

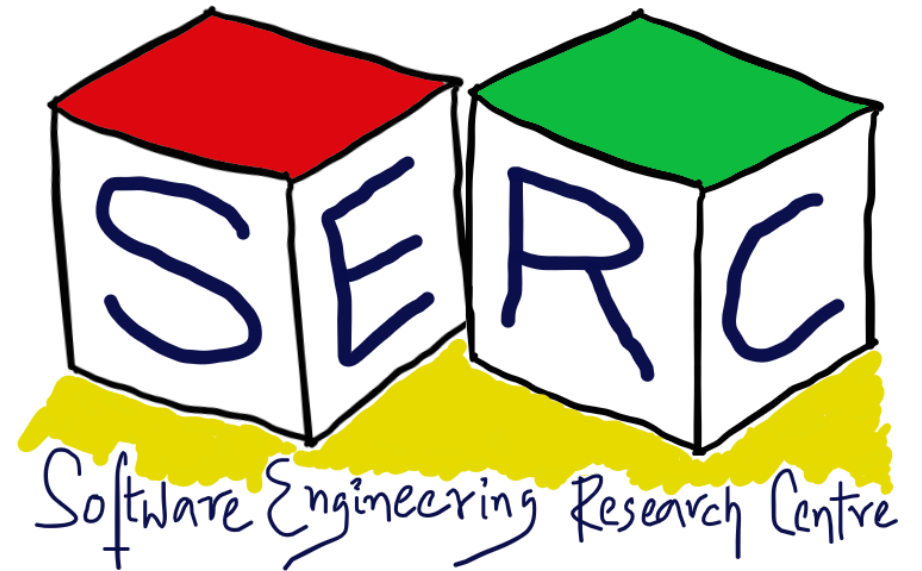
Architectural Tactics

CS6.401 Software Engineering

Dr. Karthik Vaidhyanathan

karthik.vaidhyanathan@iiit.ac.in

<https://karthikvaidhyanathan.com>



Acknowledgements

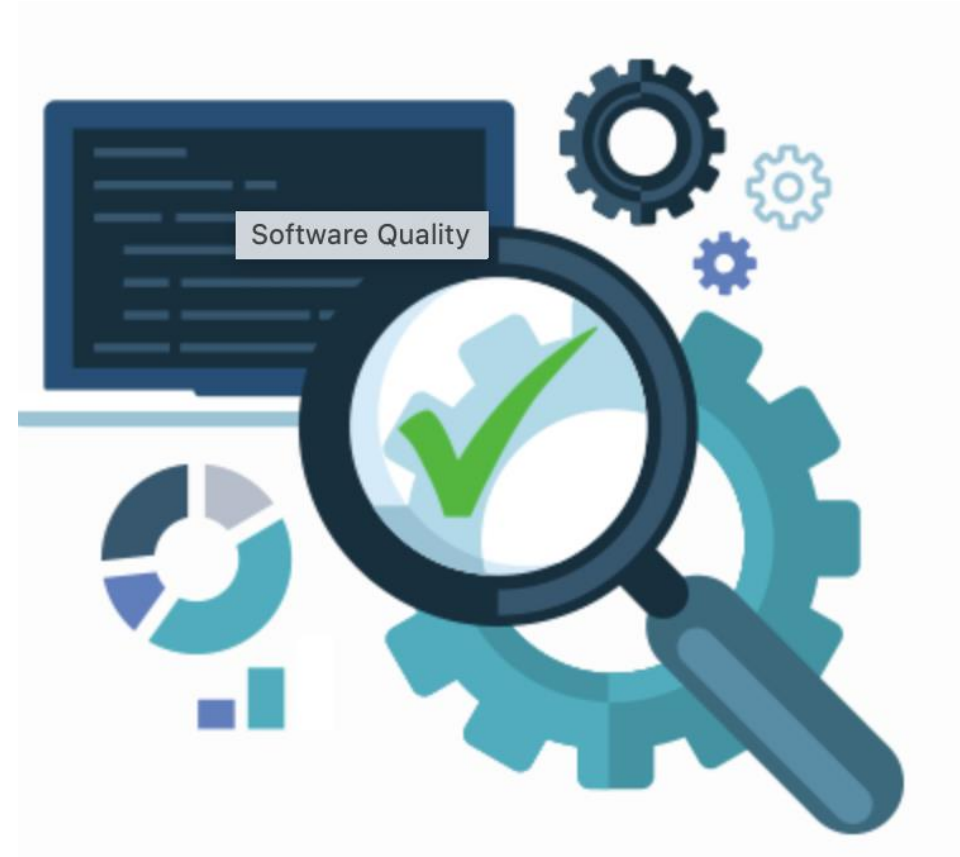
The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge -- Karthik Vaidhyanathan

Sources:

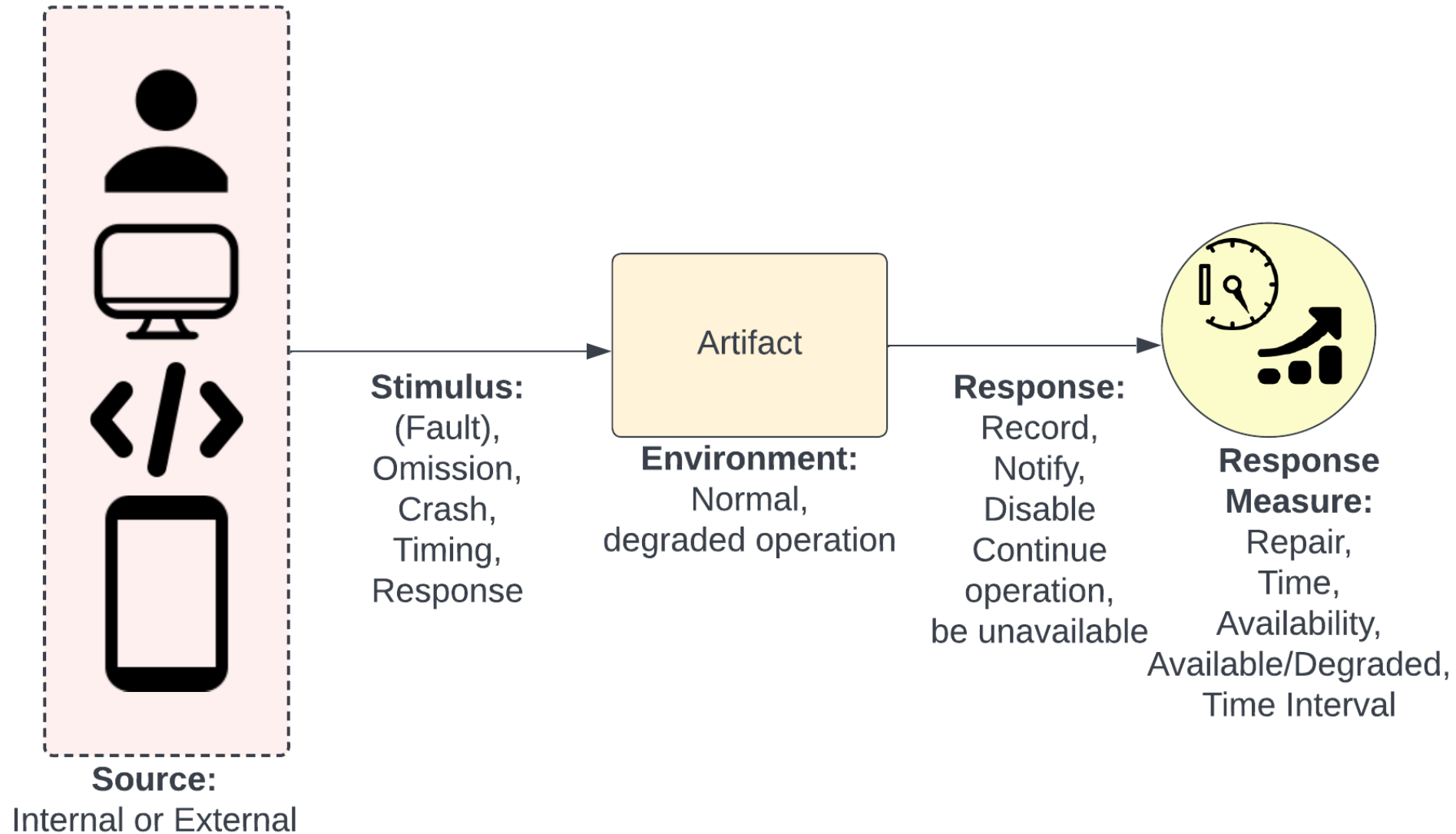
1. Software Architecture in Practice, Len Bass, 2nd, 3rd edition

System Qualities

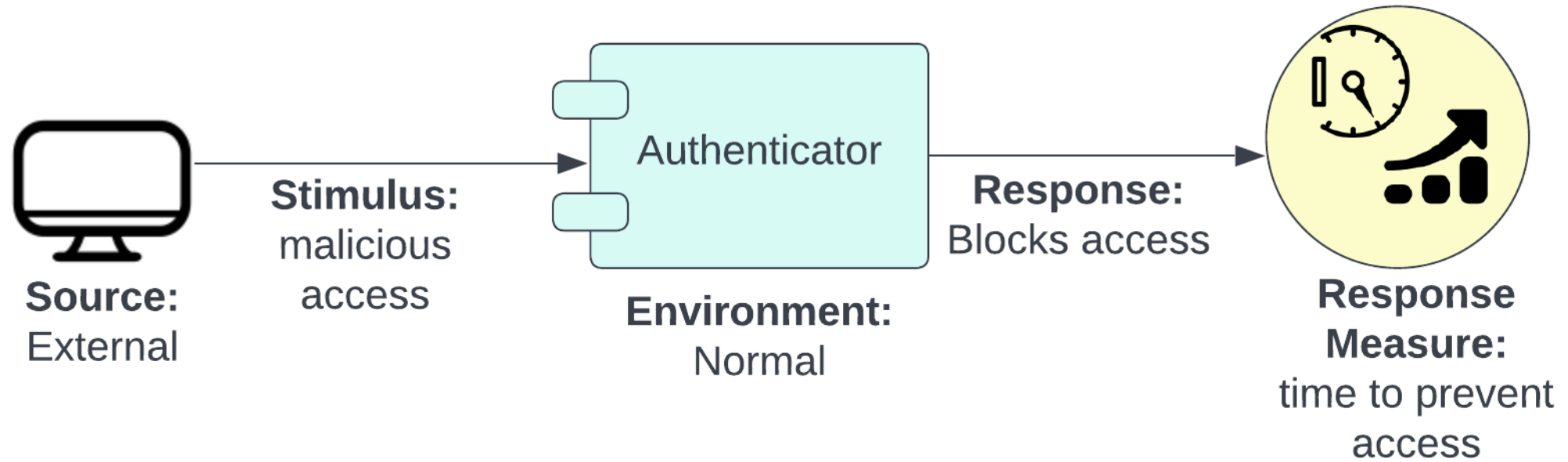
1. Availability
2. Security
3. Performance
4. Modifiability
5. Testability
6. Usability
7. Sustainability
-

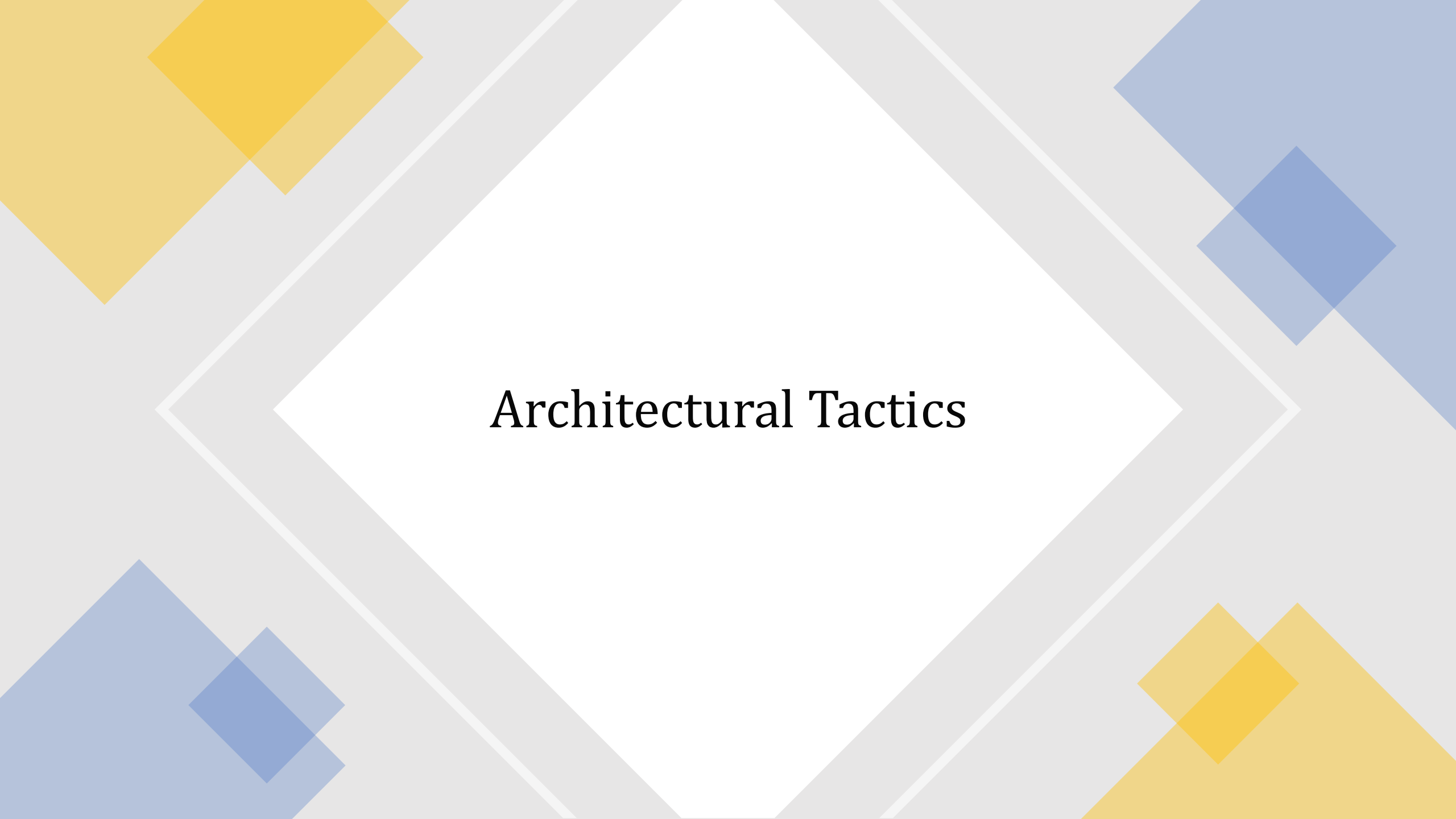


Quality Scenarios - General



Quality Scenarios - Concrete





Architectural Tactics

What is Tactic?



tactic

/ˈtæktɪk/

noun

plural noun: **tactics**

an action or strategy carefully planned to achieve a specific end.
"the minority attempted to control the Council by a delaying tactic"

What about Architectural Tactics?

“Characterization of architectural decisions that are used to achieve a desired quality attribute response”





Availability

Availability

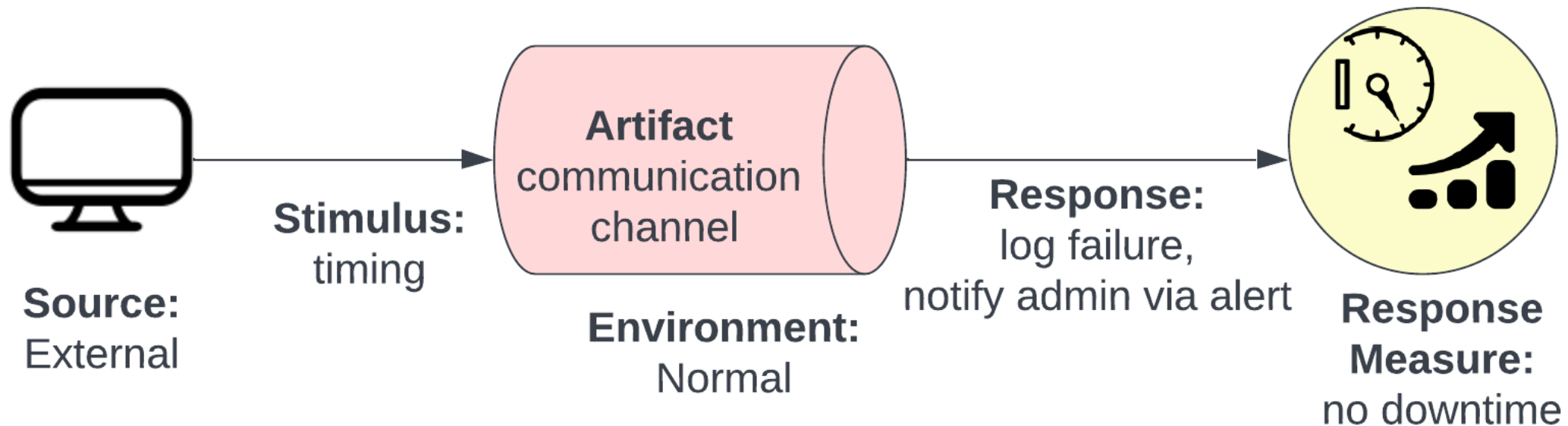
Software is ready to carry out a task when you need it to be

1. Concerned with system failures and associated consequences
2. Failure and fault are two different yet related things
3. Mean time to failure and mean time to repair
4. Scheduled downtimes

Availability %	Downtime per year
90% (one nine)	36.5 days
99% (two nines)	3.65 days
99.9% (three nines)	8.46 days
99.99% (four nines)	52.34 minutes
99.999% (five nines)	5.26 minutes
99.9999% (six nines)	32 seconds

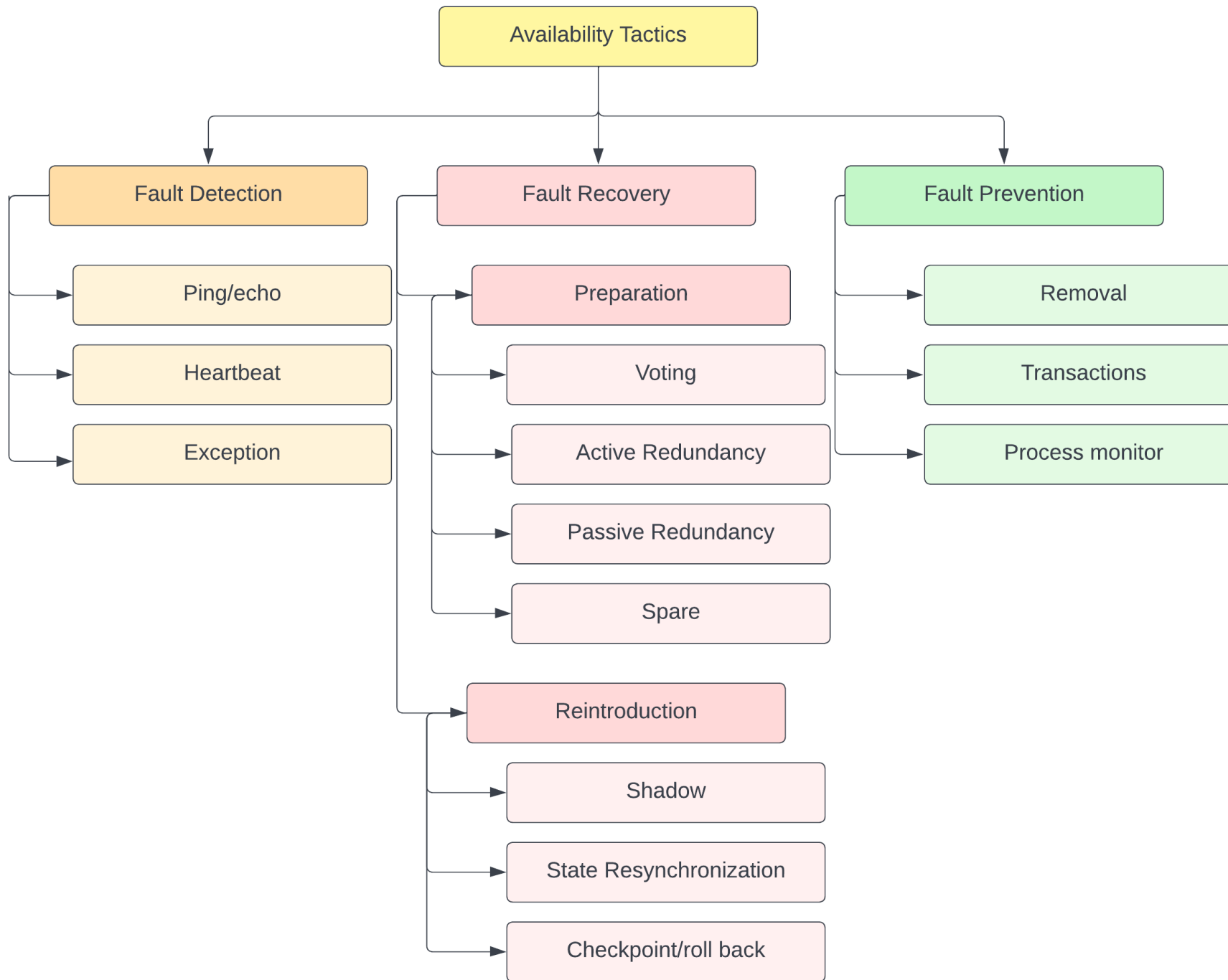
Availability Scenario Example

Scenario: user app fails to receive an acknowledgement from the museum booking service



Availability Scenarios Table

Name	Classes
Source of Stimulus	Internal and external
Stimulus	Type of fault (ommission, crash, timing, response)
Artifact	Processors, channels, storage
Environment	Normal or degraded mode
Response	Logging, notification, switching to backup, restart, shutdown
Response measure	Availability time, uptime, repair time





Performance

Performance

Performance is about timing. Events occur and system must respond to them

Event arrival patterns

1. Periodic
2. Stochastic
3. Sporadic

Event Servicing

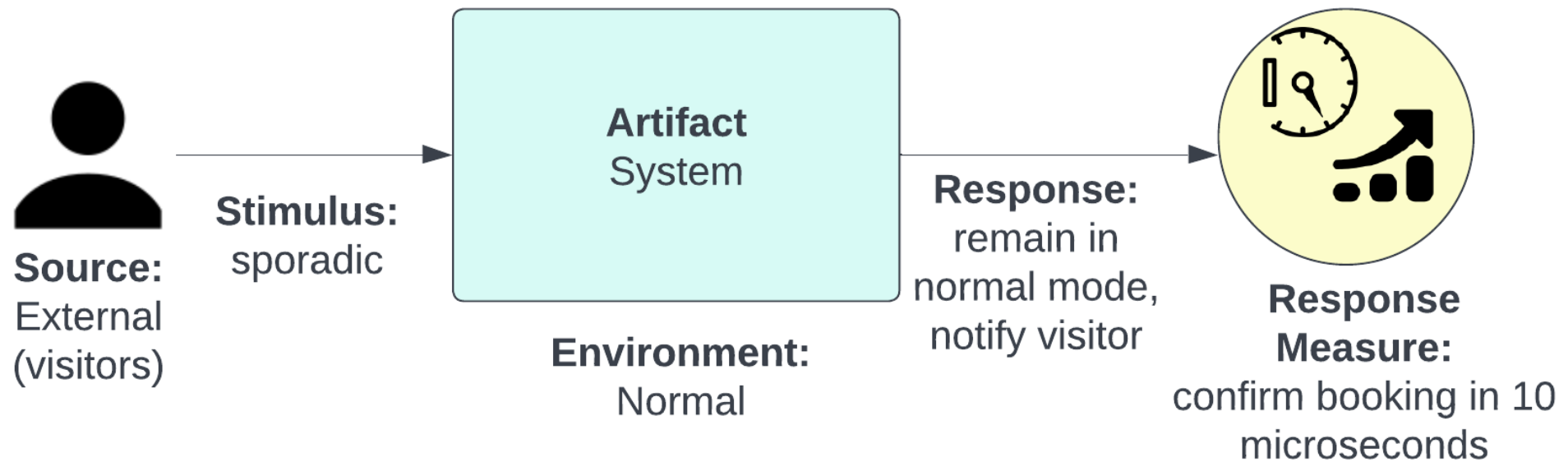
1. Latency - Time taken between arrival and response
2. Jitter – Variation in latency
3. Throughput – Number of requests processed per second

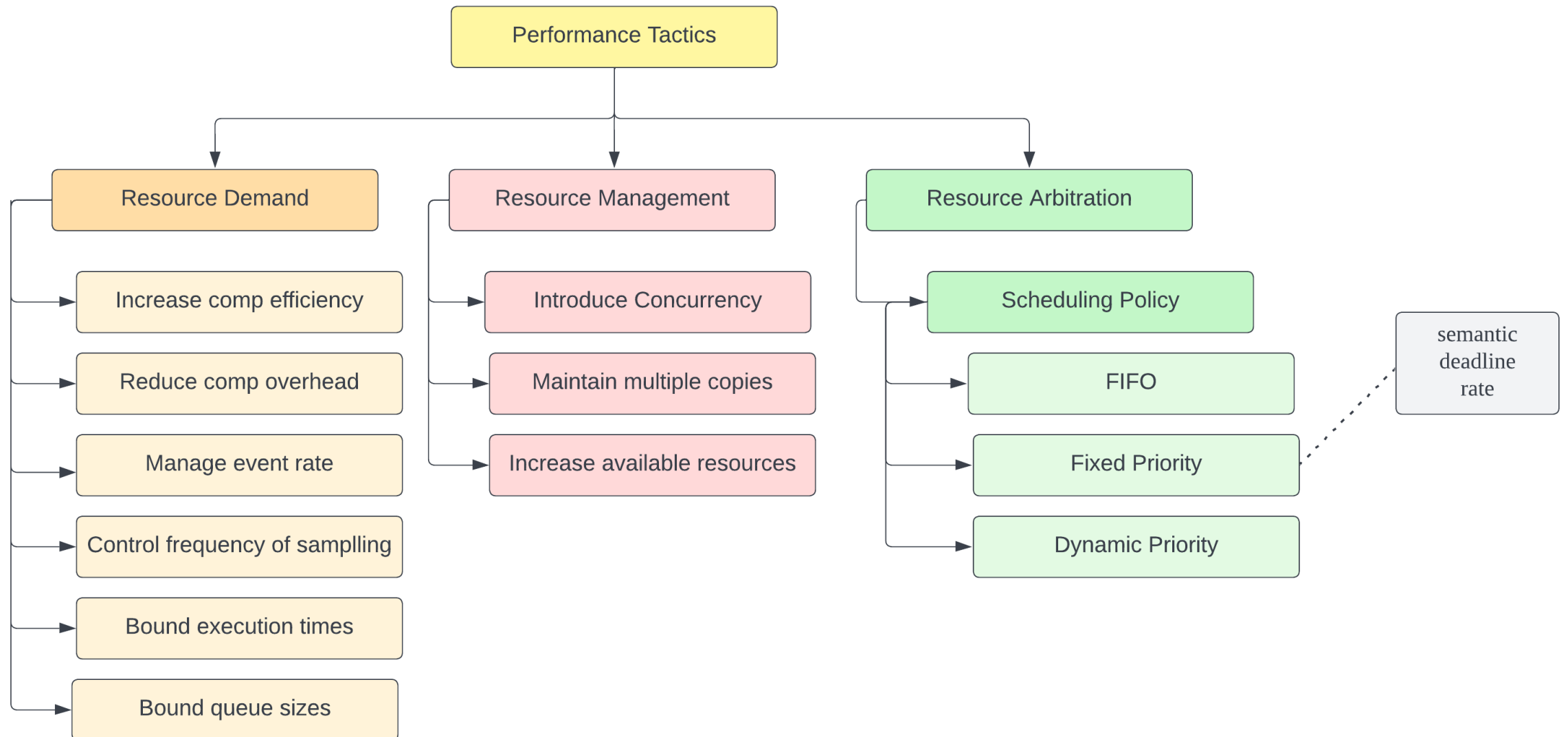
Performance Scenarios Table

Name	Classes
Source of Stimulus	Internal and external
Stimulus	Event arrival (based on pattern)
Artifact	System
Environment	Normal mode; Overload mode
Response	Changes level of service, processes stimuli
Response measure	Latency, throughput, jitter, miss rate, data loss

Performance Scenario Example

Scenario: Visitor needs to get notified as soon as booking is made







Security

Security

Measure of system's ability to resist unauthorized usage while still providing services to legitimate users

System providing:

1. Confidentiality
2. Integrity
3. Availability
4. Non-repudiation
5. Assurance
6. Auditing

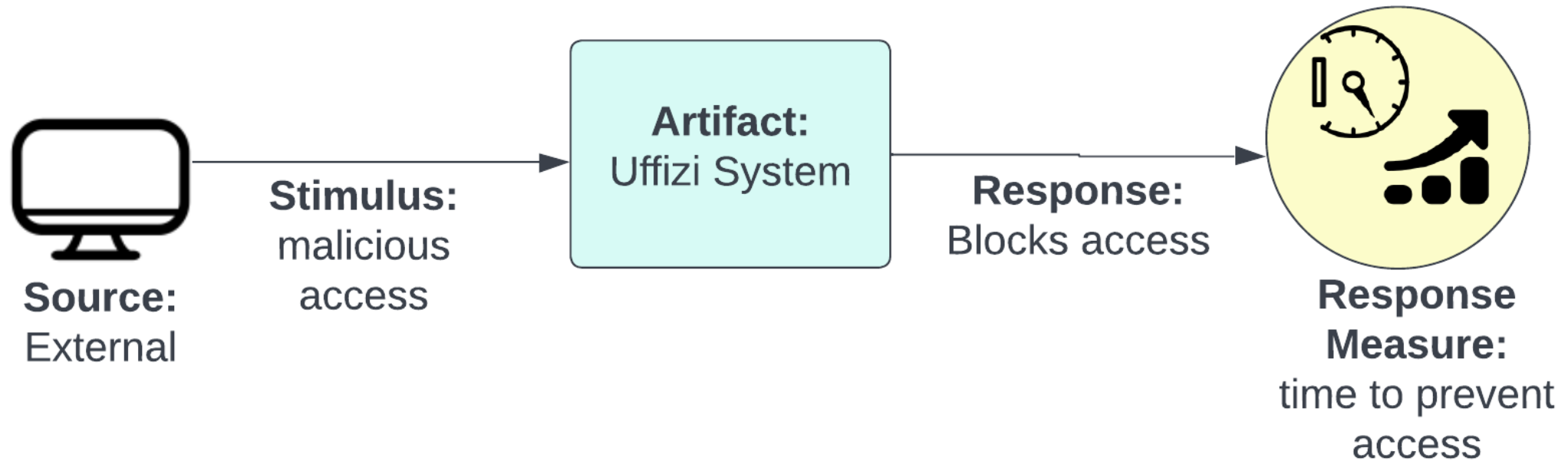


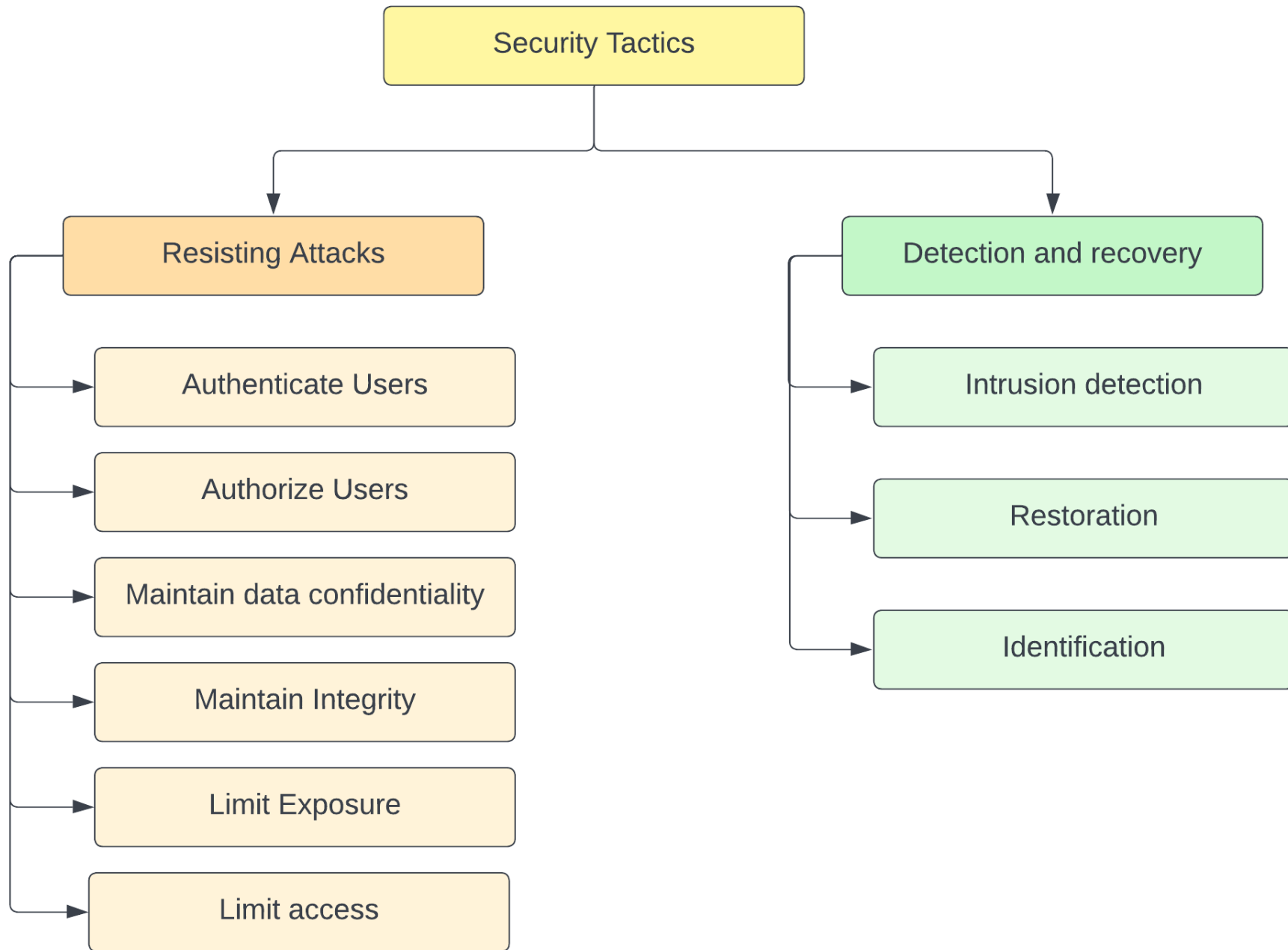
Security Scenarios Table

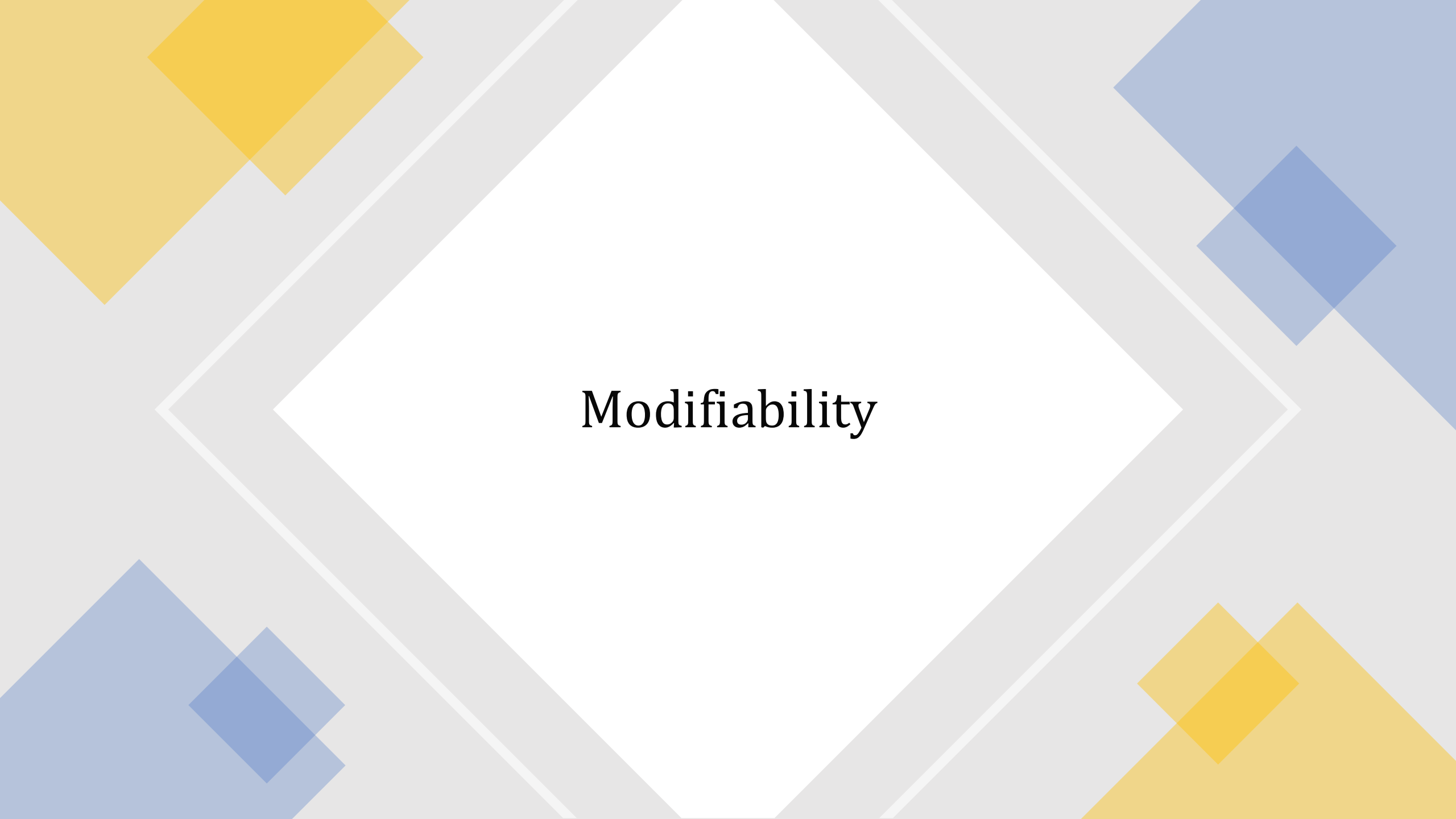
Name	Classes
Source of Stimulus	User, system, unidentified
Stimulus	Tries to Change/modify data, access system services
Artifact	System services, data within
Environment	Open, firewalled, online, offline
Response	Logging, block access, notify
Response measure	Probability of detection, recovery

Security Scenario Example

Scenario: Hackers are prevented from disabling the system







Modifiability

Modifiability

Modifiability is about the cost of change. It brings in few concerns:

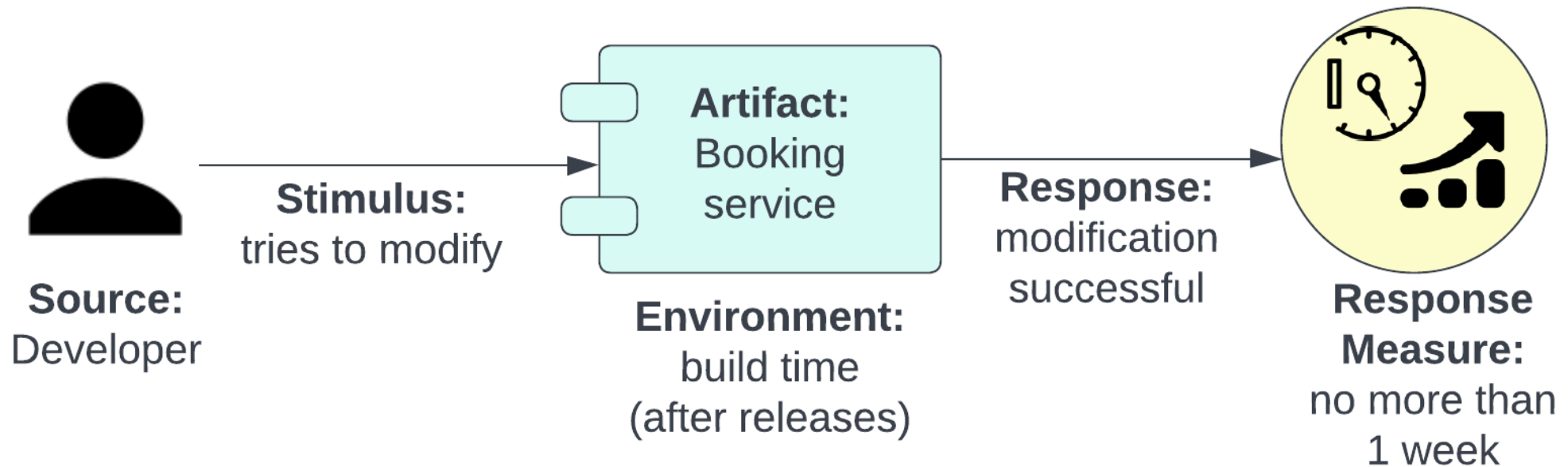
1. What can change (the artifact) ?
2. When is the change made ?
3. Who makes it (the environment) ?

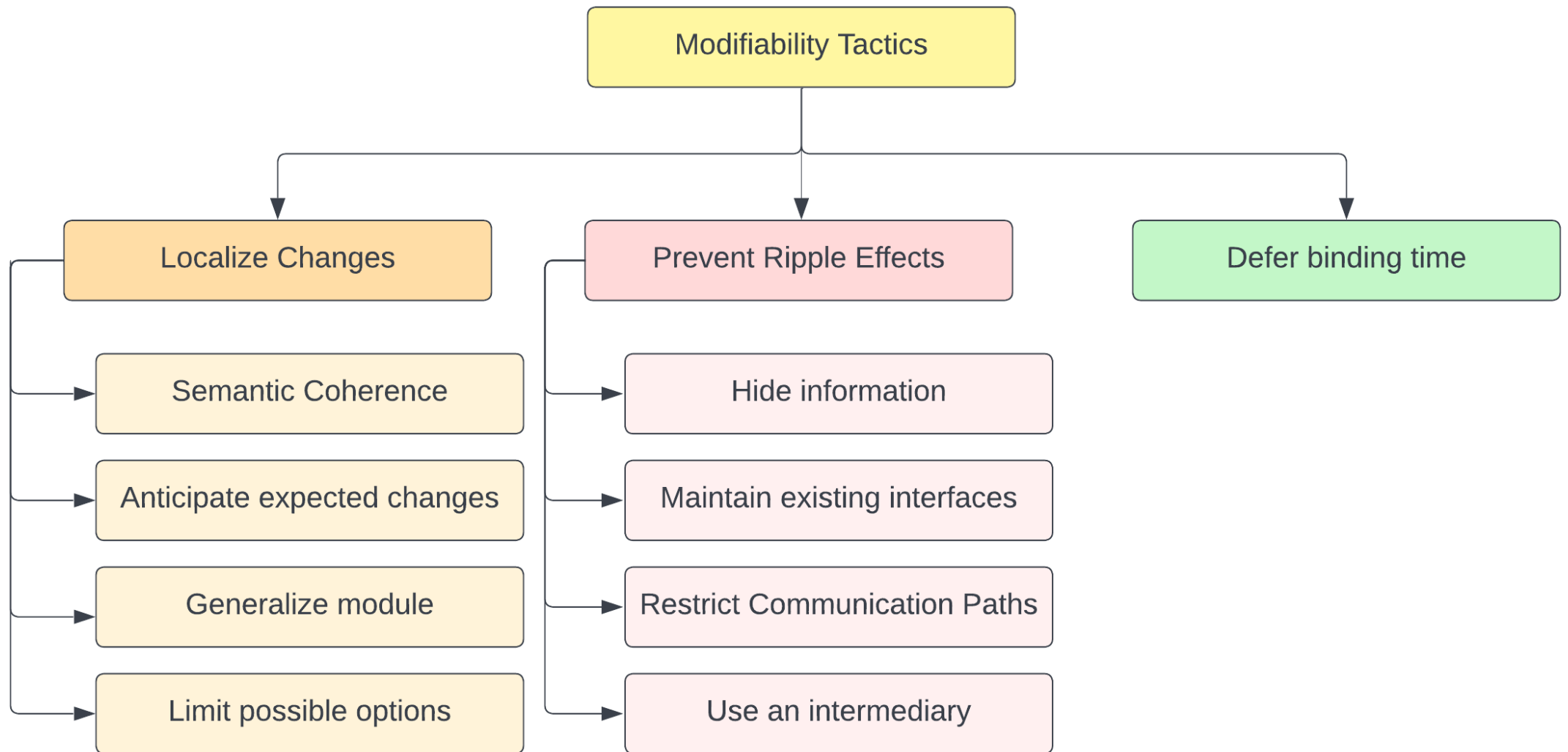
Modifiability Scenarios Table

Name	Classes
Source of Stimulus	Developer, admin, user
Stimulus	Add/modify function
Artifact	UI, Platform, environment
Environment	Runtime, compile time, build time, design time
Response	Make changes and verify
Response measure	Effort, time, cost

Modifiability Scenario Example

Scenario: Developer wants to change booking service







Testability

Testability

Ease with which the software can be made to demonstrate its faults

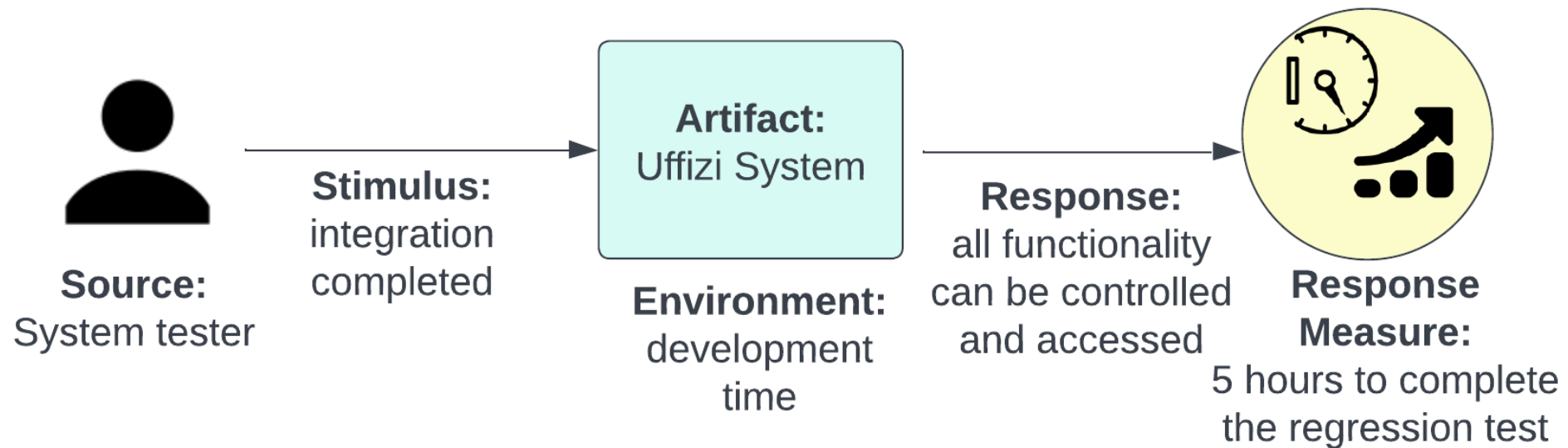
1. Probability of fault discovery
2. Need to control components
3. Component failures should be observable

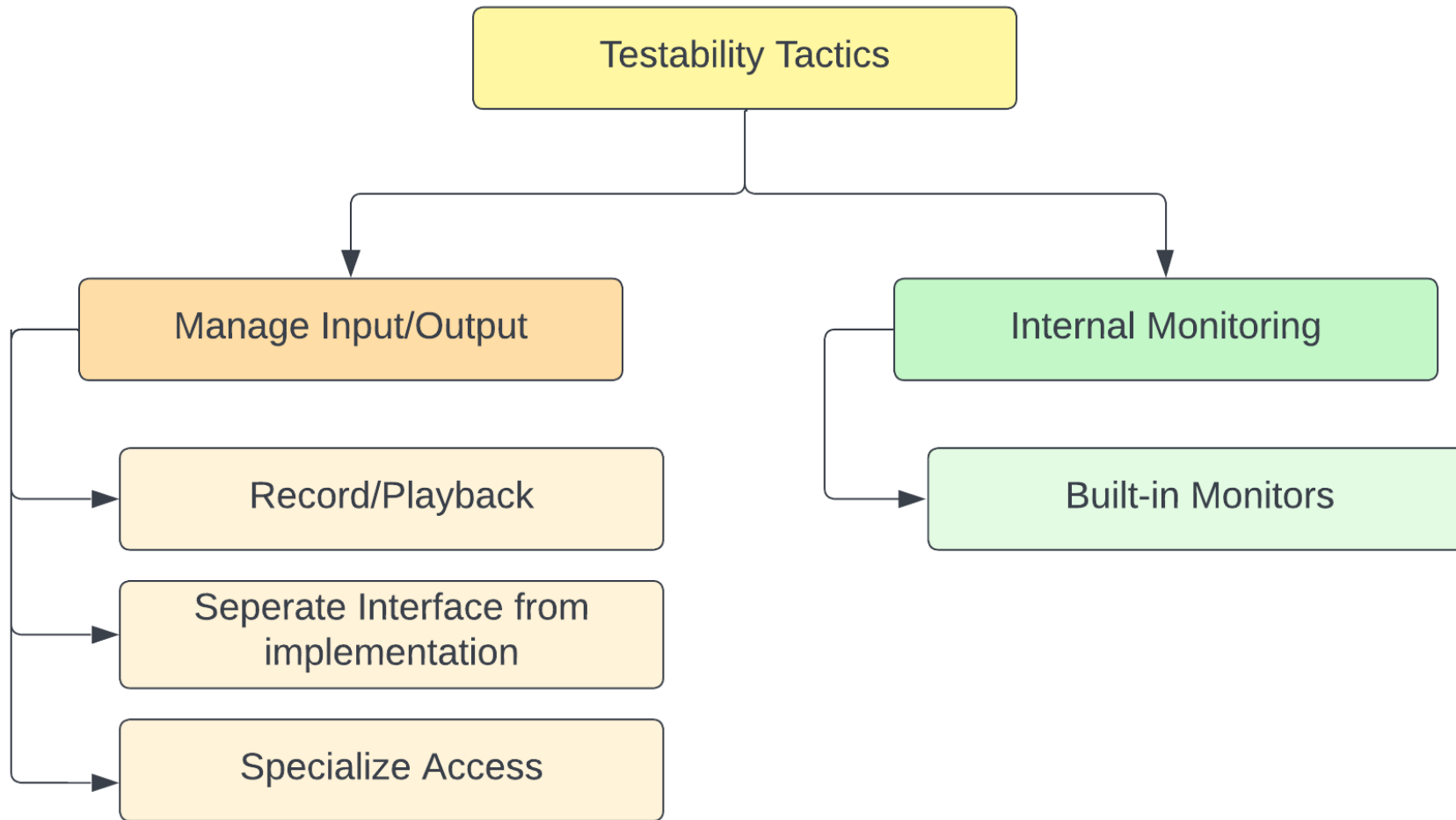
Testability Scenarios Table

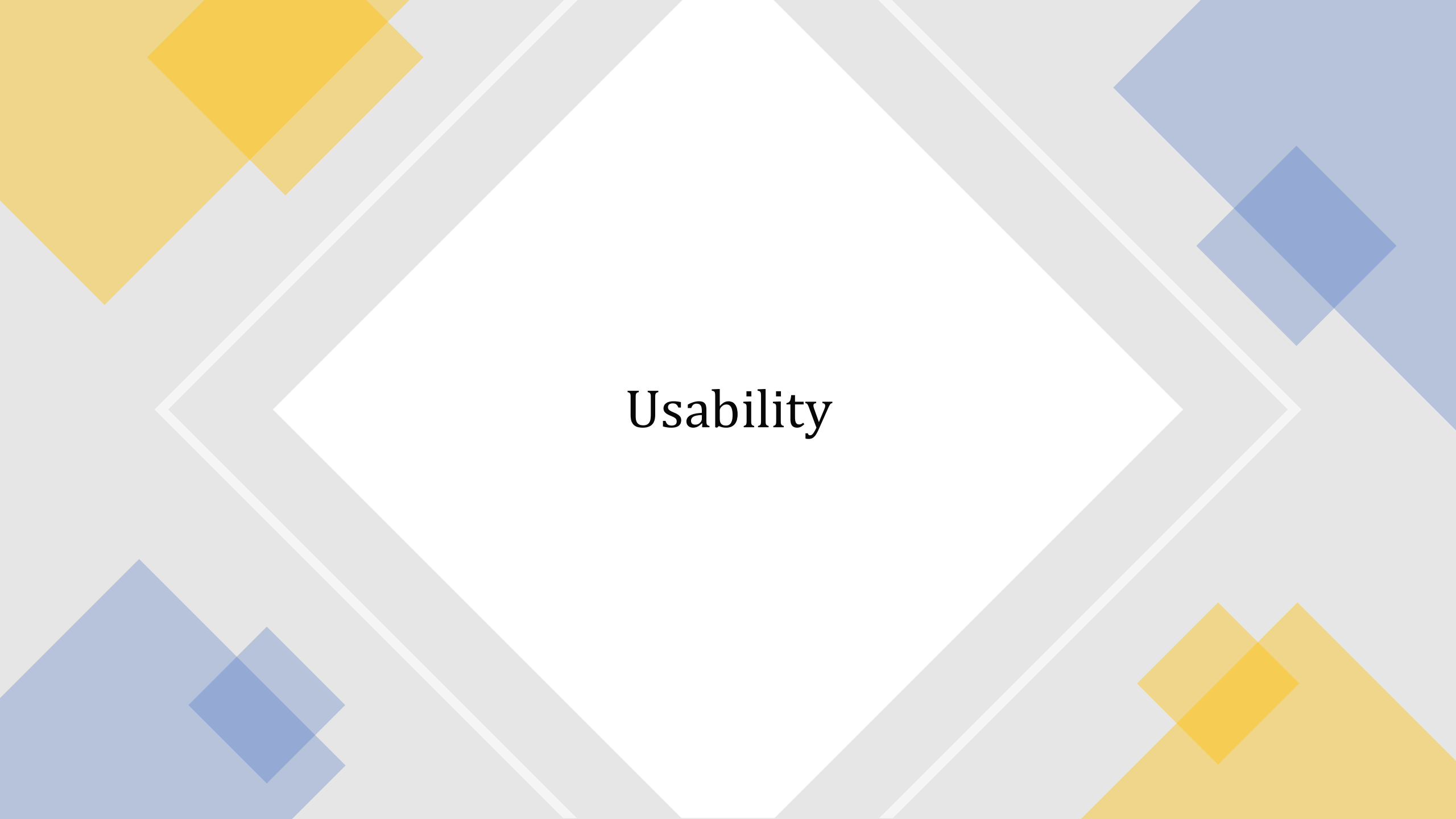
Name	Classes
Source of Stimulus	Developer, tester, user
Stimulus	Milestone completed
Artifact	Design, piece of code, component, system
Environment	Design, development, at compile deployment, run
Response	State values, computes test values
Response measure	Coverage, probability, time, length of longest dependancy

Testability Scenario Example

Scenario: New version of the system can be tested quickly







Usability

Usability

How easy it is for user to accomplish a desired task and the kind of support the system provides

1. Learning system features
2. Using a system efficiently
3. Minimizing the impact of user errors
4. Adapting system to user needs
5. Increasing confidence and satisfaction

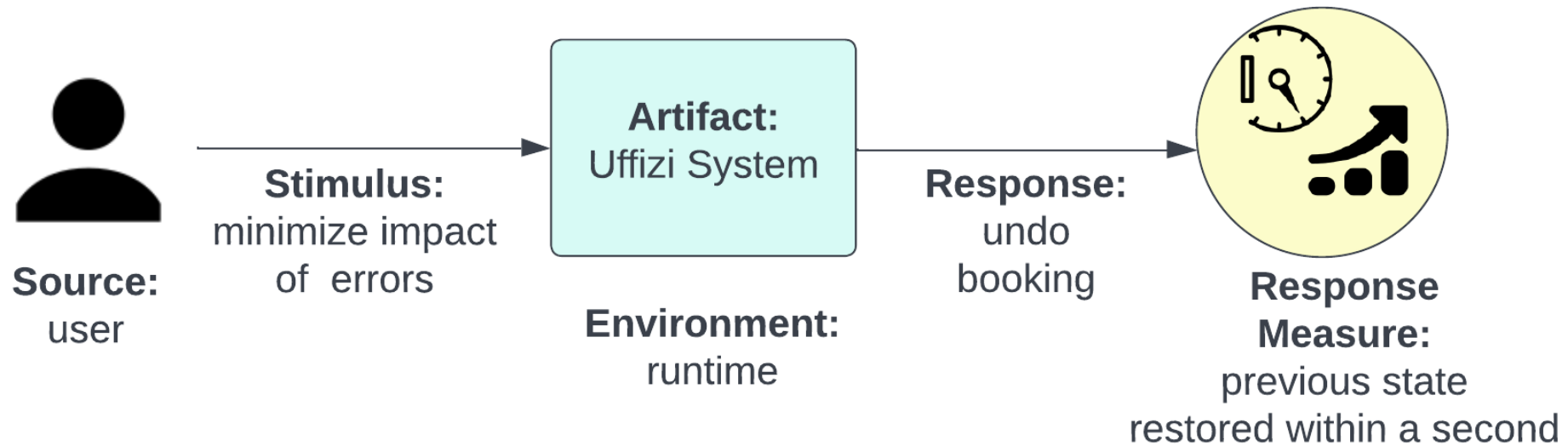
Usability has a strong correlation with modifiability

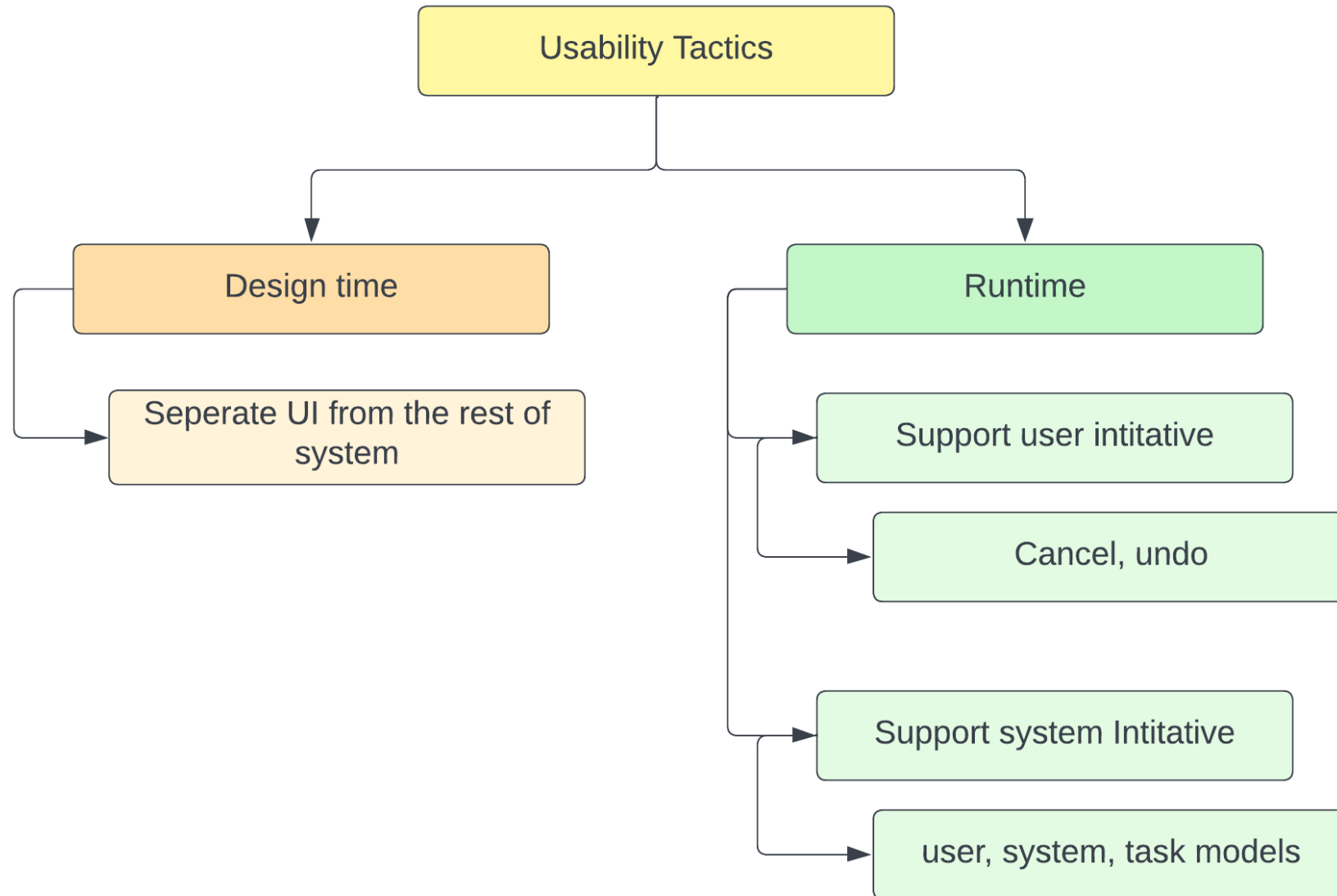
Usability Scenarios Table

Name	Classes
Source of Stimulus	End user
Stimulus	Learn the system, use efficiently, feel comfortable, minimize errors
Artifact	system
Environment	Configuration time or run time
Response	Provide users with required features, feedback to user, anticipate user needs
Response measure	Task time, number of errors, number of tasks accomplished, user satisfaction

Usability Scenario Example

Scenario: User may want to cancel during booking process





Thank You



Course website: karthikv1392.github.io/cs6401_se

Email: karthik.vaidhyanathan@iiit.ac.in

Web: <https://karthikvaidhyanathan.com>

Twitter: @karthi_ishere

