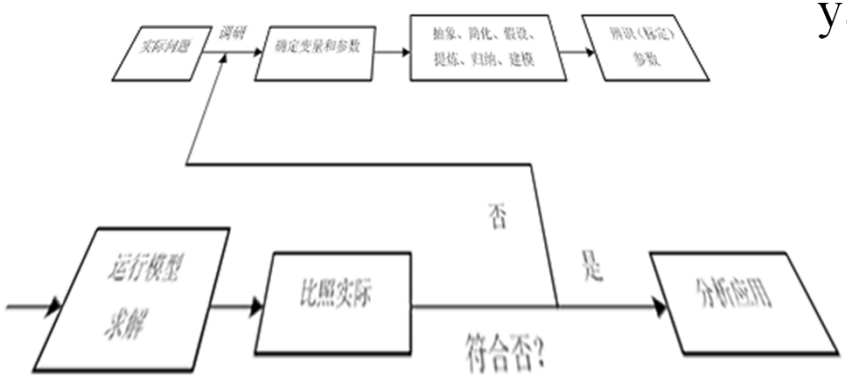
人工系统建模:
理论与工具

姚 远
yaoyuan@shu.edu.cn



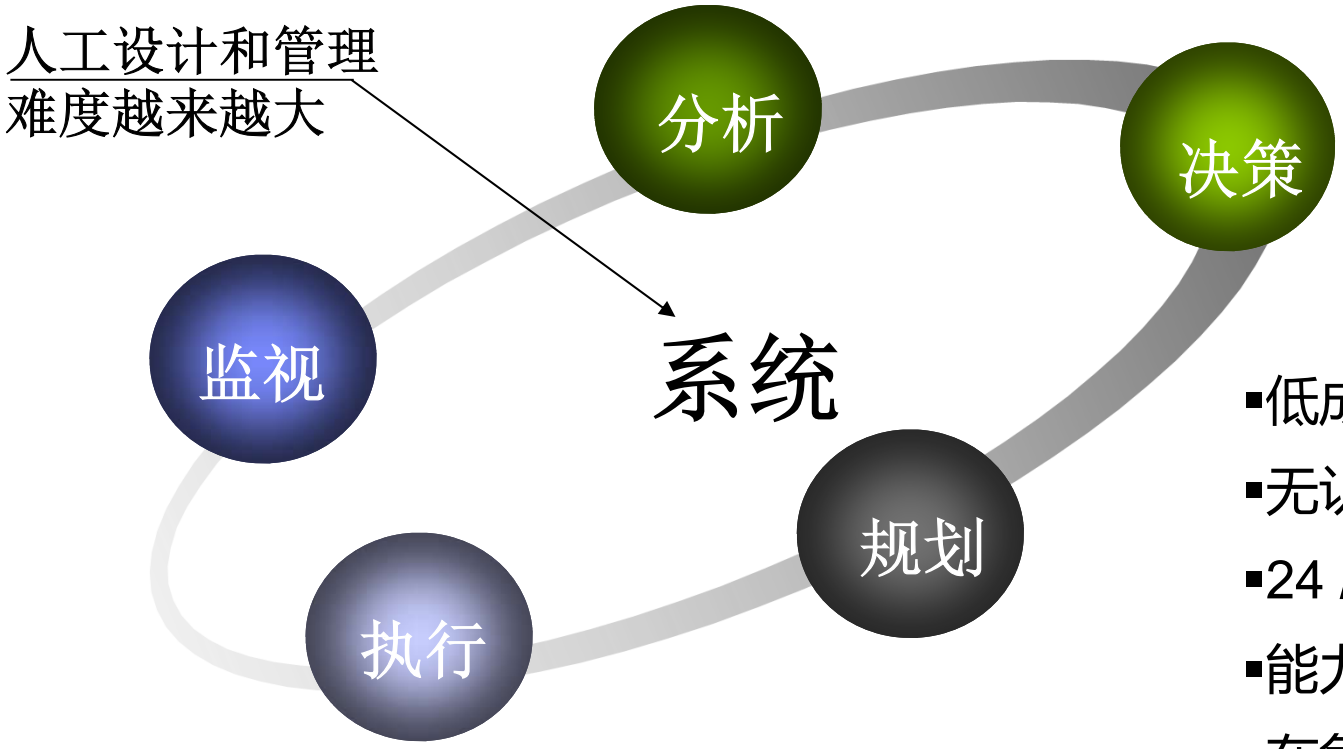
快速制造工程中心

2020/11/5

人工系统是指基于人为设计的规则集合设计的系统。

- 离散
- 复杂
- 庞大

人工系统所面临的问题



- 低成本
- 无认知偏差或逻辑谬误
- 24 / 7的可用性，高可靠性
- 能力可以快速复制
- 在危险的环境中使用
- 不情绪化

管理的需求

- 低成本
- 无认知偏差或逻辑谬误
- 24 / 7的可用性，高可靠性
- 能力可以快速复制
- 在危险的环境中使用
- 不情绪化

这需要

- 高层次的设计
- 自动化的监控
- 知识驱动的决策

内容

- 面向对象
- 统一建模语言
- 专家系统

面向对象的建模

- 来源：美国RAND公司的战争对策与空战系统
- 特点：系统由可重用的模块（对象）组成

面向对象的建模

- **什么是面向对象**
  - 从建模角度考虑，面向对象是一种思维方式
  - 从程序设计调度考虑，面向对象是使用对象、类、继承、封装、聚合、关联、消息、多态性等概念来构造系统的软件开发方法。

一种如何观察世界的世界观

一种如何进行系统构造的方法论

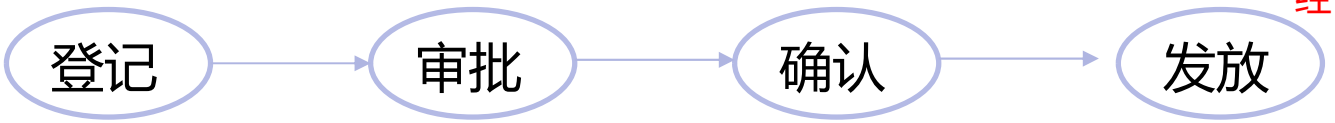
面向对象的建模

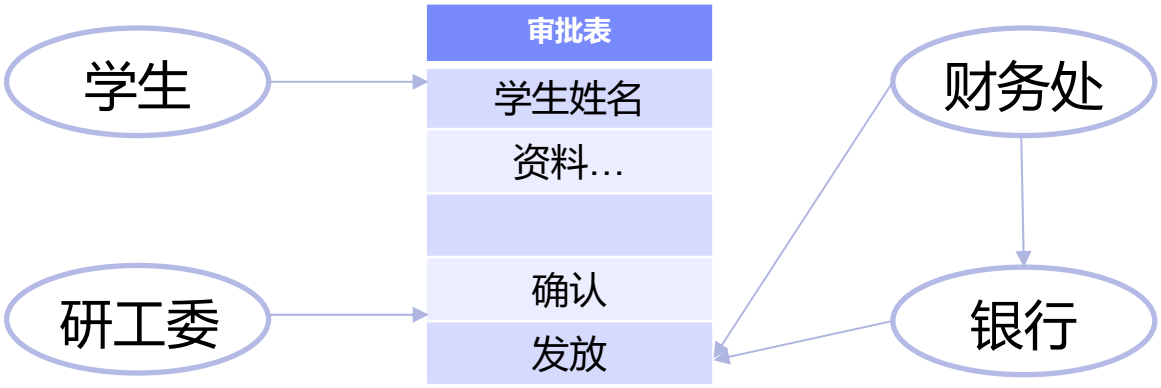
- **系统建模方法比较**

奖学金评审系统

-数据流法

不直接映射问题域
没有按照实际实物组织
经常需要设计多个数据流
导致发生信息膨胀





-面向对象方法

直接映射了问题域
与实际实物形成良好对应

面向对象的建模

- 基本概念
  - 对象： ={接口，数据，操作}
  - 面向对象：一种世界观或方法论

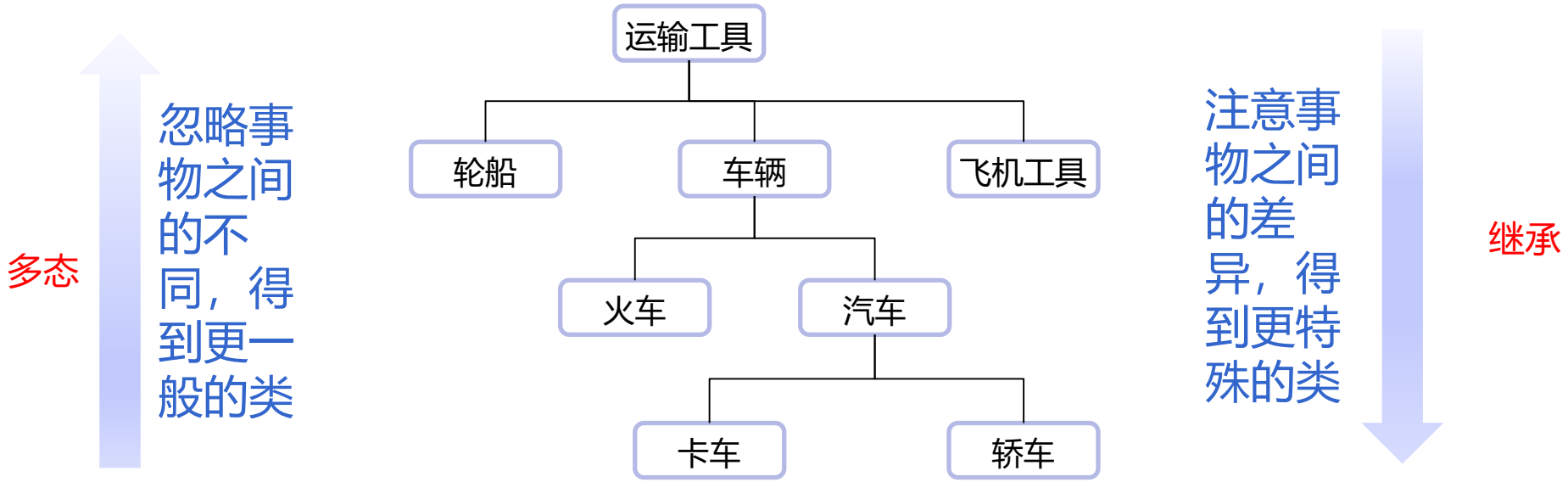
    客观对象 → 概念对象 → 计算机对象
  - 类：具有相似性质的一组对象
  - 继承：共享类中数据和方法的层次化机制
  - 多态：同类型的对象实例对同种消息产生不同的反应
  - 动态联编：系统运行时确定对象的性质
  - 消息：对象之间的通信

面向对象的建模

▪基本概念

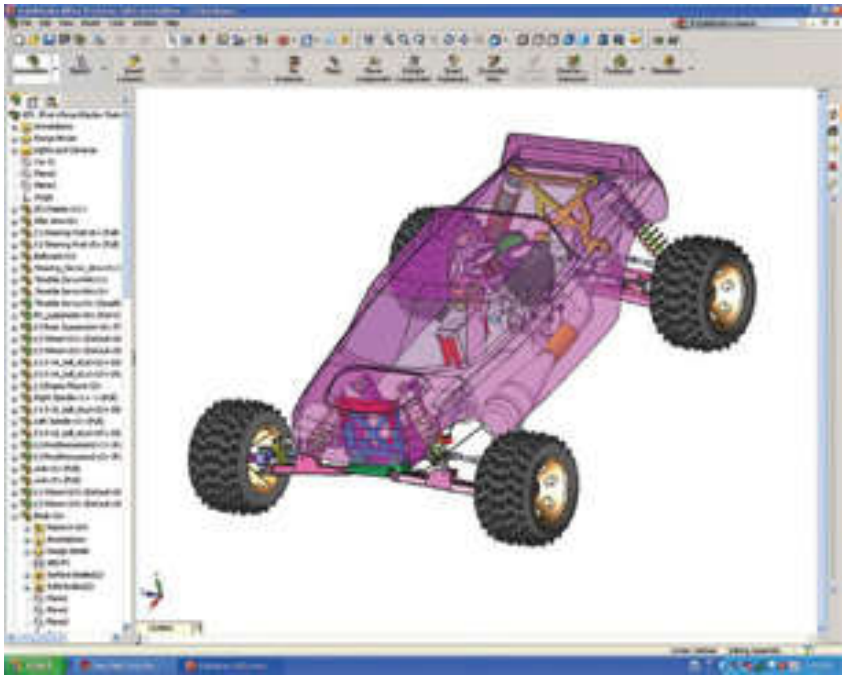
抽象是面向对象中的常用方法:忽略非本质特征，保留本质特征

不同的抽象程度可得到不同层次的类



面向对象的建模

- **优点**
  - 可理解性
  - 模块性、可重用性
  - 松散耦合、可扩充性
  - 图形用户界面
  - 易于与领域知识结合
  - 易于并行仿真



面向对象分析（OOA）

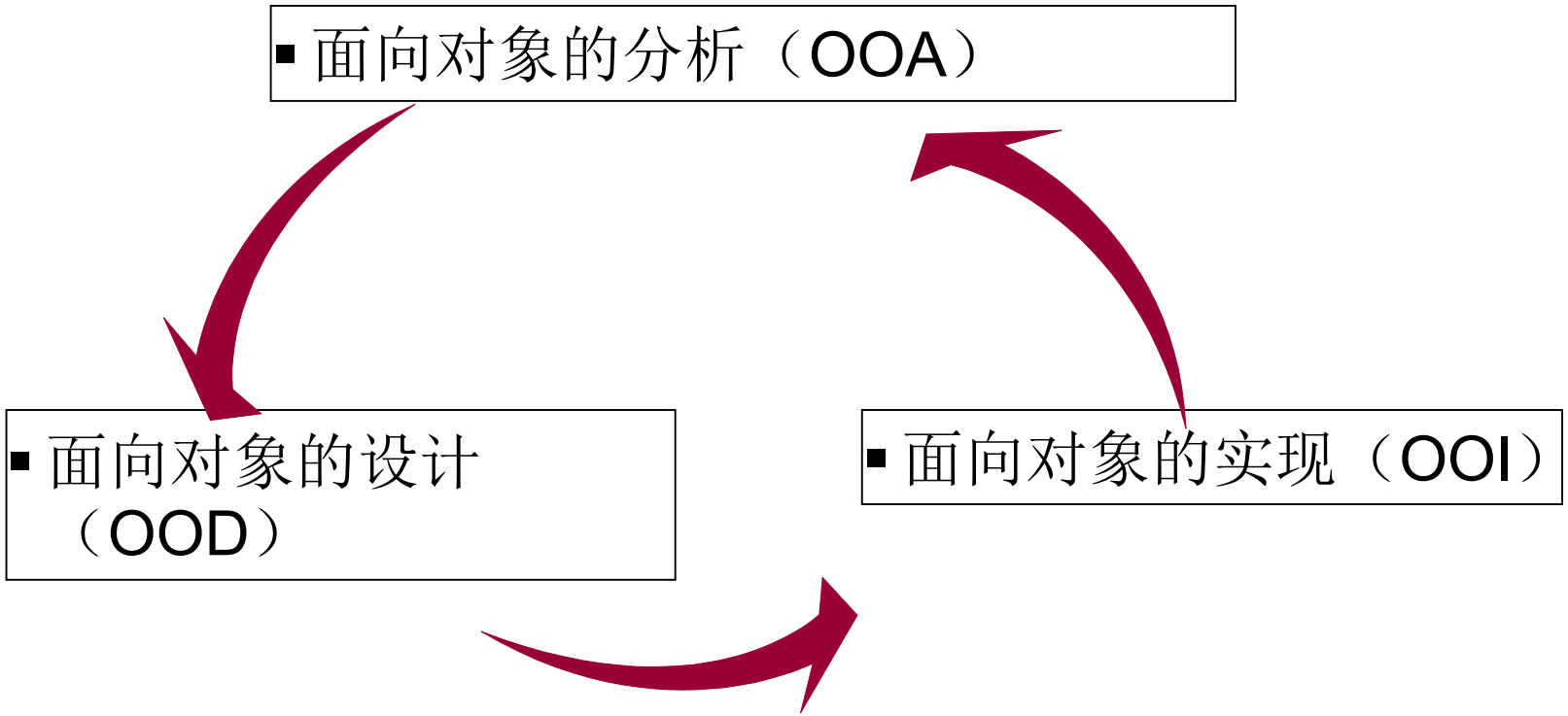
- 识别对象
  - 抽象，类比，经验
- 识别结构
  - 多种对象的组织方式
- 识别主题
  - 系统分析的概貌
- 定义属性
  - 对象的状态
- 定义方法
  - 对象的行为

- 先整体，再部分
  - 抽取和整理需求
  - 建立问题域
  - 建立精确模型

面向对象的设计（OOD）

- 概要设计
  - 明确对象及层次结构
  - 明确交互行为
- 详细设计
  - 细化
  - 添加新类
- 对重用的支持
  - 形成类库

面向对象建模框架



在整个系统分析、开发、维护的生命周期中全程使用面向对象方法

高层建模方法

- 面向对象的思想
- 统一建模语言
- 专家系统

统一建模语言（UML）

Grady Booch

1986提出面向对象建模的概念

Jim Rumbaugh

1980年代末提出OMT方法

Ivar Jacobson

1994年提出OOSE方法，引入外部角色

- UML：统一的建模语言，1997年成为OMG的标准

统一建模语言（UML）

- UML语义

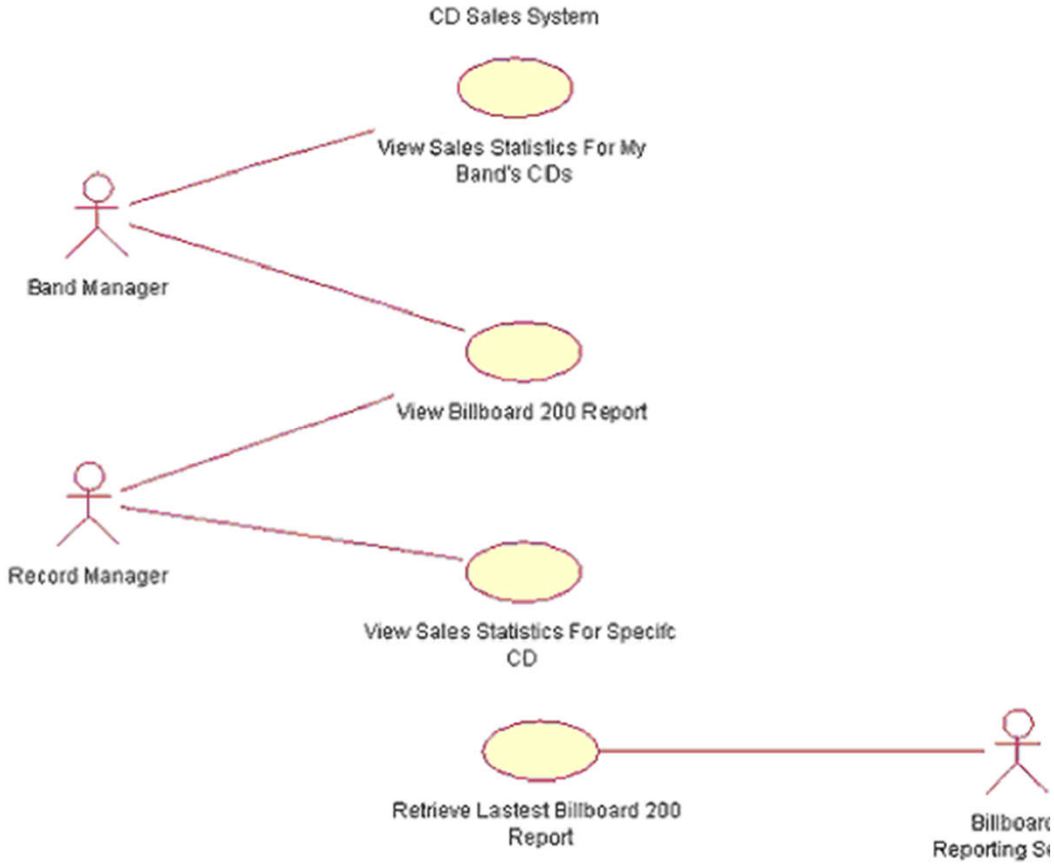
层次	描述	举例
元-元模型	**元建模体系结构的基础构造。** 定义了描述元模型的语言。	*元类、元属性、元操作*
元模型	**元-元模型的实例。** 定义了描述模型的语言。	*类、属性、操作、构件*
模型	**元模型的实例。** 定义了描述信息域的描述语言的模型。	*StockShare,askPrice, sellLimitOrder,StockQuote Server.*
用户对象 （用户数据）	**模型的实例。** 定义了一个特定的信息论域。	*<Acme_Software_Share 98789>, 654.56,sell_limit_order, <Stock_Quote_Svr 32123>*

- UML比较复杂，它由近90个元类组成，100多个元关联和近50个型板组成。

统一建模语言（UML）

- UML图形表示

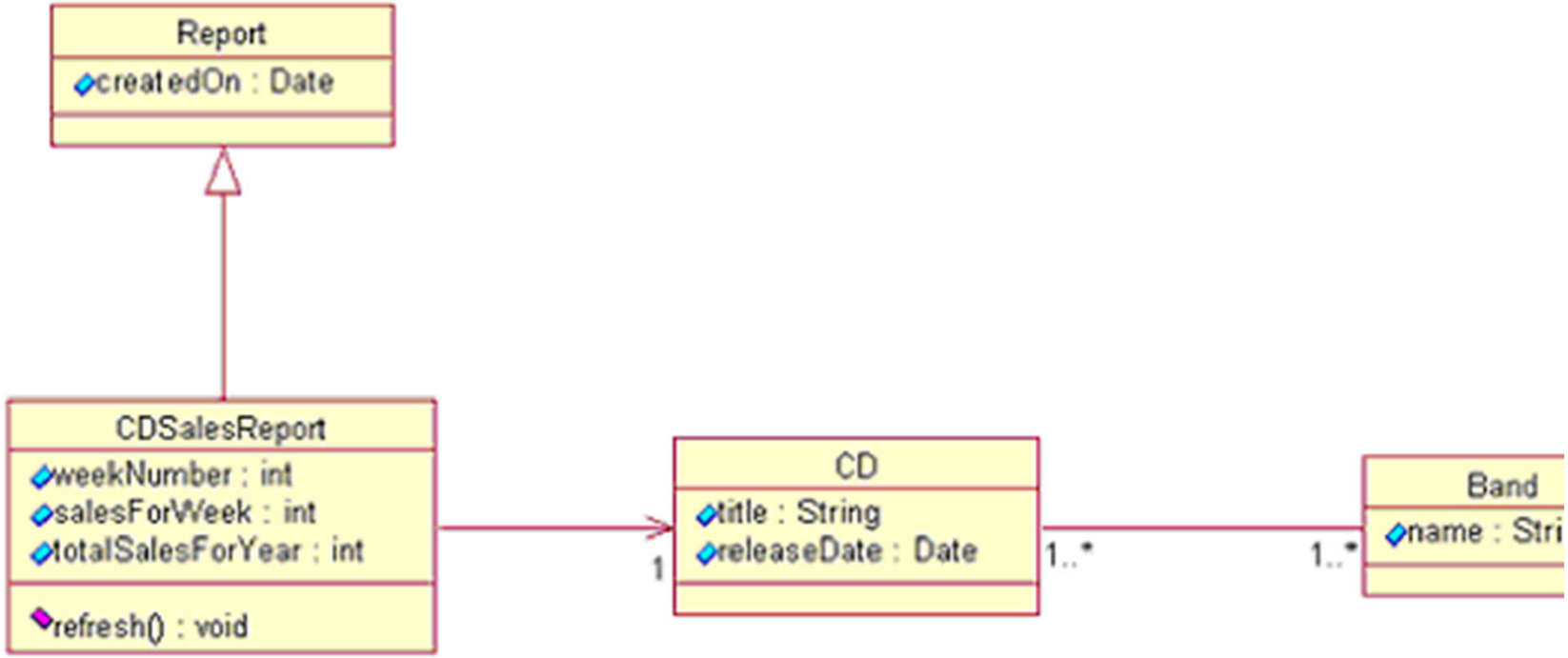
1.用例图



统一建模语言（UML）

- UML图形表示

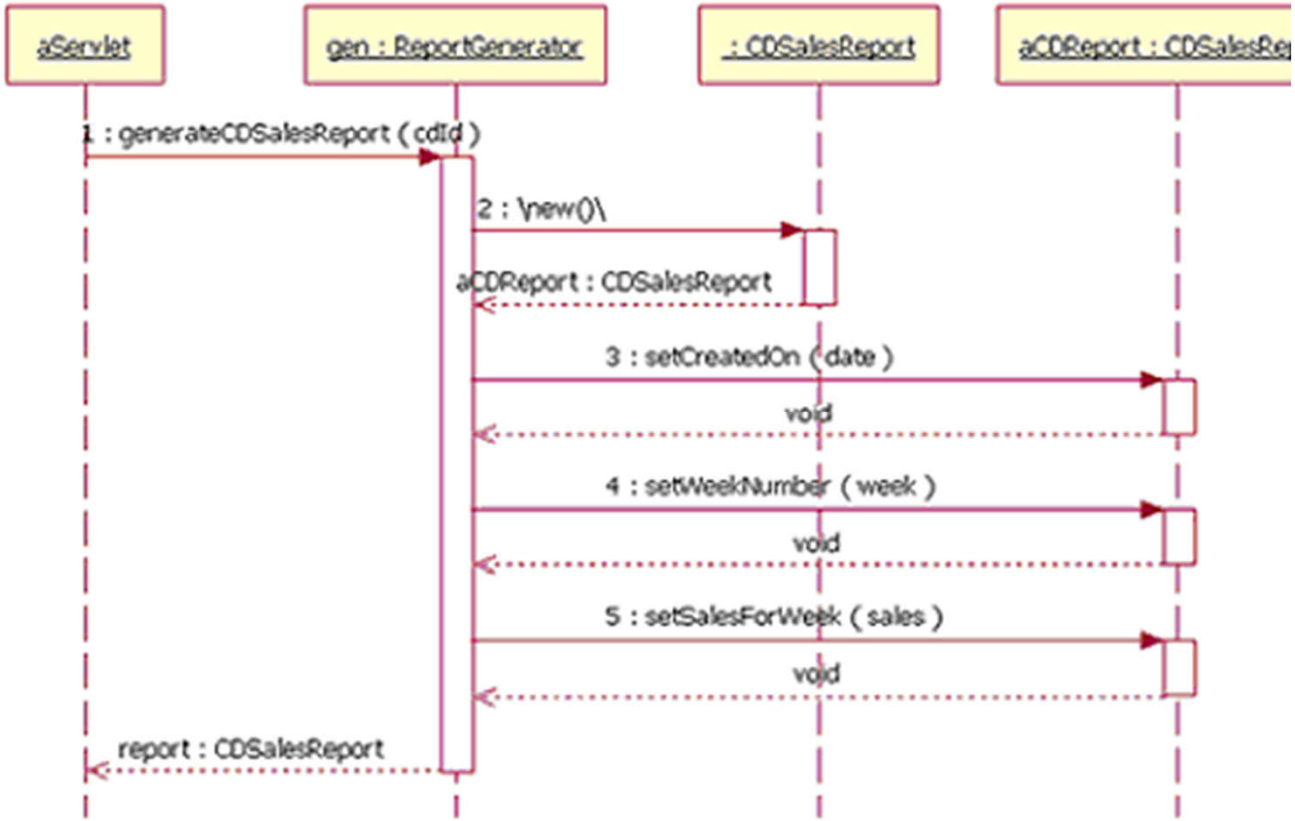
2.类图



统一建模语言（UML）

- UML图形表示

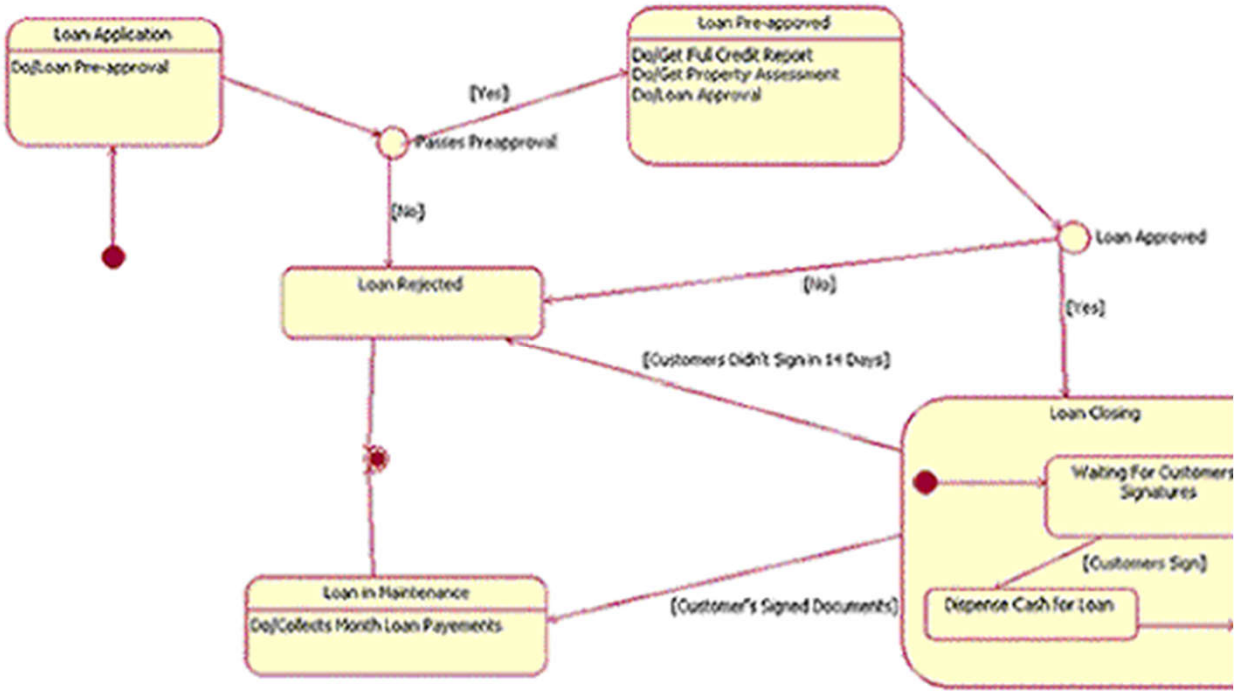
3.交互图



统一建模语言（UML）

- UML图形表示

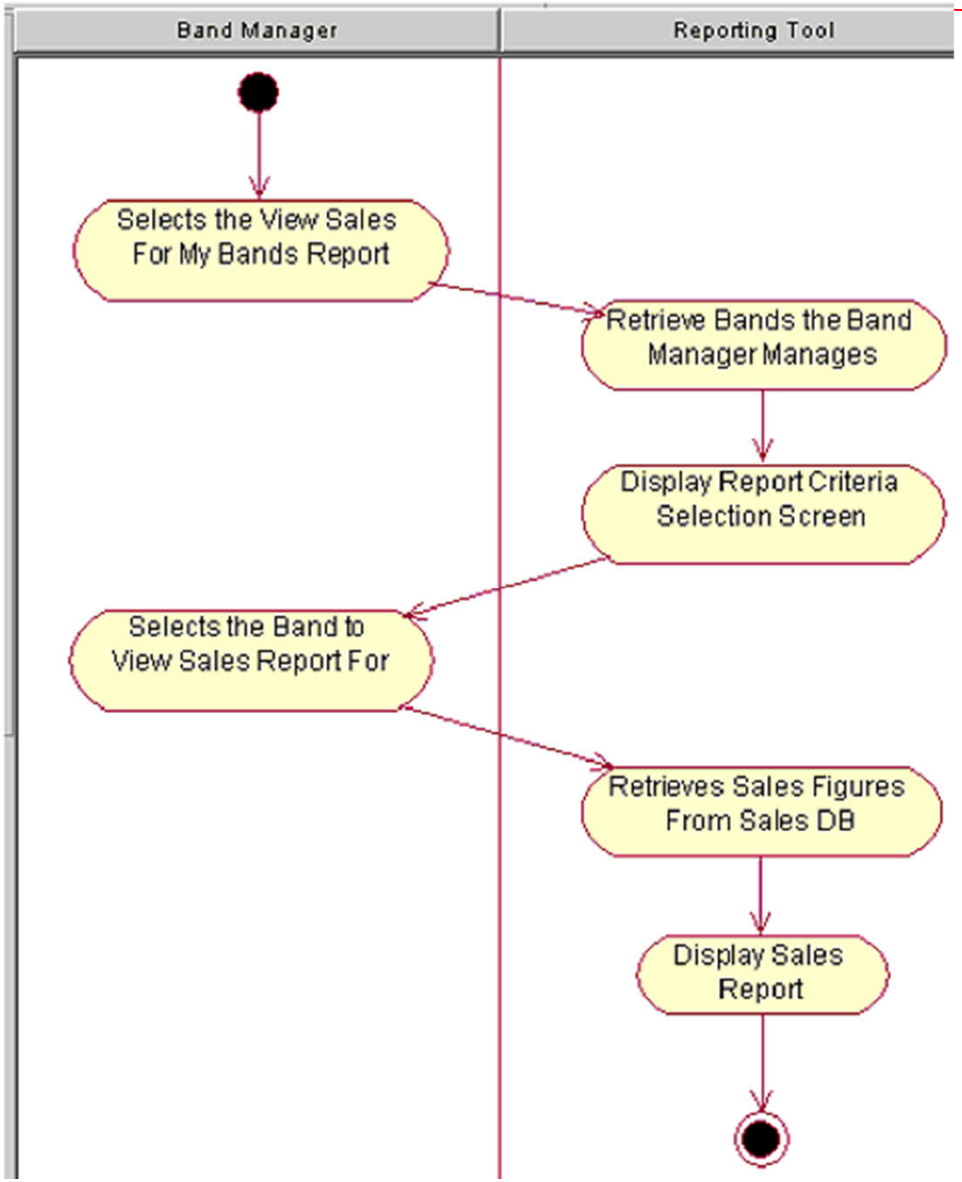
4.状态图



统一建模语言（UML）

- UML图形表

5.活动图

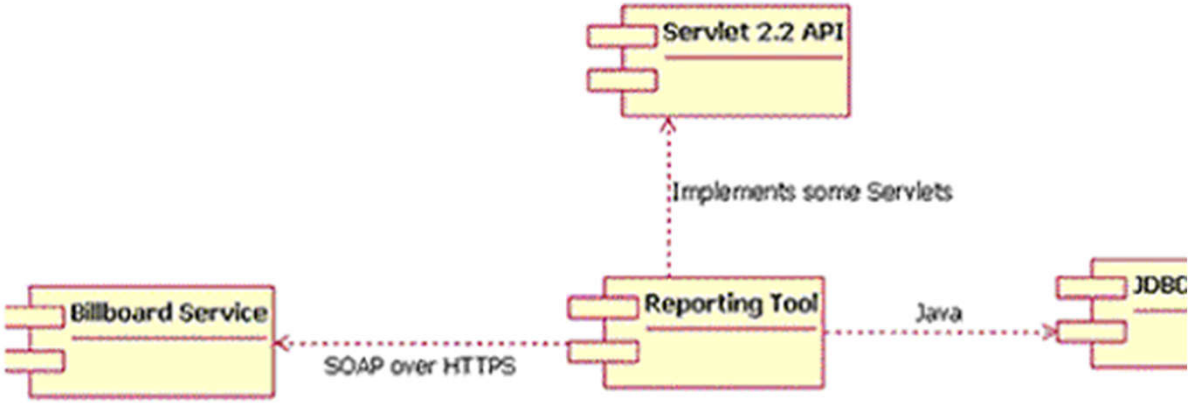


统一建模语言（UML）

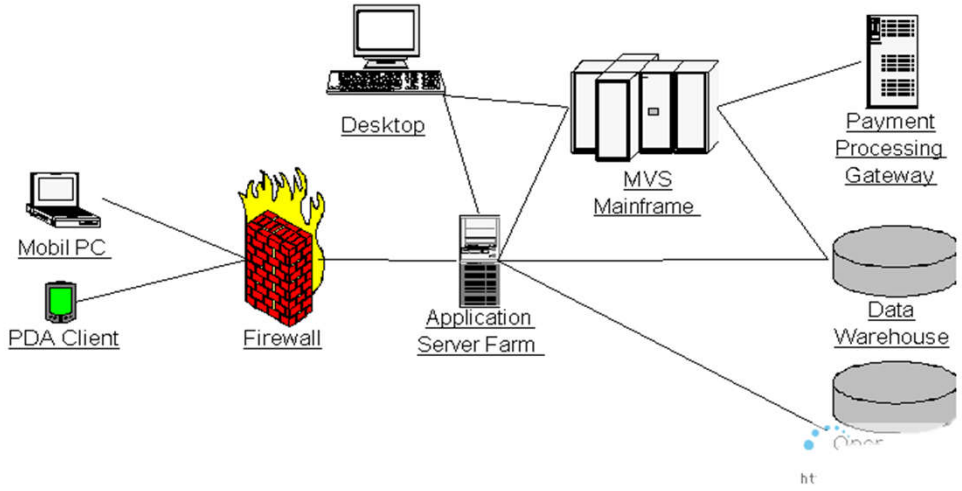
- UML图形表示

6.实现图

组件图



部署图



统一建模语言（半形式化表达）

- UML的形式化
  - UML使用形式化的规格说明描述模型的体系结构
  - 形式化语言只能准确表达静态结构
  - 需要加以自然语言辅助进行说明

人工系统架构设计与实现

- 构建复杂系统要求具有了解并管理复杂关系的特别能力
  - 面向对象的方法给出了一个基本的**方法论**
  - Rose等工具为复杂系统建模提供**模型框架**
  - UML等高层次语言为实现各类框架提供了**建模语言**

高层建模方法

- 面向对象的建模与仿真
- 统一建模语言
- 专家系统

专家系统

- 专家系统依靠知识进行推理决策
- 处理自动化决策的有效工具

如何实现?

知识与推理

亚里士多德（前384-前322）

▪古希腊哲学家亚里士多德是首先试图严格定义正确思考的人之一，他将“正确思考”定义为**不能辩驳的推理过程。**

▪他三段论提供了一种机制——允许在初始前提的条件下**机械的推导出结论。**

1）如果所有人（M）都是必死的（P），（大前提）

2）并且所有希腊人（S）都是人（M），（小前提）

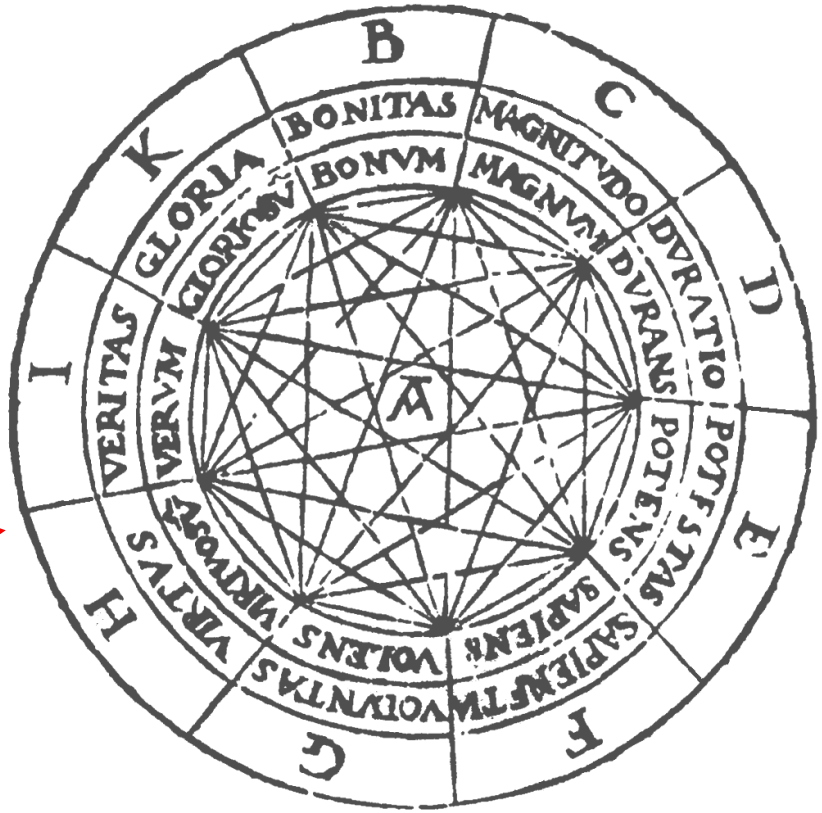
3）那么所有希腊人（S）都是必死的（P）。（结论）

知识与推理

拉蒙·柳利（1232-1315）

- Raomon Lull提出一种思想，认为推理确实可以用机械装置完成。

物理意识



概念之轮

知识与推理

- 如果我们具有一个物理意识（概念之轮），实现推理

那所用的知识从何而来?

知识与推理

弗朗西斯 培根（1561-1626）

- Francis Bacon在《新工具论》中提出了经验主义的观点，相信现代科学方法，认为理论应建立于对于事物的观察，而不是直觉或迷信。

经验 → 知识

知识与推理

Rudolf Carnap（1891-1970）

- Rudolf Carnap在《世界的逻辑结构》中定义了一个清楚的计算过程，用于从基本的实验中抽取知识。这可能是第一个把意识当作计算过程的理论。。

知识 → 规则集合

意识 → 物理系统

这就是专家系统最初的来源，把决策过程物理化。

专家系统

领域专家

概念抽象

知识工程师

编程

知识获取

知识解释

演绎搜索

冲突解决

推理引擎

模式匹配

动作执行

知识库

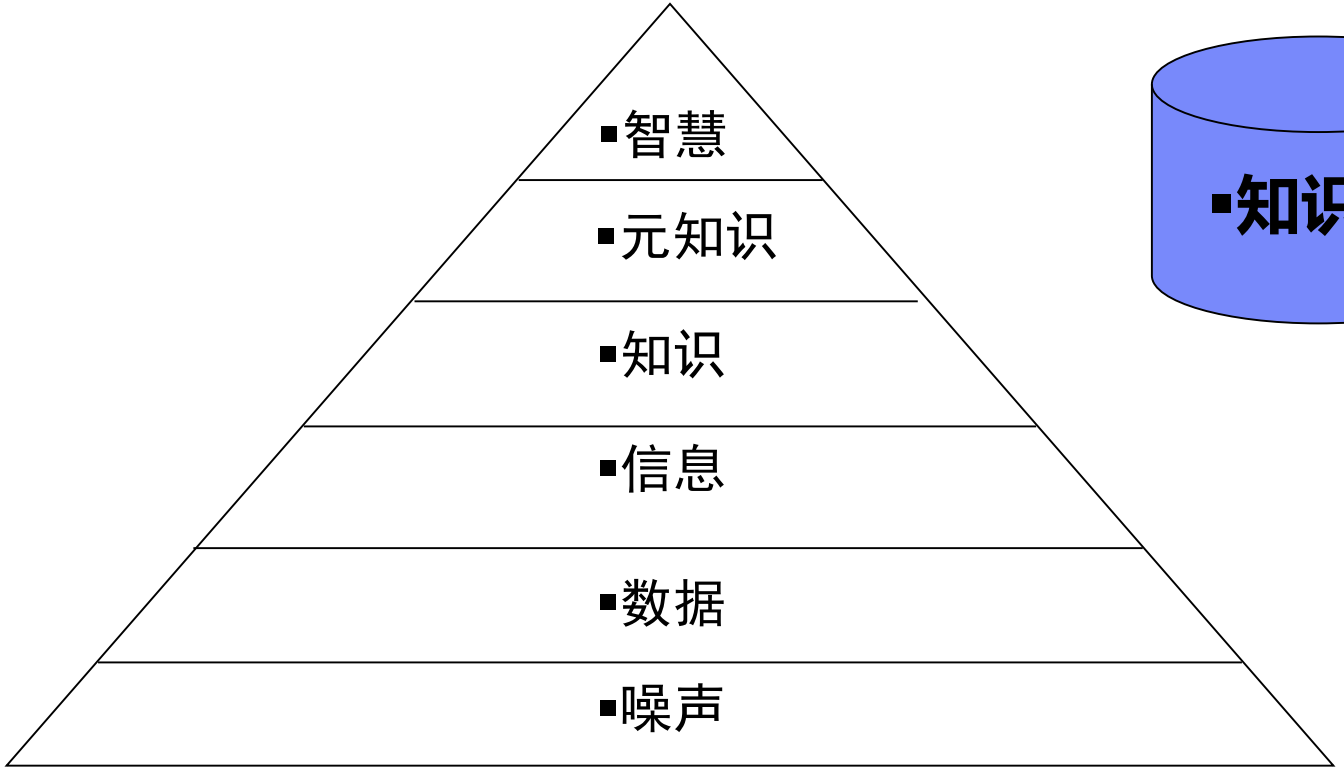
- 根据知识进行推理
- 用启发式或类比方式解决问题
- 基于已知的事实解释并证明

专家系统

查询

咨询用户

知识的层次



知识的类别

- 事实
- 过程
- 行为
- 实例
- 类比

知识的表示原则

- 表示知识的范围是否广泛?
- 是否适合于推理?
- 是否适合于计算机处理?
- 是否有高效的算法?
- 能否表示不精确知识?
- 能否模块化，以便于知识分层?
- 知识和元知识能否用统一的形式表示?
- 是否适合于加入启发式信息?
- 过程性表示还是说明性表示?

知识的表示形式

- **演绎系统**：采用命题和谓词演算进行推理
- **产生式系统**：指形如α→β或IF（α）THEN（β）
- **框架结构**：框架名（frame）和一组用于描述框架各方面具体属性的槽(Slot)组成（可以嵌套）
- **语义网络**：有向图表达的关系结构
- **过程性表示**：强调知识的动态特性，表达交互关系。
- **面向对象表示**: 封装对象的知识和行为
- **基于本体的表示**：定义语义符号含义，构造扩展体系

基于演绎系统的专家系统

专家系统是一种典型的逻辑智能体

- 数理逻辑的经典部分—演绎系统包含命题逻辑和一阶谓词逻辑

- 演绎逻辑同时作为专家系统的**知识表示**方法和**推理**方法。

演绎系统-命题逻辑

在现代哲学、逻辑学、语言学中，命题是指一个判断（陈述）的语义

- 一种经典二值逻辑：非真即假

语义	（原子）命题	真值	解释
2+2=4	*p*	= TRUE	
二月份有30天	*feb_has_30days*	= FALSE	
2018年房价大跌	*houseprice_crash*	= FALSE	
人类能登上火星	*human_wbn_mars*	= FALSE	
2月3号放寒假	*q*	= TRUE	

演绎系统-命题逻辑

命题逻辑研究复合命题之间的推导关系

- 命题可以通过逻辑连接符连接→复合命题

- 否定¬
- 合取∧
- 析取∨
- 蕴涵→
- 等价↔

优先级

原子命题	复合命题
p	$p \wedge feb\_has\_30days \vee \neg q$

演绎系统-命题逻辑

命题确定子句是命题逻辑的语言，它不允许存在不确定性

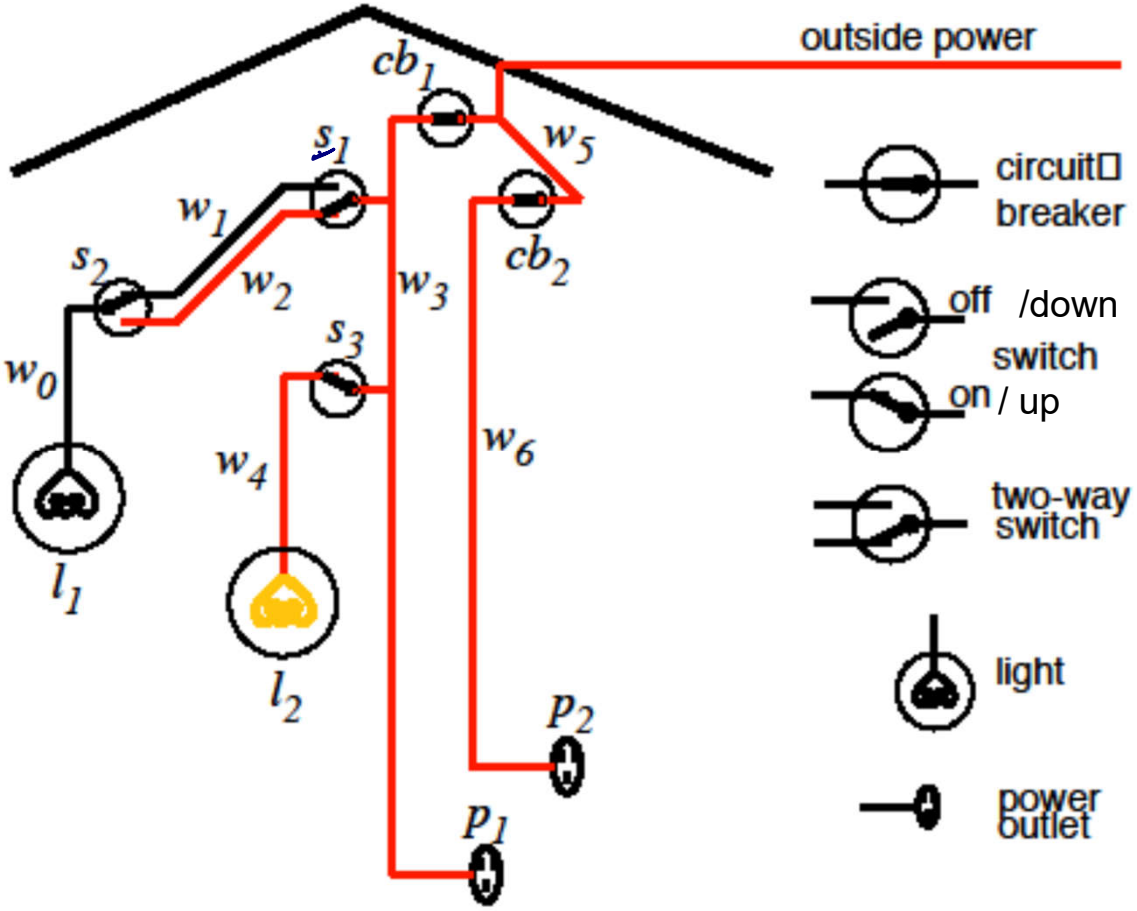
- **主体（body）** 是原子子句或多个原子的合取，支持递归的形式。
  例如$b_1 \wedge b_2$ ，这里 b_1 and b_2 都是主体

  ▪例子: $p_1 \wedge p_2$;
  $ok\_w_1 \wedge live\_w_0$

- **确定子句**是
- - 1个原子子句或
- - 规则 $h \leftarrow b$，**这里*h*是一个**
  原子子句的头部（**Head**）， ***b*是主体**

  ▪例子: $p_1 \leftarrow p_2$;
  $live\_w_0 \leftarrow live\_w_1 \wedge up\_s_2$

命题逻辑-知识库



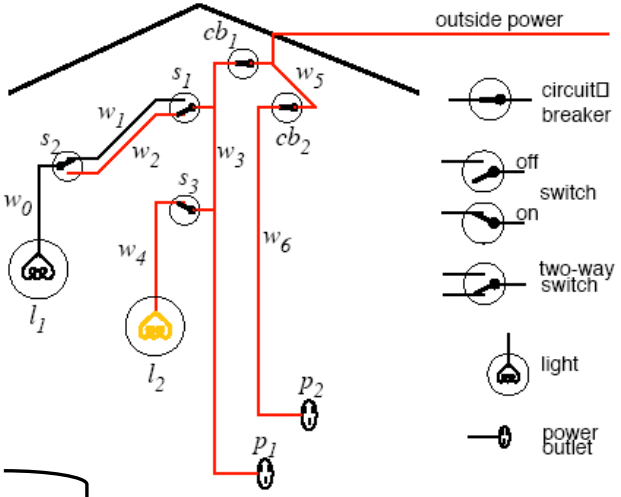
light_l1.
light_l2.
ok_l1.
ok_l2.
ok_cb1.
ok_cb2.
live_outside.

live_w0 ← live_w1 ∧ up_s2.
live_w0 ← live_w2 ∧ down_s2.
live_w1 ← live_w3 ∧ up_s1.
live_w2 ← live_w3 ∧ down_s1.
live_l2 ← live_w4.
live_w4 ← live_w3 ∧ up_s3.
live_p1 ← live_w3.
live_w3 ← live_w5 ∧ ok_cb1.
live_p2 ← live_w6.
live_w6 ← live_w5 ∧ ok_cb2.
live_w5 ← live_outside.
lit_l1 ← light_l1 ∧ live_l1 ∧ ok_l1.
lit_l2 ← light_l2 ∧ live_l2 ∧ ok_l2.

命题确定子句构成的知识库

原子

light_l1.
light_l2.
ok_l1.
ok_l2.
ok_cb1.
ok_cb2.
live_outside.



规则

live_l1 ←live_w0.
live_w0 ←live_w1 ∧up_s2.
live_w0 ←live_w2 ∧down_s2.
live_w1 ←live_w3 ∧up_s1.
live_w2 ←live_w3 ∧down_s1.
live_l2 ←live_w4.
live_w4 ←live_w3 ∧up_s3.
live_p1 ←live_w3.
live_w3 ←live_w5 ∧ok_cb1.
live_p2 ←live_w6.
live_w6 ←live_w5 ∧ok_cb2.
live_w5 ←live_outside.
lit_l1 ←light_l1 ∧live_l1 ∧ok_l1.
lit_l2 ←light_l2 ∧live_l2 ∧ok_l2.

- **确定子句**是
- - 1个原子子句或
- - 规则 ***h*** ← ***b***，**这里*h*是一个原子子句（头），*b*是主体**

测试：以下有多少个合法的确定子句?

▪A. 3 ▪B. 4 ▪C. 5 ▪D. 6 ▪E. 7

a) *class_is_fun*

合法 不合法

b) *class_is_fun* ∨ *class_is_boring*

计算机不懂我们在讨论什么！！

c) *class_is_fun* ← *learn_useful_techniques*

d) *class_is_fun* ← *learn_useful_techniques* ∧ *notTooMuch_work*

e) *class_is_fun* ← *learn_useful_techniques* ∧ ¬ *TooMuch_work*

f) *class_is_fun* ← *f(time_spent, material_learned)*

g) *srtsyj* ← *errt* ∧ *gffdgdgd*

命题逻辑-知识库

语义可以让我们将逻辑中符号和领域知识联系起来

- **解释（定义）**
  解释 I为每个原子设定一个真值（可以理解为KB当前状态）

我们可以利用解释来确定子句的真值

- **语句的真值（定义）**
  - **主体 $b_1 \wedge b_2$** 为真当且仅当I中 b_1 和b_2 都为真
  - **规则 $h \leftarrow b$** 为假（在I中）当且仅当 b为真与h为假

命题逻辑-知识库

知识库（KB）是多组命题确定子句的集合，知识库的元素是公理，

一组命题的**模型**是使该组命题都为真值的一个解释

如果g在KB的每个模型中都为真，则g是KB的一个逻辑结论

$KB \models g$

- 逻辑结论使计算机可以在不知道真实世界的语义的情况，仍然可以进行正确的推导

命题逻辑-知识库

真值表：在不同解释下的真值

F=false, T=true

	a_1	a_2	$a_1 \wedge a_2$
I_1	F	F	F
I_2	F	T	F
I_3	T	F	F
I_4	T	T	T

	h	b	$h \leftarrow b$
I_1	F	F	T
I_2	F	T	F
I_3	T	F	T
I_4	T	T	T

命题逻辑-知识库

知识库的**模型**是使该知识库每个子句都为真的一个解释

$$KB = \begin{cases} p \leftarrow q \\ q \\ r \leftarrow s \end{cases}$$

以下哪个解释是KB的模型?

	p	q	r	s
I_1	T	T	T	T
I_2	F	F	F	F
I_3	T	T	F	F
I_4	T	T	T	F
I_5	F	T	F	T

- A. I_3
- B. I_1 , I_3
- C. I_1, I_3, I_4
- D. 所有

命题逻辑-知识库

知识库的**模型**是使该知识库每个子句都为真的一个解释

$$KB = \begin{cases} p \leftarrow q \\ q \\ r \leftarrow s \end{cases}$$

以下哪个解释是KB的模型?

	p	q	r	s	$p \leftarrow q$	q	$r \leftarrow s$	KB
I_1	T	T	T	T				
I_2	F	F	F	F				
I_3	T	T	F	F				
I_4	T	T	T	F				
I_5	F	T	F	T				

命题逻辑-知识库

知识库的**模型**是使该知识库每个子句都为真的一个解释

$$KB = \begin{cases} p \leftarrow q \\ q \\ r \leftarrow s \end{cases}$$

以下哪个解释是KB的模型？ I_1 I_3 I_4

	p	q	r	s	$p \leftarrow q$	q	$r \leftarrow s$	KB
I_1	T	T	T	T	**T**	**T**	**T**	**T**
I_2	F	F	F	F	T	F	T	F
I_3	T	T	F	F	**T**	**T**	**T**	**T**
I_4	T	T	T	F	**T**	**T**	**T**	**T**
I_5	F	T	F	T	F	T	F	F

我们在做什么？

我们希望利用演绎系统表达知识，完成推理

1) 告诉系统任务领域相关的知识
   - 建立 KB
   - 表达当前对象/系统/任务的真实状态

2) 向系统提问：这个新状态为真/假？
   - 我们希望系统回答具备以下特性
     - 合理性：根据语义给出正确的回答
     - 完整性：当答案存在时一定可以找到正确的回答

例子

1) 告诉系统任务领域相关的知识

$$KB = \begin{cases} p \leftarrow q. \\ q. \\ r \leftarrow s. \end{cases}$$

2) 向系统提问：这个新状态为真/假？

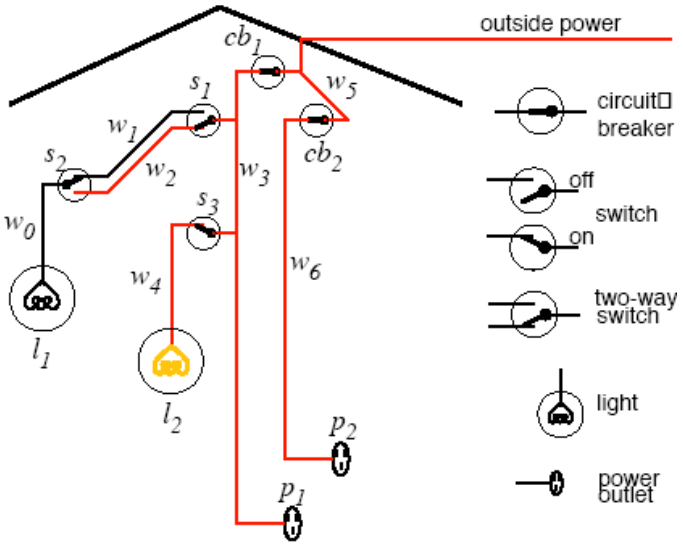
p?
r?
s?

更具体些

1) 告诉系统任务领域相关的知识

light_l1.
light_l2.
ok_l1.
ok_l2.
ok_cb1.
ok_cb2.
live_outsic

live_l1 ←live_w0.
live_w0 ←live_w1 ∧up_s2.
live_w0 ←live_w2 ∧down_s2.
live_w1 ←live_w3 ∧up_s1.
live_w2 ←live_w3 ∧down_s1.
live_l2 ←live_w4.
live_w4 ←live_w3 ∧up_s3.
live_p1 ←live_w3.
live_w3 ←live_w5 ∧ok_cb1.
live_p2 ←live_w6.
live_w6 ←live_w5 ∧ok_cb2.
live_w5 ←live_outside.
lit_l1 ←light_l1 ∧live_l1 ∧ok_l1.
lit_l2 ←light_l2 ∧live_l2 ∧ok_l2.



2) 向系统提问：这个新状态为真/假？

live_w4? ***lit_l2?***

命题逻辑-推理

我们需要一个推理过程，来找到当前KB的一个逻辑结论

- 合理性：如何任何从KB中导出的命题都是KB的逻辑结论，则推到过程是合理的
- 完整性：对KB中的每个结论都存在一个验证（推导过程）

推理引擎

命题逻辑-推理

一种简单的机械推导过程

- 枚举出KB所有的I（解释）
- 对于KB中的所有I, 检查其是否是KB的模型
  - ✓例如：对于一个I，检查 KB中的所有子句是否全为真
- 如果g在这些模型中，KB ⊨ g

实现时会有什么问题？

- 如果KB中有n个命题，必须检查2^n次解释的组合

研究推导理论（算法）的目标

- 找到的合理和完全的推导过程，使我们根据KB直接验证逻辑规则，而不用在整个状态空间上搜索

命题逻辑-推理

两种推导（演绎）形式

- 自底向上的推导（正向链接）
- 自顶向下的查询（搜索）

推理引擎

自底向上的推导

总体思路基于推导规则

If “$h \leftarrow b_1 \wedge \ldots \wedge b_m$” 是KB中的子句,
and

每个 b_i 都成立,

then 可以推导出h.

- 这个规则也包含了m=0时的情况

自底向上的推导

机械推导过程，输入KB

初始化原子集合C:={};

repeat

选择 KB中的子句“$h \leftarrow b_1 \wedge \ldots \wedge b_m$”，

对所有b_i， $b_i \in C, h \notin C$;

$C := C \cup \{h\}$

until 没有其它确定子句可以选择.

return C

自底向上的推导

初始化原子集合C:={};

repeat

选择KB中的子句“$h \leftarrow b_1 \wedge \ldots \wedge b_m$”，其中 对所有$i$， $b_i \in C, h \notin C$;

$C := C \cup \{h\}$

until 没有其它确定子句可以选择.

KB

- $a \leftarrow b \wedge c$
- $a \leftarrow e \wedge f$
- $b \leftarrow f \wedge k$
- $c \leftarrow e$
- $d \leftarrow k$
- e.
- $f \leftarrow j \wedge e$
- $f \leftarrow c$
- $j \leftarrow c$

C

- {}
- {e}
- {c,e}
- {c,e,f}
- {c,e,f,j}
- {c,e,f,j,a}

- 结束得到KB ⊨ **a**, KB ⊨ ***j…***

自顶向下的推导

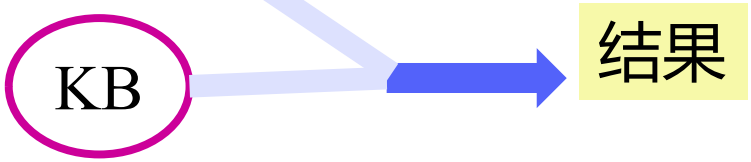
- **基本思想:** 从输入子句*g*开始反向搜索，确定其是否在KB中成立

自底向上

- 如果 $g \subseteq C$， *g*成立
- 何时算法得到*g*?
  - 算法结束

自顶向下

查询 *g*（单个子句或集合）



- 选择*g*中的一个原子，反向查询

自顶向下的推导

机械推导过程，输入KB，待证明原子集合g

初始化原子集合g :=Query;

repeat

select g中的一个原子a;

choose KB中以a为头部的确定子句“$a \leftarrow \boldsymbol{B}$ ”

将g中的a替换为$\boldsymbol{B}$

until g={}

return *yes*

自顶向下的推导-例子

初始化原子集合g :=Query;
repeat
　　select g中的一个原子a;
　　choose KB中以a为头部的确定子句“$a \leftarrow B$”
　　将g中的a替换为B
until g={}
return *yes*

$a \leftarrow e \wedge f.$
$c \leftarrow e.$
$f \leftarrow j \wedge e.$

$a \leftarrow b \wedge c.$
$d \leftarrow k.$
$f \leftarrow c.$

$b \leftarrow k \wedge f.$
$e.$
$j \leftarrow c.$

KB

查询: a

1）$yes \leftarrow a$
2）$yes \leftarrow e \wedge f$
3）$yes \leftarrow f$
4）$yes \leftarrow c$
5）$yes \leftarrow e$

1）$g=\{e,f\}$
2）$g=\{f,e\}$
3）$g=\{c,e\}$
4）$g=\{e\}$
5）$g=\{\}$

限制

- 命题逻辑采用**可能世界语义**来反映客观世界的不确定性，并以此来确定不同情境下的状态（真值），以完成推导.

可能世界

可能世界的概念由莱布尼茨提出:一个世界如果与逻辑定律不矛盾,这个世界是可能的.世界不只有一个,除了现实世界之外,一个公式成立与否，取决于在哪个世界中对它进行考察.

专家系统—工具

Datalog

CLIPS

Prolog

JESS

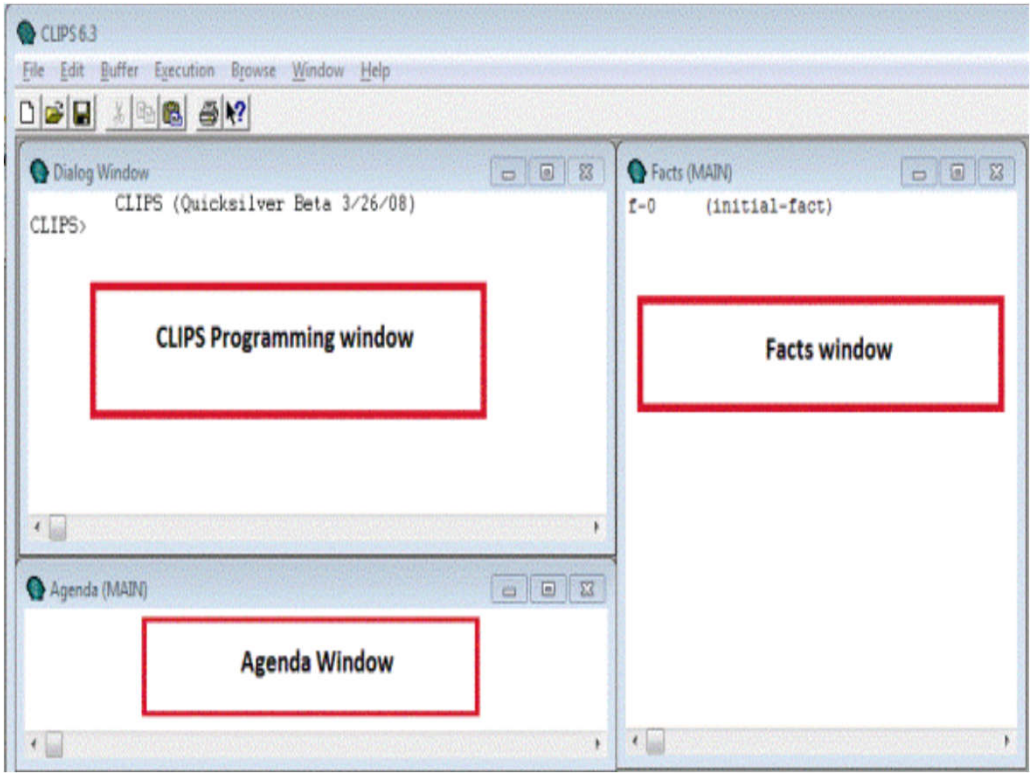
Drools

DTRules

OpenL Tablets

Java Rules Engine API

专家系统—CLIPS



- (defrule start-up
  (name condition)
  =>
  (printout t "Hello Clips!" crlf))

- (assert (name condition))

NASA开发　　持续的支持　　第三方集成

专家系

领域专家

概念抽象

知识工程师

编程

File Edit Buffer Execution Browse Window Help

Dialog Window

```
_SUMMARY Topic?
defrule start-up (name condition)

ut t "Hello clips,thanks for help!" crlf))
run)
assert (name condition))
```

Agenda (MAIN)

```
0      start-up: f-1
```

Facts (MAIN)

```
f-0     (initial-fact)
f-1     (name condition)
```

演绎搜索　　冲突解决

推理引擎

模式匹配　　动作执行

查询

咨询用户

规则&事实

知识库

工作内存

专家系统

专家系统—实例

实例

End