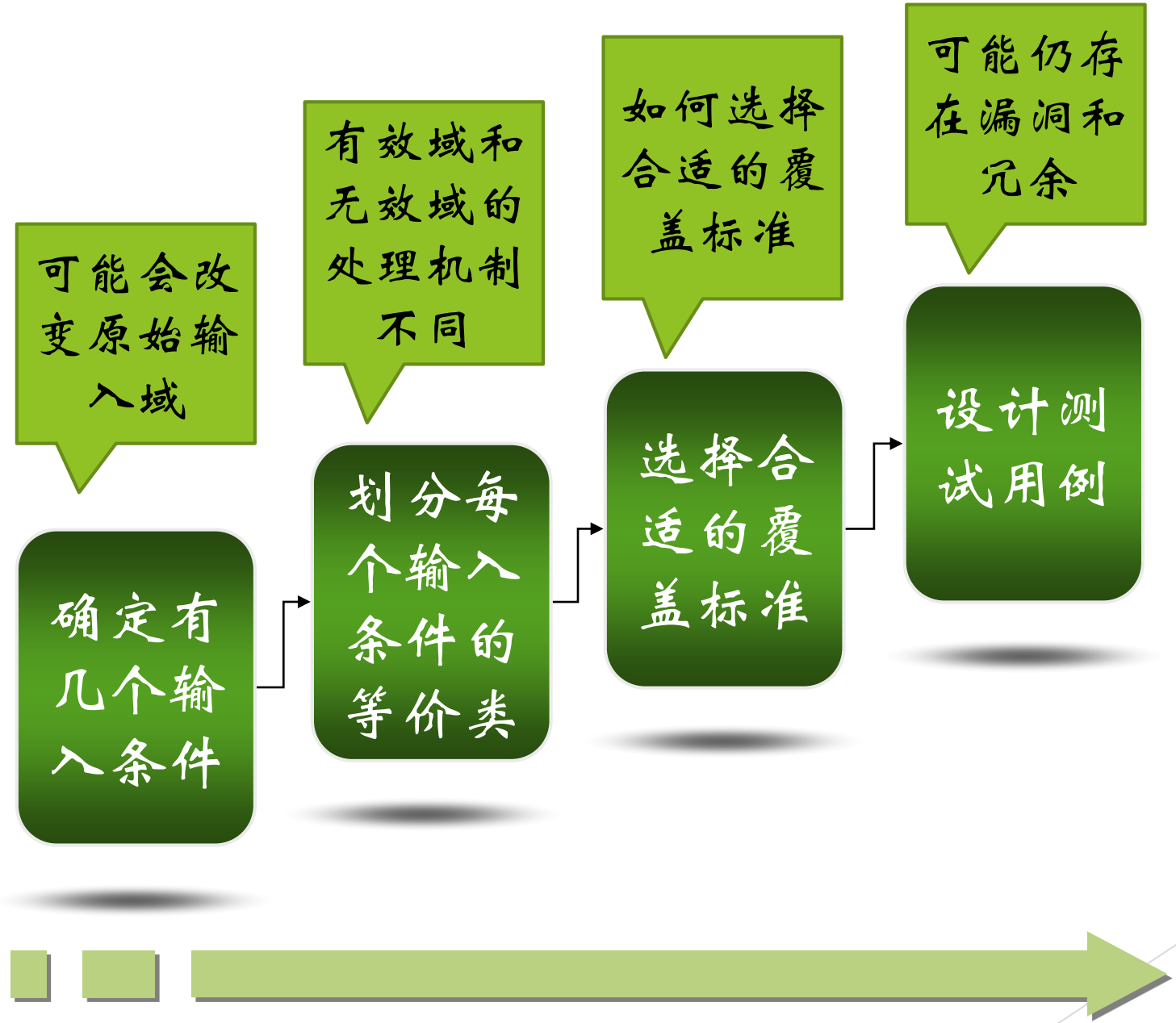
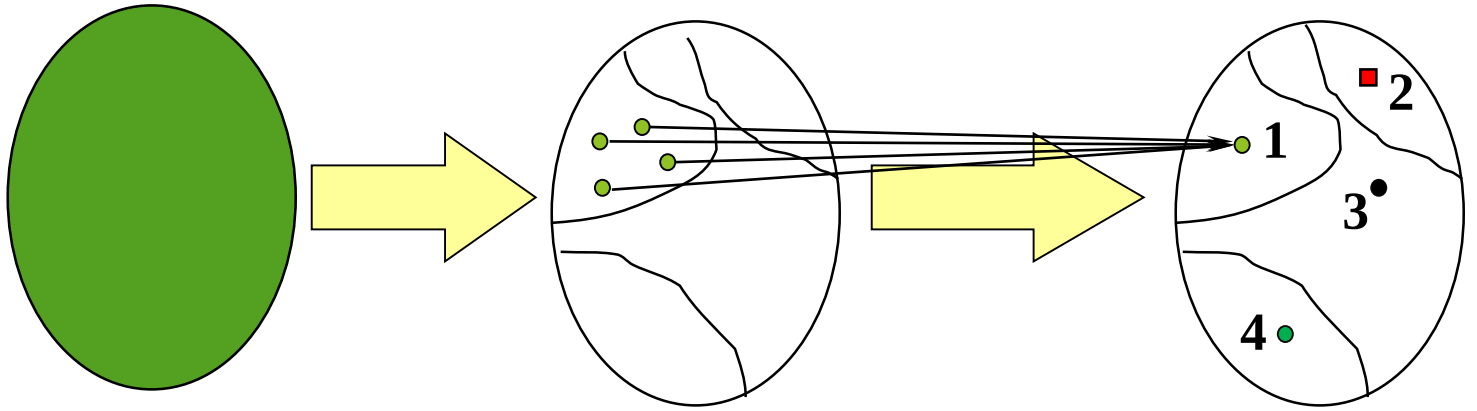
等价类测试的陷阱

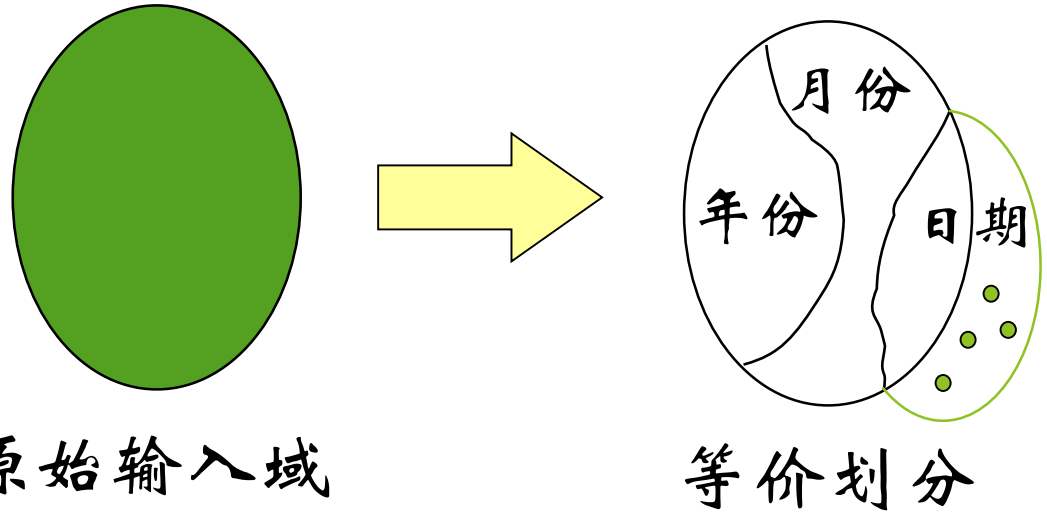

可能会改变原始输入域
有效域和无效域的处理机制不同
如何选择合适的覆盖标准
可能仍存在漏洞和冗余
确定有几个输入条件
划分每个输入条件的等价类
选择合适的覆盖标准
设计测试用例

1. 确定输入条件时，可能会改变原始输入域



3个约束：分而不交
合而不变
类内等价

1. 确定输入条件时，可能会改变原始输入域



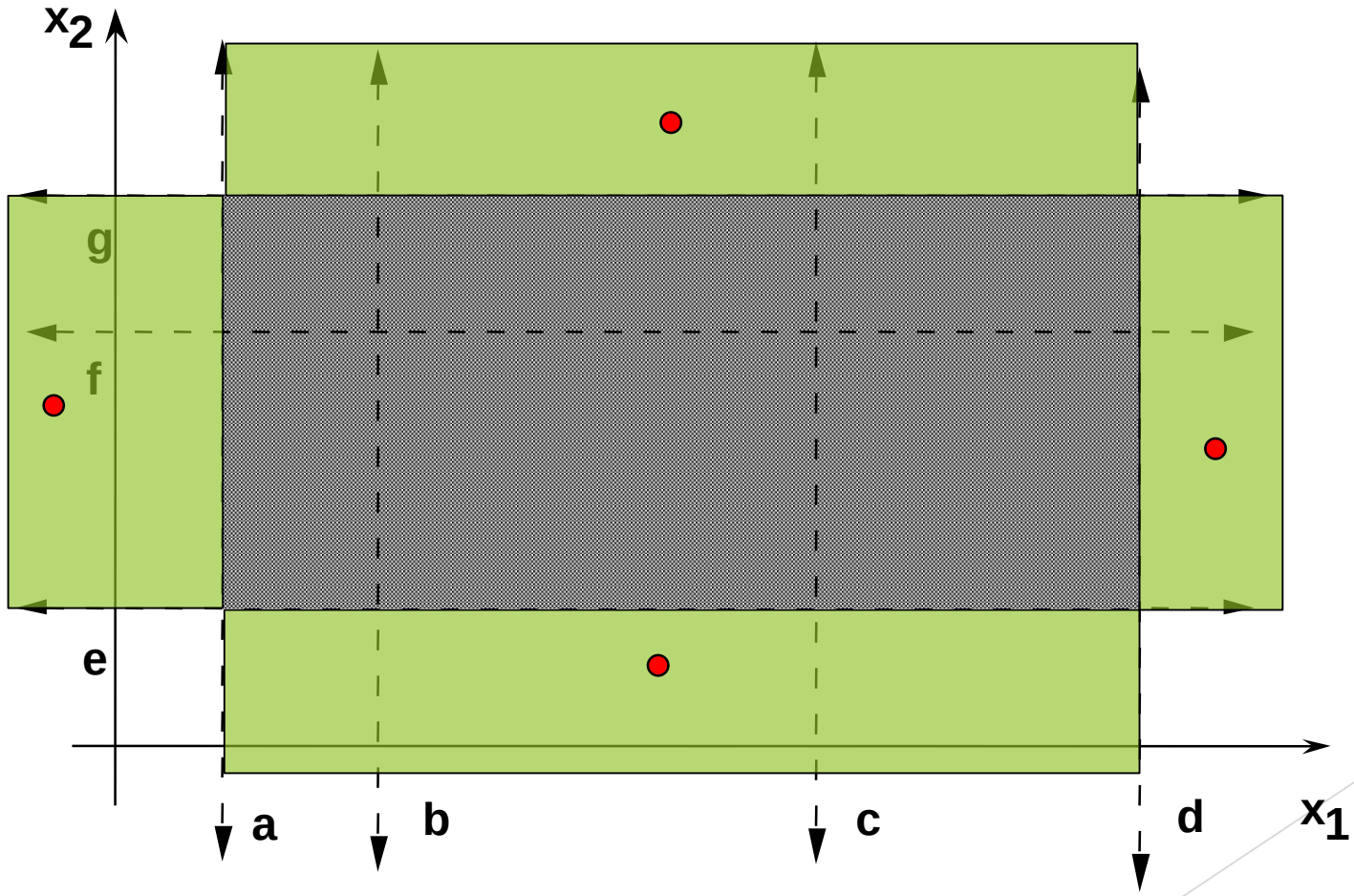
1. 确定输入条件时，可能会改变原始输入域

- 基于系统实现的等价划分
  - 从程序员实现系统的角度出发，多关注接口
- 基于用户需求的等价划分
  - 从功能和业务处理机制角度出发，多关注隐含系统内部处理流程

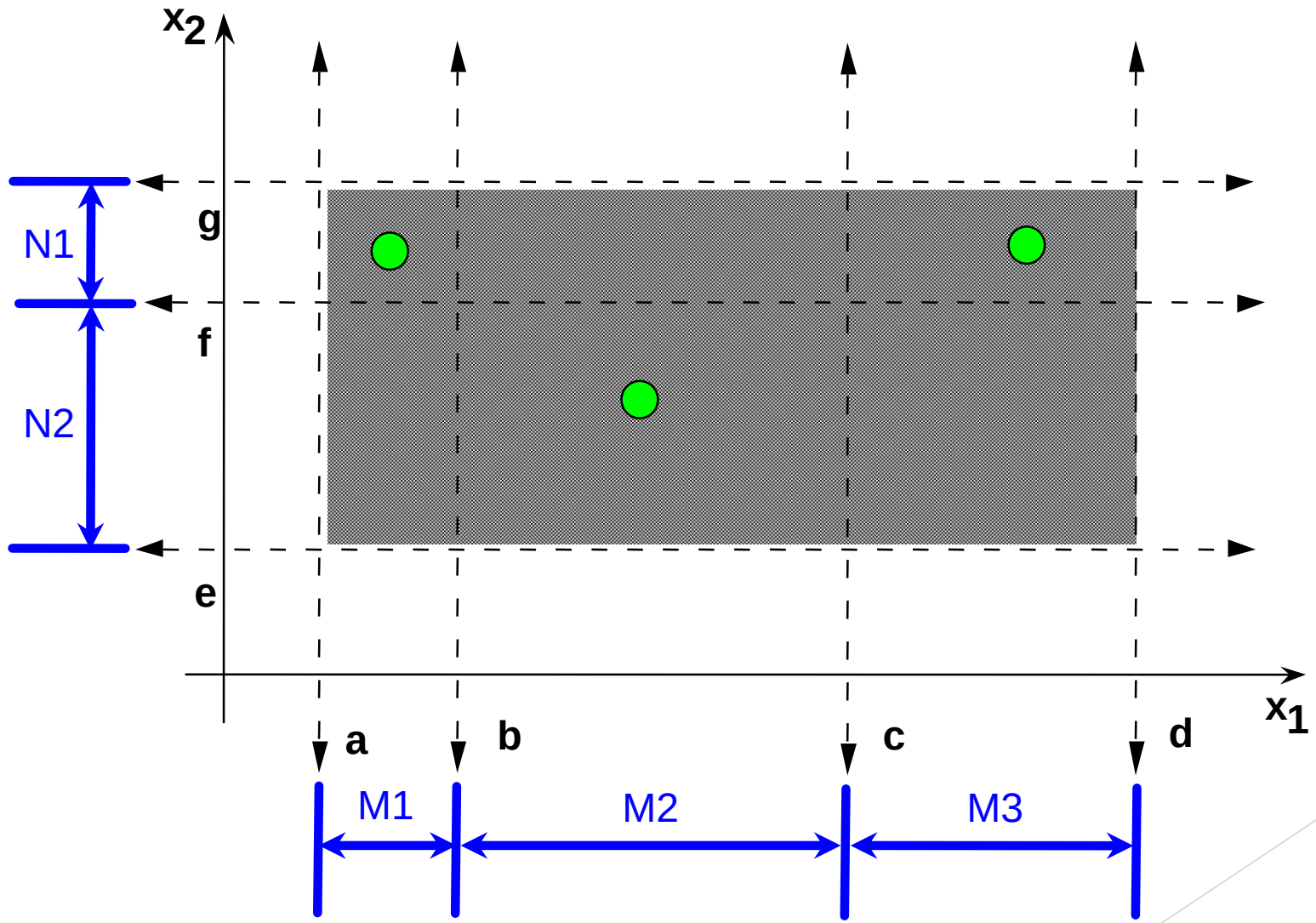
2. 对有效域和无效域可以用相同方式进行等价类测试吗?

- 有效等价类
  - 输入域中一组有意义的数据的集合
  - 有效等价类被用于检验系统指定功能和性能能否正确实现
- 无效等价类
  - 输入域中一组无意义的数据的集合
  - 无效等价类被用于检验系统的容错性

2. 对有效域和无效域可以用相同方式进行等价类测试吗?

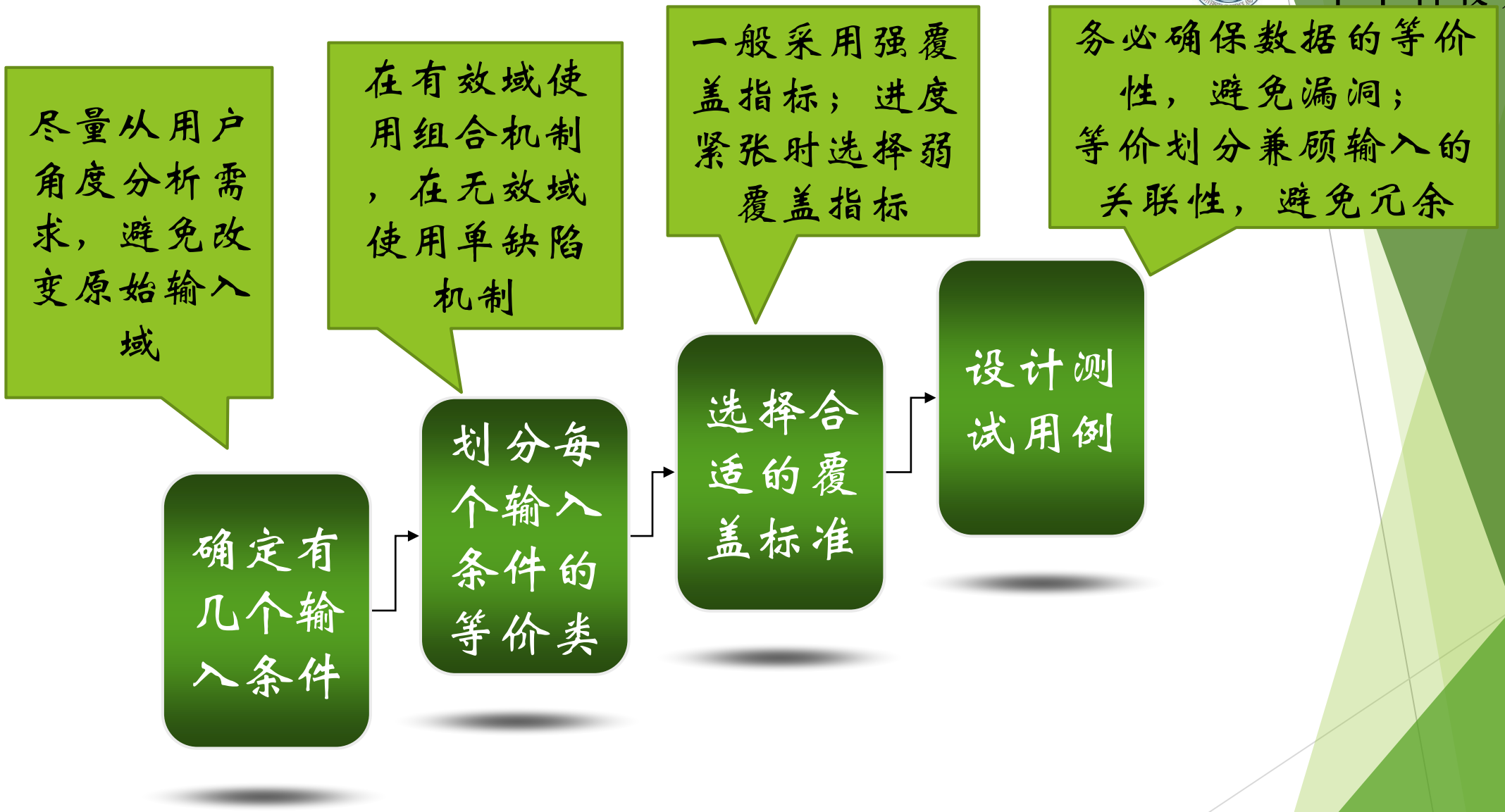


3. 如何选择合适的覆盖指标



4. 等价类测试真的没有漏洞，没有冗余吗？

有效等价类
Y1={年 | 闰年}
Y2={年 | 不是闰年}
M1={月 | 月是具有31天的月份}
M2={月 | 月是具有30天的月份}
M3={月 | 月是2月}
D1={日期 | 1<=日期<=28}
D2={日期 | 日期是29号}
D3={日期 | 日期是30号}
D4={日期 | 日期是31号}


尽量从用户角度分析需求，避免改变原始输入域
在有效域使用组合机制，在无效域使用单缺陷机制
一般采用强覆盖指标；进度紧张时选择弱覆盖指标
务必确保数据的等价性，避免漏洞；
等价划分兼顾输入的关联性，避免冗余
确定有几个输入条件
划分每个输入条件的等价类
选择合适的覆盖标准
设计测试用例