的方式。任何可以改变该状态的东西都可能成为危险状态致因场景的一部分。

受控过程组件随时间发生的失效或降级

受控过程中硬件发生故障可能导致其进入危险状态。硬件随时间发生变化（例如腐蚀），可能使其运行也发生改变。诸如硬件互锁之类的控制也可能失效或降级从而导致危险状态。

请记住，我们的目标不是做一个 FMEA，查看所有可能的失效状况及其结果，而是从危险开始，确定受控过程中的哪些条件可能导致危险状态。举个例子，我们假设致因分析所关注的危险状态是失去飞机的俯仰控制，比如缝翼、襟翼、副翼和升降舵的失效可能导致危险发生。如果危险指的是缺乏有效制动（减速），这可能是由于液压不足（泵故障、液压泄漏等）所导致的。

外部干扰

对过程的干扰可能导致危险状态，例如闪电影响飞机的电气系统。或者一只鸟被吸入发动机会影响飞机的动力（推力）。国家空域可能被天气状况干扰，加剧当下在该空域内单个飞机的危险状态。环境可能干扰受控过程的执行，例如由于潮湿跑道（例如车轮水上飞机）导致的制动不充分。

受控过程直接输入

对过程的直接输入可能会影响其状态。对于飞机，装载和卸载乘客或货物会改变其重量，从而影响飞机的可控性。对于国家空域，输入是进入空域的飞机，输出是离开空域的飞机。如果受控过程危险状态的组成部分受到任何输入（或输出）的影响，则必须在致因场景生成中考虑这些过程输入和输出。

受控过程控制器

显然，一个有危险过程的控制器可能会导致危险状态。模型将主要控制器显示在蓝色和棕色框中（图 G.2）。在飞机中可能是人工控制器飞行员和自动飞行控制系统。请注意，该模型并不意味着飞行员本身在飞机上。图 G.2 中的模型显示了另一个框（在受控过程的左侧），表示可能影响受控过程或其组件状态的其他控制器或系统。例如，维护活动或缺乏维护活动可能导致物理系统中的危险。其他自动和人工控制器的影响将在下面单独描述。

自动控制器

图中只显示了一个自动控制器（图 G.2 蓝色框），尽管可能有几个这样的自动控制器，它们可能采用分层结构设计。由于自动控制器和人工控制器的模型总是具

有相同的组件，因此此处仅显示一个。

除非在非常简单的系统中，自动控制器为执行器提供控制行为，改变受控过程状态。传感器提供当前过程状态的反馈。

通过执行器从自动控制器到受控过程的控制路径

如本手册第 2 章所述，该控制路径将控制行为从自动控制器转移到受控过程。控制路径可能由一个简单的执行器构成，可能包括一系列执行器，或者它可以通过具有开关、路由器、卫星或其他设备的复杂网络传输控制行为。无论如何实施，当防止危险所需的控制行为未被执行、执行不当或被延迟直至会产生危险时，该控制路径上的问题可导致危险的受控系统状态。比如执行器故障、沿控制路径的传输问题，或发送或执行控制行为的延迟。对于与安全漏洞相关的危险，攻击者可以阻止为防止危害而采取的控制行为，或者在受控过程中注入导致危险的控制行为。

从受控过程到通过传感器的自动控制器的反馈路径

与执行器控制路径一样，反馈路径中的缺陷可能源于传输问题，包括延迟、测量缺陷和不准确性、传感器故障或与传感器设计或运行相关的其他缺陷。同样，对手可能会对反馈路径施加负面影响。如果自动控制器有缺陷的过程模型会导致不安全的控制行为，那么应该在生成致因场景时考虑反馈路径。例如，数据可能被延迟直至不再反映当前的状态，控制器对不正确的信息起作用。在生成致因分析中不要因“有缺陷的过程模型”而停止。它没有为您提供足够的信息以提供针对不安全控制行为的保护措施。相反，您需要确定过程模型可能存在缺陷的原因。

自动控制算法

自动控制算法有两个主要功能：（1）生成控制行为;（2）维护关于受控过程状态的准确信息（模型）以及可能影响控制行为生成的外部系统组件和环境。这些功能中的任何一个产生缺陷都可能导致 UCA。

1. 生成控制行为

由于控制算法本身的缺陷，从外部源提供给自动控制器的不安全命令，或者自动控制器具有对受控过程及其环境的模型的缺陷，自动控制算法可能产生不安全的控制行为。后者的原因将在下一节中介绍。

STPA 为自动控制算法生成安全要求和约束。当然，这些必须经过验证。因此，这里的因果因素是对运行安全要求和控制算法约束的不适当的通信和验证。

2. 维护与 UCA 相关的情景因素的内部模型

即使实际的自动算法满足安全要求和约束条件，自动化控制器用于做出这些决

策的信息也会影响对执行什么控制行为进行决策的安全性。自动控制器在该决策中使用四种类型的信息：（1）受控过程的状态，（2）受控过程的其他控制器的状态，（3）自动化的操作模式，以及（4）人工控制器的状态（在一些复杂的新系统中）。这些模型中的各种类型的缺陷可能导致自动控制器不安全的控制行为。UCA 的场景将包含有关模型中哪些缺陷可导致 UCA 的信息。例如，BSCU Autobrake 在正常起飞期间提供制动控制行为。然后，当 BSCU 混淆了受控过程的状态时，即当它给出制动动作命令却不知道飞机处于正常起飞状态时，可以发生该 UCA。致因场景将包括 BSCU Autobrake 混淆了飞机状态的原因以及在这些条件下可能给出制动命令的原因。

受控过程模型

受控过程的模型是自动控制认为受控过程所处的状态。该模型可以包括受控过程运行所处环境的信息。模型由控制器更新，我们将其定义为控制算法的一部分（如果它是分开的则无关紧要[39]）。自动控制算法从用于测量受控过程变量的传感器获取有关该状态的直接信息。也可能存在这样的设计，使得过程模型在不经过控制算法的情况下自动更新。比如是在航天飞机发射之前加载特定的飞行数据。导弹系统也这样做。 《设计一个更安全的世界》的附录 B 详述了由软件飞行数据加载中的错误（拼写错误）引起的代价惨重的事故。

如果将控制算法分解为多个部分(如常规制动和自动制动的处理)，则由于控制过程模型之间未知或未经考虑的相互作用，更新过程模型的方法可能会有各种各样的缺陷。在确定导致 UCAs 的场景时，应该考虑这些情况。

在更新受控过程模型时，控制算法可以接收关于受控过程的状态的间接信息或者对其进行假设。一个常见的假设是，当控制算法发出控制行为时，它可以假设执行动作并更新其过程模型。例如，飞机上的电子飞行控制系统发出用于执行器的控制命令以使电梯在移动一定程度。然后它可以假设电梯已经移动并且它已经移动了指定的量。被命令的运动没有发生的原因可能是涉及的执行器的通信链路存在缺陷、执行器操作不足以及受控过程(电梯本身)的失效或故障。关于不执行命令的反馈可以不包括在系统设计中，或者可以包括在设计中但没被自动控制器接收或没被处理。还可能涉及各种类型的技术设计问题，例如延迟（例如，接收输入和处理输入之间的时间，这将受到软件架构设计决策（例如使用中断或轮询）的影响）。数据过时问题也导致了危险，即使用的数据不再有效，或者输出命令在执行过程中被延迟到危险点。后者可能由于致动路径的延迟或由于受控过程条件而发生。例如，在一架

[39] 分离自动控制器软件的各种功能可能很有用处，但它们在这里组合在一起。将它们分开的一个原因是，如果逻辑上分离，软件的设计将更容易实现或改变。验证软件是否正确可能更加容易。

军用飞机上，一名维修工人正在炸弹舱门上工作并启动一个机械联锁装置，防止他在经过的时候把门关上。与此同时，在驾驶舱内工作的其他人发出命令关闭炸弹舱门。直到几个小时后，当维护人员释放机械联锁时，该命令才执行（因为机械联锁）。这名工人被杀。

过程模型与实际过程状态（以及各种控制器中的模型之间）之间存在不一致的原因还有很多。一个可能的原因是更新过程模型的延迟，这可能导致不一致和不安全的控制行为。在初始启动时，设计人员可能假定了一些默认值，这些值在某些情况下是不正确的。例如，某些飞机自动化应该在起飞前启动并初始化。通常为该情况提供默认值。但是，如果设备启动晚于起飞，默认值则可能不成立。例如，TCAS 假定它是在飞机在地面时启动的，并且初始化系统的值与地面相对应。关闭维护和重新启动的设备的过程模型也是如此，可能假设受控过程状态与系统关闭维护过程时的状态相同。另一个例子是人工控制器可能决定在设备看起来没有正常工作时重启设备。例如，TCAS 允许飞行员在空中重启它。但是，设备将使用初始值重新启动，这些初始值在空中飞行的那一点可能不正确。用于更新过程模型的输入也可以在设备加电之前，关闭之后或设备断开（离线）时到达，因此可以被忽略。

运行模式模型

运行模式的模型也可能与 UCA 中定义的内容相关。必须考虑的四种类型的模式是：（1）受控过程模式，（2）自动化模式，（3）监控模式，和显示模式。

1. 受控过程模式

受控过程模式识别受控过程的运行模式。例如，一架飞机（受控过程）可能处于起飞模式、着陆模式或巡航模式。建模的必要模式将取决于确定 UCAs 的相关内容。例如，如果飞机处于巡航模式（而不是处于着陆模式），则反向推进器的操作可能是不安全的。应该在 UCA 表中标识这些内容（即，当飞行器未处于着陆模式时反向推进器的操作）。UCA 的原因（致因场景）与自动化不理解受控过程的当前模式有一定关系。

2. 基本的自动化运行模式

除了受控过程外，自动化本身（图 G.2 中的自动控制）具有运行模式。这些模式通常被用于其控制算法的逻辑中并影响其输出。比如标称（正常）行为模式、关闭模式和故障处理模式。举个例子，在自身或在受控过程中检测到故障之后，自动化可以处于部分关闭模式。在法航 447 事故中，当飞机外部的压力探测器（皮托管）结冰而且自动驾驶仪无法再判断飞机的移动速度时，自动驾驶仪自动关闭（进入关闭模式）。同时，电传操纵系统继续运行，但切换到不再提供空气动力失速保护的模式。在现代空客飞机上，电子飞行控制系统的运行模式包括正常、交替、直接和

机械控制律，每个控制律都改变了自动飞行控制系统的行为。

自动控制器中的部分关闭模式也可能使人工控制器非常困惑，后面将介绍人工控制器可能出错的问题。当受控系统的模式与控制器认为的模式不同时，会出现模式混乱。模式混乱导致了许多事故，在生成致因场景时必须加以考虑。

在故障处理模式中，自动控制器可能会改变其行为，因为它认为受控过程或其自身存在故障，并且需要不同的行为来提供安全控制。

3. 监督模式

当多个监管器可以控制（提供控制命令到）自动控制器时，该模式可以随时识别控制自动化的人或物。基本上，监督模式允许协调多个监督器之间的控制行为实施。例如，飞行器中的飞行引导系统可以由飞行员（人工控制器）发出直接命令，或者可以从其他系统组件接收命令，而其他系统组件可以是人或自动化。虽然图 G.2 将所有自动控制器抽象为一个通用框，但通常可能有多个自动控制器。一个控制器可以以分层布置控制多个其他控制器，或者多个控制器可以并行操作，或者两种方式同时进行。当然，这可能变得非常复杂，例如，多个控制器可以各自由不同的控制器监督等。

4. 显示模式

该模型指定应向自动控制器的人工控制器显示何种类型的信息。当前显示模式将影响提供的信息以及用户将如何解释这些信息。图 G.2 中人工控制器模型的一个组成部分是他/她正在看到的当前显示模式。如果这些显示模式（在人和自动化中）变得不一致，则可能导致严重的问题，即，一个或两个部分的行为可以导致系统危险。这种类型的模式混乱称为“接口解释模式混乱”，并在《设计一个更安全的世界》的第 9 章中进一步描述。

人工控制器模型

在今天的一些系统中，自动控制算法使用人工控制器的状态模型，例如困倦或注意力不集中来确定其行为。越来越多的传感器被用来衡量人工控制器的状态。例如，汽车试图通过向自动控制器提供输入的各种类型的传感器来检测驾驶员是否醉酒或注意力不集中。从传感器到人工控制器的虚线，如图 G.2 所示，表示人们获得关于传感器是否工作或者感测变量的状态是什么（例如，驾驶员的中毒水平）的反馈的设计。

其他控制器的模型

如果自动化具有多个潜在的控制器，则自动化可能需要在这些控制器中有一个重要状态变量的模型。

环境输入

可能存在来自环境的输入直接进入自动控制器而不是通过人工控制器。例如，今天的一些自动化从 GPS 获得有关位置的直接信息。GPS 位置信息是根据环境中的信号（例如，通过天线）计算的，并且进入自动控制器，而不是通过人工控制器。在最新的飞机中，有一种称为 ADS-B（自动依赖性监视）的东西可用于在不同飞机之间传递信息，而无需通过人工控制器。注意，这不包括关于飞机本身状态的信息（例如，来自皮托管的输入），其通过受控过程中或受控过程上的传感器发送，是受控过程模型的一部分。

其他控制器/系统

在当今的一些复杂系统中会有多个自动化控制器，当然在未来的系统中肯定更多。 例如，可能有对飞机自动化的常规人工控制，也可能在飞机本身之外对飞机自动化进行一些控制(通常在地面上，但也可能位于与自动飞机相连的另一架飞机上)，这可能是人工控制的，也可能是自动的。控制职责可由不同的控制器共享。图 G.2 仅显示了一个控制器（棕褐色）以及其他控制器连接到自动化的可能性，其中每个控制器将具有相同的内部细节。

正如在描述运行模式时所提到的，自动化可能需要随时知道是什么控制器在控制它。使用共享控制可能会导致冲突命令的出现。一些混乱可能源于自动化模型与其控制器模型不一致而导致 UCA。可能出现的不一致和混乱的类型可以为致因场景的生成提供重要的贡献。

自动化与其监管器之间的信息传输

人工控制器通过各种类型的控制来传输控制命令，并通过显示器从自动控制器获得反馈。再一次重申，自动控制器与其控制器之间的信息传输将是致因场景生成过程中的一个重要（尽管非常直接）部分。

不通过控制算法直接更改自动控制器

在当今的一些系统中，当然在未来的系统中会更多，控制算法和其他软件以及自动化控制算法所使用的所有内部模型可以直接从外部改变，而无需人工控制器了解它。 例如，在某些飞机上，新软件可以通过互联网在飞机上更新，而无需飞行员甚至航空公司的维护人员（例如，变更可能源自 OEM）。这些变化可能会影响控制算法本身及其使用的模型，例如自动控制器中的地图和其他信息，以及人工控制器间接（可能通过显示屏）。 这里显然存在潜在的安全性问题。这些变化可能为生成 UCA 中的场景因素提供可能性。

人工控制器

人工控制器是图 1 中模型中最复杂的组成部分（尽管系统设计人员正在迅速增加自动控制器的复杂性），因此提供了致因场景生成过程中一些最重要的部分。模型的许多相关组件将有所贡献，包括控制行为生成和思维过程以及与这些功能相关的心智模型。

控制行为生成/思维过程

虽然这两个功能可以被建模为不同的组件，但它们具有在这种分离中可能丢失的重要交互。因此它们在这里被结合起来。该组件有两个基本功能：生成控制行为和更新心智模型。

生成控制操作：

使用来自图 1 中所示的人工控制器中的所有模型（环境、自动化和受控过程）以及培训、书面程序和经验的信息来生成控制行为。控制行为生成将受到与操作员交互的其他控制器的命令的影响。例如，飞行员部分地受到从航空公司运营中心或从其他控制器接收的控制行为或信息的控制，例如空中交通管制或机载仪表和系统，例如 TCAS（交通警报和碰撞避免系统[40]）。当控制责任划分不明确时，协调方面的挑战也会增加，因为不安全控制行为的致因场景也会增加。例如，FAA 将飞行员和航空公司运营中心之间的关系定义为 50/50，以便做出与飞机控制相关的一些具体决策。如果没有仔细划分责任（这可能最终是航空公司运营总监的责任），不明确是谁做出决定的混乱可能导致危险和事故（这种情况已发生过）。

当然，在控制行为生成期间可能存在大量不正确的行为。UCA 生成过程（其结果记录在 UCA 表中）用于识别控制行为不安全的场景，人工控制器用于识别当前场景的过程，这一过程将用于限制致因场景生成的数量。在场景生成过程中，分析师需要确定如何识别与 UCA 关联的已识别场景以及控制器为什么可能混淆当前场景。

在一些系统中，人工控制器可以直接与受控过程（由人工控制器与执行器和传感器之间的虚线以及受控过程展示）交互，而无需经过自动化。

更新心智模型：

除了生成控制行为外，人工控制器的主要责任是更新它们的心智模型。心智模型总是通过人为处理来进行更新，尽管其中一些处理可能是潜意识的。因此，模型

[40] 虽然 TCAS 可以被认为是一个自动控制器（在图 G.2 中的分层控制结构的中间），但是飞行员需要遵循 TCAS 解决方案咨询（控制命令），因此在某些方面 TCAS 控制着飞行员。但是，飞行员可以输入 TCAS 的设置来控制它如何生成解决方案建议。随着越来越多的功能在飞机和其他系统上实现自动化，人与自动化之间的各种“伙伴关系”将是必要的，协调方面的挑战将会增加。

中存在双向箭头（人们在其心智模型中更新和使用信息，这两种功能都可能出错）。

人们在此更新中使用的信息部分通过显示器获得，部分来自其控制器或与其交互的其他控制器（例如，从空中交通管制或航空公司运营中心传输的信息）的输入，部分地通过直接传感输入控制器获得（例如，看着窗外，看到龙卷风接近或感觉到飞机上的湍流）。例如飞行员接收从航空公司运营中心或其他控制器（例如空中交通管制）发送的信息（例如 NOTAM）。在一些系统中，操作人员具有直接观察执行器操作的能力，其在图 G.2 中由从执行器到人工控制器的虚线描绘。例如，在一些工厂中，操作员可能能够走到执行器和阀门并观察它们是否打开和关闭以检查是否正常运行。举个例子，飞行员通常在起飞前或在飞行前检查期间进行巡视，他们可以移动控制装置并直接观察飞机副翼以确保它们沿正确的方向移动。人工控制器还可以从控制器的状态接收输入，例如，驾驶员可以看到方向盘本身移动并且假设自动化已经发出与转向相关的控制命令。

模型更新也可能受到操作员（控制器）所受训练的影响。另一种微妙的输入类型可能来自控制器被飞行员操作时的反应（如图 G.2 中从控件到人工控制器的虚线所示）。操作员可以基于控制器的移动来做出关于自动控制器和受控过程的状态的假设，例如控制杆在飞机上的移动或自动车上的方向盘。

更新模型和生成控制行为之间的交互：

与自动化控制器及其过程模型的更新一样，人工控制器控制生成过程与心智模型之间可能存在重要的相互作用，类似于自动控制器的描述。例如，如果人工控制器发出命令“Do X”，则人工控制器可能会认为“DoX”命令已由自动控制器和受控过程的执行器执行，并且无意识地更新了它们的心智模型自动化和（或）受控过程的状态。实际上，如果从未执行“Do X”命令，则人工控制器的过程模型将与自动化和（或）受控过程的实际状态不一致。可能有许多原因导致命令无法执行，例如，如果自动化认为命令不安全或在该时间点不可能发生，或者受控过程中可能存在失效或故障，则可能会阻止和忽略该命令。

在自动控制算法一节中描述了相同类型的过程模型的有缺陷的更新。但是在自动化的情况下，可以检查算法以确保不会发生此问题。要从人类的思维过程中排除这一点更加困难。

人工控制器可能会收到不执行命令的反馈，但是由于分心或其他原因，人们可能不会注意到它。

人工控制器使用的心智模型

图 G.2 中为人工控制器确定了四种类型的心智模型：环境模型、自动化状态模型、受控过程模型以及其他控制器模型。这些模型中的任何一个都可能与实际状态

不一致，从而导致不安全的控制行为。这些模型和自动控制器中的相同模型具有相似的内容和用途，尽管它们存在缺陷的原因可能完全不同。致因场景必须包括造成危险不一致的潜在原因，这个可以引出 UCA 的环境条件。

因为人工控制器通常控制自动化（直接）和受控过程（间接但在某些情况下直接），人工控制器需要两种状态的模型。

为了监督自动化，人工控制器需要对自动化的运行方式有基本的了解。缺乏这种了解可能导致不安全的控制和损失。

人工控制器心智模型可能不正确的原因有很多。对于之前给出的模型在自动控制器中不正确的原因，有一些可能适用。但人工处理也带来了许多其他机会。我们强烈建议人为因素专家参与人工控制器的致因场景生成。以下是应该考虑的一些（但只有一些）因素的列表：

- 模式混乱是模式丰富系统中发生事故的常见原因，其中人们不了解自动化模式，或者自动化与受控过程的当前模式混淆。模式混乱可能是由过程模型中的不正确更新引起的，输入可能不正确或延迟，可能会延迟更新或者可能会通知人工控制器模式改变但不会注意或处理该信息。当自动化可以改变受控过程的模式而无需人工控制器指导时，可能导致模式混乱和潜在的不安全控制行为。
- “致因场景意识”通常被认为是事故的原因，该术语的含义却缺乏具体定义。大量错误可归入此类别。 请注意 STPA 中使用的模型可以识别态势感知错误，因为它们仅仅意味着人工控制器心智模型与真实状态不匹配。
- 人们很容易对不确定的，或者大部分时间以某种方式行动但在一些特殊情况下表现不同的自动化感到困惑。这种自动化也大大增加了训练难度。
- 对某些自动化进行编程，使得逻辑产生了人工控制器无意或不知道的副作用。
- 虽然系统的设计可能包括对人们的反馈，但有许多原因可能导致这一反馈被忽视，例如当监控几乎不变或者很少发生错误的事情时注意力分散或警觉性降低。
- 人们对自动化的自满和过度依赖日益成为当今自动化系统中的一个问题
- 自动化可能如此优雅地失效，以至于人工控制器直到过程的后期才会注意到这一问题。人们也可能误认为自动化已经关闭或者失效。当机器人和人必须在相同区域工作时，就会出现这种类型的问题。注销/锁定问题，人们认为能源已关闭但实际上已经开启，导致工作场所发生大量事故。

此列表仅用于表示当人是系统的一部分时，许多原因必须要考虑在内。它远非详尽无遗。如果致因场景需要达到这种详细程度才能消除或控制，与人为因素专家合作是个不错的选择。

一些最后的想法

本附录中提供的模型和指南不应用作生成致因场景的清单。它肯定是不完整的，特别是对于一些独特类型的系统。您需要仔细考虑自己系统的工作方式。我们发现让系统设计和运行专家参与致因场景生成是非常有益的。但是，本附录中的模型可能有助于排除一些基本的遗漏，帮助您确定更完整的场景合集。

我们还想提醒的一点是，试图查看系统中的所有潜在缺陷，看它们是否可以导致 UCA，基本上还是 FMEA 中所使用的过程，这对复杂系统来说是没有用处的。始终从危险和 UCA 及其背景因素开始，然后从它们向后工作以确定潜在原因。可能数量巨大，但比尝试识别可能出错的所有内容然后查看是否会导致 UCA 要少得多。