**洒水a**

**软件架构设计之——统一建模语言UML**

**本教程为本人亲自整理，结合书本相关资料汇总而成**

**版权所有，转载请注明出处，谢谢合作！**

****安伟超

达内时代科技集团有限公司

天津天大中心教学部

[anwc@tedu.cn](mailto:anwc@tedu.cn)

17600945626

最后修改：2019.7.17

# 写在前面

给大家讲一个UML的免费学习网站：<https://www.w3cschool.cn/uml_tutorial/>

# UML简介

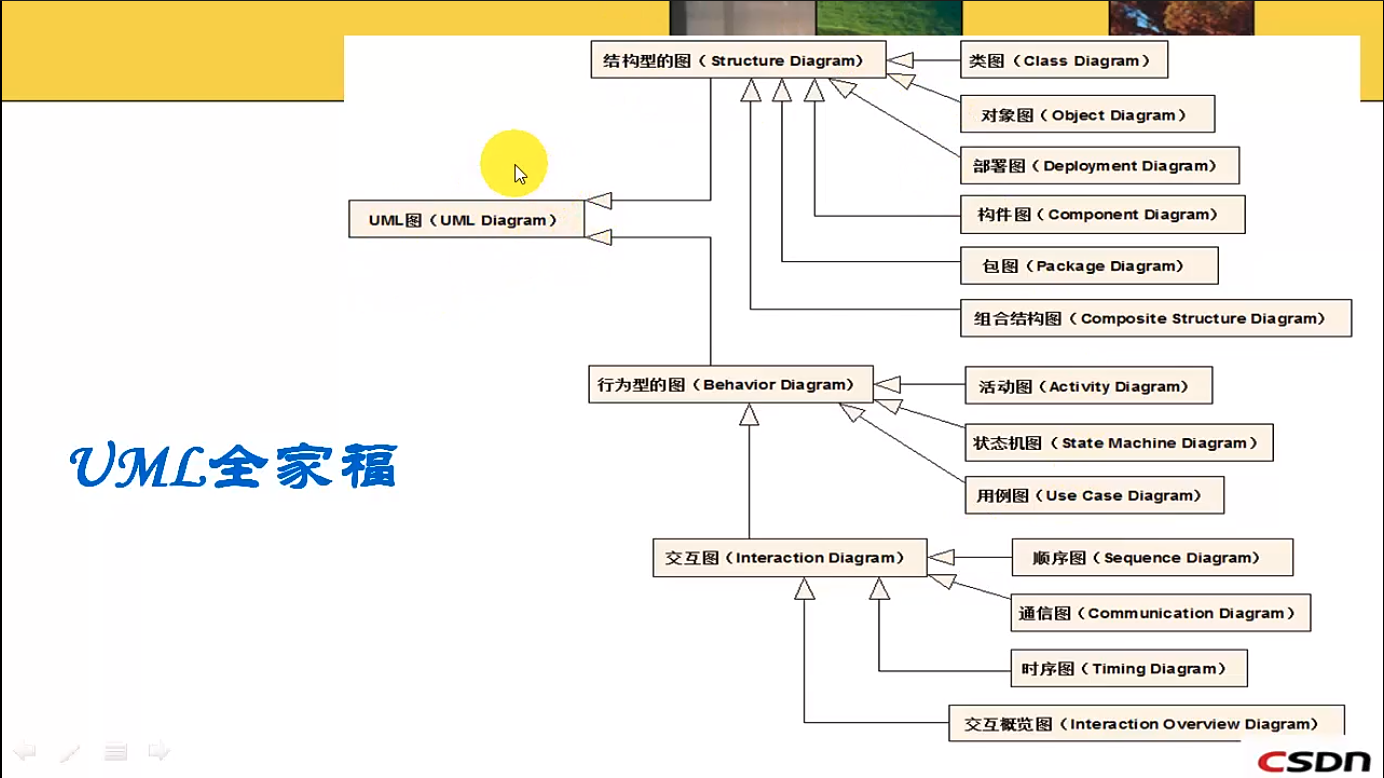
## 什么是UML

Unified Modeling Language (UML)又称统一建模语言或[标准建模语言](https://baike.baidu.com/item/%E6%A0%87%E5%87%86%E5%BB%BA%E6%A8%A1%E8%AF%AD%E8%A8%80/10967573)，是始于1997年一个[OMG](https://baike.baidu.com/item/OMG/3041465)标准，它是一个支持模型化和软件系统开发的图形化语言，为软件开发的所有阶段提供模型化和可视化支持，包括由需求分析到规格，到构造和配置。 面向对象的分析与设计(OOA&D，OOAD)方法的发展在80年代末至90年代中出现了一个高潮，[UML](https://baike.baidu.com/item/UML)是这个高潮的产物。它不仅统一了Booch、Rumbaugh和Jacobson的表示方法，而且对其作了进一步的发展，并最终统一为大众所接受的[标准建模语言](https://baike.baidu.com/item/%E6%A0%87%E5%87%86%E5%BB%BA%E6%A8%A1%E8%AF%AD%E8%A8%80/10967573)。

## UML基础概念

类（对象的）、对象、关联、职责、行为、接口、用例、包、顺序、协作，以及状态。

## UML图分类



这些图从不同的侧面对系统进行描述。系统模型将这些不同的侧面综合成一致的整体，便于系统的分析和构造。尽管UML和其它开发工具还会设计出许多派生的视图，但上述这些图和其它辅助性的文档是软件开发人员所见的最基本的构造。

## 不使用UML会产生的问题

a) 没有统一的标准

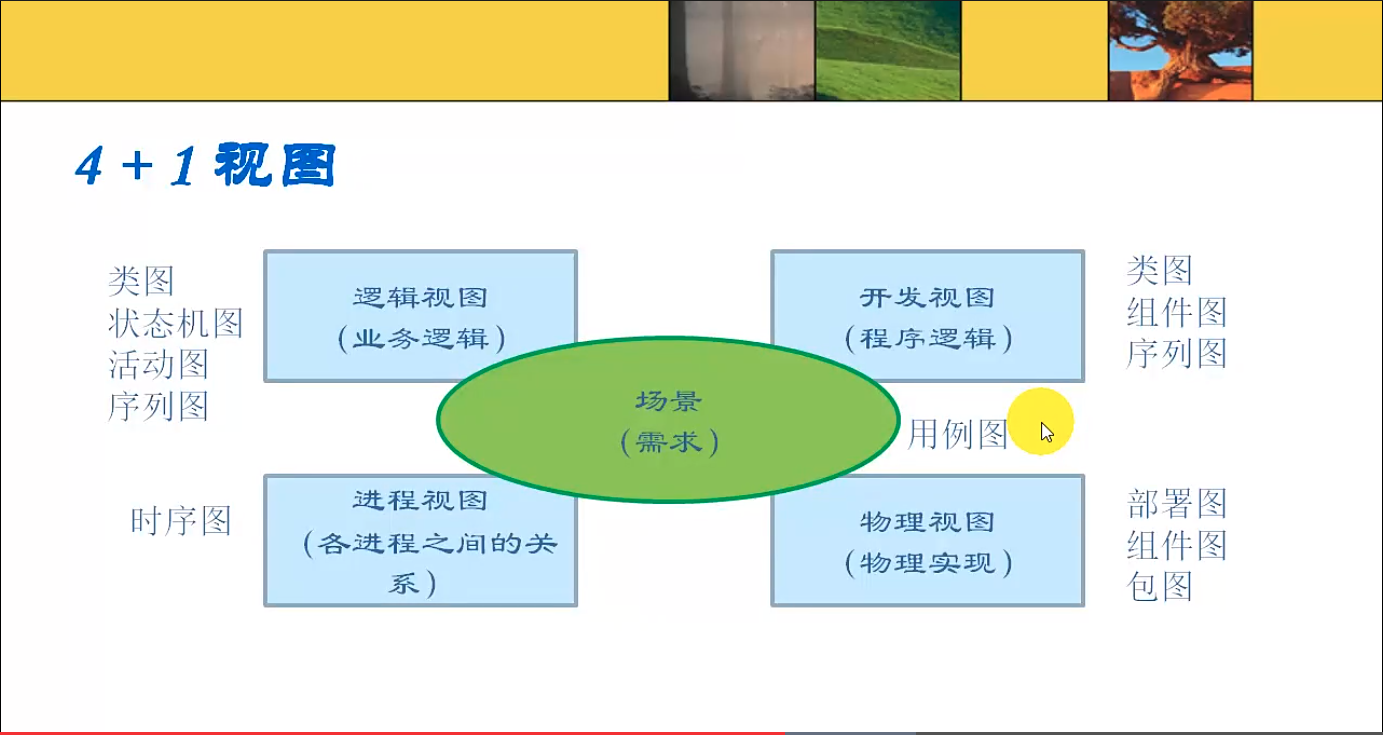
b) 工作方法落后

c) 自以为很OO(ObjectOriented),但没有展示OO的载体

d) 需求与开发环节脱节，无法做到‘需求驱动设计’

e) 难以帮助公司实现利润最大化。利润 = 需求-设计，需求，就是软件的价值，设计，就是软件的成本。

## 4+1视图

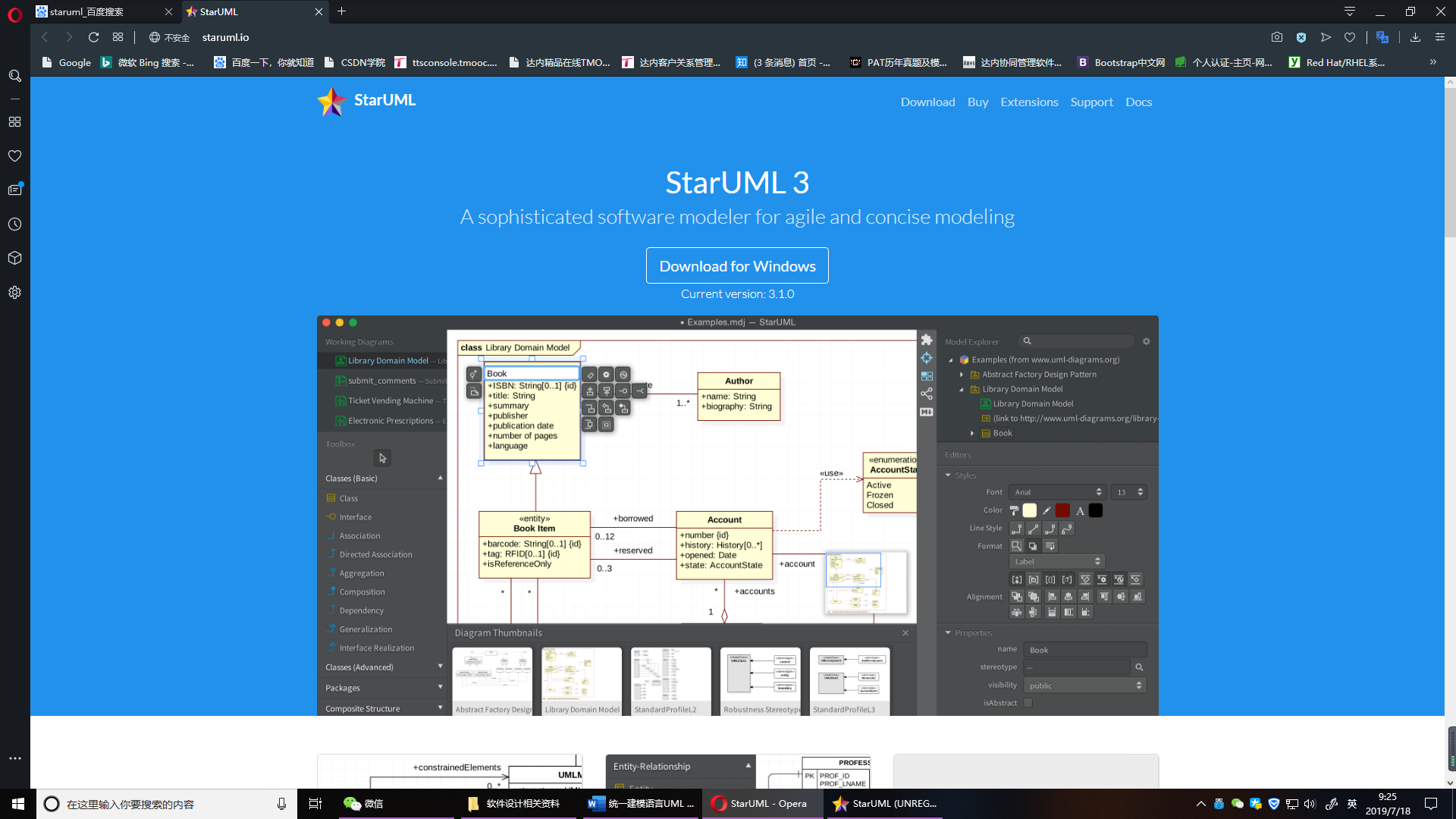


## UML制作工具

### 工具名称

starUML

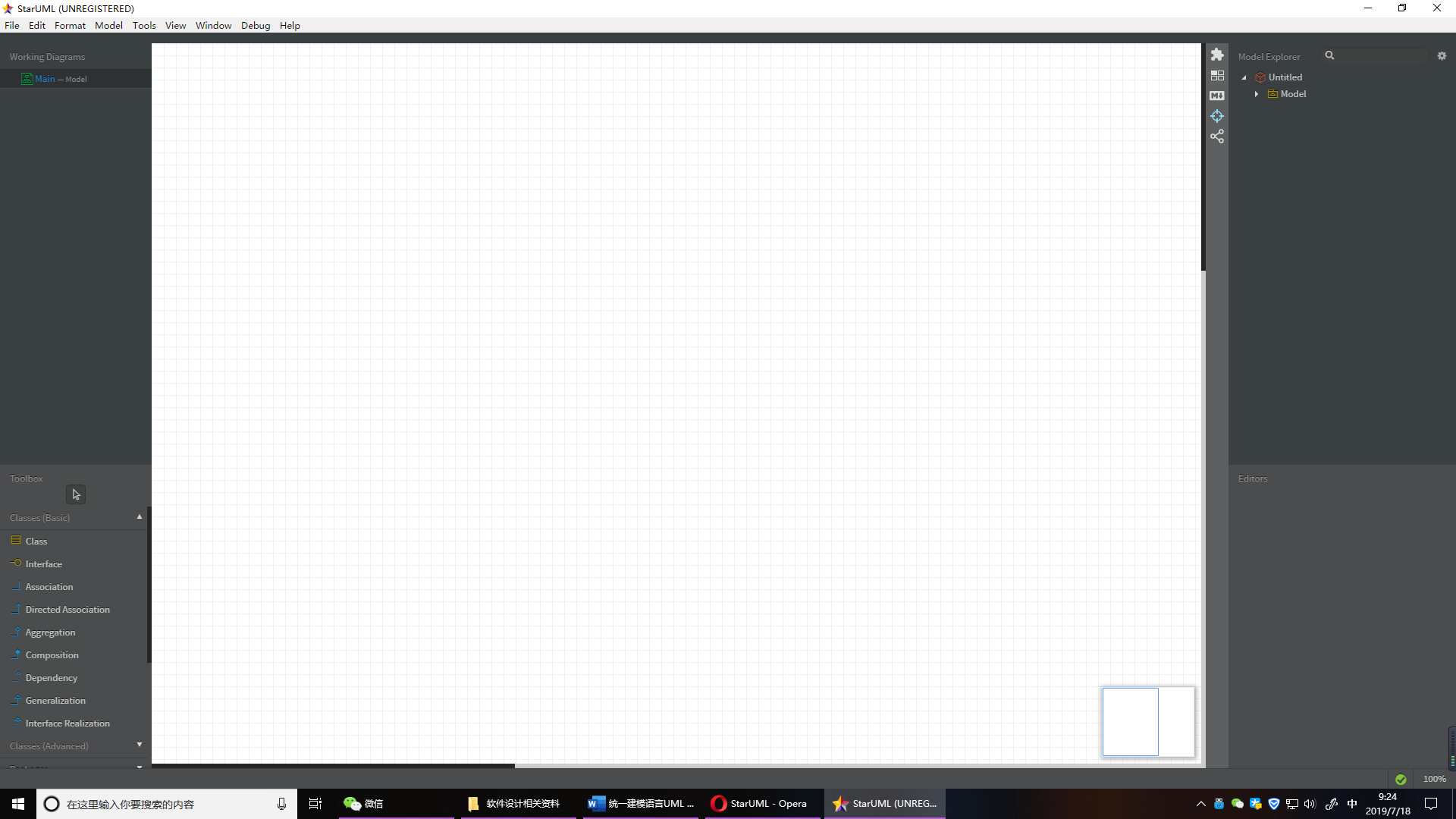
### 官网截图



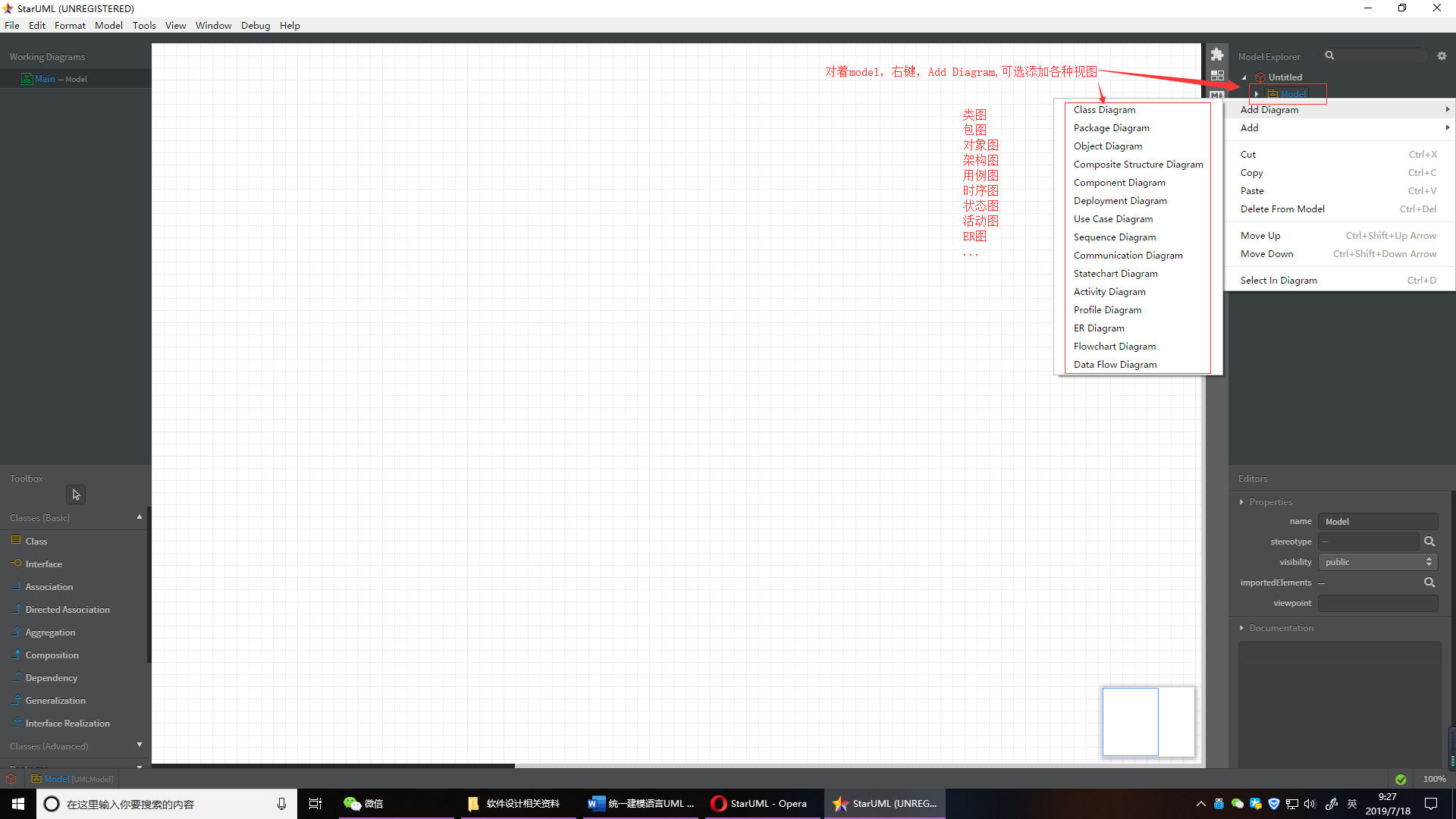
### 下载链接

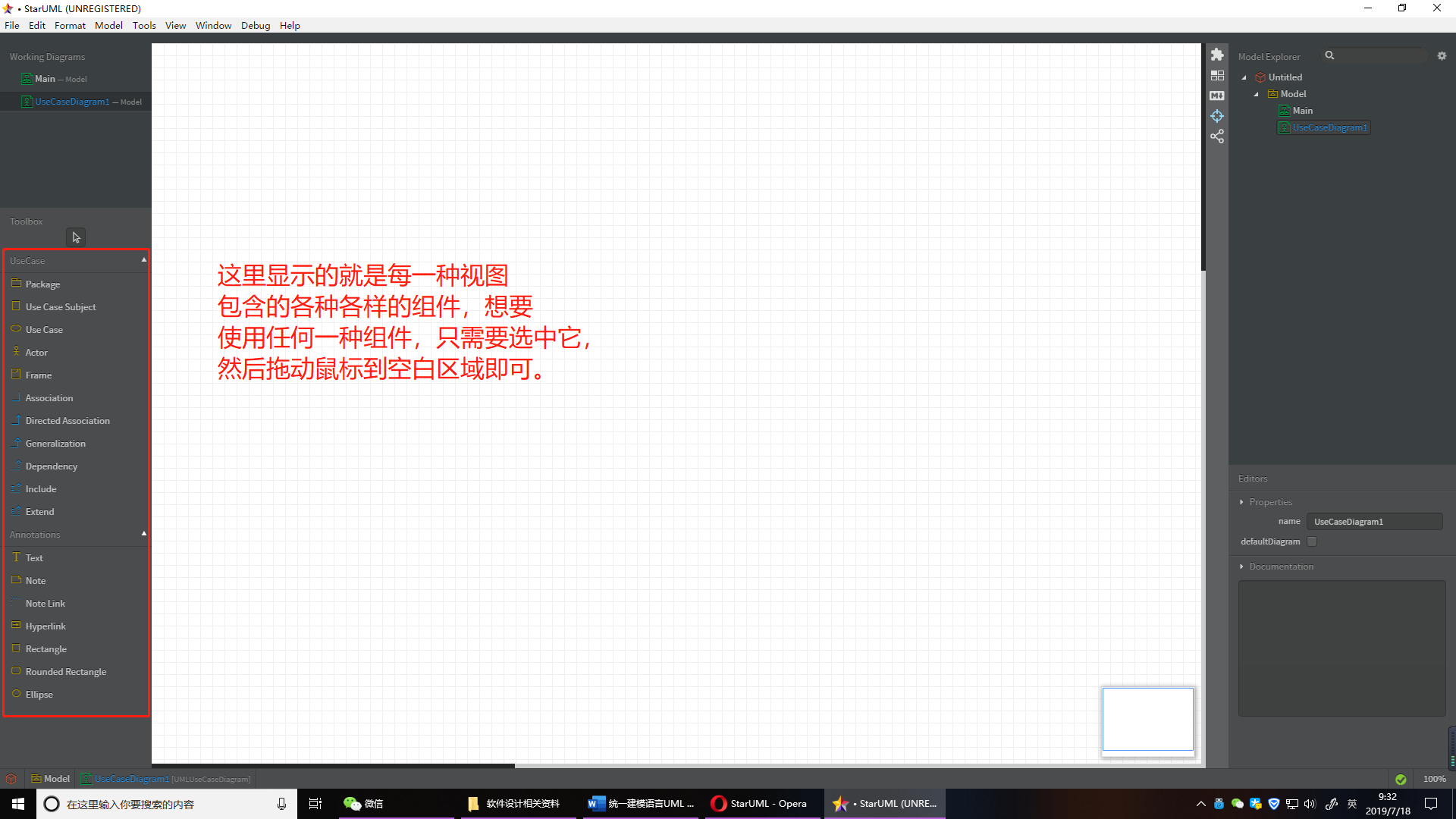
<http://staruml.io>

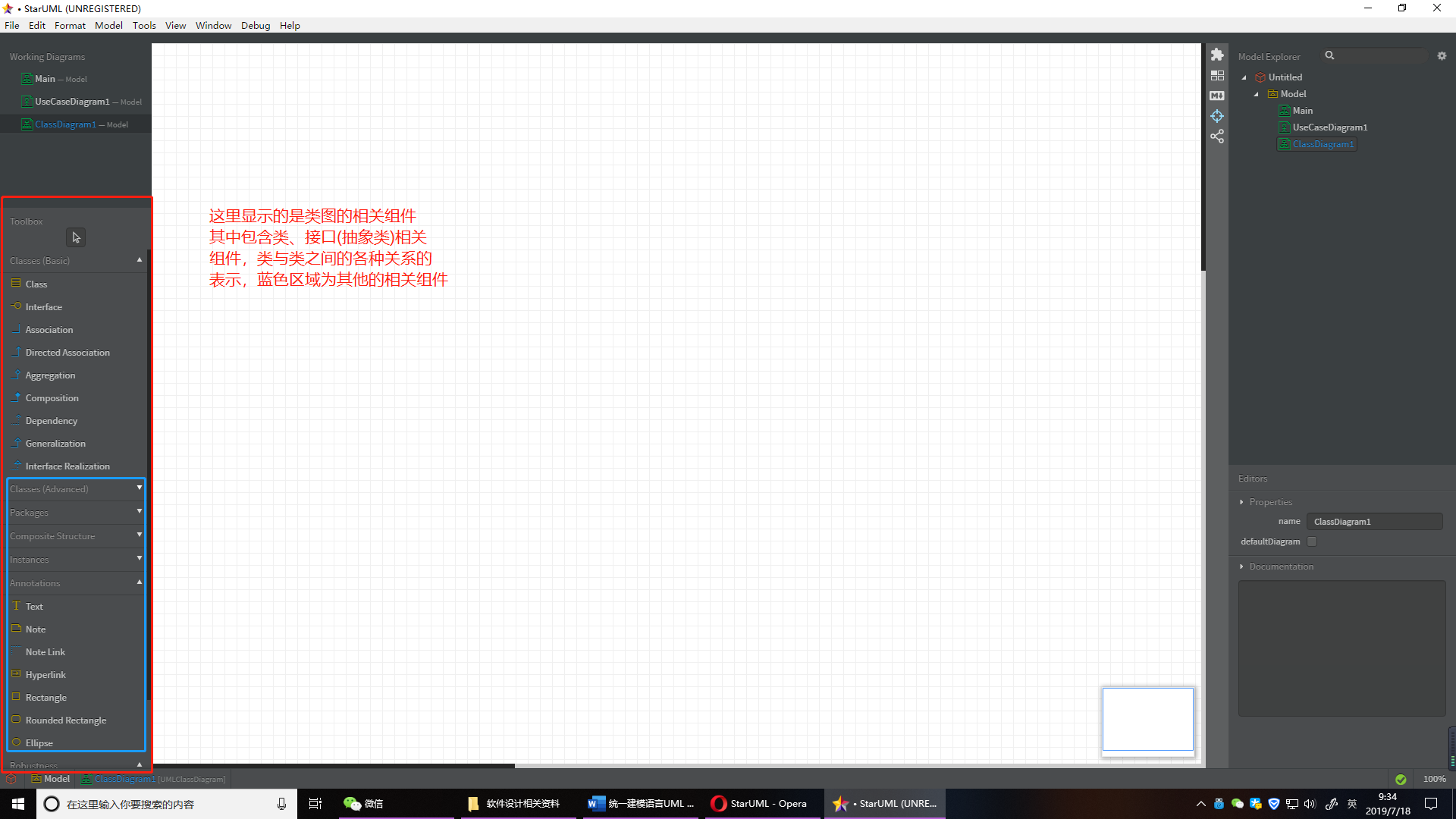
### 软件截图

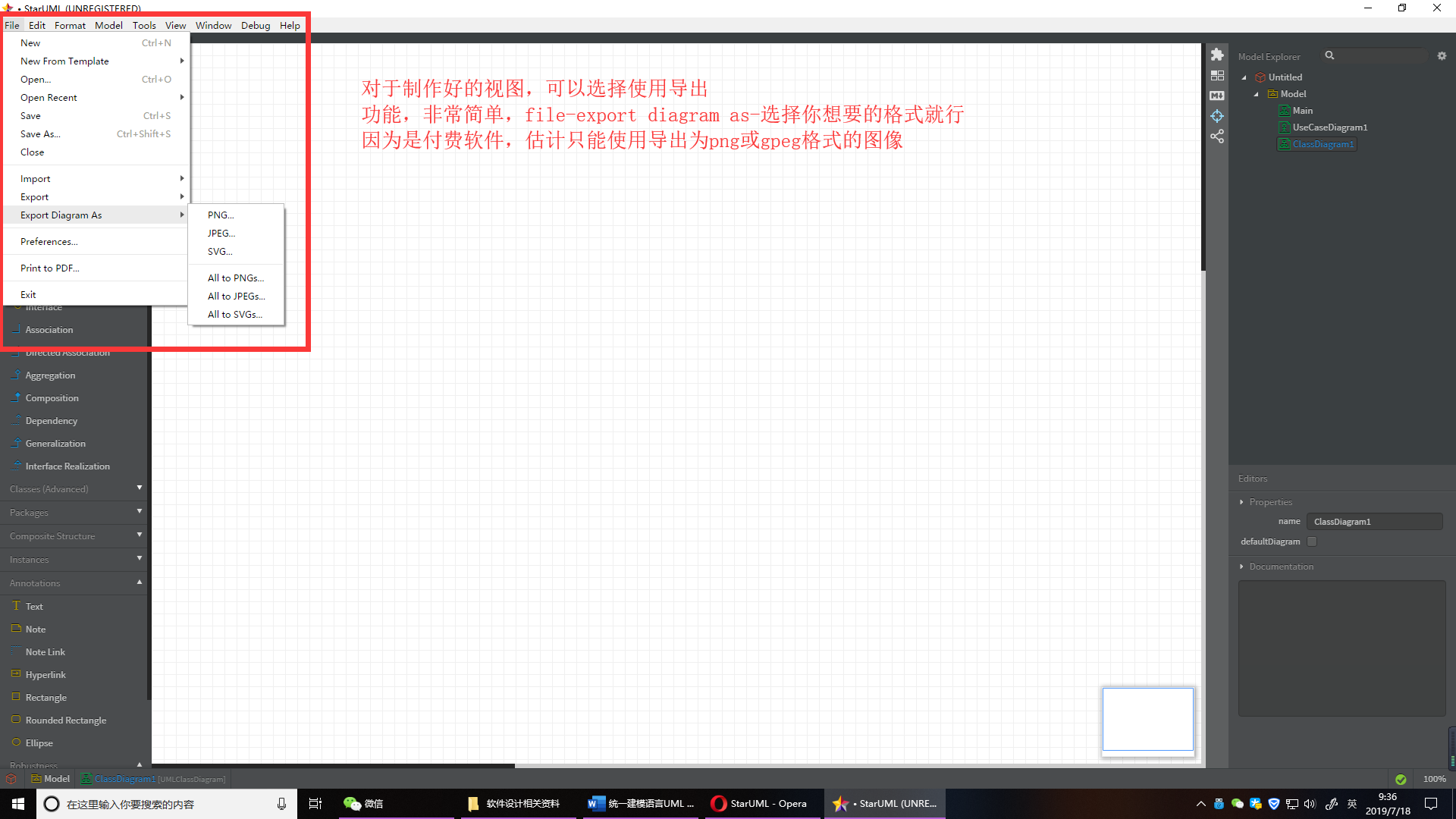


### 软件使用教程









# UML在需求分析中的应用

## 基本需求分析——用例图

### 什么是用例图

用例图是指由参与者（Actor）、[用例](https://baike.baidu.com/item/%E7%94%A8%E4%BE%8B/163511)（Use Case），边界以及它们之间的关系构成的用于描述系统功能的视图。用例图（User Case）是外部用户（被称为参与者）所能观察到的系统功能的模型图。用例图是系统的蓝图。用例图呈现了一些参与者，一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行[建模](https://baike.baidu.com/item/%E5%BB%BA%E6%A8%A1/814831)。

### 用例图的关键要素

1. 系统边界
2. 执行者(或者叫参与者)
3. 用例
4. Include：包含关系，即一个大的用例可能由若干个小用例具体组成

### 第一次使用用例图时常见的问题

1.这个系统只有一种用户，或者只有用户和管理员，再没别人了。

2.不能合适的拆分和组织用例

以上问题出现的本质：

1.未能切换思维方式和角度，应该从用户的角度思考问题

2.未能从用户的角度来拆分和组织用力

3.过多的从工程师和技术实现的角度来思考。

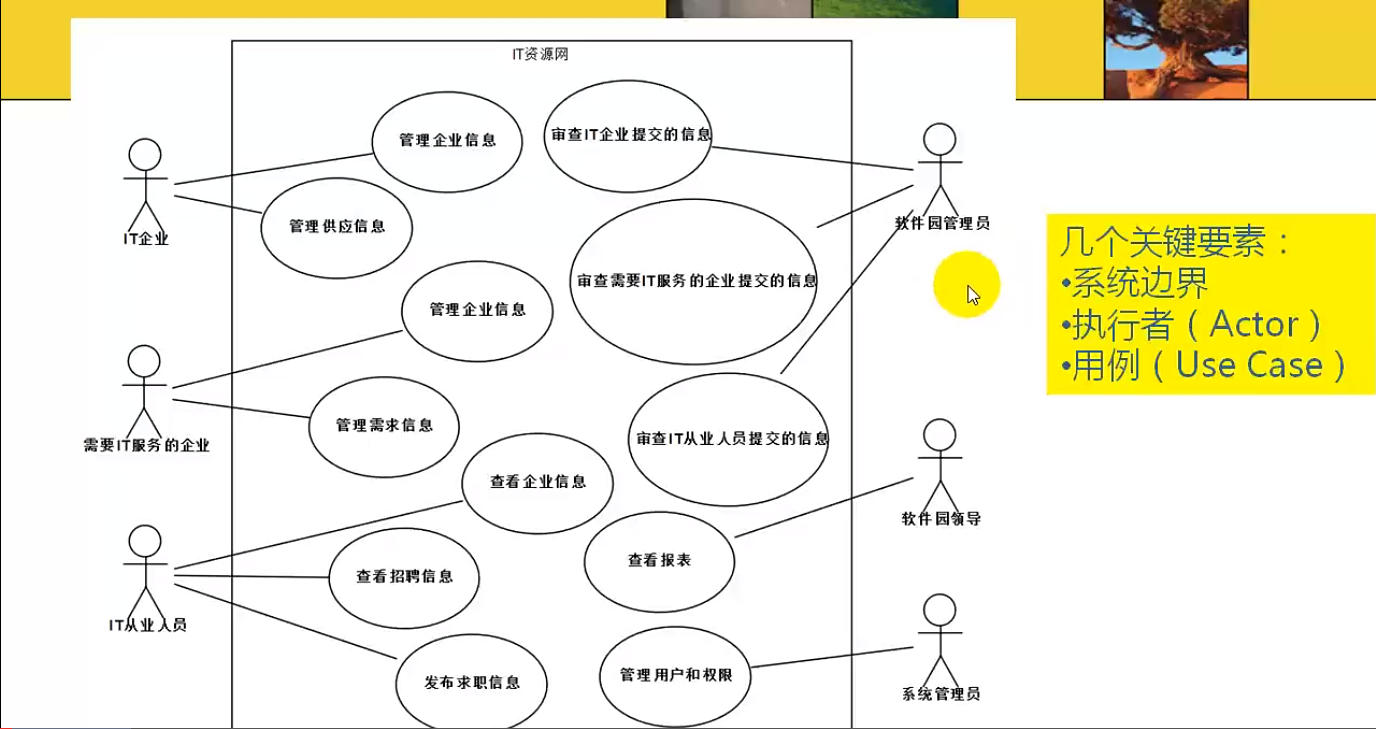
### 使用用例图需要注意的

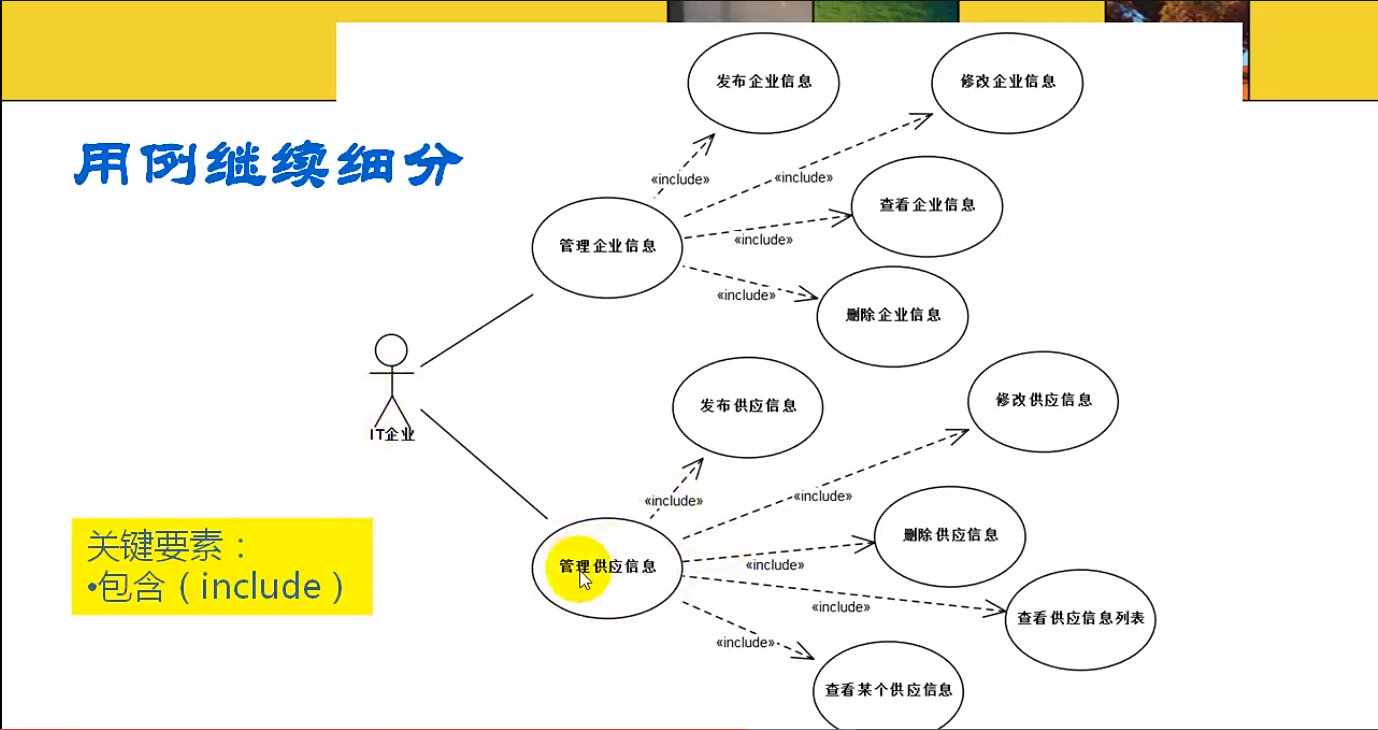
1.思考系统为谁服务

2.系统应该为这些‘谁’提供怎样的用例

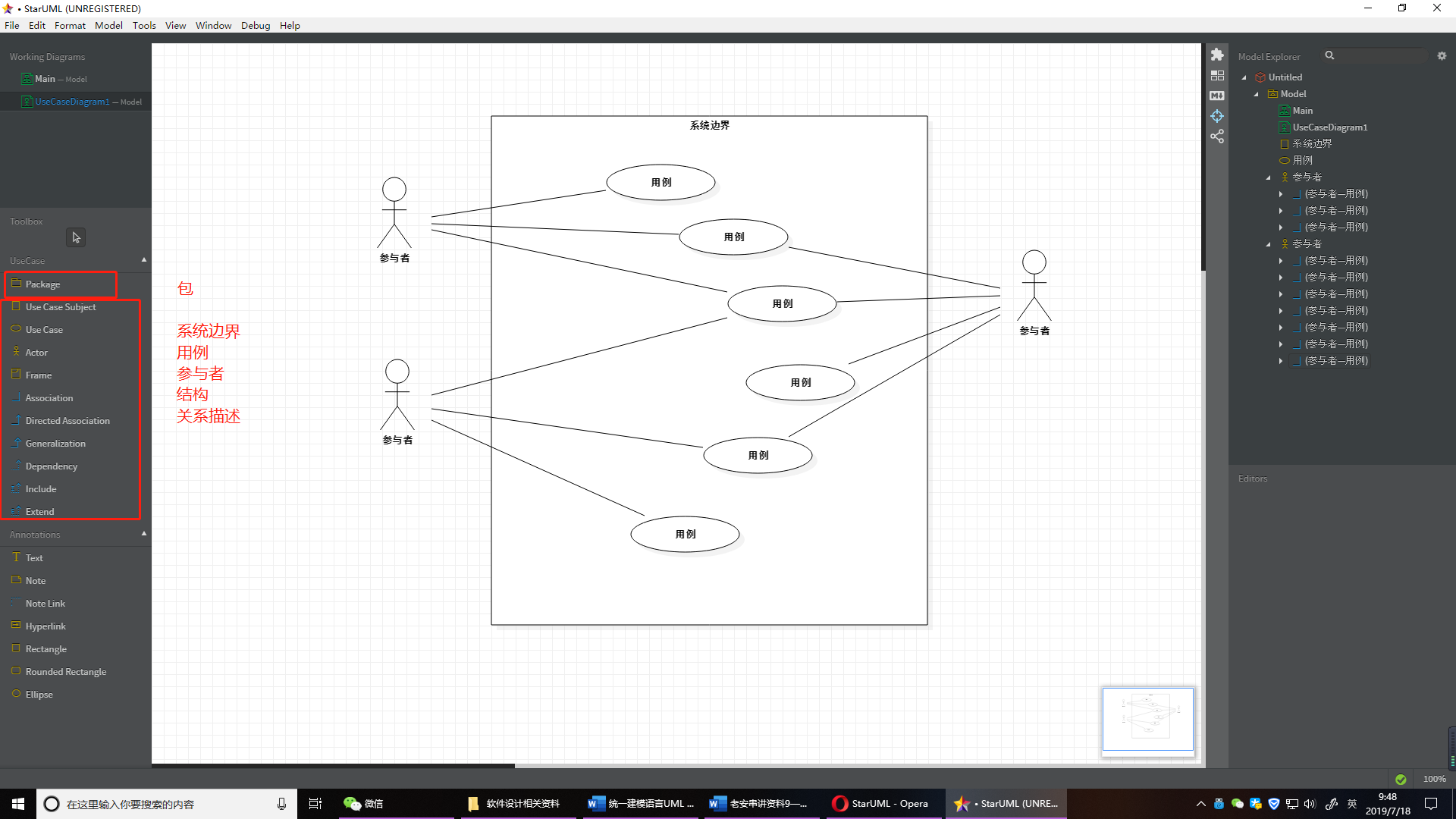
3.这些用例能为这些‘谁’带来怎样的价值

### 用例图图示举例





### 使用starUML画用例图



# 四、UML在概要设计中的应用

## 1.业务流程分析——活动图、状态机图、时序图

### 1.活动图

#### 什么是活动图

活动图（Activity Diagram）是阐明了业务用例实现的工作流程。是状态图的一种特殊形式。是UML用于对系统的动态行为建模的一中常用工具，描述活动的顺序，展现从一个活动到另一个活动的控制流。

活动图在本质上是一种流程图，着重表现从一个活动到另一个活动的控制流，是内部处理驱动的流程。

活动图侧重从行为的动作描述

#### 活动图的关键要素

（1）起点：用于便是活动图中所有活动的起点，有且只有一个

（2）终点：用于表示活动图中活动的终点，可以有多个

（3）活动名称：活动图中要描述的一个动作

（4）判断条件：活动流程出现的分支与决策

（5）同步条：活动之间的同步，由一个或多个信息流引入或引出

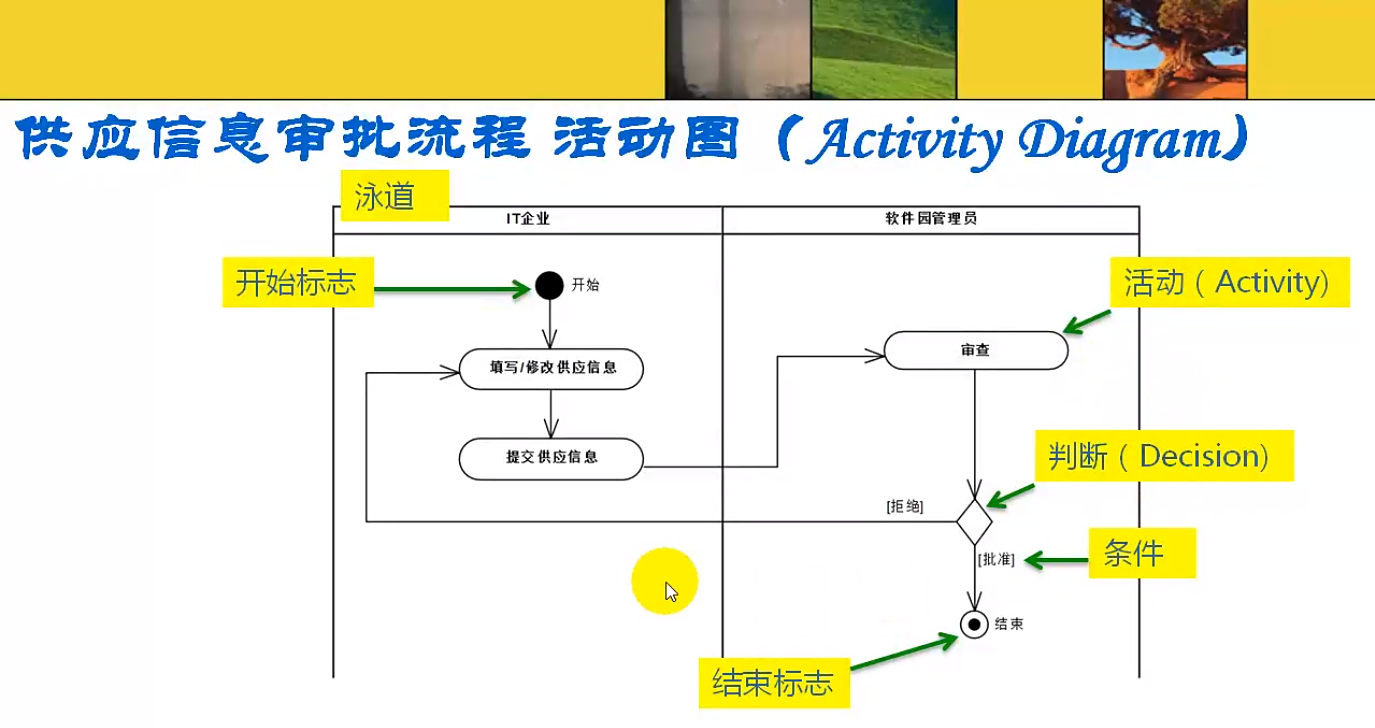
（6）接收信号：从外部过程接收一个信号，并执行相应的活动

（7）发送信号：向外部过程发送一个信号

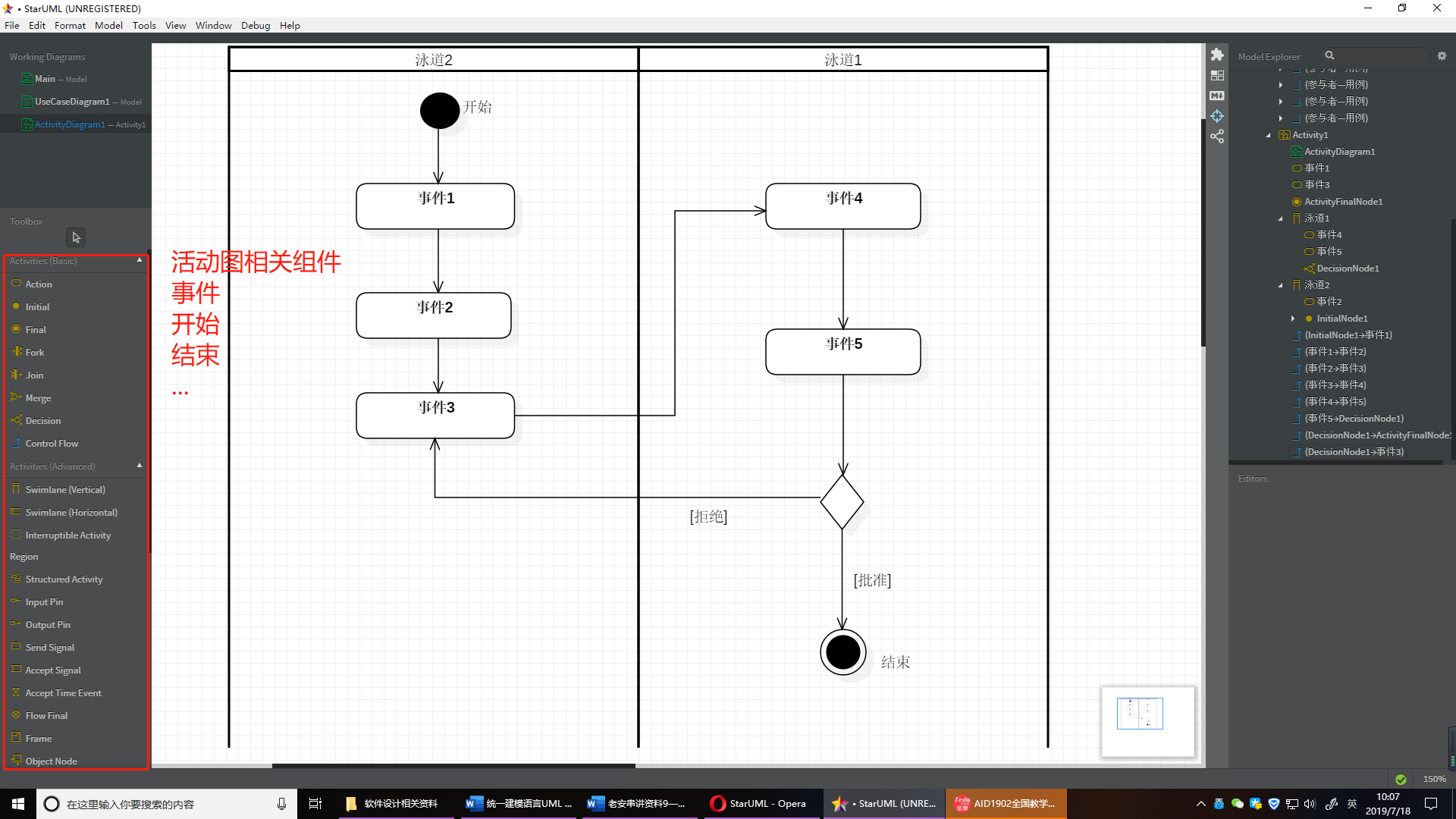
（8）泳道：用于对活动按照对象进行分组，用于描述对象之间的合作关系

（9）转移：用于表示活动之间的转移，连接活动、七点、终点、同步条、判断等

#### 活动图图示举例



#### 使用starUML画活动图



### 2.状态机图

#### 什么是状态机图

状态机图描述的是围绕某一事物状态变化的图。它也是三大流程分析利器之一。它和活动图的区别在于，活动图是描述事物发生的流程，是多个角色参与的，而状态机描述的是事物的状态变化，并没有角色这个概念

#### 状态机图的关键要素

1.开始状态

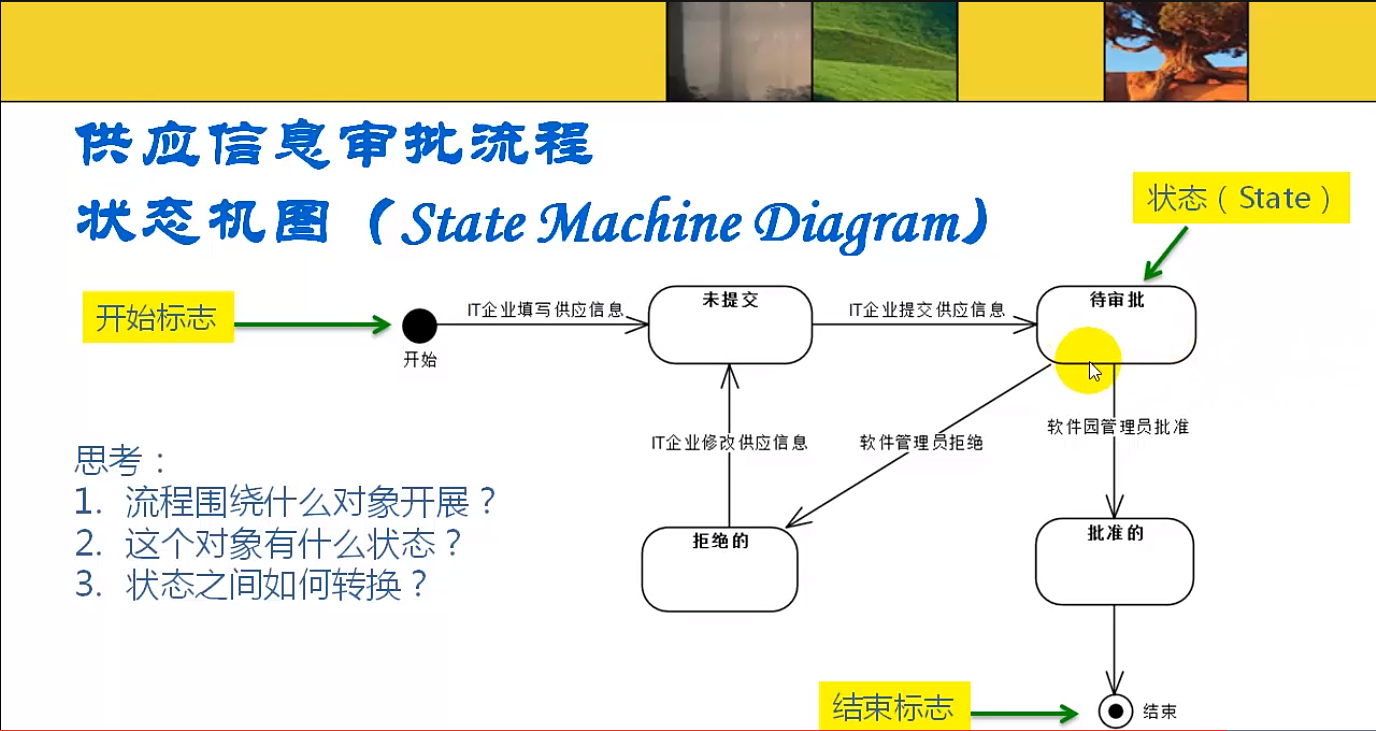
2.结束状态

3.转换

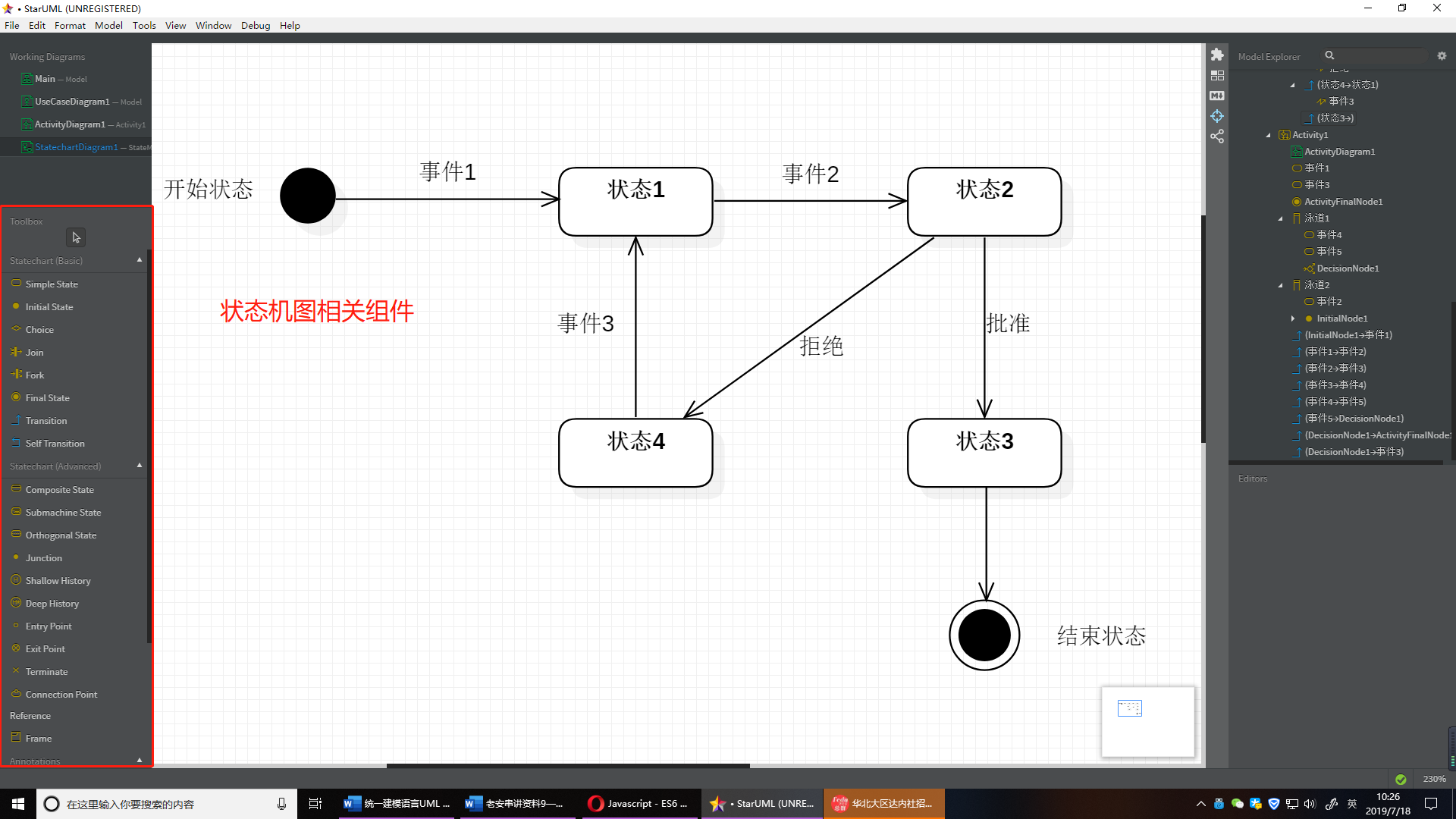
4.状态

5.监护

#### 状态机图图示实例



#### 使用starUML画状态机图



### 3.时序图

#### 什么是时序图

序列图，亦称时序图，它通过描述对象之间发送消息的时间顺序显示多个对象之间的动态协作。它可以表示[用例](https://baike.baidu.com/item/%E7%94%A8%E4%BE%8B)的行为顺序，当执行一个用例行为时，时序图中的每条消息对应了一个类操作或[状态机](https://baike.baidu.com/item/%E7%8A%B6%E6%80%81%E6%9C%BA)中引起转换的触发事件。

#### 时序图的关键要素

1.激活

2.对象生命线

3.参与者生命线

4.循环片段

5.可选片段

6.备用片段

7.交互操作数

8.其他片段

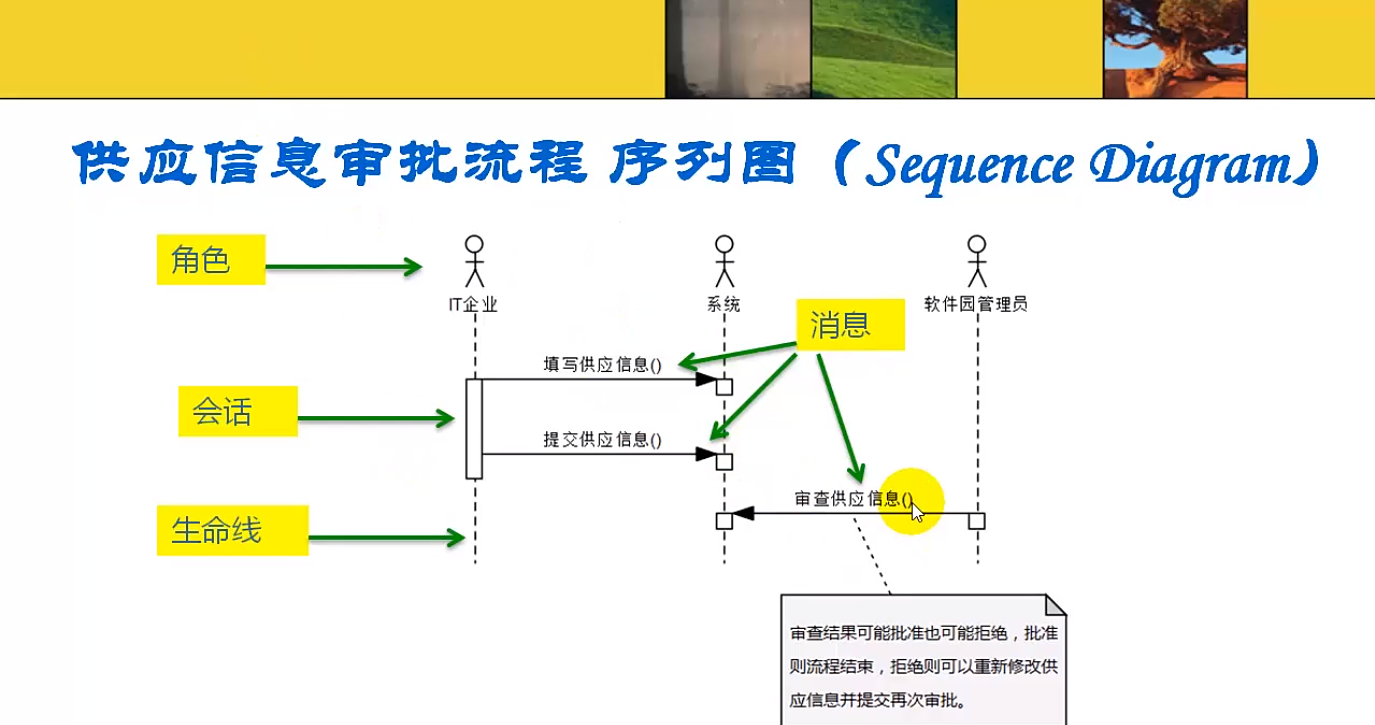
9.消息

10.返回消息

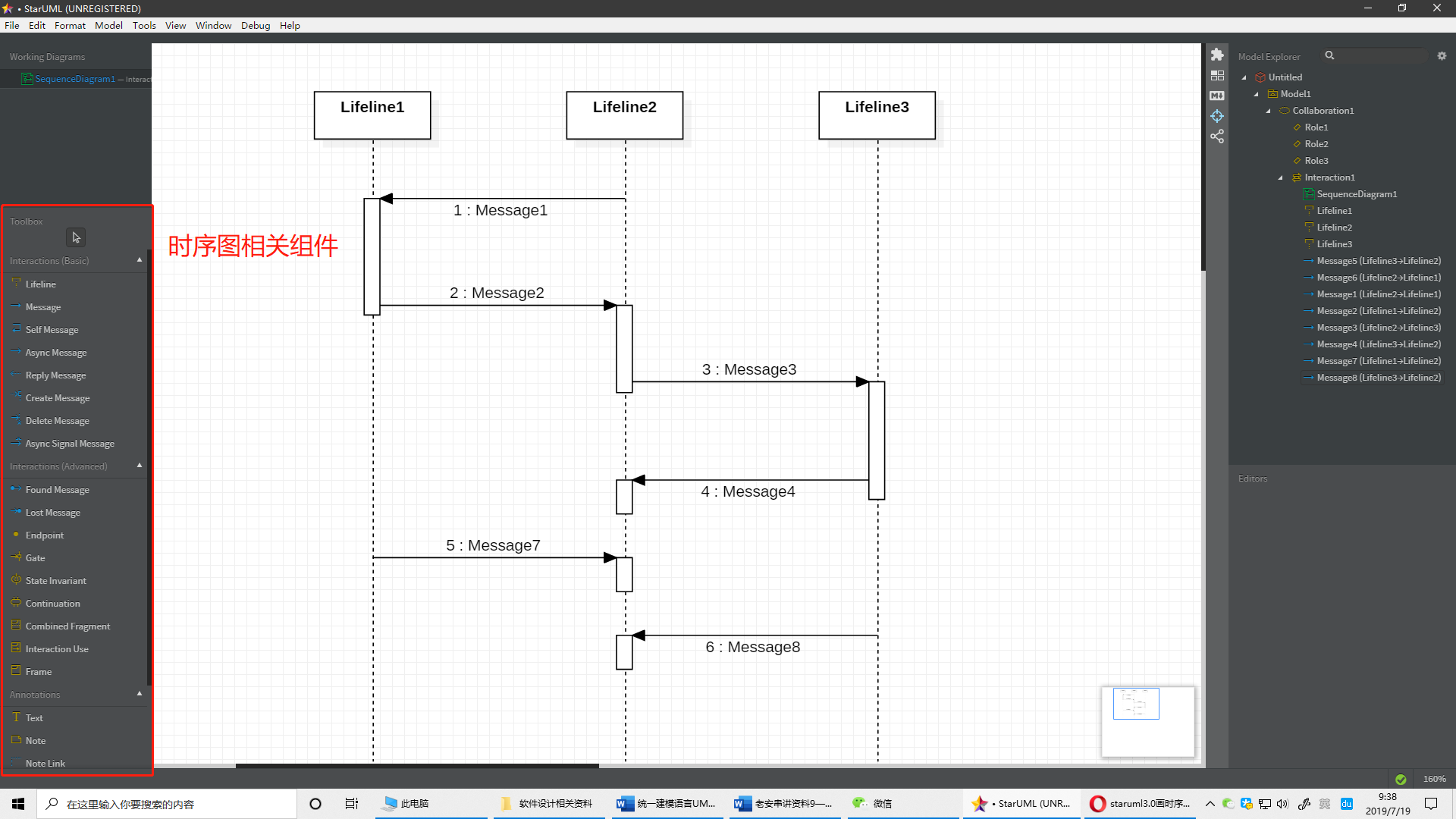
11.自关联消息

12.异步消息

#### 时序图图示举例



#### 使用starUML画时序图

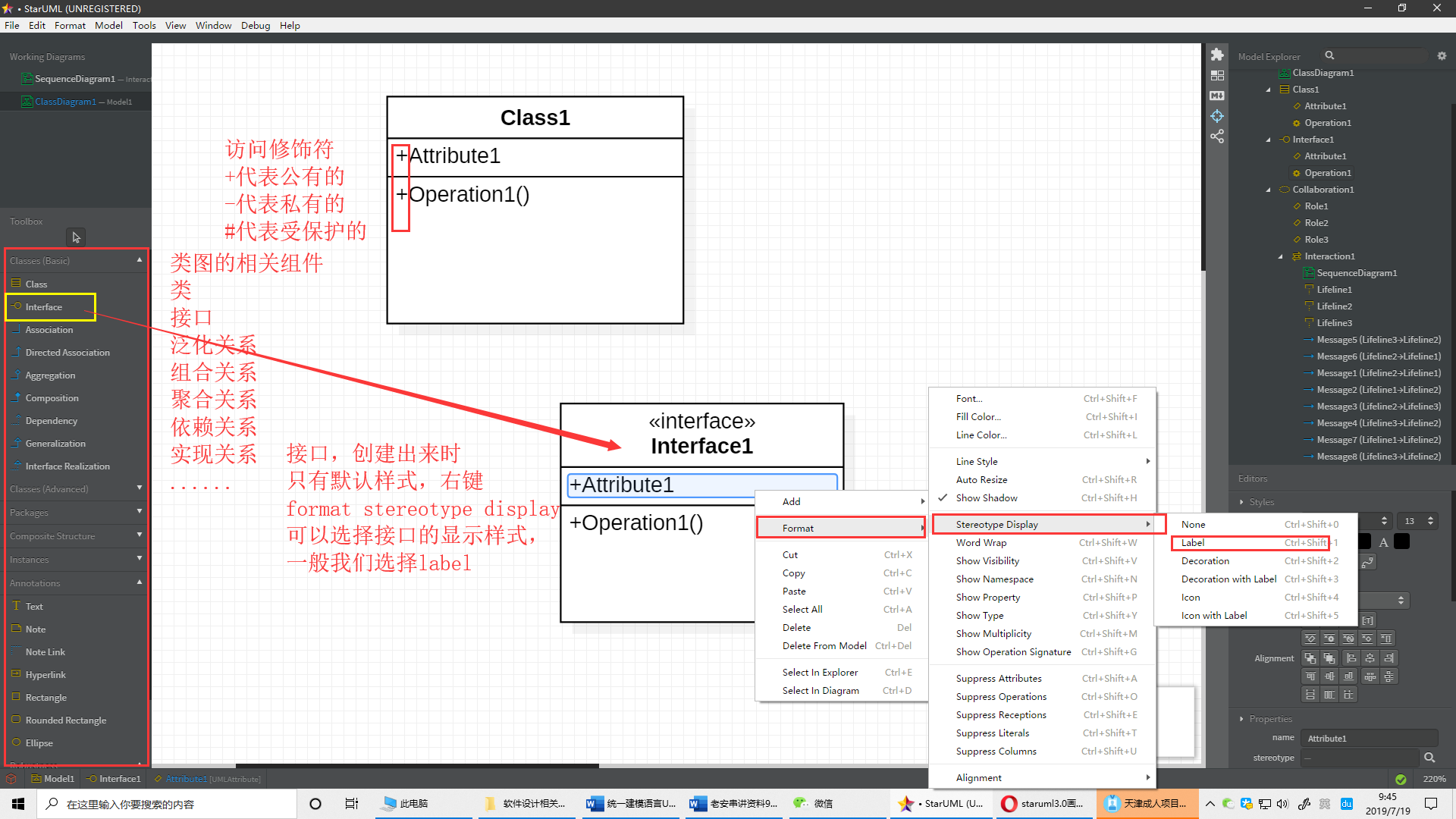


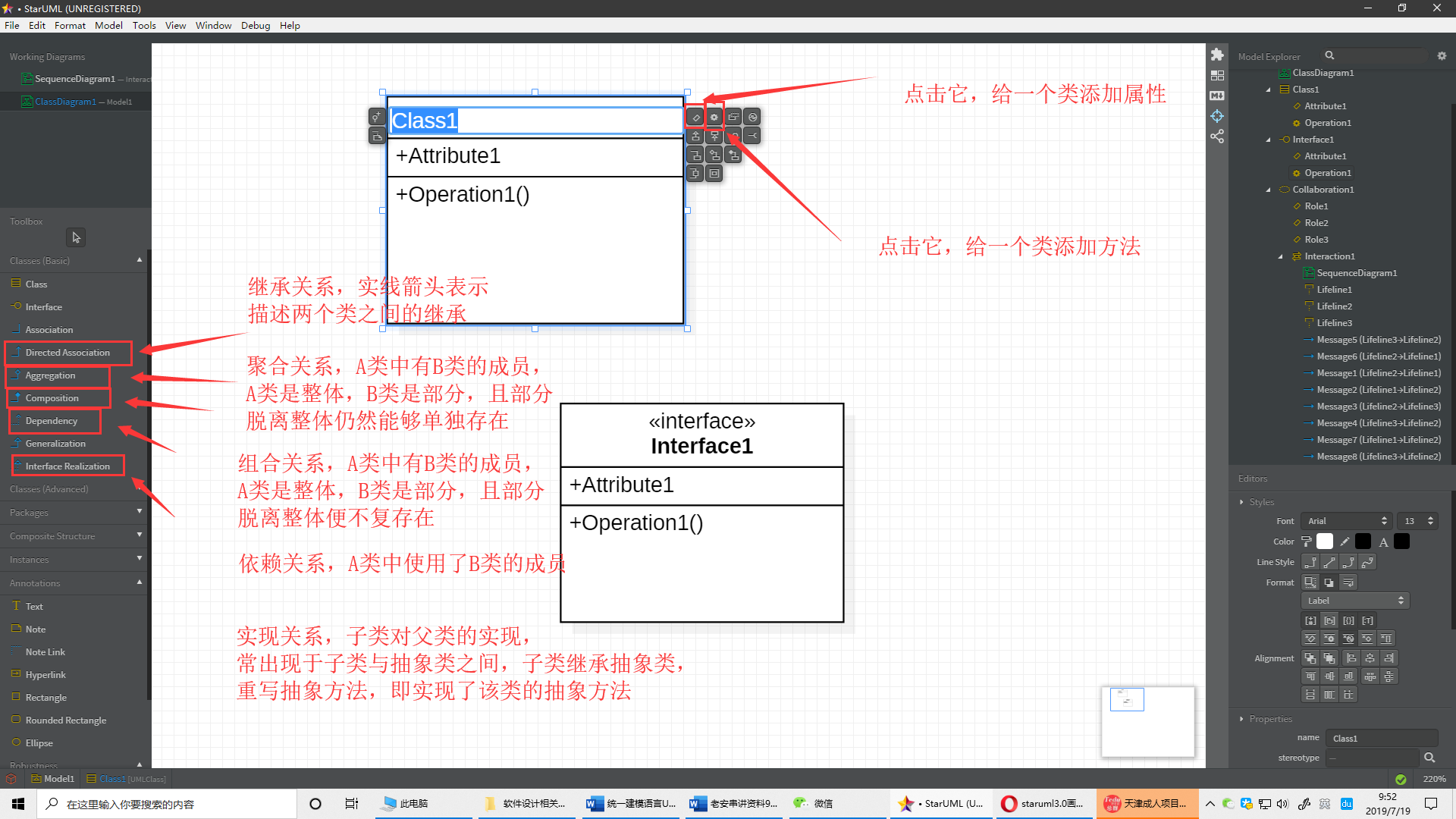
## 2.业务概念分析——类图

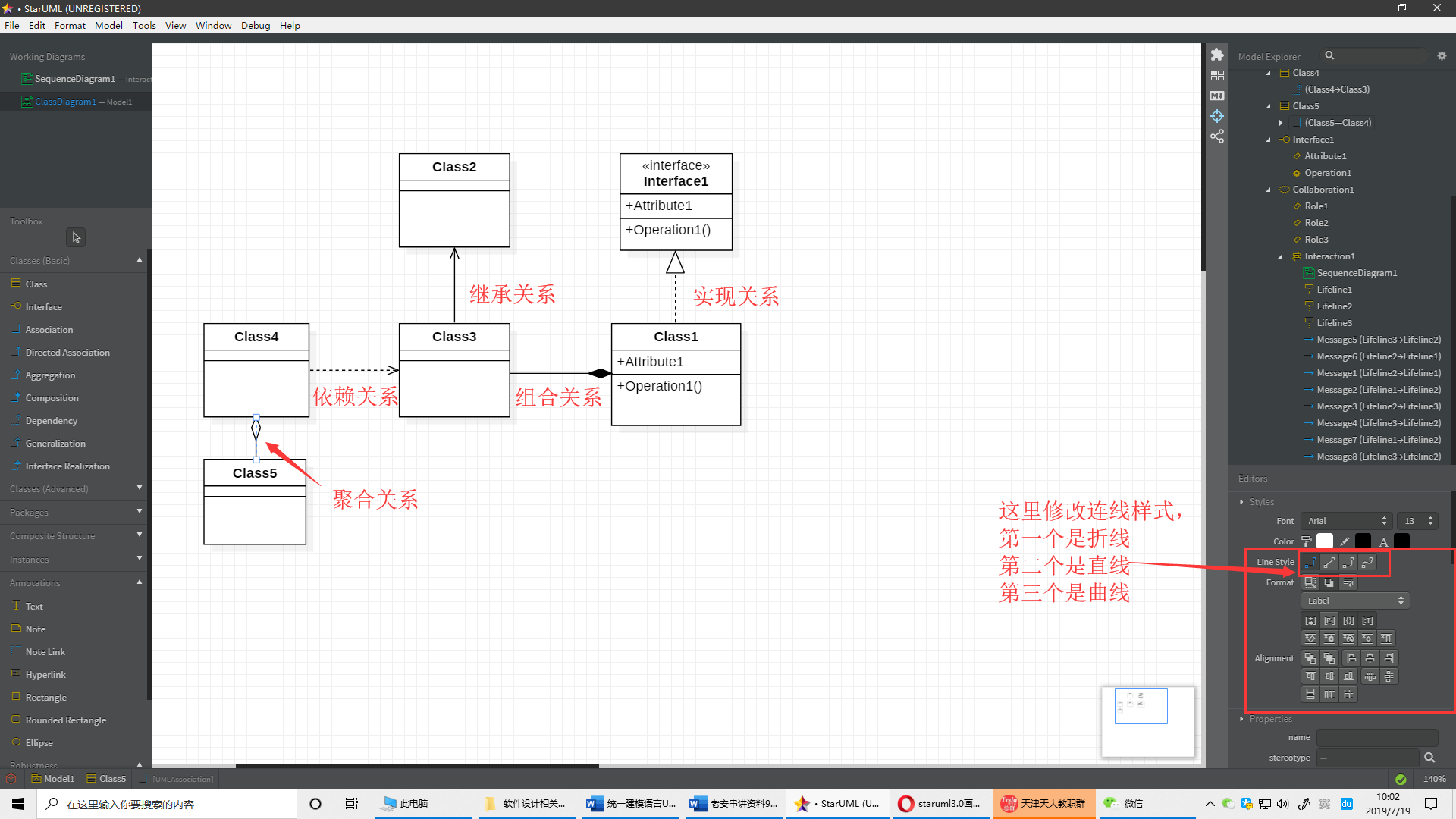
#### 什么是类图

类图(Class diagram)是显示了模型的静态结构，特别是模型中存在的类、类的内部结构以及它们与其他类的关系等。类图不显示暂时性的信息。类图是面向对象建模的主要组成部分。它既用于应用程序的系统分类及业务概念的一般概念建模，也用于详细建模，将模型转换成编程代码。类图也可用于数据建模。

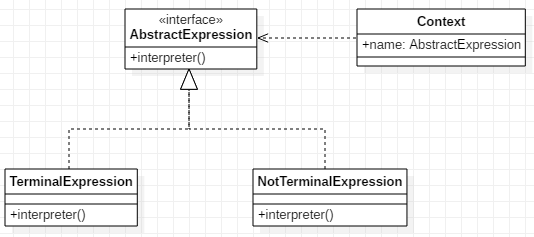
#### 类图的关键要素







#### 类图举例



## 3.软件架构设计——部署图、包图、组件图

### 部署图

#### 什么是部署图

部署图(deployment diagram，配置图)是用来显示系统中软件和硬件的物理架构。从部署图中，您可以了解到软件和硬件组件之间的物理关系以及处理节点的组件分布情况。使用部署图可以显示运行时系统的结构，同时还传达构成应用程序的硬件和软件元素的配置和部署方式。

#### 部署图的关键要素

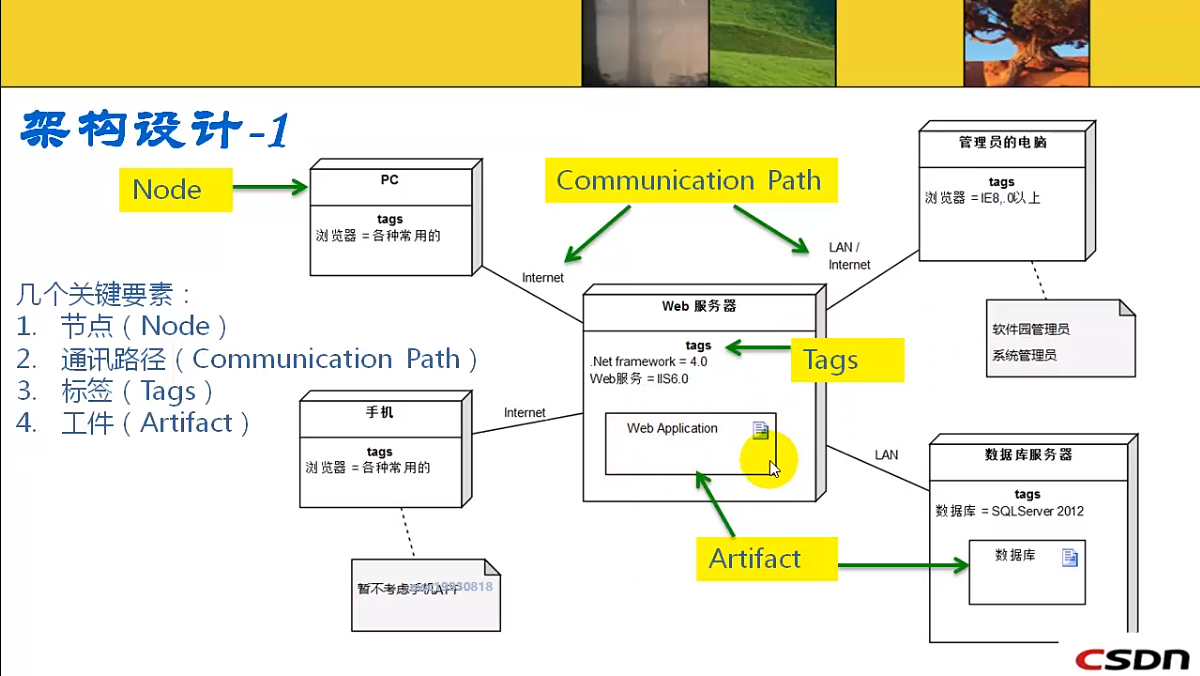
节点：代表某一种访问系统的终端

通讯路径：终端与服务器之间依靠什么通信

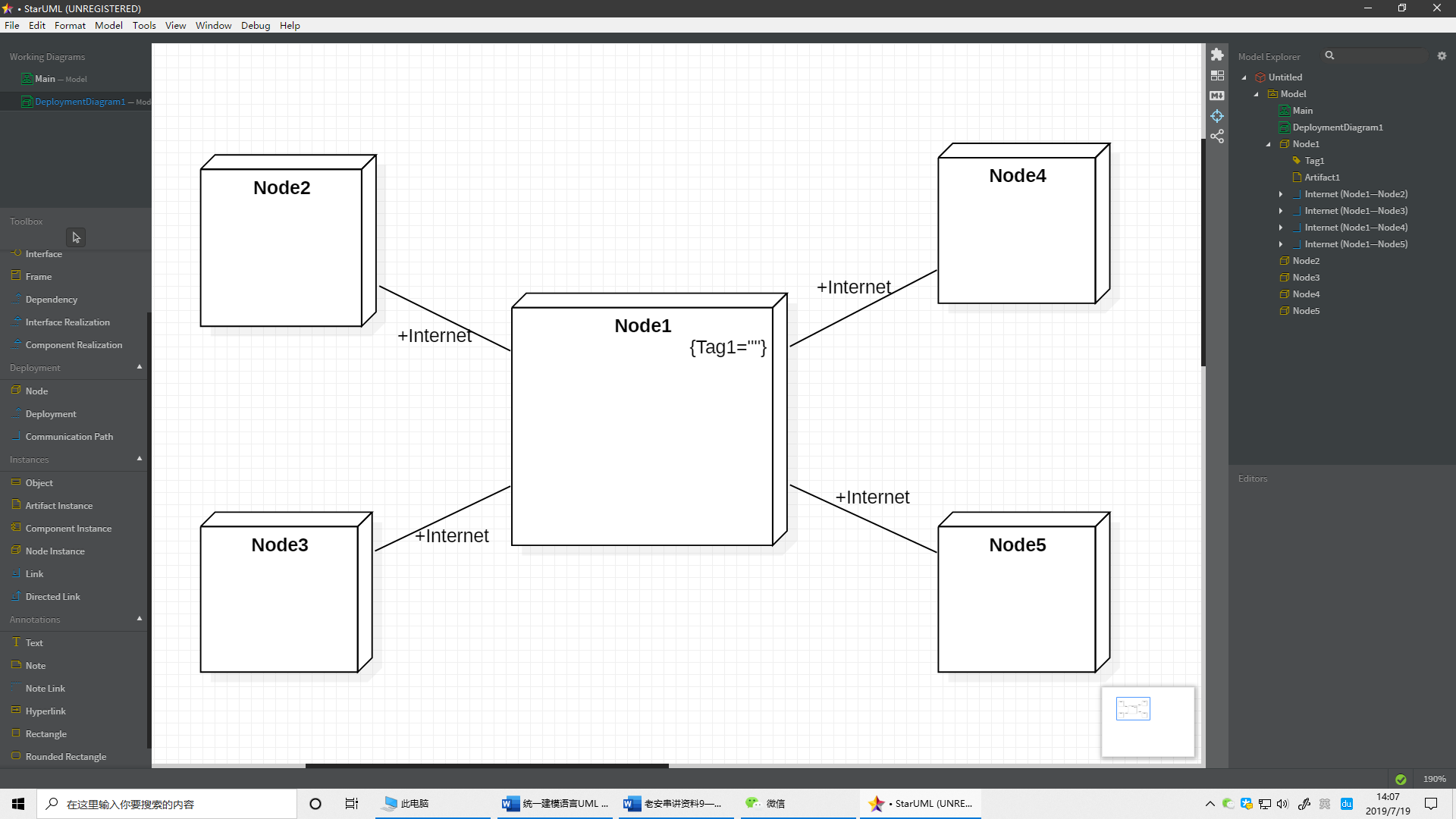
标签：各节点的一些标识，一些特点

工件：我们要开发与设计的软件。

#### 部署图图示举例



#### 使用starUML画部署图



### 包图、组件图

#### 什么是包图、组件图

包图：包图是在 UML 中用类似于文件夹的符号表示的模型元素的组合。系统中的每个元素都只能为一个包所有，一个包可嵌套在另一个包中。使用包图可以将相关元素归入一个系统。一个包中可包含附属包、图表或单个元素。

组件图：组件图(component diagram)是用来反映代码的[物理结构](https://baike.baidu.com/item/%E7%89%A9%E7%90%86%E7%BB%93%E6%9E%84/9663422)。从组件图中，您可以了解各[软件组件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%BB%84%E4%BB%B6/9817461)（如[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81/3814213)文件或动态链接库）之间的[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8/8853067)和运行时依赖关系。使用组件图可以将系统划分为[内聚](https://baike.baidu.com/item/%E5%86%85%E8%81%9A/9857344)组件并显示代码自身的结构。

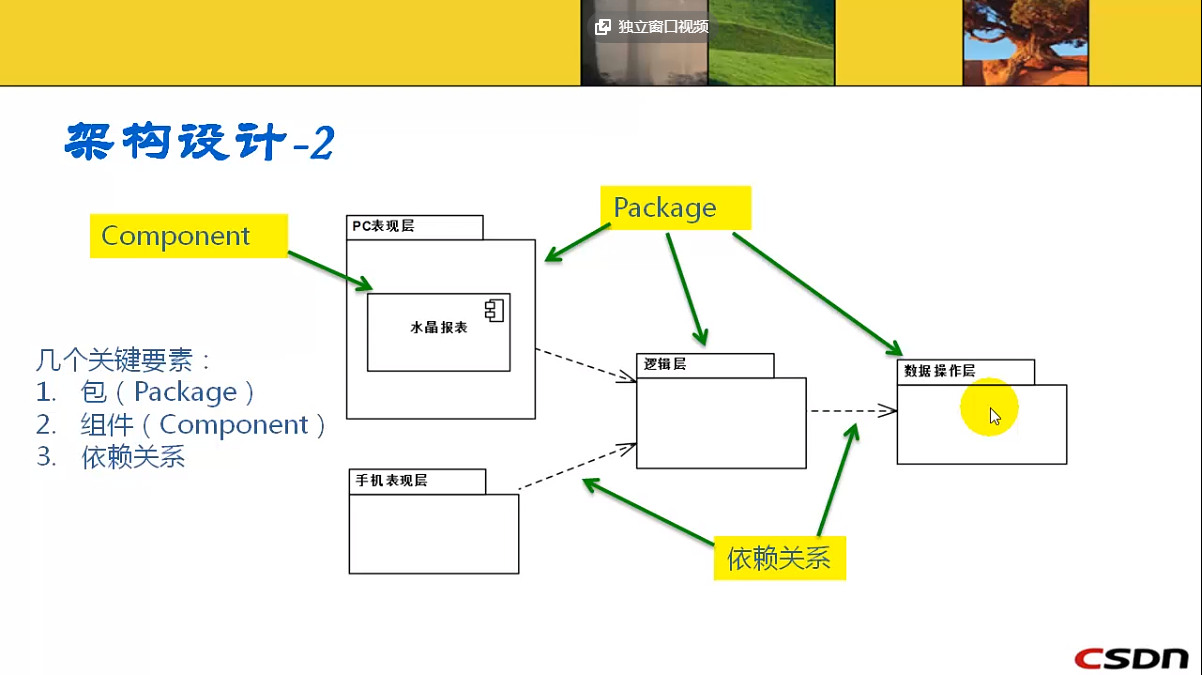
#### 包图、组件图的关键要素

包：一个逻辑的划分概念，类似模块

组件：物理上独立的，能提供一定服务的应用

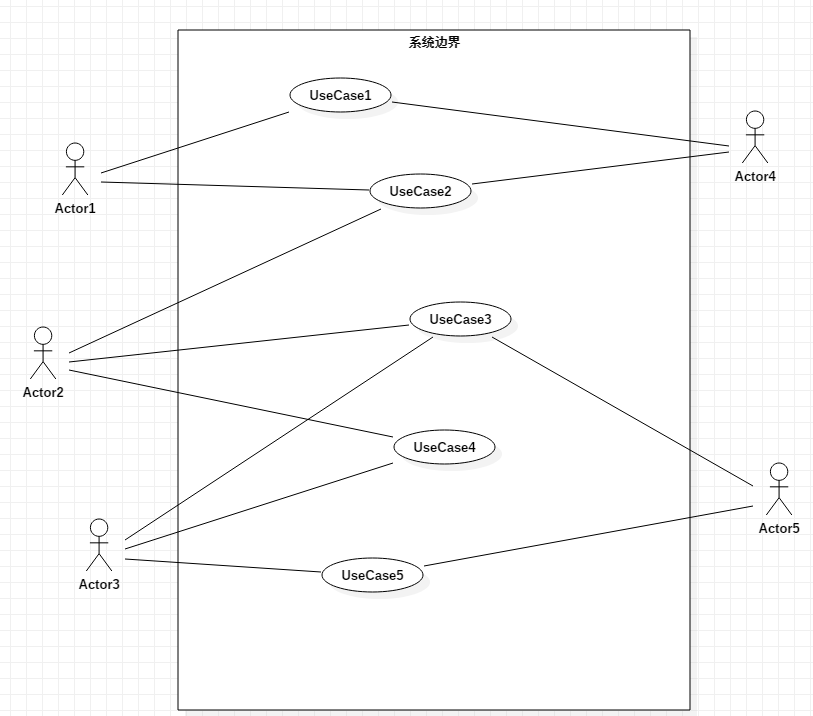
依赖关系：各层代码的实现需要依赖其他层的代码等。

#### 包图、组件图图示举例

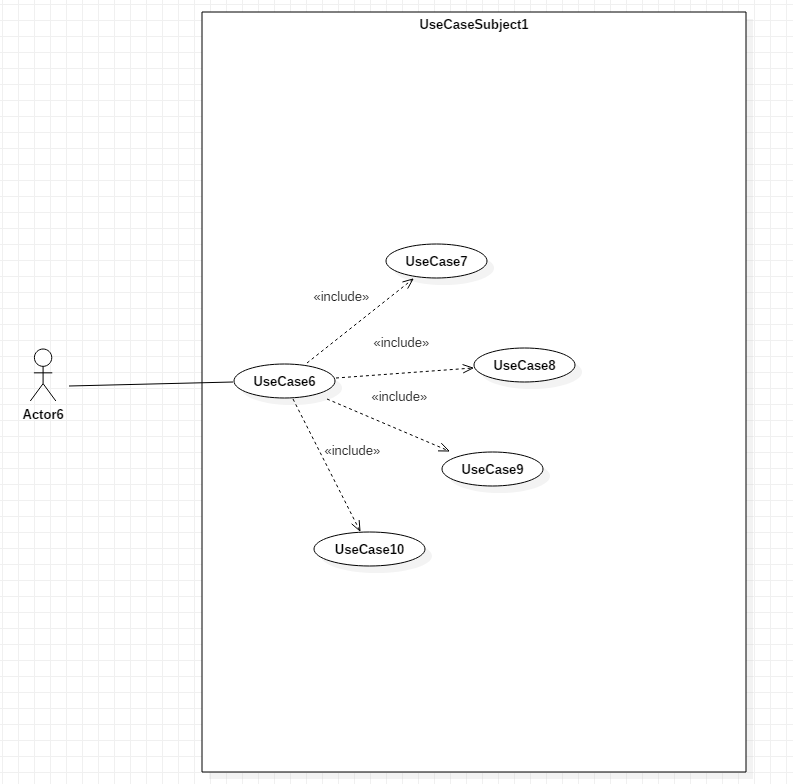


# 五、UML在详细设计中的应用——时序图、类图

## 详细设计定义



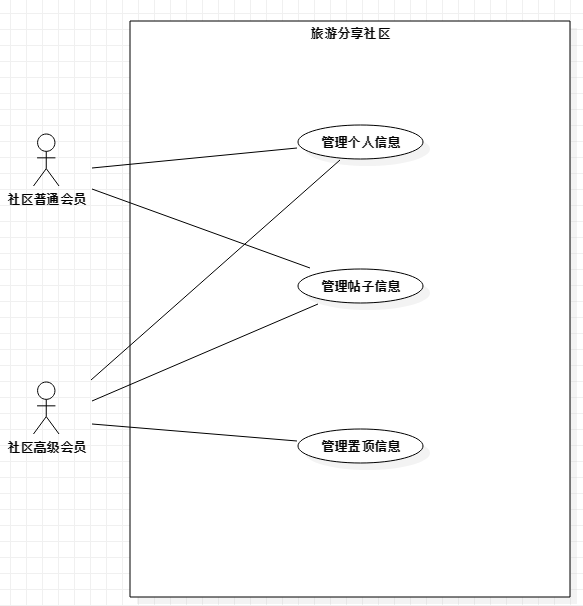
如图所示，通常，一个项目有几十个用例，所以，前期我们的基本需求分析过程中就要做出来我们项目的总的用例图。那么这个总的用例图其实就是一种宏观的概括。总的用例图中，肯定会有很多用例，那么我们就需要针对总用例图中的每一个用例进行拆解，如图所示：



将每一个大用例拆解为多个子用例图，根据拆解出来的每一个子用例图，我们分别使用时序图来表示这个用例工作的流程是什么，通过类图来表示这个用例中类与类之间的逻辑关系，这个完整的过程就叫做详细设计。

## 详细设计简易流程举例

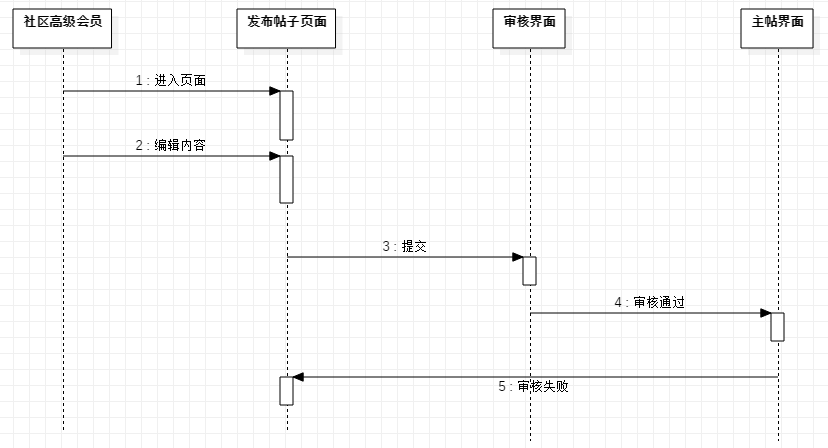
对于一个旅游分享社区，社区会员是其中一类参与者，其具备以下用例：



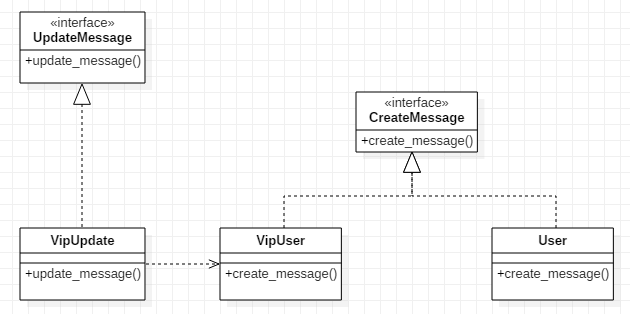
对于管理帖子信息，我们可以继续拆解：



对于拆解出来的发布帖子功能，我们可以用时序图来描述其实现流程：



用类图描述该用例中类与类之间的关系(简易举例)：



基于以上，其实就很好理解了：详细设计阶段各个建模图的设计都是基于基本的用例的，每一个用例，对应一个时序图，一个类图。

详细设计中最难的部分，即需要考虑到设计模式。一个好的设计模式，即在正确的时间用在正确的地方，会大大简化你的代码的架构设计，进而达到提升性能等作用。通过我们对于详细设计的后期，即需要针对每一个用例进行类图或对象图的详细设计过程中，我们就需要考虑到设计模式。用一个合适的设计模式，表示当前用例的合理的实现流程，合理的代码架构。又因为设计模式分为创建型，结构型，以及行为型模式，所以又需要针对每一个用例的独特的应用场景，采取对应类型的合适的设计模式。

# 总结

一个完整的软件设计流程大致是以下的步骤：

1. 需求分析：此阶段使用用例图进行建模
2. 概要设计
3. 行为建模：业务流程分析，此阶段使用活动图、状态图、时序图建模
4. 结构建模：业务概念分析，使用简易的类图进行建模
5. 架构建模：物理设计，使用部署图、包图、组件图进行建模
6. 详细设计

针对前期分析出来的每一个用例，使用时序图和类图建模，描述其执行流程以及用例中类与类之间的关系

以上所有的建模阶段，都是需要基于用例的。也就是说，前期的需求分析，至关重要。需求分析的到不到位，直接决定后续任何一步的设计工作。所以，一个良好的，优秀的设计，应该是遵循“需求驱动设计”这条准线的。

而在完整的设计过程中，除了需要考虑设计模式，又需要考虑采用什么样的数据结构，所以，讲到这里，各位应该真正理解了，为什么数据结构，软件设计，设计模式等编程理论很重要了。

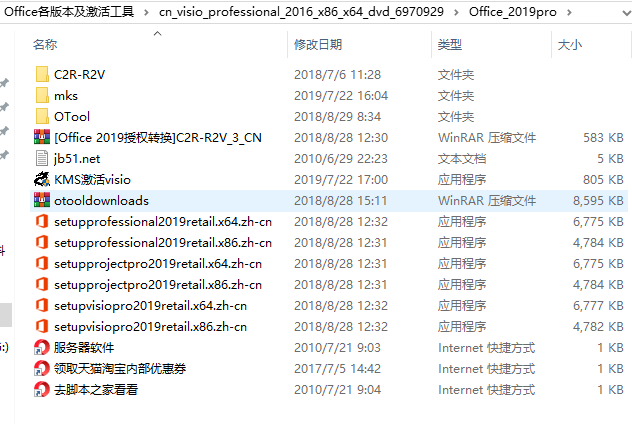
以上我们说到的，只是在软件设计过程中需要用到UML建模图的内容。话说回来，需求分析固然非常重要，因为用例驱动一切，但是，一个项目，一个软件，设计的到底怎么样，性能是否足够优化，最至关重要的一个步骤，其实是数据库设计。接下来，我们看一下数据库设计中，需要用到哪些建模工具。

# 专题讲解：数据库设计——E-R图和概念数据模型

## Microsoft Visio 2019 专业版介绍

Microsoft Visio 2019 专业版，是微软名下的Office办公软件系列的一部分，是运行在Windows操作系统下的一款流程图和矢量绘图的集成绘图工具。Visio无论是标准版还是专业版，都是收费软件，专业版更是高达5200多元人民币的买断费用。幸运的是，KMS可以伪激活180天。

本文提供Visio 2019专业版的安装包及激活工具：



需要的同学带上你的U盘找我拷贝即可，文件大小4G左右。

为什么这里又改用Visio呢？因为starUML里自带的E-R图模板不太……反正没见过这样画E-R图的……所以我们对于数据库设计中需要用到的建模改使用Visio实现。

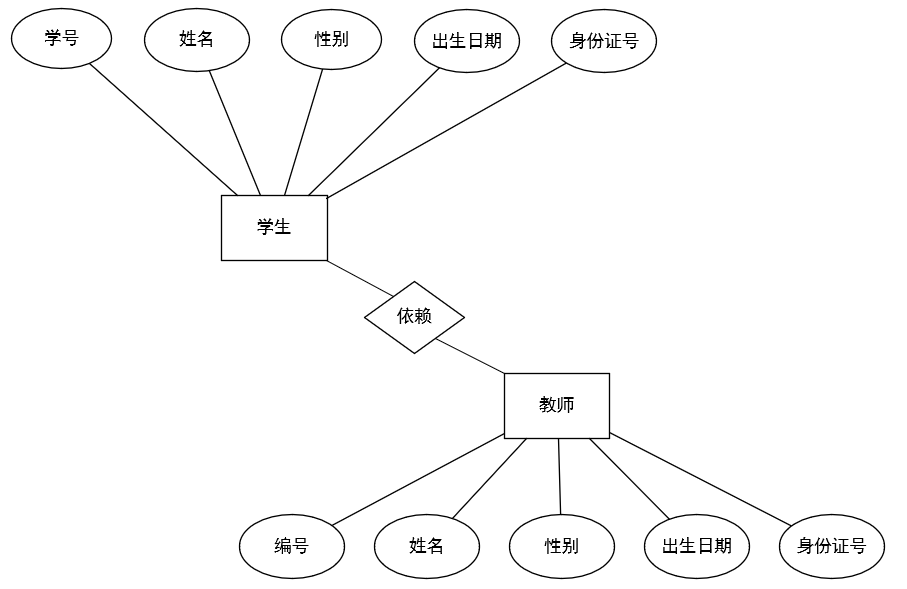
## 什么是E-R图

ER模型最早由Peter Chen（[陈品山](https://baike.baidu.com/item/%E9%99%88%E5%93%81%E5%B1%B1/9952166)）于1976年提出，它在[数据库设计](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E8%AE%BE%E8%AE%A1)领域得到了广泛的认同，但很少用作实际[数据库管理系统](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F)的[数据模型](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%A8%A1%E5%9E%8B)。即使对SXL-92数据库来说，设计好的数据库也是具有挑战性的。它们可以在许多关于数据库设计的文献中找到，比如Toby Teorsey 的著作（1994 ）。

大部分数据库设计产品使用实体-联系模型（ER模型）帮助用户进行数据库设计。ER数据库设计工具提供了  
　　一个“方框与箭头”的绘图工具，帮助用户建立ER图来描绘数据。

实体联系模型，实体[关系模型](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%A8%A1%E5%9E%8B)或实体联系模式图(ERD)是由美籍华裔计算机科学家陈品山(Peter Chen)发明，是概念数据模型的高层描述所使用的数据模型或模式图，它为表述这种实体联系模式图形式的[数据模型](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%A8%A1%E5%9E%8B)提供了图形符号。这种数据模型典型的用在[信息系统](https://baike.baidu.com/item/%E4%BF%A1%E6%81%AF%E7%B3%BB%E7%BB%9F)设计的第一阶段；比如它们在[需求分析](https://baike.baidu.com/item/%E9%9C%80%E6%B1%82%E5%88%86%E6%9E%90)阶段用来描述信息需求和/或要存储在数据库中的信息的类型。但是数据建模技术可以用来描述特定论域(就是感兴趣的区域)的任何本体(就是对使用的术语和它们的联系的概述和分类)。

## 简易E-R图举例



## 概念数据模型

数据模型是现实世界中数据特征的抽象。数据模型应该满足三个方面的要求：  
1）能够比较真实地模拟现实世界  
2）容易为人所理解  
3）便于计算机实现

概念数据模型也称信息模型，它以实体－联系(Entity-RelationShip,简称E-R)理论为基础，并对这一理论进行了扩充。它从用户的观点出发对信息进行建模，主要用于数据库的概念级设计。

通常人们先将现实世界抽象为概念世界，然后再将概念世界转为机器世界。换句话说，就是先将现实世界中的客观对象抽象为实体(Entity)和联系(Relationship),它并不依赖于具体的计算机系统或某个DBMS系统，这种模型就是我们所说的CDM;然后再将CDM转换为计算机上某个DBMS所支持的数据模型，这样的模型就是物理数据模型,即PDM。

CDM是一组严格定义的模型元素的集合，这些模型元素精确地描述了系统的静态特性、动态特性以及完整性约束条件等，其中包括了数据结构、数据操作和完整性约束三部分。

1）数据结构表达为实体和属性;  
2）数据操作表达为实体中的记录的插入、删除、修改、查询等操作;  
3）完整性约束表达为数据的自身完整性约束（如数据类型、检查、规则等）和数据间的参照完整性约束（如联系、继承联系等）

## 实体、属性及标识符的定义

实体（Entity），也称为实例，对应现实世界中可区别于其他对象的“事件”或“事物”。例如，学校中的每个学生，医院中的每个手术。

每个实体都有用来描述实体特征的一组性质，称之为属性，一个实体由若干个属性来描述。如学生实体可由学号、姓名、性别、出生年月、所在系别、入学年份等属性组成。

实体集（Entity Set）是具体相同类型及相同性质实体的集合。例如学校所有学生的集合可定义为“学生”实体集，“学生”实体集中的每个实体均具有学号、姓名、性别、出生年月、所在系别、入学年份等性质。

实体类型（Entity Type）是实体集中每个实体所具有的共同性质的集合，例如“患者”实体类型为：患者｛门诊号，姓名，性别，年龄，身份证号.............｝。实体是实体类型的一个实例，在含义明确的情况下，实体、实体类型通常互换使用。

实体类型中的每个实体包含唯一标识它的一个或一组属性，这些属性称为实体类型的标识符（Identifier），如“学号”是学生实体类型的标识符，“姓名”、“出生日期”、“信址”共同组成“公民”实体类型的标识符。

有些实体类型可以有几组属性充当标识符，选定其中一组属性作为实体类型的主标识符，其他的作为次标识符。

## 概念数据模型建模图示

