

第14章  
测试策略



14.1软件测试基础

什么是“好的”测试?

一个好的测试有很高的概率发现错误  
好的测试是不冗余的

好的测试应该是“品种中的佼佼者”

一个好的测试既不能太简单，也不能太复杂

测试用例设计

目标:发现错误

标准:完整

约束:用最少的精力和时间

1/21



14.1软件测试基础

•可测试性

可操作性——它工作得越好，效率就越高  
测试

可观察性——所见即所测

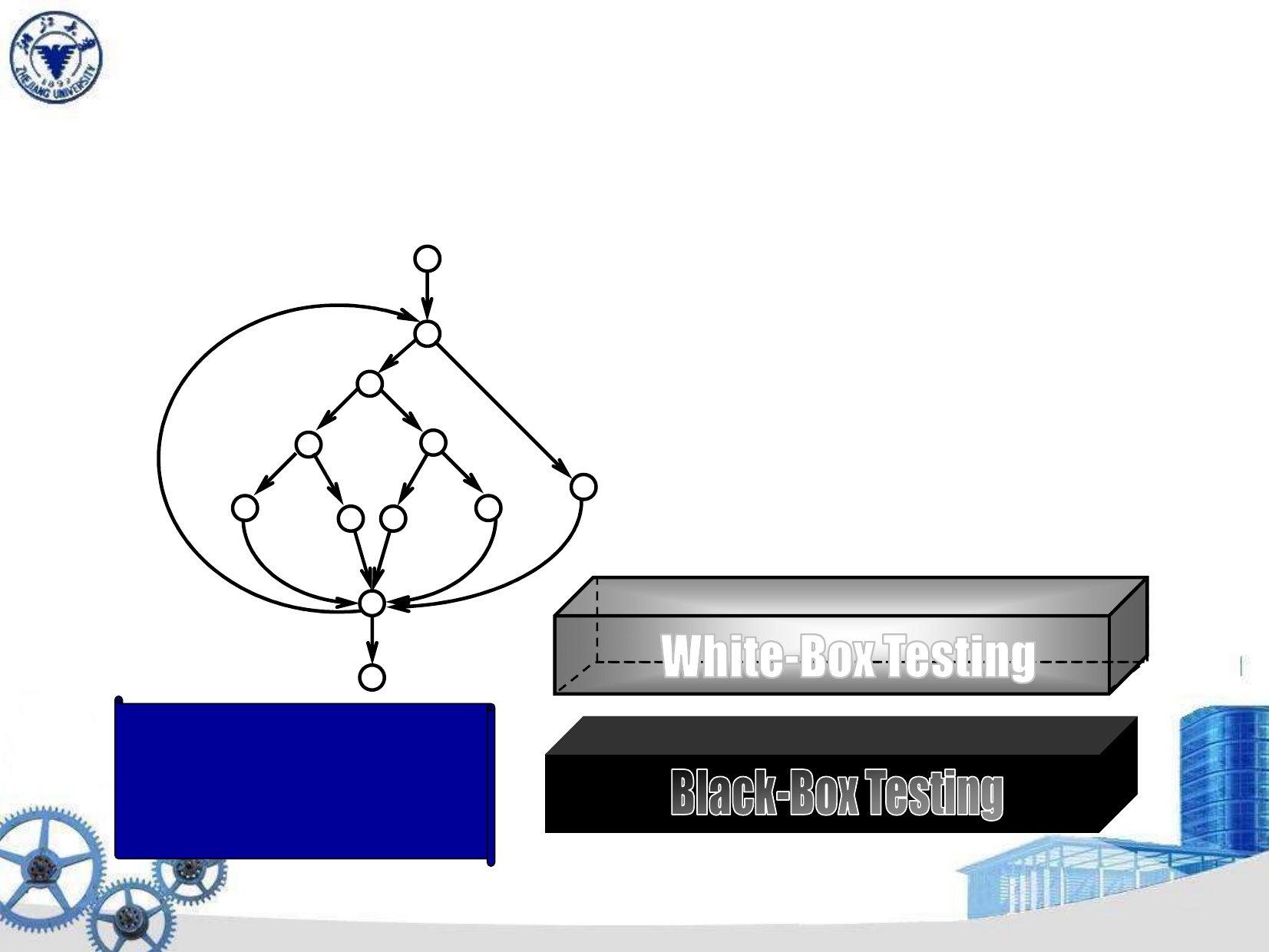
可控性——越好的软件越能被控制  
测试可以自动化和优化

可分解性——通过控制测试的范围，越多  
快速地将问题隔离出来，智能地重新测试  
简单性——可测试的东西越少，我们测试的速度就越快

稳定性——变化越少，中断越少  
测试

可理解性——知道的信息越多，越聪明  
测试

2/21



14.2黑盒一个白盒测试

为什么不尝试详尽的测试?

循环= 20 X

有5201014可能  
路径!如果我们执行一个测试  
每毫秒，它需要  
3170年才能验证这一点  
程序! !

注意:测试不能  
显示缺失  
错误和缺陷。

3/21



14.3白盒测试

定义:白盒测试或玻璃盒测试  
了解产品的内部工作原理，进行测试  
检查所有可能的逻辑路径的工作方式——在小范围内进行测试。

•问题

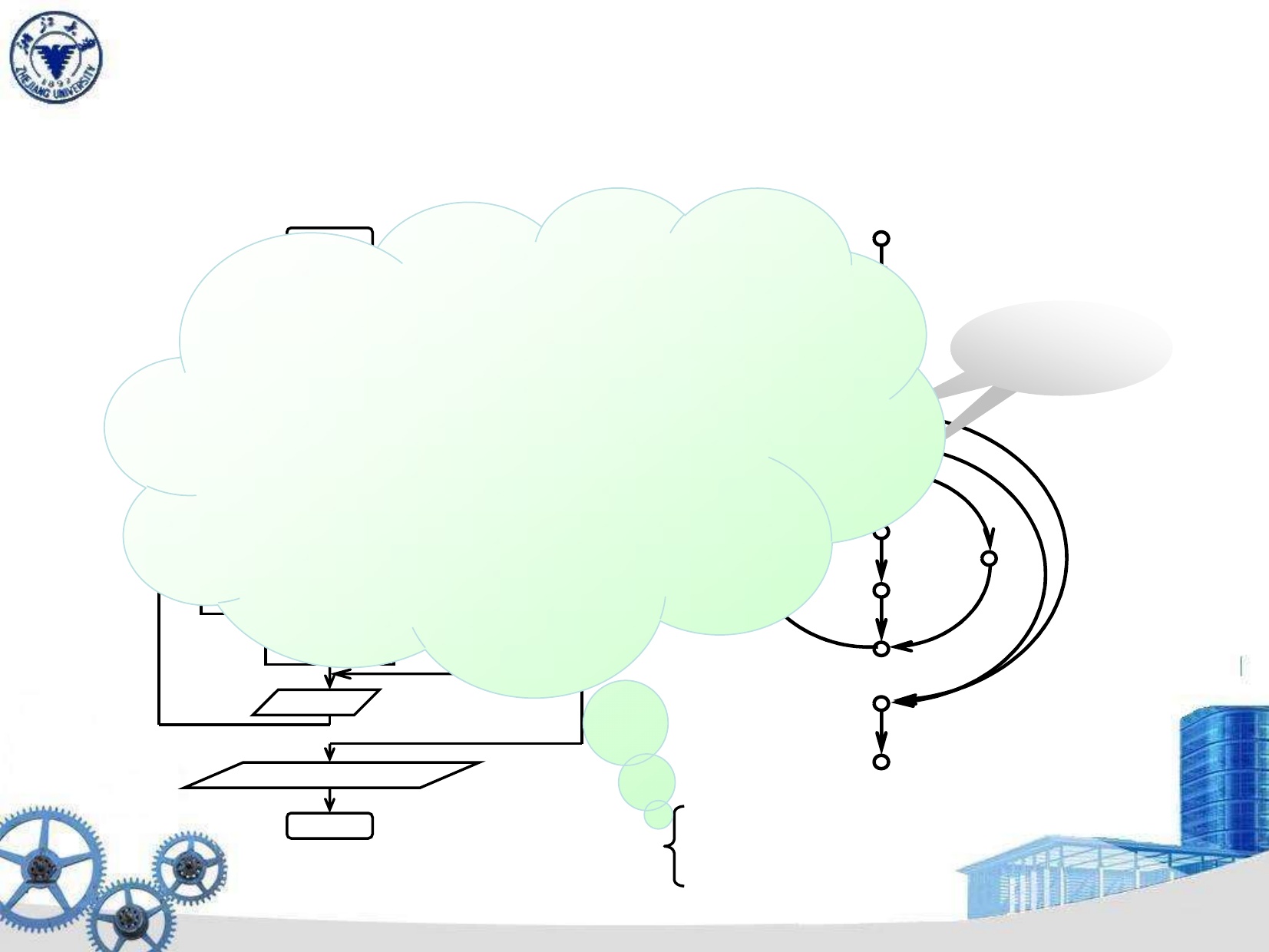
你能保证一个模块内的所有独立路径  
会被执行至少一次?

你能根据他们的真假来锻炼所有的逻辑决策吗  
分支机构?

所有的循环都会在它们的边界和内部执行  
业务范围?

能否通过锻炼内部数据结构来保证其有效性?

4/21



14.4 Basis路径测试

目标:每条语句至少执行一次

h

R3



输入一个



j

输出K、L、合计



k停止

圈  
复杂性=

#的地区  
EN + 2  
P + 1

5/21

一个开始 (开始)

流图bK=0复合条件L=0b(in)

TOTAL=0定义谓词的数量  
c  
c独立输入基节点中的路径  
程序的集合，并提供给我们d

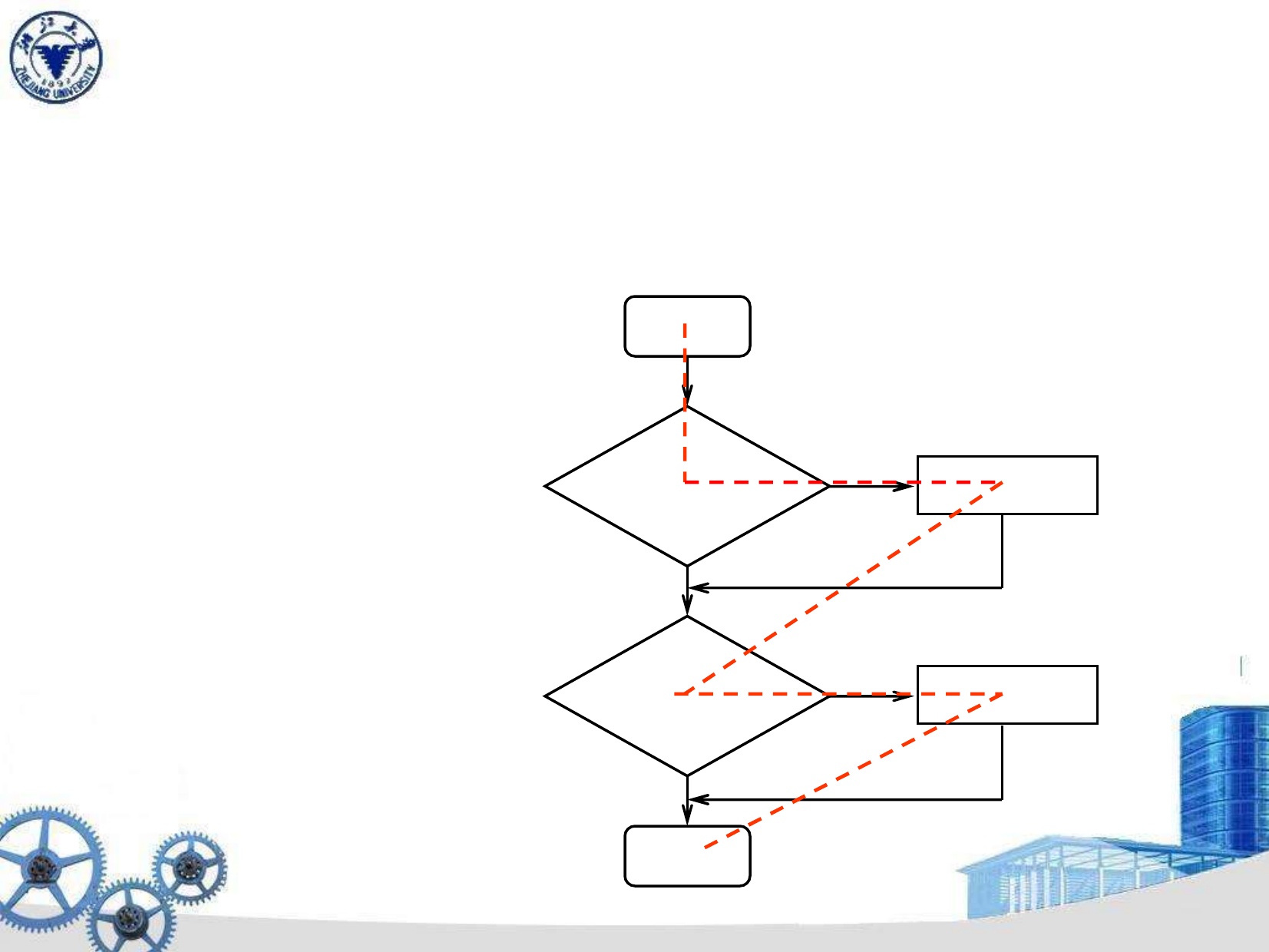
d d '

上层总< 1000  
Do While绑定and测试次数eR4  
那A0T必须进行。

e F f

A>0 R2 R1 i  
英国《金融时报》  
总=总+ Ag)

g i L++ K++



14.4基路径测试

准备测试用例，强制执行每条路径  
在基础集合中的路径-只执行每条语句  
至少执行一次是不够的

例子:

在

CC = 5 ?

测试用例:  
= 3, B = 0, X = 4。



注:If和为  
如为或，则错误将  
不被检测到  
以上测试用例。



T

或者x > 1

X = X + 1



6/21



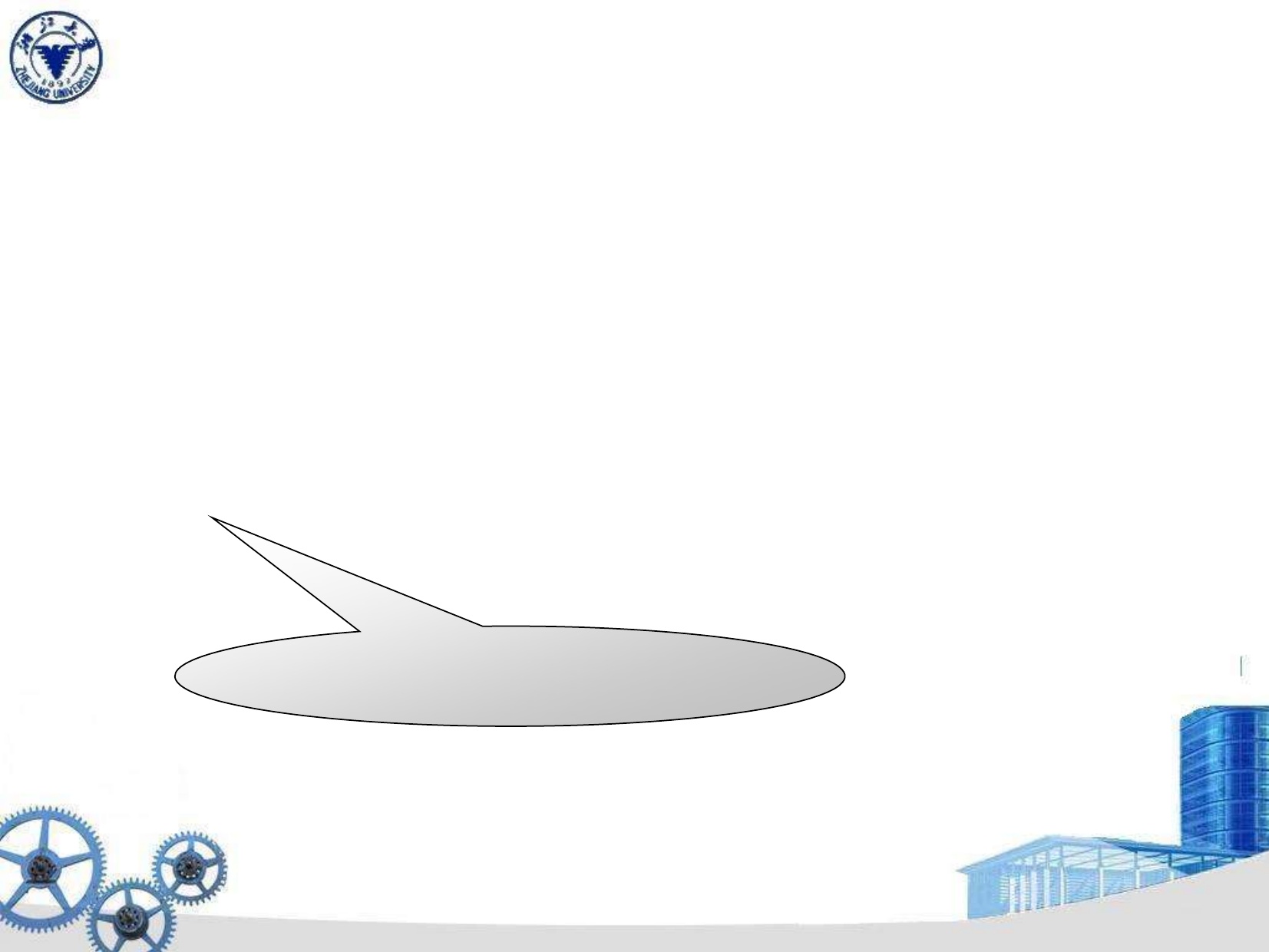
14.5控制结构测试

条件测试:每个条件至少测试一次

例子:测试(E1 > E2)和(E3 = E4)  
B1和B2



7/21



14.5控制结构测试

数据流测试:根据  
中变量定义和使用的位置  
程序

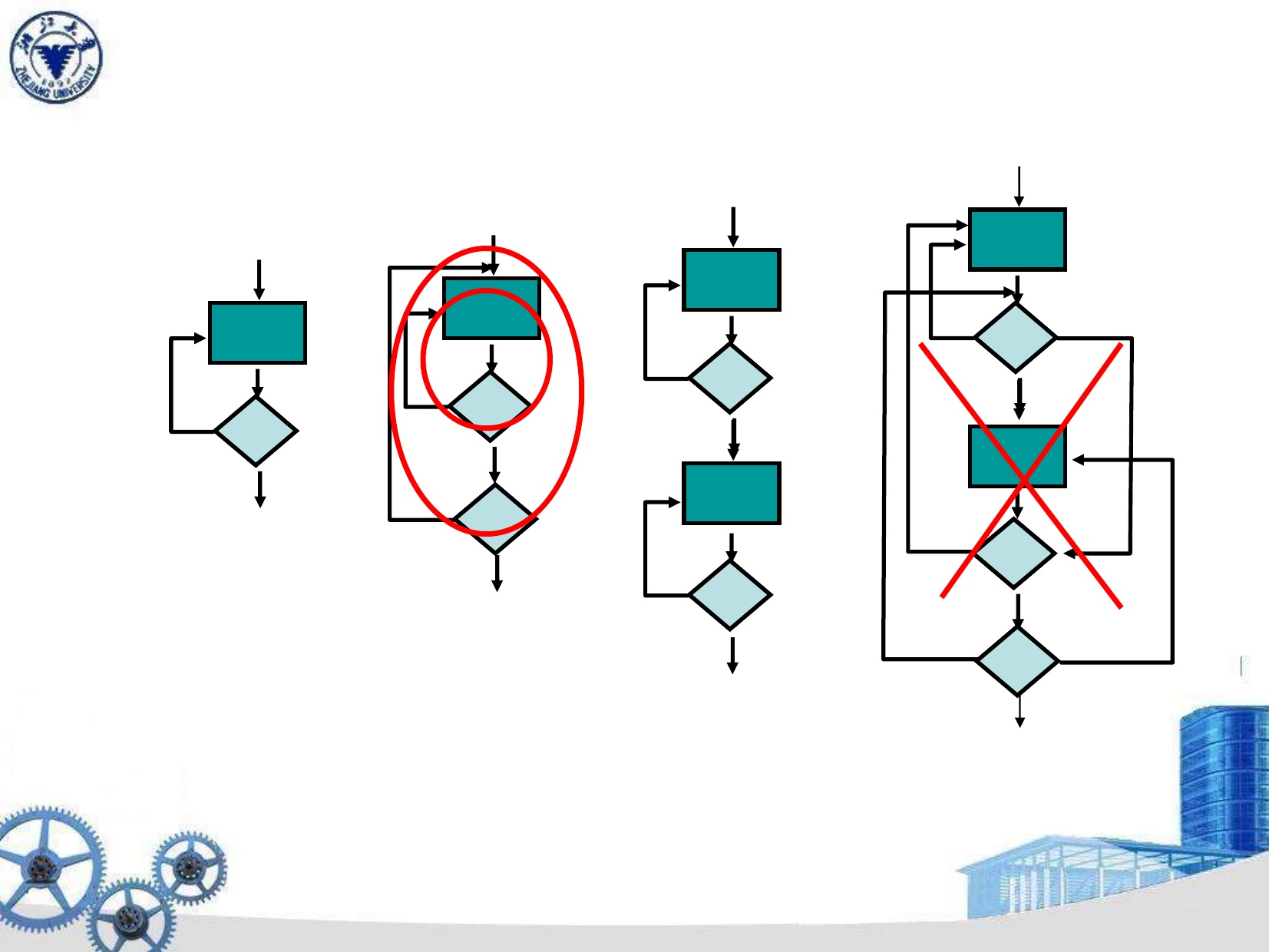
DEF(S) = {X| S包含X的定义};  
USE(S) = {X| S包含X的使用}。

DU链的X = [X, S, S ']其中XDEF(S)和X使用(S ')，  
而S中X的定义是live at S’---- X的流动。

每一条DU链都被覆盖

一次。

8/21



14.5控制结构测试

•循环测试

0 1

简单的

循环

0

1

2

2 < m < n

n - 1

n

n + 1

嵌套的  
循环

连接  
循环

非结构化  
循环

9/21



14.6黑盒测试

定义:黑盒测试或行为测试  
了解产品要执行的特定功能  
仅根据其规格说明演示正确的操作  
而不考虑其内在逻辑——在大范围内进行测试。

•问题

如何测试功能有效性?

如何测试系统行为和性能?  
哪些类别的输入将成为好的测试用例?  
系统是否对某些输入值特别敏感?  
数据类的边界是如何隔离的?  
系统能容忍的数据速率和数据量是多少?  
数据的特定组合会对系统产生什么影响

操作?

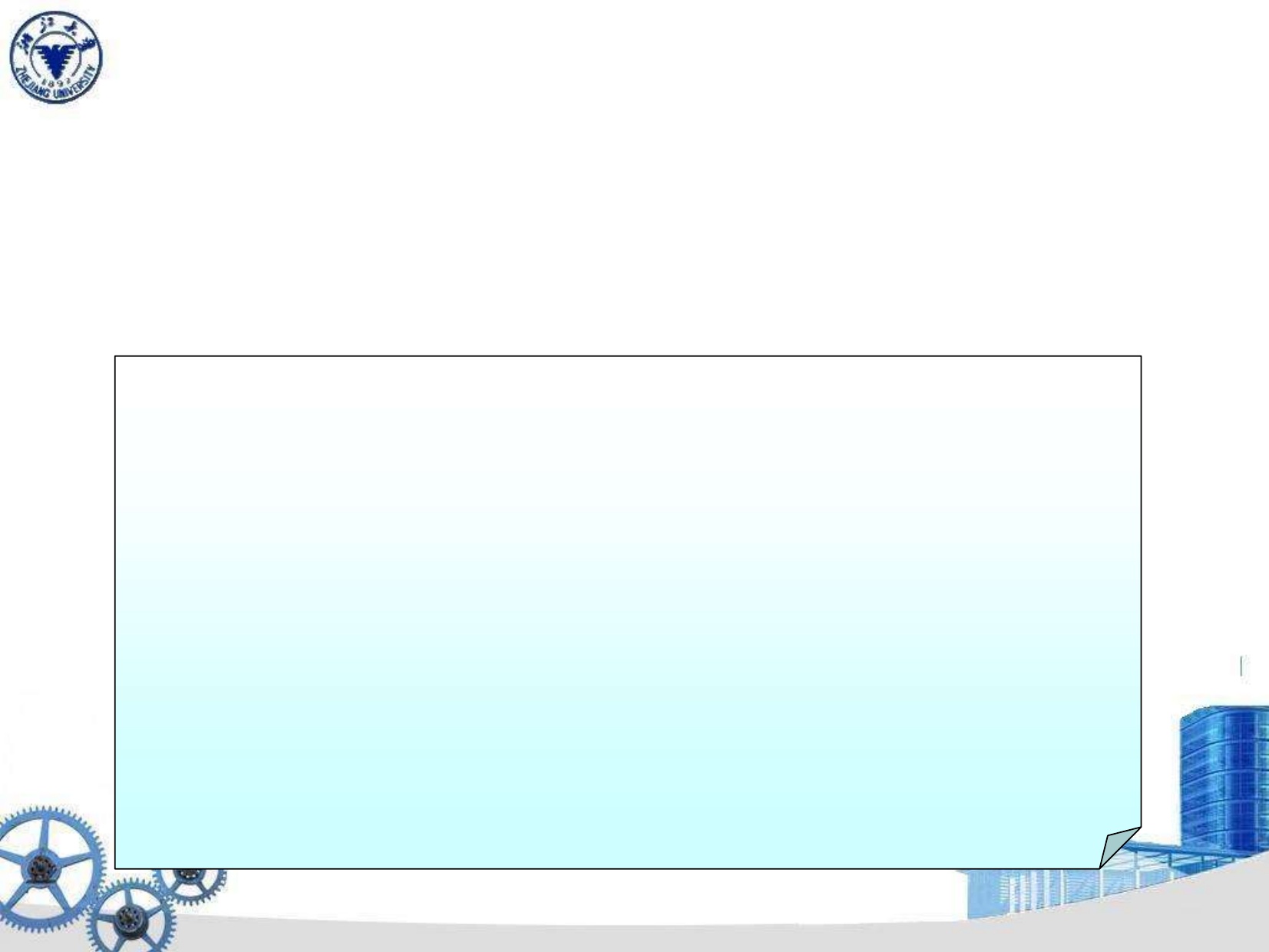
10/21



14.6黑盒测试

•技术  
基于测试  
等价划分  
边界值分析  
正交数组测试

11/21



14.6黑盒测试

等价划分

等价类表示一个有效或无效的集合  
状态为输入条件，因此没有特别的原因  
选择一个元素而不是另一个元素作为类代表。

的指导方针

•

•

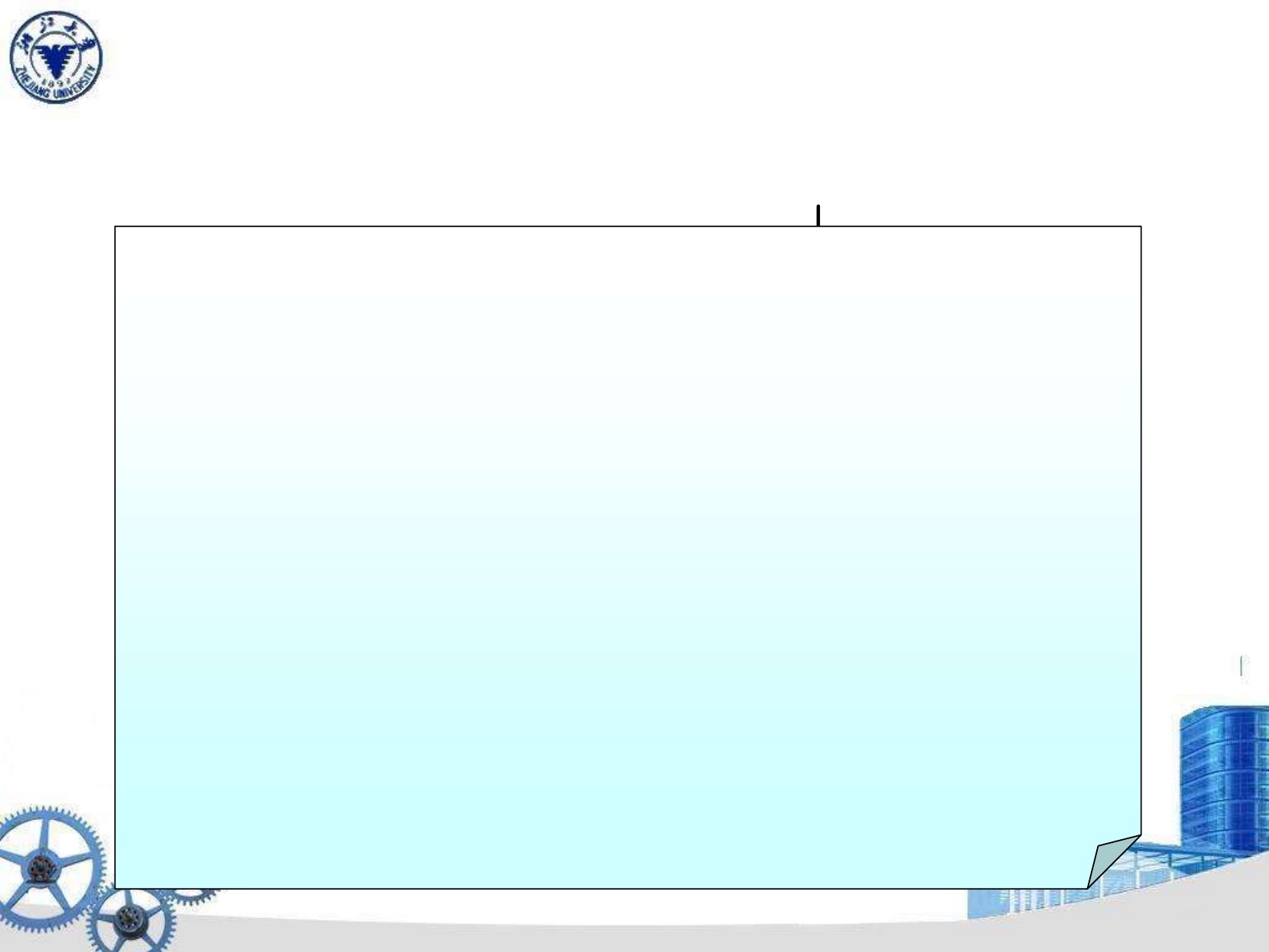
•

•

如果输入条件指定了一个范围，一个有效，两个有效  
无效的等价类被定义。

如果一个输入条件需要一个特定的值，一个有效  
并且定义了两个无效的等价类。  
如果一个输入条件指定了集合中的一个成员，则其中一个有效  
而定义了一个无效的等价类。  
如果一个输入条件是布尔值，一个有效一个无效  
等价类被定义。

12/21



14.6黑盒测试

边界值分析(BVA)

的指导方针

•

•

•

•

“虫子潜伏在角落里

如果inputandconditioncongregate指定一个以值a为边界的范围  
B，测试用例应该和边界…”包括a和B，值就在上面  
就在a和b下面。

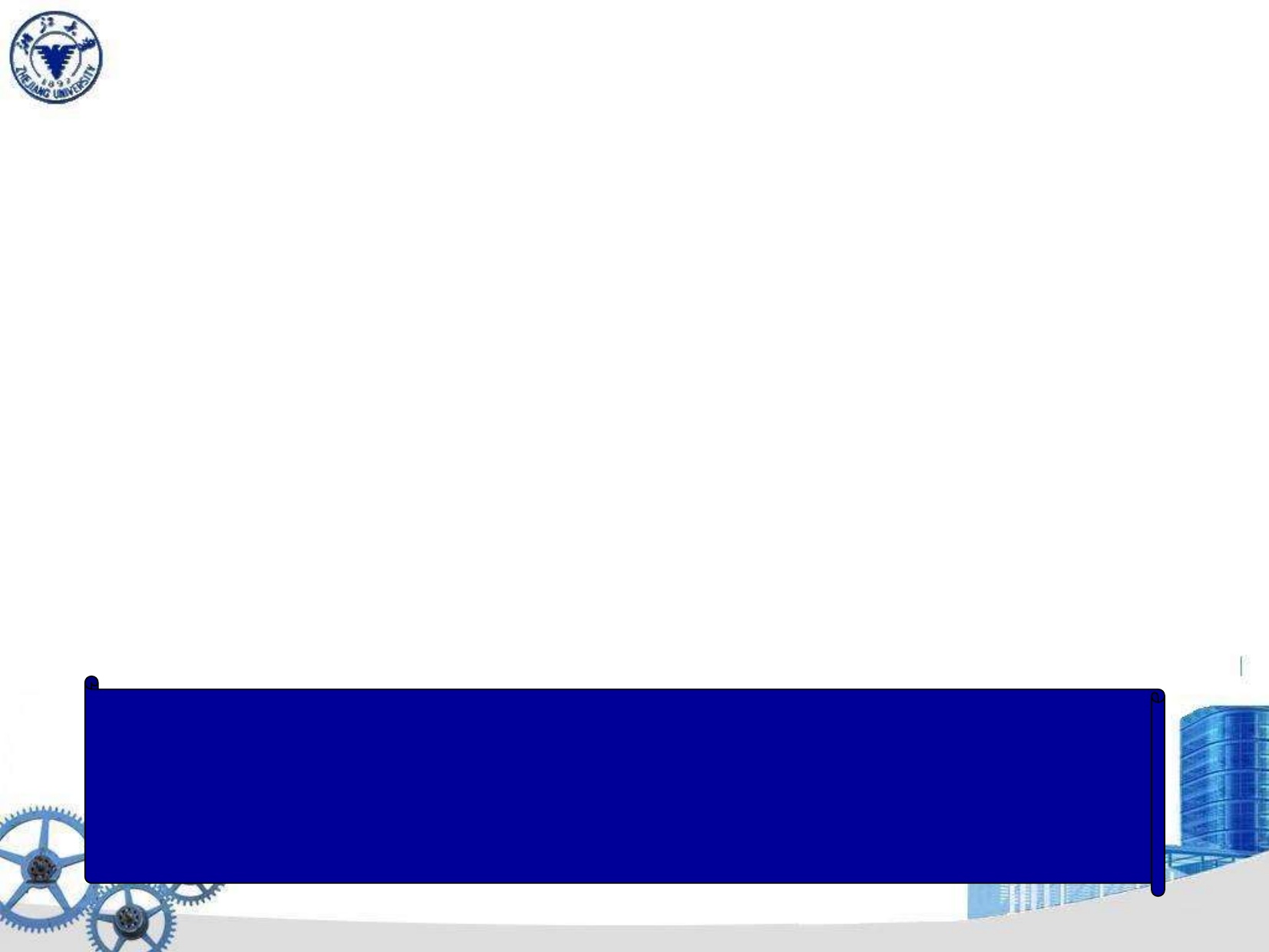
如果输入条件化器和数值，测试Boris指定

例应行使最小值和最大值  
数字，以及正上方和正下方的值  
最小值和最大值。

将准则1和2应用于输出条件，测试用例  
应该设计成生产最小值和最大值吗  
输出报告。

如果程序内部数据结构有边界(例如:  
大小限制)，一定要测试边界。

13/21



14.7面向对象测试方法

测试用例设计

1.每个测试用例都应该被唯一地标识并且应该是明确的  
与要测试的类相关联，

2.应说明测试的目的，

3.应该为每个测试制定一个测试步骤的列表  
包含(BER94):

a.被测试对象的指定状态列表  
B.将被执行的消息和操作的列表  
测试的结果

C.测试对象时可能发生的异常列表  
D.外部条件列表

E.有助于理解或理解的补充信息

实施测试。

不像传统的测试用例设计，由I-P-O驱动  
软件视图或单个模块的算法细节，面向对象  
测试的重点是设计适当的操作序列  
练习类的状态。

14/21



面向对象的测试方法

•结合测试

测试人员寻找合理的错误(例如:的方面  
可能导致系统缺陷的实施方面)。来  
确定这些故障是否存在，设计测试用例  
来练习设计或编码。

类测试和类层次结构

-继承不排除彻底测试的需要  
所有派生类。事实上，它实际上可以使  
测试过程。

•基于场景的测试设计

基于场景的测试专注于用户做什么，  
而不是产品做了什么。这意味着捕捉任务  
(通过用例)用户必须执行的任务，然后应用  
它们和它们的变体作为测试。

15/21



14.8类级别测试方法

•随机测试

识别适用于某个类的操作

-定义使用它们的约束

-确定最小测试序列

定义最小寿命历史的操作序列  
类(对象)的

-生成各种随机(但有效)测试序列

•练习其他(更复杂的)类实例生命历史  
需要大量的数据排列和

组合，可能会效率低下。

16/21



14.8类级别测试方法

•分区测试

-减少测试类所需的测试用例数量  
与常规的等价类划分方法相同  
软件

——基于状态的分区

•根据操作的能力进行分类和测试  
改变类的状态

——属性的划分

•根据属性对操作进行分类和测试  
他们使用

——category-based分区

•基于泛型函数对操作进行分类和测试  
每一个执行

17/21



14.9类间测试用例设计

•多类测试(随机)

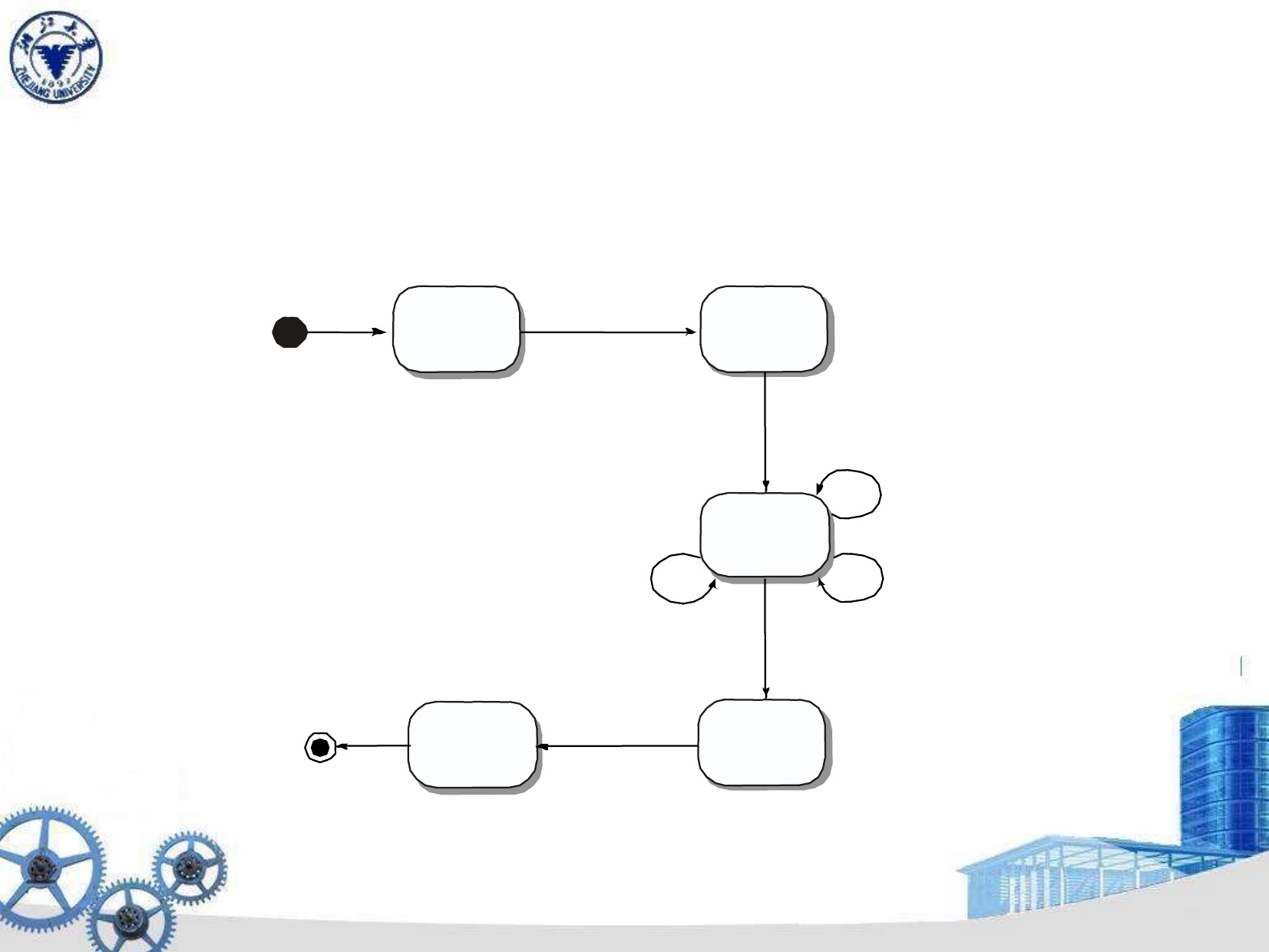
-对于每个客户端类，使用类操作符列表来生成  
一系列随机测试序列。操作员将发送  
消息发送给其他服务器类。

-对于生成的每条消息，确定  
Collaborator类和服务器中相应的操作符  
对象。

-对于服务器对象中的每个操作符(已被调用的)  
从客户端对象发送的消息)，确定消息  
它传递。

-对于每条消息，确定下一级的  
被调用的操作符，并将它们合并到测试中  
序列。

18/21



14.9类间测试用例设计

•行为测试

-要设计的测试应该实现所有状态的覆盖[KIR94]。

开放

空  
acct

设置Accnt

设置  
acct

存款  
(初始)

存款

平衡

信贷  
accntInfo

工作  
acct

撤回

撤军  
(最终)

死  
acct

关闭

非职业  
acct

图1 4.12 St at e diagram f or Account class(改编自[KIR9 4])

19/21



14.10专业测试

图形用户界面

•客户端/服务器架构

•文档和帮助设施

•实时系统

14.11测试模式

定义模式并复用

20/21



关于测试的结语

测试永远不会结束，它只是被转移  
从你(软件工程师)到你的  
客户。每次你的客户使用  
程序，就会进行一项测试。

21/21