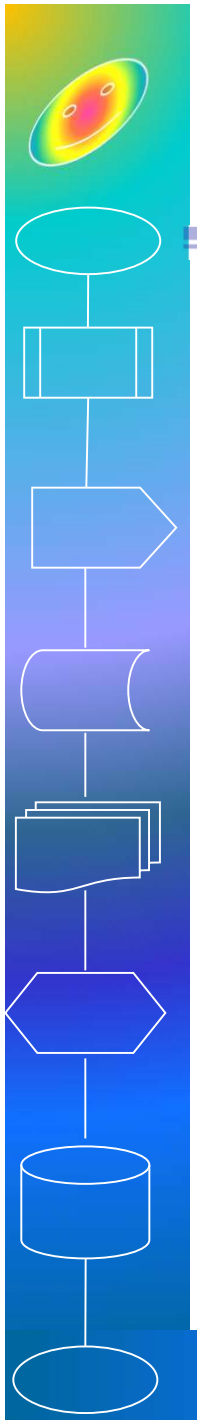


第二章 基于计算机的系统

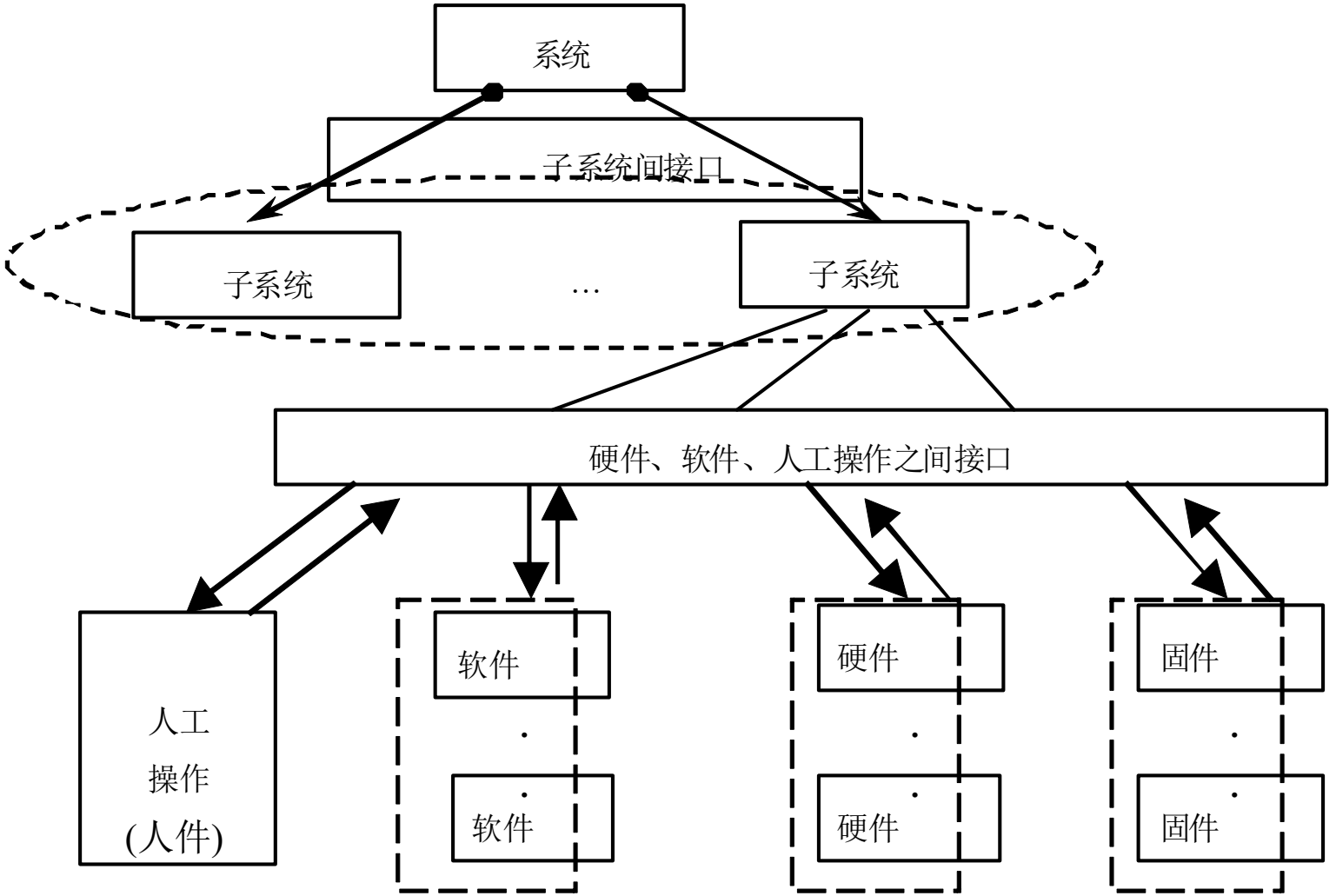
Hardware, Software, People-ware

目录

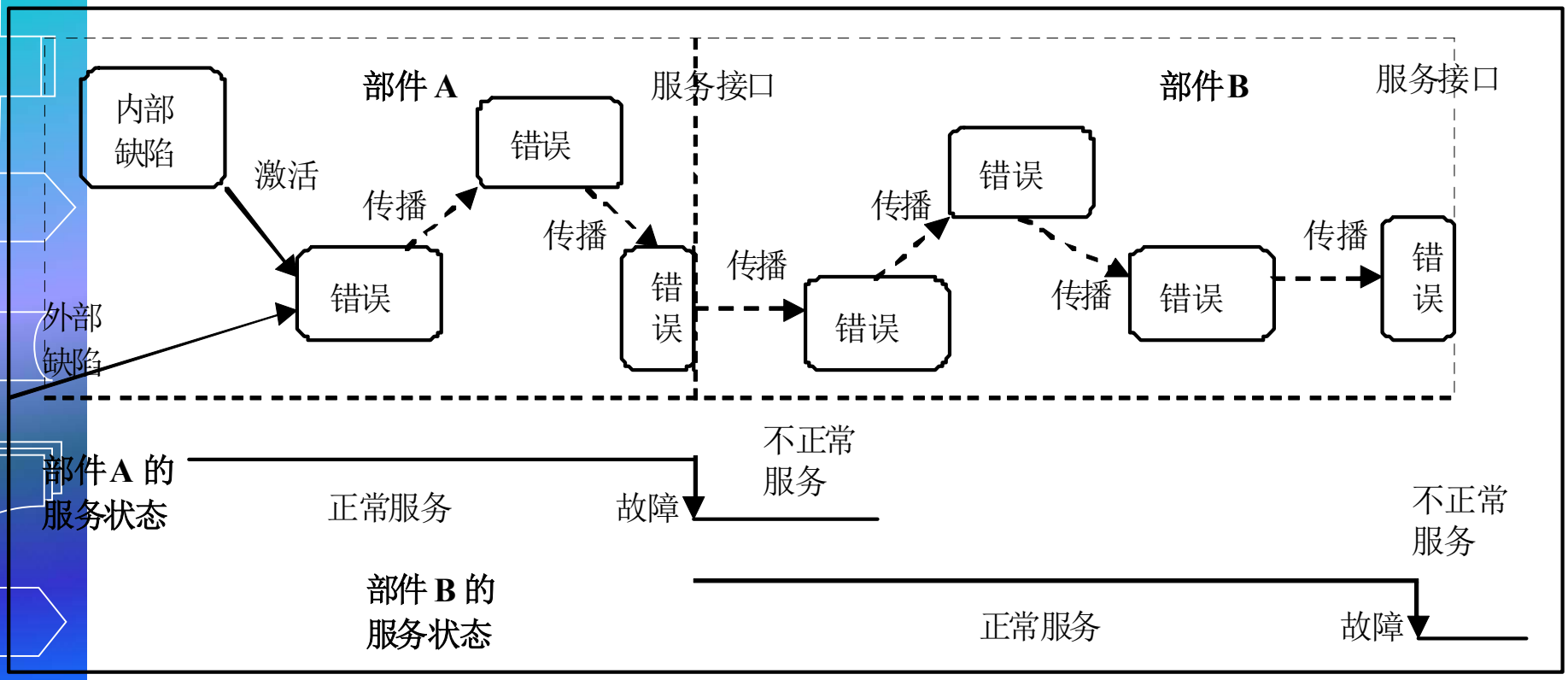
2.1基于计算机的系统组成和特征

- 2.1.1 系统的组成
- 2.1.2 系统故障
- 2.1.3 硬件的连续性
- 2.1.4 软件的离散性
- 2.1.5 人的特征与管理
- 2.1.6 固件与嵌入式系统

系统的组成



导致故障或服务失效的错误传播链



发生错误的例子

1）程序员(开发人员)的工作中的缺陷，例如，程序代码缺陷或数据定义错误。这些缺陷没有在评审和测试等工作中被发现。

2）硬件设计或生产的缺陷，例如，集成电路模块中的线路短路，导致逻辑的布尔值失效，或电路功能变化。只要该缺陷未被激活，处于休眠状态的短路不会发生错误。

3）系统的使用和操作人员的做了不恰当的人机交互动作，例如，输入的值超界，企图改变系统数据处理结果，而导致的错误。

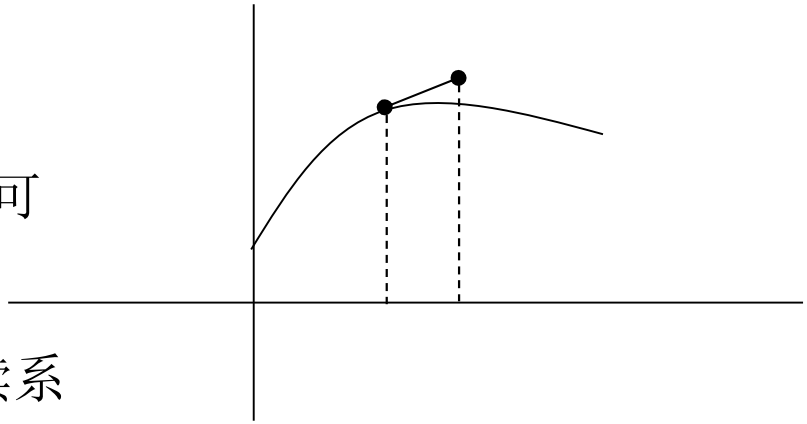
4）系统的操作手册或维护手册上的缺陷导致操作人员不正确的操作。

故障和错误的现象和原因

系统故障												
本质		现象			系统界限		制造(开发)阶段		持续性			
偶然	故意	物理	人为		内部	外部	设计	操作	永久		临时	
X		X			X			X	X			物理故障
X		X				X		X	X			
X		X				X		X			X	瞬时故障
X		X			X			X			X	周期性故障
X					X		X				X	
X			X		X		X		X			设计故障
			X			X		X			X	交互作用故障
	X		X		X		X		X			有意的恶意错误
	X		X		X		X				X	有意的非恶意错误
	X		X			X		X	X		X	干扰

2.1.3 硬件的连续性

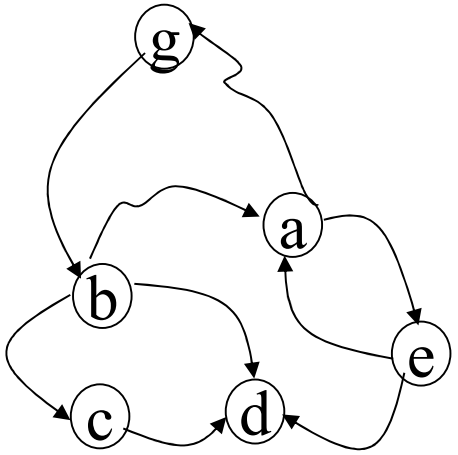
- 连续系统
  - 至少是一阶可导的, or
  - 多阶可导函数
  - 其行为遵守小线性的规律, 因此,可以用小线性方法预测



- 通常，硬件是一个物理方式存在连续系统。
- 人们通过建立系统的动力学方程等，依据当前的状态和可能输入变量预测系统的下一步的行为，从而对系统的特性进行有效的控制。当系统非常复杂时，或者说，无法建立系统的动力学方程时，人们可以采用建立小线性方程，对系统进行近似。
- 由于系统的连续性，采用小线性对系统的行为进行预测是可行的。线性化的离散间隔越小，预测的越接近于实际。

2.1.4 软件的离散性

- 离散系统又可以分为两类：时间离散和事件离散。
  - 事件离散是指系统接收系统外界的某个或多个事件的输入，从而引起系统的状态和输出的变化。“事件”可能是外部人工干预的输入，也可能是计算机硬件输入。
- 一个计算机程序是一个离散系统: 字长限制，条件转移
- 你不知下一个状态将会发生什么，除非你测试完所有的状态和路径。



2.1.5 人的特征与管理

- 系统的使用者----人也是导致系统故障的重要因素之一。
  - 要避免系统运行时的错误，也必须研究操作者的错误特征，建立人的故障和错误模型。
- 系统的建设者也是人。
  - 人在软件的开发和硬件的制造过程中扮演者着重要的角色，往往会给系统带来诸多的系统设计和软件开发错误。
  - 对于软件开发来讲，程序员或软件开发者是最了解软件内部构成和脆弱点的人。

2.1.6 固件与嵌入式系统

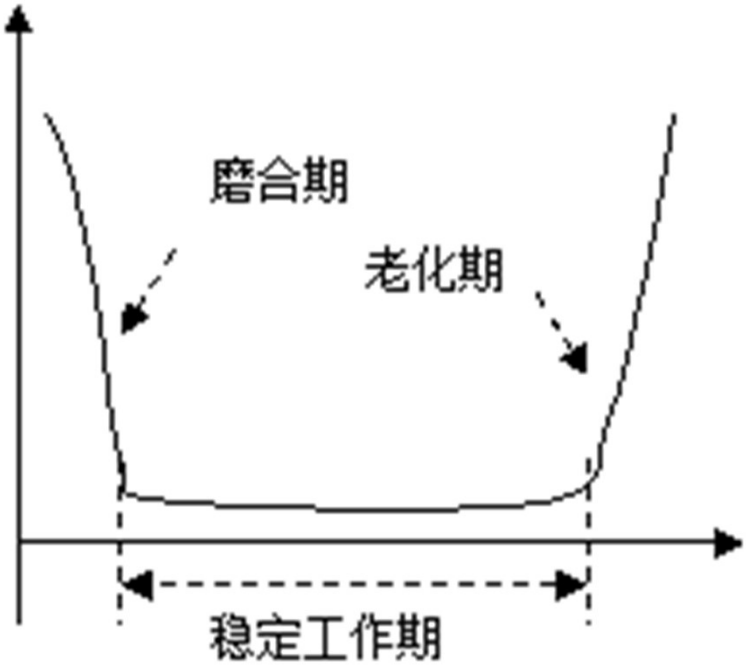
- 固件(Firmware)是指写入EROM或EPROM(可编程只读存储器)中的程序及其数据，通俗的理解就是“固化的软件”。
  - **微处理器硬件质量：**带有微处理器和外围设备的硬件的质量直接影响计算机系统的可靠性。
  - **内嵌软件的质量：**嵌入(内置)的软件是整个嵌入式产品计算或控制功能的一个重要元素。
- 网络能够让固件接入互联网，让供货商可以直接对固件升级。
  - “黑客”可以对用户的固件升级“危害性的”软件，破坏嵌入式系统的，或偷取信息等。

2.2 硬件特征和系统建造理念

- 2.2.1 硬件的故障特征
- 2.2.2 硬件系统的全生命周期设计
- 2.2.3 硬件生产质量的统计学控制

硬件的故障特征

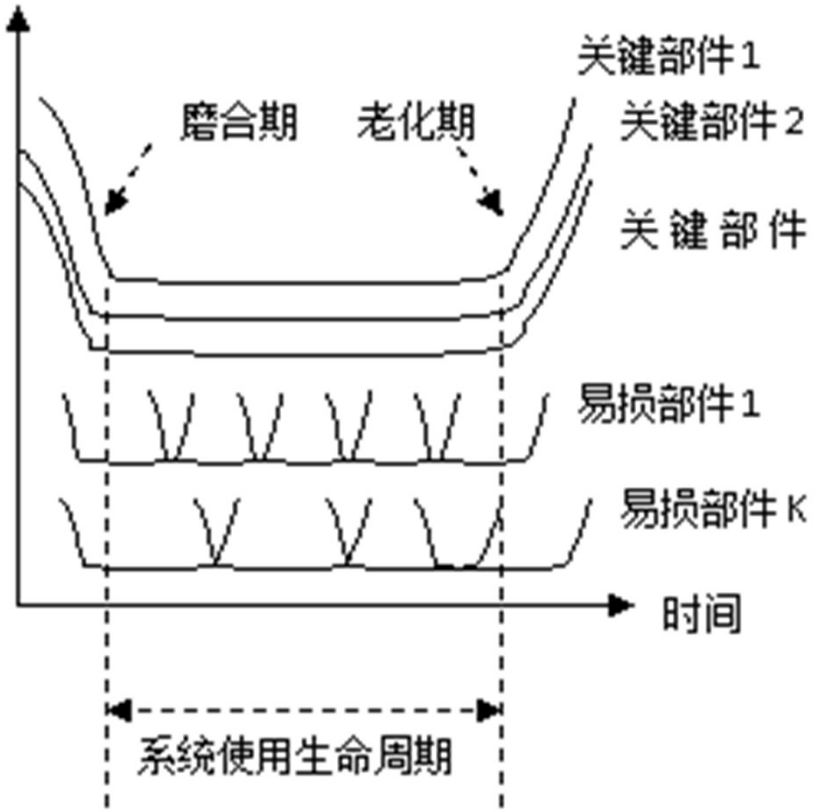
计算机的硬件，例如电路板、机械设备等是以物理形似存在的。这些物理体是占据空间的。自然界的物理体一定会受到风吹、日晒、湿气、空气清洁度等外来因素的影响，从而会出现老化或磨损。物理体的这种老化和损坏是一个连续渐变的过程。



硬件是人们制造出来的，所制造出的机械和电子器件装置具有一段的磨合期或老化期。经过磨合或老化物理硬件可以进入比较稳定的工作期，其故障率相对很低。硬件在其生命期内发生故障的规律基本符合如上图的规律。

硬件系统装配成系统的故障特征

由N个关键部件组成的系统和K个易损部件组成的系统。设计师期望关键部件在系统的使用寿命内不要更换，例如汽车的发动机和车架，因此要求关键部件的使用寿命尽可能一样长。其它的易损坏的部件，例如汽车城的轮胎和电瓶等，可能在系统的生命周期内需要更换多次。整个系统的使用寿命取决于寿命最短的关键部件，而不是可更换的一般部件的寿命。



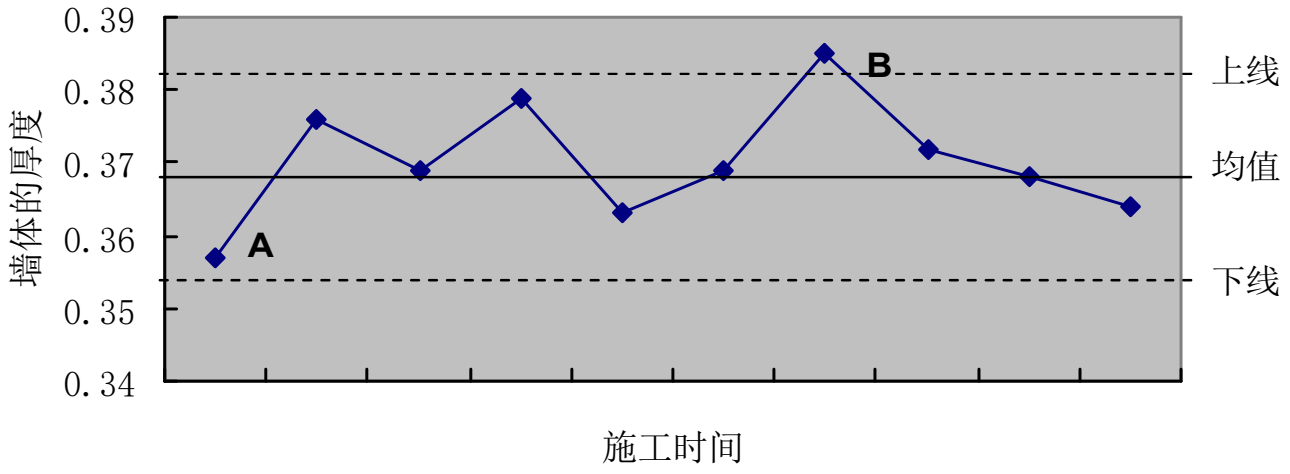
全生命周期设计

- 如果不能保证每个硬件部件具有相同稳定工作期。那么，系统设计者和工程师们就需要统计每个部件的稳定工作时间长度，将那些寿命短而需要经常更换的部件暴露出来，便于维修工程师的修理。并规划出系统中各部件的修理规律和时间，从而在降低维修的成本，为客户提供可预测的高质量服务。
- 当系统的设计者、生产者、维修人员和用户预先知道合适需要维修、更换系统的相关部件，并对系统及时保养时，系统使用中的质量是可以信任的。
- 从系统使用的质量看，所谓的系统使用质量取决每个关键部件质量、易损部件的及时更换和安装质量等。
- 设计者不仅要考虑系统的功能性、性能、易用性、可靠性等因素，还要在设计中考虑系统的可维护性、部件易更换性等。

硬件生产质量的统计学控制

Shewhart的控制图的例子如图所示，横坐标表示时间，纵坐标表示(中间)产品的质量性能，例如墙体的厚度。我们假设一个建筑队的多名工人修建一段砌墙体，每个工人堆砌的砖块都会偏离理想线---中间线。

中间线是理想质量线或所期望质量，其它两条虚线分别表示偏差的上线和下线。



- **控制线内的点**，例如，点A虽然与中间线有差异，但是在限制线内，属于正常的质量偏差范围，质量工程师能够保证这段墙不会倒，只要每个点都在控制线范围内。属于正常的生产范围，除非要求更高的质量精度，即压缩上下线偏差范围。
- **控制线外的点**，例如，点B偏离了超出了限制范围，就需要寻找其原因。点B的出现可能是由于工人的不认真工作造成的。属于不合格的缺陷，随着B点的增加，墙体倒掉的可能性增加。生产者需要控制B点的出现，而允许A点存在。

2.3软件故障和建造理念

- 2.3.1软件故障表现和分类
- 2.3.2程序正确性证明
- 2.3.3测试的充分性问题

软件故障表现和分类

- 软件引起计算机停机的主要原因通常有：1）计算机除零错误，2）指向内存地址的指针超出规定的内存范围，3）整型数上溢出或下溢出，4）浮点数上溢出或下溢出。必须避免这类严重的错误。
- 其他类型的错误包括：a）数据计算精度不够，例如，银行利率计算误差太太，b）输出或显示的数据位数不够等。

CO= Computational Error(计算类错误)	OP = Operational Error(操作错误)
LO = Logic Error(逻辑错误)	RI = Requirements Incorrect(需求不准确)
DH = Data Handling Error(数据处理错误)	DE = Design Error(设计错误)
IN = Interface Error(接口错误)	CL = Clerical Error(书写错误)
DB = Data Base error(数据库错误)	OT = Other

Web服务器的典型故障

- 资源耗尽，例如，内存泄露，线程池耗尽导致的速度降低，资源密集处理导致服务请求超时，快速增长的日志文件消耗掉磁盘
- 计算/逻辑错误，例如，引用数据库中不存在的表，使用未释放的内存，指针崩溃，程序死锁等造成的同步错误等。
- 系统过载，系统过载导致的资源竞争，以及与其他软件进程之间的频繁交互。
- 可恢复代码错误，系统的容错设计过于复杂，并且没得到充分的测试。
- 系统的升级错误。例如，升级前的备份工作出现问题。升级时，未检查升级前后的软件兼容问题。软件升级后自动进入到缺省状态。集成错误或第三方软件有错。

程序正确性证明

- 如果能够证明一个程序系统是正确的，或者存在一些方法能够自动生成正确的程序，软件的开发就成为很容易的事情了。
- 理论界许多学者一直致力于对程序正确性的证明，形式化软件开发方法则企图实现代码的自动生成，保证代码的正确性。
- 但是，还没办法证明任意的一个程序是否正确。
- 程序的正确性证明是软件理论界研究的方向之一。

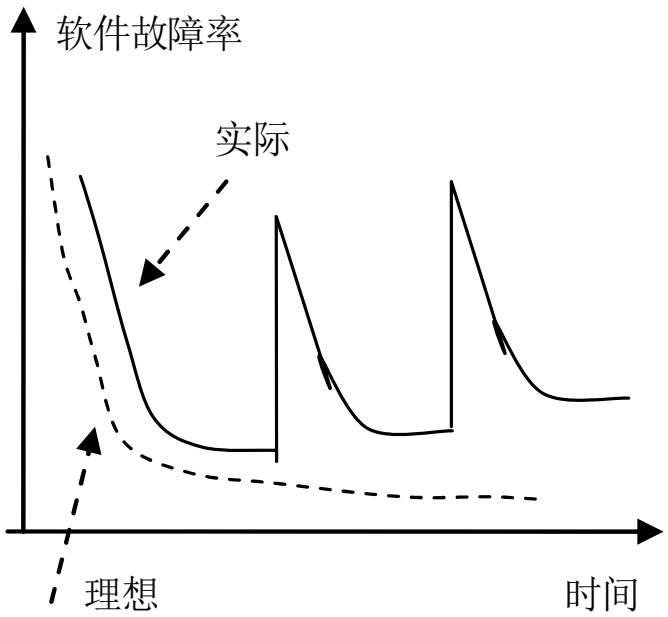
测试的充分性问题

- 由于测试时间与计算机程序的状态、输入事件和程序中的逻辑判断的个数组合是一个指数级的关系，彻底遍历一个复杂的程序所花费的时间往往会大于软件开发者所期望的时间，甚至会大于软件使用的生命周期。
- 企图通过用测试证明软件的正确是徒劳的。
- Dijstra劝告系统的测试者：“测试可以表明缺陷的存在，而不是不存在”。

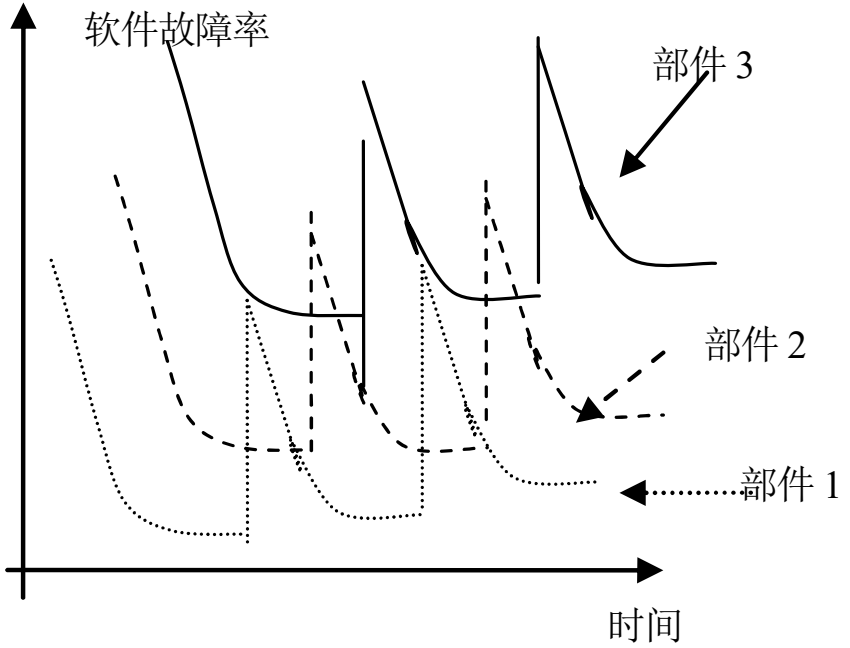
代码复用

- 使用的次数越多、时间越长、用户使用越多的软件要比首次使用的软件更可信一些。长时间、大批量用户的使用促进了对软件更进一步的测试。因此，最好能重复使用(复用)经过运行考验的软件部件。
- 建立软件库可以提高代码的复用率，且省时省力。基于软件部件复用的设计理念可以提高软件系统建设效率，并最大限度地降低编写新代码所带来的缺陷问题。
- 但是，一些软件故障的出现是短暂的，很难再发生或将其复现出来。
- 软件的这种不稳定的表现很难复现和测试出来。因此，对于复用的代码也需要尽可能的充分测试。

部件组装



a．软件理想的和实际故障率示意



b．软件系统多处和多次被修改故障

单个部件与多部件组装、修改所的故障规律

软件开发理念

- **理念1：降低修改频度，控制变更。**如果能降低一个部件更改产生的对其它部件故障率的影响，就能极大的整个降低系统故障率。第19章的配置管理讨论变更管理。换一个角度看，也可以从修改的情况来预测软件中的缺陷个数，参见第15.4节的讨论。
- **理念2：降低部件之间的耦合度。**尽可能保持部件与部件之间的松耦合，从而降低部件相互之间的影响。要尽可能让系统中的部件的结合时“松散的”，而不能过于紧密。例如，部件之间的通过数据传递信息，就比部件之间的直接调用的耦合程度低。修改一个部件时，对另一个部件的影响就小。第10和11章讨论软件体系结构设计。

软件开发理念

- **理念3：区分需求稳定性。**需求变更是引起系统变更的主要因素之一。因此，将长期稳定的需求设计为系统中不变的核心部件，保持核心部件的稳定性；将需求不稳定的部件与核心部件区分开来，适应频繁的修改，就可以降低由于变更引起的故障率的上升。第8章讨论需求工程与管理。
- **理念4：复用。**采用稳定的、成熟的、质量可信的部件是降低故障的一个保证。针对应用领域建立可信赖的软件部件库，提高部件和代码的复用率。

软件开发理念

- **理念5：充分(回归)测试。**一旦对某个部件进行修改，就必须对整个系统进行充分测试。但是，由于面临着财政和进度等的压力，不允许不计成本地进行测试。那没，就需要针对修改部件，评估系统影响范围，测试所有受影响的部件、数据、逻辑路径等，常常，这种测试称为“回归测试”。第13和14章专题讨论测试的充分性。
- **理念6：过程质量控制。**把软件开发过程比作为硬件生产过程。采用统计学的质量控制技术，力求代码生产达到6σ的要求。这样的理念导致了软件质量统计学方法，见第16章和20章将讨论质量控制和过程改进。

2.4使用者的错误与避免

- 2.4.1操作员的错误
- 2.4.2 人的信息处理模型
- 2.4.3 操作错误的避免

人件:
humanware/peopleware

《人件》第1版于 1987 年出版，专门讨论了软件开发和维护团队的管理问题，并向人们的传统认识提出了挑战。作者在书中推崇人本管理思想，正确指出知识型企业的核心是人，而不是技术，呼吁给予软件工作者充分的自由和信任。本书推出后，立即在西方引起了轰动，被誉为“几十年来对美国软件业影响最大的理念”。与《人月神话》一样，《人件》现已成为软件团队管理的经典之作。

操作员的错误

错误分类	错误原因	现实例子
配置错误	● 错误的配置导致系统复位缺省的配置 ● 配置文件偶然崩溃 ● 启动脚本忽略了必要的服务	2001 年 1 月，不正确的配置更改导致微软公司 DNS 络 microsoft.com、MSN.com、MSNBS.com、hotmail.c 和 Encarta.com 的中断。 http://www.internetnews.com/xSP/article.php/570171
规程错误	● 数据库备份故障 ● 回复错的备份 ● 忘记启动 Web 服务 器 ● 错误地删除了文件 ● 不正确的输入 ● 忘记删减日志文件(磁盘被占满)	美国密执根大学的 ICPSR Web2002 年 6 月服务器机。原因是数据增加太快，导致 Web 日志太大。 http://www.icpsr.umich.edu/org/policies/webouts-2002ml
综合错误	● 设备滑落造成的创伤 ● 意外断开电源	

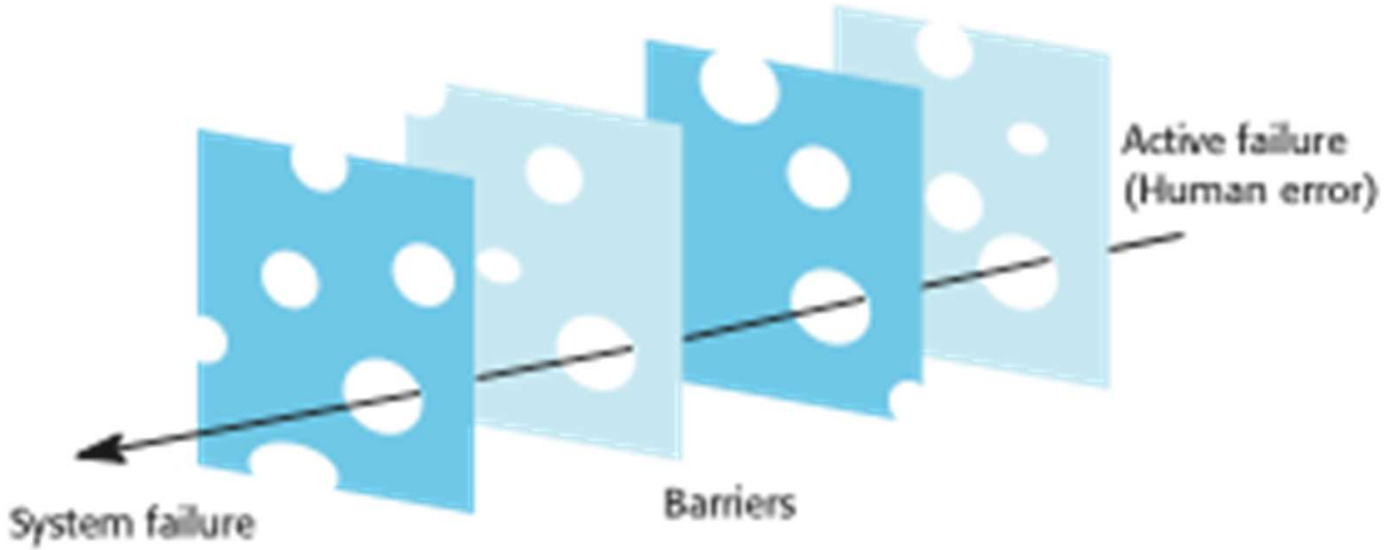
人的信息处理模型

- 1）**传感信息存储：**将物理现象(例如，光、声)转换为神经感知运动，并存储起来。
- 2）**模式识别：**将传感信息存储的物理代码转换为有意义的元素(符号转换为字母、再组成词)。一个物理代码可能映射为一个记忆代码，也可能映射成几个贮存的编码。
- 3）**判断/响应选择：**　存储在工作存储区的信息供未来使用，或者直接与其他信息结合，或者启动决策过程，做出响应。由于信息具有偶然性，因此能否依据此信息做出合理的响应是不确定的。
- 4）**执行：**这个阶段依据响应选择，把高层的响应分解为要求的听觉、运动和认知步骤。的执行情况再次反馈到传感器存储。

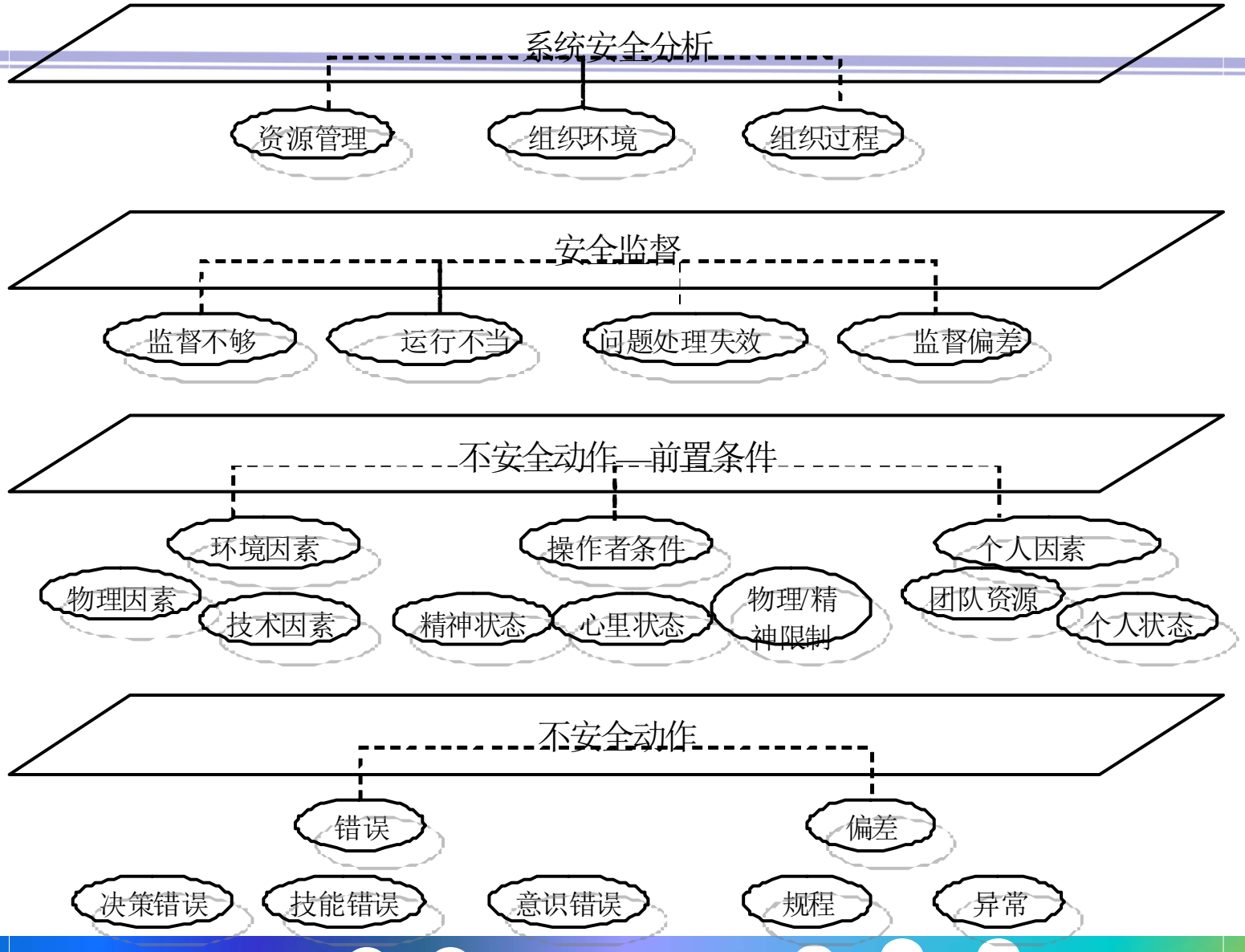
“瑞士奶酪”原理

- 瑞士奶酪是一种奶酪和面包片相互夹层的食品。只有当奶酪通过各面包层上的小孔时，上层的奶酪才会掉到地面。
- James Reason建议“瑞士奶酪”用于系统安全(safety)设计。Reason建立了描述人的错误的“瑞士奶酪”模型。在其模型中，认为应当从4个(面包)层面上分析事故的发生原因，并且这四个层面的因素从上到下产生影响：
  - 1）组织影响
  - 2）不安全的监督
  - 3）不安全动作的前置条件、以及
  - 4）不安全动作本身。

操作错误的避免---Reason's Swiss cheese model of system failure



瑞士奶酪


系统安全分析
资源管理
组织环境
组织过程
安全监督
监督不够
运行不当
问题处理失效
监督偏差
不安全动作—前置条件
环境因素
操作者条件
个人因素
物理因素
技术因素
精神状态
心里状态
物理/精神限制
团队资源
个人状态
不安全动作
错误
偏差
决策错误
技能错误
意识错误
规程
异常

案例1

德银行职员打盹压住键盘 误转出2亿欧元

- 中国网6月12日讯 据法国《费加罗报》6月10日报道，德国一银行职员在工作时不慎打了个盹儿，压住了键盘上的“2”键，将一笔62欧元的转账付费变成了一笔高达2.22亿欧元的交易，还造成了该雇员的一名女同事被“躺枪”
- 经过审理，黑森州地区的劳工法院责令银行撤回对被告的开除处罚。劳资调解委员认为，“肇事雇员在信息录入时睡着，压住了键盘的‘2’键，未能尽到确认转款信息的责任，因此酿成此大错，并非被告过失。”
- 值得庆幸的是，由于银行发现及时，这笔高达2亿元的巨款并未打入对方账户，未造成更多损失。
- http://finance.sina.com.cn/money/bank/bank_hydt/20130612/101015767985.shtml

案例2--光大证券“乌龙指”事件

- 光大证券乌龙 超低价卖出10年期国债
  - 2013-08-16收盘：沪指跌0.65%演过山车 诡异暴涨系光大证券操作问题
  - 2013-08-16早盘股市诡异暴涨 光大证券公告承认系统出错
  - 2013-08-16光大证券临时停牌 已申请交易作废
  - 2013-08-16光大证券：策略投资部门自营业务在使用其独立套利系统时出现问题
- 不同的反应：
  - 感谢光大证券乌龙指让我从中兴重工赢利出来
    - http://guba.eastmoney.com/news,601608,85141974.html
  - **“光大证券乌龙指事件带来了什么”**
    - 初中作文， *www.zww.cn/...html/261/743348.htm*
  - **光大证券乌龙指 股民上午拿到40万红包下午只剩2万**
    - http://news.ifeng.com/mainland/detail_2013_08/17/28719353_0.shtml

案例2--光大证券“乌龙指”事件

- 光大证券乌龙 超低价卖出10年期国债
  - 2013-08-19证监会对光大证券立案调查 定性为极端个别事件
    2013-08-18证监会：光大证券自营交易异常未发现人为差错
  - 2013-08-18上海证监局已决定先行对光大证券采取行政监管措施
  - 2013-08-18证监会：16日股市交易有效 对光大证券正式立案调查
  - 2013-08-188.16光大证券事件完整解析和影响评估
  - 2013-08-17光大证券面临三重质疑 涉赔偿和是否内幕交易问题
    2013-08-17牵动7807亿市值 光大证券敲出A股史上最大乌龙指
  - 2013-08-17 71只权重股瞬间封涨停 别太高兴，是光大证券摆大乌龙
  - 2013-08-17证监会回应股市瞬间暴涨事件：系光大证券自营账户大额买入

案例2--光大证券“乌龙指”事件

- 光大证券乌龙 超低价卖出10年期国债
  - 2013-08-19“乌龙指”主角杨剑波去年给光大证券挣了几个亿
  - 2013-08-19光大证券回应操纵市场：错单后做空单是本能 2013-08-19皮海洲:光大证券事件投资者面临索赔难
  - 2013-08-19光大证券估值遭下调 基金再次“躺着中枪” 2013-08-19盘前：光大证券事件继续发酵 股指压力不容小觑
  - 2013-08-19证监会立案调查光大证券乌龙事件 索赔难度大
  - 2013-08-19“乌龙指”蝴蝶效应 投资者遭遇横祸谁来赔
  - 2013-08-19光大证券今日继续停牌一天 8月20日复牌交易
  - 2013-08-19中金所：对光大证券自营业务股指期货限制开仓
  - 2013-08-19光大证券公告难释质疑 两大问题待解
  - 2013-08-19A股惊天“乌龙”背后：“内控之殇”伤了谁？
  - 2013-08-19光大证券公告称8·16事件致当日损失约1.94亿
  - 2013-08-19异常交易非乌龙指 证监会:光大证券系统存缺陷

错误预防措施

- 1）操作者：将错误看作是操者的责任，避免人的“不安全动作和行为”。
  - 通常的办法是建立训练规程，对操作者进行培训和训练。通过更严格训练程序，避免错误系统运行和维护过程中错误的发生。
- 2）系统设计。假设人是会犯错误的。
  - 系统设计错误会导致系统错误的工作方式。
  - 运行和维护系统的组织管理制度也会影响到系统的操作人员，从而引起操作错误。
  - 良好的系统设计应当可以识别出人的错误，并允许系统失效前进行自我修复。
  - 不应当把系统失败的责任完全归结于操作人员，而是系统设计者或开发者的责任。

错误预防措施

- 3）运行和操作规程。一旦系统交付使用，从组织到个人按规程进行系统的运行和操作。
  - 而使用和操作规程的制定也是系统开发者和用户等相关方的主要责任之一。
- 4）缺陷预防体系。系统的设计者、开发者、用户必须从先前的失败教训中吸取经验，从而在新系统设计或旧系统的进化中预防类似问题的发生。即，建立闭环的预防体系。
  - 特别地，软件系统的错误都是人为制造的。
  - 基于缺陷预防的设计和开发是避免软件错误的重要手段。

2.5 总结

- 当今的世界，计算机无处不在。都离不开基于计算机构造的系统。在这样的系统中，硬件(HardWare)、软件(SoftWare)、以及人件(PeopleWare)等混合在一起，形成了人-机-软件混合系统。
- 这些系统中的各种“件(ware)”的发生故障机理的不同，导致了系统的分析、设计、开发上需要采取不同的措施和对策。从而降低系统出现故障的概率，提高整个系统的可靠性和稳定性。
- 软件的故障机理不同于硬件，必须在软件设计和开发中采取有效的方法降低软件的缺陷，从而降低软件运行后的故障率，应对软件无处不在所带来的问题。

Homework

- 上网查找相关行业的计算机故障案例，按软件、硬件、人工操作、其他等因素分析出现故障的原因。
  - 要求：
    - 1）故障案例不得少于5个；
    - 2）硬件、软件、人故障各至少一个
    - 3）准确标注出来源；
    - 4）写出故障原因分析报告