第1章 绪论

讲授： 任胜兵
中南大学 计算机学院

中南大學

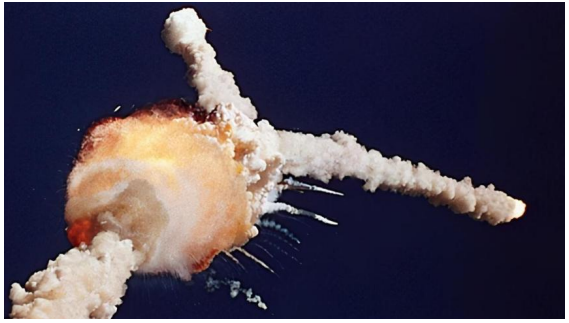
内容提要

- 软件失效
- 软件可信性属性
- 软件可信性工程

软件不可信的根本原因是什么?

可信性（Dependability）反映了一个系统向其用户提供预期服务水平的能力。

1996年6月4日，阿里亚娜”V型火箭升空约30秒后姿态控制失灵，随后解体并引发了剧烈的爆炸，造成5亿欧元损失。



“阿里亚娜”V型火箭首次发射失败是由于火箭导航的软件系统发生失效造成的，其中存放火箭水平速率的值出现溢出（制导系统需要将测得的速度数据从64位浮点数格式转换为16位有符号整数格式）。

关注软件是否可信的最根本原因是软件失效。

可信性（Dependability）反映了一个系统向其用户提供预期服务水平的能力。

1991年海湾战争期间，美国爱国者导弹系统未能成功拦截伊拉克飞毛腿导弹。

Rounding errors may have deadly consequences

On February 25, 1991, during Operation Desert Storm, a Scud missile fired by the Iraqi army hit a U.S. army barracks in the Dhahran Air Base in Saudi Arabia, killing 28 soldiers. The air base was protected by a Patriot Air Defense System, which nevertheless failed to track and intercept the incoming Scud missile. An investigation of the incident by the U.S. army revealed that accumulation of rounding errors in the Patriot tracking software was responsible for the failure of the interception[5].

Each battery of the Patriot Air Defense system consists of a ground based radar and eight missile launchers. The radar detects airborne objects and tracks their motion. In order to distinguish between different types of objects and, in particular, in order to identify ballistic missiles, a weapons control computer calculates the expected trajectory of the missile. If the airborne object detected by the radar follows the expected trajectory, a Patriot missile is fired to intercept it. The weapons control computer uses an internal clock to keep track of time and perform the calculations of the trajectories of the detected objects. The time is stored in 24-bit variables (registers) in increments of 0.1 seconds. In the binary system, however, $(0.1)_{10} = (0.0001100011...)_2$ and hence any calculation involving this time increment needs to be rounded. On February 25, 1991, after approximately 300 hours of continuous operation, the control computer had accumulated enough rounding error that it did not predict accurately the trajectory of the Scud missile. As a result, the missile was not identified and a Patriot was not fired to intercept it.

错误是由于爱国者导弹系统的内置时钟在长时间运行后产生了微小的误差，导致雷达系统无法准确预测飞毛腿导弹的弹道，从而未能成功拦截导弹：十进制的1/10用二进制来表示就会产生一个微小的精度误差。

关注软件是否可信的最根本原因是软件失效。

可信性（Dependability）反映了一个系统向其用户提供预期服务水平的能力。

12306订票系统崩了

故障的根本原因在于系统架构规划以及客票发放机制存在缺陷，无法支持如此大并发量的交易。

关注软件是否可信的最根本原因是软件失效。

软件失效往往是由于软件系统错误状态引起。

引起软件系统错误状态的原因，我们称之为软件故障。

故障类型

- 退化故障，又称为耗损故障，是由于产品的规定性能随时间增加而逐渐衰减引起。

  软件故障不是退化故障。

- 设计故障是系统设计中存在的缺陷。

  设计故障在软件生命周期的任何时刻都会出现。

- 拜占庭故障是一类行为完全无法预测的故障，其对系统不同部分影响不同。

  由于恶意攻击和软件错误会导致故障节点表现出拜占庭故障。

要解决软件可信问题，关键是如何处理软件故障。

什么是软件可信性?

软件可信性是一系列软件质量属性的集合。

可信性（Dependability）

信任（Trust）

不同的系统，这些质量属性的重要性不一样。

软件可靠性是对软件给定条件下规定的时间内持续提供正确服务的一种概率度量R(t)。

可靠性

假定系统从第0时刻开始正确提供服务，R(t)表示在[0,t]的时间段内在规定的运行环境中软件系统正确运行的概率。

给定条件包括软件运行的软硬件环境以及软件运行的输入空间及其概率分布。

规定的时间一般可分为三种：日历时间、执行时间或时钟时间。

可靠度R(t)描述了系统成功运行的概率。
失效的概率Q(t)=1-R(t)

软件缺陷导致的失效往往没有先兆

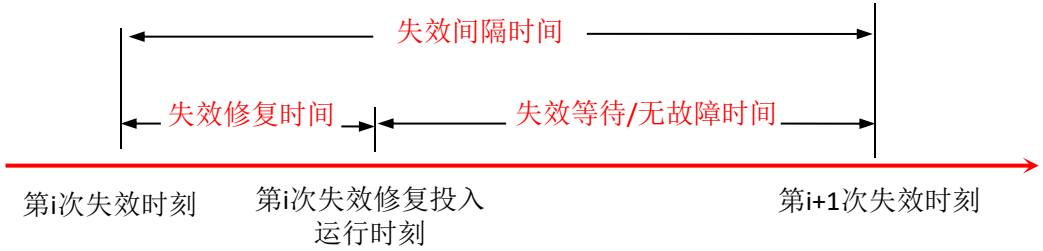
一些微小的变化也可能引起失效

软件可靠性比硬件可靠性更加复杂。

软件可用性A(t)是指软件系统在t时刻在一定条件下能够正确运行的概率。

可用性

可靠性要求系统提供连续的服务，而可用性允许系统失效，在经过维修后仍能正确运行即可。



稳态可用度 $A_\infty = \dfrac{MTTF}{MTTF + MTTR}$

MTTF平均失效等待时间

MTTR平均失效修复时间

最大修复时间MaxTTR $P(MaxTTR \le T) = q$

MaxTTR以概率q不超过时间T

修复时间越短，用户可接受程度越高。

软件防危性定义为单位时间内软件失效损失的期望值小于规定的阈值。

防危性

安全攸关的系统，必须在系统开发早期阶段就将防危性考虑在内。

风险Risk $\quad Risk = \sum_i P(失效_i) * L(失效_i)$

$P(失效_i)$表示单位时间失效$_i$出现的概率

$L(失效_i)$表示失效$_i$出现带来的损失

WPS文档备份

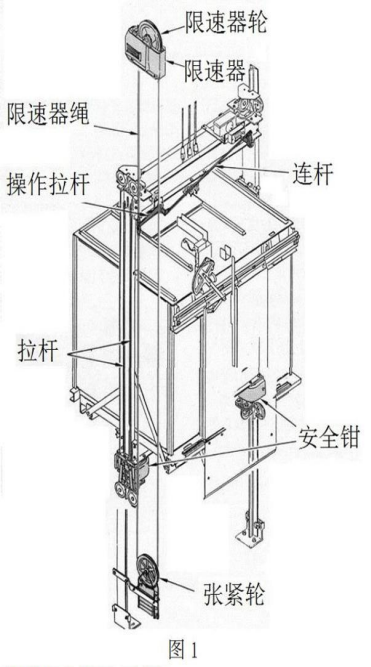


图 1

保持失效带来的损失在可接受的范围内：失效安全操作

软件完整性是指对软件系统不存在未经授权的修改以及对软件系统行为不存在未经授权的操控。

完整性

静态完整性是指软件本身对应的程序代码不被非授权篡改。

动态完整性是指软件运行过程中不存在对软件行为的未经授权的操控。

完整性保护

- 预防技术用来防止软件受到攻击（登录口令）
- 检测技术用来确定在攻击无法避免的情况下检测攻击正在发生或已经发生，并向用户报告攻击的性质、严重程度等（输入密码连续错误三次）
- 恢复技术是通过容错或备份等处理技术来阻止攻击，使系统保持正常运行

软件完整性保护是软件保护技术的一部分。

CIA：机密性、完整性、可用性三者统称为软件安全性三要素。

安全性

机密性，又称隐私。信息只能在所授权的时间、所授权的地点暴露给所授权的个人、机构或计算机软件程序，对于未被授权者，这些信息是不可获得或不可理解的。

安全性：遵守交规
事故前发生作用

防危性：买保险
事故后发生作用

软件安全性与软件可信性有密切联系。

软件可维护性M(t)是指软件失效后在一段给定的时间内可修复的概率。

可维护性

软件可维护性反映了软件维护人员为纠正软件缺陷或满足新需求而理解、修改和改进软件的难易程度。

$$M(t) = 1 - e^{-\mu t}$$ μ为系统的维修率

影响软件可维护性的因素有：

- 可理解性：理解软件的结构、接口、功能和内部过程的难易程度
- 易分析性：诊断软件故障，并对需修改的部分进行定位的难易程度
- 可修改性：软件便于修改的难易程度
- 易测试性：为程序建立验证准则和性能评价的难易程度

可维护性的提高有助于可用性。

如何满足软件可信性?

软件可信性工程是指在软件开发过程中利用有关方法和技术确保最终系统满足可信性需求。

主要目标是正确处理软件系统中的故障：

- 故障避免：在软件开发和维护过程中避免引入故障
- 故障移除：检测、识别和修复故障
- 容　　错：出现故障时，系统仍能继续提供可接受服务的能力
- 故障预测：预测将来可能出现的故障种类、数量和故障对于系统的影响

故障是可信性问题的核心。

故障避免：在故障引入之前应该进行的处理。

在软件开发和维护的各个阶段，都可能引入故障：

- 严格按照工程的方法来开发和维护软件

掌握工程方法
遵循工程方法

- 积极使用故障避免处理技术

原型技术可以避免需求规格说明不完整
形式化方法有助于软件正确性验证
制定并遵循一致的编码标准和规范提高可维护性
重视培训避免使用不当

故障是可信性问题的核心。

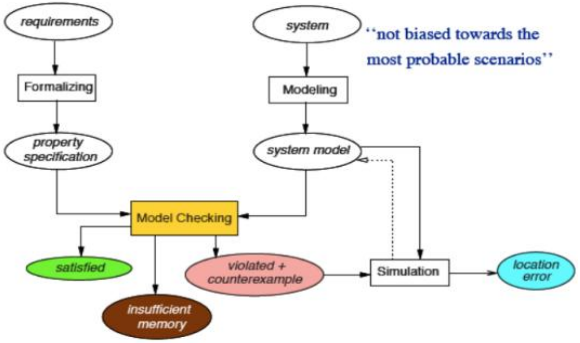
故障移除：在故障引入之后进行的处理。

- 人工审核技术

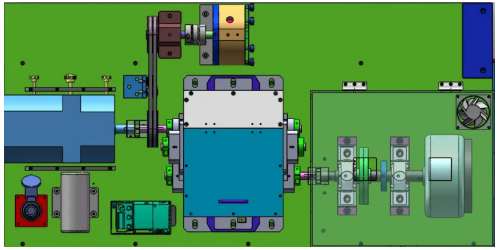
质量管理

- 软件测试技术

- 模型检测技术



- 系统仿真技术

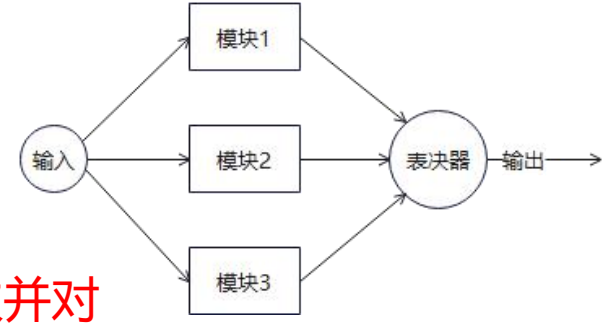


故障是可信性问题的核心。

软件容错：在故障存在于已投入运行的系统中进行的处理。

容错就是要避免故障引起不可接受的失效。

- 软件容错处理是一件复杂的工作
  - ✓ 不知道故障的表现形式
  - ✓ 难以确定故障在软件中的位置
  - ✓ 不清楚故障的失效语义
  - ✓ 设计故障的影响很难估计
- 冗余是软件容错处理的常用方法
  - ✓ 时间冗余：重复计算或数据传输多次并对结果进行比较
  - ✓ 空间冗余：通过增加多余的部件或数据项和表决器，降低故障引起的失效



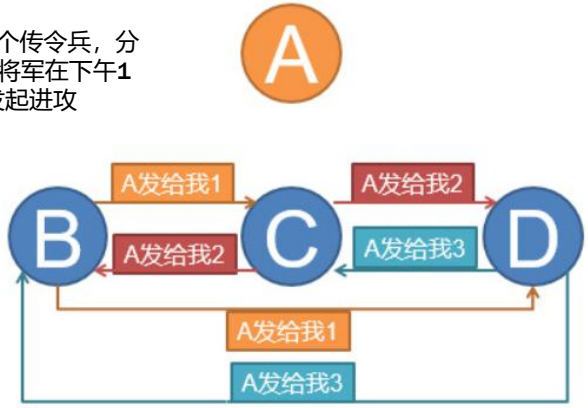
故障是可信性问题的核心。

拜占庭容错：系统在出现拜占庭故障的情况下仍能够保持正确处理的容错技术。

拜占庭将军问题：考虑4个将军的情况，同时假设4个将军中最多只有1个背叛者。当4个将军A、B、C、D把敌人包围了之后，必须协商一个统一的时间去发起进攻。对于忠诚的将军来说，他不知道谁是背叛者，所以，他不能完全相信接收到的命令，他必须对命令做出判断。

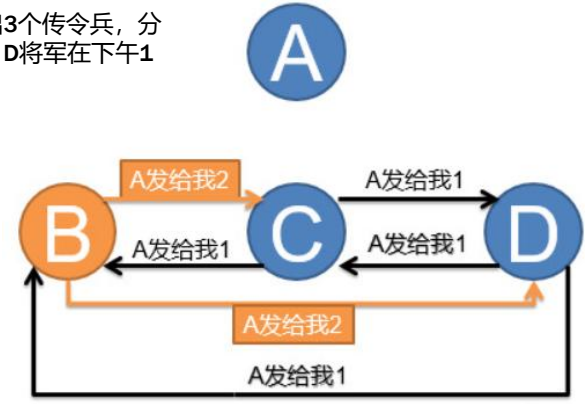
解决问题的核心思想是：对于每一个收到命令的将军，都要去询问其他人，他们收到的命令是什么。

情况1：A派出3个传令兵，分别告诉B、C、D将军在下午1点、2点、3点发起进攻



情况1：A是叛徒

情况2：A派出3个传令兵，分别告诉B、C、D将军在下午1点发起进攻



情况2：B是叛徒

故障是可信性问题的核心。

软件故障预测：是指通过评估系统的行为、结构等预测将来可能出现的故障种类、数量和故障对于系统的影响。

软件故障预测方法：

- 定性故障预测：对导致系统失效的模式或事件组合进行排序
- 定量故障预测：采用概率来评估某些可信性属性满足的程度
- 静态故障预测：基于故障相关的软件度量数据，对故障的数量或者分布进行预测的技术。
- 动态故障预测：基于故障或者失效产生的时间，对故障随时间的分布进行预测的技术。

故障是可信性问题的核心。

软件可信性工程主要任务

- 针对系统的功能需求，确定系统可能遭受的风险。
- 针对每个风险，确定导致该风险的故障。
- 针对每个需要处理的故障确定可信性需求。
- 采用四种故障处理技术来处理软件故障。
- 对系统是否满足可信性需求进行评估。
- 重复上述任务，确保软件系统尽可能满足系统的可信性需求。

故障是可信性问题的核心。

感谢各位聆听！
祝大家学习愉快！