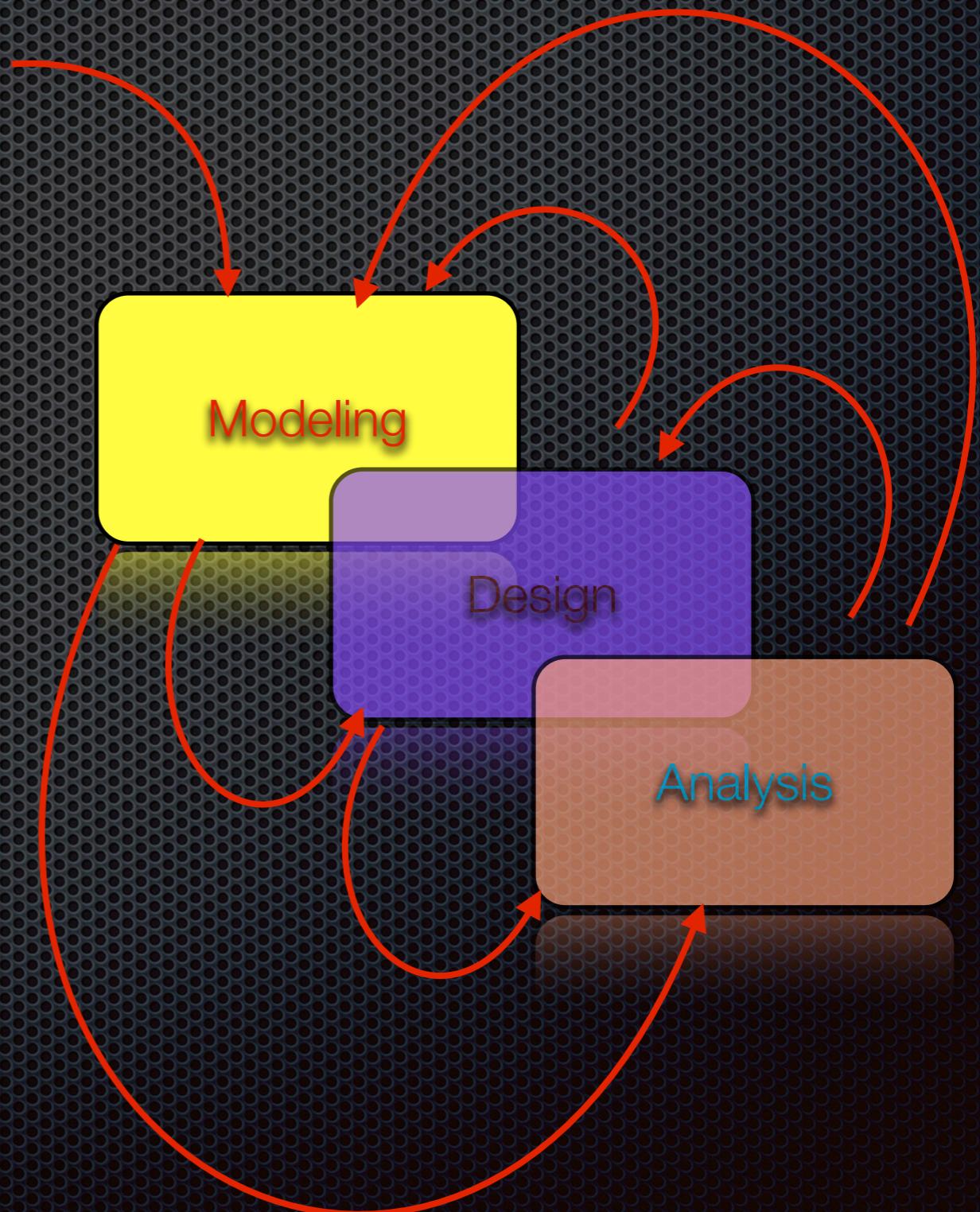


嵌入式系统建模

建模、设计、分析

- 建模是通过模拟加深对系统理解的过程。
- 模型是对系统的模拟，并反映系统的特性，模型指定了系统的功能。
- 设计是创建系统的结构。指定系统如何完成其功能。
- 分析是通过剖析获得对系统更深入理解的过程。具体说明系统为什么能够完成其功能(或者无法完成模型认为其能完成的功能)。



建模

- 模型是对所研究的系统、过程或概念的一种表达形式
- 使用模型的目的是要给出系统的抽象视图，每个模型都表示一组对象以及这些对象之间的相互关系
- 例如，数学模型是以一组定义和数学公式的形式建立的模型

模型应具备的基本特征

- 简单
- 经得起理论检验
- 高表现力
 - 简洁的表示，可以提高生产率
- 提供逻辑推理能力
- 可执行
 - 仿真/验证
- 可综合
 - 通常要求设计正交性
- 能够适应各种不同的任务
- 描述不存在歧义，易于理解易于修改

常见的系统建模技术

- 面向状态的建模
 - 有限状态机Finite State Machines
- 面向活动的建模
 - 数据流图/过程模型，把系统描绘成一组与数据或者执行的相关性有关的活动的集合
- 面向结构的模型
 - 框图
 - 着重强调系统的物理构成
- 面向数据的模型
 - 实体-关系图
- 异构模型
 - 综合前四种模型特征
 - 表达复杂系统的多种不同视图

基于模型的设计

- 创建嵌入式系统各个部分的数学模型
 - 物理世界
 - 控制系统
 - 软件环境
 - 硬件平台
 - 网络
 - 传感器和执行器
- 从模型构造实现
 - 目标：构造自动化，就像编译器一样
 - 实际上，只有部分是自动构造的

建模技术

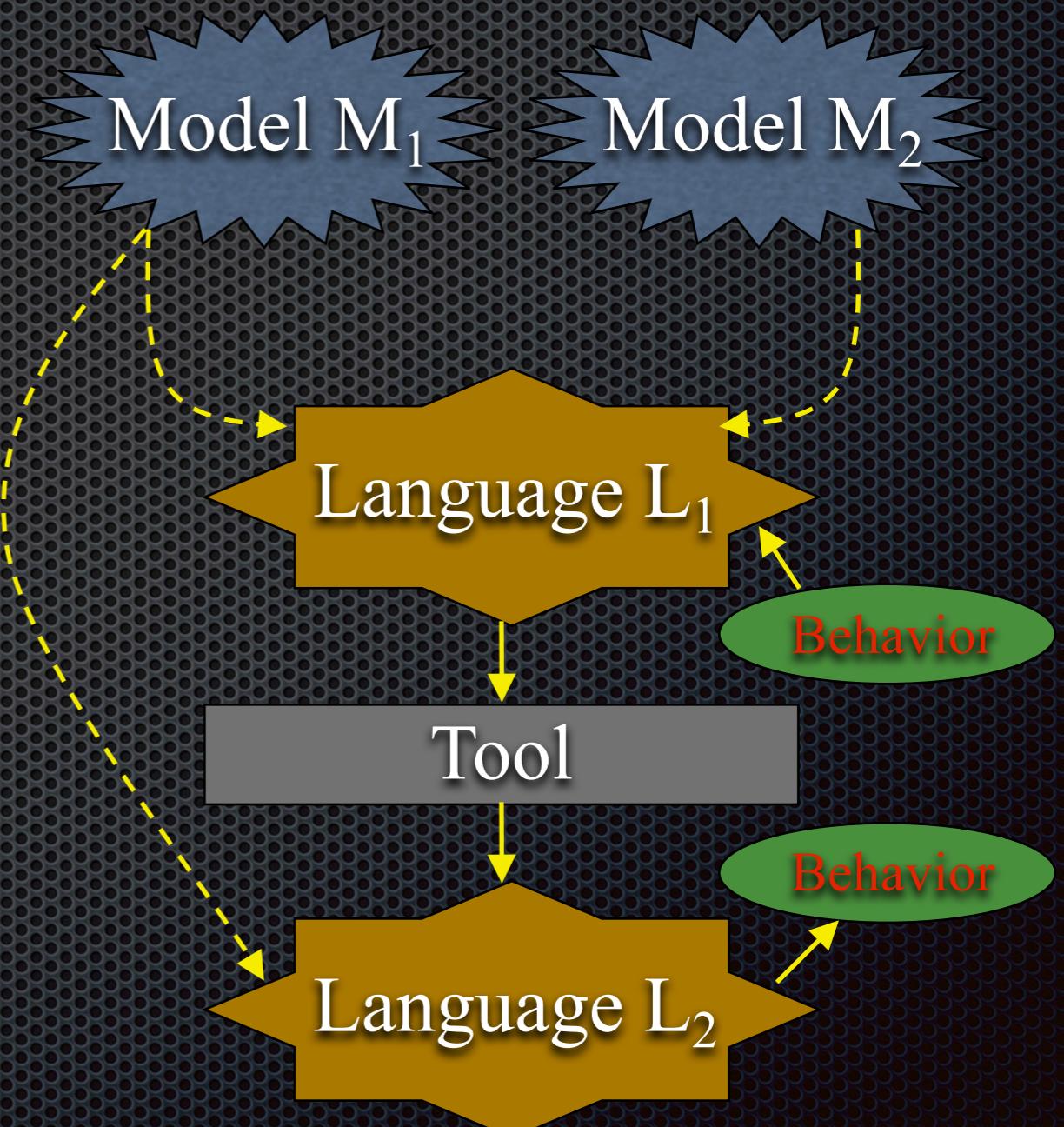
- 系统动态的抽象模型
 - 动态是时间上的演化，其状态如何改变
- 例子：
 - 建模物理现象 – ODE常微分方程
 - 反馈控制系统 – 时域建模
 - 模态行为建模 – FSMs, 混合自动机
 - 传感器和执行器建模 – 标定, 噪声
 - 软件建模 – 并发, 实时模型
 - 网络建模 – 延迟、错误率、丢包

分类

- 周期性与事件驱动
 - 周期性的例子：流式应用、处理器……
 - 事件驱动：反应式(由环境中的变化触发)
- 实时
 - 硬实时、软实时
- 控制、数据、通信
- 并行，地理距离
 - 高度并发与顺序执行；分布式系统
- 确定性与非确定性
 - 可预测性的程度

模型、语言与工具

- 计算模型用于捕获系统行为：
 - 一组对象
 - 组合的规则
 - 执行语义
- 语言定义了描述计算模型的语法
- 工具是将一种语言描述的模型转换为另一种语言描述的模型的“编译器”



嵌入式系统模型的用途

- 通过使用现代建模软件工具，可以离线仿真方式进行设计和执行初步验证
- 然后，可以使用模型来作为所有后续开发阶段的基础
- 建模（涉及硬件原型设计）将降低出错风险，通过在整个开发过程中执行验证和确认测试来缩短开发周期
- 以系统模型为基础，可以更快、更可靠地进行设计评估和预测
- 这种迭代方法可以在性能和可靠性方面改进设计
- 由于模型的可重用性以及对物理原型的依赖的减少，降低了资源成本
- 通过使用代码自动生成技术，可以减少开发错误和开销

为何为嵌入式系统建模？

- 一个良好的集成建模方法可以大大减少系统文档、设计、测试和实际实现之间的偏差。
- 对于某些嵌入式系统的问题，建模方法可以直接带来额外的效率和准确性的提高

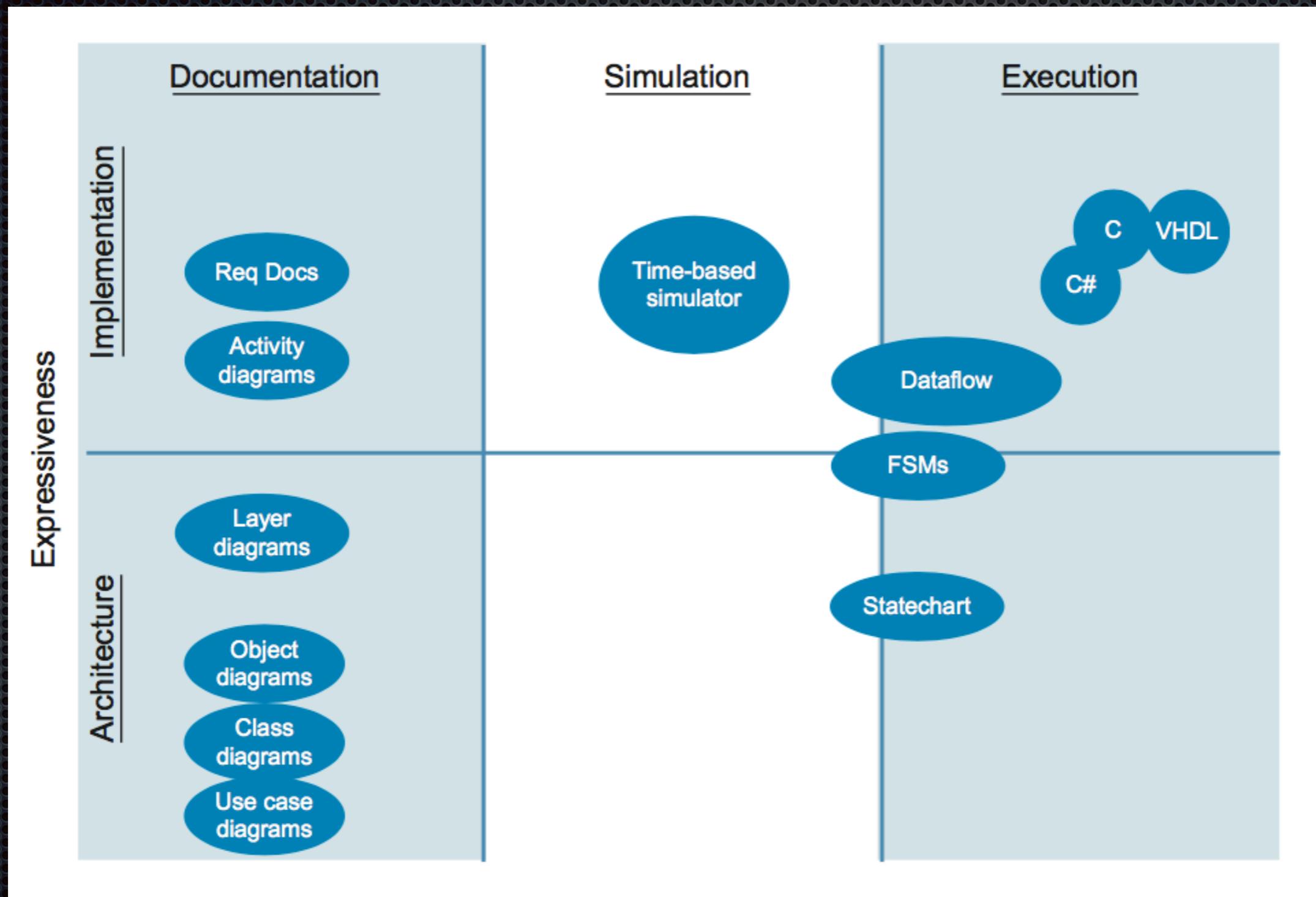
软件建模

- 对于工程和设计任务，特别是与嵌入式系统相关的任务，通常使用某种形式的软件建模作为切入或制定应用程序的整体设计的最初方法
- 软件模型在格式、详细程度和功能方面有显著不同
 - 一些软件模型是行为的
 - 有些只是直观地帮助理解和构建
 - 另一些则更多地用作框架，以确保类似应用程序之间的一致性，或促进工程师团队之间的沟通。
- 嵌入式系统设计人员面临的挑战是，要知道哪种类型和级别的模型最适合当前的具体情况和问题

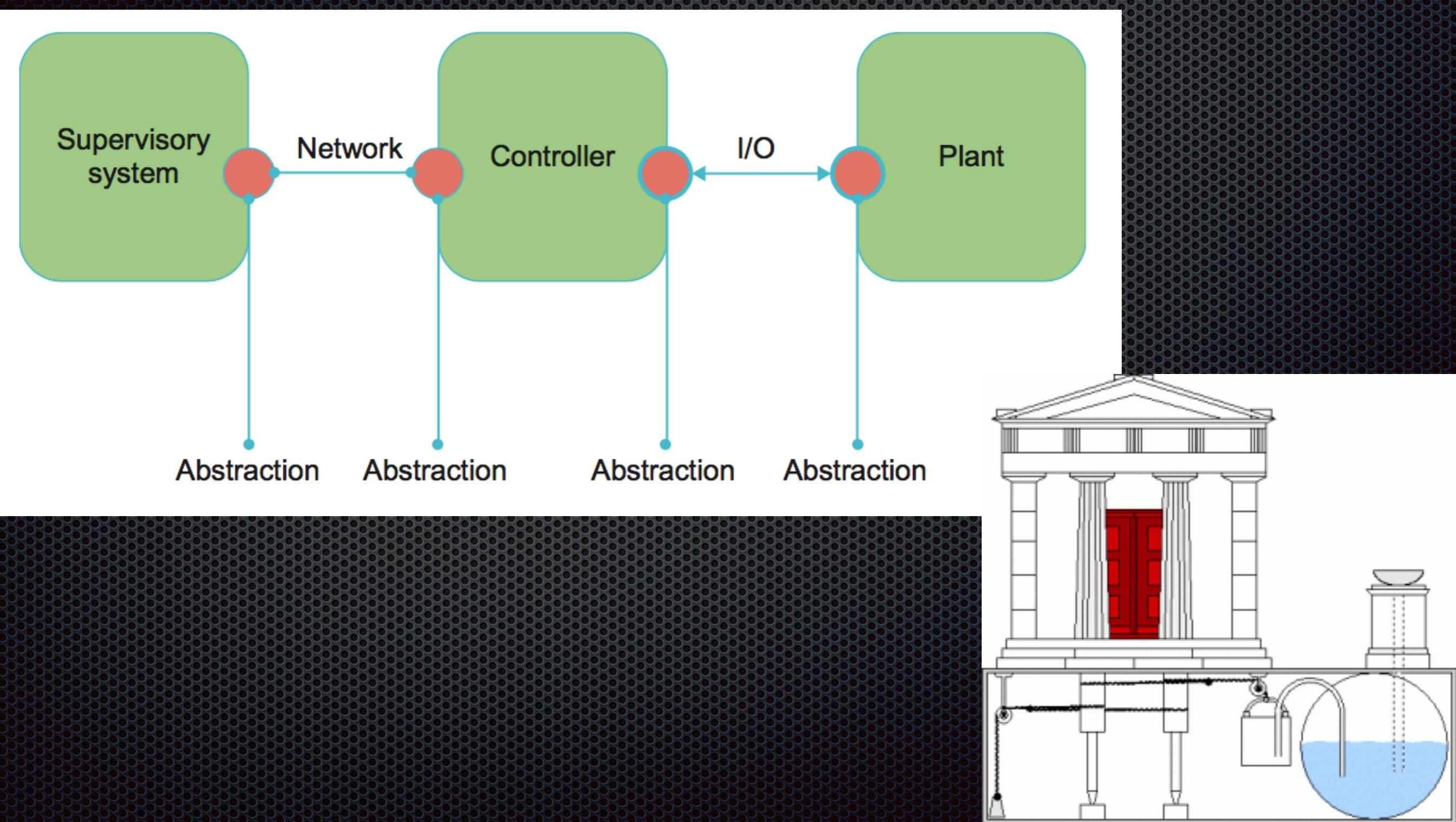
建模语言

- 与编程语言一样，建模语言有明确的定义和标准语法，用于表示结构和功能参与者及其随时间变化的主要关系
- 建模语言有多种形式
 - 图形、文本
 - 面向文档、仿真或执行
 - 专注于体系架构层面内容、实现层面内容

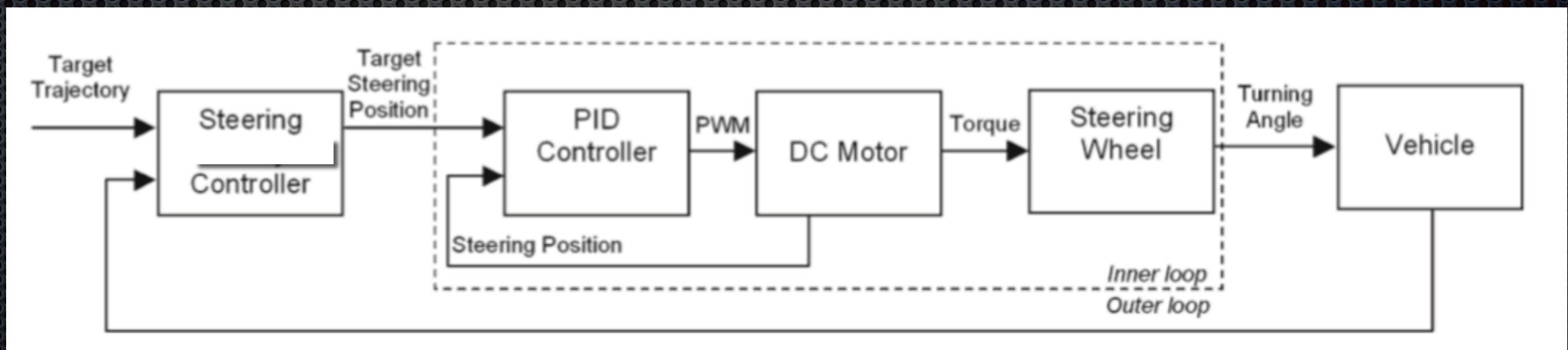
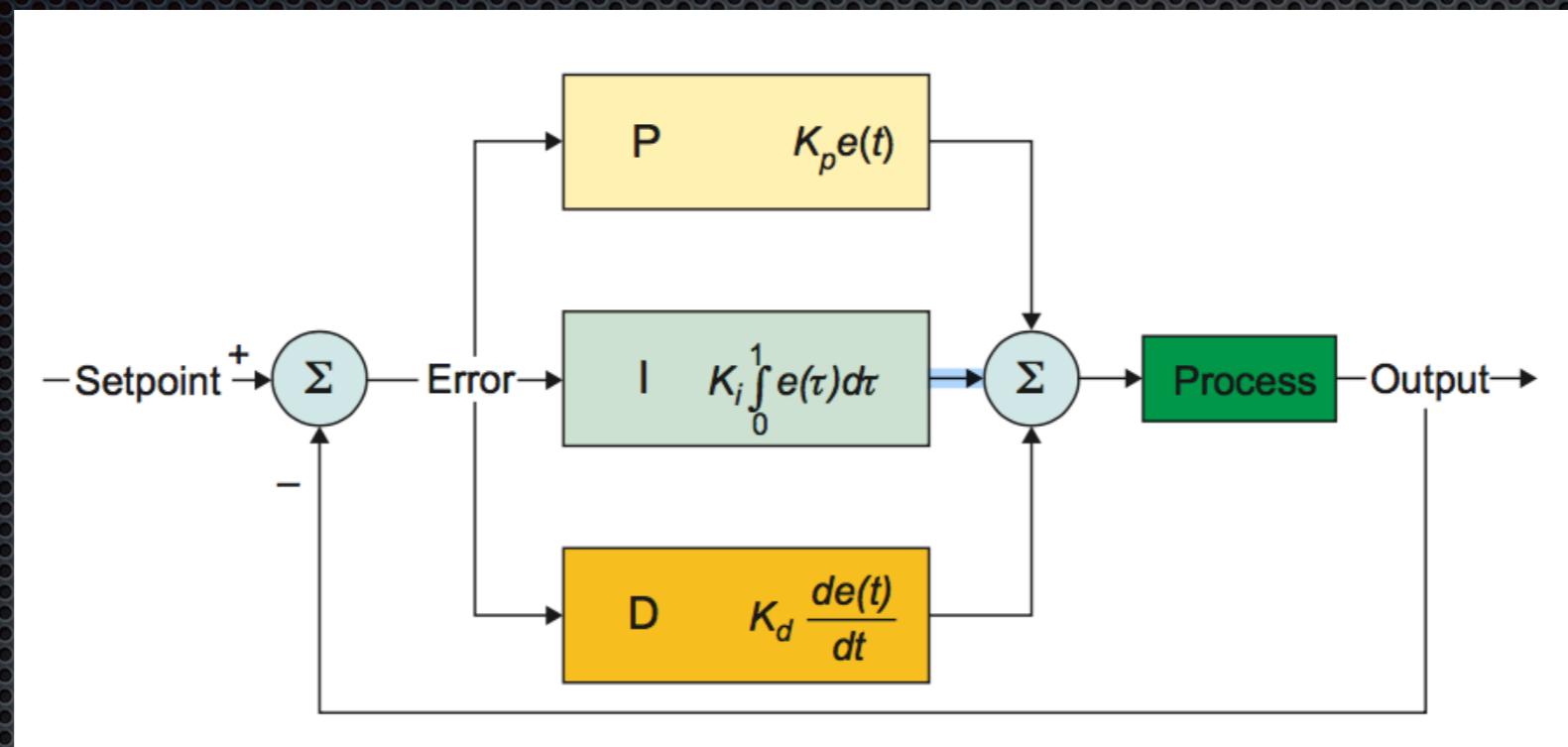
建模语言的几个关键方面



一个典型的嵌入式控制系统模式



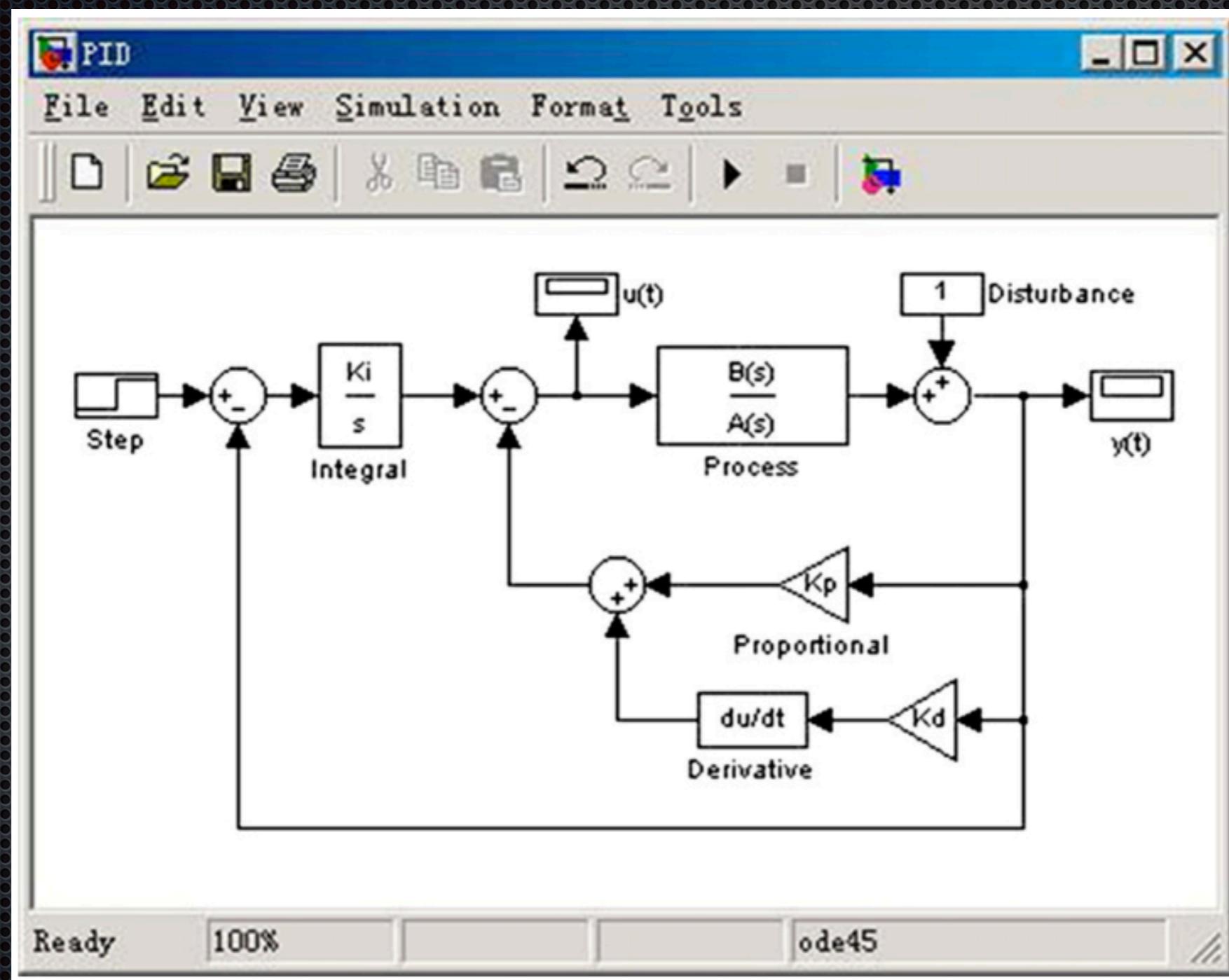
建模语言例子



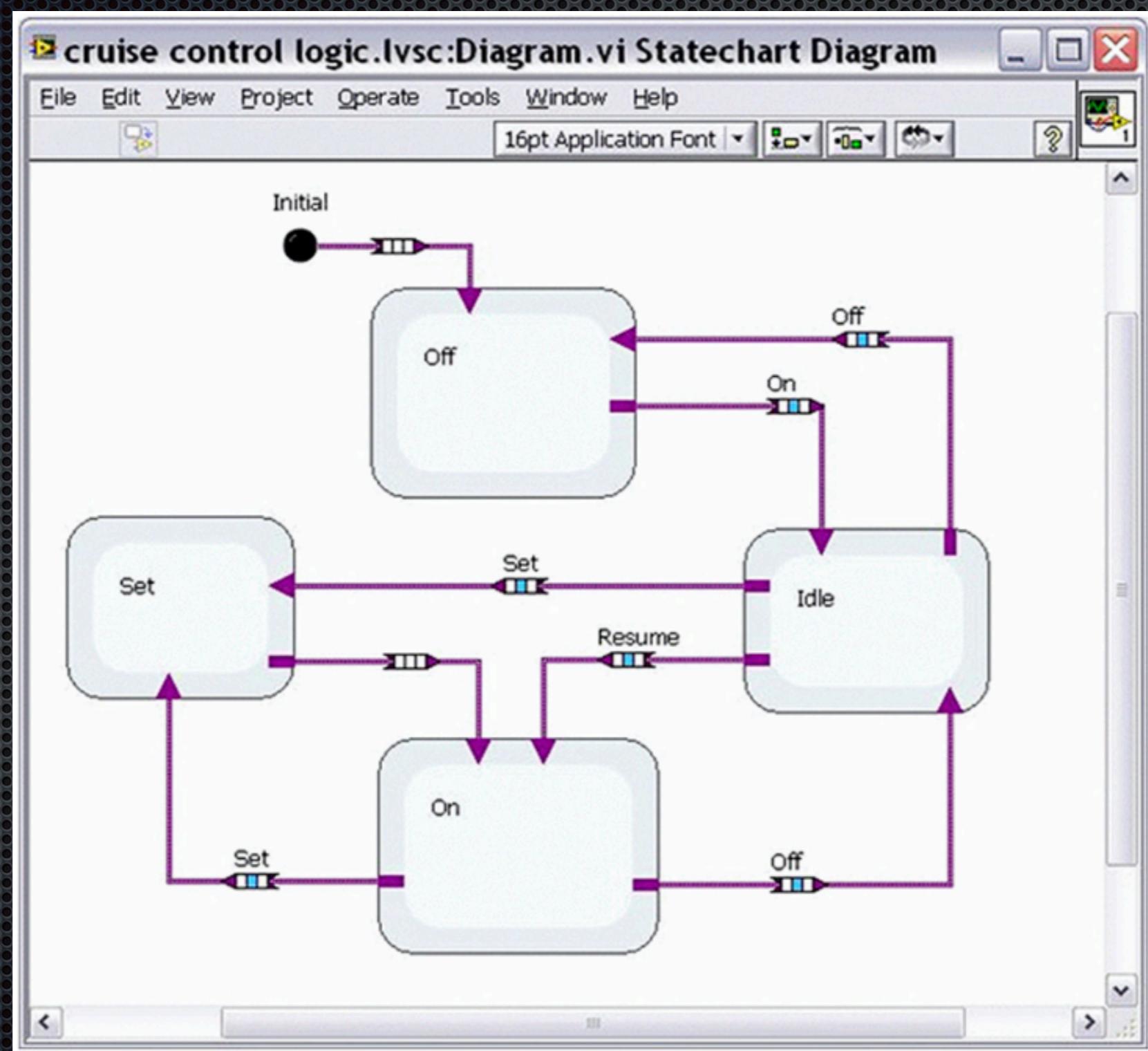
C代码: 文本, 执行, 实现层面的语言

```
previous_error = setpoint - process_feedback
integral = 0
start:
    wait(dt)
    error = setpoint - process_feedback
    integral = integral + (error*dt)
    derivative = (error - previous_error)/dt
    output = (Kp*error) + (Ki*integral) + (Kd*derivative)
    previous_error = error
    goto start
```

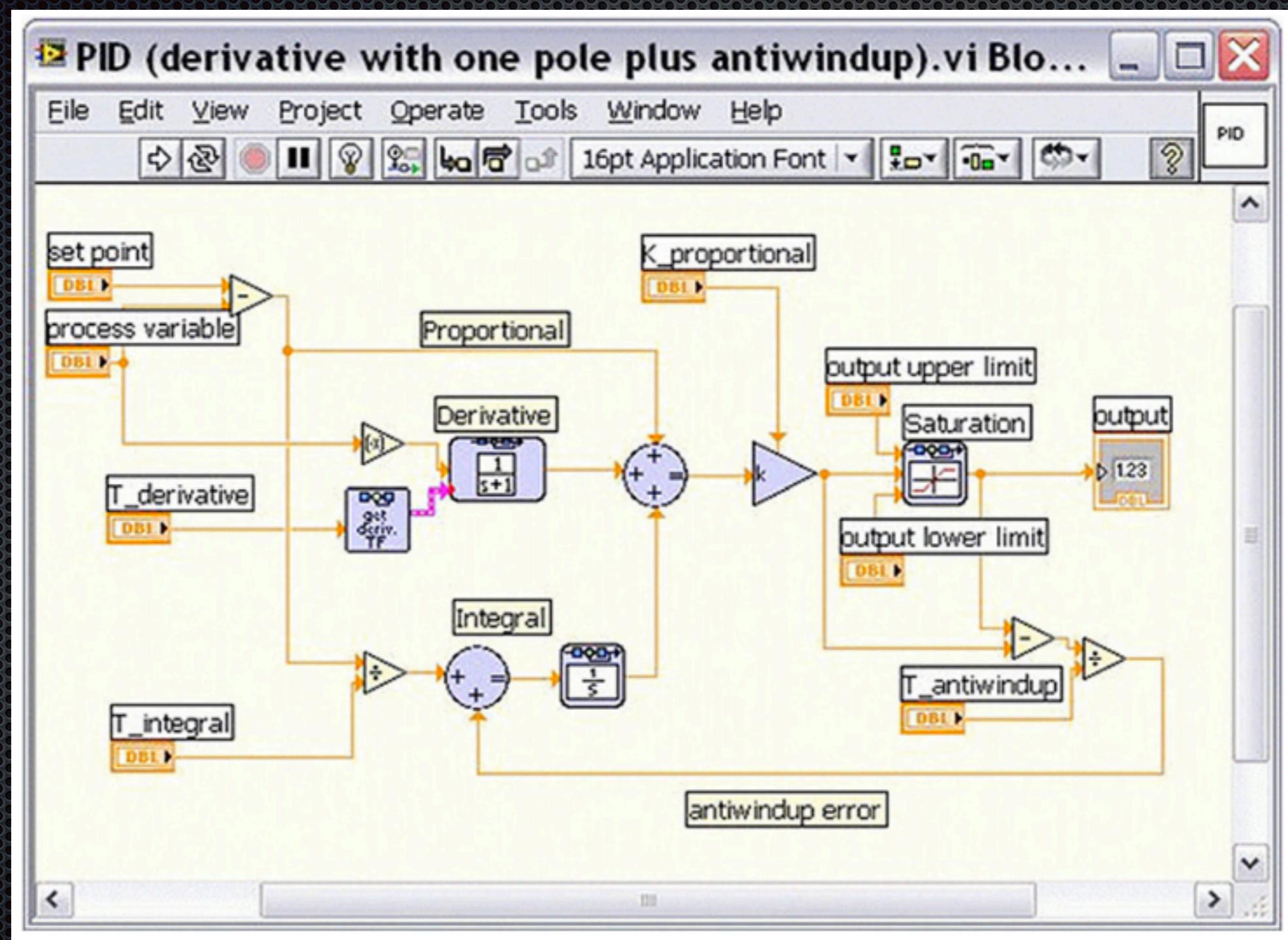
基于时间的仿真：图形、实现、仿真工具



状态图：图形化、架构和实现工具

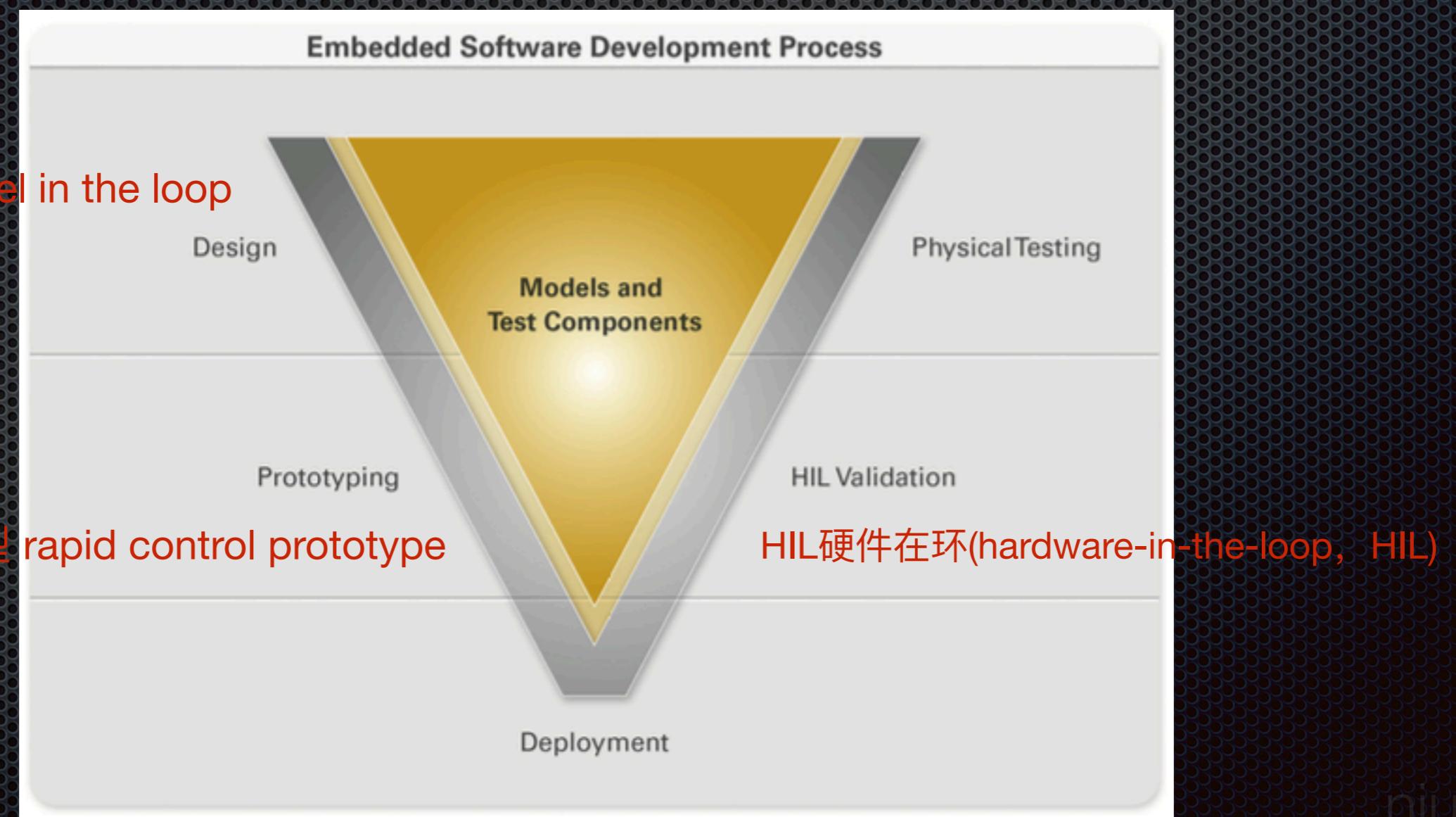


数据流：图形化、架构和实现软件



V图承诺

- 通常用于描述一个嵌入式建模开发周期



何时为嵌入式系统建立模型

- 任务和安全关键型应用
- 高度复杂的应用程序和系统
- 大型开发团队
- 没有其他选择（当没有原型时）

任务和安全关键型应用程序

- 任务关键这里特指故障可能导致巨额损失，对企业声誉造成伤害或严重损坏
- 对此类系统而言，软件建模、软件工程最优方法和正式行业标准等都要结合起来，以促进和保障嵌入式系统的安全和可靠

高度复杂的应用程序和系统

- 强调嵌入式设计复杂性演变的例子：
 - 1960年代喷气式战斗机约5万行 → 现代汽车10亿行代码
- 现代汽车动力传动控制系统受益于建模技术，因为需要不断调整满足各种要求，包括：
 - 控制废气排放
 - 日益严格的标准
 - 更好的燃油经济性
 - 满足客户的要求，提供卓越的性能和舒适度等
- 上述目标既相互关联又相互冲突
 - 例如稀燃技术可以大大降低燃油消耗，但也降低了三元催化转换效率，造成了更多的空气污染

缺陷检测的时间和成本

- 使用正确建模和仿真技术不只是获得一个更好的设计，还可以节约成本以及更快的完成开发
- 建模结合软件工程最优方法将有助于解决大多数缺陷
-

大型开发团队需要建模

- 庞大的开发团队面临几个关键问题的挑战：
 - 通信问题
 - 地域分散和语言不同
 - 工具差异
- 建模可以帮助解决上述一些问题：
 - 建模可以帮助嵌入式项目不同团队内明确的沟通，设计一个有用的、能够准确表示项目动态的系统模型，可以方便沟通
 - 嵌入式模型提供了项目一致性的表达，而不是依赖于翻译书面文档
 - 要求保持软件模型可以在工具之间共享

建模通常是最唯一的选择

- 当嵌入式系统不存在 — 当设计的硬件尚未完成或芯片尚未准备好，这样的情形，建模、模拟、仿真和原型机设计都是非常有用的方法
- 尚未发布的芯片，因为项目大小或成本的关系，不能开发原型机，软件模型可以很有帮助

References

- 嵌入式系统导论：CPS方法
- 嵌入式系统软件工程，第三章