

# Chp3 Case Studies

在实际项目设计中，如何应用体系结构知识进行设计



## 3.1 Key Word in Context

### ○ Problem

#### ● 输入

- An ordered set of lines
- Line : an ordered set of words
- Word: an ordered set of characters

#### ● Circularly shifted --- Line

- Removing the first word
- Appending it at the end of the line

#### ● 输出

- A listing of **all circular shifts** of **all lines** in **alphabetical order**



# The effect of changes on software design

- Changes in the processing algorithm
  - E.g. 进行line shifting的时机
- Changes in data representation
  - E.g. lines, word, characters 等的存储方式
- Enhancement to system function
  - E.g. 消除无效的circular shifts
  - E.g. 交互式功能
- Performance
  - Space and time
- Reuse
  - 部件可以在何种程度上支持重用

### 3.1.1 Solution 1 : Main Program/Subroutine with Shared Data

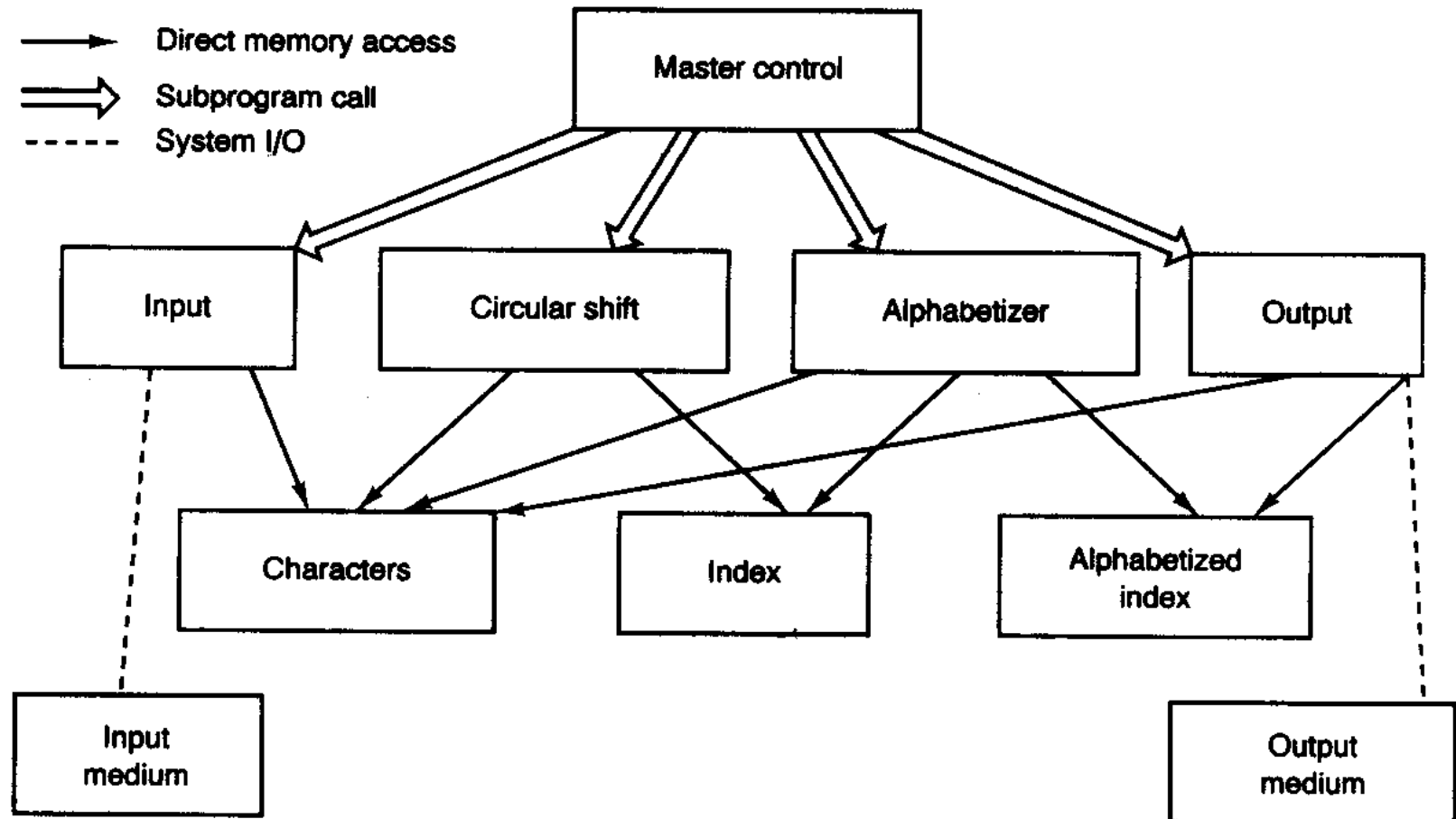


FIGURE 3.1 KWIC: Shared Data Solution



# 设计方法

- 按功能划分为4个基本函数
  - Input, shift, alphabetize, output
- 调度： 利用main程序完成
- 部件通信：
  - 共享内存
  - 无限制的读/写协议

（由各个函数共同保证正确性）



# 优缺点

## ○ 优点

- Allows data to be represented efficiently
- 将不同的处理功能划分为不同的模块

## ○ 缺点

- 数据存储格式的变化会影响所有的模块
- 不易实现：
  - Change in the overall processing algorithm
  - Enhancements to system function
- 对重用的支持不够

## 3.1.2 Solution 2: Abstract Data Types

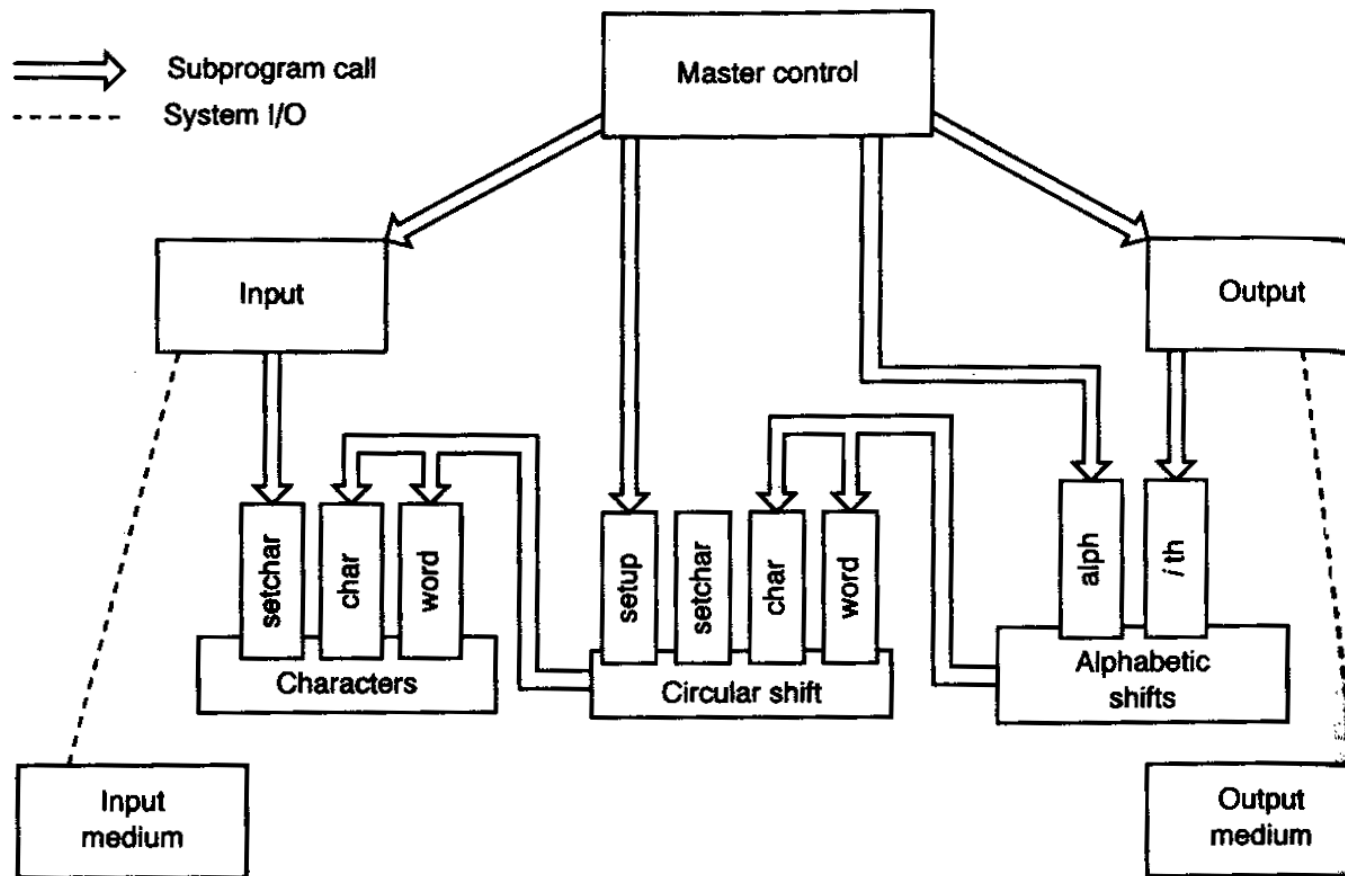


FIGURE 3.2 KWIC: Abstract Data Type Solution



# 设计方法

- 划分为5个模块
- 数据存储
  - 非共享数据
  - 提供接口供其他模块访问





# 优缺点

## ○ 优点

- 可以在模块内部独立进行算法和数据表示的修改
- 与Solution 1相比，重用性较好

## ○ 缺点

- Not particularly well suited to certain kinds of functional enhancements
- 难以在系统中添加新的功能

# 3.1.3 Solution 3 : Implicit Invocation

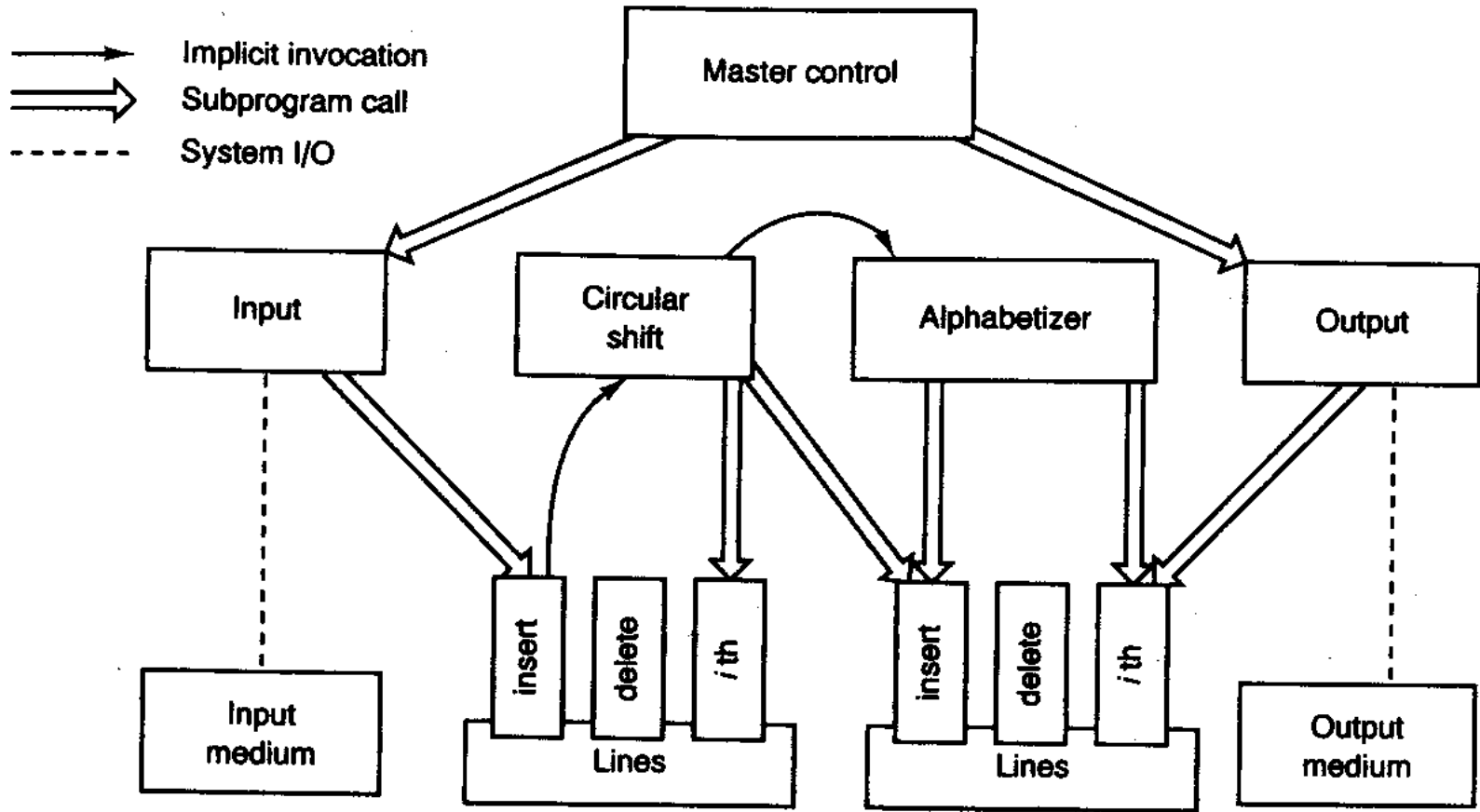


FIGURE 3.3 KWIC: Implicit Invocation Solution



# 设计方法

- 基于共享数据的部件集成
- 与Solution 1 的区别
  - 对数据的接口更为抽象
    - 通过ADT概念（List / set）
  - Computations are invoked **implicitly** as **data is modified**
    - 在数据更新时进行自动更新



# 优缺点

## ○ 优点

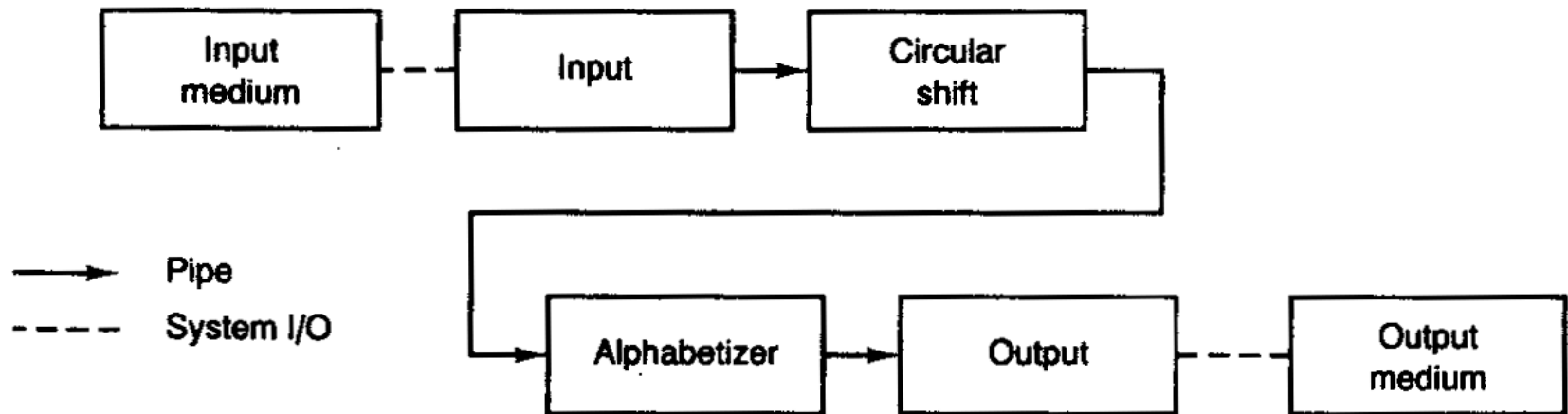
- 有利于进行functional enhancements
  - 新的模块进行注册即可（e.g. Observer）
- 有利于处理data representation改变的情况
- 对重用性支持较好

## ○ 缺点（考虑chp2中的event style）

- Be difficult to control the processing order
- Use more space

## 3.1.4 Solution 4

# Pipes and Filters



**FIGURE 3.4** KWIC: Pipe-and-Filter Solution



# 设计方法

- 使用pipeline方法
  - Filters :
    - Input, shift, alphabetize, output
  - Distributed control
  - Filter之间的数据传输
    - 通过pipe来完成



# 优缺点

## ○ 优点

- Maintain the intuitive flow of processing
- Support reuse
  - 便于添加新的功能（便于添加新的**filter**）
- 便于进行修改（**filter**与其他**filter**无关）

## ○ 缺点

- 难于实现交互式应用
- 存储空间的使用



## 3.1.5 Comparisons

---

	<b>Shared Data</b>	<b>Abstract Data Type</b>	<b>Implicit Invocation</b>	<b>Pipe and Filter</b>
Change in Algorithm	—	—	+	+
Change in Data Rep	—	+	—	—
Change in Function	+	—	+	+
Performance	+	+	—	—
Reuse	—	+	—	+

---

**FIGURE 3.5** KWIC: Comparison of Solutions





## 3.2 Instrumentation Software

- Purpose

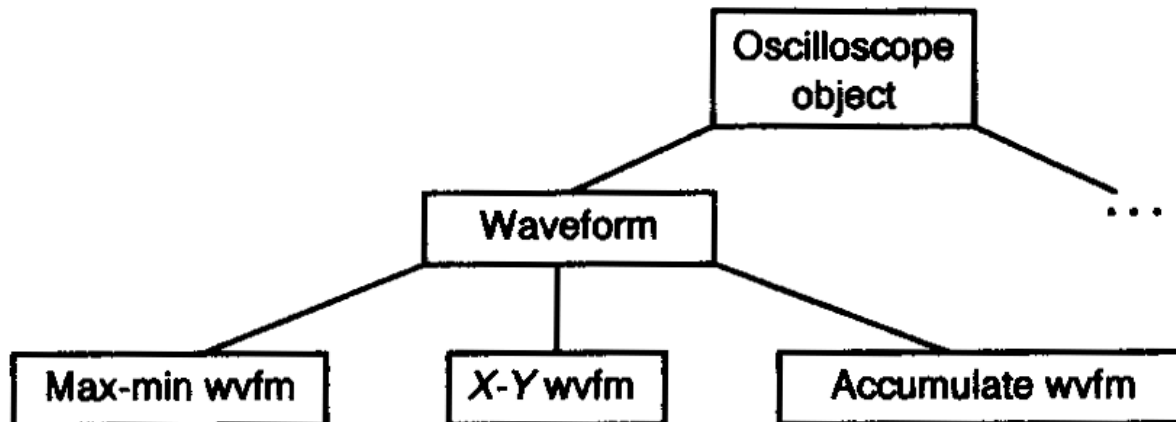
- A reusable system architecture
  - For oscilloscopes
    - 采样电子信号，并在屏幕上显示traces
    - Measurement on the signals
    - 其他
      - 巨量数据采集
      - 提供网络接口（e.g. 海底光缆上的数据采集）
      - 提供集成的用户界面
- e.g. touch screen, help , color display, ...



# Problem

- Little reuse across different oscilloscope products
  - 不同厂家的产品
  - 同一公司产品，存在硬件功能，用户界面要求等不同
  - 定制系统的存在
- Performance
  - 配置问题
    - Software was not rapidly configurable within the instrument

## 3.2.1 An Object-Oriented Model



**FIGURE 3.6** Oscilloscopes:  
An Object-Oriented Model



# 分析

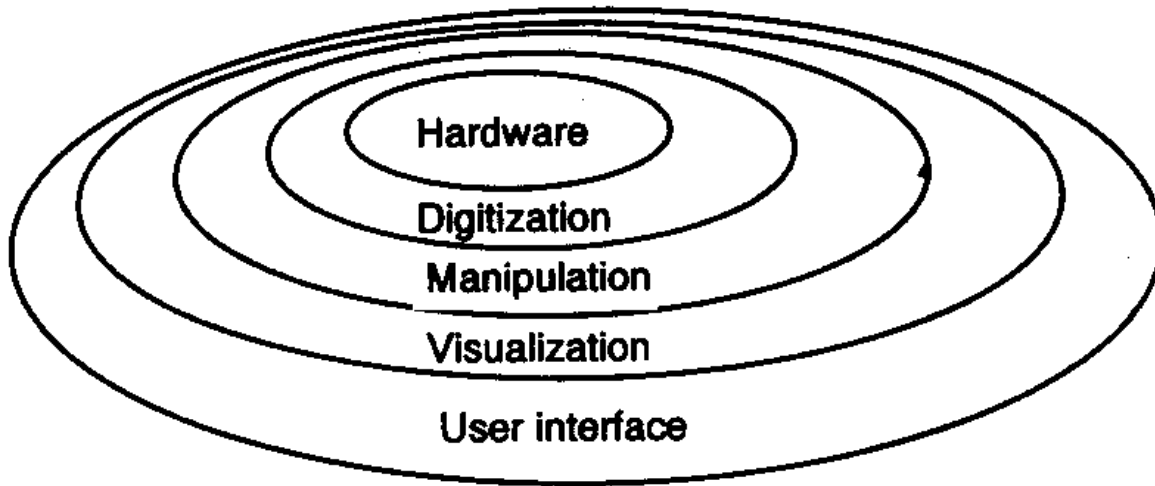
## ○ 优点

- Classified the data types used in oscilloscope  
Waveform, signal,  
measurement, trigger mode, ...

## ○ 问题          无法得到预期结果

- 没有统一的模型来描述这些data types如何协同工作
- Confusion about the partitioning of functionality

## 3.2.2 A Layered Model



**FIGURE 3.7** Oscilloscopes:  
A Layered Model



# 层次组织

- Core Layer
  - 信号输入时的处理功能
  - 通常用硬件实现
- Digitization
  - 信号的数字化及存储
- Manipulation      Waveform manipulation
  - Measurement, waveform addition, Fourier transformation
- Visualization
  - Display功能，将数字信息与图形表示相对应
- User Interface
  - 与用户交互



# 问题

- 边界的划分与各种功能之间的交互矛盾
  - 实质上是越层访问的问题
  - 在采用“分层”模式时，特别需要注意的问题

## 3.2.3 A Pipe\_and\_Filter Model

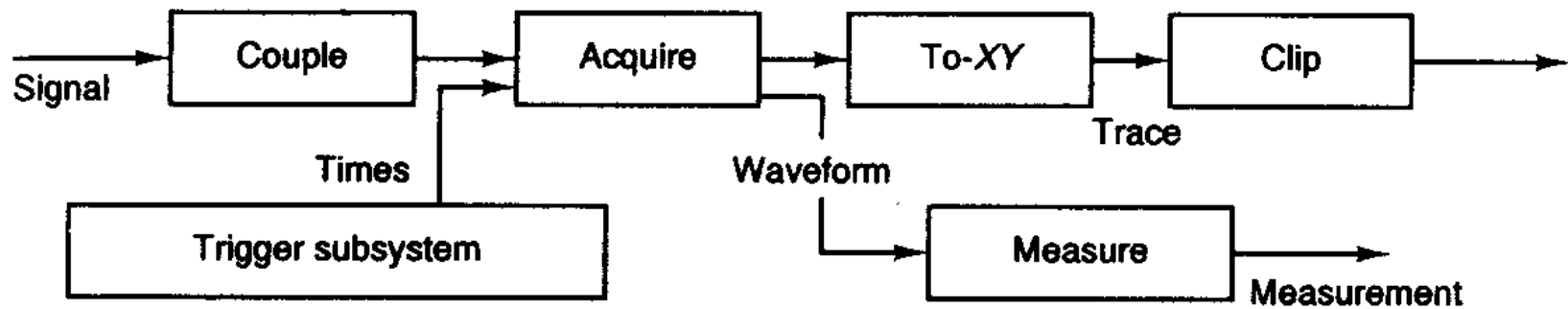


FIGURE 3.8 Oscilloscopes: A Pipe-and-Filter Model





# 评价

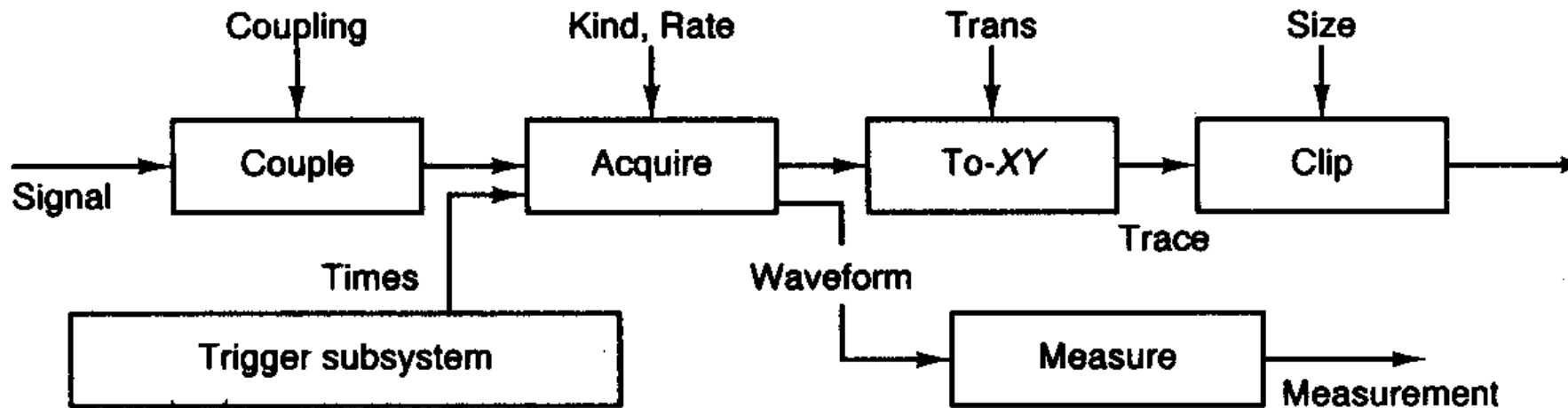
## ○ 优点

- 与Layered Mode相比，不是将功能简单划分为独立的块
- 与实际流程处理方式相同

## ○ 问题

- 用户如何与之进行交互
  - 如果只是和最后的Visual End交互，则比Layered Model还差

### 3.2.4 A Modified Pipe\_and\_Filter Model



**FIGURE 3.9** Oscilloscopes: A Modified Pipe-and-Filter Model

在每个Filter上提供进行设置的控制接口  
(考虑工业流水线的控制)



# 评价

- 很好地解决了用户交互的问题
  - 提供了所有可以进行动态修改的设置
  - 清晰表明了用户如何在软件上调整从而控制示波器
- 信号处理过程与用户界面分离
  - 用户界面实质就是对一些控制参数进行调整
  - **Hardware**和**signal-processing**的变化对界面无影响



## 3.2.5 Further Specialization

- 3.2.4中存在的问题
  - Poor performance(管道过滤器本身的问题)
    - E.g. 多次数据复制，导致冗余数据
  - 不同filter之间的速率不匹配
- 改进
  - Several “colors” of pipes
    - 有的可以不复制数据即可处理
    - 低速的filter处于忙状态时，可以忽略数据
  - 有利于针对具体问题进行剪裁



## 3.2.6 Summary

- Different architectural style have different effects on the solution to a set of problem
- 软件体系结构的设计必须进行修改和调整，以适合特定问题的需要
- 结论
  - 修改后的pipe\_and\_filter architecture



## 3.3 Mobile Robotics

- Purpose : 实现对vehicle的控制
- 问题
  - 处理外部的sensors 和 actuators
  - 实时对外部环境作出反映
  - 包括
    - 通过Sensor获取输入
    - 控制运动部件
    - 决定下一步的路线



### 3.3.1 Design Considerations

针对具体问题，考虑要达到的目标

- Req 1 :

- accommodate **deliberative** and **reactive** behavior      如何协调？
- Deliberative : designated objective
- Reactive : 对环境的反应

- Req2 :    allow for uncertainty

- 能够处理不完备，不可信的信息



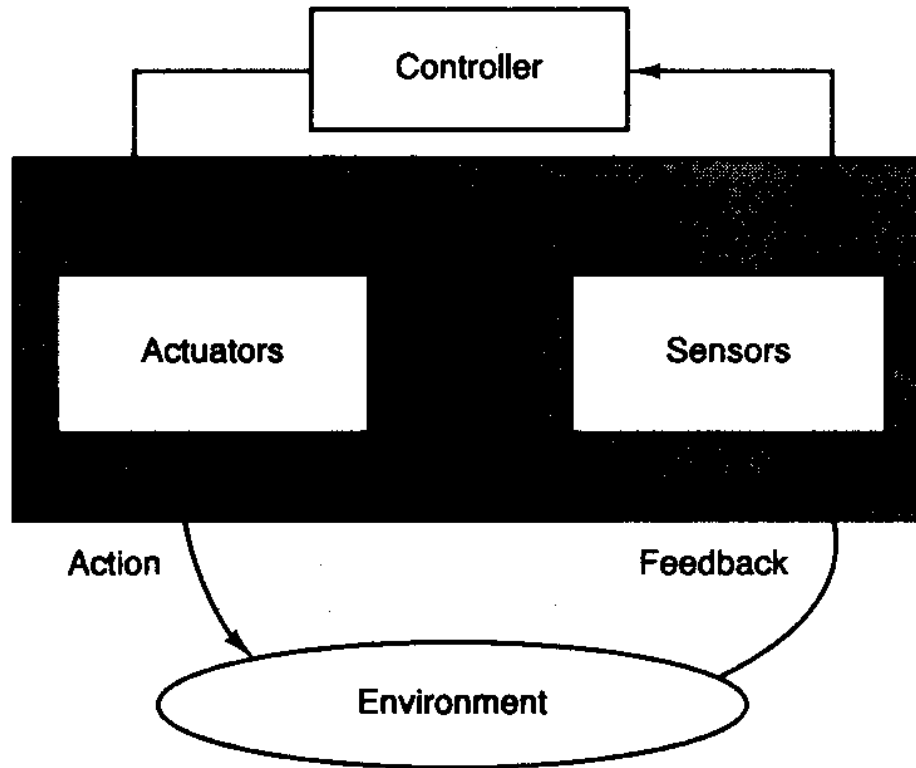
# 目标

- Req 3 : Account for danger
  - Fault tolerance
  - Safety
  - Performance
- Req 4 : give the designer flexibility (弹性)
  - 考虑到进行修改的难易程度

对上述要求的紧迫性依赖于预期目标&环境



## 3.3.2 Solution 1 : Control Loop



**FIGURE 3.10** A Control-Loop Solution for Mobile Robots

Most industrial robots → open-loop paradigm  
Mobile robots → feed back (close-loop architecture)



# 分析

## ○ Req1

### ● 问题1

- 假设：环境的变化是continuous and require continuous reactions
- 结果：对突变情况无法处理

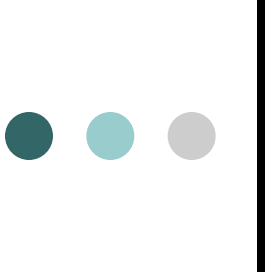
### ● 问题2

- 无法将系统细分为相互合作的部件
- 需要引入其他paradigm



# 分析

- Req 2
  - 对uncertainty问题的解决
    - 通过循环reducing the unknowns
  - 难于加入其他subtle steps
- Req 3
  - 支持fault tolerance , safety
  - 通过在循环过程中降低风险
- Req 4
  - 主要部件分离， 可以进行分布置换
  - 更细的过程需要在部件内部实现



# 结论

## ○ 适用于

- simple robotic system
- Handle only a small number of external events
- Tasks do not require complex decomposition

### 3.3.3 Solution 2 : Layered Architecture

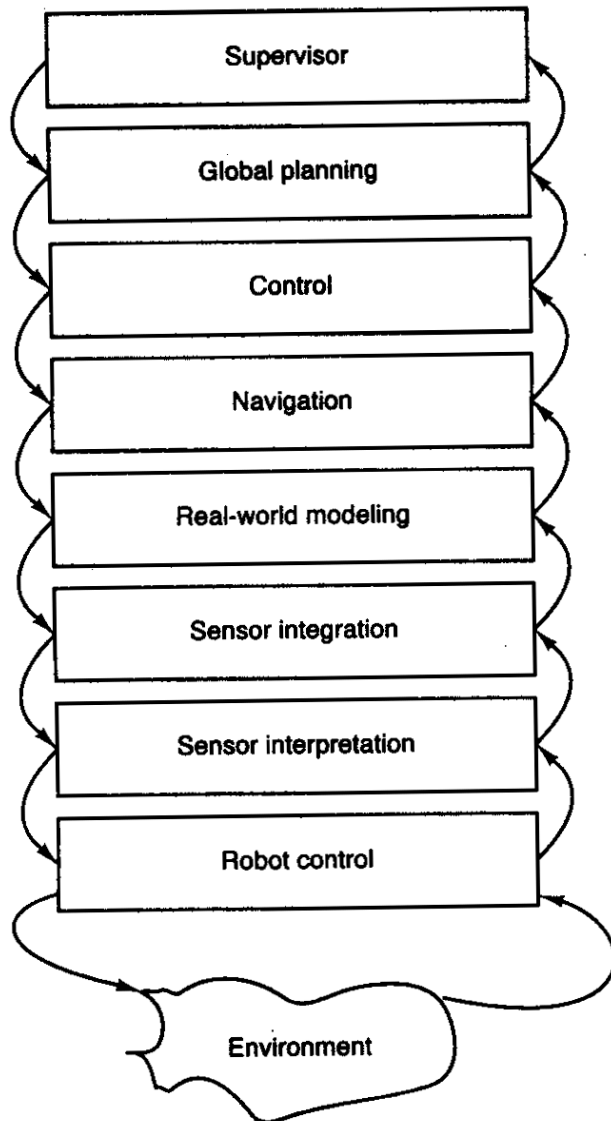


FIGURE 3.11 A Layered Solution for Mobile Robots

Lev 1 : robot control routines

Lev 2, 3 : 实现sensor输入/整合

Lev 4 : 对环境世界模型的管理

Lev 5 : 导航

Lev 6,7 : 调度

Lev 8: user interface ,  
supervisory func, ...



# 分析

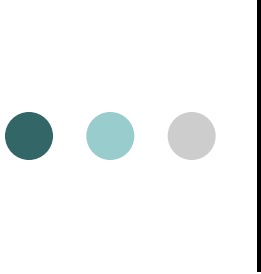
## ○ Req 1 :

- 可以较好地实现deliberative和reaction的协同
  - Robot control / navigation
- 信息传递可能需要越层
  - E.g. 对紧急信息的处理
- 对data层次， control层次没有区分开来



# 分析

- Req 2 能对uncertain信息进行处理
  - E.g. sensor信息 + Real\_World模型
- Req 3
  - Fault tolerance, passive safety → OK
  - Performance, active safety → 需要越层信息处理
- Req 4
  - 层次间关系复杂，难以进行replacement，以及addition of component

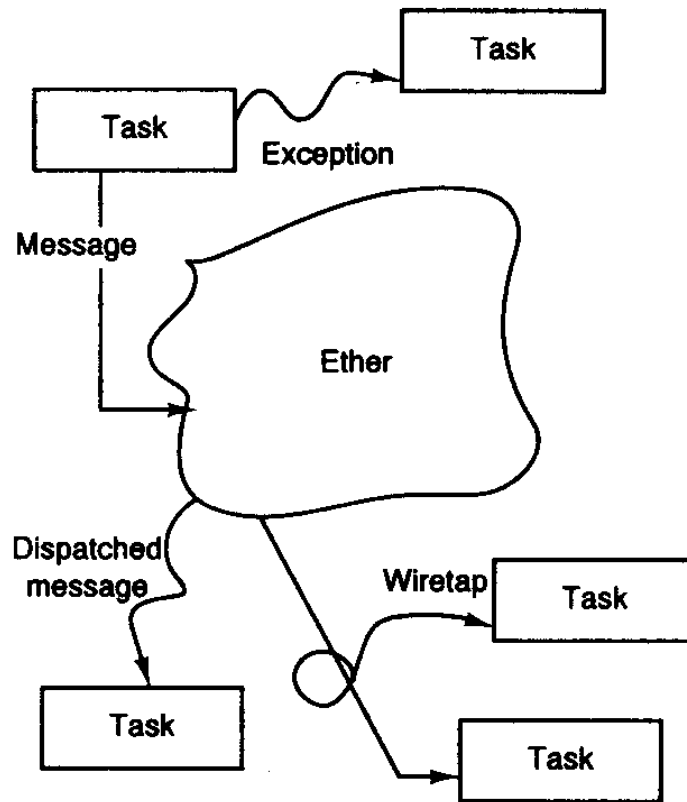


# 结论

- The abstraction levels defined by the layered architecture provide a **framework** for organizing the components that succeeds because it is **precise about the roles** of the different layers
- 缺点
  - It breaks down when it is taken to the greater level of detail demanded by an actual implementation
  - The communication patterns in a robot are not likely to follow the orderly scheme implied by the architecture



# 3.3.4 Solution 3 : Implicit Invocation

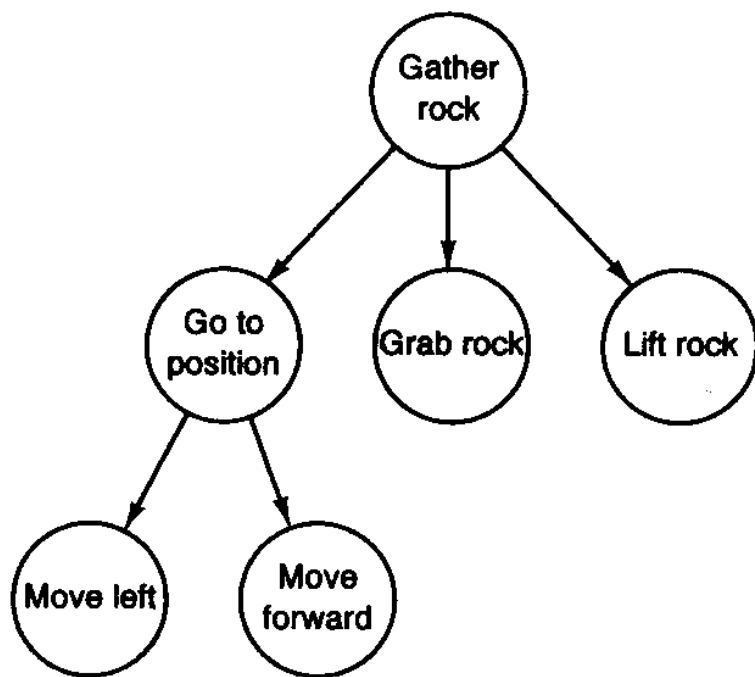


Task Control Architecture (TCA)

隐式调用：  
通过向message server  
广播消息来实现

FIGURE 3.12 An Implicit Invocation Architecture for Mobile Robots

# Task Tree



1. 可在task之间定义相关性，从而实现selective concurrent
2. 可以动态重定义task tree

FIGURE 3.13 A Task Tree for Mobile Robots



# Support three functions

- Exception

- 处理异常或者突发情况，可以对task放弃或重试

- Wiretapping

- 对其他task进行监听 e.g. Safety-check

- Monitor

- 循环检测某些信息 e.g. 电池检查



# 分析

## ○ Req 1

- A clear-cut separation of
  - Action : task tree
  - Reaction: exception, wiretapping, monitor
- Concurrent agent 任务的并发与协同
- 弱点
  - Reliance on a central control point



# 分析

## ○ Req 2

- How TCA addresses uncertainty is less clear
- 使用试验性的task tree + exception handler来处理

## ○ Req 3

- 可较处理好Performance, safety, fault tolerance
  - Fault tolerance by redundancy
  - Concurrent agent : 有利于提高效率



# 分析

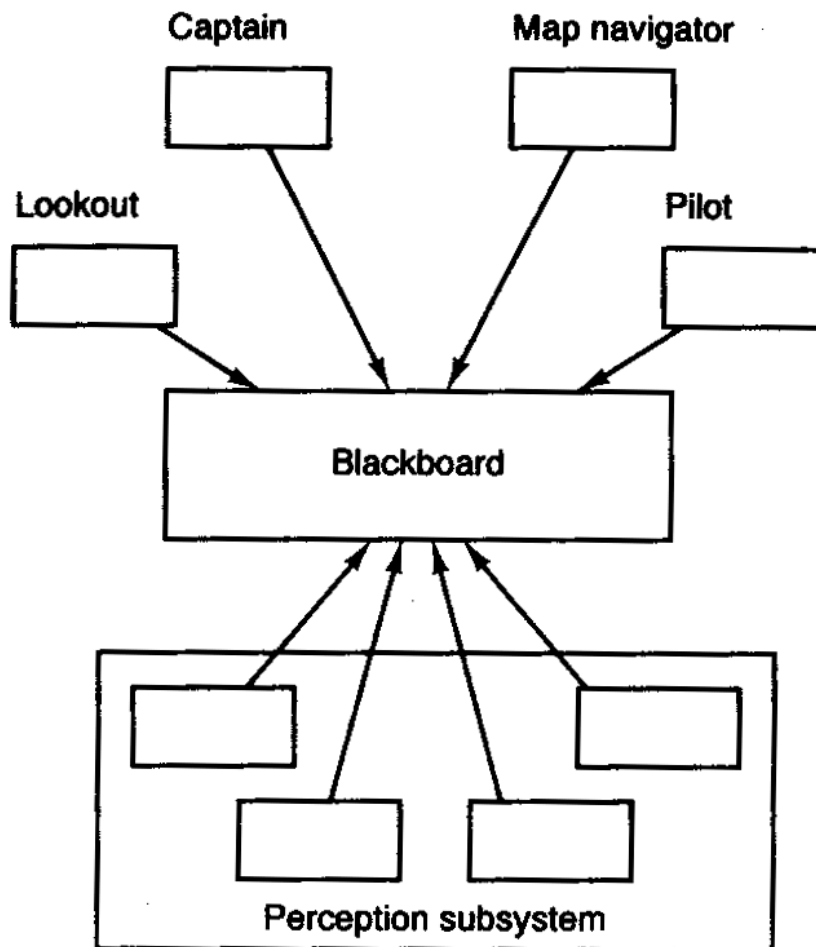
- Req 4

- 修改简单（Event Pattern的优点）

- 结论

- TCA offers a comprehensive set of features for **coordinating** the tasks of a robot while respecting the **requirements for quality and ease of development**
- The richness of the schemes makes it most appropriate for more complex robot projects

## 3.3.5 Solution 4 : Blackboard Architecture



**FIGURE 3.14** A Blackboard Solution for Mobile Robots



# 分析

- Req 1

- 通过数据仓库实现部件间的交互
- 缺点：控制流必须经过**DB**，而有时直接传递可能更好

- Req 2

- 有利于处理confilict or uncertain

- Req3 与TCA类似，好

- Req4 与TCA类似，好





## 结论

- Is capable of modeling the cooperation of tasks, both for coordination and resolving uncertainty in a flexible manner, thanks to an implicit invocation mechanism base on the contents of the database
- These features are only slightly less powerful than TCA's equivalent capabilities



## 3.3.6 Comparisons

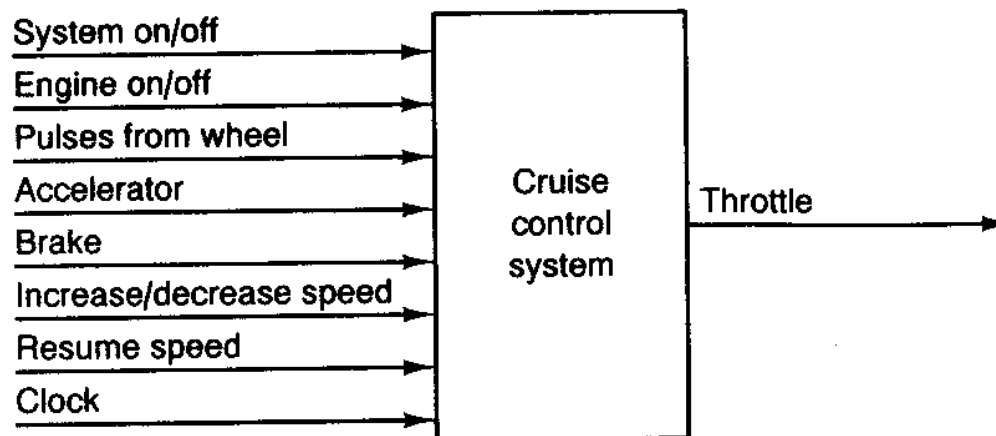
	<u>Control Loop</u>	<u>Layers</u>	<u>Implicit invocation</u>	<u>Blackboard</u>
Task Coordination	+ -	-	+ +	+
Dealing with uncertainty	-	+ -	+ -	+
Fault intolerance	+ -	+ -	+ +	+
Safety	+ -	+ -	+ +	+
Performance	+ -	+ -	+ +	+
Flexibility	+ -	-	+	+

FIGURE 3.15 Table of Comparisons

## 3.4 Cruise Control

### ○ Problem

- A cruise-control system that exists to maintain the speed of a car, even over varying terrain.



**FIGURE 3.16** Booch Block Diagram for Cruise Control



# 问题分析

- Input涉及的内容

- Whether the cruise control is active
- If so , what speed it should maintain

- Output

- A value for the engine throttle setting

- Clock

- 问题细化

Whenever the system is active, determine the desired speed, and control the engine throttle setting to maintain that speed.

## 3.4.1 Object View of Cruise Control

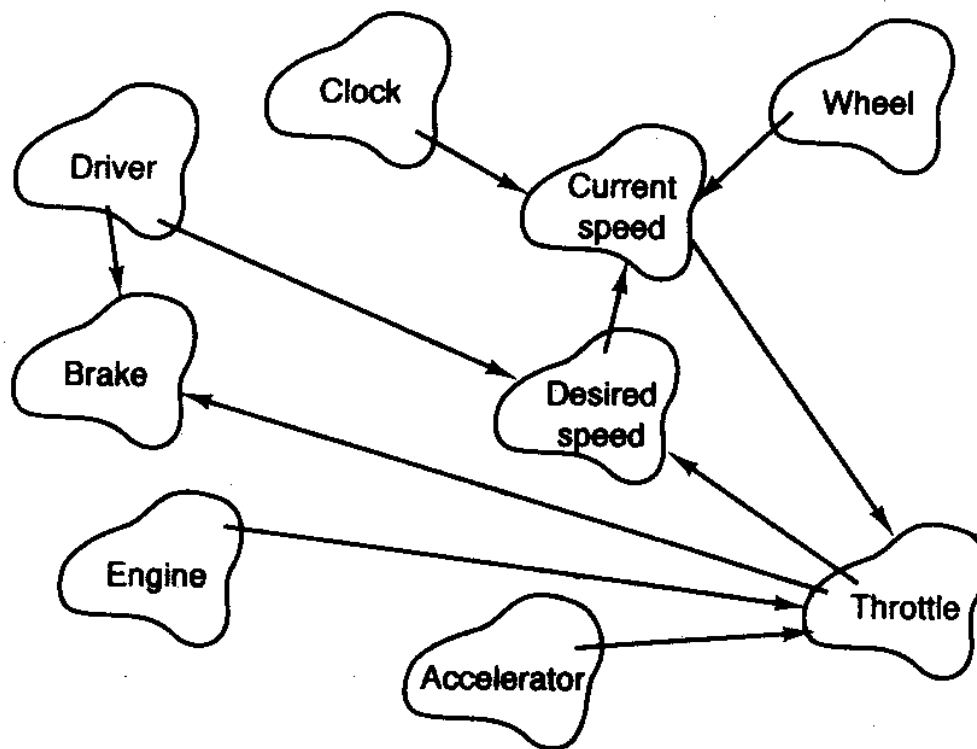


FIGURE 3.17 Booch's Object-Oriented Design for Cruise Control

常规的方案



## 3.4.2 Process-Control View of Cruise Control

### ○ Control-loop Style

- 适合设计连续动作控制的物理系统
- 基本组成元素
  - Computational elements
  - Data elements
  - The control loop paradigm



# Computational elements

- Process definition

- Input : throttle setting
- Controls the speed of the vehicle

- Control algorithm

- 从wheel pulse获取当前速度(结合clock)
- 与预设速度比较
- 修改throttle setting
- 在算法中需要
  - Maintain current throttle setting
  - 实现策略

如何根据current/desired速度间的差异修改 throttle setting



# Data elements

- Controlled variable : current speed
- Manipulated variable : throttle setting
- Set point
  - Desired speed 受限于  
Accelerator input, increase/decrease speed input
  - Cruise control 系统是否生效  
System on/off, engine on/off, brake, resume
- Sensor for controlled variable
  - Current speed (wheel pulse + clock)





# 问题处理

- 划分为两个部分

- Interface with the driver

- 处理问题： whenever the system is active determine the desired speed

- Control loop

- 处理问题： Control the engine throttle setting to maintain that speed

## 问题2 Control\_loop

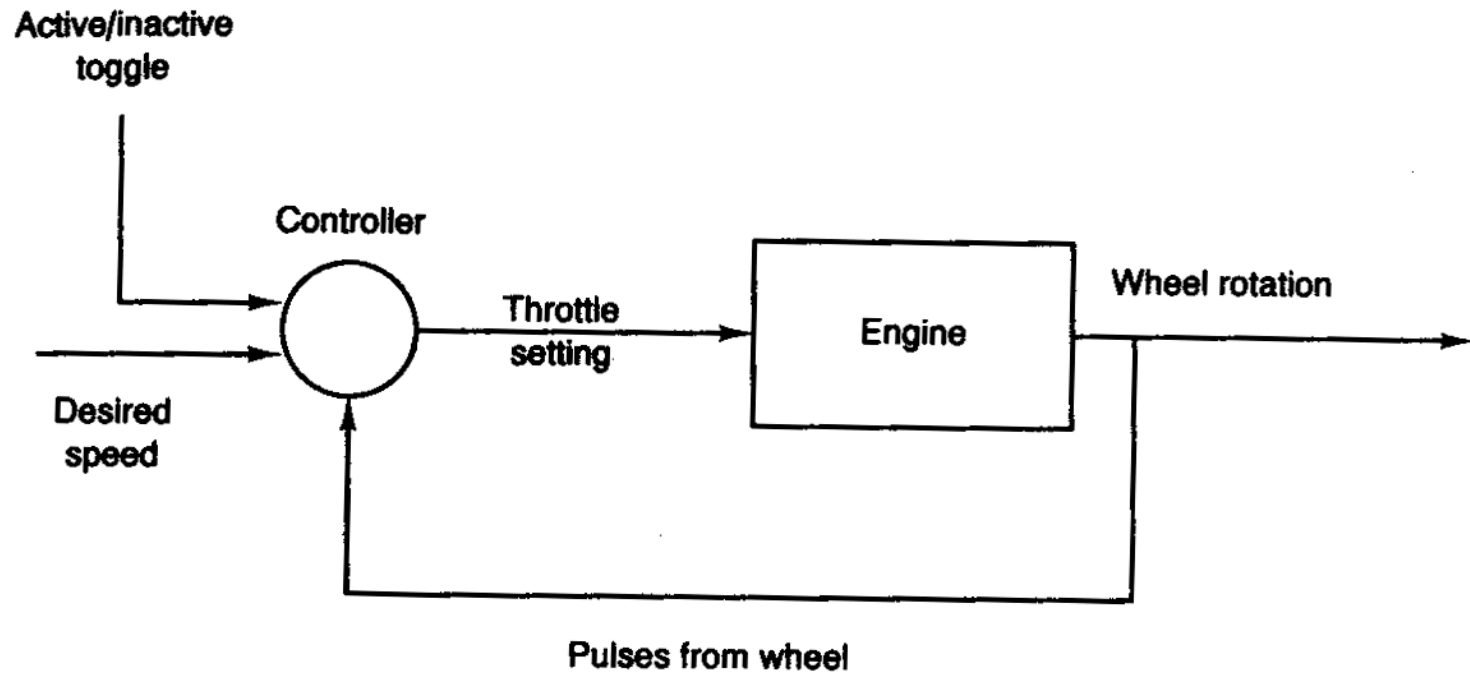


FIGURE 3.18 Control Architecture for Cruise Control



# 问题1 Interface with the driver

- 需要考虑的问题

- (实际工程中, 设计不周往往会带来严重后果)

- 速率采样
  - Controller权限问题

# 问题1 Inter face with the driver

- Set\_Point划分为两个部分
  - Whether or not the automatic system is active
  - The desired speed for use
- Fig 3.19 state machine for activation

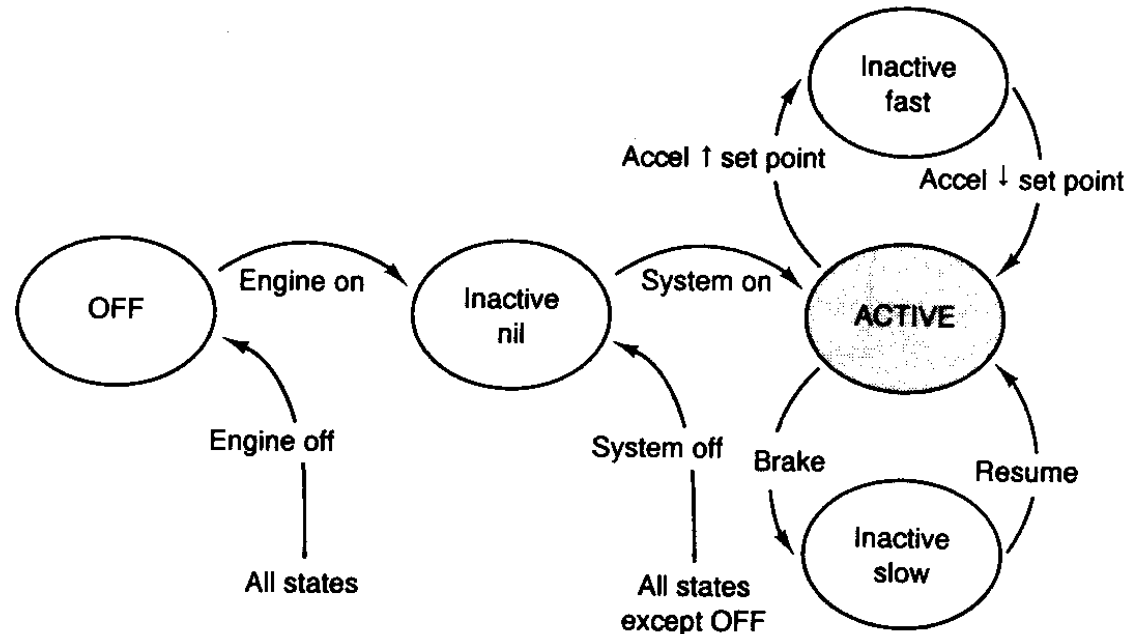


FIGURE 3.19 State Machine for Activation

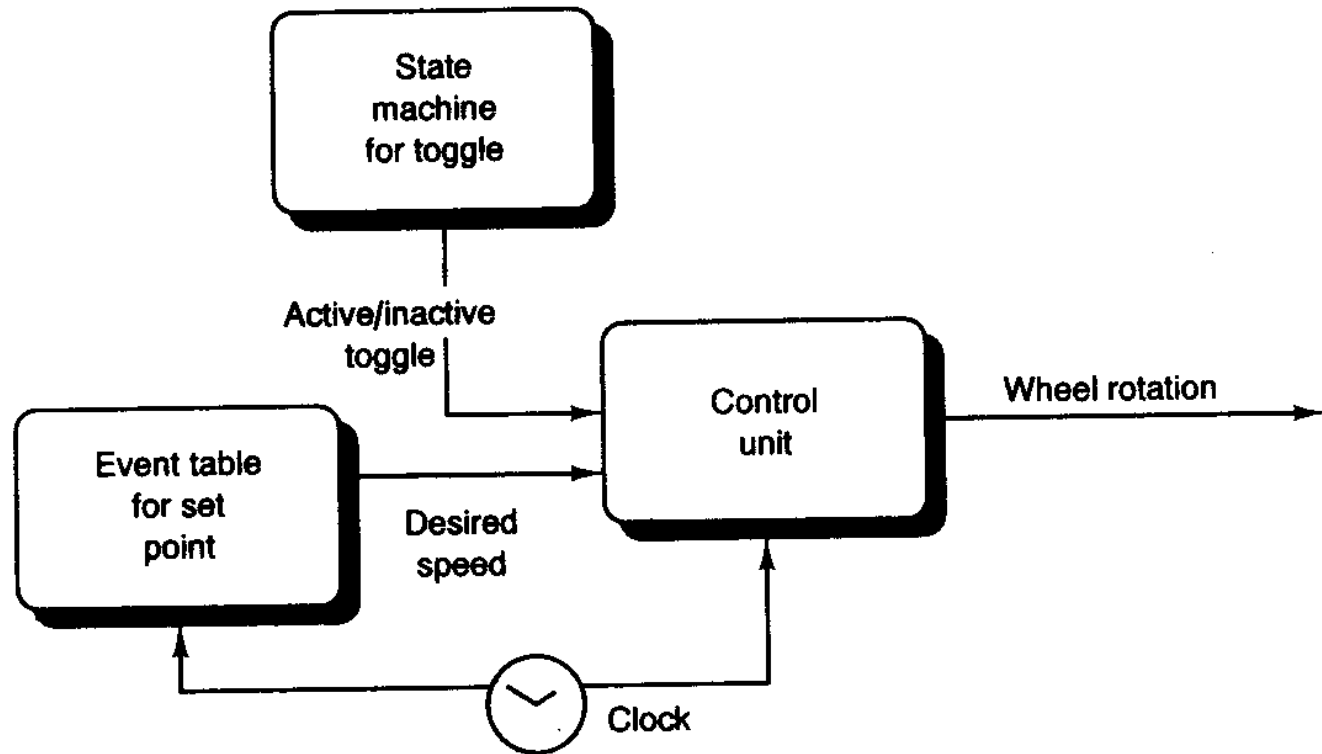


# Determining the desired speed

Event	Effect on desired speed
Engine off, system off	Set to "undefined."
System on	Set to current speed as estimated from wheel pulses.
Increase speed	Increment desired speed by constant.
Decrease speed	Decrement desired speed by constant.

**FIGURE 3.20** Event Table for Determining Set Point

# Control architecture



**FIGURE 3.21** Complete Cruise Control System



## 3.4.3 Analysis and Discussion

- Correspondence between architecture and problem
  - Object-orient
    - 按照Object的概念对系统进行分解
    - 系统中的每个module代表一个major step
  - Control view
    - Control view is particularly appropriate for a certain class of problem



# Methodological Implications

- Methodology can 帮助设计者
  - 决定when the architecture is appropriate
  - Identify the elements of the design and their interactions
  - Identify critical design decision





# Methodological Implications

- Process-Control top-down methodology
  - Choose the control principle
  - Choose the control variables
  - Choose the measured variables
  - Create subsystems
- Methodology for system modifications
  - Add a digital speedometer  
通过wheel pluses进行处理
  - Use separate microcomputers for current/desired speed and throttle



# Analysis and Discussion

- Performance : System response to control
  - On/Off Control
  - Proportional Control
  - Proportional plus Reset Control
- Correctness
  - E.g. Runaway feedback



## 3.4.4 Summary

- Design methodology
  - Focuses attention on significant decisions at appropriate times
  - Decomposing the problem in such a way that development of the software structure proceeds hand in hand with the analysis for significant decisions
  - This localizes decisions and limits the ripples caused by changes



### 3.5.1 Three Vignettes in Mixed Style

## 3.5.1 A Layered Design with Different Styles for the Layers

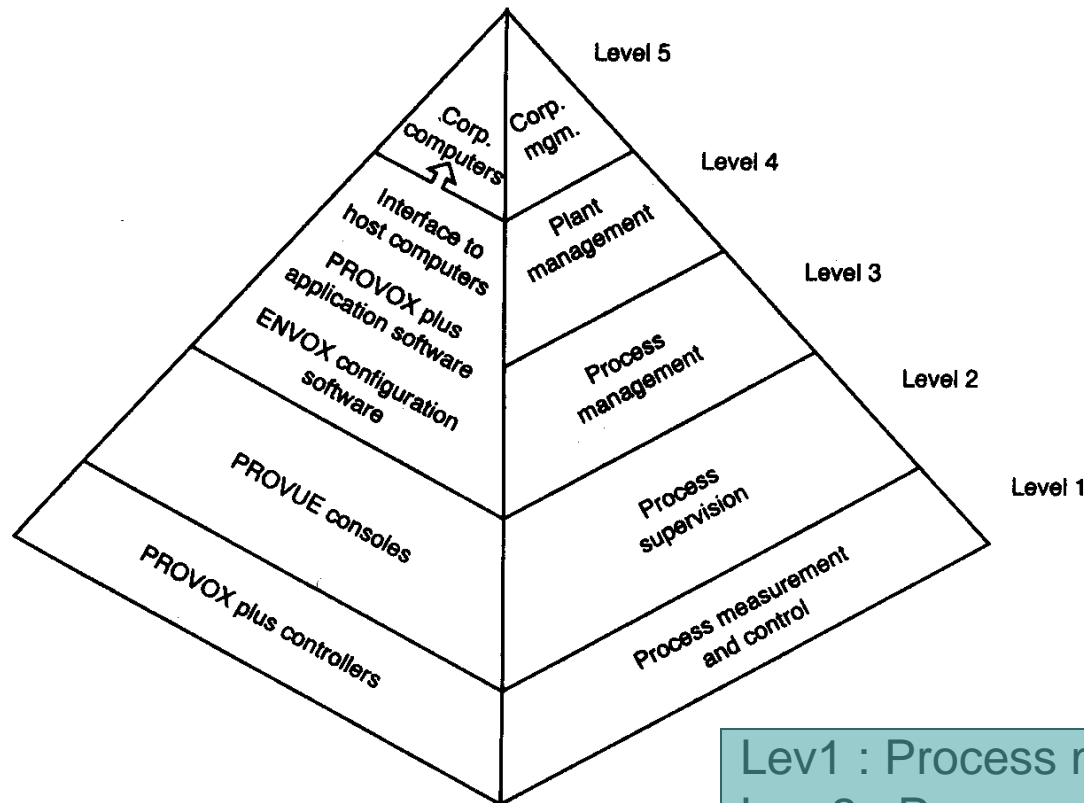


FIGURE 3.22 PROVOX—Hierarchical Top Level

Lev1 : Process measurement and control

Lev 2 : Process supervision

Lev 3 : Process management

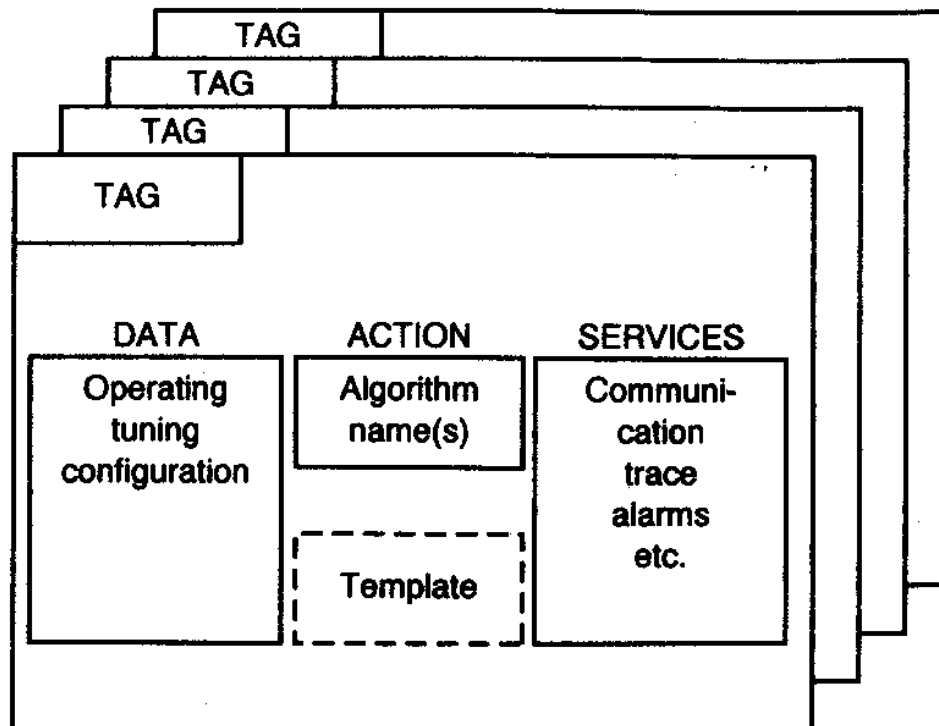
1~3 : Object-Oriented

Lev 4 & 5 : Plant and corporate management

4~5 : dataprocessing repository model

# Lev 2 : Process supervision

- Point : locus of process control





# Point

## ○ Point

- Object-oriented design elements that
  - Encapsulate information about control points of the process
- Data associated with a point
  - Operating parameters
  - Including current process value
  - Setpoint(target value)
  - Value output
  - Mode(automatic or manual)
  - Tuning parameter
    - Gain, reset, derivative, alarm trip-points
  - Configuration parameters
    - Tag(name) , I/O channel



# Point的分析

- Include a template for a control strategy
  - Include procedural definitions
    - Control algorithm
    - Communication connections
    - Reporting services
    - Trace facilities
  - 多个Point通过Communication services进行协作完成一项具体任务





# Hierarchical Layers

- 数据仓库
  - Lev 4 & 5 : 数据仓库repositories
  - Lev 2(report), lev 3 (data-collection)为上层提供支持
- Hierarchical Layers permit
  - Strong separation of different class of functions and clean interface between layers

## 3.5.2 An Interpreter Using Different Idioms for the Components

### ○ Rule-Based System

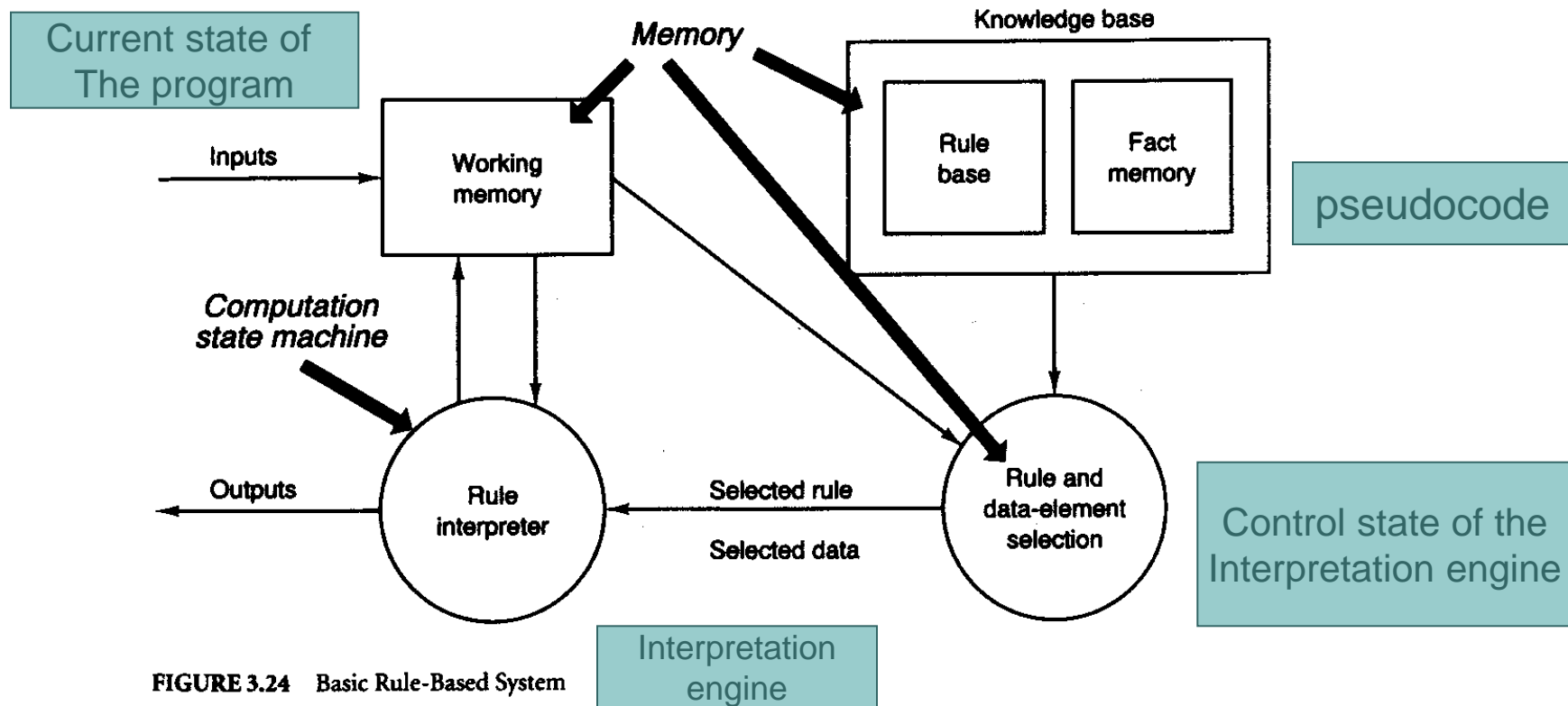


FIGURE 3.24 Basic Rule-Based System

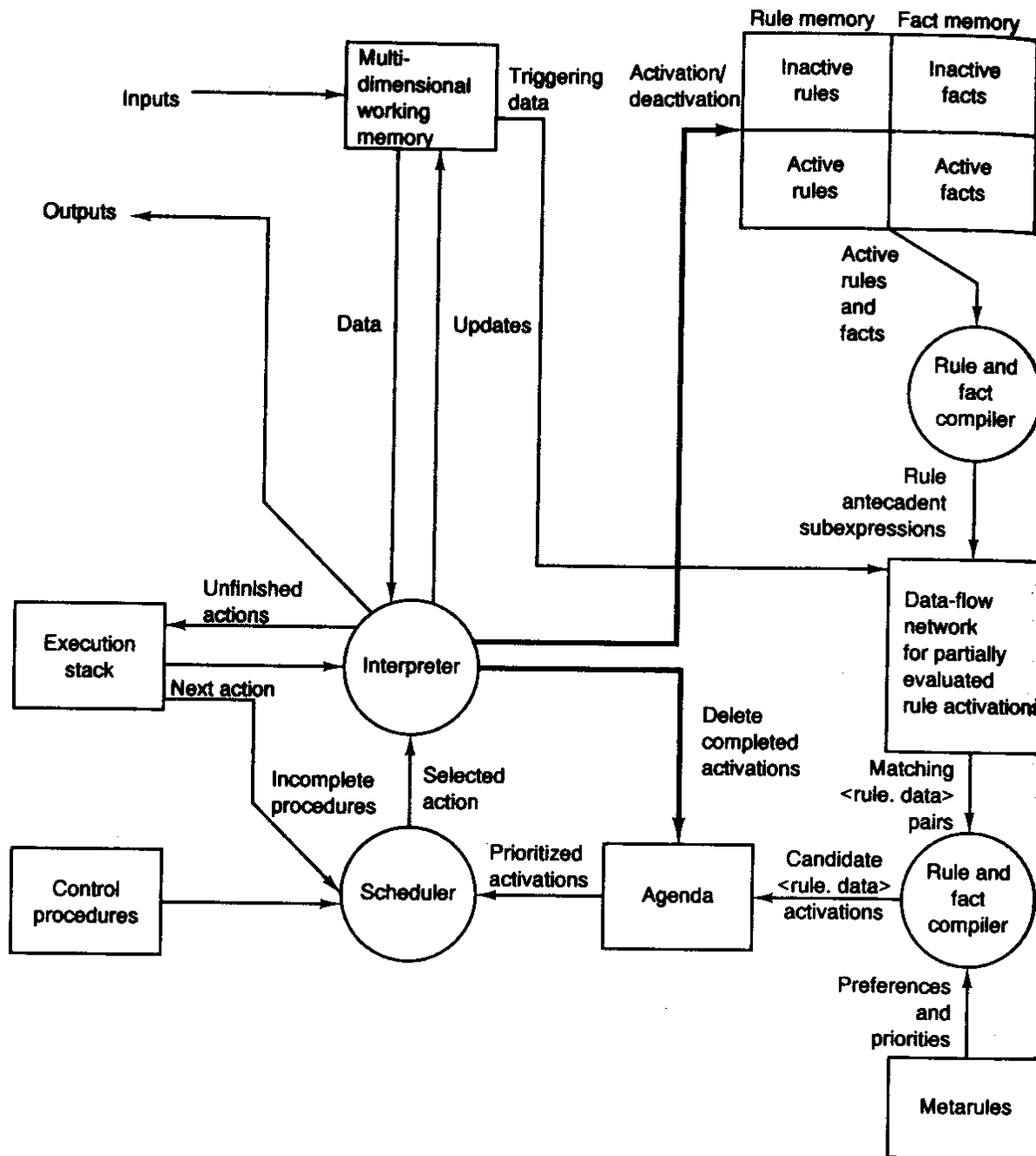
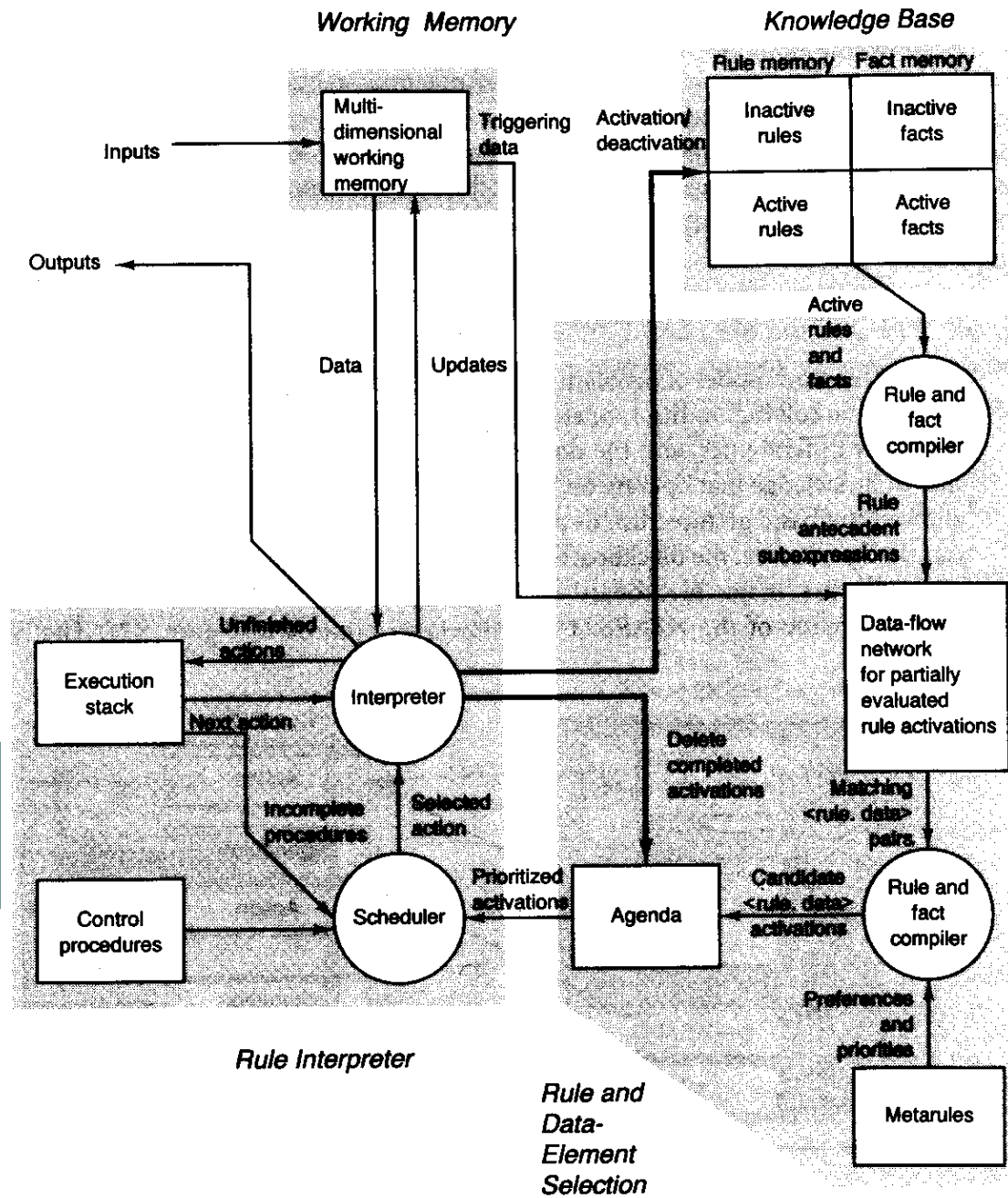
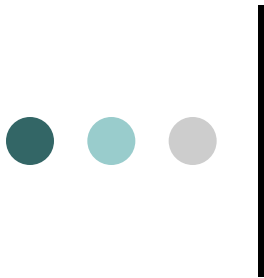


FIGURE 3.25 Sophisticated Rule-Based System



pipeline

Table-driven  
interpreter

FIGURE 3.26 Simplified Rule-Based System

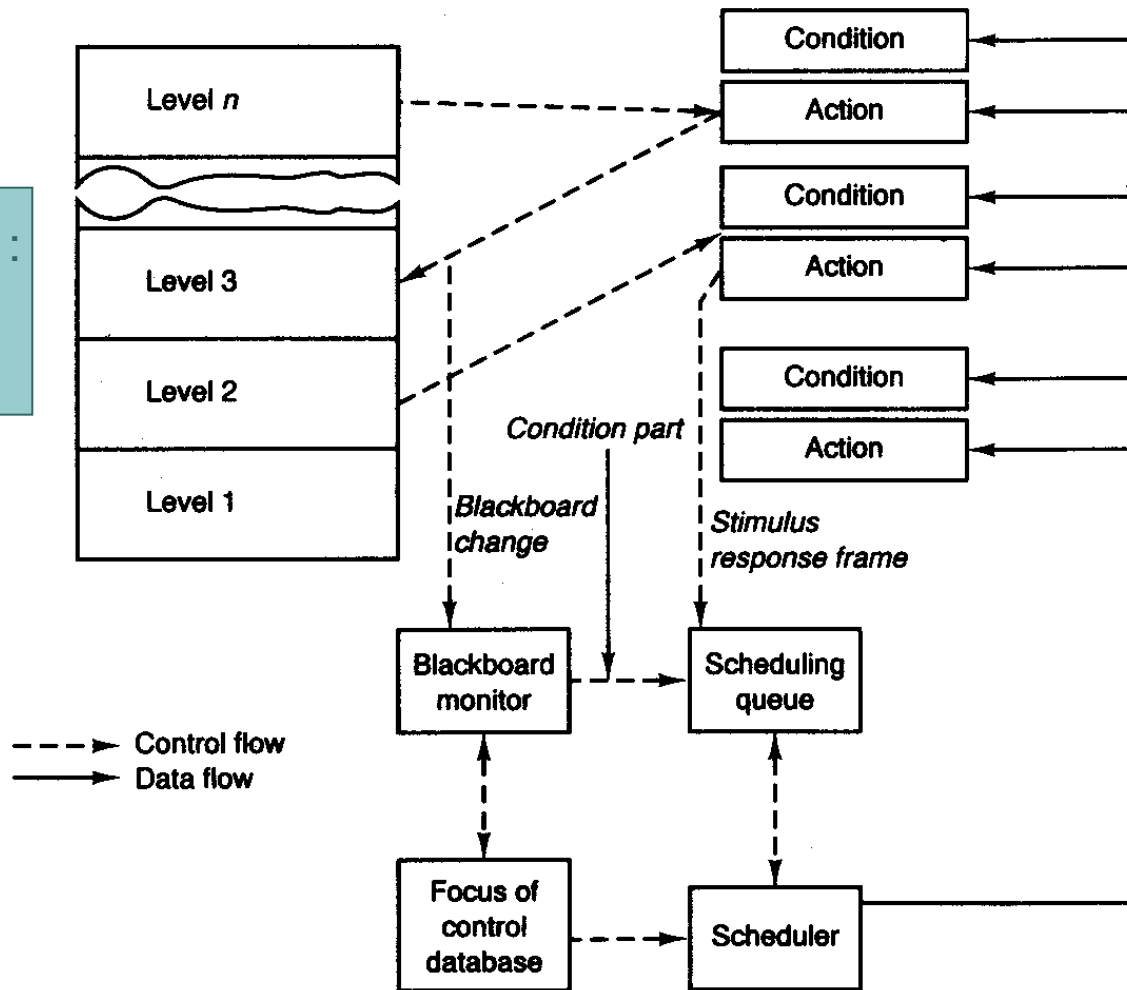


# 结论

- ...the elements ... are elaborated ... to execution characteristics of particular class of languages ...
- Different components ... can be elaborated with different idiom

# 3.5.3 A Blackboard Globally recast as an interpreter

Solution space :  
与应用相关的  
层次结构



在层次内（间）  
进行处理

FIGURE 3.27 Hearsay-II

# Blackboard model + realization of that model by a virtual machine

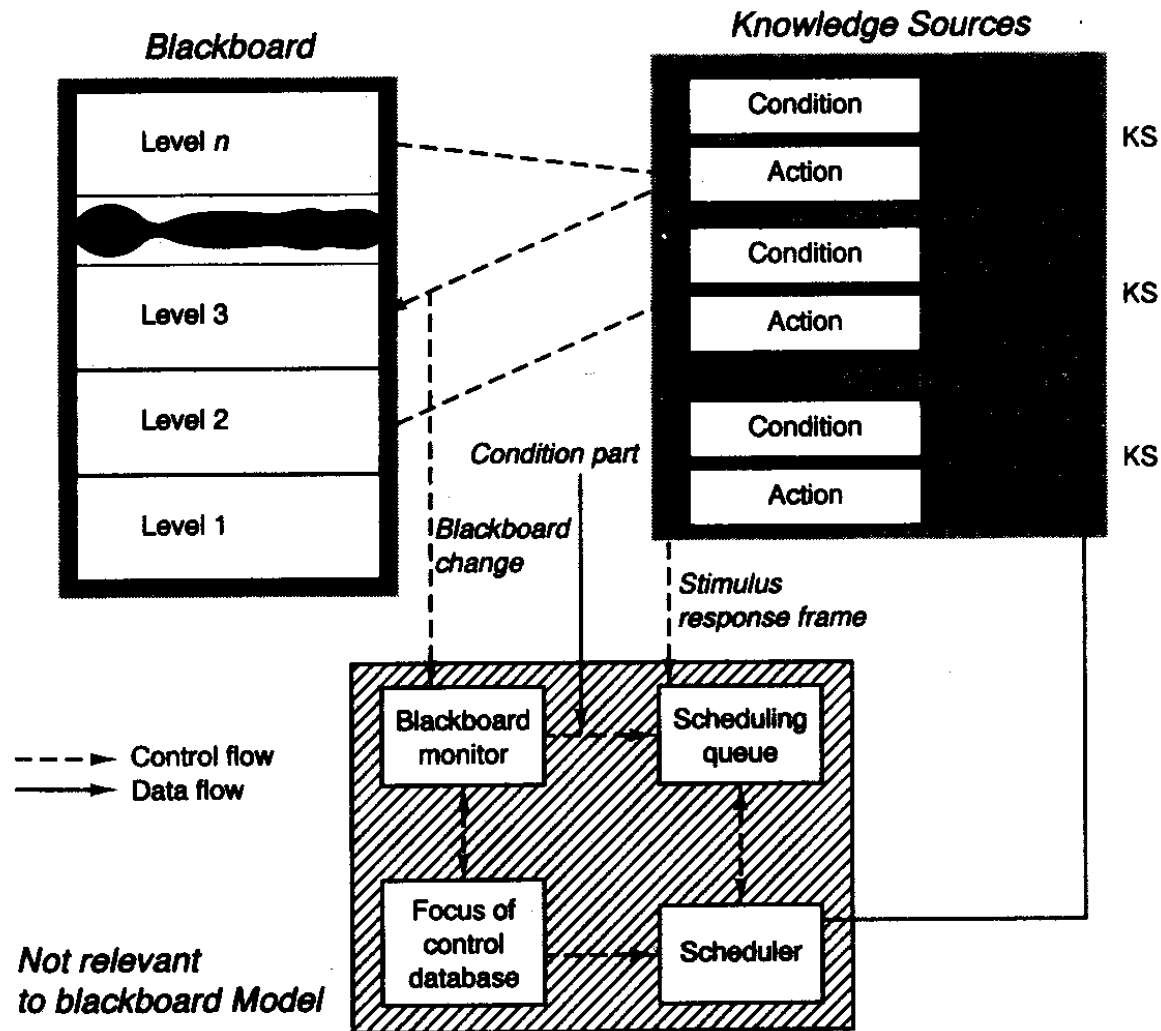


FIGURE 3.28 Blackboard View of Hearsay-II

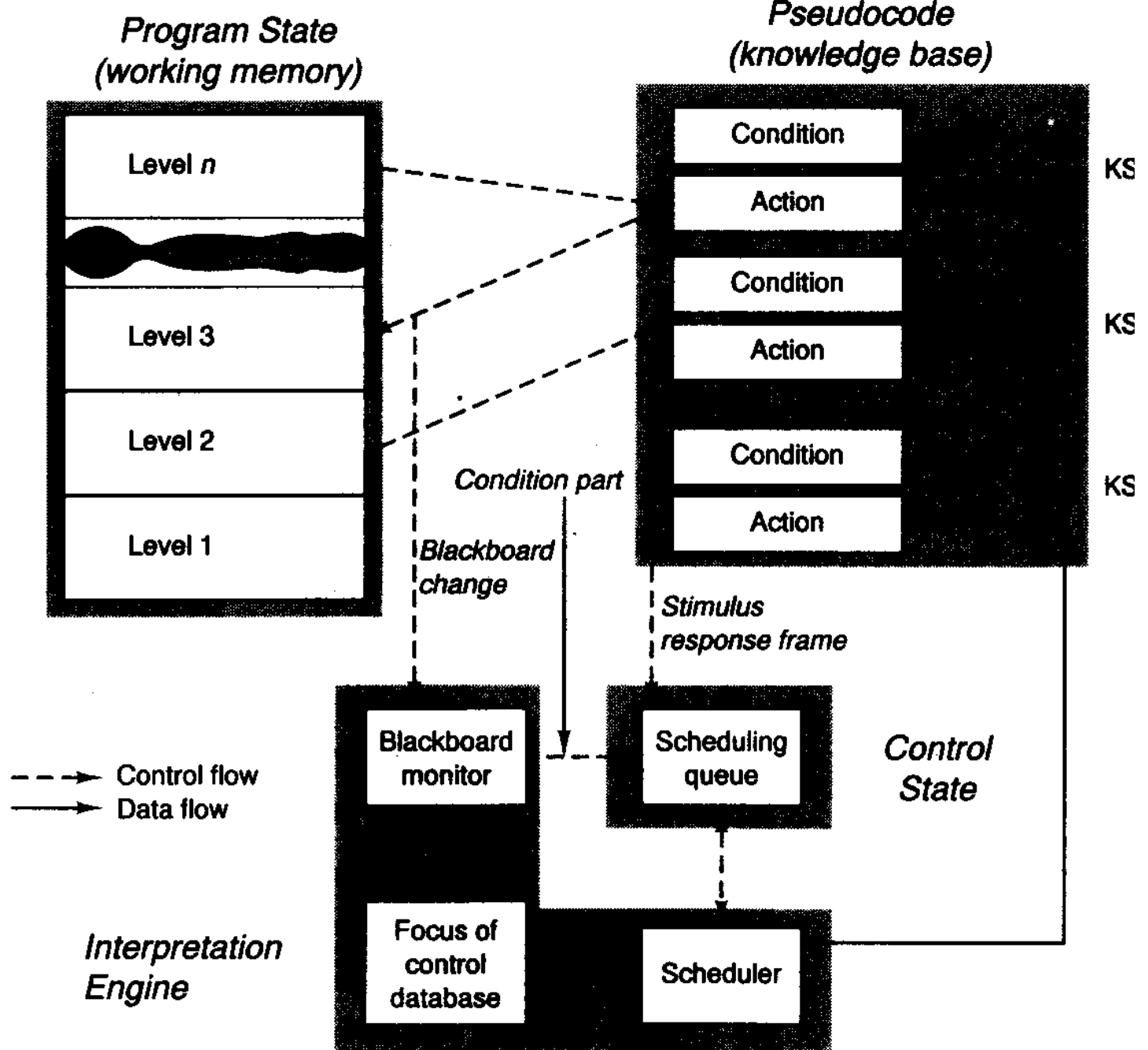


FIGURE 3.29 Interpreter View of Hearsay-II





谢谢!