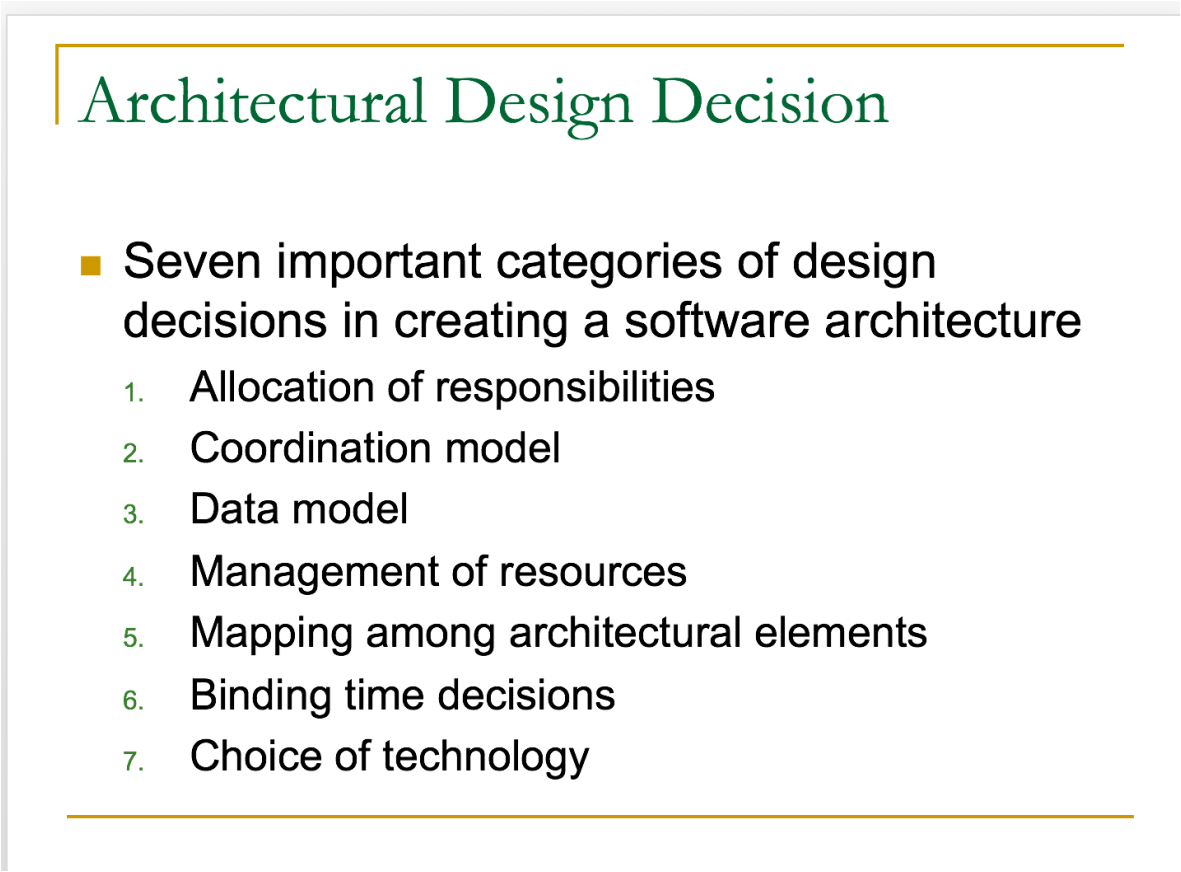
软件体系架构

chapter1
体系模式（优缺点）
1.pipe and filter
2.objected oriented
3.evented based system model(隐式调用）
4，layered pattern (每一层只向其上一层提供服务，实际操作中有跨层调用出现违反设计规则-出现的损伤）-用来屏蔽底层的实现细节
5. interpreters pattern（效率低，性能较差，解释执行）

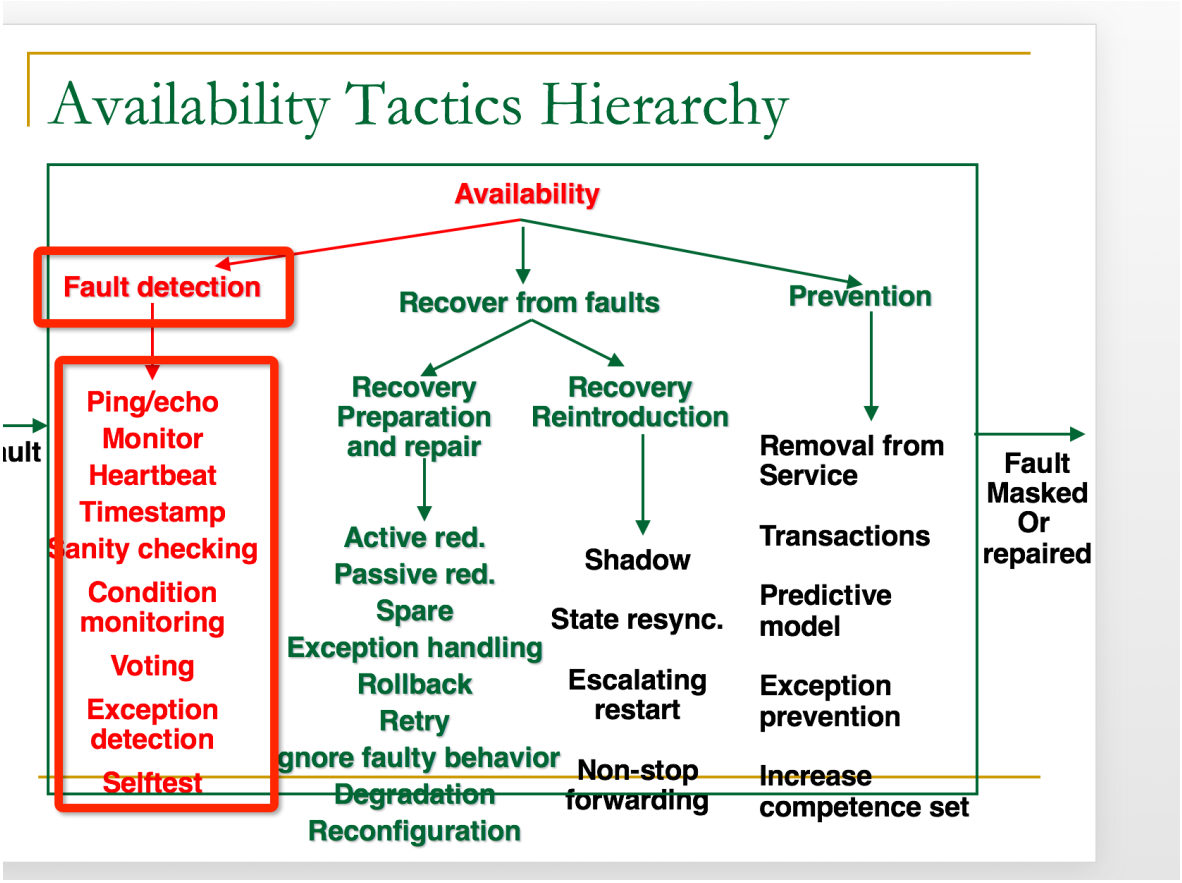
chapter 2
并发，模块分解结构等各种structures和案例
基本的概念及常识

chapter3
质量属性：决定系统划分的关键因素
功能与构架关系
重要的质量属性：可用性，可修改性。performance

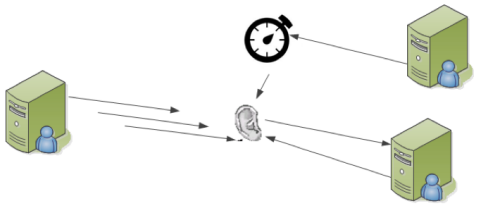


（质量属性如何进行分析，大致看懂）

*构架设计策略
chapter4 tactics for availability
failure :可看到
fault : 不可见的
策略:



Monitor, Watchdog and Heartbeat (1)



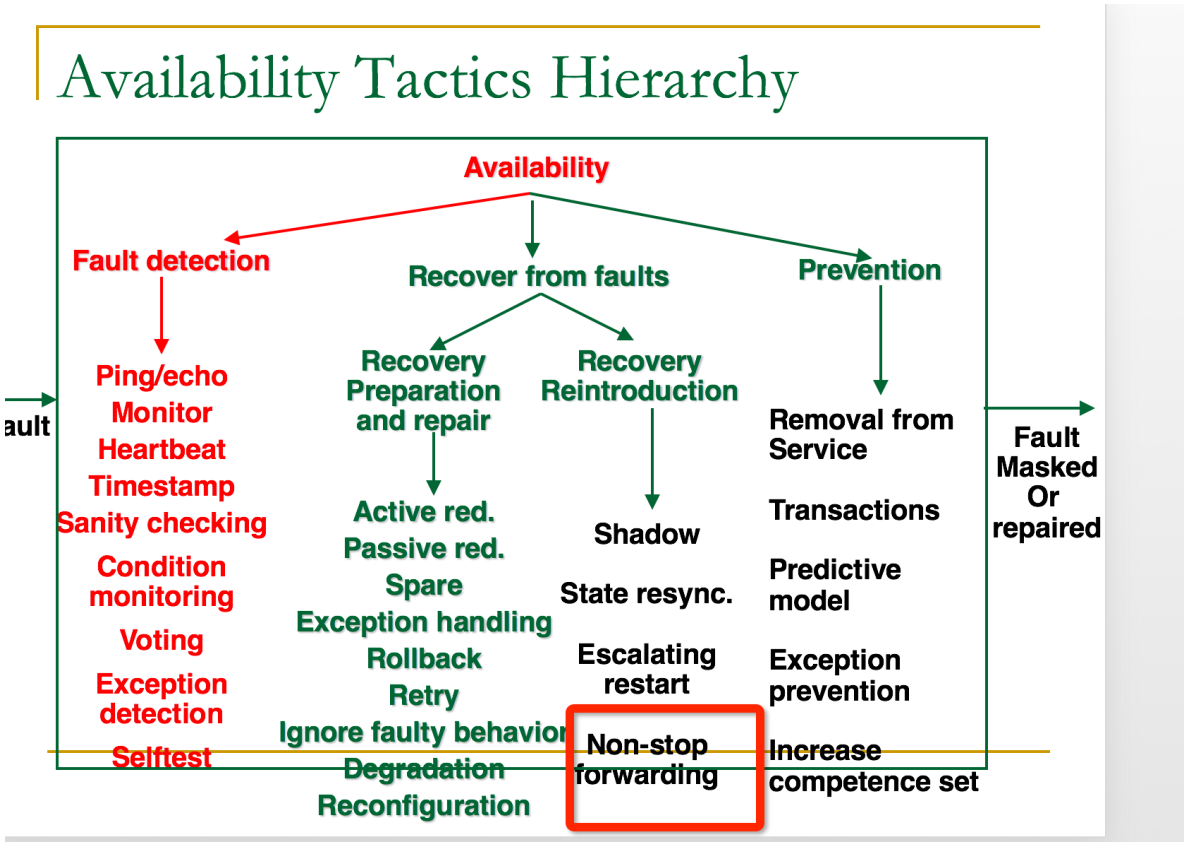
- A system monitor watches the running state of other system components: processors, processes, I/O, memory, detect failure or congestion in the network or other shared resources
  - E.g. Process monitor in Oracle
  - Detect failure or congestion in the network or other shared resources

watch dog 策略

voting 策略（代价昂贵，只用在要求苛刻，极端的场景当中）—理论上讲没有宕机事件
analytic redundancy最昂贵策略
电商不宜采用voting策略，太昂贵

active redundancy/
#数据同步问题
#适用场景：
高可用性的数据库系统，要求较为苛刻的web server中-active redundancy
passive redundancy（电商策略）
spare
其他策略理解

chapter 5



1. nfs：

适用场合：路由部分

2. 状态异步

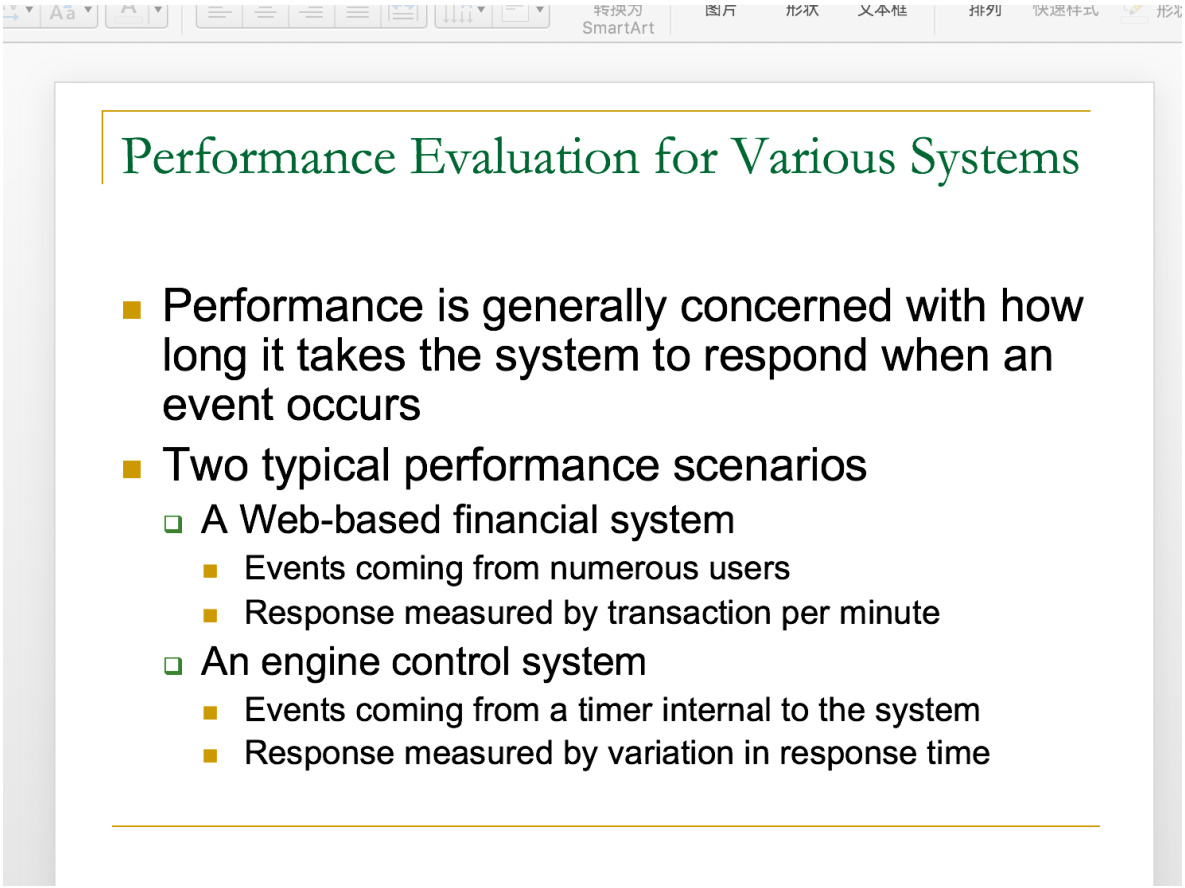
3. transaction

chapter 6
可交互性：概念

architectural mismatch
interface mismatch

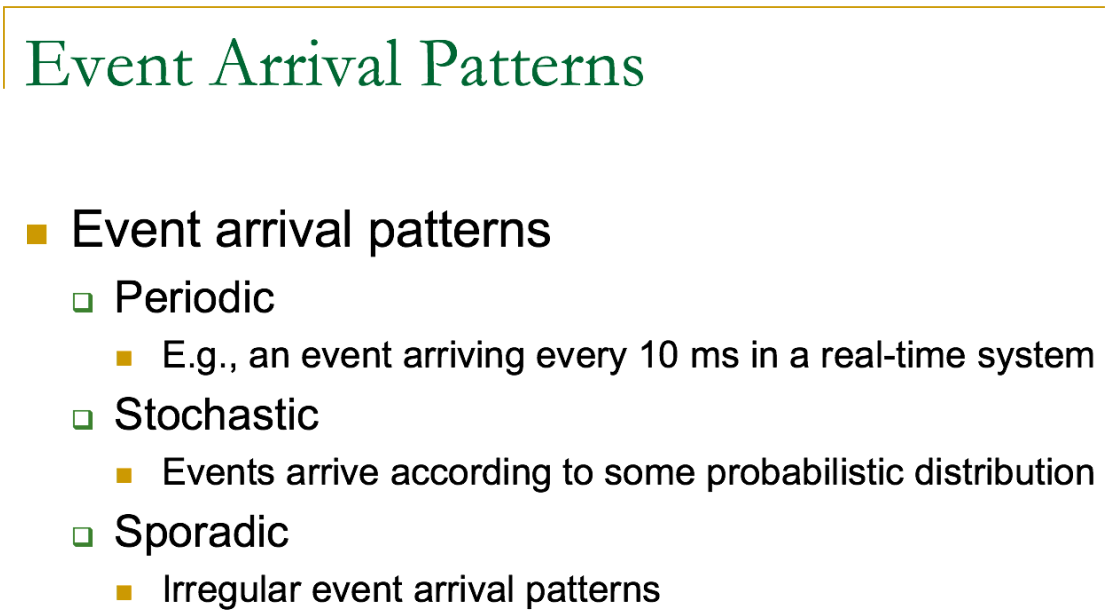
locate & manage interface
manage interface 案例
工作流引擎

chapter 7 性能策略
性能表现的衡量指标：
吞吐量等
（不同系统的衡量指标不同）



jitter

其他概念性的恭喜：
事件到达方式



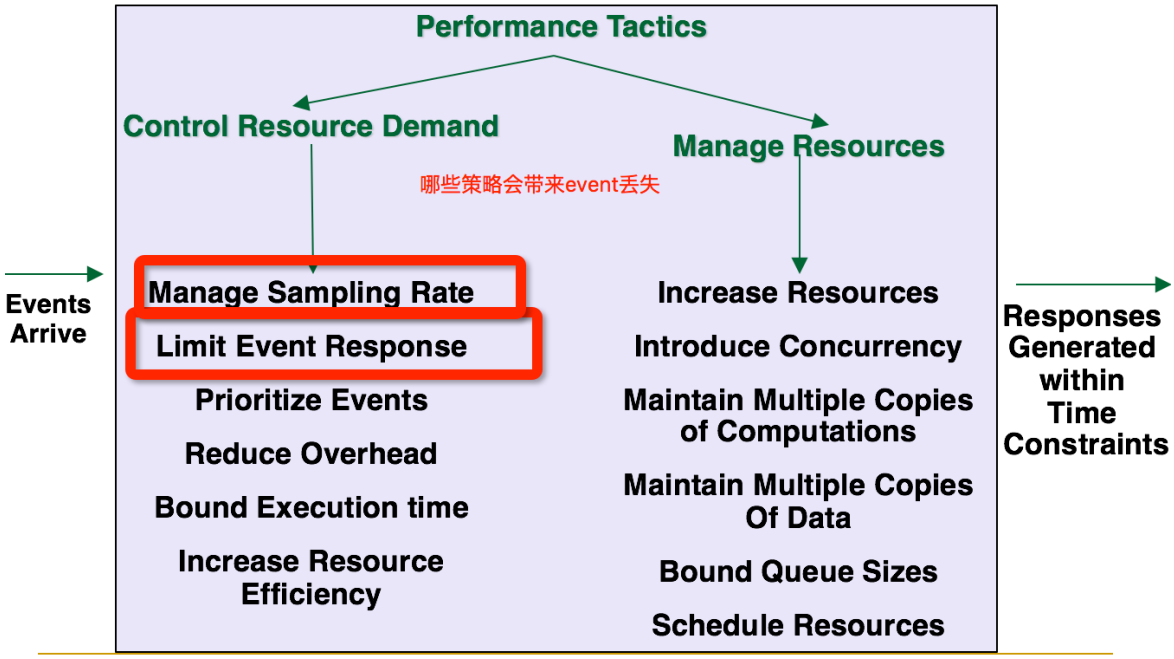
response measures

Response Measures

- Response can be measured by:
  - Latency
    - The time between the arrival of the stimulus and the system's response to it
  - Deadlines in processing
    - The event shall be processed within a deadline
  - Throughput
    - E.g., transactions per minute
  - Jitter
    - The variation in latency
  - Events not processed when the system was too busy (miss rate)

response time = processing time + block time
响应时间 = 处理时间+等待时间（拥塞时间）

Performance Tactics Hierarchy



上图策略的应用案例：
计算机系统设计的cache设计、
备份、

*引入并发 introduce concurrency（存在限制：在实时系统的设计当中没有作用。）（多线程在增加吞吐量，但是时效性差）

调度算法：
与 实时系统的设计：抢占策略，实时系统的优先级设计设置为-1（作为最高优先级）。
FIFO等调度策略的应用案例

chapter 8 可修改性
对可修改性策略的评估

A Simple Equation for Planning Change-handling Mechanism

- For N similar modifications, a simplified justification for a change-handling mechanism:
  - *N* x Cost of making the change without the mechanism < Cost of installing the mechanism + (N x Cost of making the change using the mechanism)
  - N is the expected number of modifications

Change-planning equation

概念：
cohesion

Increase Cohesion: Increase Semantic Coherence

- Semantic coherence: the relationships among responsibilities in a module
  - Keep things that are related together (serve the same purpose)
- If the responsibilities A and B in a module do not serve the same purpose, they should be placed in different modules
  - Creating a new module
  - Moving a responsibility to an existing module

reduce coupling:

Reduce Coupling: Encapsulate

- Encapsulate: visible API + transparent implementation
- Interface limits the ways in which external responsibilities can interact with the module
  - The external responsibilities can now only directly interact with the module through the exposed interface

中间组件减低耦合性：（降低性能？）

Reduce Coupling: Use an Intermediary

- Intermediaries are used to break a dependency between responsibility A and responsibility B, e.g.:
  - Directory service in service invocation
  - Data repository separates readers of a piece of data from writers of that data
  - Memory handles for runtime memory location
  - Resource manager for resource contention
  - Publish-subscribe pattern removes the data producer's knowledge of its consumers

降低显示调用的依赖性

defer binding & 推迟绑定

chapter 9 安全性策略
安全性三要素+assurance+authentication nonrepudiation authorization

Resist Attacks: Authenticate & Authorize Actors

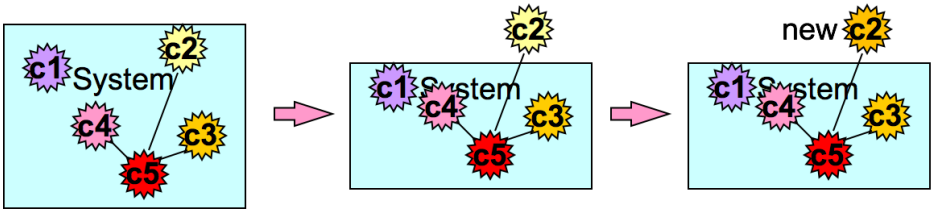
- Authenticate actors– assuring that an actor is actually who or what it says it is
  - Passwords
  - One-time passwords
  - Digital certificates
  - Biometric identification
  - ……
- Authorize actors– ensuring that an authenticated actor has the right to access/modify data or services
  - Access control by privileges or by roles

limit access
limit exposure

chapter 10 可测试性
基本概念
abstract data sources
component replacement

Replacement Techniques for Testability

- Component replacement: (used with "separate interface from implementation") replace component with a different implementation

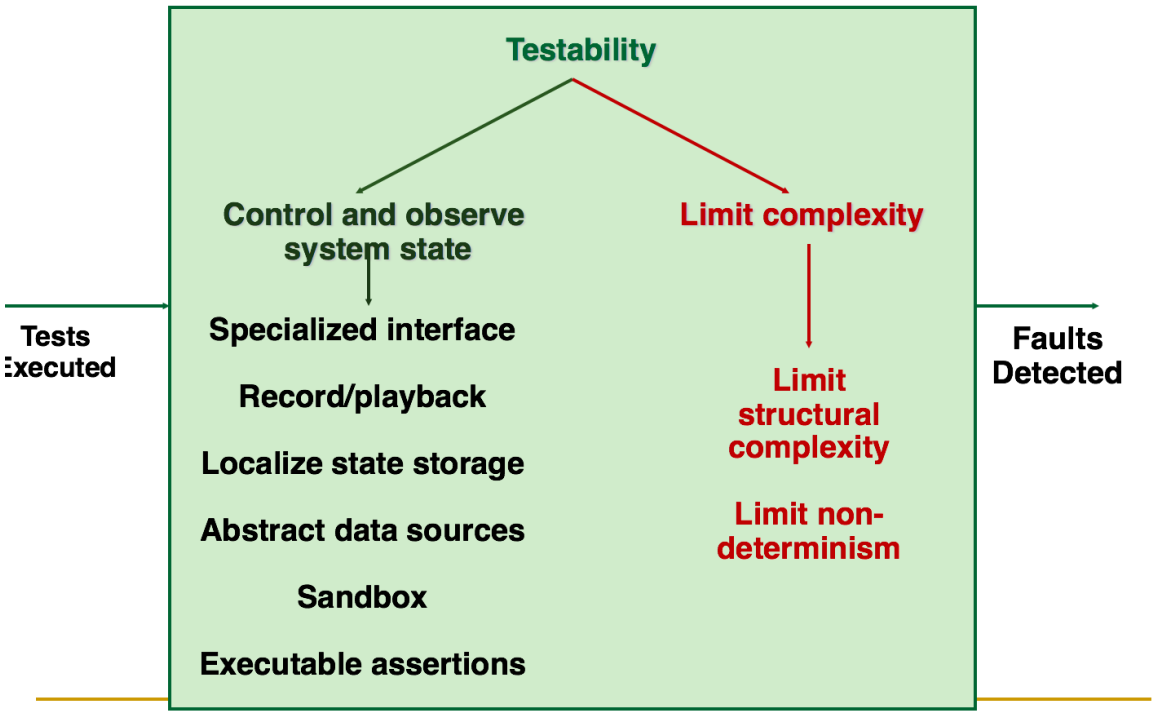


- Preprocessor macros: when activated, will enable state-reporting or probe statements that return or display information, or return control to a testing console

limit structural complexity

在提升可修改性的同时也会提高可测试性

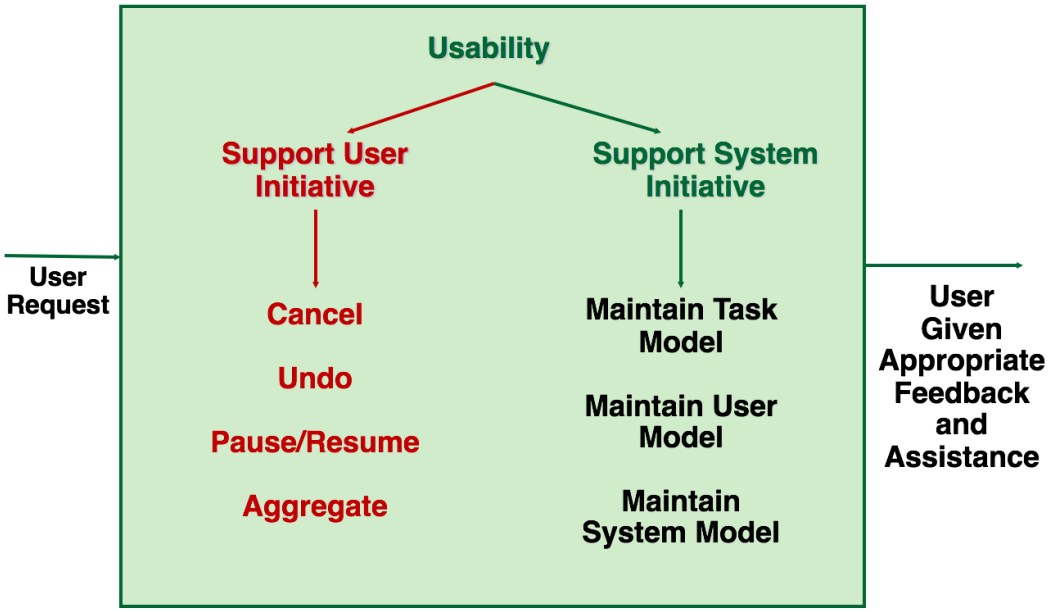
Testability Tactics Hierarchy



Human Computer Interactions

- User initiative – user leads system by issuing requests
  - Command interfaces
- System initiative – system leads the user; asking for input
  - “Wizards”
- Mixed initiative - dialog systems
  - E.g. a user cancels an operation
    - When canceling a command, the user issues a cancel---user initiative
    - During the cancel, the system may put up a progress indicator---system initiative

Usability Tactics Hierarchy



Support System Initiative (2)

- **Maintain user model**: modeling different aspects of the user
  - User's knowledge of the system
  - User behavior in terms of expected response time
  - User preference…
  - E.g. a model can control the amount of assistance and suggestions automatically provided to a user
- **Maintain system model**: the system maintains an explicit model of itself
  - Modeling system behavior so that appropriate feedback can be given to the user
  - E.g. a progress bar

设计题：

实时性系统，在某个ddl之前作出响应（测控们开启系统）设计方法，有资源抢占（调度）、其他算法实现

chapter 12，13， 14
设计+分析策略的优缺点

When Do We Start Designing the Software Architecture?

- Requirements come first
  - But not all requirements are necessary to get started
- An architecture shaped by some:
  - Functional requirements
  - Quality requirements
  - Business requirements
- We call these requirements “architecturally significant requirements (ASRs)
  - Requirements that will have profound effects on the architecture
  - Often takes the form of quality attribute requirements

chapter 14 多层的电商应用

How the WWW Achieved Its Initial Quality Goals

Goal	How Achieved	Tactics Used
Remote access	Build Web on top of Internet	Adherence to defined protocols
Interoperability	Use libWWW to mask platform details	Abstract Common Services Hide Information (Encapsulate)
Extensibility of software	Isolate protocol and data type extensions in libWWW; Plug-ins	Abstract Common Services, Hide Information, replace components (Defer binding), config files (Defer binding)
Extensibility of data	Make each data item independent except for references it controls	Restrict Dependencies
Scalability	Use client-server architecture and keep references to other data local to the referring data location	Introduce concurrency Reduce computational overhead

答疑时间：周日下午14.00~16.00 地点：曹楼517