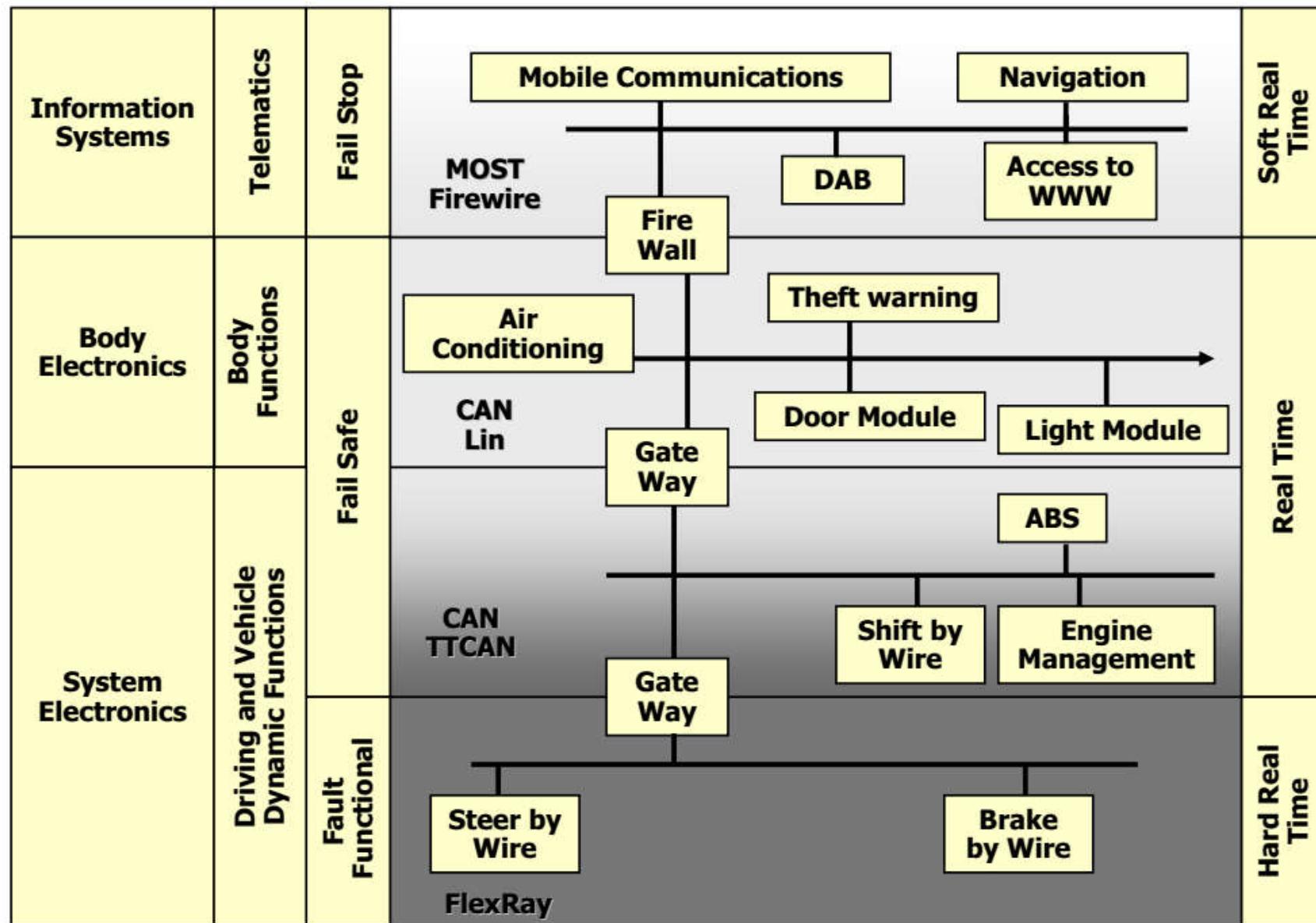




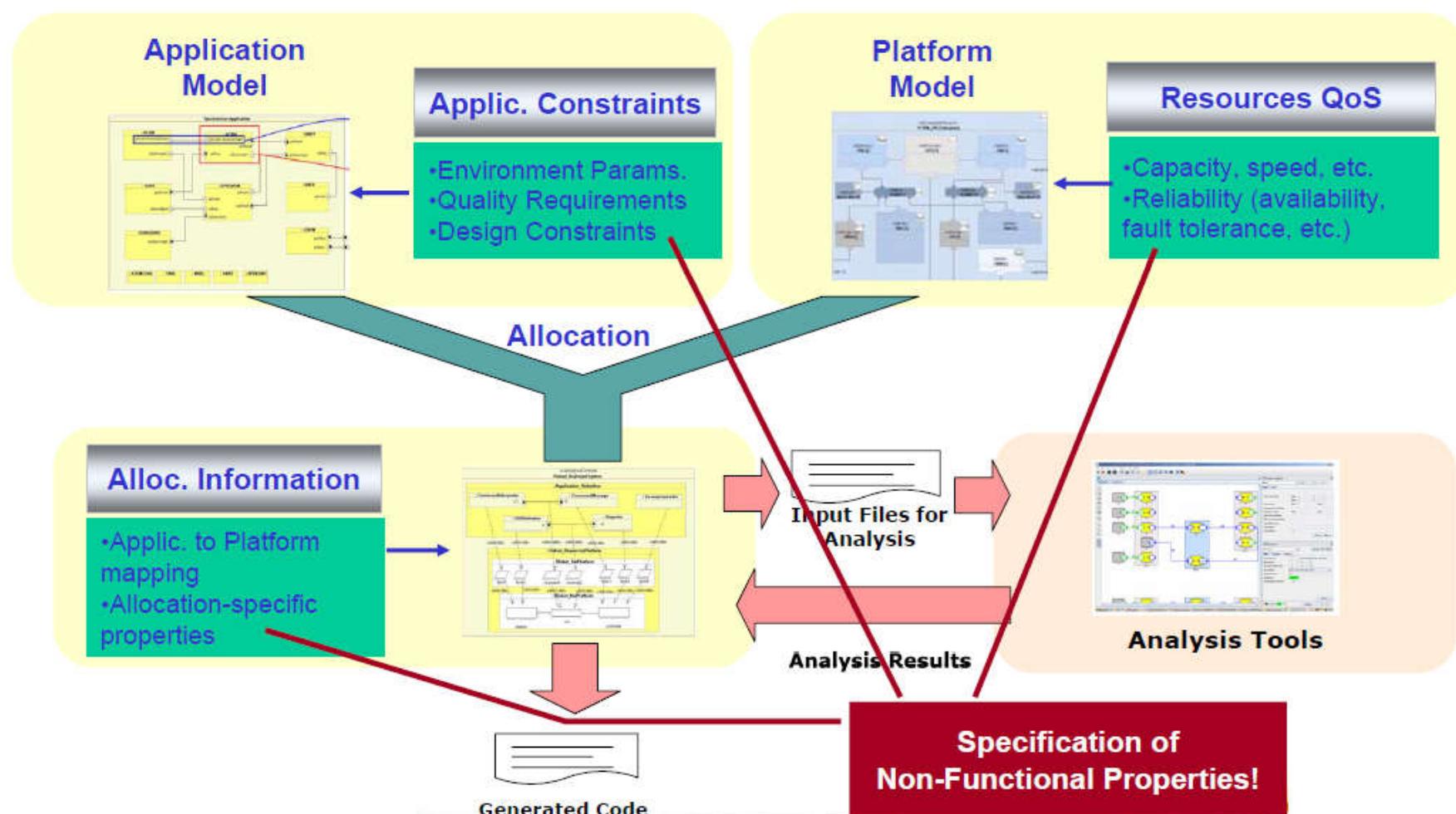
RTE规范与建模方法之 Domain Specific Modeling Languages

Distributed Car Systems Arch.



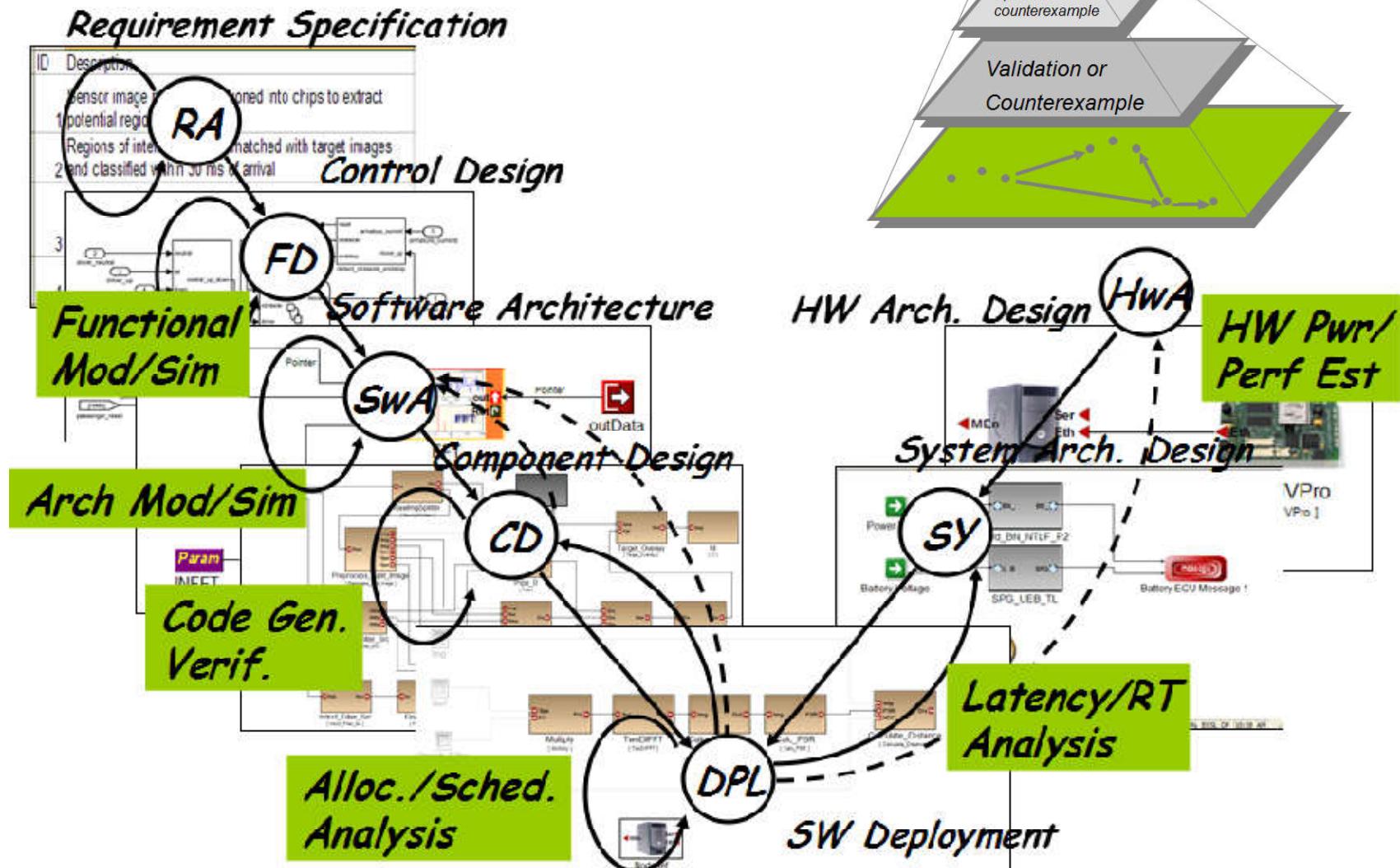
“Y-Chart” Approach for Model-Based Analysis

- Applications, Platform, Allocation



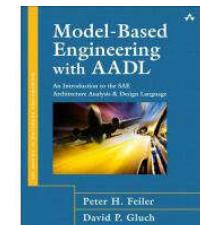
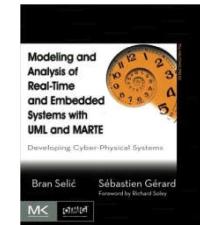
RTEs系统设计过程: top-down

V & V, Analysis! ! !



内容提要

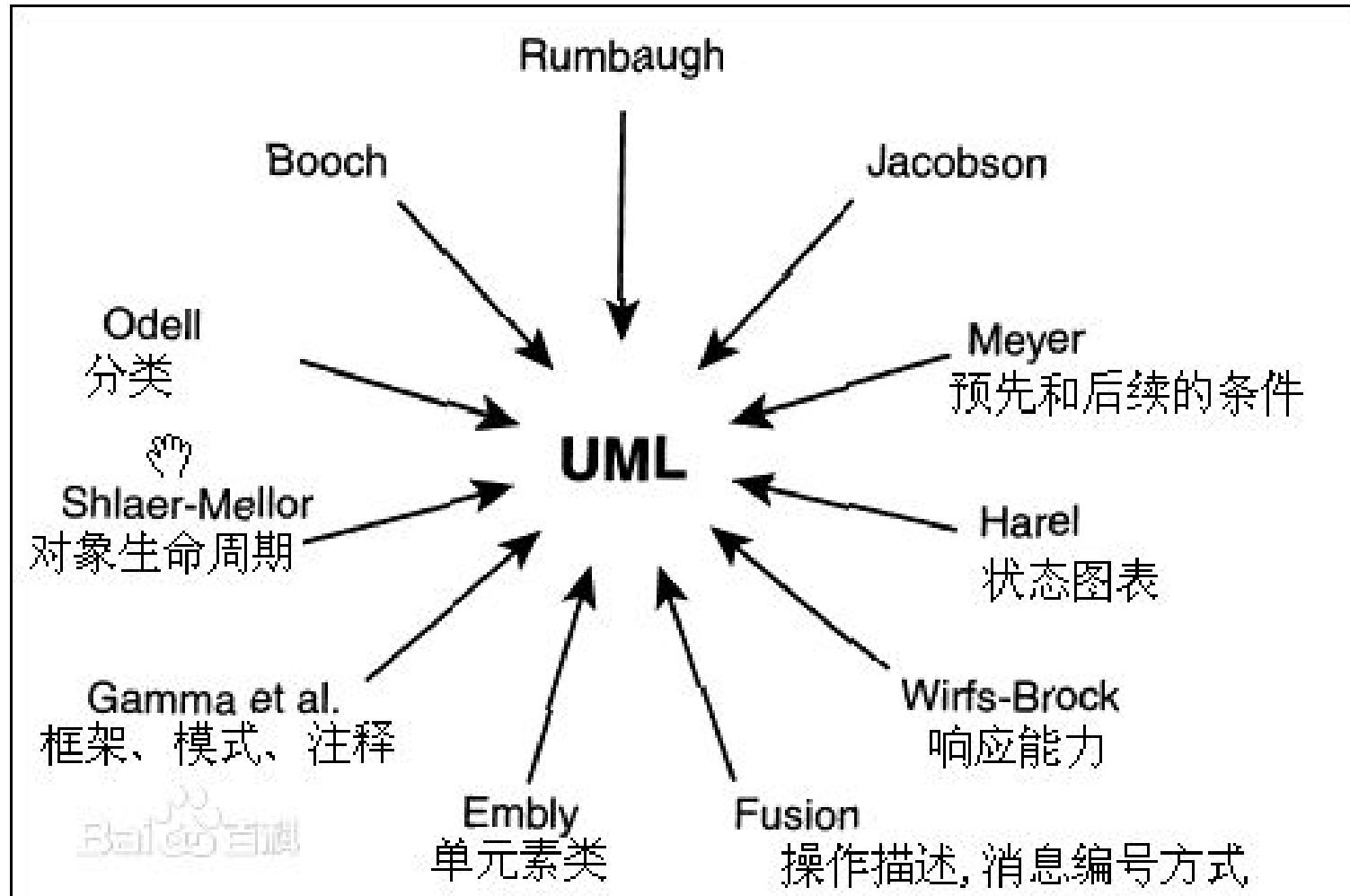
- RTE系统体系结构建模、验证、分析
 - Domain Specific Modeling Languages
 - MARTE@inria, 工具Papyrus+MAST, 2008 OMG标准
 - Modeling and Analysis of RT and Embedded systems
 - UML: Specification languages
 - AADL@SEI, OSATE+Cheddar, 2004 SAE标准AS5506
 - Architectural Analysis and Design Language
 - Autosar方法: 汽车电子 (ECU)
- Peter Marwedel, TU Dortmund教授
 - 《嵌入式系统设计·嵌入式CPS系统基础》, 第2版2011
 - 第2.10节: UML
- Bran Selic, 资深工程师@ibm.ca
 - Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE: Developing **Cyber-Physical Systems**, 2014
- Peter H. Feiler, CMU教授
 - Model-Based Engineering with AADL, 2012



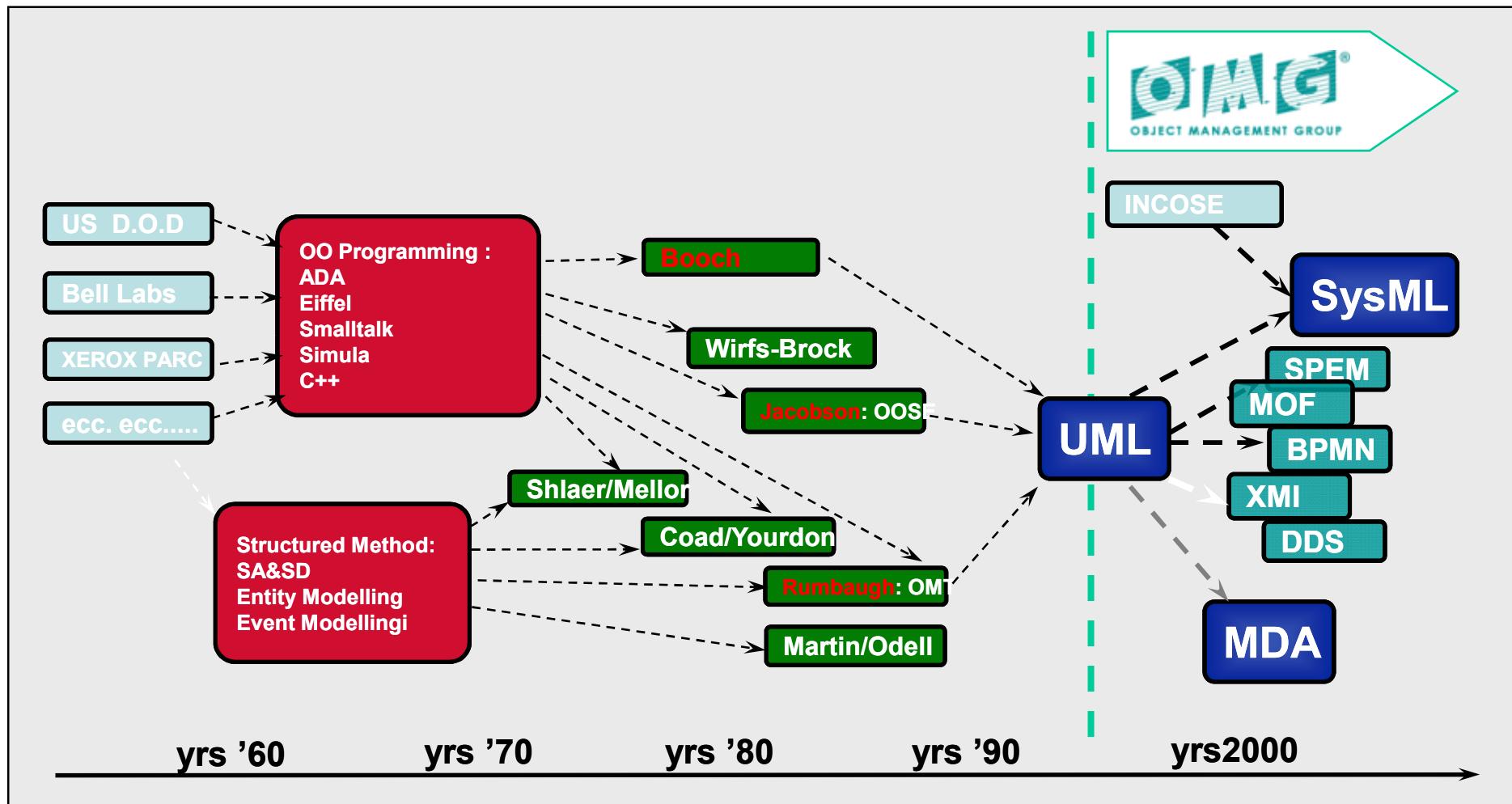
UML (Unified Modeling Language)

- 统一建模语言是**可视化的设计说明语言**
 - 统一了Booch、Rumbaugh和Jacobson的表示方法
 - Booch: 描述对象集合和它们之间关系的方法
 - Rumbaugh: 对象建模技术(OMT)
 - Jacobson: 用例方法
 - 统一描述系统的硬件和软件，对系统的功能建模
 - 可自动产生实际设计的HDL或C++代码
- I-Logix公司的Rhapsody系列产品
 - 用于建立软件系统模型，也可以描述非软件系统
 - 机械系统,企业机构,业务过程,信息系统
 - NASA火星探路者:运用Rhapsody在VxWorks上开发
 - 通过表达系统的需求使软件系统的模型文档化

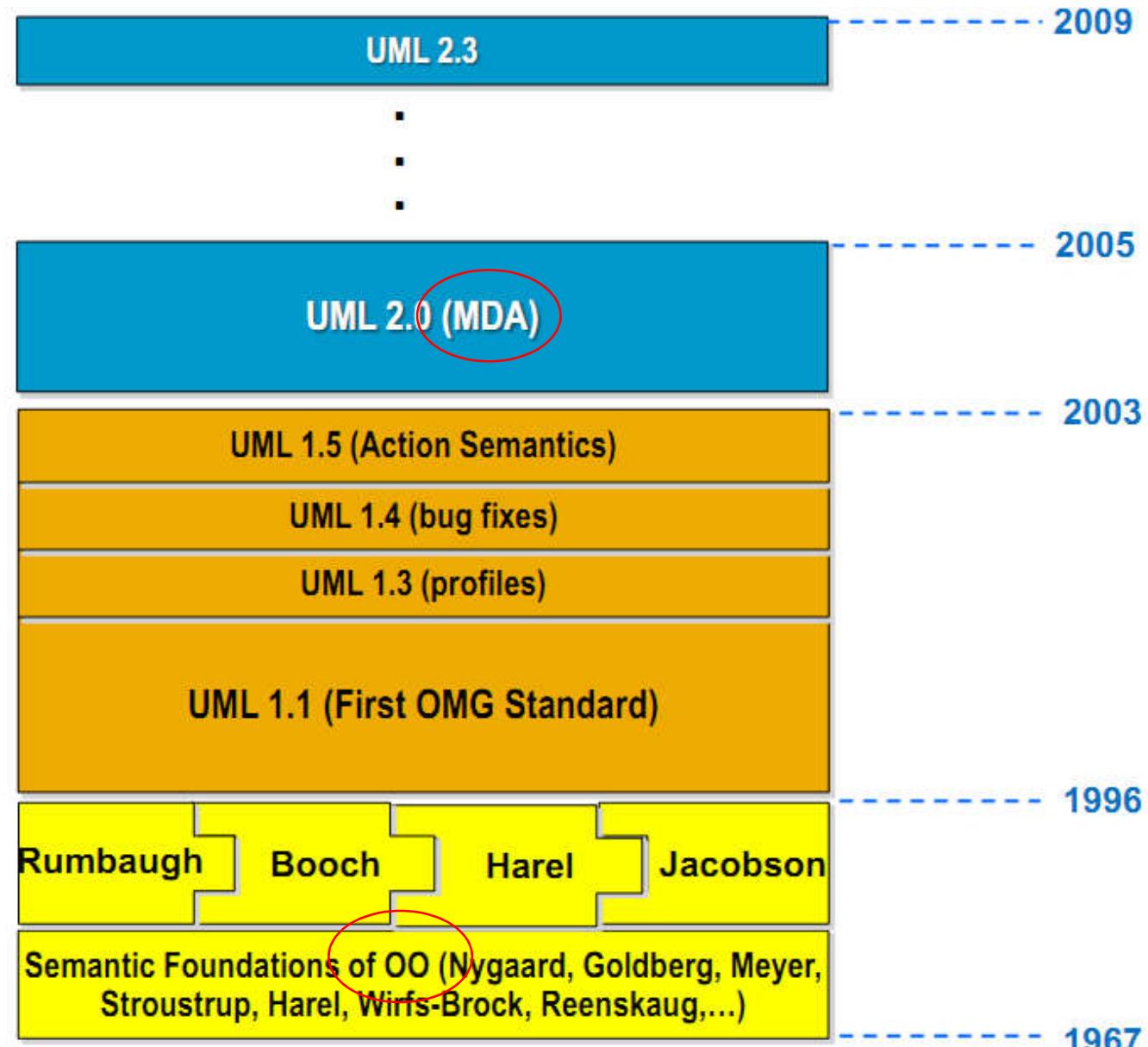
UML方法



A Historical Perspective



UML Roots and Evolution: UML2



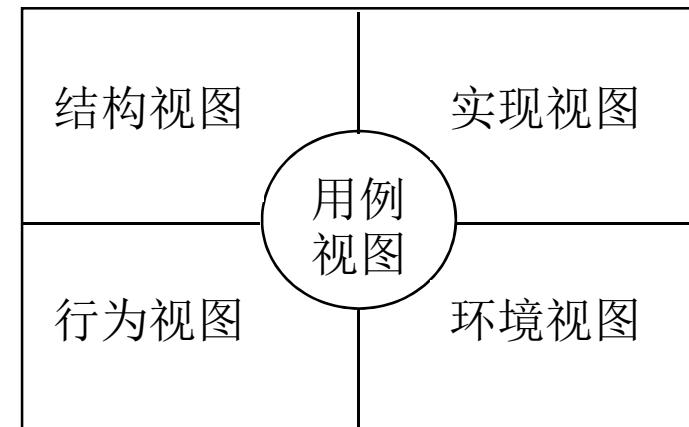
UML的概念模型

- UML的视图

- 用例视图—表示系统的功能和场景（用例图）
- 结构视图—表示系统的静态或空闲的状态（类图/对象图）
- 行为视图—表示系统状态的变化（顺序/协作/状态/活动图）
- 实现视图—表示系统的逻辑元素的分布（组件图）
- 环境视图—表示系统的物理元素的分布（部署图）

- 对系统的功能建模

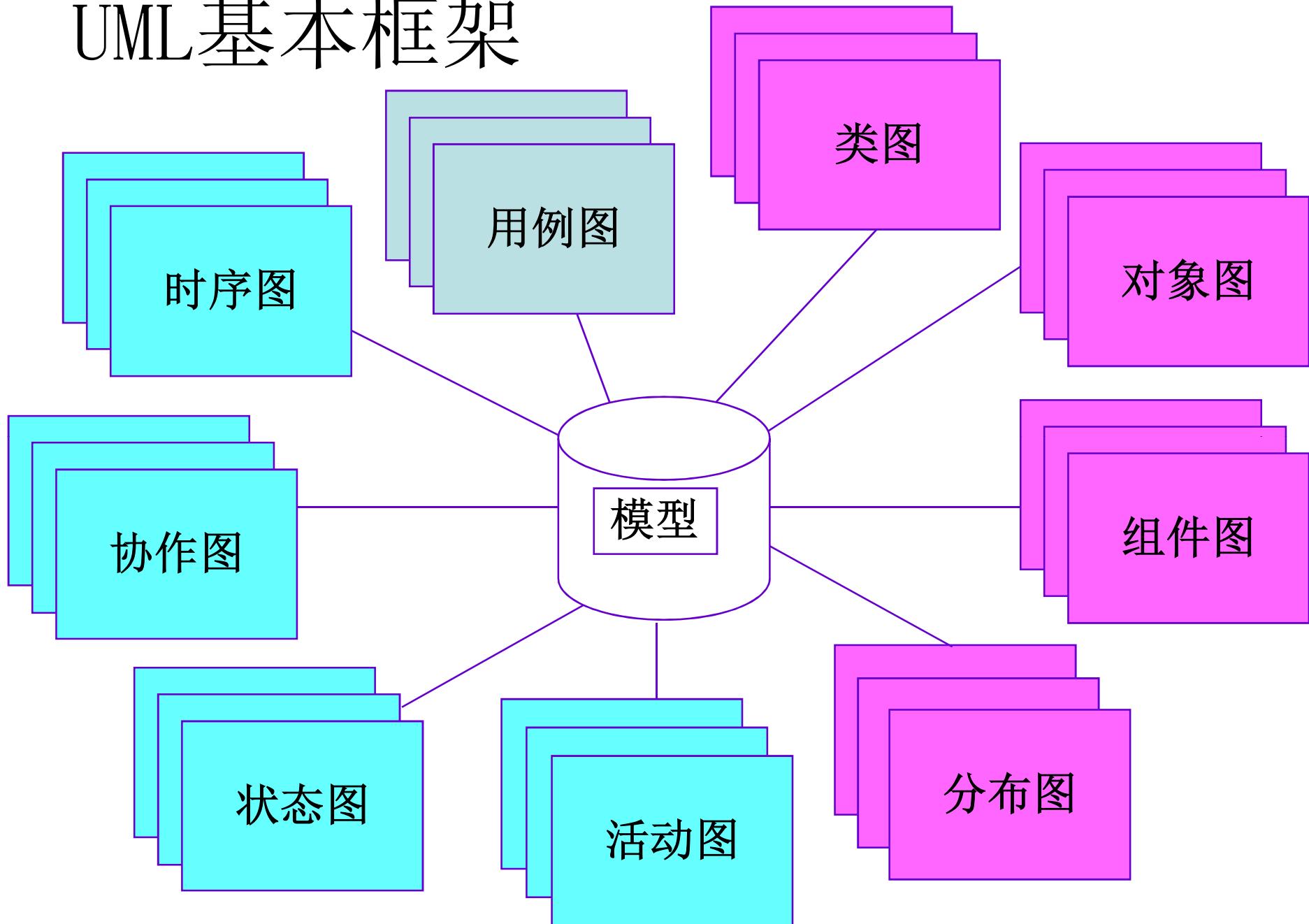
- 用例模型：解释用户的要求
- **类模型：描述其静态结构**
- **交互模型：描述消息流**
- **状态模型：表示对象的动态行为**
- 实现模型：工作单元分布
- 部署模型：进程分配。应用与环境（OS+HW）绑定



UML建模语言：OOD， CBD

- UML架构：图+元模型
 - 图——UML的语法，元模型——语义
 - 元元模型（定义全部事物）、元模型、UML模型、用户模型
- 事物：模型中最具有代表性的成分的抽象
 - 结构事物
 - 类（Class）、接口（Interface）、协作（Collaboration）、用例（UseCase）、主动类（ActiveClass）、组件（Component））、节点（Node）
 - 行为事物
 - 交互（Interaction）、状态机（State machine）
 - 分组事物（包， Package）
 - 注释事物（注解， Note）
- 关系：事务的结合
 - 依赖、关联、泛化和实现

UML基本框架

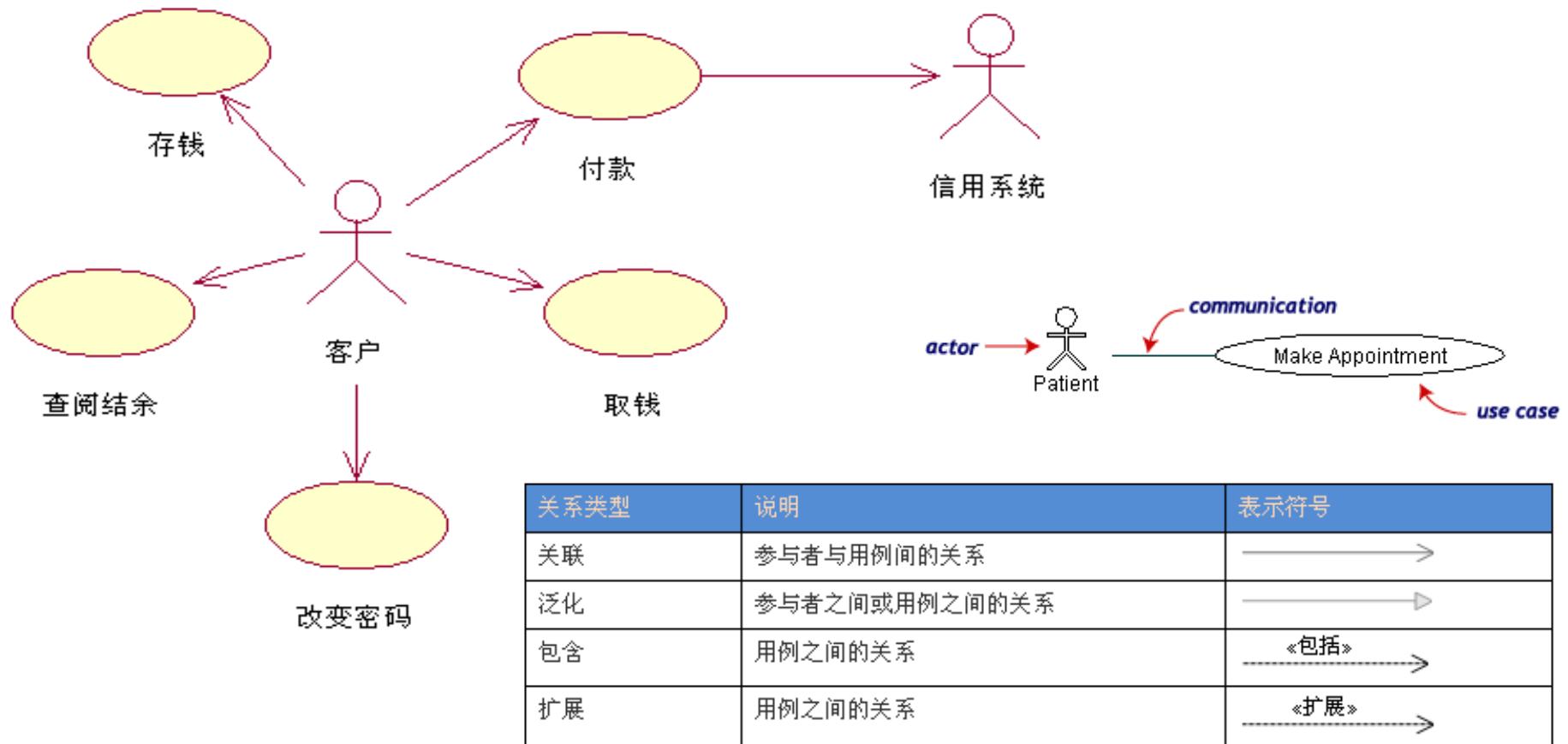


UML的9个核心框图

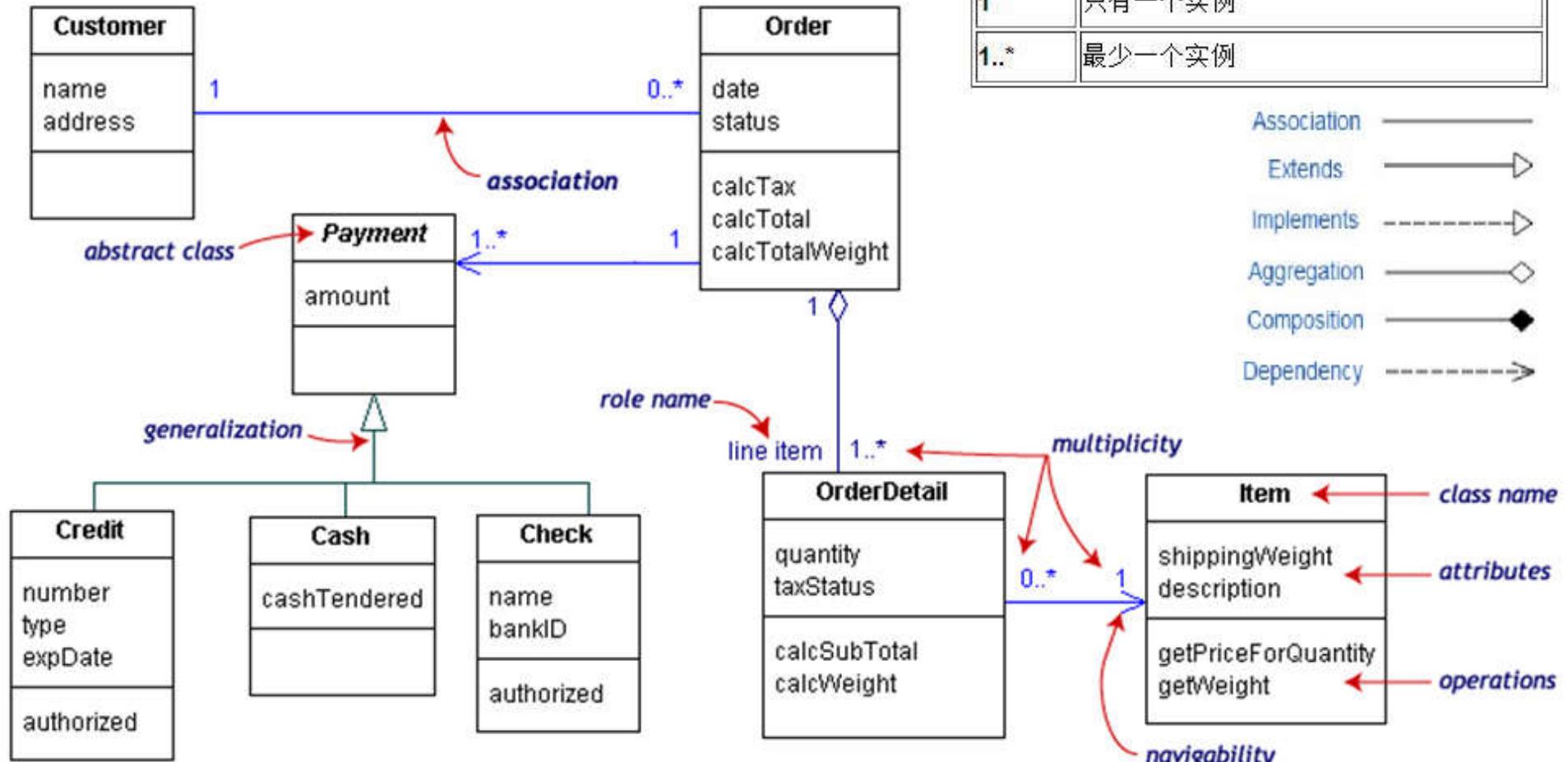
- 功能
 - 用例图：描述一个系统做什么（功能），而不说明怎么做
- 结构
 - 类图：类与类之间的交互
 - 对象图：一组对象（类的实例）以及它们之间传送的消息
- 行为
 - 顺序图（时序图）：按时间顺序对控制流建模
 - 协作图：按对象的组织对控制流建模（交互？）
 - 状态图：显示一个对象的生命期中，响应事件的状态序列
 - 活动图：描述了类的活动，一种特殊的状态图
 - 被内部进程或实体访问时描述了类的行为
- 实现
 - 组件图（构件图）：一组组件之间的组织和依赖关系
 - 部署图（实施图、分布图）：描述物理组件及其分布和关联

用例图（系统功能/服务）

- 要素: Actor, Use Case (功能), 关系 (通信)
 - use case被映射成系统中可运行的线程 (称任务)
 - 类图中的各个类对象, 按照顺序图交互, 完成一个用例

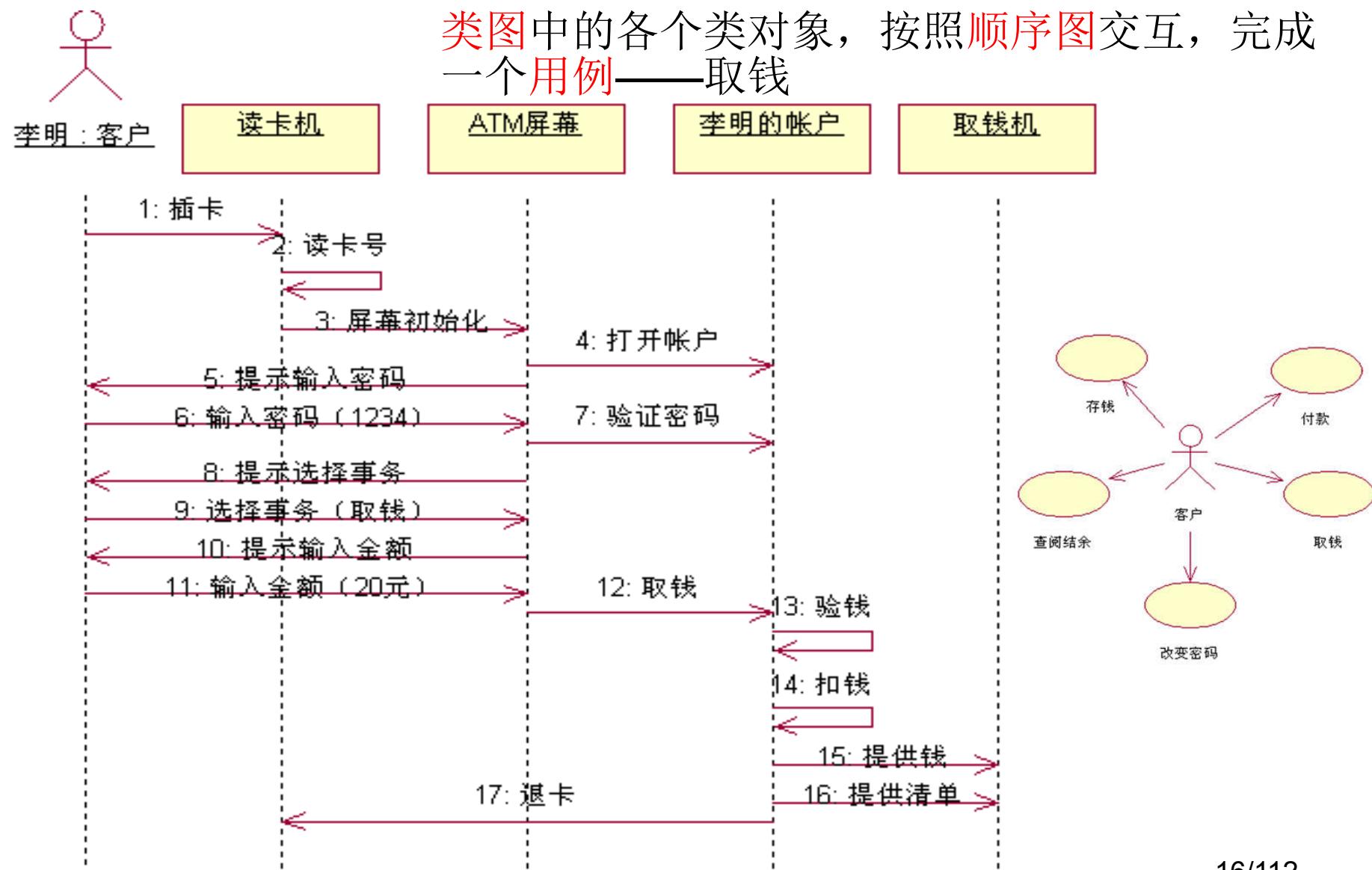


类图及其关系



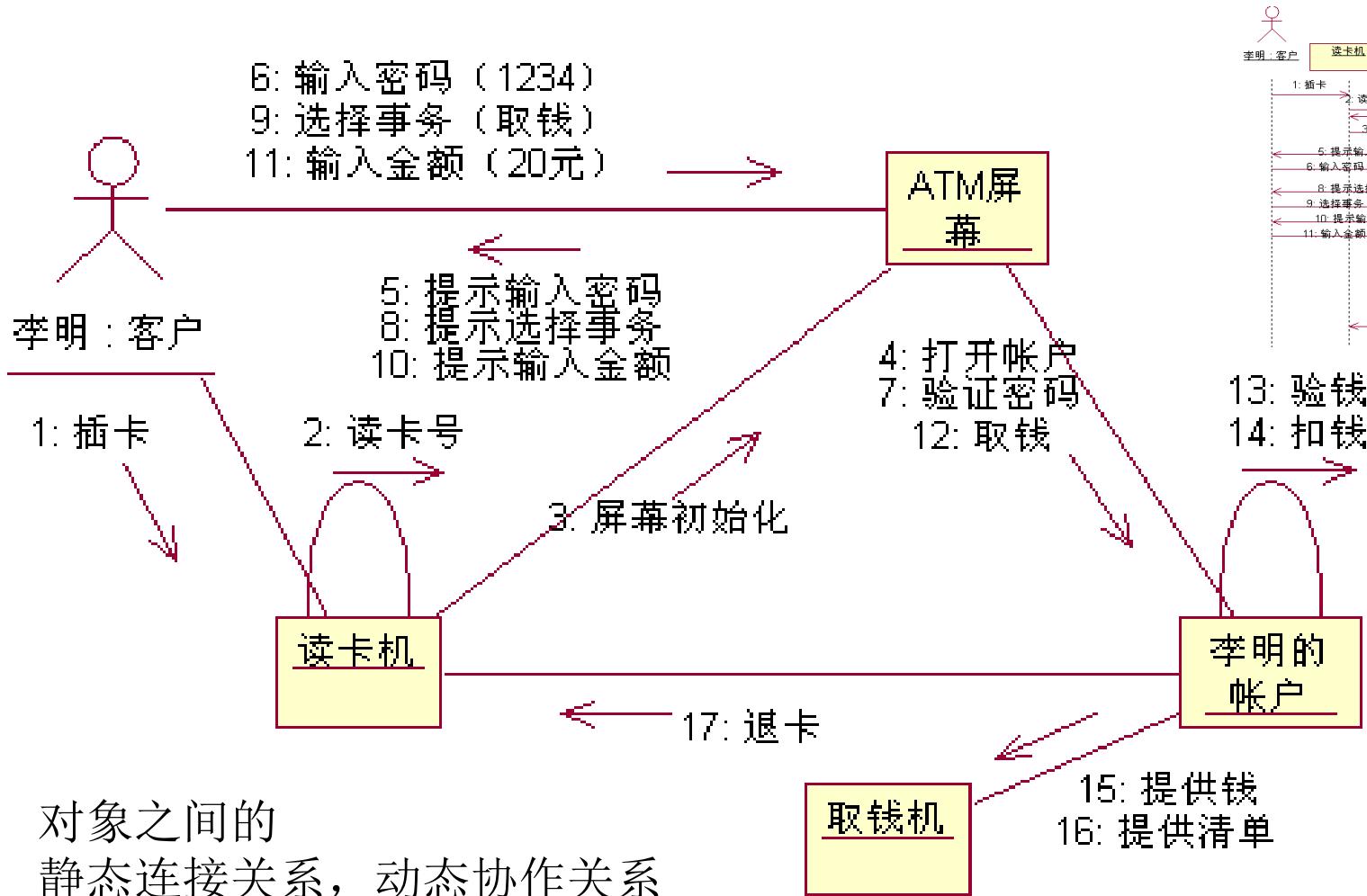
- 具体->抽象
 - 泛化 = 实现 > 扩展 > 组合 > 聚合 > 关联 > 依赖?

顺序图（按时间关系，描述过程）



协作图 (Collaboration Diagrams)

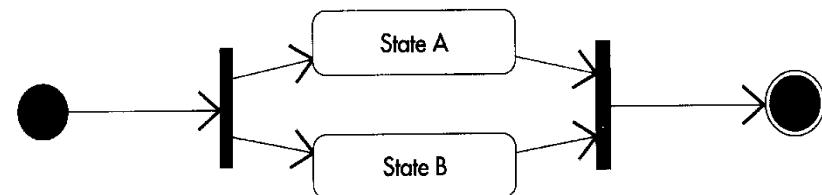
- 箭头表示消息发送方向，消息编号表明消息执行的顺序



对象之间的
静态连接关系，动态协作关系

UML状态图(StateCharts)

- 传统的**FSM**是平面的、顺序的，难以描述复杂的控制系统。**UML StateCharts**在**FSM**的基础上增加了层次、并发和通信机制。——类HCFSM
- 基本元素
 - 状态：对象的存在条件
 - 基本状态
 - 复合状态：表示状态的层次结构
 - 非并发：OR-状态
 - 并发：AND-状态
 - 伪状态：Initial, Join, Fork, Junction, Choice, Final等
 - 状态转换：对象对事件响应而改变状态的方式
 - 事件：发生在时间和空间上的值得注意的事情。
 - 动作：一种原子行为



状态种类	描述	表示法
简单状态	没有子结构的状态	
并发组成状态	被分成两个或多个并发子状态的状态，当组成状态被激活时，所有的子状态均被并发激活	
顺序组成状态	包含一个或多个不连接的子状态的状态，特别是当组成状态被激活时，子状态也被激活	
初始状态	伪状态，仅表明这是进入状态机真实状态的起点	
终止状态	特殊状态，进入此状态表明完成了状态机的状态转换 历程中的所有活动	
结合状态	状态，将两个转换连接成一次就可以完成的转换	
历史状态	伪状态，它的激活保存了组成状态中先前被激活的状态	
子机器引用状态	引用子机器的状态，该子机器被隐式地插入子机器引用状态的位置	
桩状态	伪状态，用来在子机器引用状态中标识状态	

简单状态

- 当对象探测到一个事件后，依当前的状态做出反应
- 反应：包括执行一个动作和转换到新状态



事件

- 在时间上的一点发生，没有持续时间
- 调用事件（同步）、改变事件（状态）、信号事件（异步通信）、时间事件等

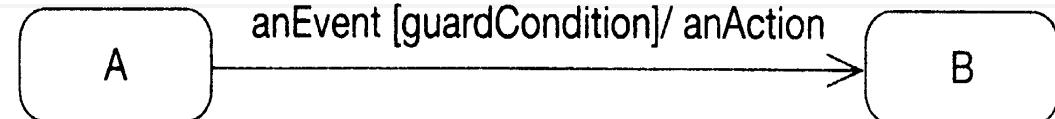
事件类型	描述	语法
调用事件	接受等待应答的对象的明确形式的同步请求	<code>op (a:T)</code>
改变事件	对布尔表达式值的修改	<code>When (exp)</code>
信号事件	接受一个对象间外在的、命名的、异步的通信	<code>Sname (a:T)</code>
时间事件	绝对时间的到达或者相对时间段的终结	<code>After (time)</code>



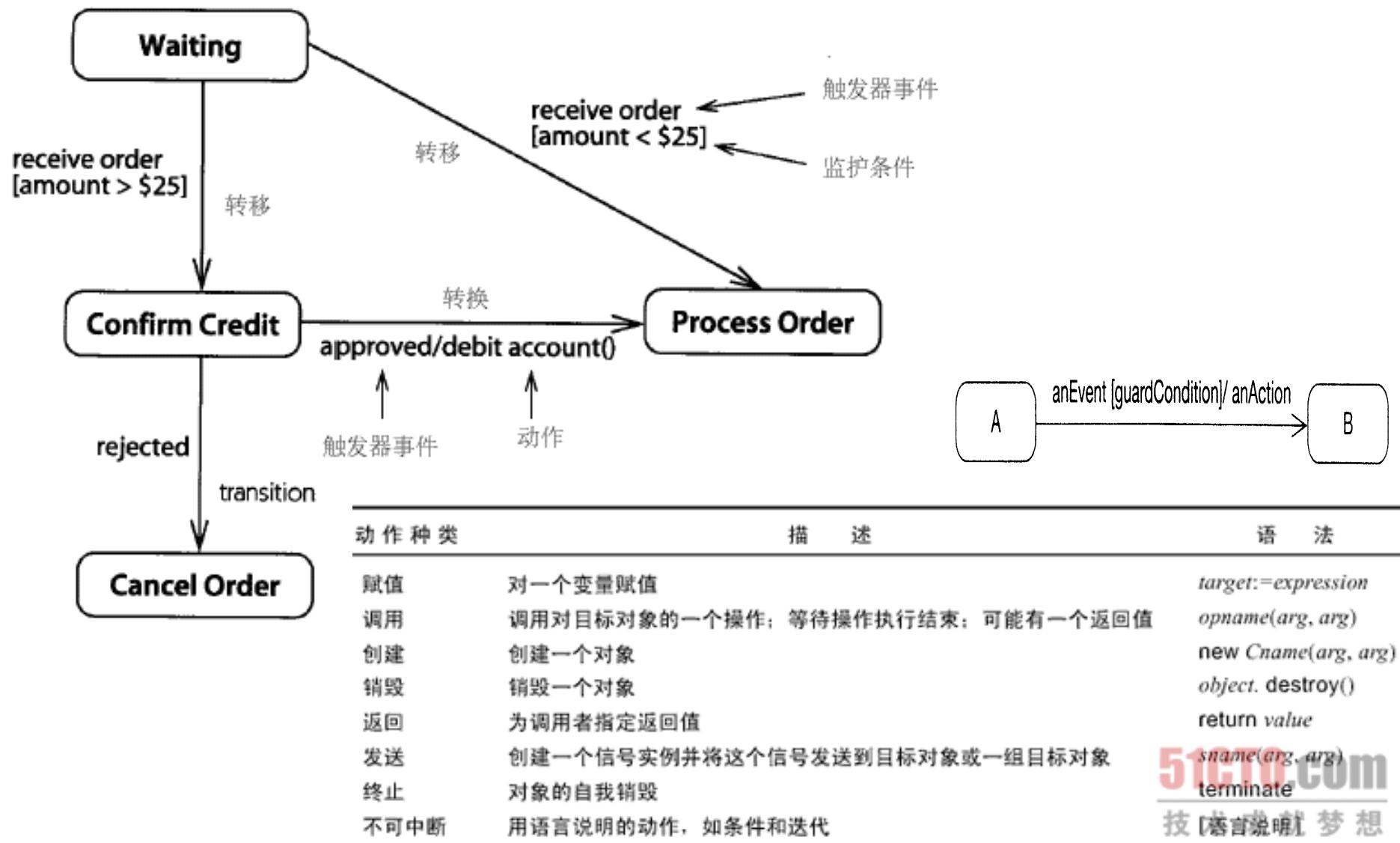
状态转换

- 从状态出发的转换定义了处于此状态的**对象**对外界发生的事件所做出的反应
- 定义一个转换要有引起转换的触发器事件、监护条件、转换的动作和转换的目标状态

转换的种类	描述	语法
入口动作	进入某一状态时执行的动作	<code>entry[/action]</code>
出口动作	离开某一状态时执行的动作	<code>exit[/action]</code>
外部转换	引起状态转换或自身转换，同时执行一个具体的动作，包括引起入口动作和出口动作被执行的转换	<code>e(a:T)[exp]/action</code>
内部转换	引起一个动作的执行但不引起状态的改变或不引起入口动作或出口动作的执行	<code>e(a:T)[exp]/action</code>

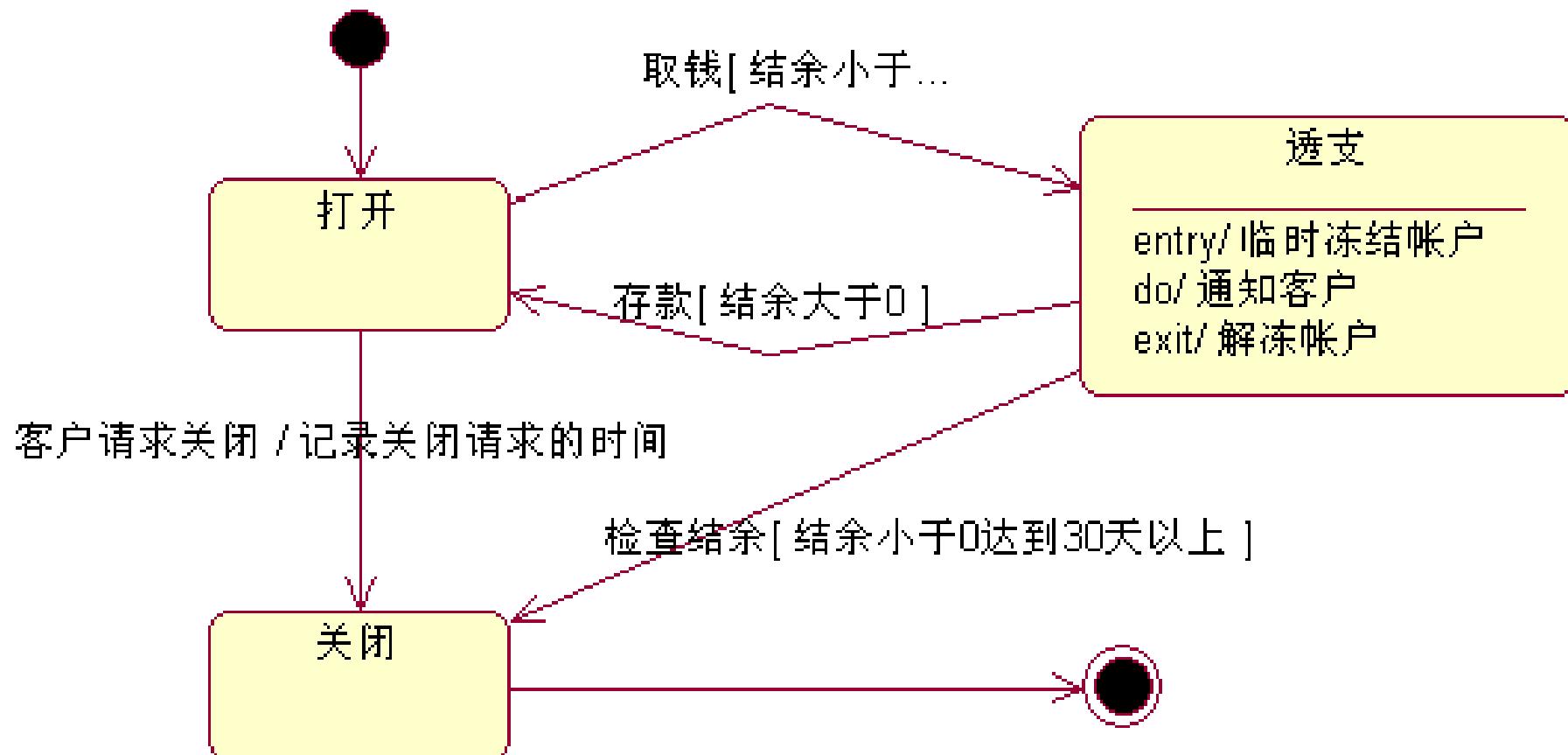


监护条件，动作

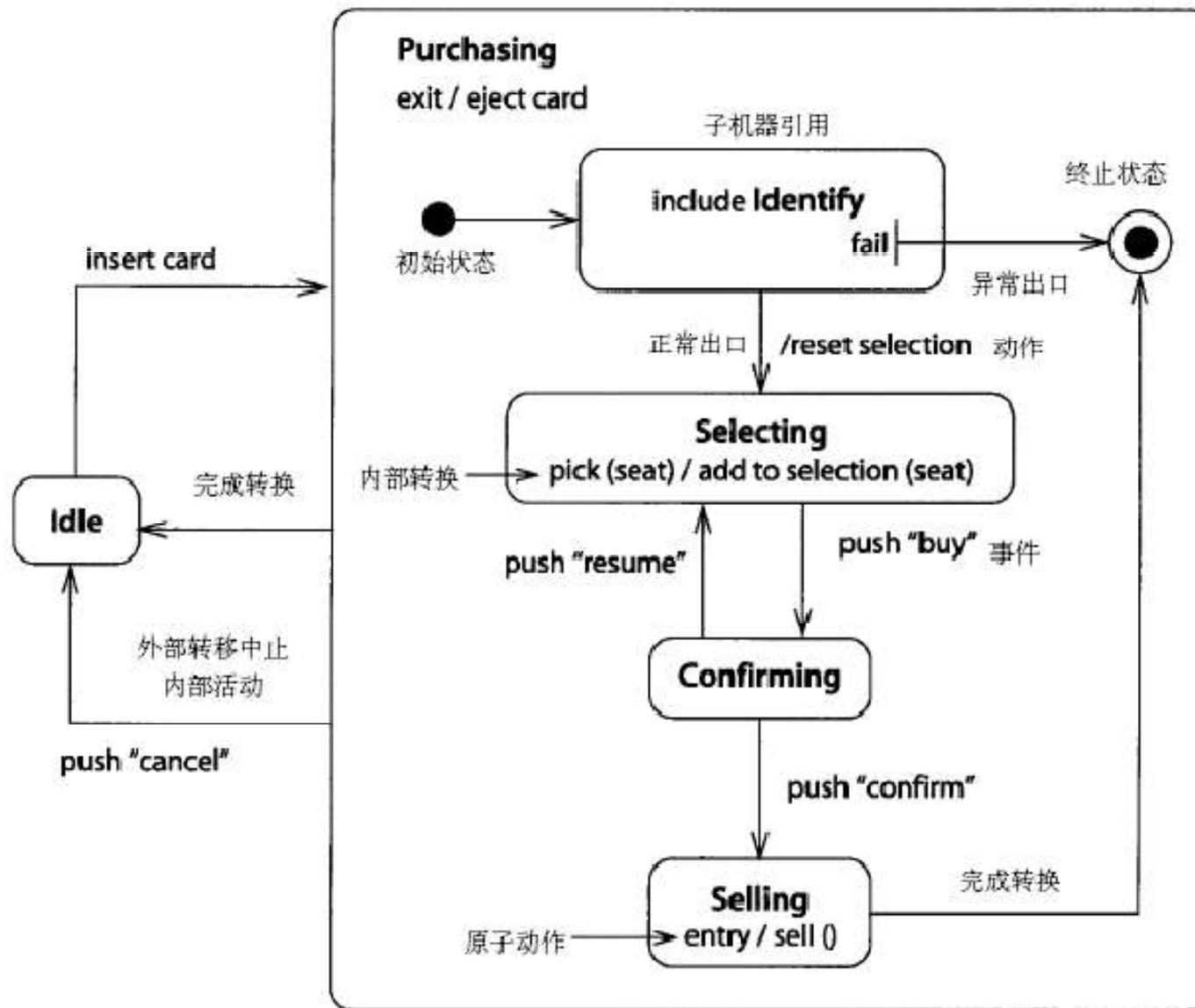


入口动作、出口动作 (Account对象)

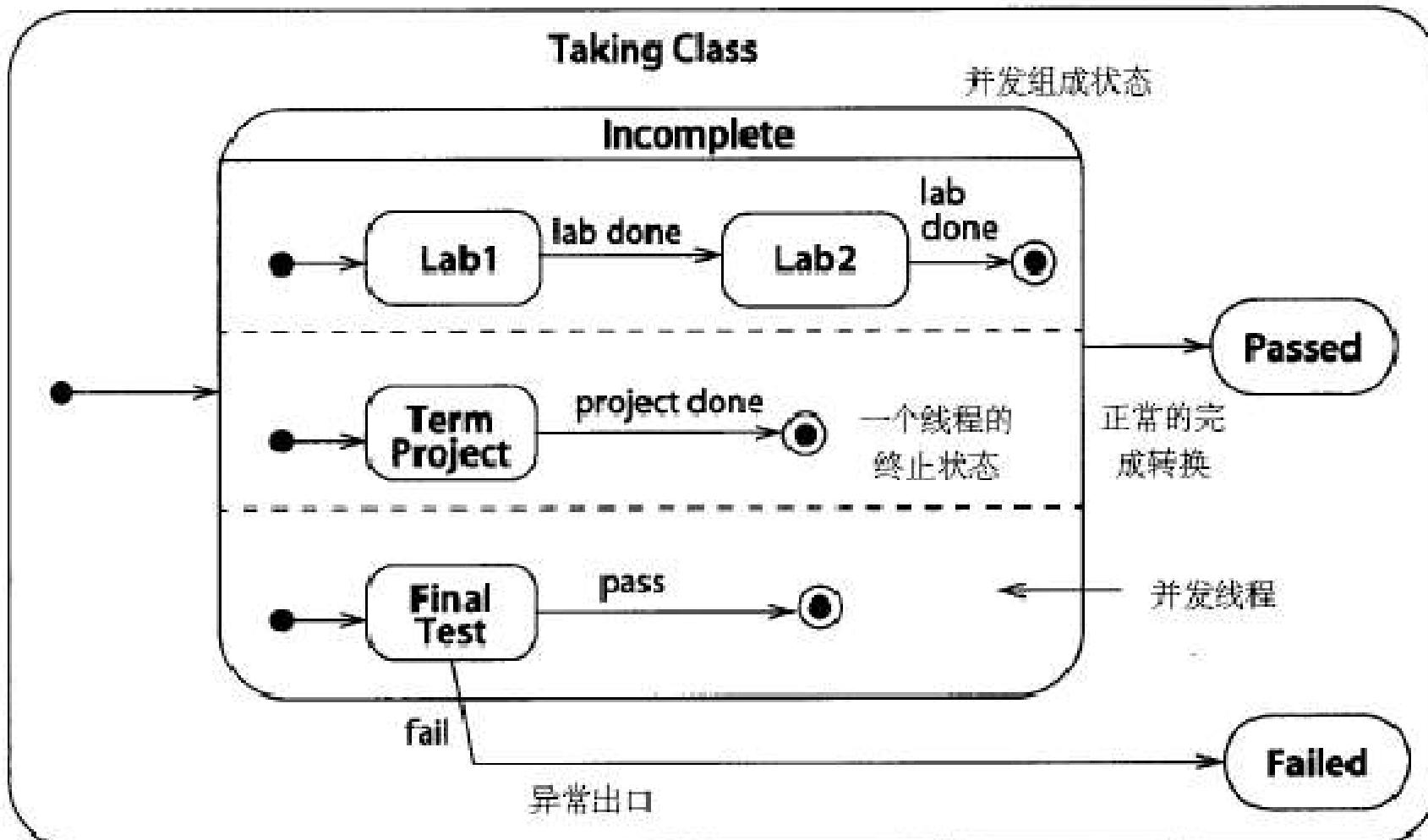
- entry: 入口动作; exit: 出口动作



顺序组成状态

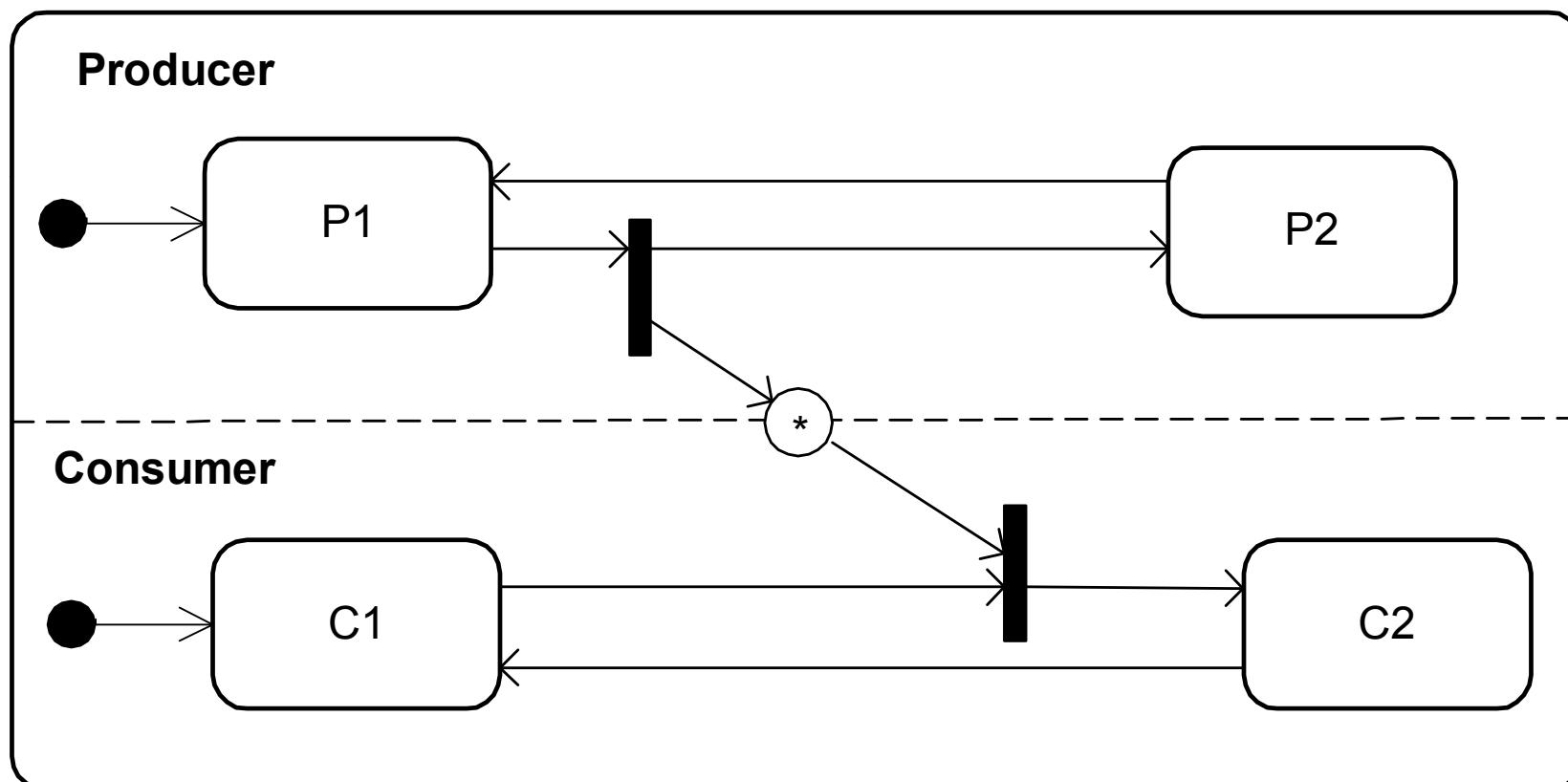


并发组成状态

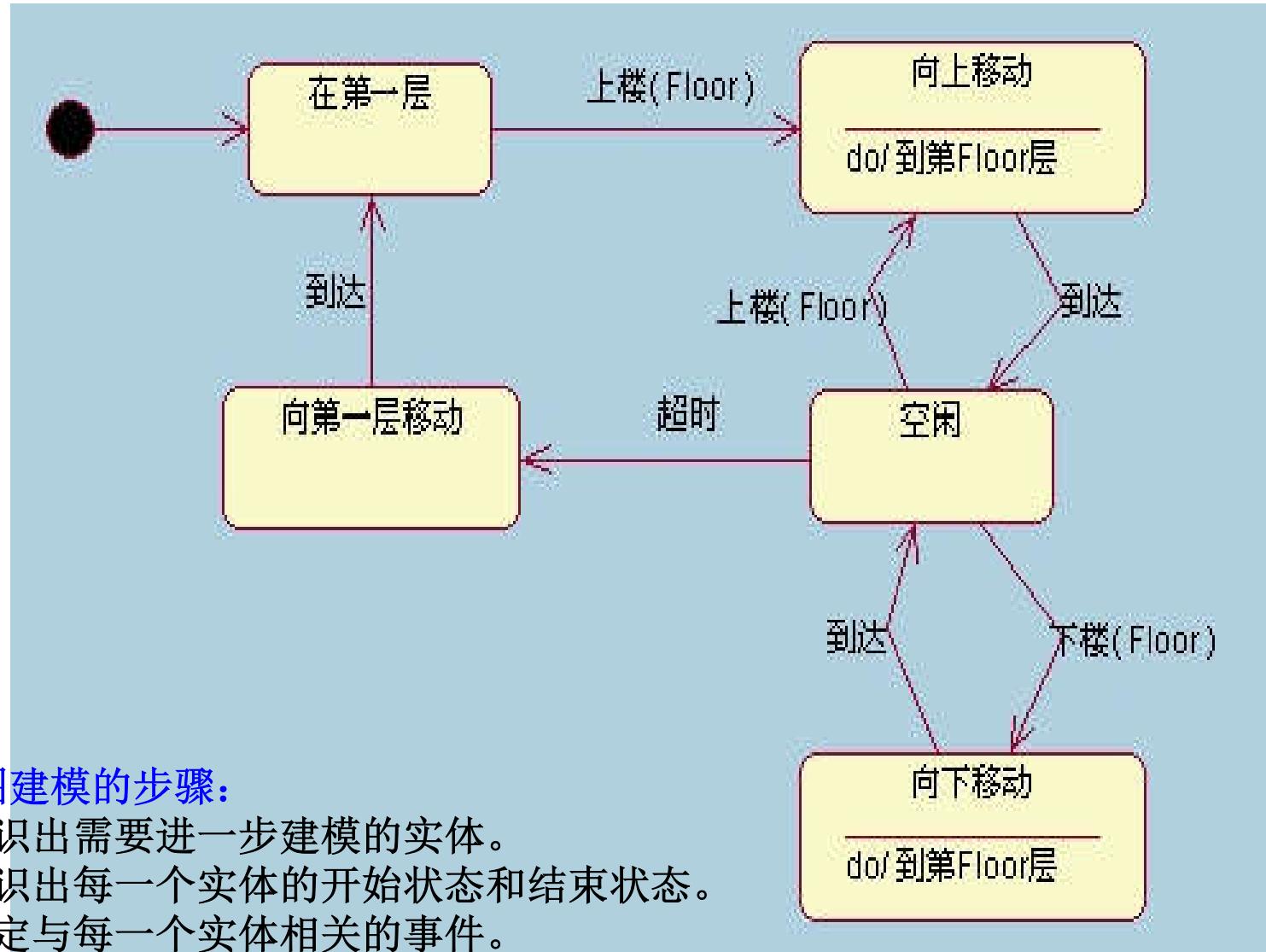


Sync State

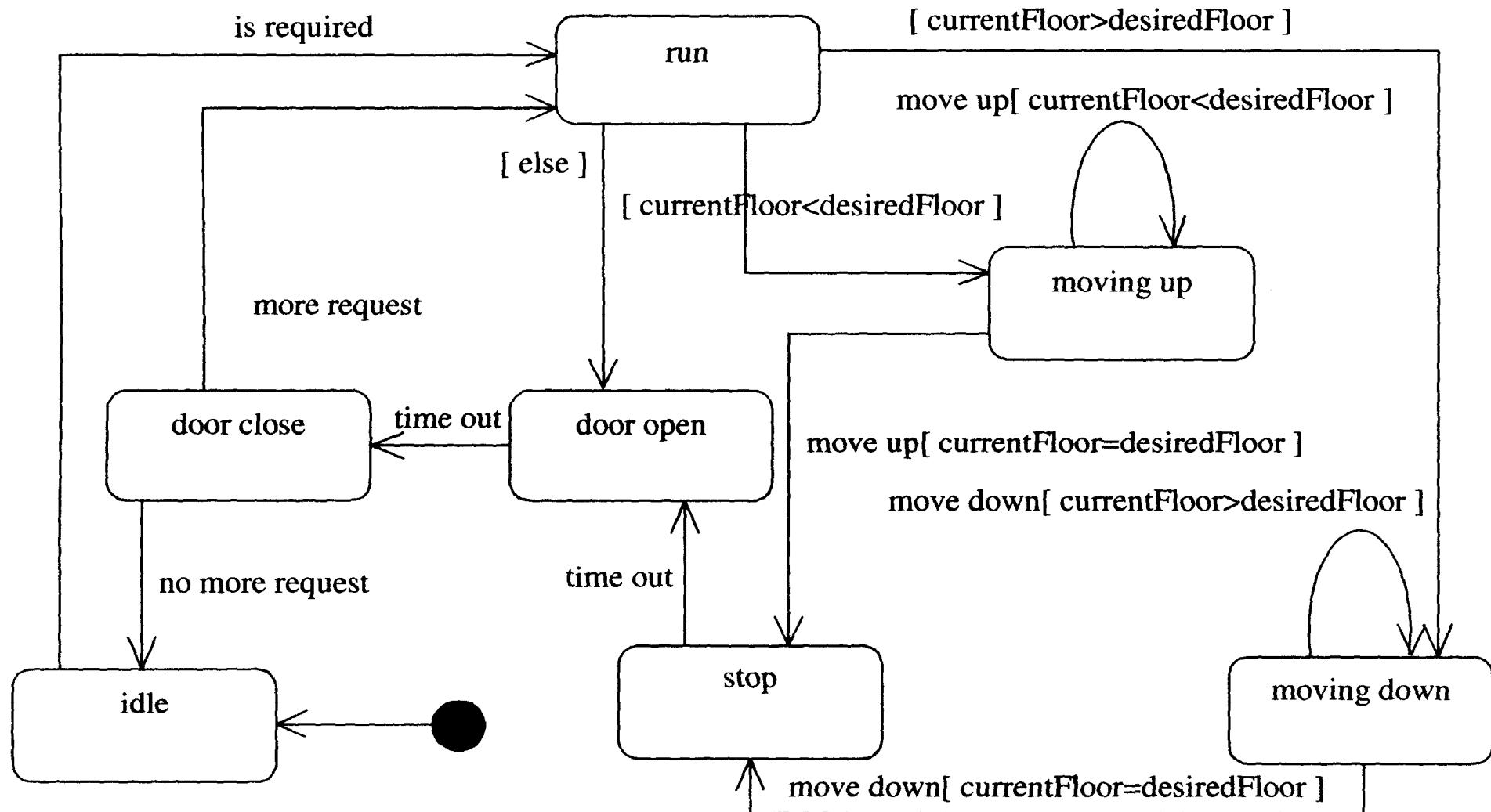
- **Vertex** used for synchronizing the concurrent regions of a state machine



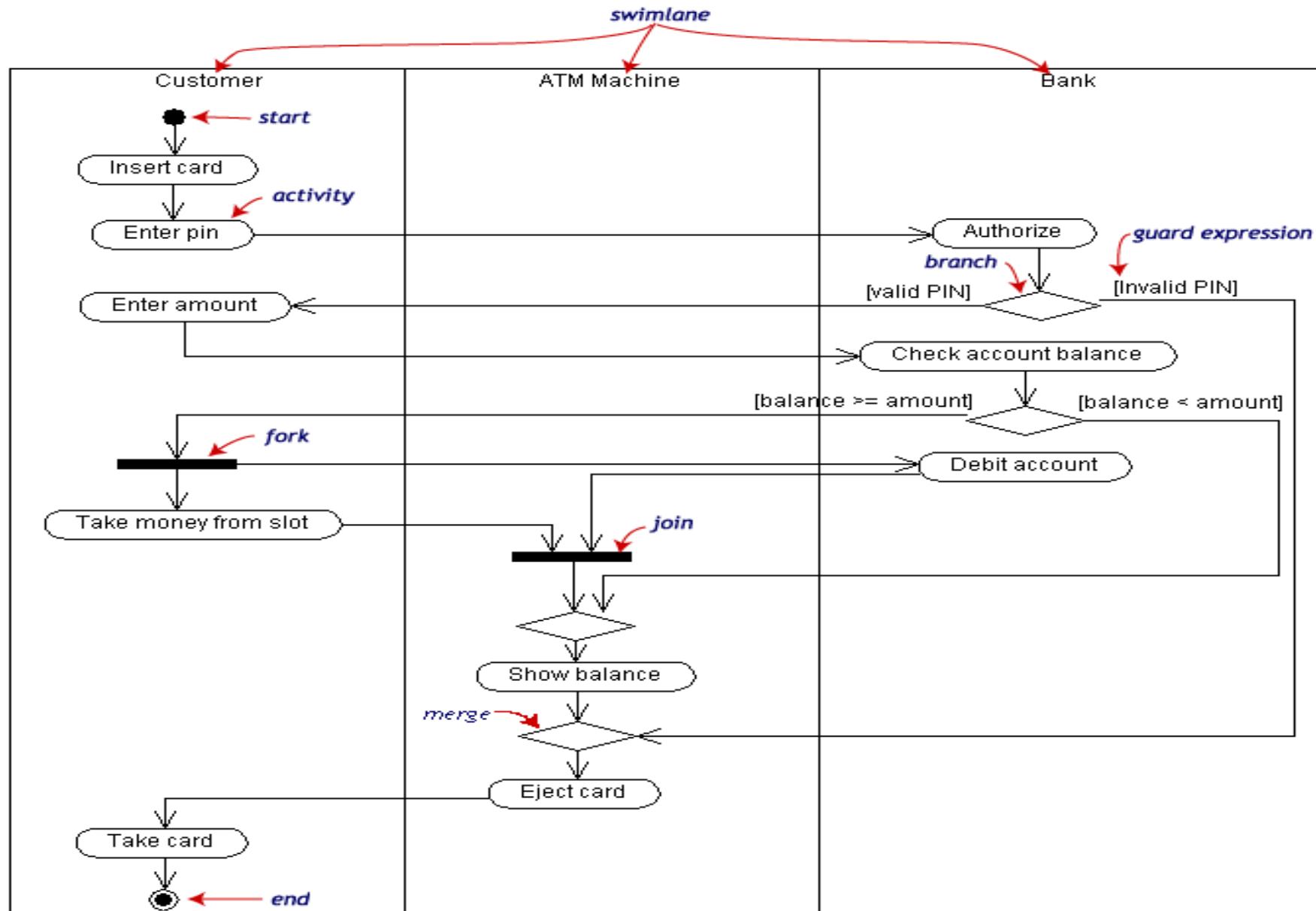
电梯状态图



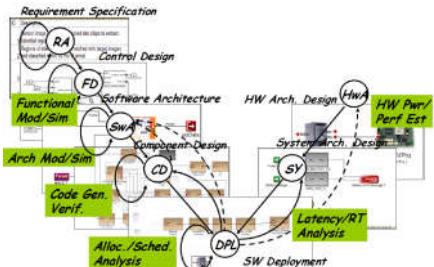
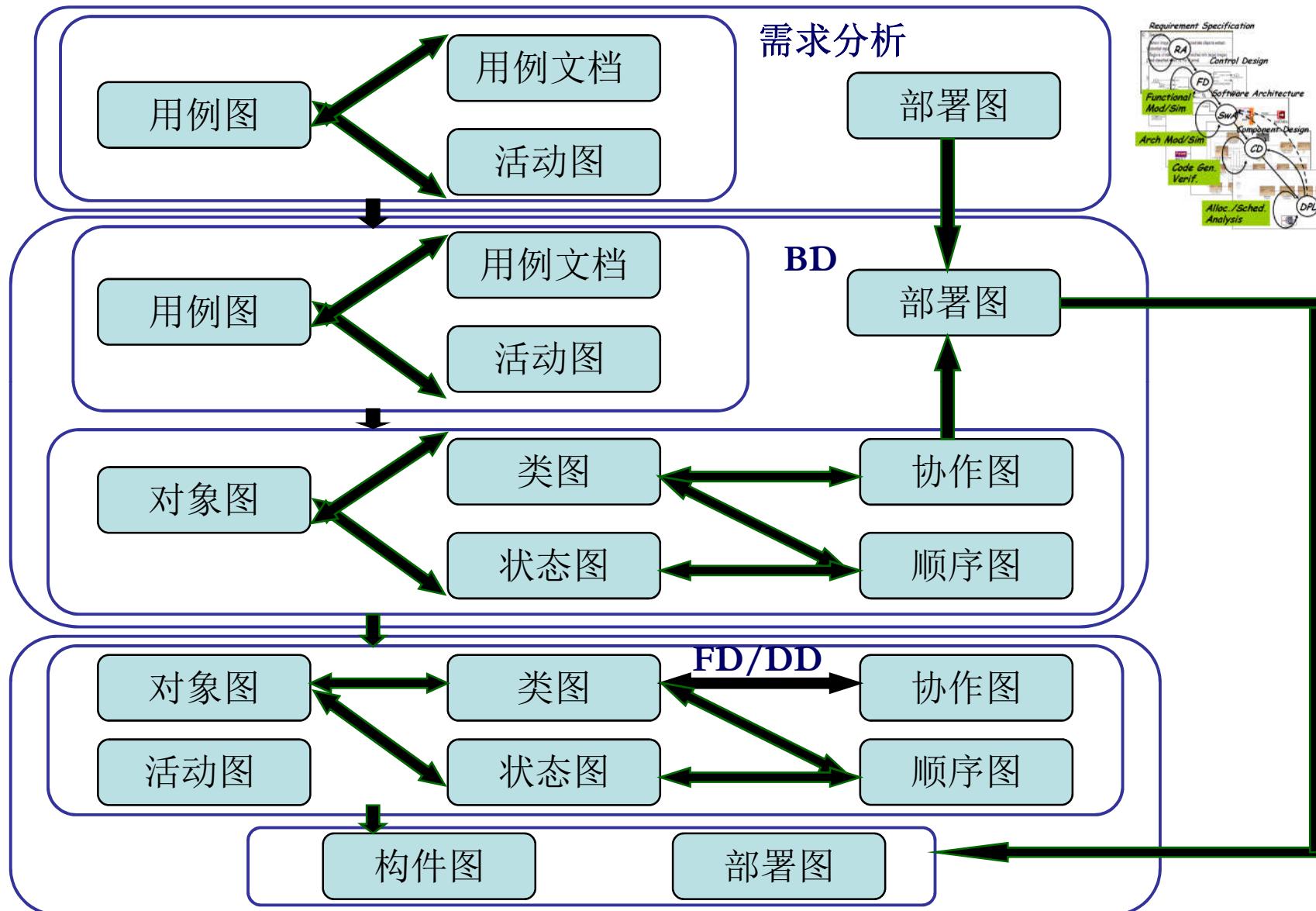
电梯



Activity Diagrams(对象活动的流程图)



基于UML的设计过程



系统建模：静态建模机制

- 用例图：刻画系统功能和环境约束。
 - 用例：系统执行的一系列动作。
- 类图：类之间的关联、聚集、继承、依赖关系。
- 对象图：与类图具有相同的表示形式。
 - 对象是类的一个实例，对象间的链接是类间关联的实例。
 - 静态对象图：某一时刻的状态
 - 动态对象图：某一段的状态变化
- 包图：类、包和包之间的关系，描述软件体系结构
- 部件图：可以用来显示编译、链接或执行时部件（如源代码、二进制文件等）之间的依赖关系
- 配置图：硬件拓扑结构，可以显示运行时刻包、类、对象等在物理平台上的分布情况

系统建模：动态建模机制

- 顺序图：以时间顺序显示对象之间的交互活动
 - 水平线代表对象间交互的消息
 - 强调消息发送顺序。强调时间和顺序。
- 协作图：对象间的动态交互关系和联结关系
 - 体现对象之间的静态联结关系
 - 强调上下文关系。消息收发关系（交互关系）。
- 状态图：对象接受激励后的状态迁移
 - 状态的改变需要事件的激发
- 活动图：状态图的特殊形式
 - 根据**状态**变化来捕获动作和动作执行的结果
 - 活动图中，一个活动结束后立即执行下一个活动

建模过程：静态与动态

- 识别系统的动态和静态属性
 - 系统包含一些过程。过程是由方法或行为实现的。
 - 预定义方法构成了系统的**静态属性**
 - 当把现实场景应用于实现特定任务时，方法构成了系统的**动态属性**
- 画出协作图：按对象的组织对控制流建模
- 画出时序图：按时间顺序对控制流建模
- 画出状态图：对象、事件、Moore状态机
- 画出构件图：一组组件之间的组织和依赖关系
 - 识别系统的软件构件
- 画出部署图
 - 识别系统的节点

What UML Offers to RT Modelers

- **Modeling of complex structures**
 - Class diagrams
- **Concurrency specification and management**
 - Active objects, run-to-completion
 - activity modeling, interaction modeling
- **Time: Timing diagrams** (时序图)
 - Simple Time model: a single global time source
 - Insufficient refinement for precise time modeling
- **Event handling: State machines**
- **Deployment: Deployment modeling**

What is Missing from UML

- A more sophisticated model of **time**
- A more sophisticated model of **concurrency**
- Lack of real-time domain concepts
 - E.g., traditional concurrency control mechanisms (semaphores, etc.), schedulers, scheduling policies, deadlines, deployment
- Ability to precisely specify quantitative information
 - values and functional relationships

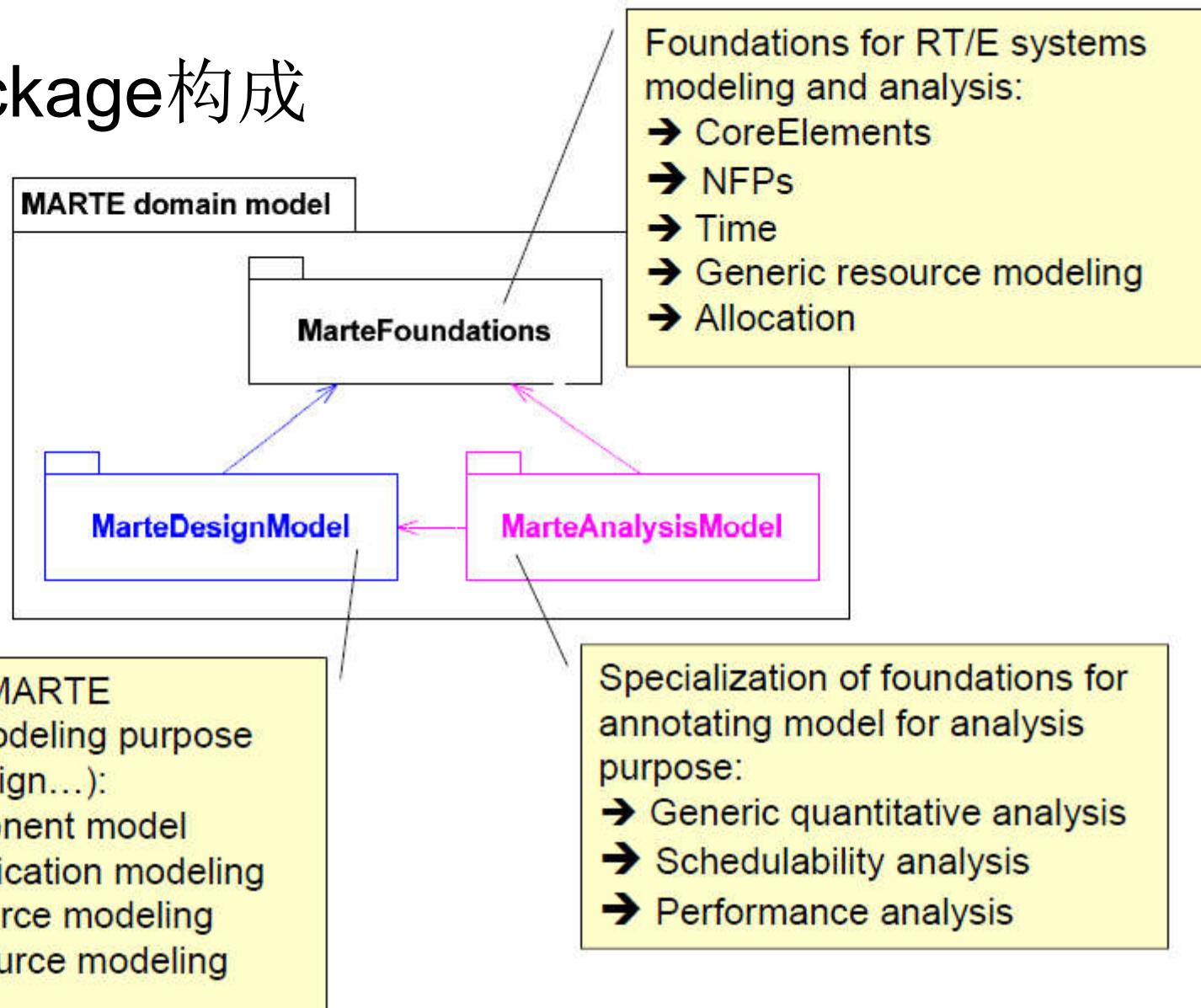
UML Profile for MARTE

- a domain-specific-interpretation (*profile*) of the general UML that coverage of the real-time domain!
 - replaces an earlier standardized UML profile SPT
 - *Scheduling, Performance, and Time*
 - SPT aligned with UML1.3, MARTE aligned with UML2.0
- MARTE defines the **language constructs only!**
 - 通用模式, 基本组件, 标准NFP附注 (annotation)
 - 一般性约束
- MARTE支持
 - 定义UML模型的NFP
 - 时间模型 (支持分布式)
 - 硬件资源建模: processors, mem, I/O devices, NWs
 - 软件资源建模: threads, processes, mutexes
 - 软硬件映射: allocation (deployed)
 - 实时性分析: schedulability, end-to-end delays, response

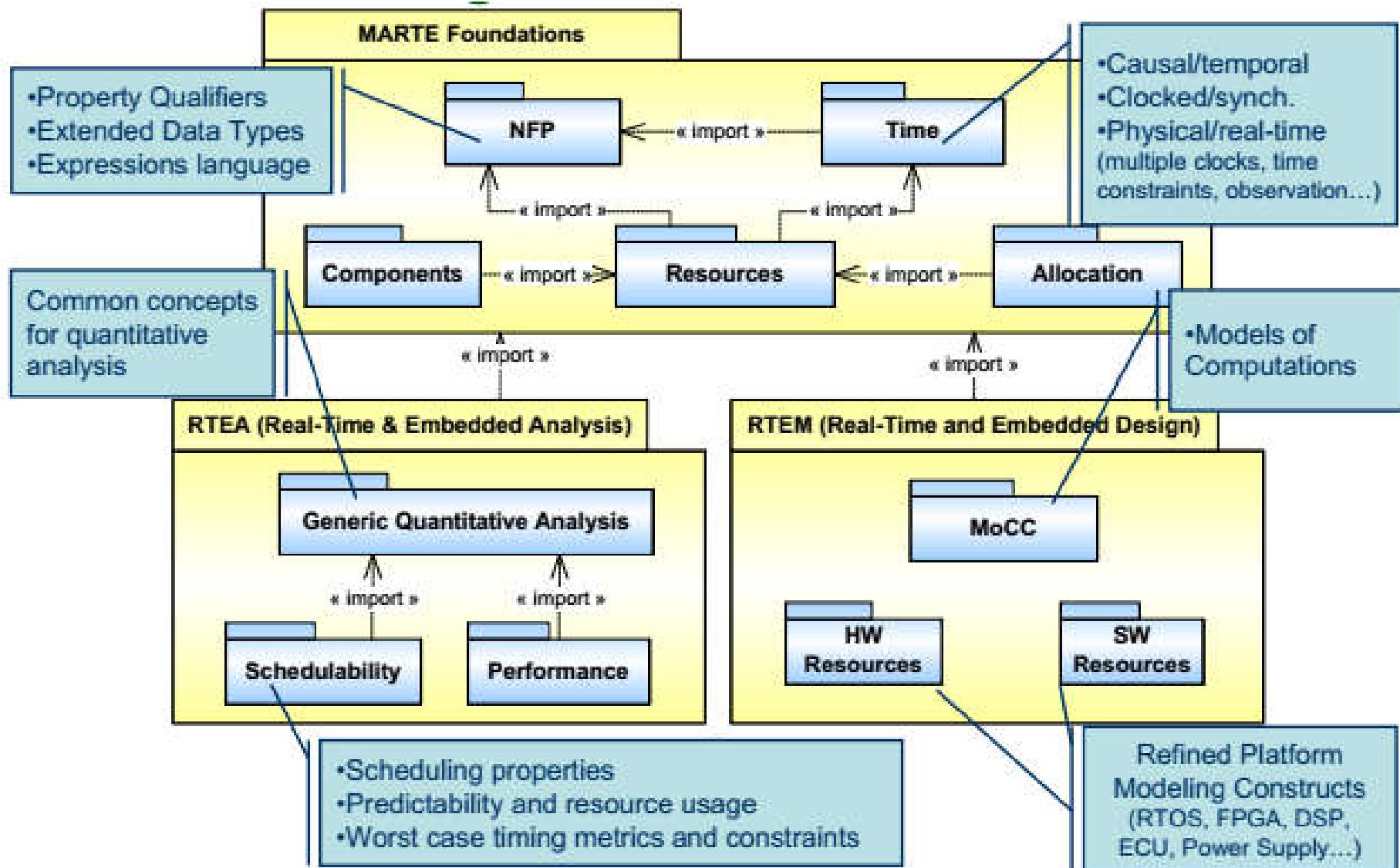


MARTE Overview(OMG, 2007+)

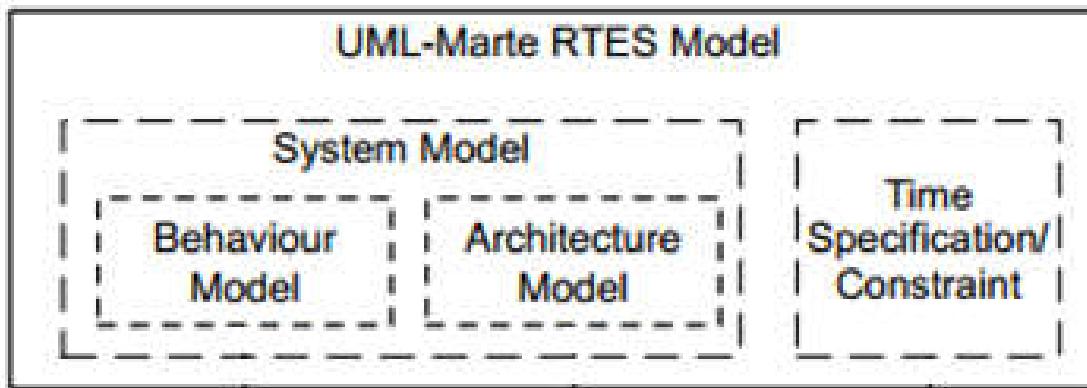
三个Package构成



MARTE High Level Architecture



System Model and Time



- Principles of Time modeling with MARTE
 - 在设计的不同阶段, time有不同的概念
 - Instant (point), Interval, Duration
 - **Time Base:** an ordered set of *Instants*
 - A **timed event** occurrence refers to one instant
 - 基于partial ordering(**simultaneity**) of instants
- UML Behavior modeling using **Clocks**
 - Clocks as model elements
 - multiple heterogeneous clocks
 - Relations and constraints between Clocks

Structure of Time

- time bases
- multiple time bases
- instants
- time relationships

Access to Time

- clocks
- logical clocks
- chronometric clocks
- current time

Using Time

- timed elements
- timed events
- timed actions
- time constraints

Clocks Relations and constraints

- 活动的temporal ordering主要有三种（精度）模型
 - **Causal/temporal:** 关心precedence/dependency
 - 时态逻辑
 - 并发表示为偏序关系
 - 实体间通信：完全异步、阻塞（同步等待）、握手同步三种
 - **Clocked/synchronous:** 基于同步假设
 - *instantaneous reaction:* 以tick为单位（zero-delay, 无duration）
 - 可为软件模型（同步语言）和硬件模型（HDL）
 - **Physical/real-time:** 采用实际的持续时间值
 - 可用于Clocked/synchronous模型
- Clocks的二元关系a binary relation: instant graph
 - 同时发生coincident , \equiv , same, 同步
 - 先于precedence, \leq , 允许同时发生
 - 严格先于strict precedence, $<$, 不允许同时发生
 - 排除exclusion, #, 永远不会在同一时刻发生

Concrete time instant relations

- CoincidenceRelation: 同时发生，强

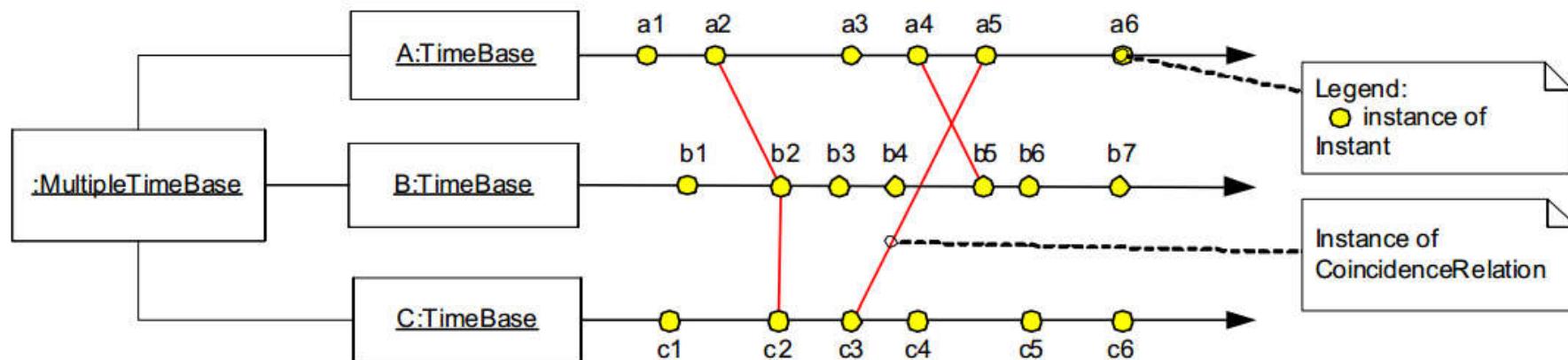


Figure 9.6 - Example of multiple time base with coincidences

- PrecedenceRelation

instant graph

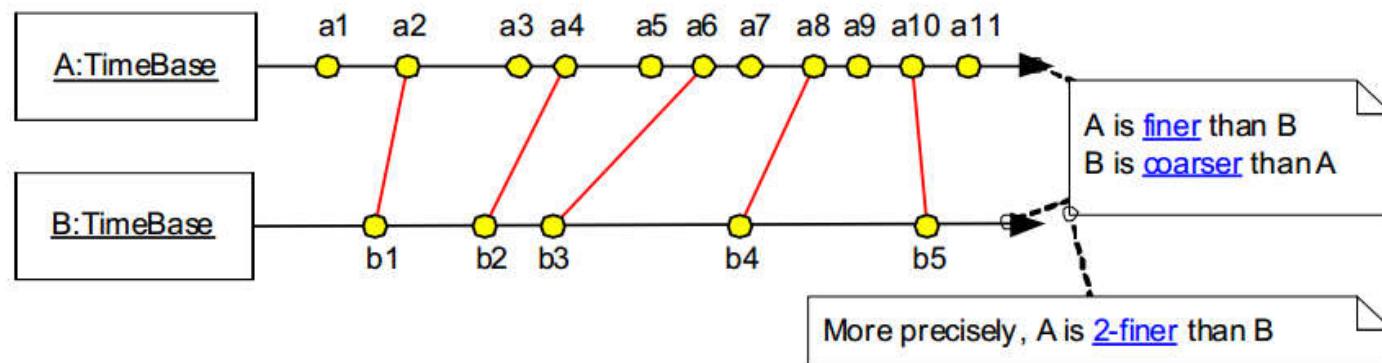


Figure 9.7 - Example of time relations between two time bases

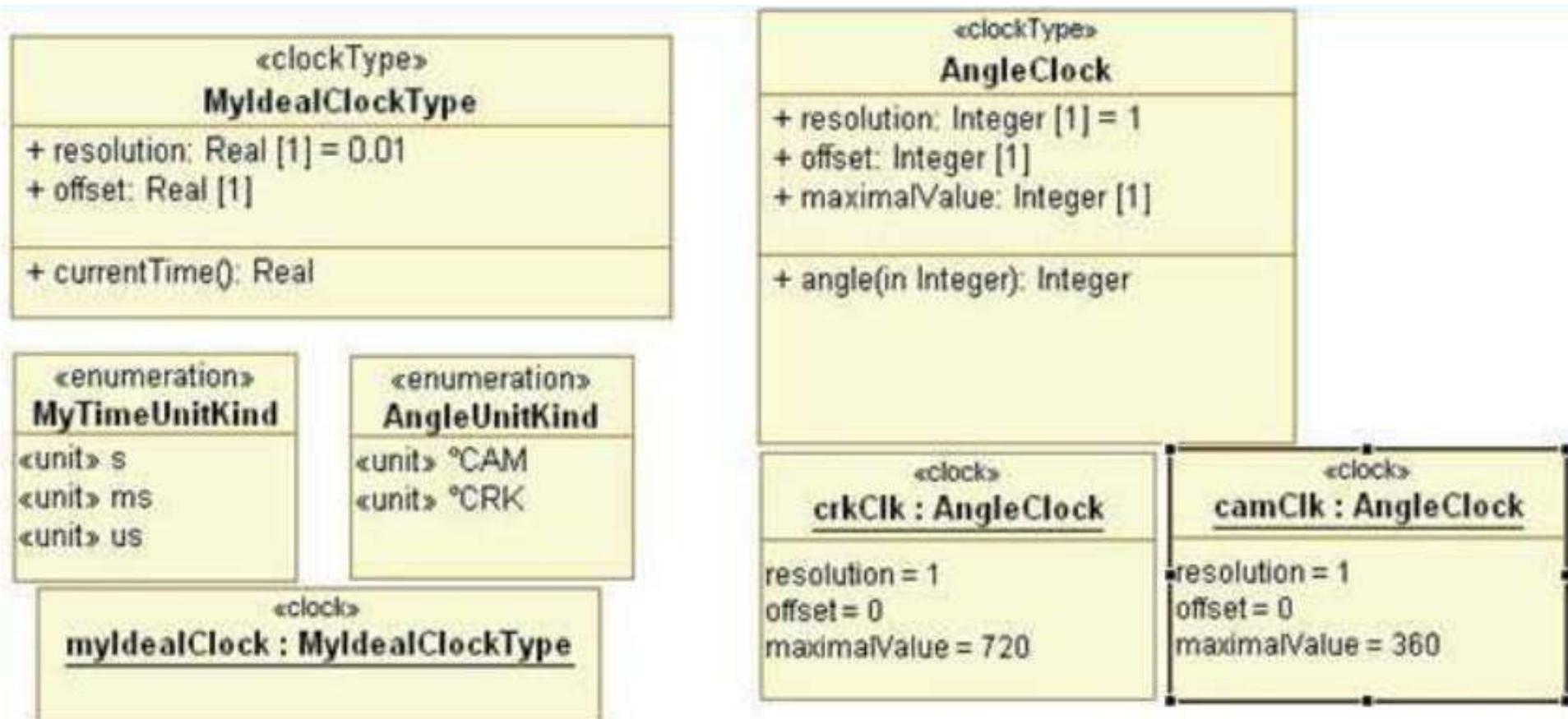
Temporal characteristics & timing constraints

- mathematical clock model: a 5-tuple $(\mathcal{I}, \preceq, \mathcal{D}, \lambda, u)$
 - (instants, order relation, labels, labeling function, symbol)
 - unit: s, ms, us, tick, cycle, step...
- application model or execution platform model
 - Model time(Logical time)
 - platform time (Logical or physical time)
- Logical or physical Clock
 - an “ideal” clock: IdealClock
 - The MARTE::TimeLibrary
 - chronometric clocks: 精密时钟
 - User defined clock based on IdealClock
- Clock constraints
 - (value = 1.5, unit = ms, onClock = 'idealClk')
 - 1.5 ms on idealClk
 - (value = 1, unit = ms) + (value = 150, unit = us)
 - This expression is implicitly on the idealClk clock.
 - Its value is (value = 1.150, unit = ms) or (value = 1150, unit = us)



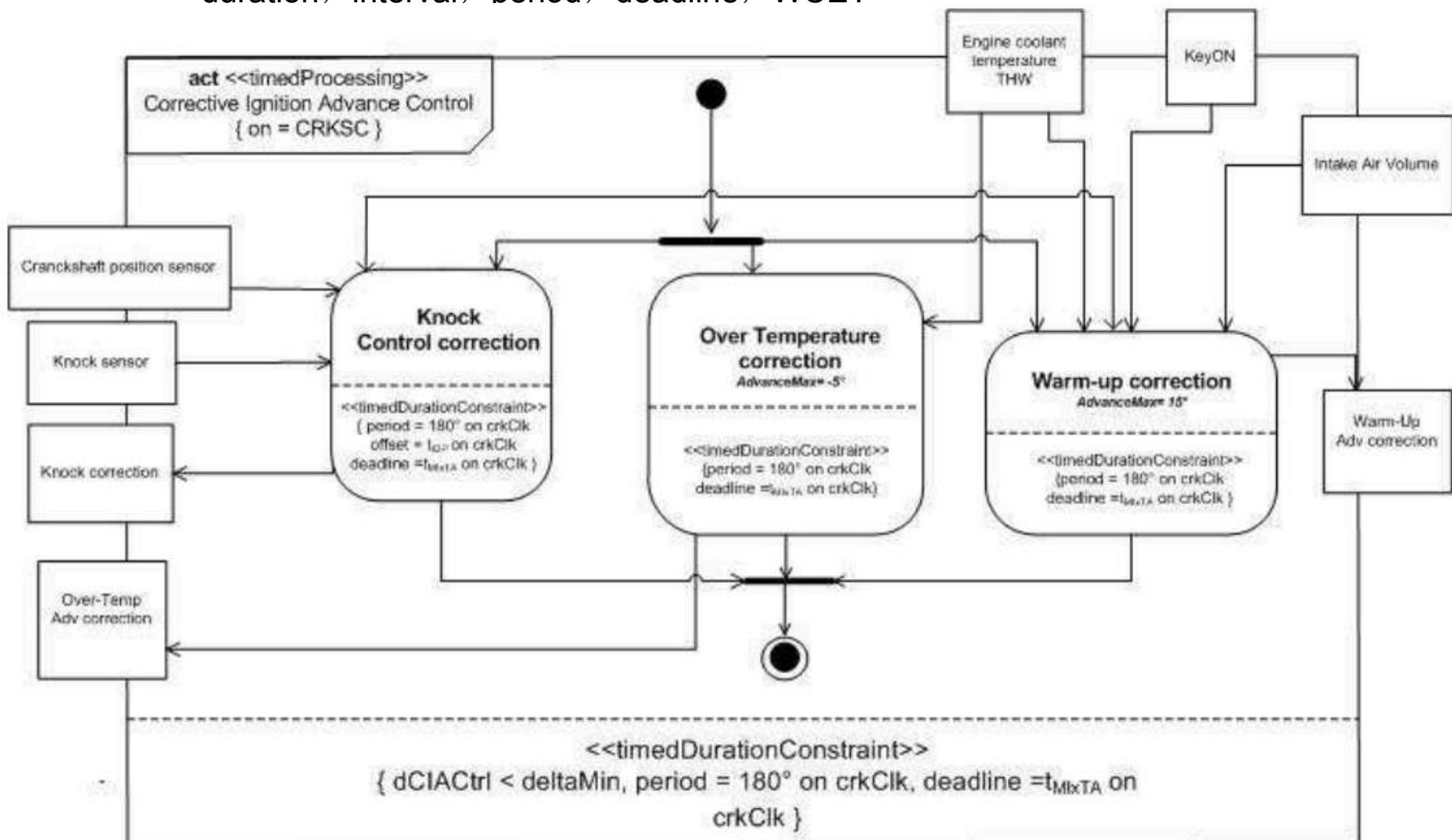
multiple clocks

- 模型中存在不同类型的多个CLOCK
 - CCSL(Clock Constraint Spec Lang)定义各clk的关系
 - Coincidence、Precedence等
 - 如: camClk = crkClk filteredby 0b(10)



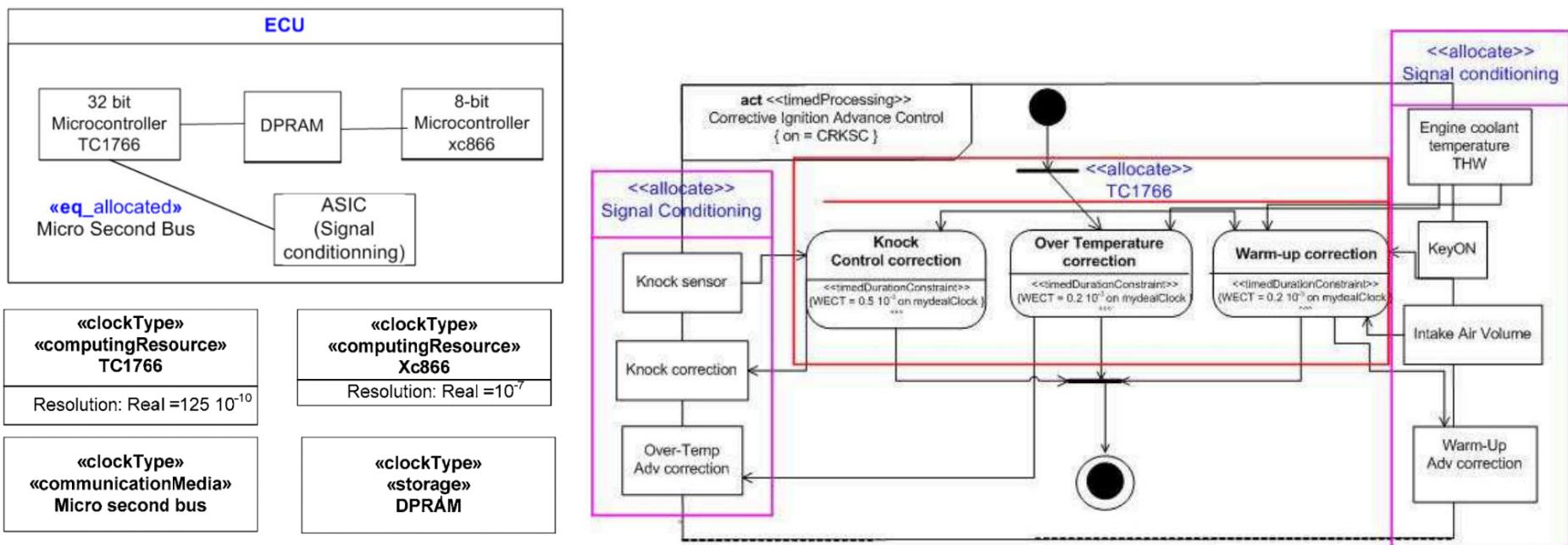
UML Behavior modeling using Clks

- activity diagram, sequence diagram, state machine
 - duration, interval, period, deadline, WCET



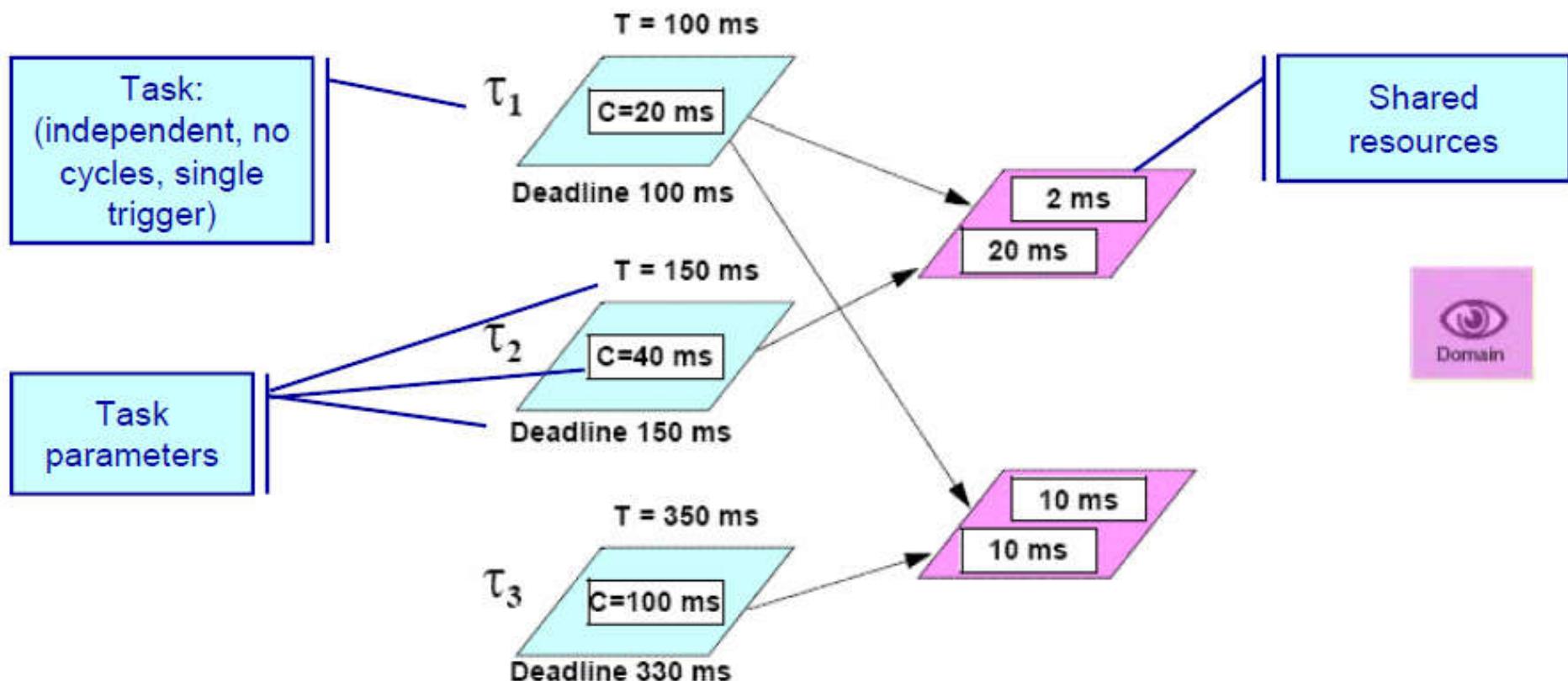
Allocations

- 关联application model和platform model
 - Allocations are annotated with time constraints and temporal characteristics.
 - 将模型时间与实际物理时间相关联 (WCET)
- Time model of hardware: 指令执行时间, 传输率



Typically analyzed with RMA

- Critical instant calculation
- Utilization bound test or Response time calculation for the first deadline



Verification of Sync Related Properties

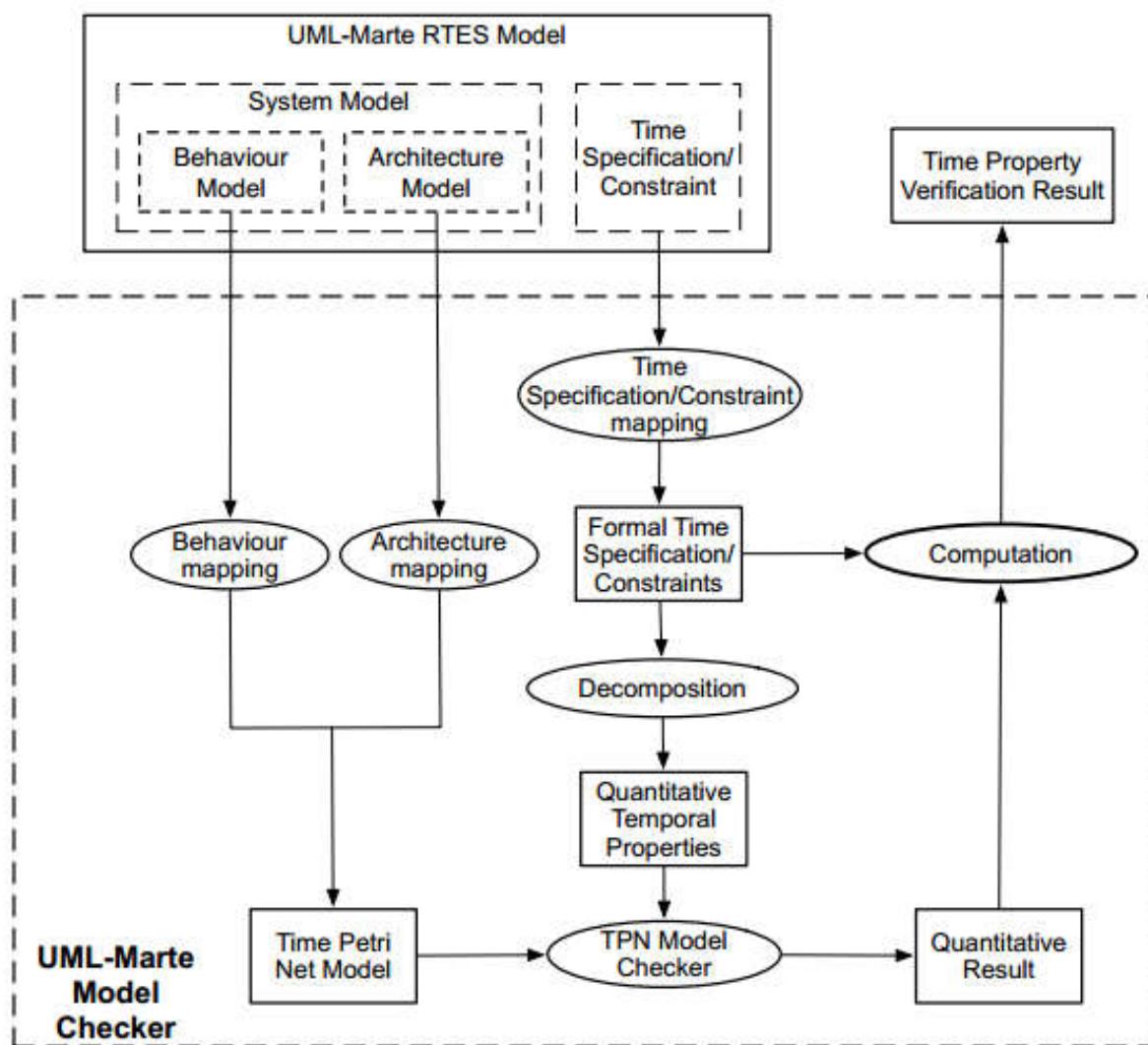


Figure 1. UML-MARTE Model Checker

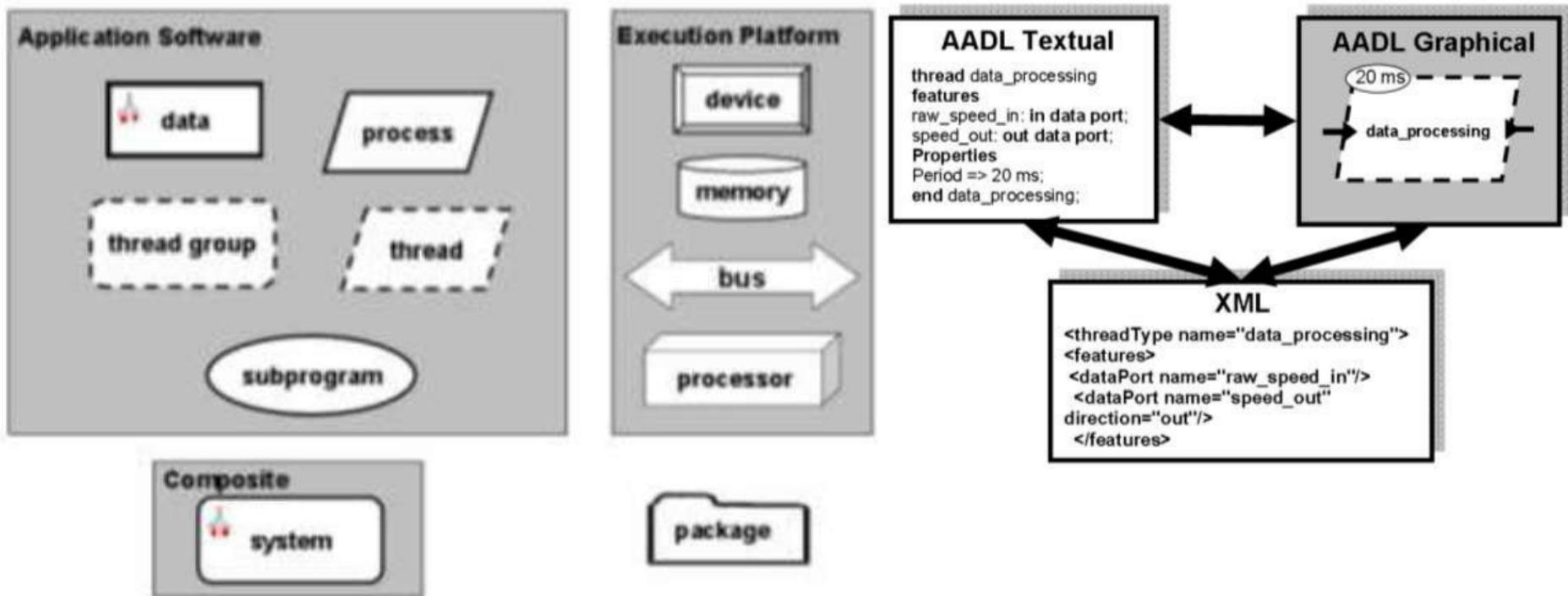
The AADL Language

- Architectural Analysis and Design Language
 - Derived from an earlier ADL called MetaH developed by Honeywell for the US DoD
 - Defined by the “AS 2C ADL Subcommittee of the Embedded Systems Committee of the Aerospace Avionics Division of SAE Aerospace”
 - SAE report: AS-5506
- For the design of dependable embedded real-time systems
 - Hardware and software components
 - Control, data, and access connections
 - **Strong focus on timing** (schedulability) and reliability characteristics
 - Supports automated analysis through specialized tools
 - aids in design space exploration
- Formal execution semantics in terms of **hybrid automata**



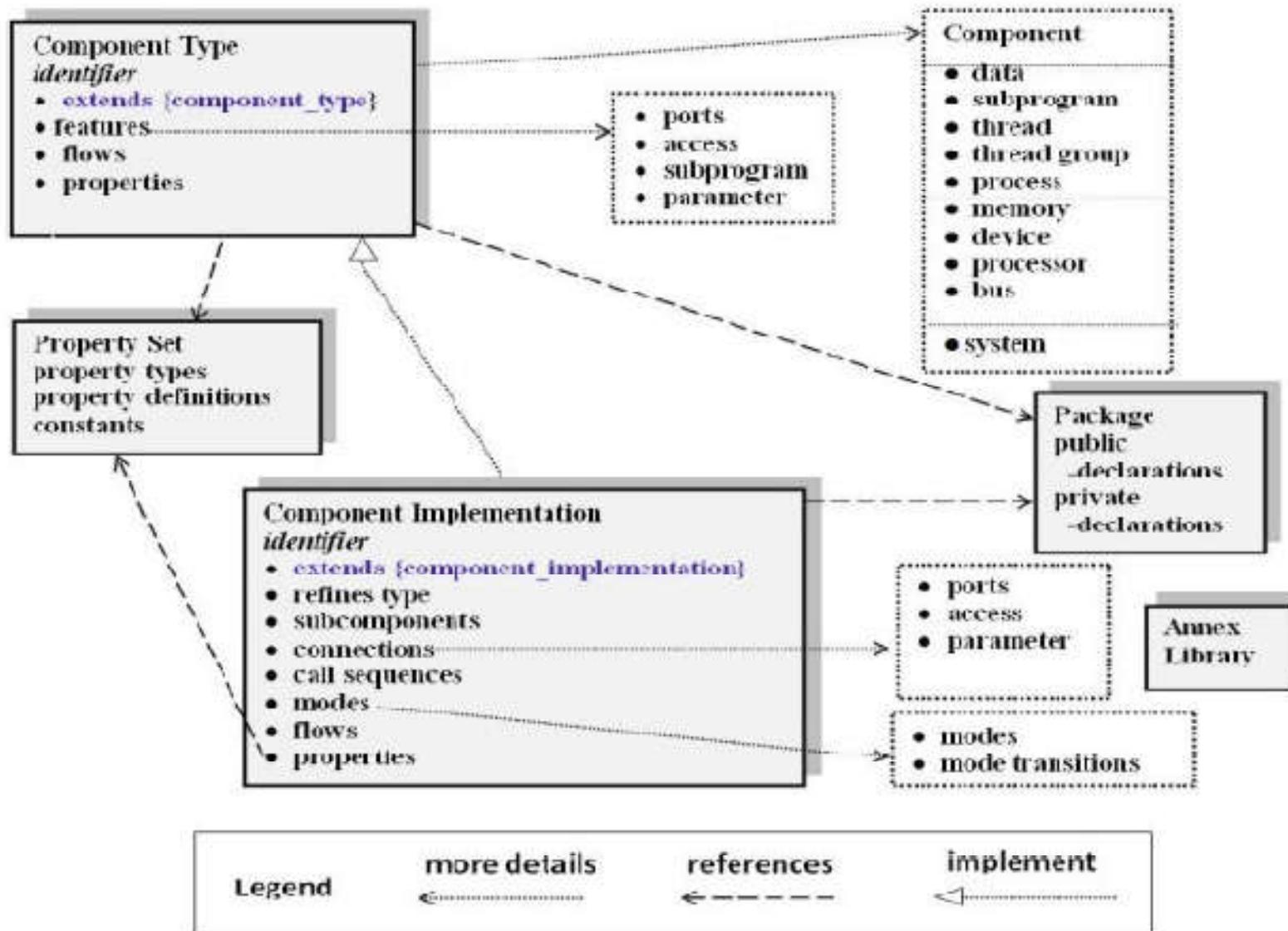
AADL语法

- Architectural Elements



- 三种表示形式：文本、图形、XML

AADL components 定义——CBD



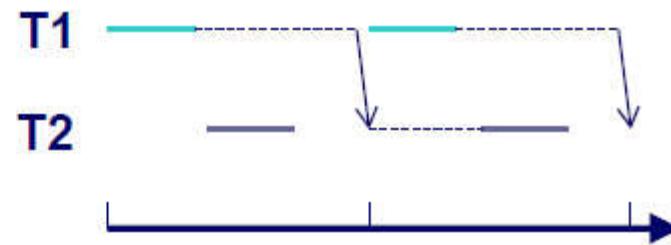
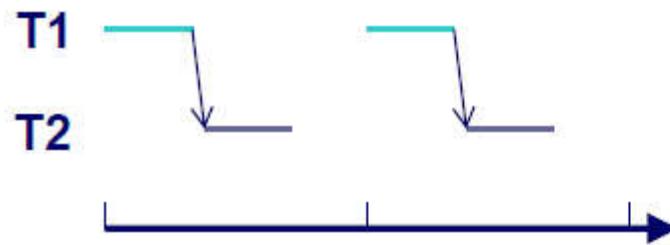
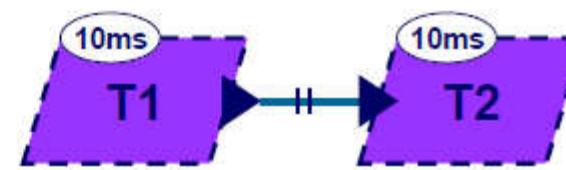
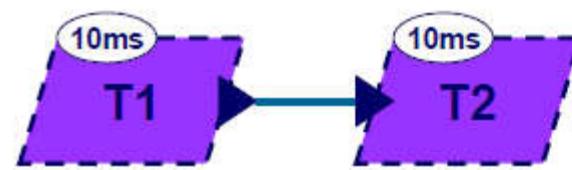
Thread dispatch

- Periodic threads are dispatched (触发) periodically
 - Event arrivals are queued
- Non-periodic threads are dispatched by incoming events
- Pre-declared ports
 - Event in port **Dispatch**
 - If connected, all other events are queued
 - Event out port **Complete**
 - Can implement precedence



Immediate and delayed connections

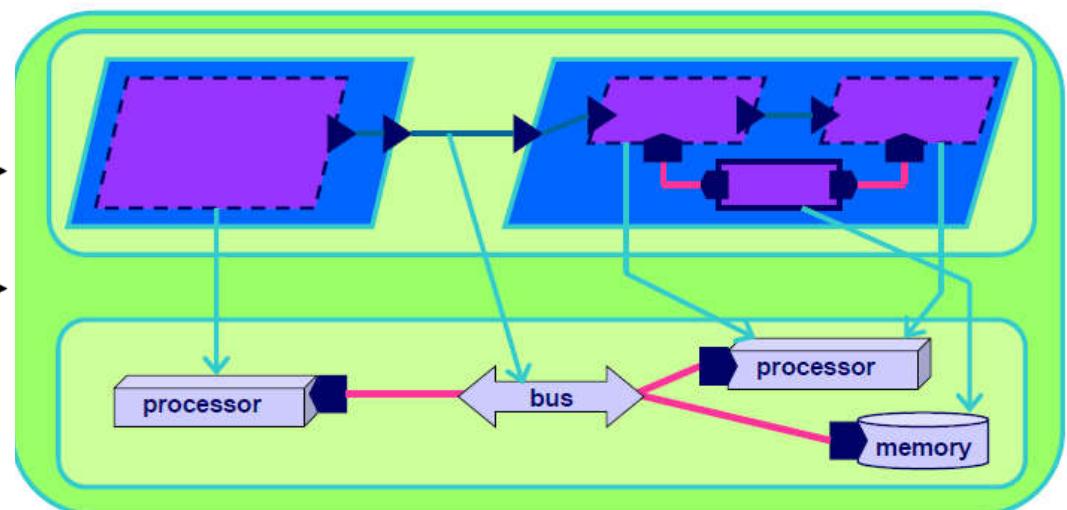
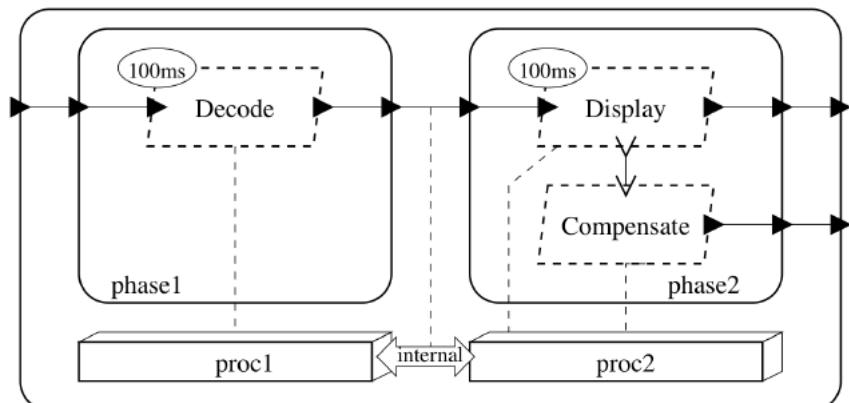
- Data connections between periodic threads



Component bindings

- Putting it all together: systems
 - Software components are **bound** to platform components

```
system Display
features
    speed : in data port speed_port;
    position : in data port position_port;
    screen_position : out data port position_port
end Display
```

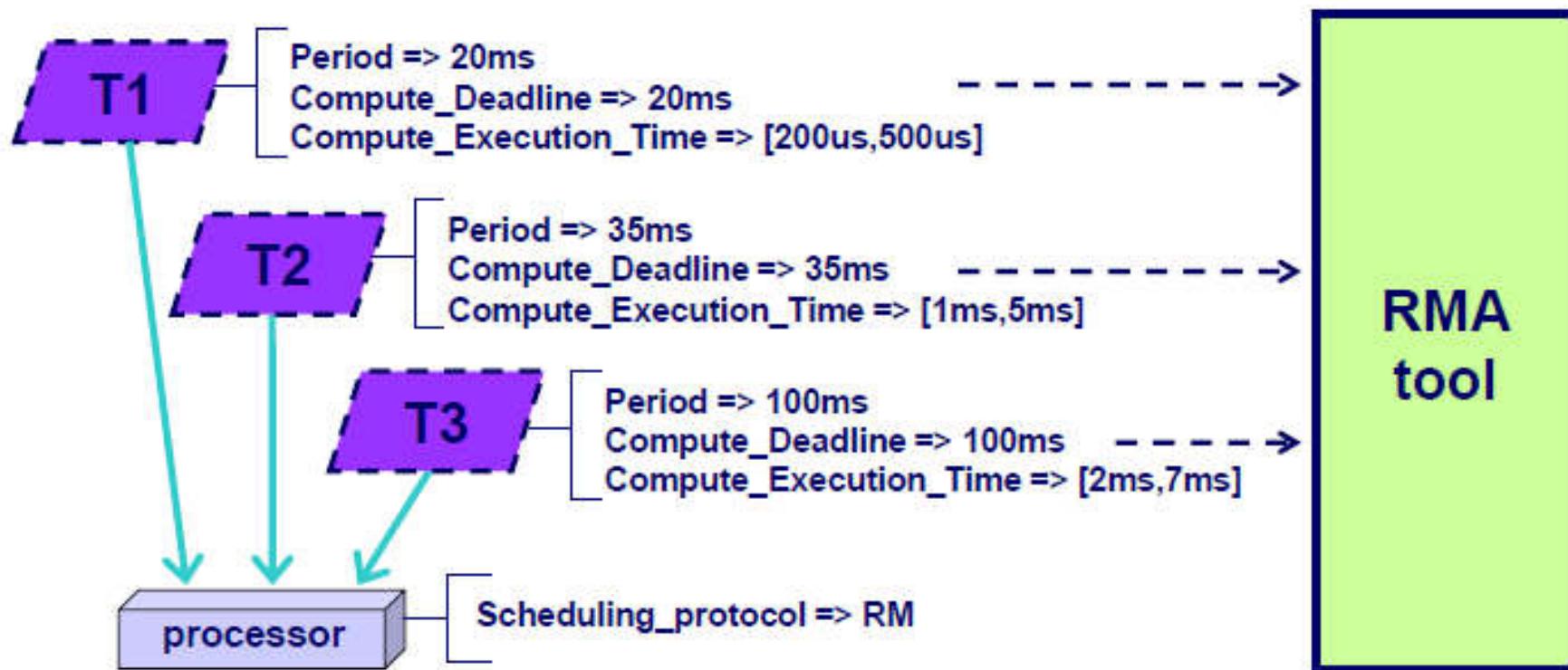


Analysis

- Flow Latency Analysis
 - end-to-end time
- Resource Consumption Analysis
- Real-Time Schedulability Analysis
- Safety Analysis
 - Checks the safety criticality level of system components
- Security Analysis
 - identifying the security loopholes

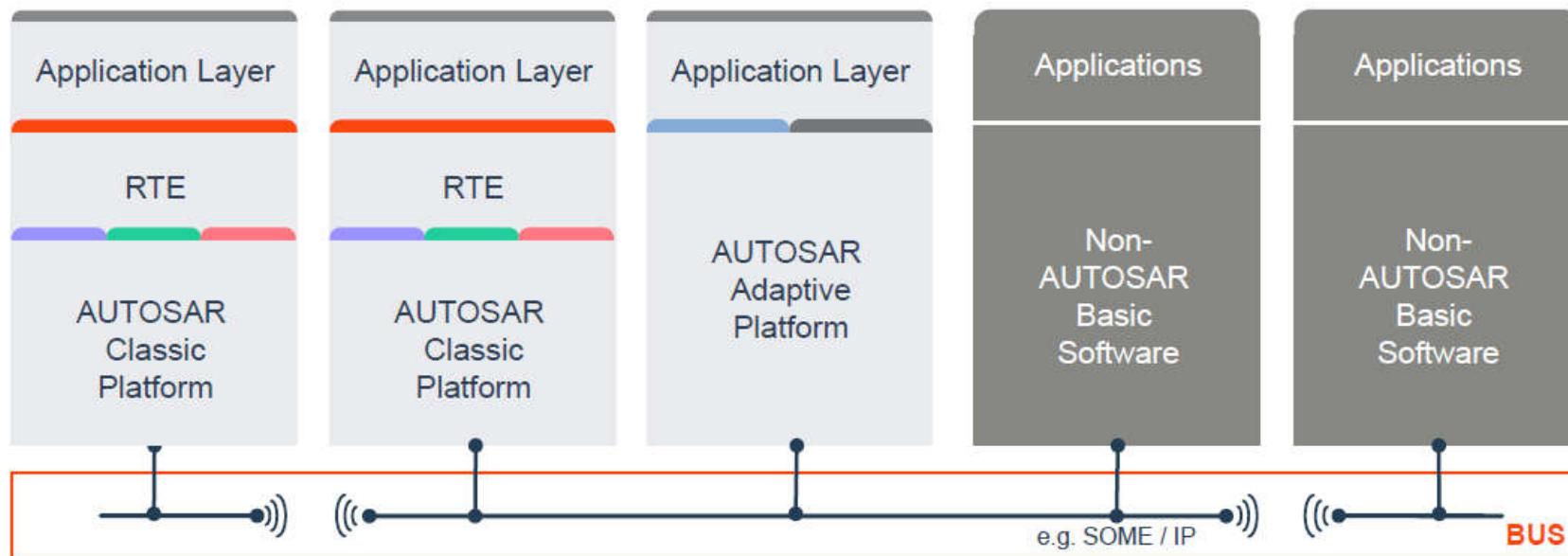
Dynamic architectural analysis

- Relies on **thread semantics**
- Processor scheduling

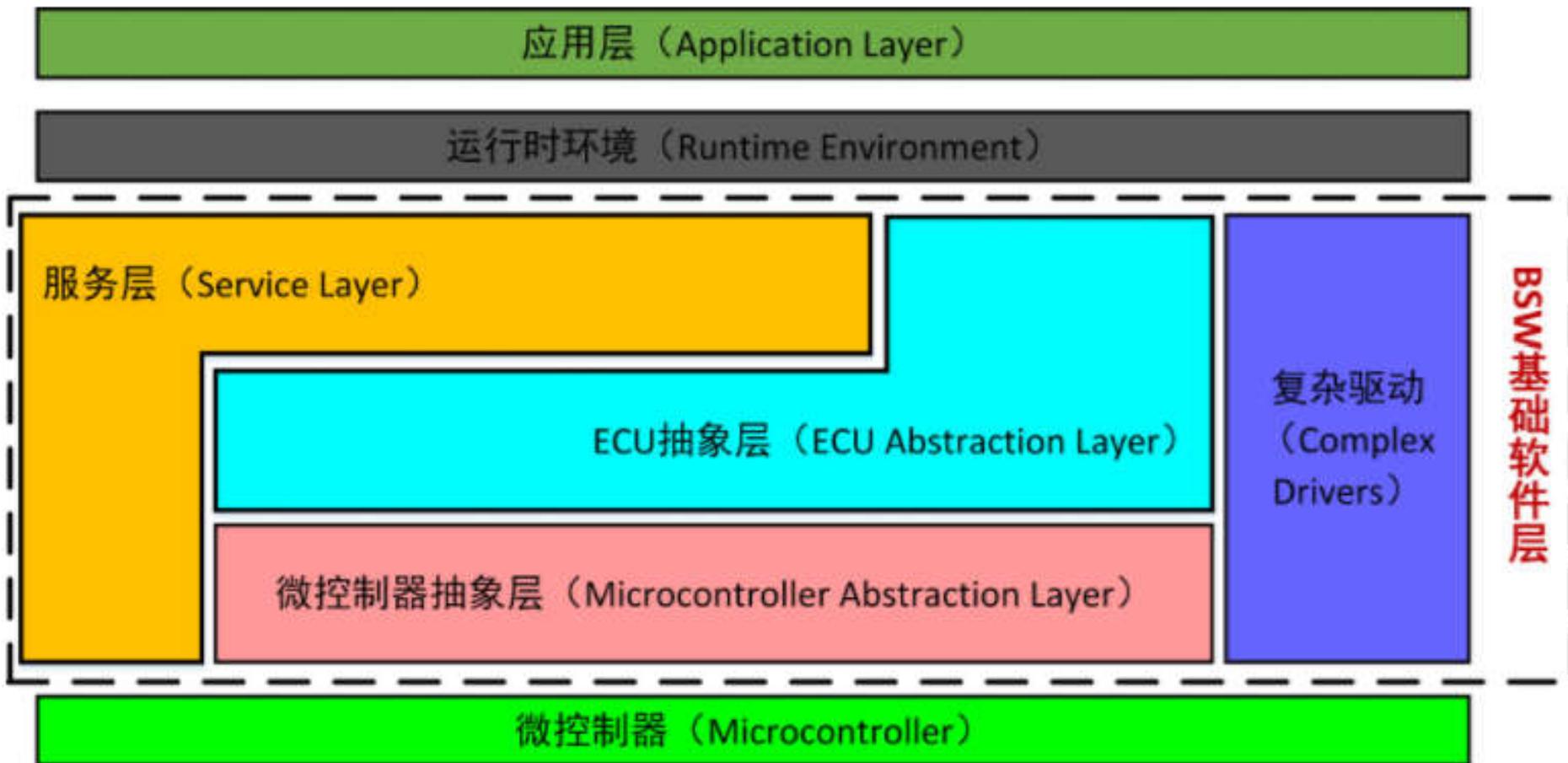


AUTOSAR Vision

- 通过提升软件模块的复用性和设计者之间的交流，改善汽车电子系统的复杂性管理
- 提出两种软件平台
 - **Classic:** 针对中小规模的微控制器
 - **Adaptive:** Car-2-X Apps, IoT, cloud services

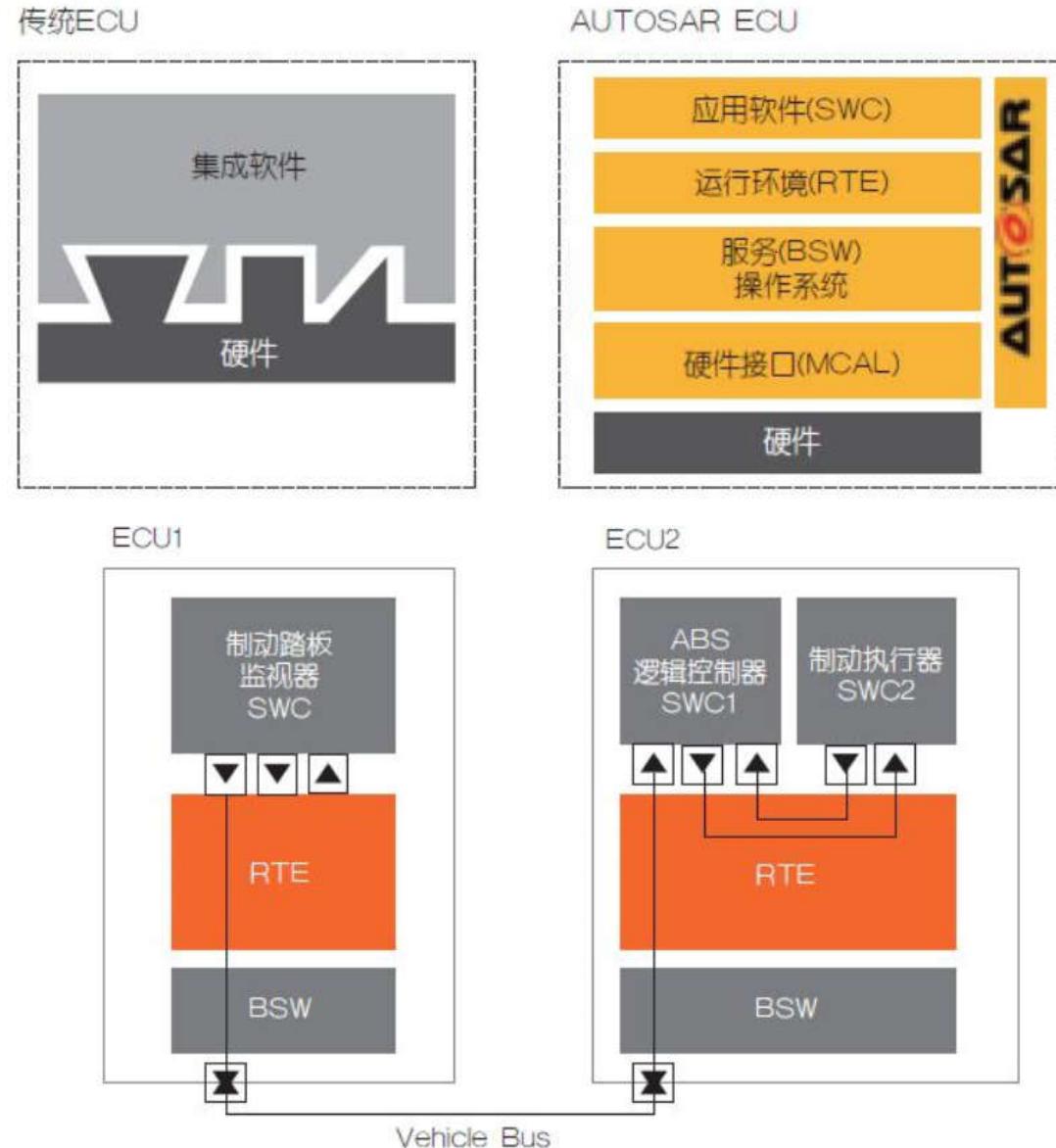


AUTOSAR体系结构分层模型

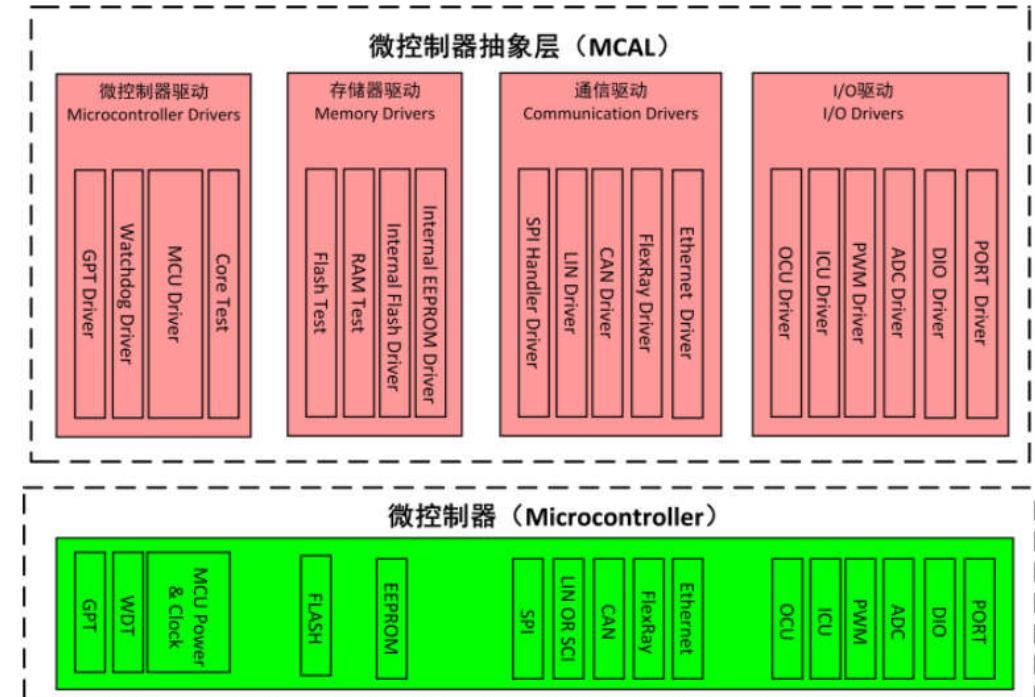
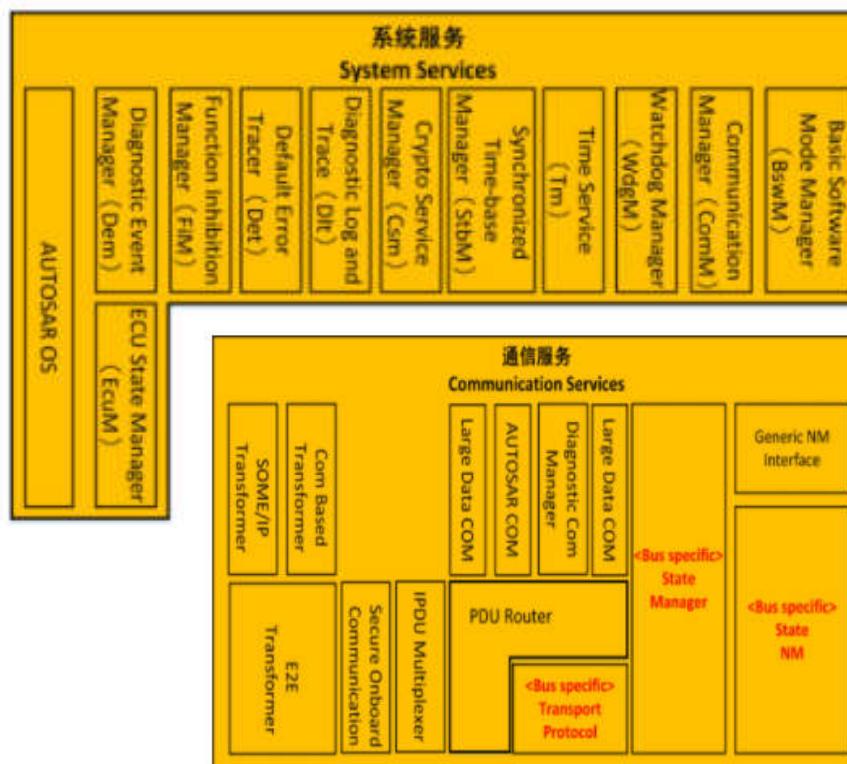
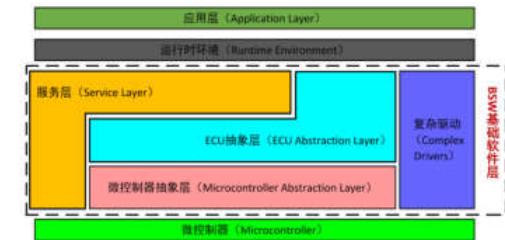


RTE

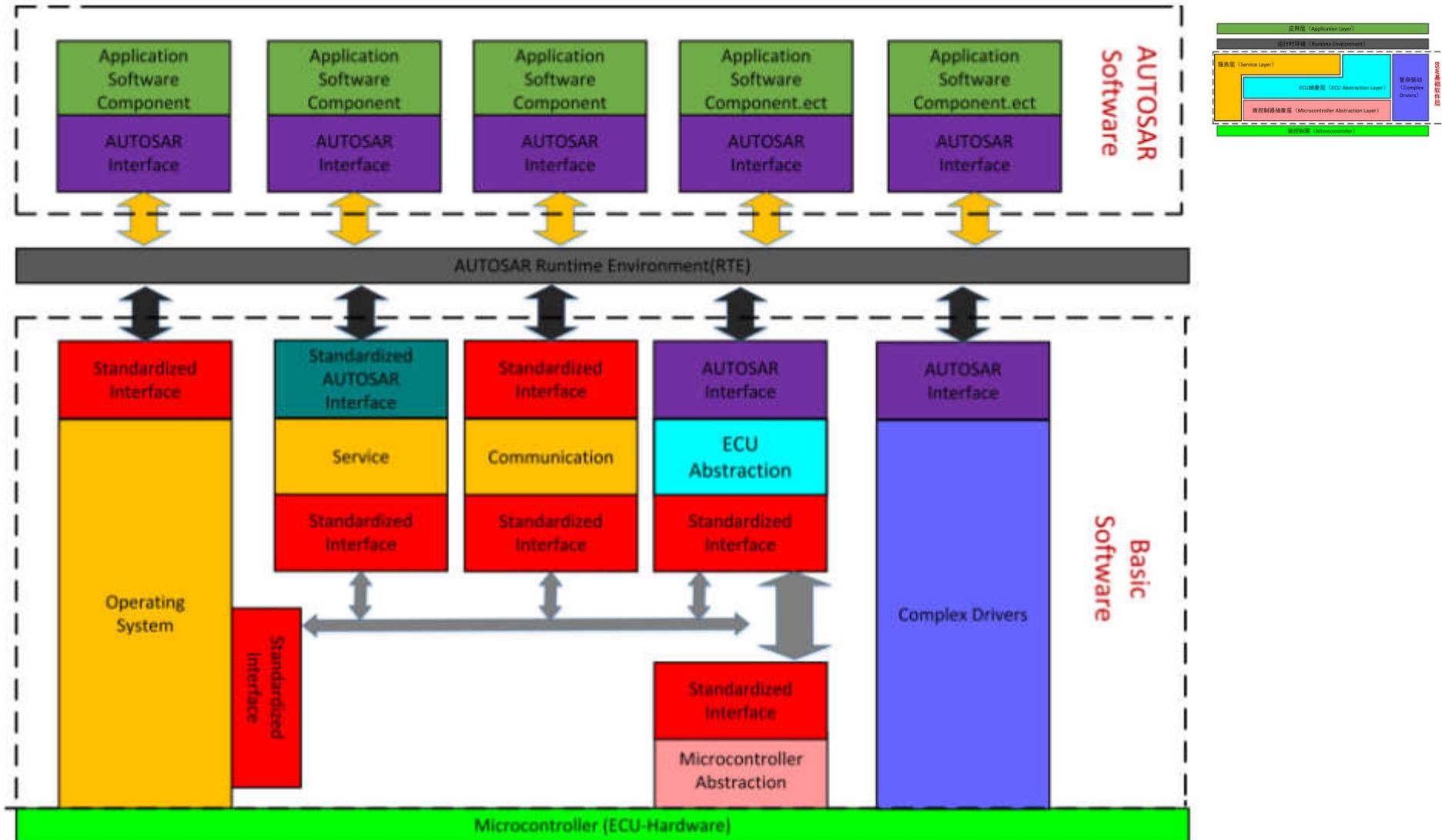
- Separation of system into its ECU plus common infrastructure



BSW和微控制器组件

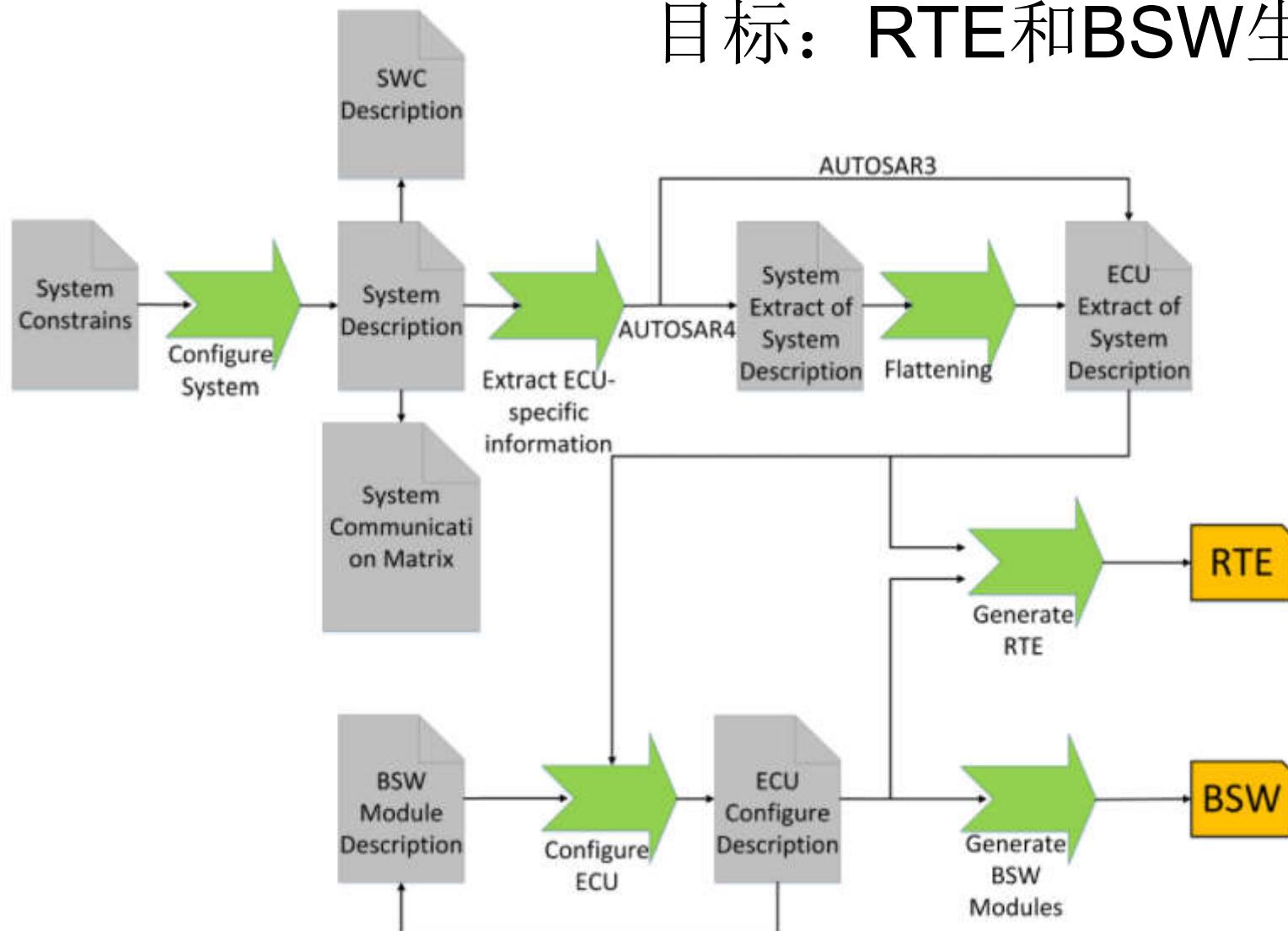


标准的AUTOSAR接口



AUTOSAR方法论

目标：RTE和BSW生成



Classic vs. Adaptive Platform



Based on OSEK

Execution of code directly from ROM

Same address space for all applications
(MPU support for safety)

Optimized for signal-based communication
(CAN, FlexRay)

Fixed task configuration

Specification



Based on POSIX

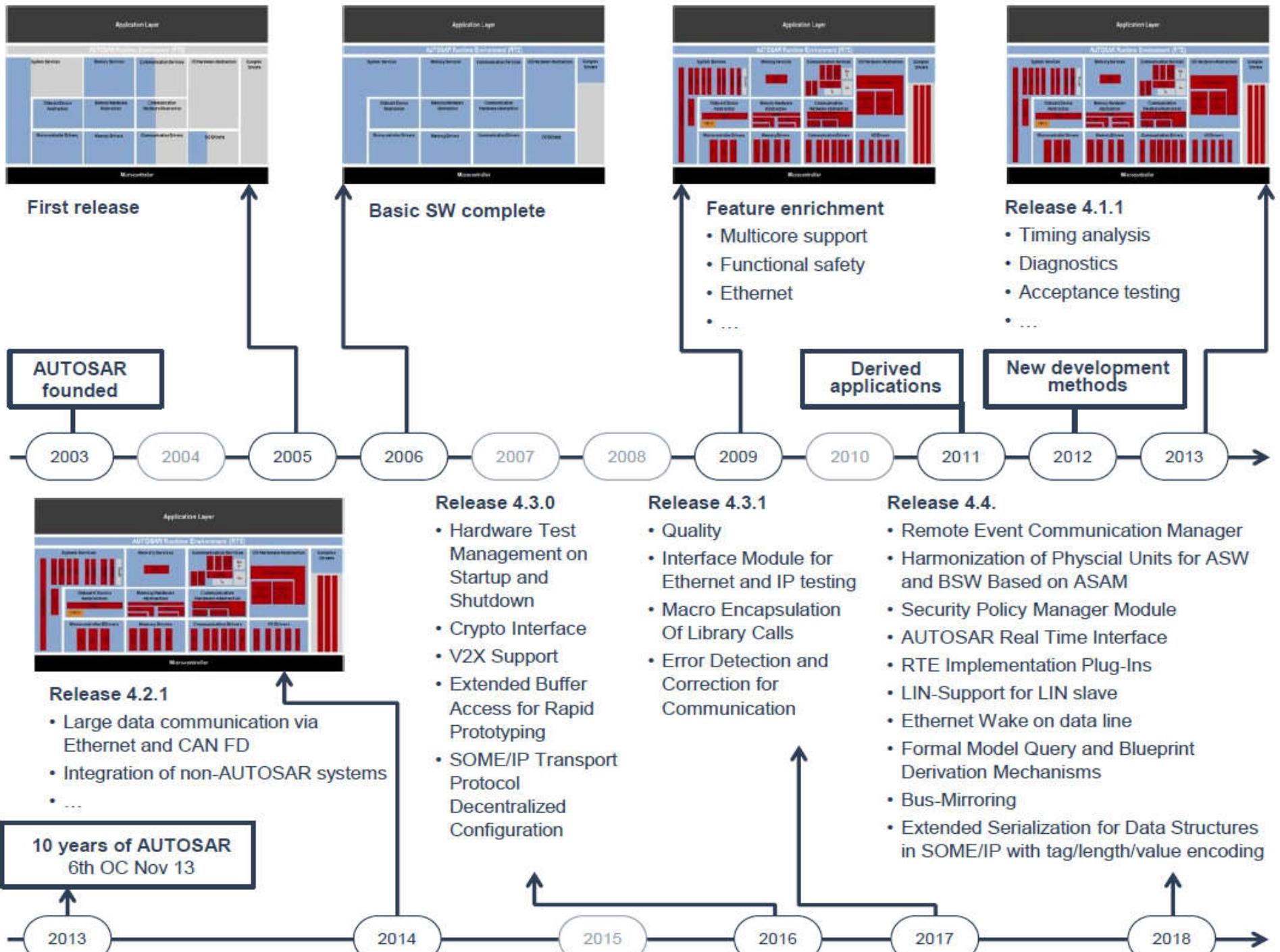
App. is loaded from persistent memory into RAM

Each application has its own (virtual) address space (MMU support)

Service-oriented communication

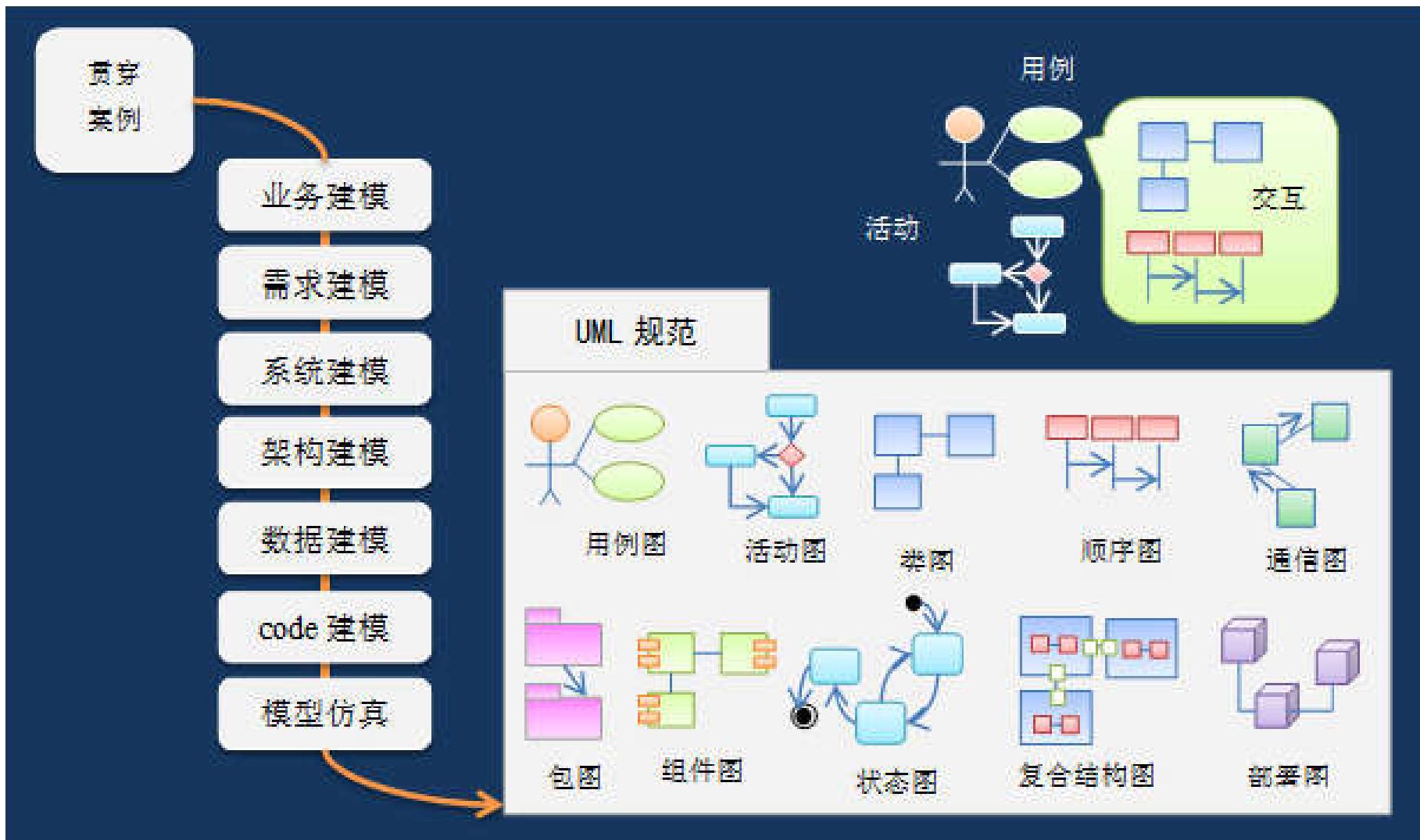
Support of multiple (dynamic) scheduling strategies

Specification and code



UML语言

- a modeling language for specifying, visualizing, constructing, and documenting software systems and business processes.



小结

- UML的时间概念是隐式的，没有定义
- MARTE补充了UML的时间概念
- AADL强调**timing**分析
- 作业与调研（选一）
 - UML的状态机是Mealy机还是Moore机？
 - 调研：Papyrus+MAST
 - 基于Eclipse平台MARTE建模工具
 - 调研：OSATE+Cheddar
 - Open Source AADL Tool by SEI

Thank you