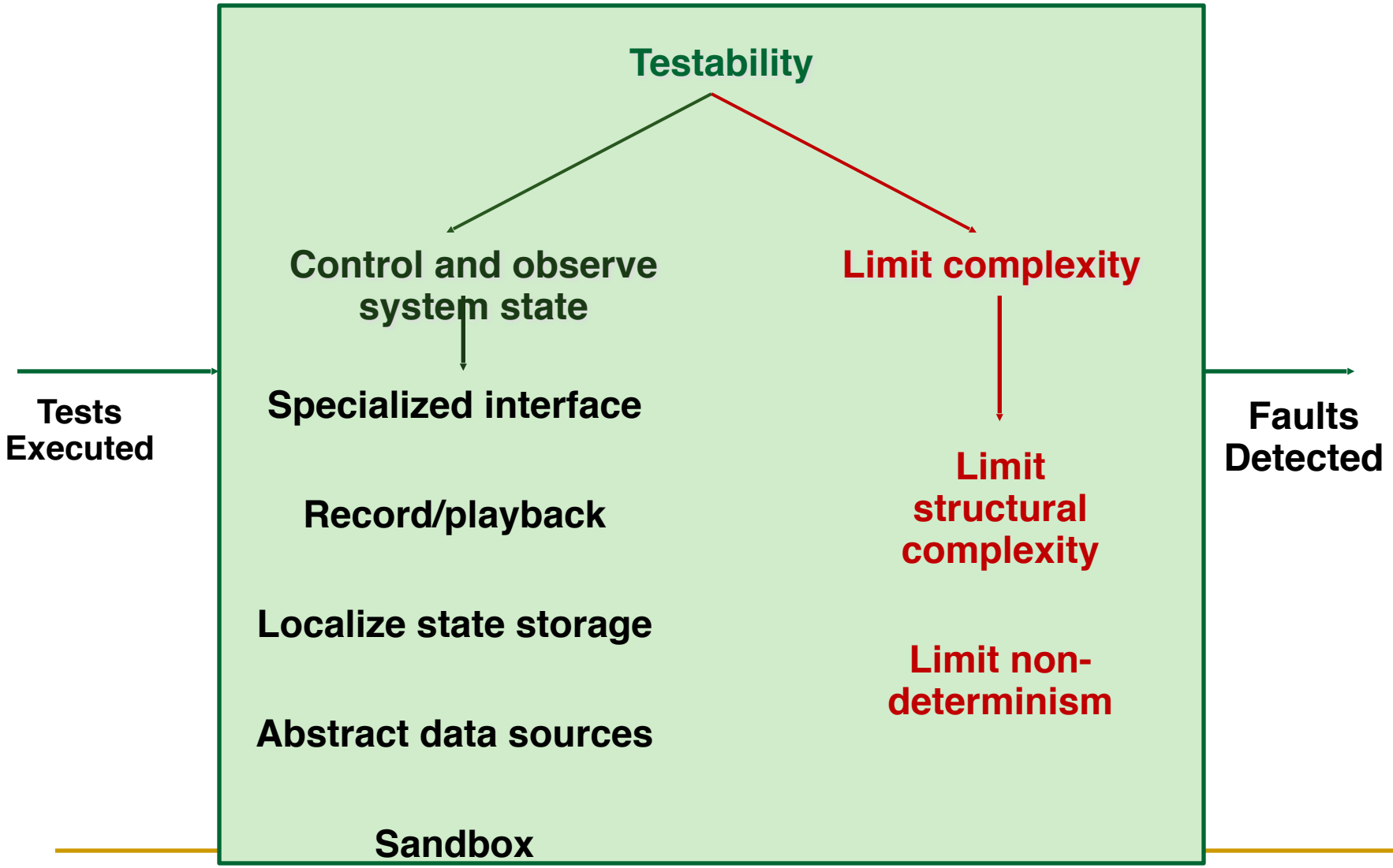
Tactics for Testability (2)

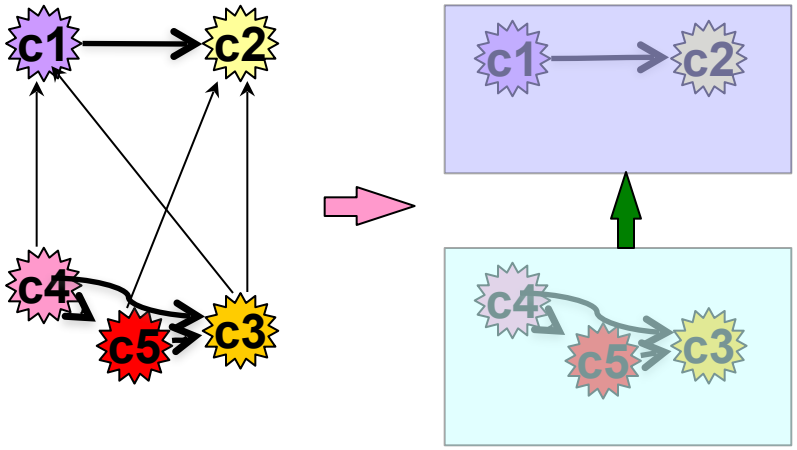
主讲教师：王灿
Email: wcan@zju.edu.cn
TA: 李奇平 liqiping1991@gmail.com
Course FTP: ftp://sa:sa@10.214.51.13

Testability Tactics Hierarchy



Limit Structural Complexity (1)

- Avoiding cyclic dependencies between components
- Isolating and encapsulating dependencies on the external environment
- Reducing dependencies between components in general



Limit Structural Complexity (2)

- In object-oriented systems
  - Inheritance hierarchy:
    - Number of classes derived from (multiple inheritance)
    - Number of classes derived
    - Depth of the inheritance tree
  - Polymorphism and dynamic calls
  - Method invocation: ***response*** of a class
    - The number of methods of a class plus the number of methods of other classes that invoked by this class

Limit Structural Complexity (3)

- Using modifiability tactics such as high cohesion, loose coupling, and separation of concerns etc. can effective reduce the complexity of the architectural elements
  - Giving each element a focused task with limited interaction with other elements

Limit Non-determinism

- Nondeterministic systems are harder to test than deterministic systems
  - This tactic involves finding all the sources of non-determinism, such as unconstrained parallelism, and weeding them out as much as possible

Architectural Design Support for Testability

- We check the architectural design and analysis process for testability from the following 3 aspects:
  1. Allocation of responsibilities
  2. Coordination model
  3. Data model
  4. Management of resources
  5. Mapping among architectural elements
  6. Binding time decisions
  7. Choice of technology

Allocation of Responsibilities

- Make sure the allocation of functionality provides high cohesion, low coupling, strong separation of concerns, and low structural complexity
- Support in system responsibilities include:
  - Execute test suite and capture results
  - Capture (log) the activity that resulted in a fault or that resulted in unexpected behavior
  - Control and observe relevant system state for testing

Coordination Model

- Execution of a test suite, capturing the results within a system or between systems
  - Capturing activity that resulted in a fault
- Injection and monitoring of state into the communication channels for use in testing, within a system or between systems
- Do not introduce needless non-determinism

Data Model

- Capturing the values of instances of the data abstractions
- The values of instances of the data abstractions can be set when state is injected into the system
- The creation, initialization, persistence, manipulation, translation, and destruction of instances of these data abstractions can be exercised and captured

Resources Management

- Ensure sufficient resources available to execute a test suite and capture the results
- Ensure system supports for:
  - Test resource limits
  - Capture detailed resource usage
  - Provide virtualized resources for testing

Binding Time Decisions

- Ensure that components that are bound later than compile time can be tested in the late-bound context
- Ensure that late bindings can be captured in the event of a failure, so that you can re-create the system's state leading to the failure
- Ensure that the full range of binding possibilities can be tested

Tactics for Usability

Usability

- Usability is about how easy it is for the user to accomplish a desired task and the kind of support the system provides to the user.

- Dimensions of usability:
  - Learning system features
  - Using the system efficiently
  - Minimizing the impact of errors
  - Adapting the system to the user's needs
  - Increasing confidence and satisfaction

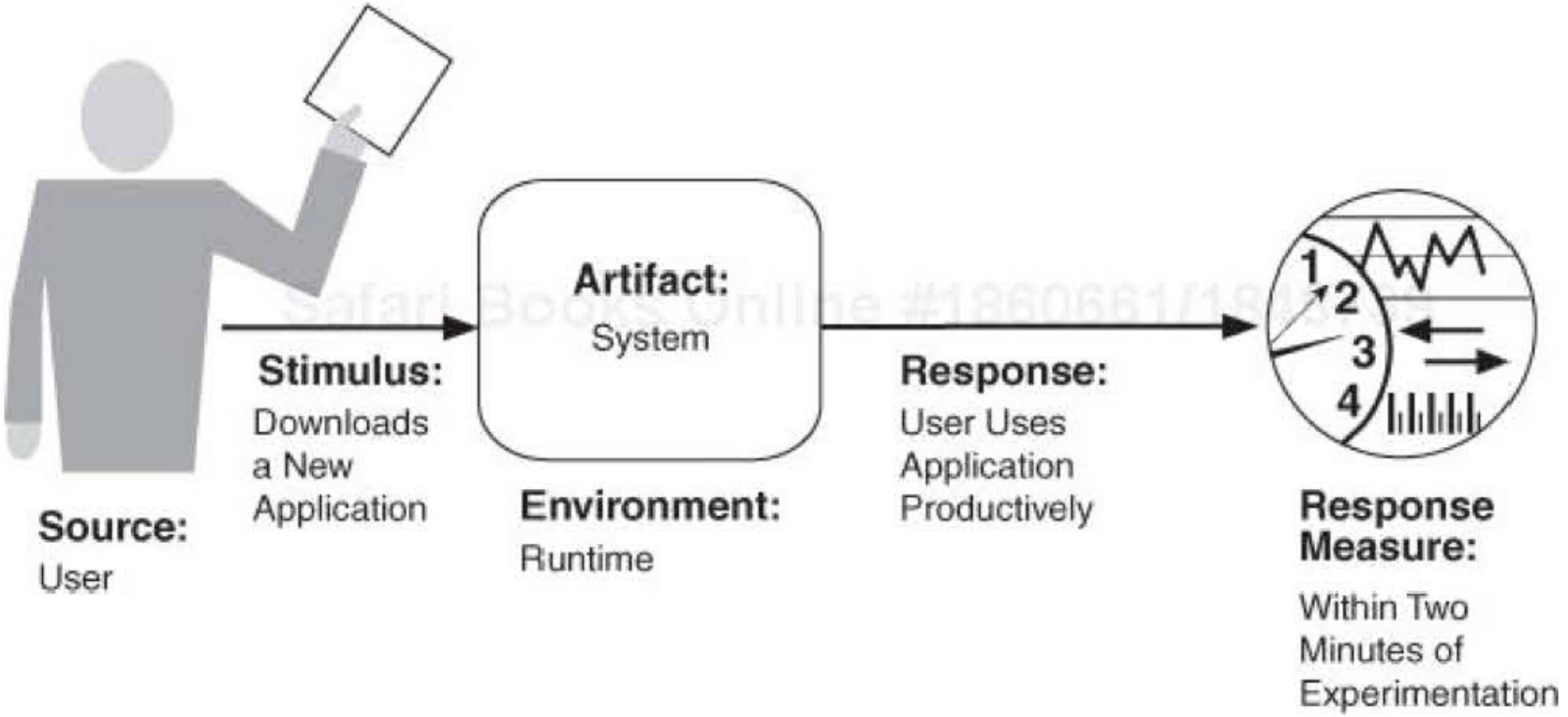
Usability General Scenario (1)

- Stimulus
  - An end user tries to use a system efficiently, learn to use the system, minimize the impact of errors, adapt the system, or configure the system
- Source of stimulus
  - End user, possibly in a specialized role
- Response
  - The system should either provide the user with the features needed or anticipate the user's needs

Usability General Scenario (2)

- Response Measure
  - Task time, number of errors, number of tasks accomplished, user satisfaction…
- Artifact
  - System, or a specific portion of the system with which the user is interacting
- Environment
  - At runtime or configuration time

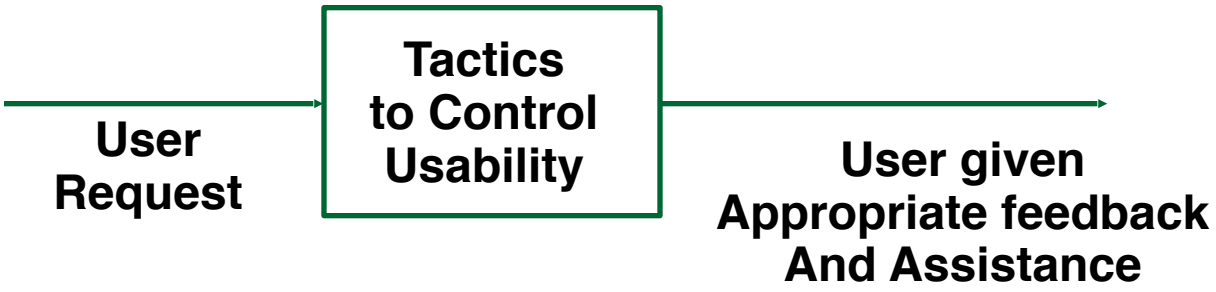
A Sample Concrete Usability Scenario



Human Computer Interactions

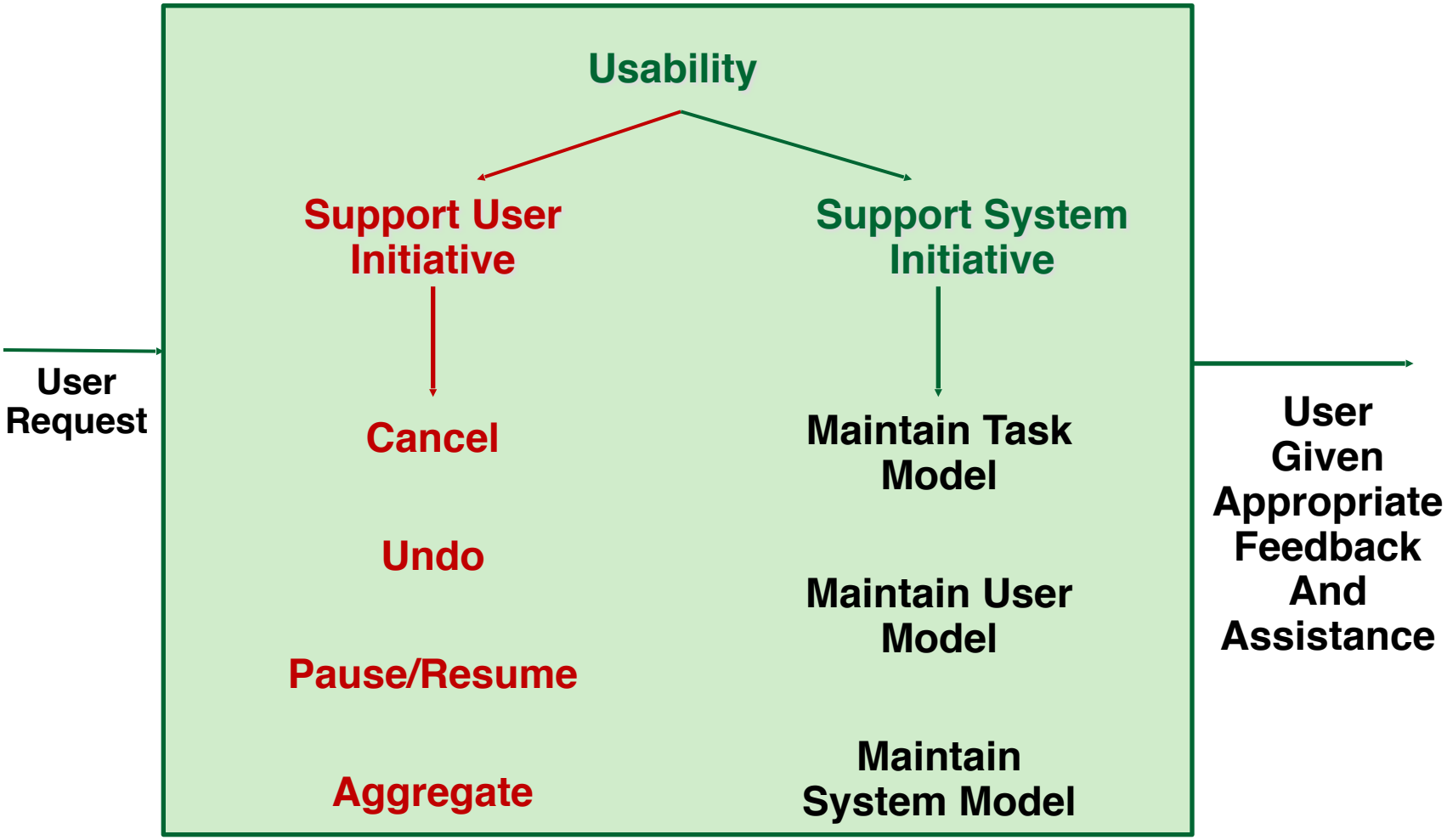
- User initiative – user leads system by issuing requests
  - Command interfaces
- System initiative – system leads the user; asking for input
  - “Wizards”
- Mixed initiative - dialog systems
  - E.g. a user cancels an operation
    - When canceling a command, the user issues a cancel---user initiative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

Usability Tactics



- Support user initiative: correcting user errors or improving user efficiency
- Support system initiative: rely on a model of the user, the task or the system state itself

Usability Tactics Hierarchy



Cancel

- When the user issues a cancel command:
  - The system must be listening for it
    - A constant listener that is not blocked by the actions of whatever is being canceled
  - The command being canceled must be terminated
  - Any resources being used by the canceled command must be freed
  - Components that are collaborating with the canceled command must be informed so that they can also take appropriate action

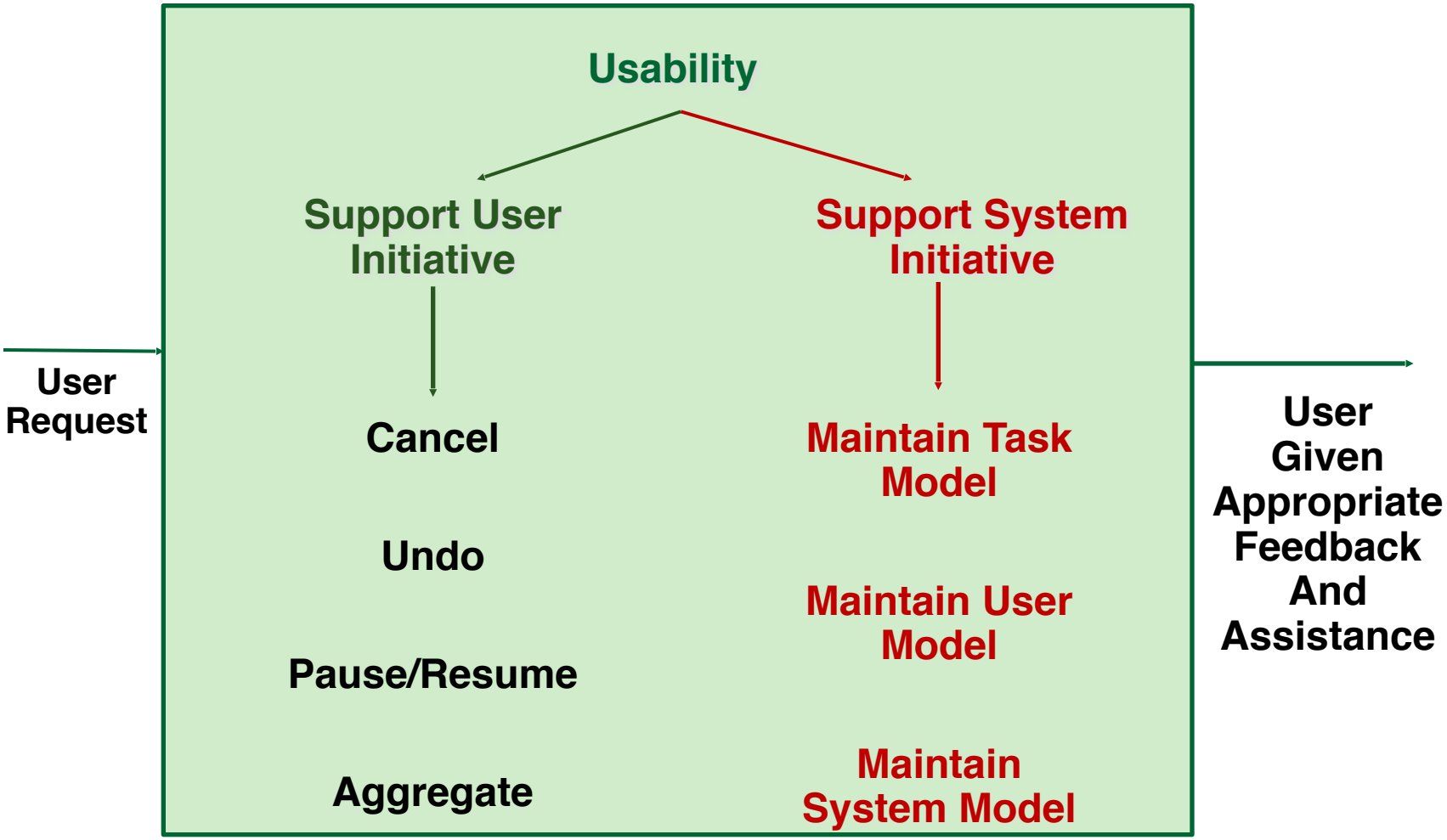
Undo

- To support undo, the system must maintain a sufficient amount of information about system state so that an earlier state may be restored
  - E.g. using checkpoints + rollback to reverse operations
- Not all operations can be easily reversed
- Some operations cannot be undone, such as ringing a bell

Pause/Resume & Aggregate

- Pause/resume: In a long-running operation, provide the ability to pause and resume the operation can improve usability
  - Effectively pausing a long-running operation requires the ability to temporarily free resources
- Aggregate: when performing repetitive operations, or operations that affect a large number of objects in the same way, aggregate the objects into a group and apply the operation to the group
  - Decrease error and improve efficiency

Usability Tactics Hierarchy



Support System Initiative (1)

- When the system takes the initiative, it relies on models of the user, the task or the system its self so as to predict either its own behavior or the user's intention

- Maintain task model: Model the context of the task being undertaken by the user so as to learn user’s attempts and provide assistance
  - E.g. knowing that sentences start with capital letters would allow an application to correct a lowercase letter in that position

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

Architectural Design Support for Usability

- We check the architectural design and analysis process for modifiability from the following 5 aspects:
  1. Allocation of responsibilities
  2. Coordination model
  3. Data model
  4. Management of resources
  5. Mapping among architectural elements
  6. Binding time decisions
  7. Choice of technology

Allocation of Responsibilities

- Ensure system responsibilities to assist users in:
  - Learning how to use the system
  - Efficiently achieving the task at hand
  - Adapting and configuring the system
  - Recovering from user and system errors

Coordination Model

- Determine how coordination model's properties affect usability
  - E.g., can the system respond to mouse events and give semantic feedback in real time?
  - Can long-running events be canceled in a reasonable amount of time?

Data Model

- Focus on data abstractions involved with user-perceivable behaviors
  - E.g., can the data abstractions support undo and cancel operations?
  - Should the transaction granularity be so great that canceling or undoing an operation takes an excessively long time?

Resources Management

- Pay attention to resource management involved with user-controlled configurations
  - E.g., avoid configurations that would result in excessively long response times

Choice of Technology

- The chosen technology shall aid in the creation of online help and tutorial, and the collection of user feedback

Reading Assignment

- Browse through Chapter 16 to learn practices of gathering ASRs
- Read Chapter 17 of the textbook.