SENG 350
- Software Architecture & Design

Shuja Mughal

Architectural Tactics

Fall 2024

University
of Victoria

Achieving Qualities

- What is Quality?
- How can we achieve quality?
- What are the Non-Functional Requirements?

Decisions we to focus on

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

Non-Functional Requirements

Accessibility	Efficiency	Reliability	Responsiveness
Availability	Extensibility	Robustness	Scalability
Compatibility	Failure Transparency	Safety	Security
Compliance	Fault Tolerance	Maintainability	Supportability
Configurability	Interoperability	Operability	Testability
Dependability	Performance	Traceability	Usability
Documentation	Recoverability	Disaster Recovery	Flexibility

University of Victoria

Architectural Tactics

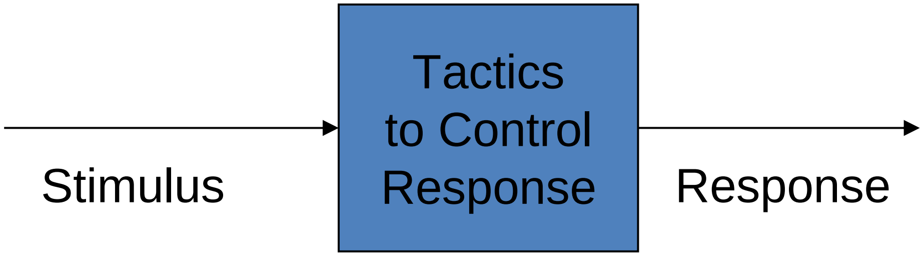
- A tactic is a design decision that influences the control of a quality attribute response.
- A collection of tactics is called an architectural strategy.
- A system design is a collection of decisions
  - Some ensure achievement of the system functionality
  - Others help control the quality attribute responses (which we call the tactics)

Architectural Tactics (Cont’d)

- Tactics can refine other tactics—e.g., redundancy is a tactic for achieving availability, and it can be refined into redundancy of data or redundancy of computation.
- Patterns package tactics—for example, a pattern might package redundancy and synchronization tactics (along with others).

University of Victoria

Architectural Tactics (Cont’d)



Stimulus

Tactics to Control Response

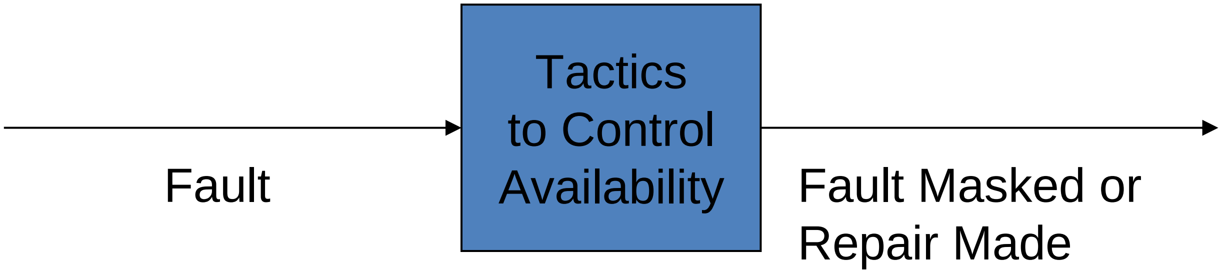
Response

Availability?

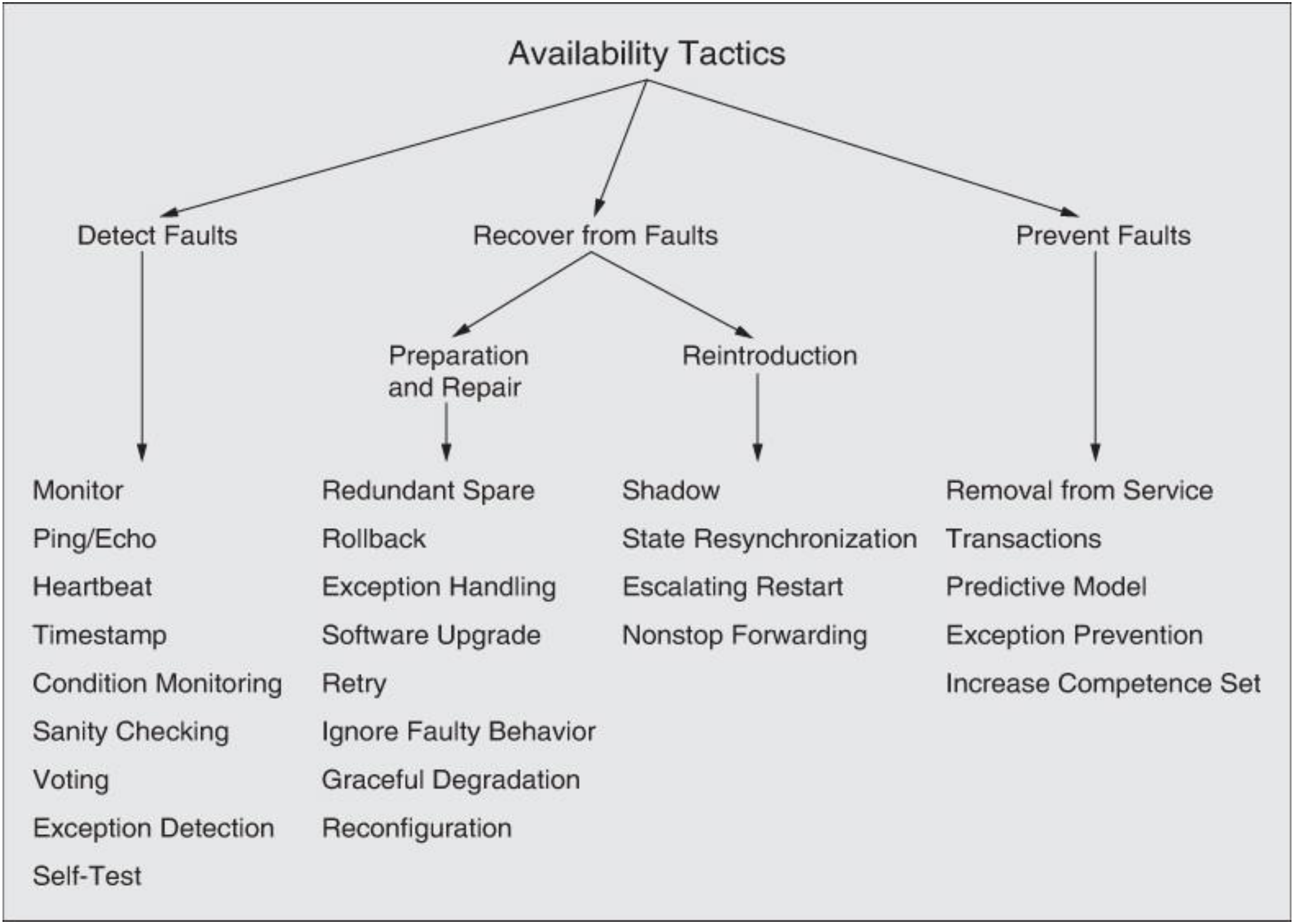
Availability Tactics

- All approaches to maintaining availability involve:
  - Some type of redundancy
  - Some type of health monitoring to detect a failure
  - Some type of recovery when a failure is detected (either automatic or manual).

Goal of Availability Tactics



Availability Tactics



Availability Tactics

Detect Faults

Recover from Faults

Prevent Faults

Preparation and Repair

Reintroduction

Monitor
Ping/Echo
Heartbeat
Timestamp
Condition Monitoring
Sanity Checking
Voting
Exception Detection
Self-Test

Redundant Spare
Rollback
Exception Handling
Software Upgrade
Retry
Ignore Faulty Behavior
Graceful Degradation
Reconfiguration

Shadow
State Resynchronization
Escalating Restart
Nonstop Forwarding

Removal from Service
Transactions
Predictive Model
Exception Prevention
Increase Competence Set