Software System Architectures (NSWI130)

Availability

Martin Nečaský

Faculty of Mathematics and Physics

Charles University in Prague

XML Web Engineering research group

availability

=

reliability + ability to recover

Availability Quality Attribute

- availability refers to a property of software that it is there and ready to carry out its task when users need it
  - ability to mask problems
  - ability to repair problems

Availability Quality Attribute

- probability that the system is operational when needed

$$\frac{\text{mean time to failure}}{\text{mean time to failure + mean time to repair}}$$

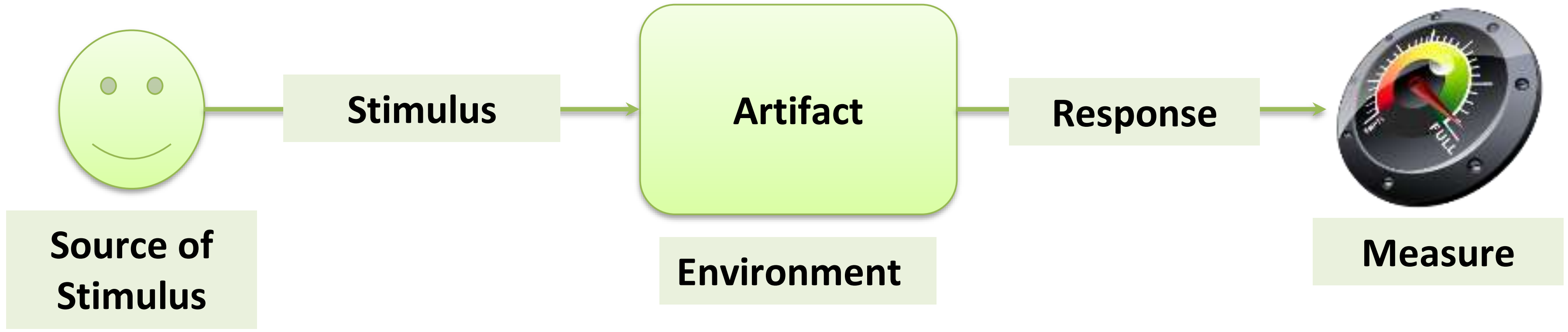
Availability Quality Attribute

- failure occurs when the system no longer delivers a service that is consistent with its specification and which is observable by users or other systems
  - failure is availability problem
- fault is a problem in the system which occurred but is not observable
  - fault is not availability problem

Techniques for availability

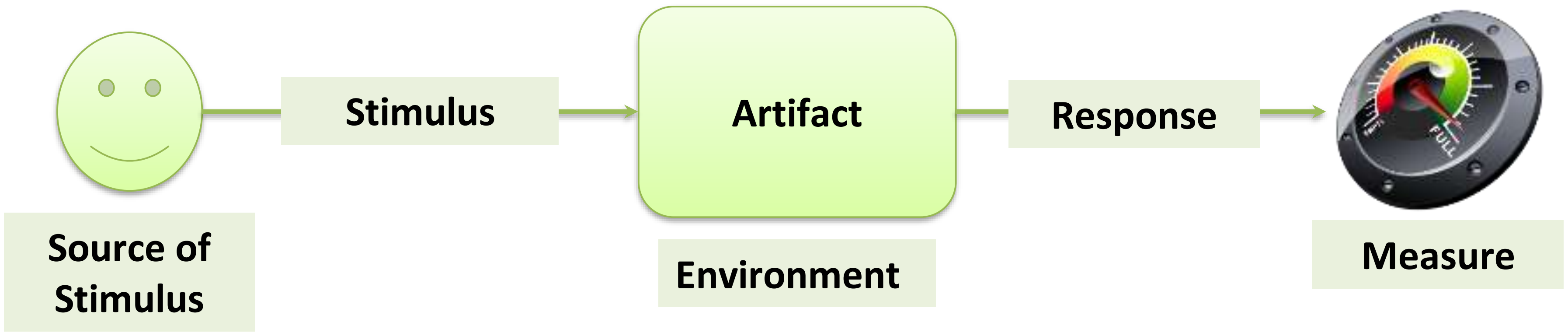
- fault recovery
  - keep faults from becoming failures
- fault repair
  - modify the system so that fault will not appear again

Availability Requirement Scenario



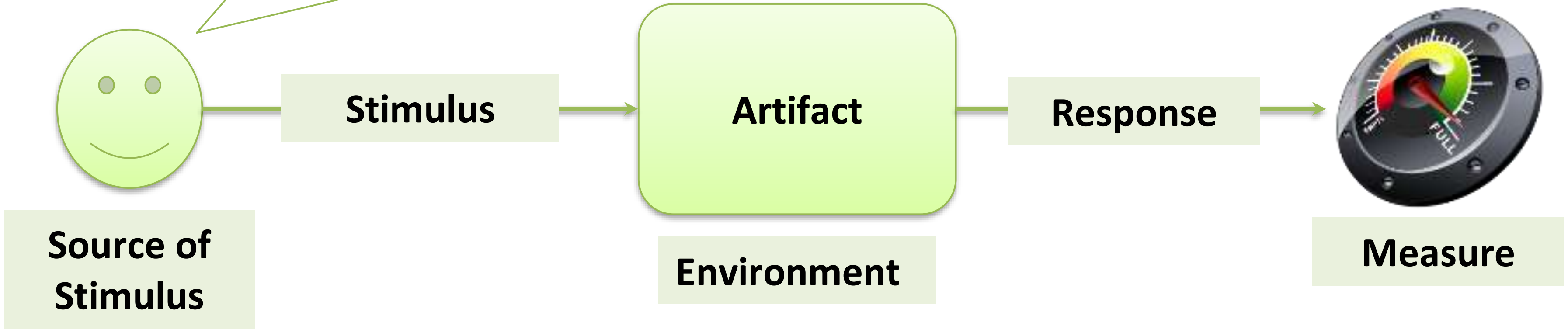
Availability Requirement Scenario

- component that is required to be available



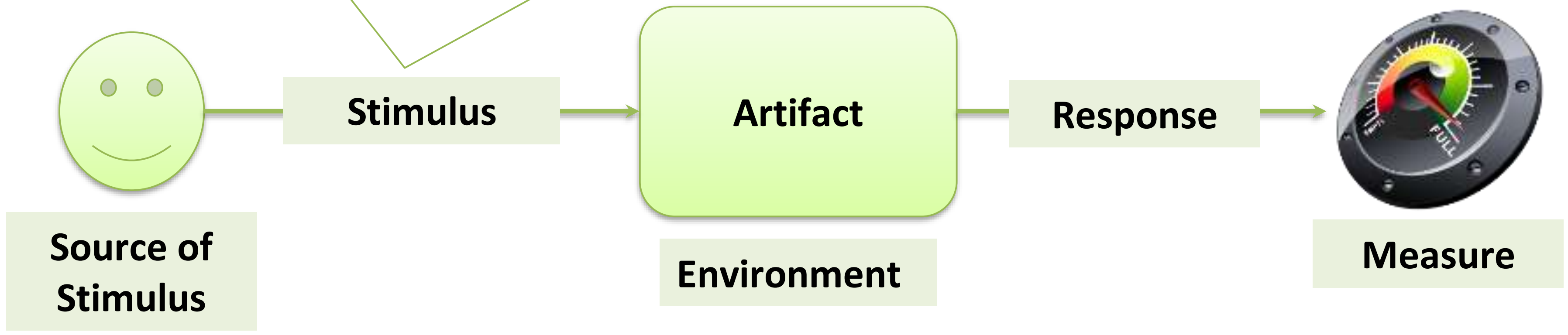
Availability Requirement Scenario

- something which observes the fault
- internal
  - component
- external
  - human user
  - another system



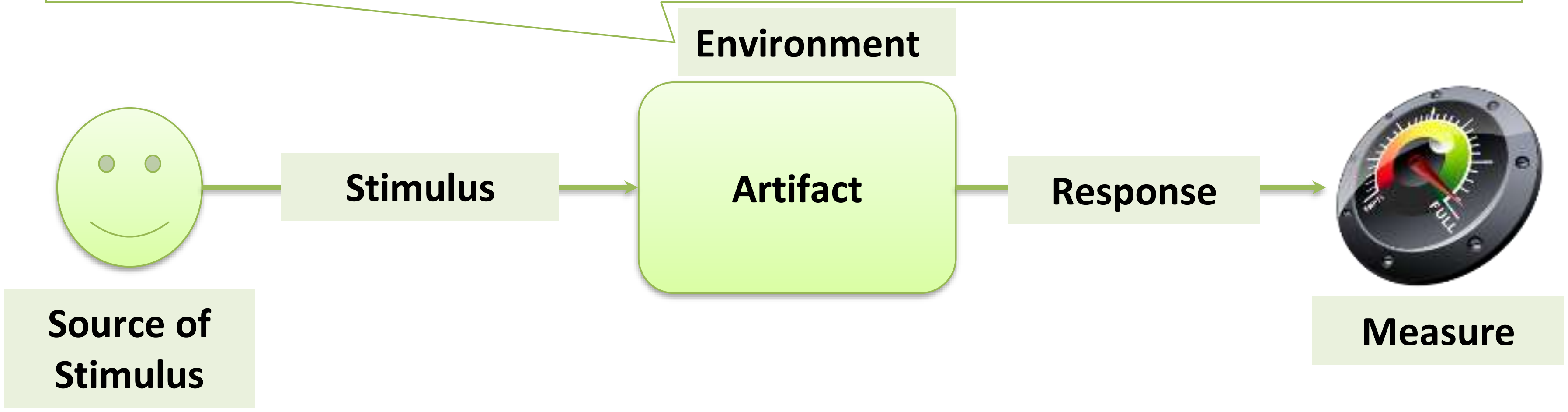
Availability Requirement Scenario

- observation of the fault
- 4 types of faults
  - omission
  - crash
  - incorrect timing
  - incorrect response



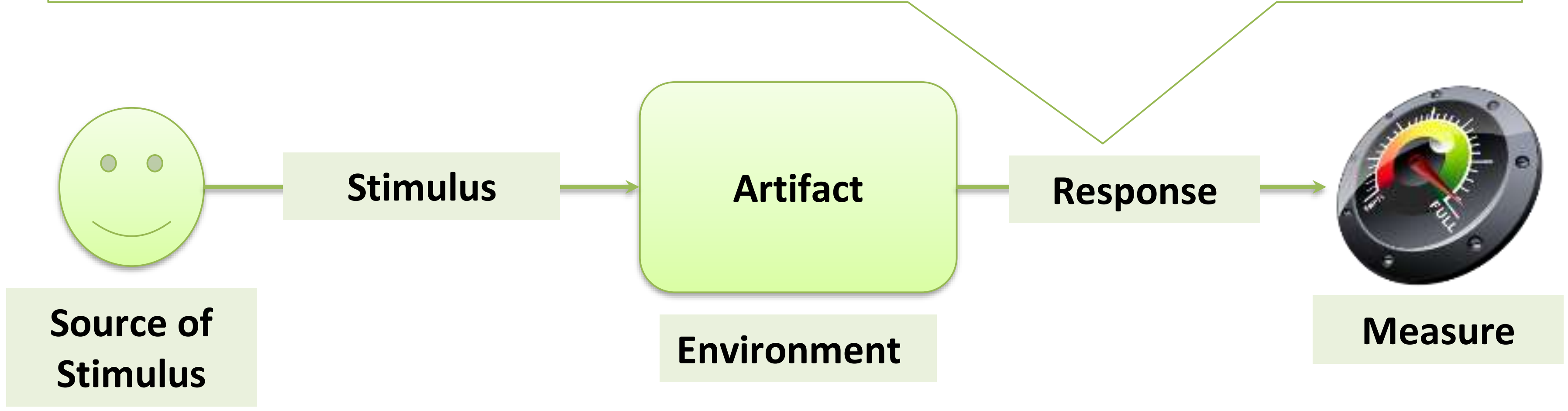
Availability Requirement Scenario

- conditions of the artifact and its surrounding environment under which the fault and its observation is considered
- startup, shutdown
- normal operation, overloaded operation
- first fault, repeated fault



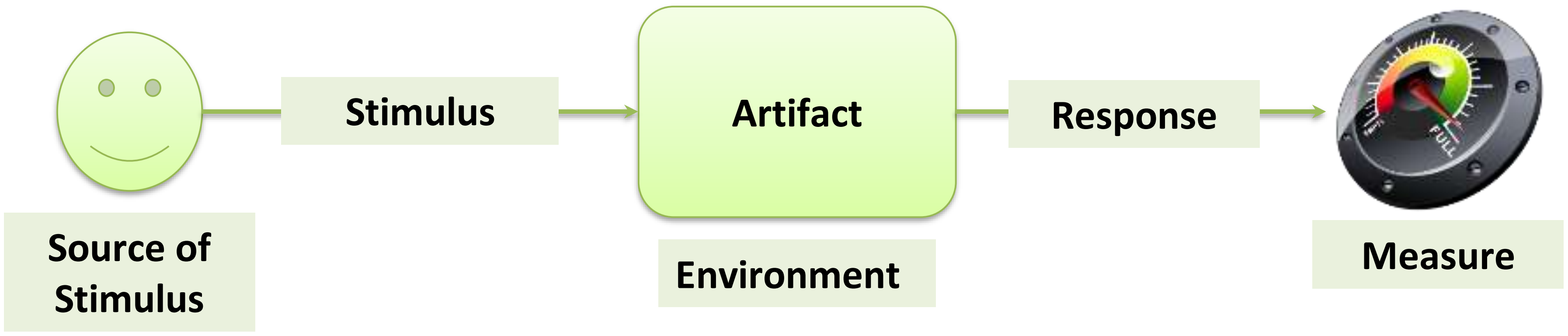
Availability Requirement Scenario

- reaction of the system to the failure
- mask fault
- try to recover from the fault
- supportive actions
  - logging, notifications, degraded mode, etc.



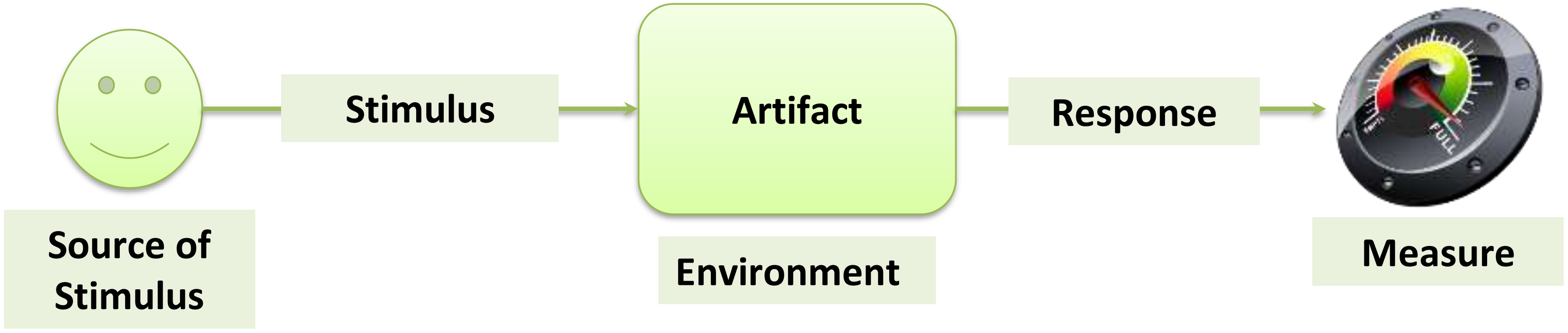
Availability Requirement Scenario

- how the fault and its repair are measured
  - how often the fault may appear
  - time required to detect the fault and repair the fault
  - time the whole system or the artifact is in a degraded mode or down



How to read scenario

- A source of stimulus wants some service from an artifact.
- The artifact, being in a given environment, fails in providing the service.
- The source of stimulus observes this fault.
- The observation stimulates the system to do something.
- The system ensures the prescribed response under the given measurable restrictions.



Availability Quality Attribute

Source: Component A

Stimulus: Unable to W (omission)

Artifact: Internal database

Environment: Normal operation

Response: mask (postpone), log

Measure: No downtime W in database in 10m

Source: Component B

Stimulus: Unable to R in transaction (omission)

Artifact: Internal database

Environment: Normal operation

Response: mask (repeat), log

Measure: 5s downtime

Source: Component C

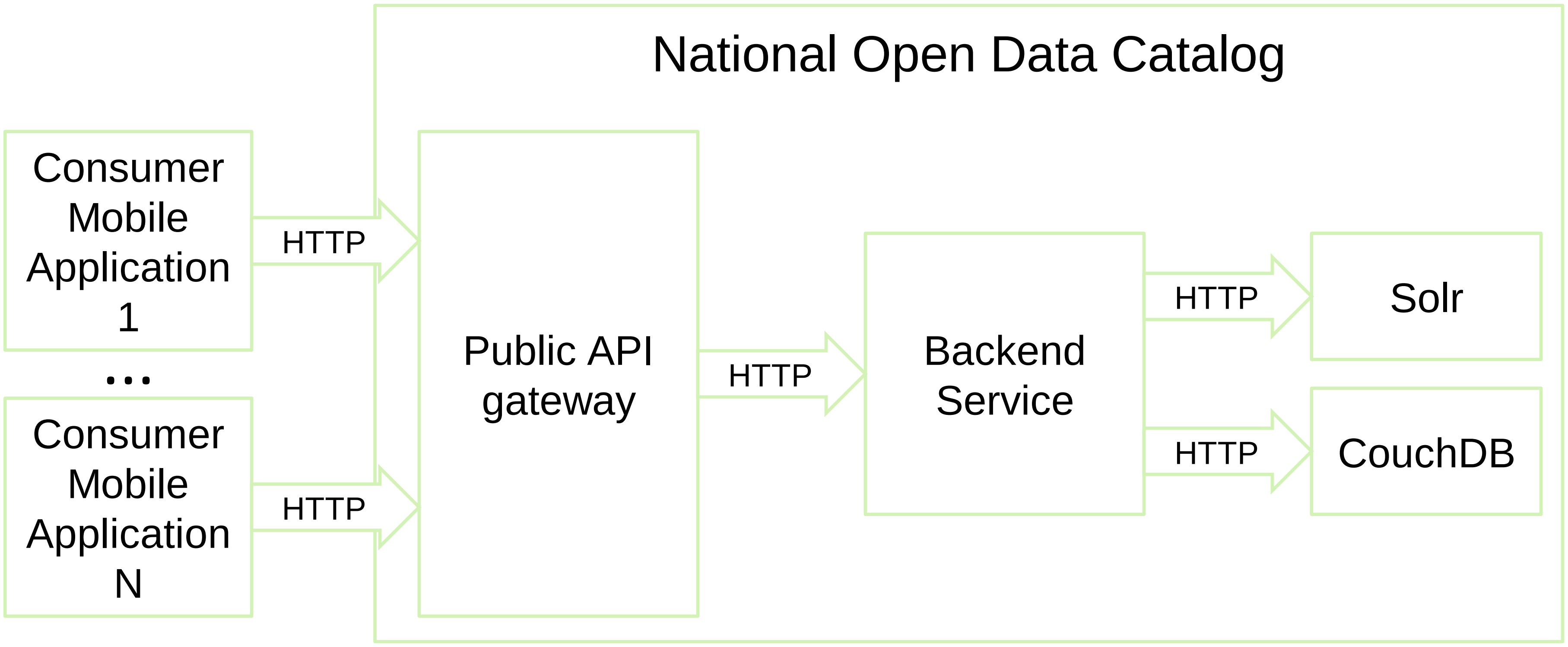
Stimulus: Unable to R for analysis (omission)

Artifact: Internal database

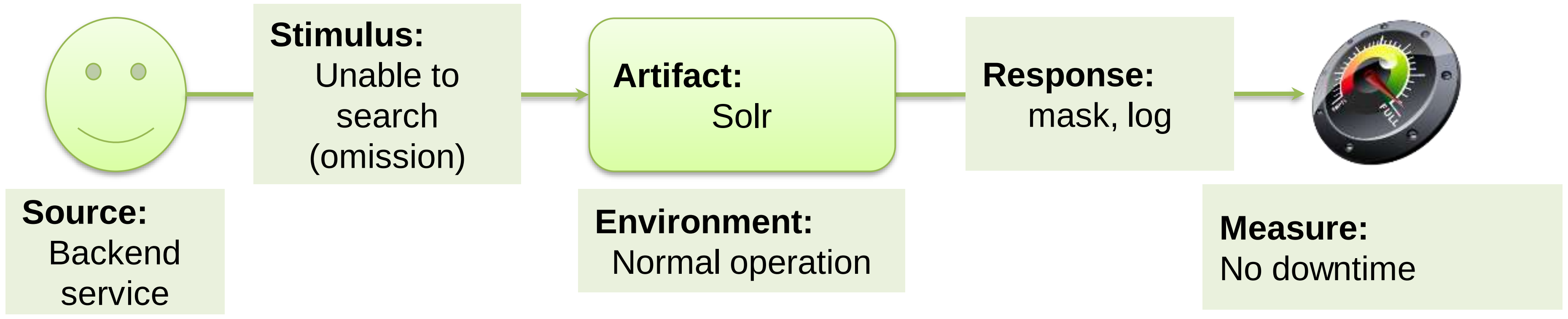
Environment: Normal operation

Response: mask (repeat), log

Measure: till 6am next day


National Open Data Catalog
Consumer Mobile Application 1
...
Consumer Mobile Application N
HTTP
HTTP
Public API gateway
HTTP
Backend Service
HTTP
HTTP
Solr
CouchDB

Availability Quality Attribute



Source: Backend service

Stimulus: Unable to search (omission)

Artifact: Solr

Environment: Normal operation

Response: mask, log

Measure: No downtime

Availability Quality Attribute

Source: Backend service

Stimulus: Unable to search (crash after 2 omissions)

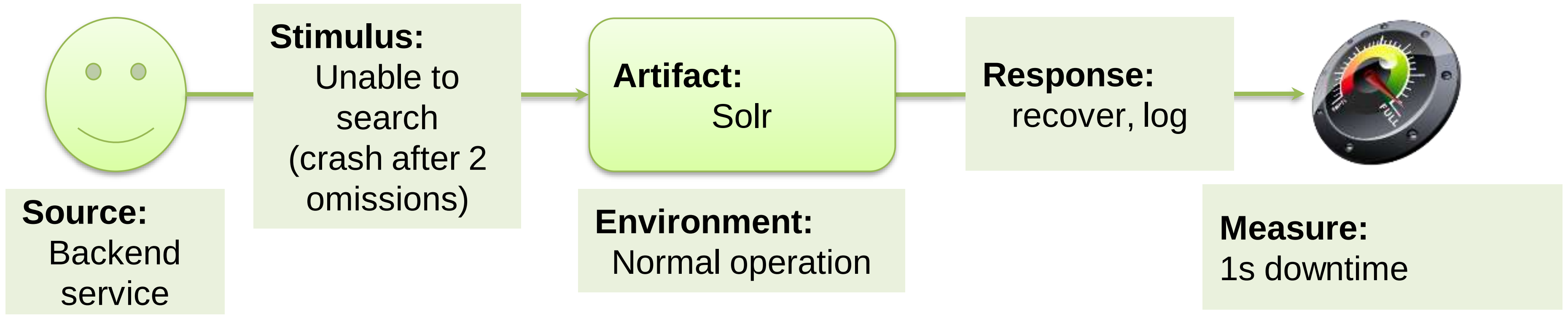
Artifact: Solr

Environment: Normal operation

Response: recover, log

Measure: 10s downtime

Availability Quality Attribute



Source: Backend service

Stimulus: Unable to search (crash after 2 omissions)

Artifact: Solr

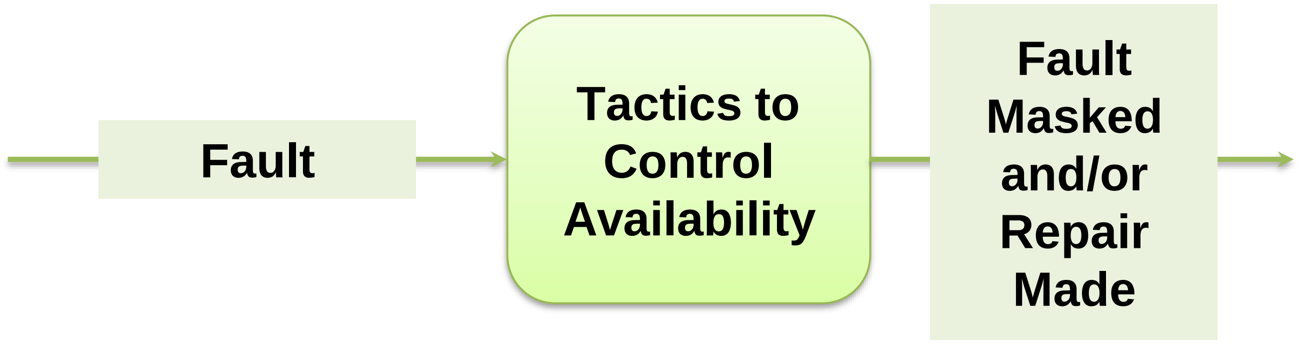
Environment: Normal operation

Response: recover, log

Measure: 1s downtime

Availability Tactics and Their Goals

- mask fault (to not become failure)
- repair fault


Fault
Tactics to Control Availability
Fault Masked and/or Repair Made

Availability Tactics

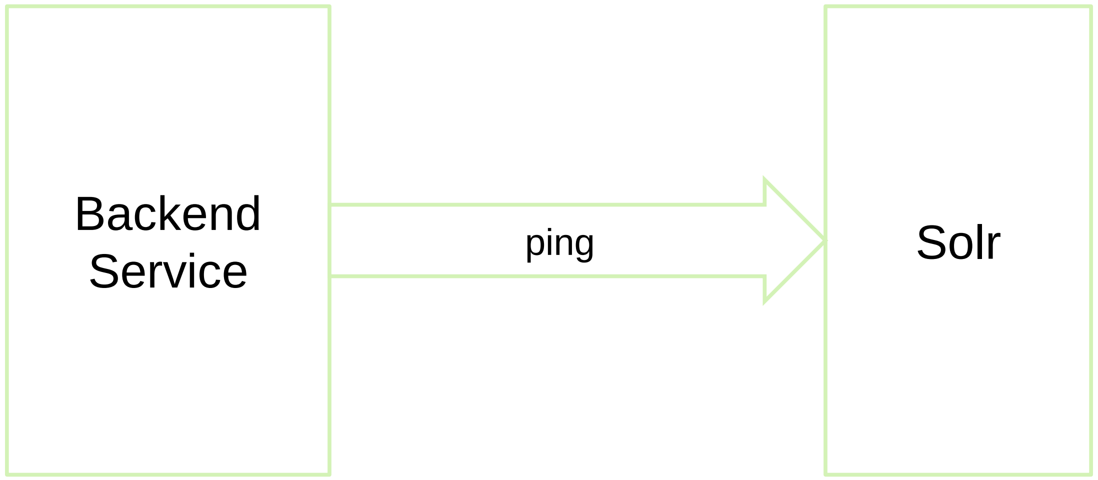
- detect faults
- recover from faults
- prevent faults

Fault Detection

- ping/echo
  - component pings another component and awaits echo in some defined amount of time
  - hierarchical ping/echo to reduce communication bandwidth
  - used to determine reachability and the round-trip delay through the associated network path
  - implementation depends on communication protocol (e.g. HTTP HEAD or ICMP)

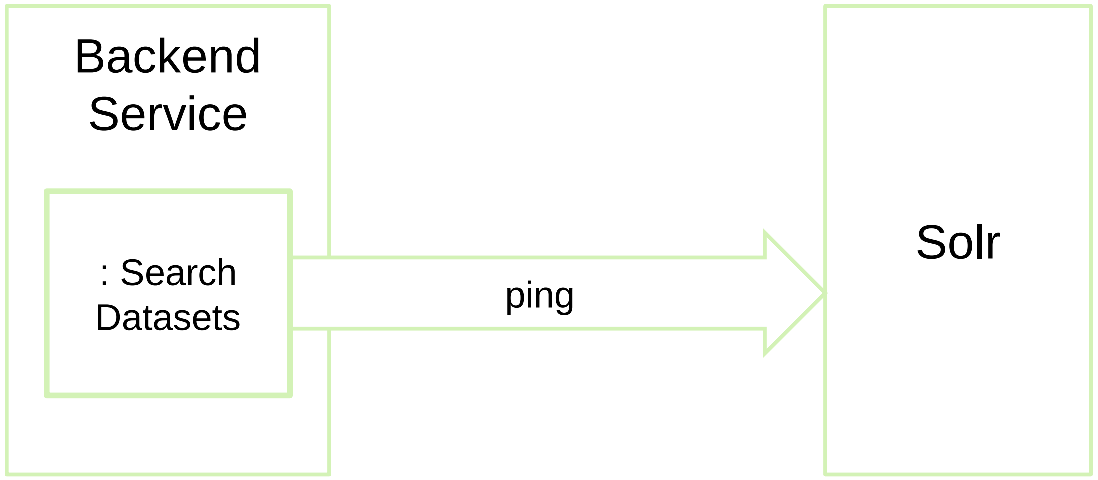
Fault Detection

- ping/echo



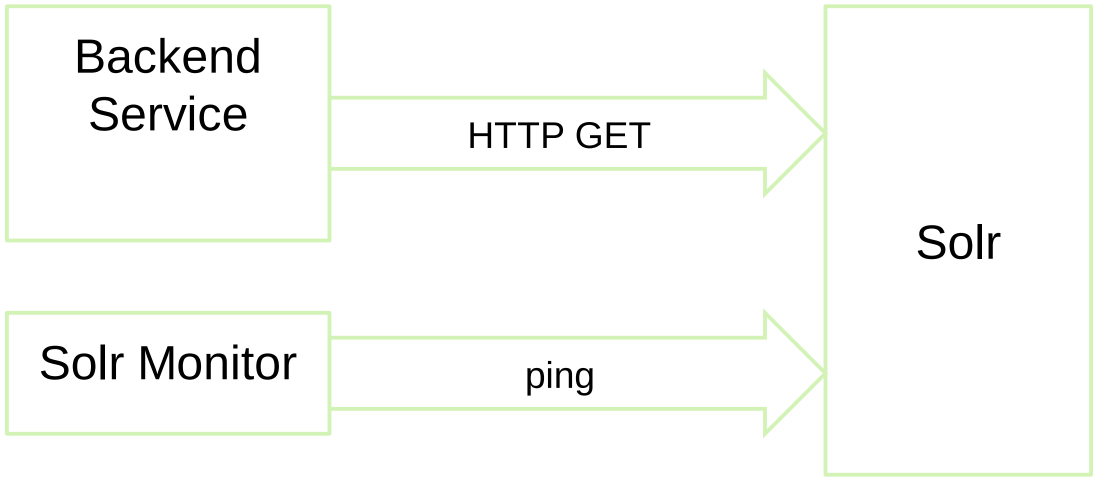
Fault Detection

- ping/echo



Fault Detection

- ping/echo



Fault Detection

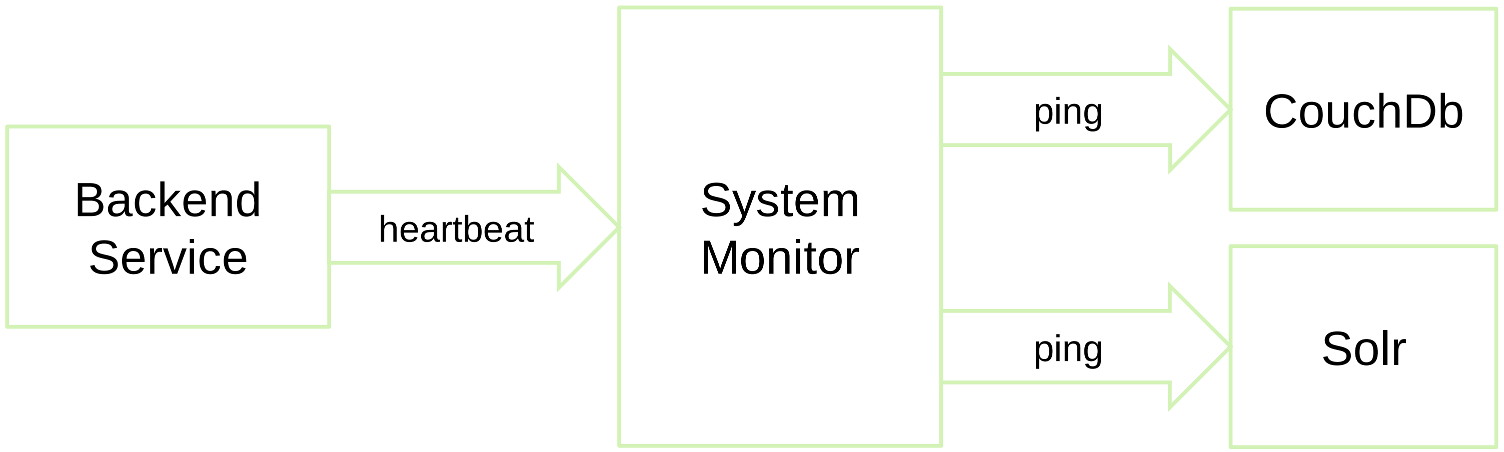
- heartbeat
  - one component emits heartbeat messages periodically and another component listens to them
  - heartbeat can also carry data

Fault Detection

- monitor
  - component that is used to monitor the state of health of various other parts of the system
  - introduced to architecture when the monitoring logic needs to be separated from the other logics of the system

Fault Detection

- monitor



Fault Detection

- time stamps
  - for monitoring incorrect order or timing of responses of a component
  - important in distributed environments
  - timestamps based on local clock of the system or on some logical clock approach, e.g. Lamport timestamps
- timeout

Fault Detection

- voting (Triple Modular Redundancy)
  - replication
  - functional redundancy
  - analytic redundancy

Fault Recovery

- preparation and repair tactics
- reintroduction tactics

Preparation and repair tactics

- active redundancy (hot spare)
  - redundant components perform the same tasks on the same inputs
- passive redundancy (warm spare)
  - one component performs the tasks and informs others periodically about state updates
- cold spare
  - redundant spares remain out of service until a fault on the main component occurs

Preparation and repair tactics

- rollback
  - system is reverted to a previous known good state (checkpoints)
- saga
  - distributed system executes sagas consisting of local transactions
  - when a fault occurs performed local transactions are reverted using compensation transactions

Preparation and repair tactics

- retry
  - an operation fault is transient and retrying the operation may lead to success
- ignore
  - messages sent from a particular component with faulty or spurious behavior
- degradation
  - maintains only the most critical system functions
- reconfiguration
  - reassigning responsibilities to resources left functioning

Reintroduction tactics

- supporting tactics to recover failed component (reintroducing a failed component)

Reintroduction tactics

- shadow
  - operating a previously failed component in a “shadow mode” for a predefined duration of time
- state resynchronization
  - supporting tactic to check synchronization between components
  - based on data sampling or checksums
- escalating restart
  - system or component restarted or its memory freed

Prevent faults

- removal from service
  - a preventive restart or reconfiguration of a component in order to scrub latent faults, e.g. memory leaks
- transactions
  - operations in the system are executed in transactions which ensure ACID properties
  - 2PC protocol
- predictive model
  - evaluates the state of health of a component by monitoring its outputs and predicting possible faults

Prevent faults

- chaos engineering
- https://principlesofchaos.org/
- https://netflix.github.io/chaosmonkey/