嵌入式系统设计

翁恺

2016春

嵌入式系统的特性

并行、响应式行为

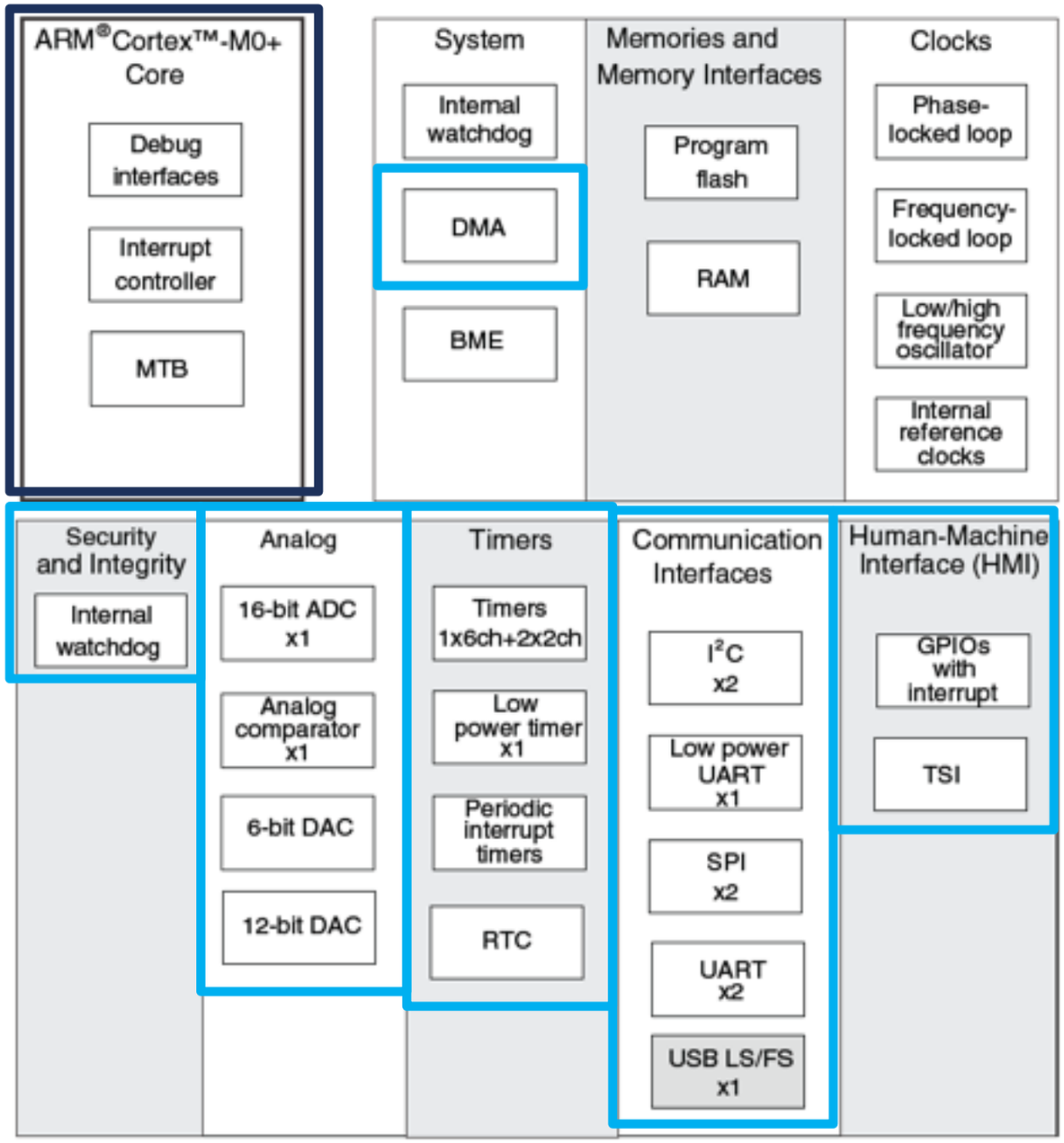
- 始终运行
- 必须响应连续和混合的事件
- 实时系统对于响应有底限要求
- 通常必须并行处理多个独立的任务

嵌入式软件架构模型

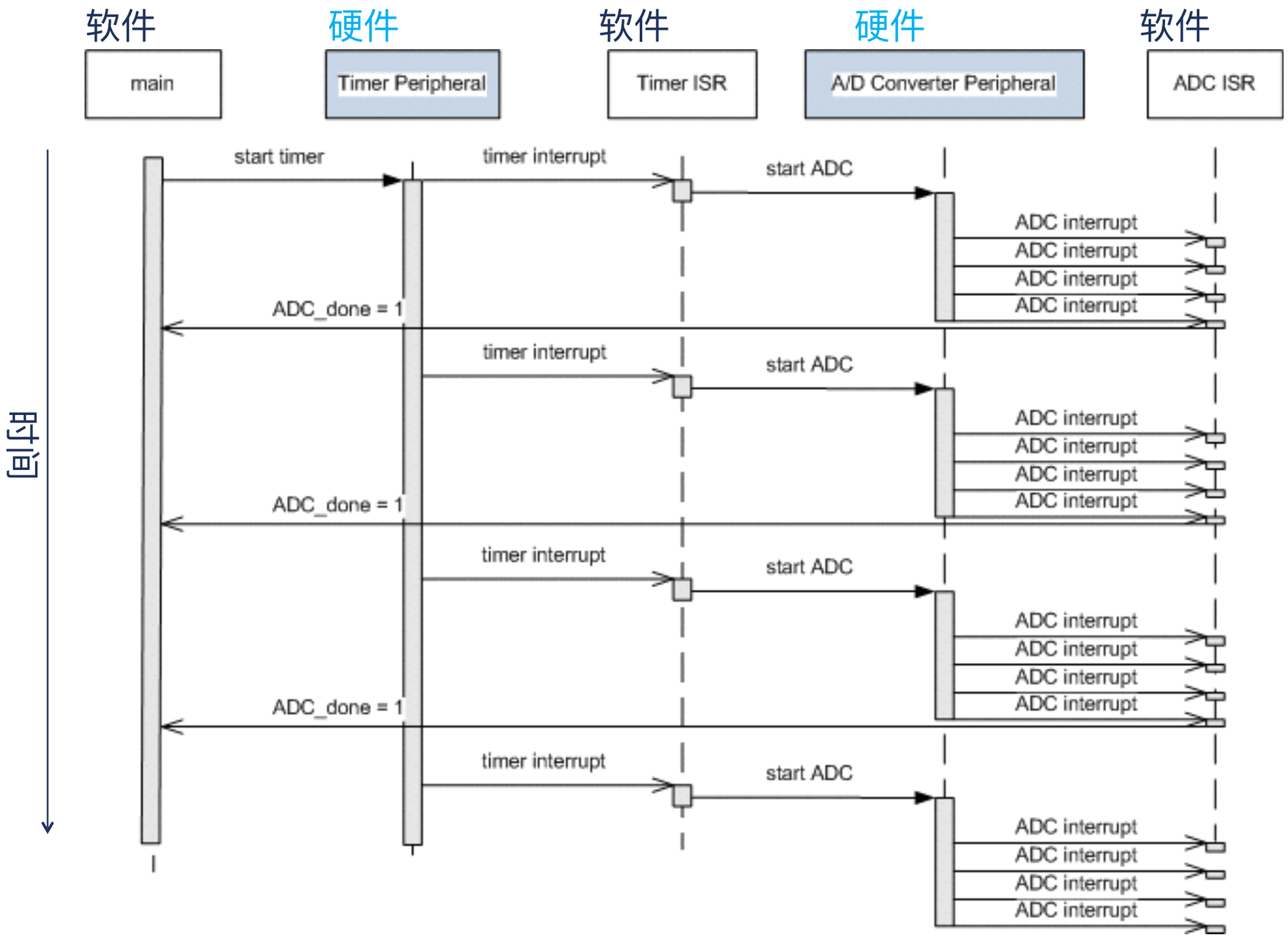
- 桌面、BS有各自独特的架构模型
- 嵌入式也有自己的架构模型
- 裸机、RTOS、Linux所体现的不仅仅是基础平台的不同，更重要的是架构模型的不同
- Linux的并行性是建立在进程/线程调度基础上的
- 嵌入式系统的并行性是硬件/软件协同实现的

支持并行的MCU硬件

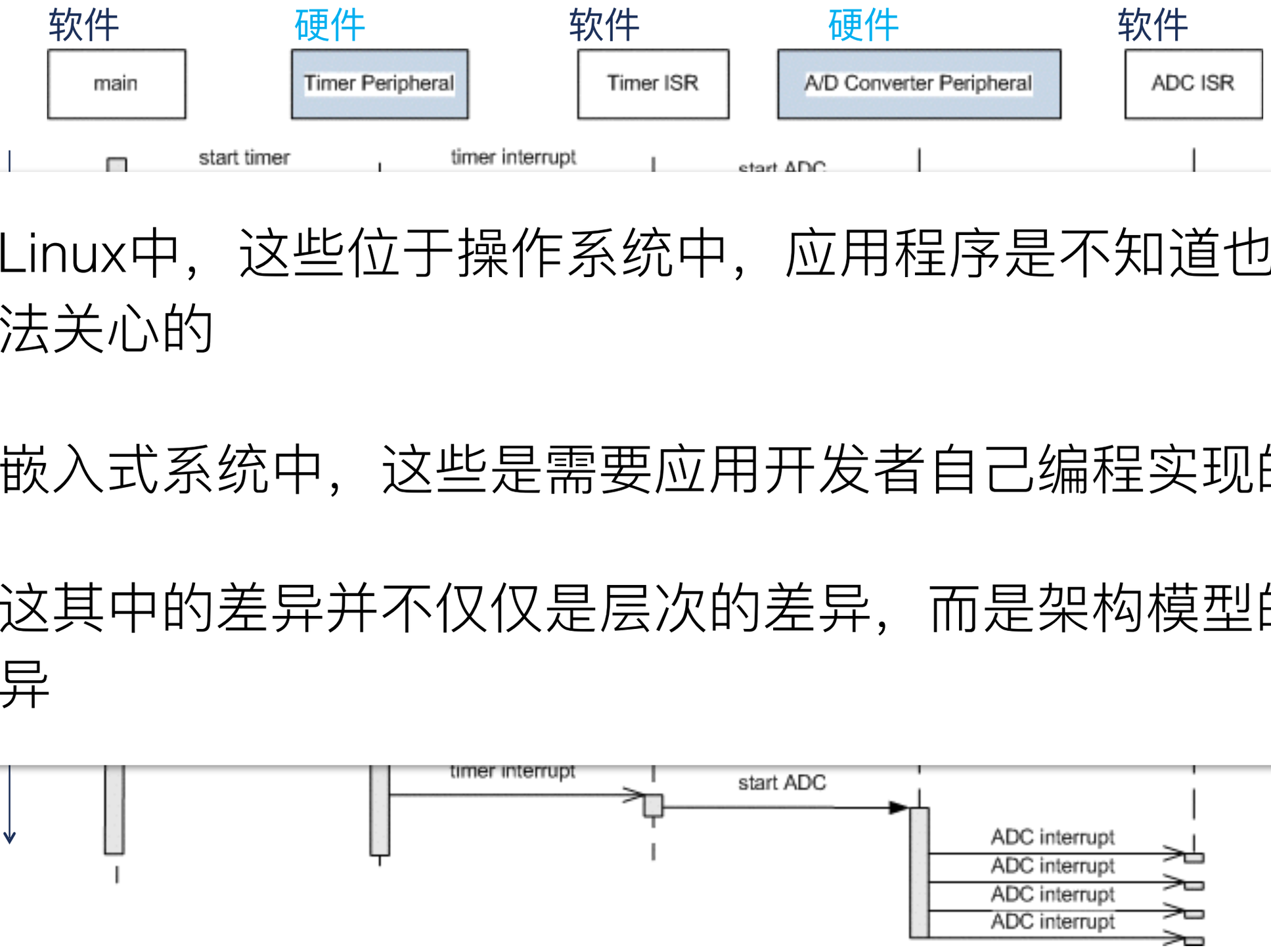
- CPU执行一个或多个线程的指令
- 特殊的硬件外围部件实现专门的并行处理
  - DMA – 在内存和外围器件之间传输数据
  - ADC
  - 通信
  - 定时器及定时动作
  - 外部事件
    - 外围部件用中断来通知CPU事件的发生



并行的硬件和软件操作


软件
硬件
软件
硬件
软件
main
Timer Peripheral
Timer ISR
A/D Converter Peripheral
ADC ISR
start timer
timer interrupt
start ADC
ADC interrupt
ADC interrupt
ADC interrupt
ADC interrupt
ADC_done = 1
timer interrupt
start ADC
ADC interrupt
ADC interrupt
ADC interrupt
ADC interrupt
ADC_done = 1
timer interrupt
start ADC
ADC interrupt
ADC interrupt
ADC interrupt
ADC interrupt
ADC_done = 1
timer interrupt
start ADC
ADC interrupt
ADC interrupt
ADC interrupt
ADC interrupt
时间

并行的硬件和软件操作



- 在Linux中，这些位于操作系统中，应用程序是不知道也无法关心的
- 在嵌入式系统中，这些是需要应用开发者自己编程实现的
- 但这其中的差异并不仅仅是层次的差异，而是架构模型的差异

长期运行

- 失效处理
  - 许多系统必须独立运行很长时间，需要系统处理可能的失效而不会崩溃
  - 通常失效处理代码比正常情况的代码更大更复杂
- 诊断手段
  - 需要能帮助维修人员快速判断问题的手段

其他设计约束

- 成本
  - 竞争性的市场惩罚不能产生与其成本相衬的效益的产品
- 大小和重量限制
  - 移动（航空、汽车）和便携（比如手持）系统
- 功率和能耗限制
  - 电池容量
  - 散热限制
- 环境
  - 温度范围可能是-40°C到 125°C，甚至更厉害

设计约束的影响

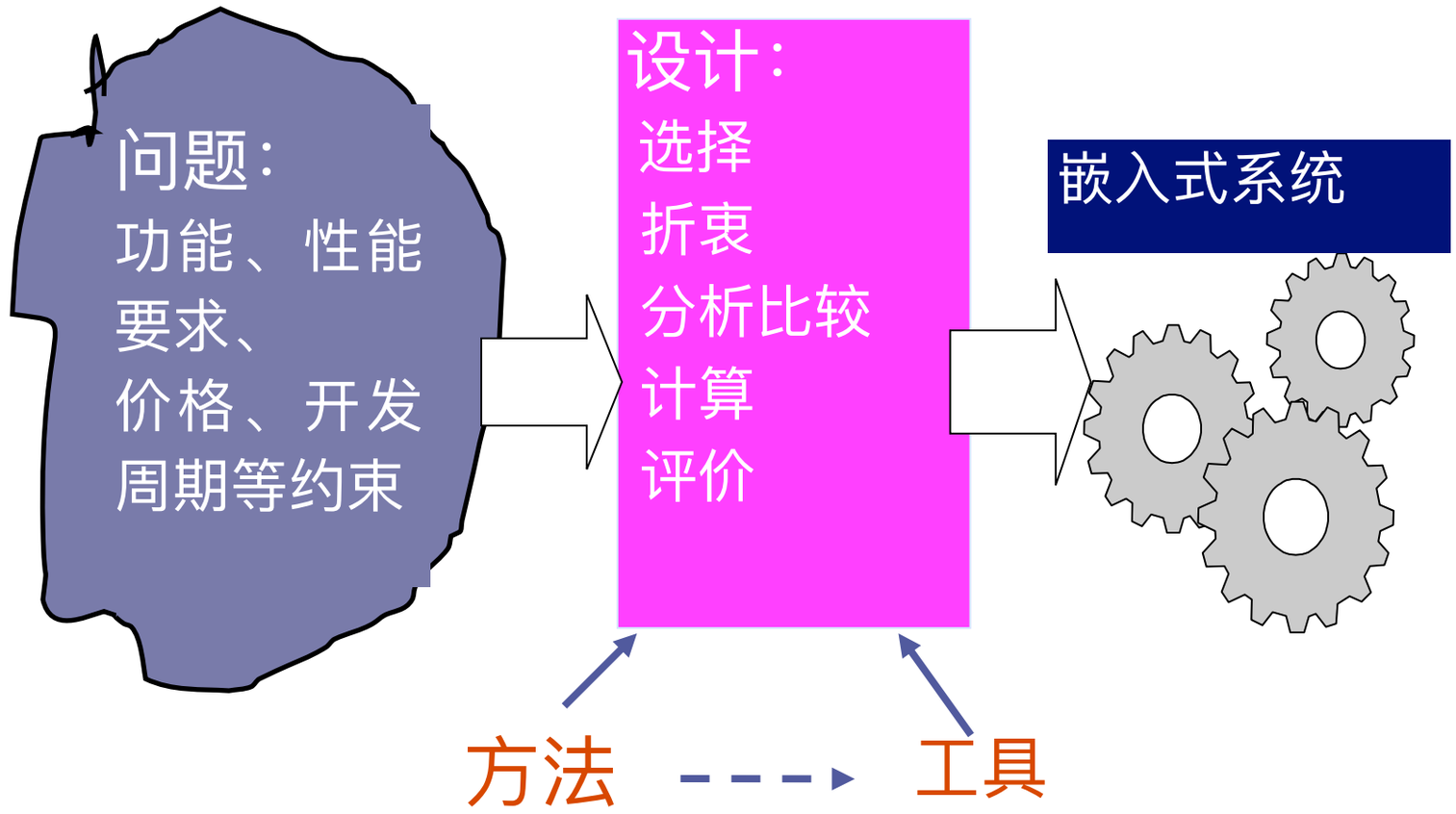
- 所用的单片机（而不是微处理器）
  - 包含了与其他器件接口的外围部件，能高效地做出响应
  - 片内RAM、ROM降低了电路板的复杂度和成本
- 编程语言
  - 用C编程而不是Java（代码更小、更快，因此可以用更便宜的MCU）
  - 有些性能要求严苛的代码可能是汇编的
- 操作系统
  - 通常没有OS，但是会有简单的调度器（或只是中断＋主代码——前/后台系统）
  - 如果用了OS，很可能倾向于使用RTOS

嵌入式系统设计

做出够好的系统还要够快

- 如何能做出够好的软件，而不至于破产?
  - 需要能有效地开发（及测试）软件
- 遵循好的计划
  - 从客户的需求开始
  - 设计体系结构来定义系统的构件（任务、模块等）
  - 加上遗漏的需求
    - 失效检测、管理和日志
    - 实时性问题
    - 与固件标准手册的兼容性
    - 故障安全保障
- 做出详细设计
- 遵循良好的开发过程来实现代码
  - 经常做设计和代码的评审
  - 经常做测试（单元和系统测试，最好是自动进行的）
  - 用版本控制来管理变更
- 事后总结来改进开发过程

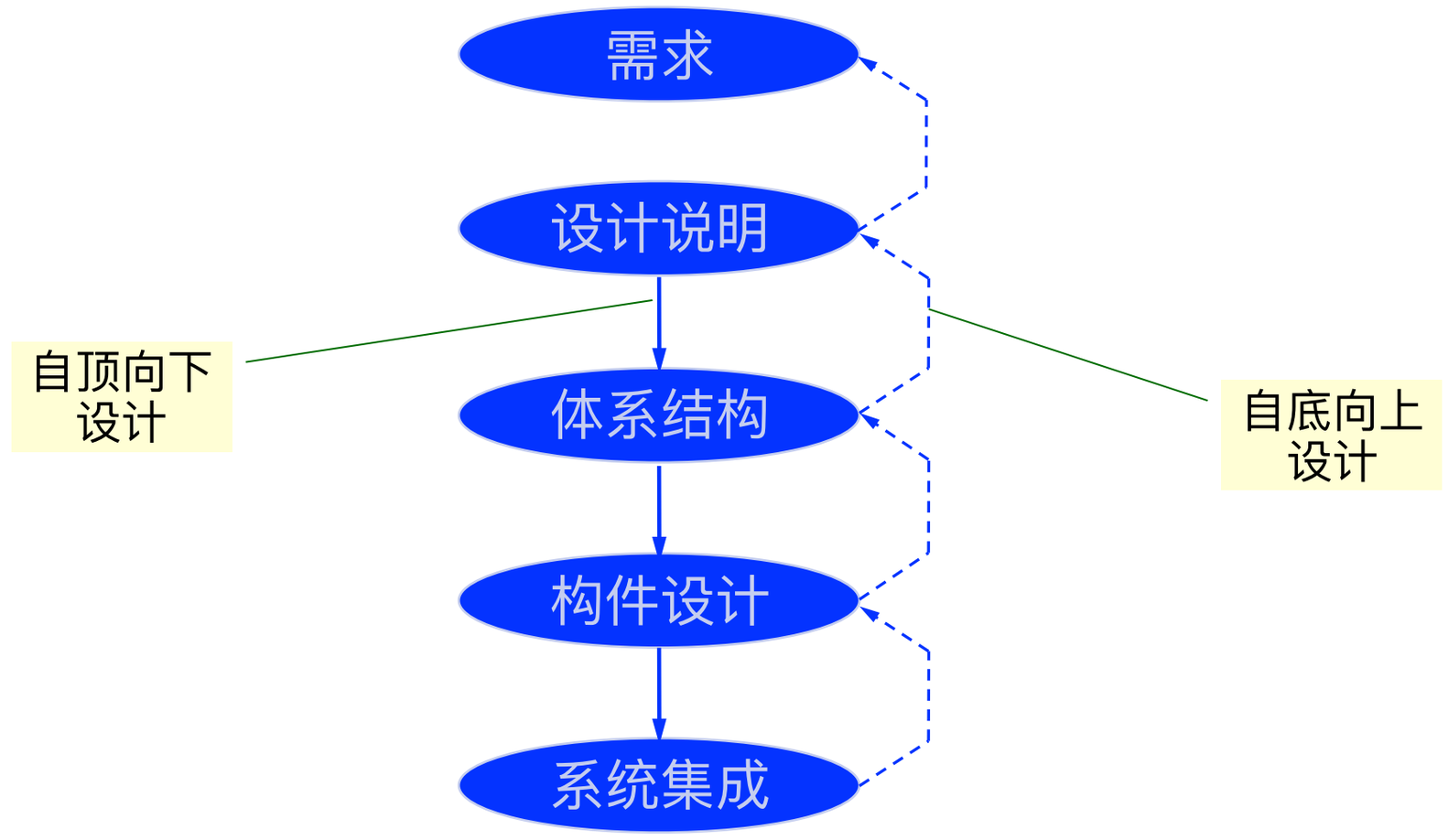
嵌入式系统的设计



自顶向下还是自底向上

- 任何企业都不会是从零开始做新产品的
  - 体系是承袭已有的
  - 构件是利用已有的

自顶向下还是自底向上



- 任何企业都不会是从零开始做新产品的
  - 体系是承袭已有的
  - 构件是利用已有的

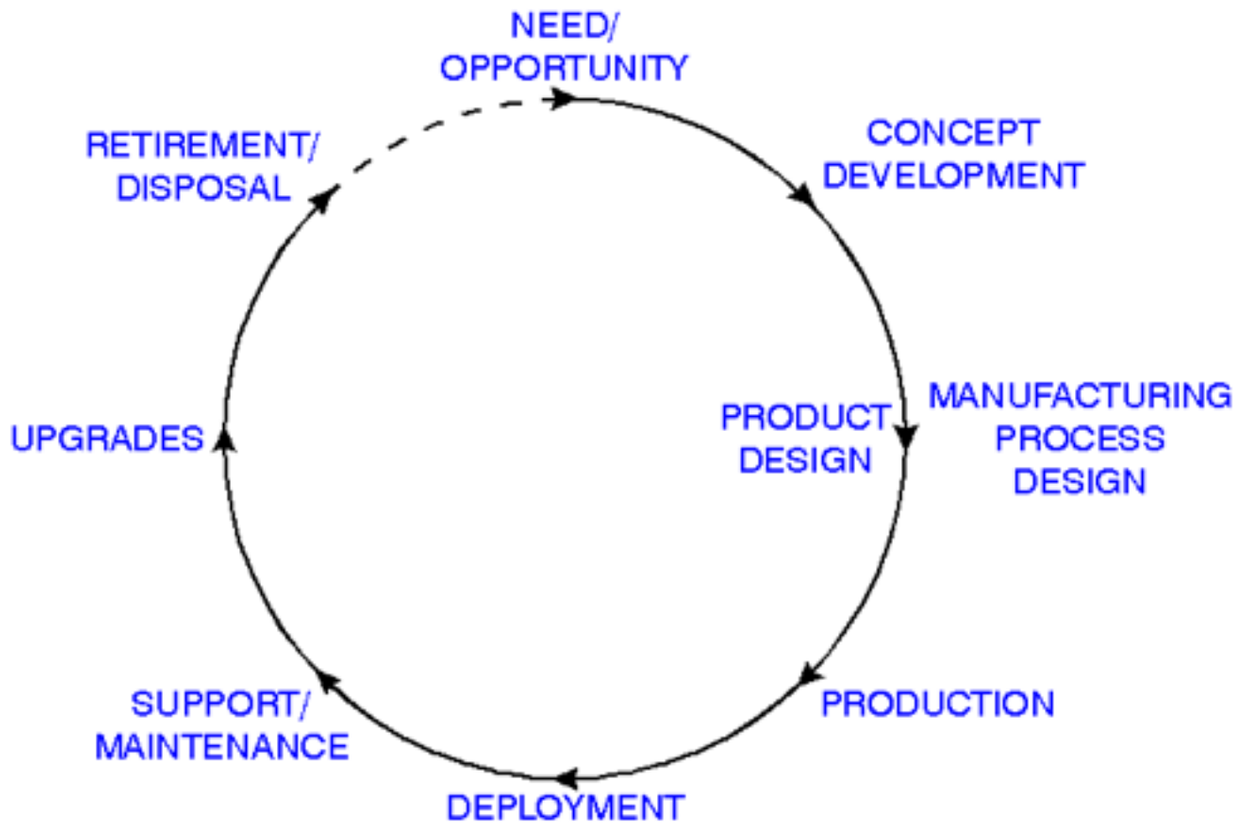
当计划遇上现实

- 我们希望有一个强壮的计划，考虑到了可能的风险
    - 如果代码比预期的复杂很多怎么办?
    - 如果代码（或库）里有错误怎么办?
    - 如果系统没有足够的内存或吞吐能力怎么办?
    - 如果系统太贵了怎么办?
    - 如果主要开发人员离职了怎么办?
    - 如果主要开发人员不胜任、懒惰，或又笨又懒（还不肯离职）怎么办?
    - 如果团队中其他人生病了怎么办?
    - 如果客户增加了新的要求怎么办?
    - 如果客户想要提前两个月交货怎么办?
- 成功的工程有赖于很多因素的平衡取舍，很多因素都不是技术的!

软件生命周期概念

- 编程是软件开发最显而易见的阶段，但不是唯一的
- 在可以编程之前，必须知道
  - 代码必须做什么？需求定义
  - 代码要如何组织？设计定义（只有在这之后才开始写代码）
- 如何能知道代码是可用的？测试计划
  - 在定义需求的时候就做计划是最好的
- 软件可能会随着时间的推移而变化——大量的顺势的修改的维护工作！
  - 纠错、适应、增强和预防性维护

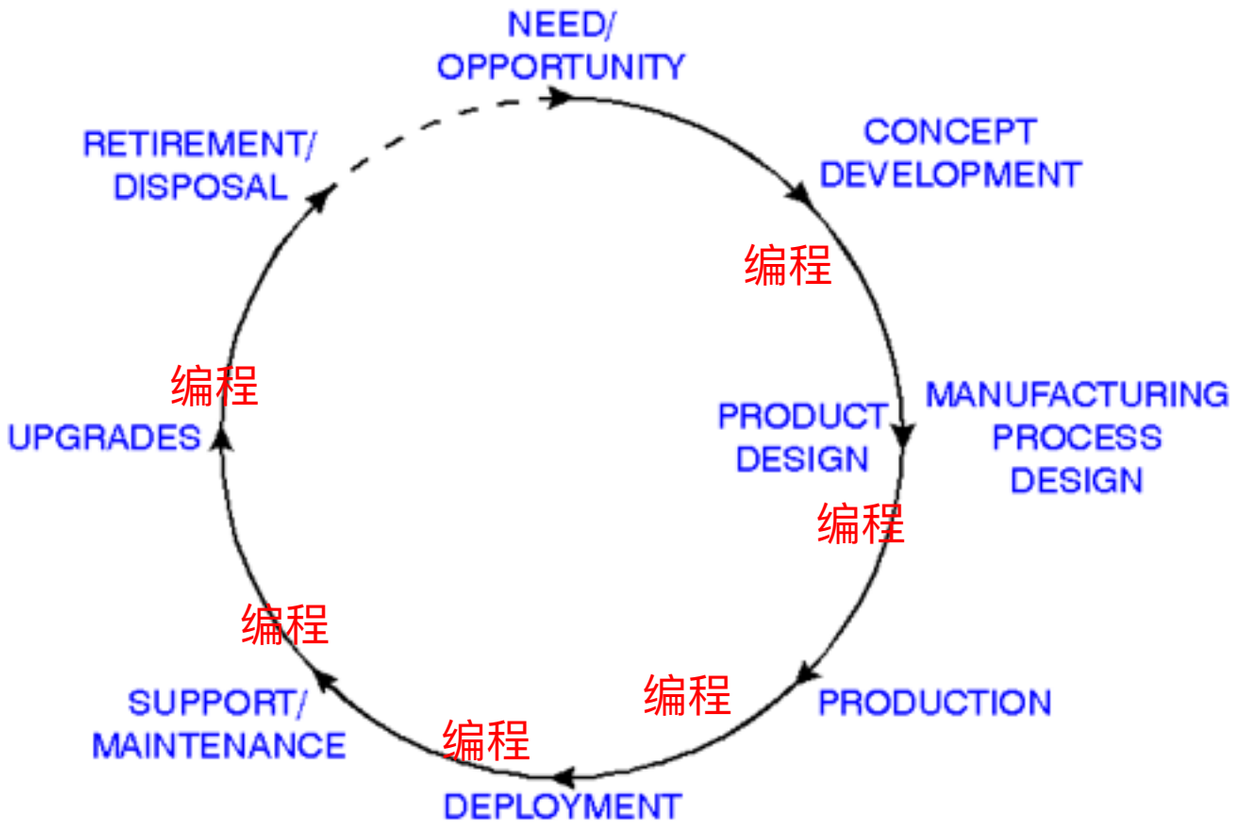
产品开发生命周期



绘图：Phil Koopman, Carnegie Mellon University

- 在整个代码开发和修改的过程中，值得多加努力来简化代码开发工作：理解、维护、改进和测试

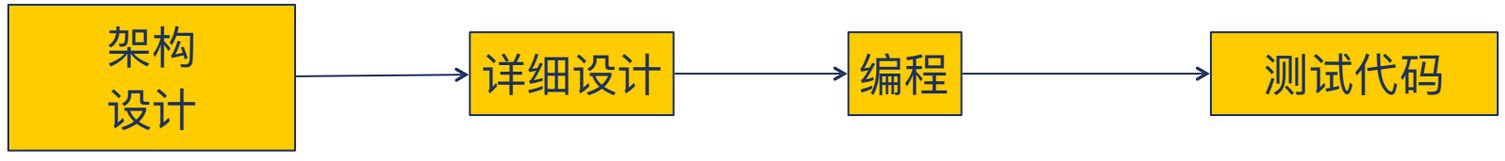
产品开发生命周期



绘图：Phil Koopman, Carnegie Mellon University

- 在整个代码开发和修改的过程中，值得多加努力来简化代码开发工作：理解、维护、改进和测试

先设计再编程



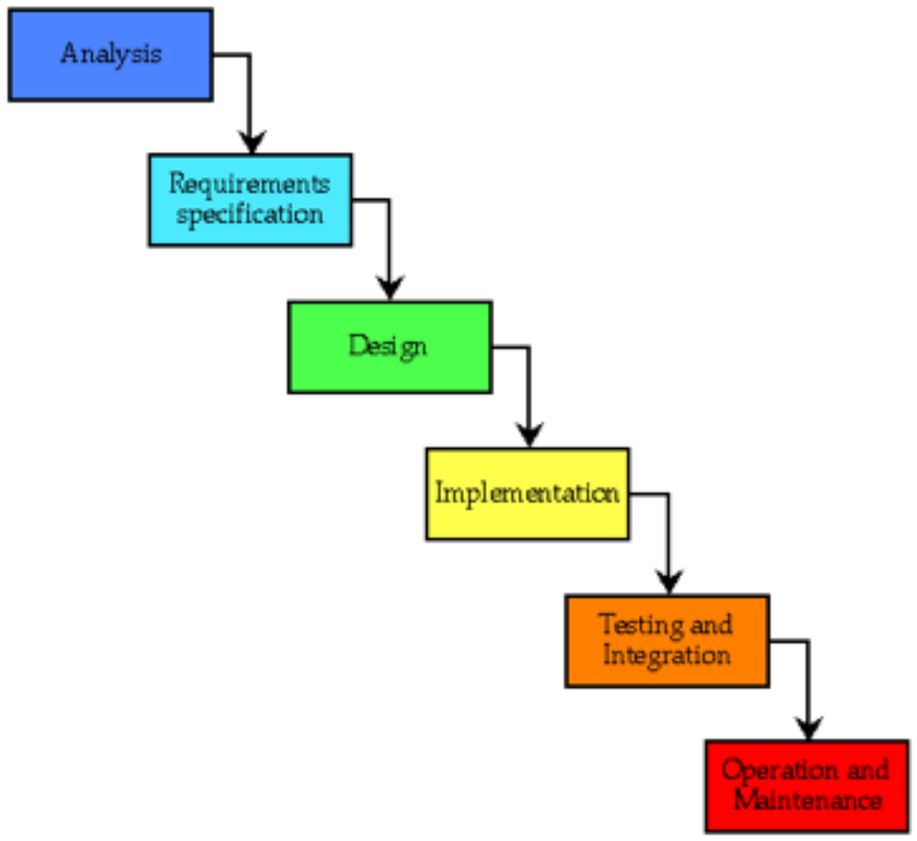
- 低估了所需软件的复杂度是非常常见的风险
- 写代码使你囿于特定的实现中
  - 太早开始会让你向隅而泣
- 先设计再编程的好处
  - 预先看到系统的复杂程度，对投入能做出更精准的估计和计划
  - 用设计图而不是代码来讨论系统应该做什么和怎么做
  - 可以在文档中使用设计图来简化代码的维护，降低员工换岗的风险

开发模式

- 考虑开发风险的量
  - 新的单片机?
  - 例外的需求（吞吐量、功率、安全认证等等）
  - 新产品?
  - 新客户?
  - 需求变更?
- 根据风险选择模式
  - 低：可以做详细规划，做完整的事先设计，瀑布式
  - 高：用交互或敏捷的开发模式，螺旋式，先做高风险部分的原型

瀑布（理想化的）

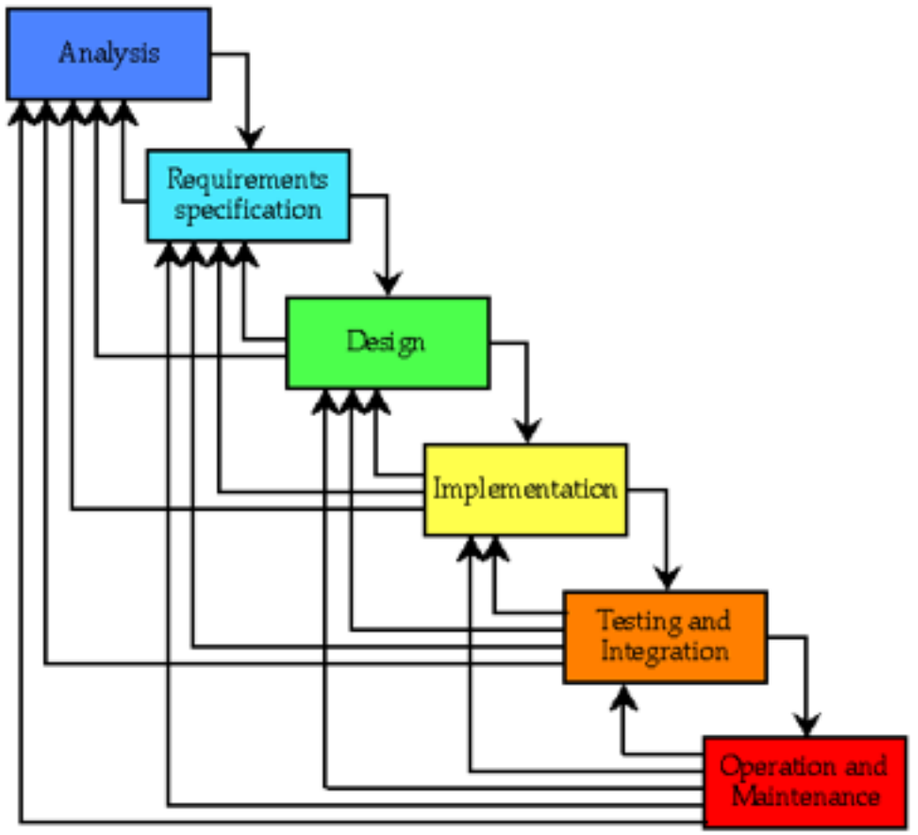
- 规划好工作，然后照规划来工作
  - BUFD: Big Up-Front Design
    事先规划好的设计
- 这个模式意味着我们和客户都
  - 预先知道所有的需求
  - 知道部件之间所有的交互
  - 知道要花多久来写软件和调试软件



绘图: Jon McCormack, Monash University

瀑布（实际的实现）

- 现实：我们并非无所不知，所以存在大量的返工



绘图: Jon McCormack, Monash University

设计

GPS手持地图终端设备

- 目的：移动离线地图上的实时定位
- 输入：GPS数据、两个控制按钮
- 输出：5寸LCD
- 性能：1秒更新当前地图一次
- 成本约束：300元
- 重量约束：可以装在自行车把手上
- 电源约束：可以连续使用4小时

设计要决定什么?

- 满足功能性能需要要用什么软件硬件
- 首先是软件，因为软件会提出隐含的硬件需求
  - 显式的硬件需求：LCD、GPS、电池、按键输入
  - 隐式的硬件需求：CPU？RAM？ROM?

GPS地图需要什么?

- GUI如何实现?
  - 做桌面不用考虑，因为图形界面就在哪里
  - 嵌入式做图形有四种可能:
    - 上一个桌面系统，如Linux
    - 上一个手机/平板系统，如安卓
    - 上一个嵌入式专用GUI库，如MiniGUI
    - 自己写一个GUI库

嵌入式图形界面

- 不同的软件方案带来不同的硬件需求
- 不同的硬件架构导致不同的软件成本
  - 桌面系统，如Linux
  - 手机/平板系统，如安卓
  - 嵌入式专用GUI库，如MiniGUI
  - 自己写一个GUI库

GPS移动地图的扩展

- 支持在线地图
- 支持在线地图下载成为离线地图
- 实现通信
  - 3G/4G？Wi-Fi？蓝牙？对讲机？
  - 采用何种通信手段取决于通信的目的
    - 车队？与家/大本营的通信？救援？
    - 汽车用？自行车用？户外徒步用？登山用？

系统结构设计

- 系统如何实现设计说明书描述的功能
- 软件/硬件如何进行划分
- 嵌入式系统中软件和硬件协同完成系统的功能
- 软件/硬件划分通常由速度、灵活性以及开销来决策

软硬件的划分

- 嵌入式系统的设计涉及硬件与软件部件，设计中必须决定什么功能由硬件实现，什么功能由软件实现。
- 硬件和软件具有双重性
- 软硬件变动对系统的决策造成影响
- 划分和选择需要考虑多种因素
- 硬件和软件的双重性是划分决策的前提

通常由软件实现的部分

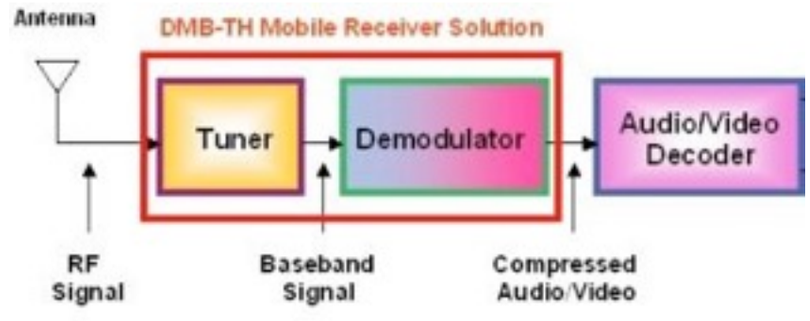
- 操作系统功能
  - 任务调度
  - 资源管理
  - 设备驱动
- 协议栈
  - TCP / IP
- 应用软件框架
  - 除基本系统、物理接口、基本逻辑电路，许多由硬件实现的功能都可以由软件实现。

双重性部分

- 算法
  - 加密 / 解密
  - 编码 / 解码
  - 压缩 / 解压
  - ……
- 数学运算
  - 浮点运算，FFT，…
  - ……
- 按键去抖动
  - 硬件：积分电路
  - 软件：判断算法
- 无线电设备
  - 硬件：窄带调谐、变频、中频放大滤波、检波解调
  - 软件：宽带入口、高速AD、直接求解译码

双重性部分

- 算法
  - 加密 / 解密
  - 编码 / 解码
  - 压缩 / 解压
  - ……
- 数学运算
  - 浮点运算，FFT，…
  - ……

- 按键去抖动
- 

- 无线电设备
  - 硬件：窄带调谐、变频、中频放大滤波、检波解调
  - 软件：宽带入口、高速AD、直接求解译码

嵌入式开发的特殊性

- 软件是基于具体而且特殊的硬件的
- 在硬件完成之前很可能无法开展软件工作
  - 采用通用硬件，如Linux机器
  - 采用核心硬件，如核心板、评估板
  - 采用虚拟技术

软硬件分配的权衡

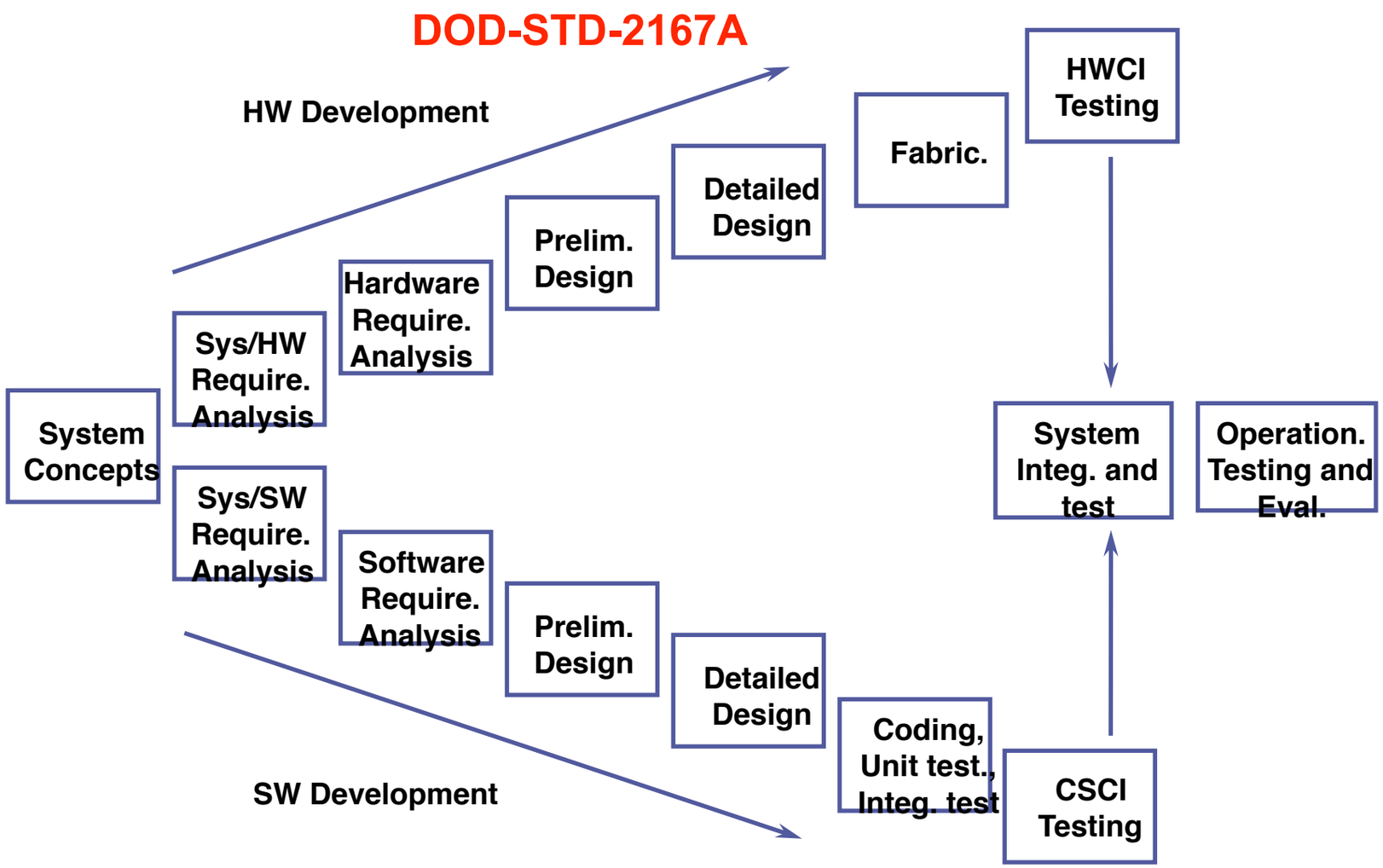
- 增加硬件实现的比重往往意味着更好的性能
  - 或降低软件开发的时间（成本）
    - 如采用Linux，则软件开发和桌面无差异
      - 但是同时也增加产品成本
- 增加软件实现的比重往往意味着更长的开发周期
  - 但是降低了产品成本
- 可以采取短期以较高的硬件成本产品先抢占市场，再行长期研发低成本硬件

软硬件协同开发

嵌入式产品开发的过程

- 硬件：设计电路、PCB、结构件、生产工艺、包装
  - 在作出最终产品的板子之前，一般起码需要两个版本的硬件
    - 在第一个版本硬件出来之后才能写软件
    - 用现成的核心板加外围元件可以快速做一个可以写软件的硬件原型出来
- 软件：写程序

传统的嵌入式系统设计模型



传统的嵌入式系统设计过程

- 传统软硬件设计过程的基本特征：
  - 系统在设计一开始就被划分为软件和硬件两大部分
  - 软件和硬件独立进行开发设计
    - “Hardware first” approach often adopted
- 隐含的一些问题：
  - 软硬件之间的交互受到很大限制，软硬件之间的相互性能影响很难评估
  - 系统集成相对滞后，NRE（Non-Recurring Engineering cost）大
- 因此：
  - 设计修改困难、研制周期无法有效保障

传统设计过程中的尖锐矛盾

- 随着设计复杂程度的提高，软硬件设计中的一些错误将使开发过程付出昂贵的代价
- “Hardware first” approach often compounds（混合） software cost because software must compensate for（补偿） hardware inadequacies（不充分）

软硬件协同设计

- 一个方案是用HDL语言和C语言进行系统描述并进行模拟仿真和系统功能验证；

  - 对软硬件实现进行功能划分，分别用语言进行设计并将其综合起来进行功能验证和性能预测等仿真确认(协调模拟仿真)；

    - 如无问题则进行软件和硬件详细设计

- 更实际的方案是让软件和硬件人员互相了解对方的设计手段、能力和局限，在尽可能早的阶段开始互相服务

  - 硬件为软件提供核心板和原型板，从而能开始写程序

  - 软件为硬件展示核心算法和基本操作，从而理解对性能、容量、接口的要求

嵌入式图形界面

- 不同的软件方案带来不同的硬件需求
- 不同的硬件架构导致不同的软件成本
  - 桌面系统，如Linux
  - 手机/平板系统，如安卓
  - 嵌入式专用GUI库，如MiniGUI
  - 自己写一个GUI库

重视通用件的作用

- 传统制造业很早就发现通用件对于降低成本、简化设计的重要作用
    - 所有的产品都用螺丝，但是没有一个产品公司设计和生产螺丝
- 嵌入式硬件和软件都存在通用件

通用件

- 标准（通用）构件
  - 已经产品化
  - 形成规模生产
- 标准构件 ＋ 自行设计构件 ＝ 用户系统
  - 构件包括了硬件构件和软件构件
  - 构件本身可以是层次性的，可以由子构件组成

标准硬构件

- IC：集成电路
- PCB：核心板、接口板
- IP：Intellectual Property
  - 标准 IC
    - CPU, DSP, ……
    - RAM, ROM, 接口控制器
    - ASIC,……
  - 标准 IP
    - CPU核，……
  - 标准模块
    - GPRS模块,GSM模块,蓝牙模块, ……
    - 显示模块，……
  - 标准计算平台
    - 嵌入式Linux机
    - 安卓板

标准软构件

- OS / RTOS
- 协议栈
  - TCP/IP
  - 路由协议
  - H.323
- 图形开发包
  - MiniGUI
  - VxWorks的ZINK
- 驱动程序

企业积累

- 企业自己做的标准构件（包括硬件和软件）是企业重要的知识积累
- 没有企业是从零开始做产品的

作业：GPS地图设备方案设计

- 根据课堂上所介绍的GPS手持地图终端，给出你的设计方案，要求至少涉及：
  - 功能性能描述
  - 硬件组成和软件组成
  - 功能如何由软硬件实现