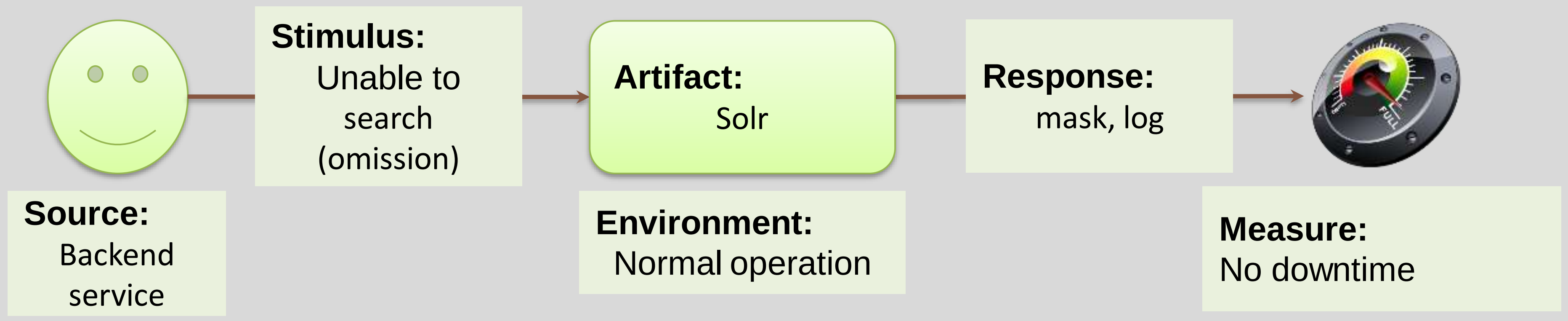
Software System Architectures (NSWI130) Quality attributes in C4 model

- Martin Nečaský, Ph.D.
- Department of Software Engineering
- Faculty of Mathematics and Physics
- Charles University in Prague

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 - scenario



Source: Backend service

Stimulus: Unable to search (omission)

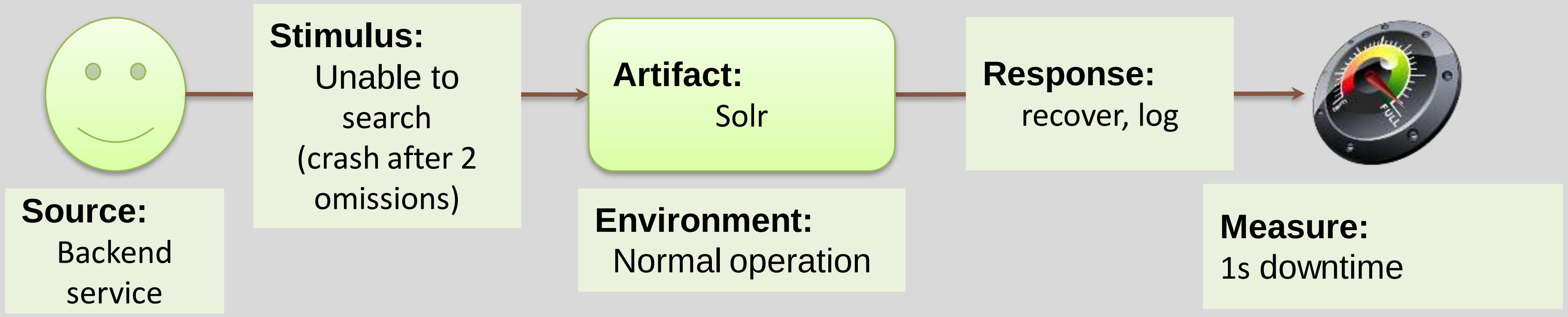
Artifact: Solr

Environment: Normal operation

Response: mask, log

Measure: No downtime

Availability - scenario



Source: Backend service

Stimulus: Unable to search (crash after 2 omissions)

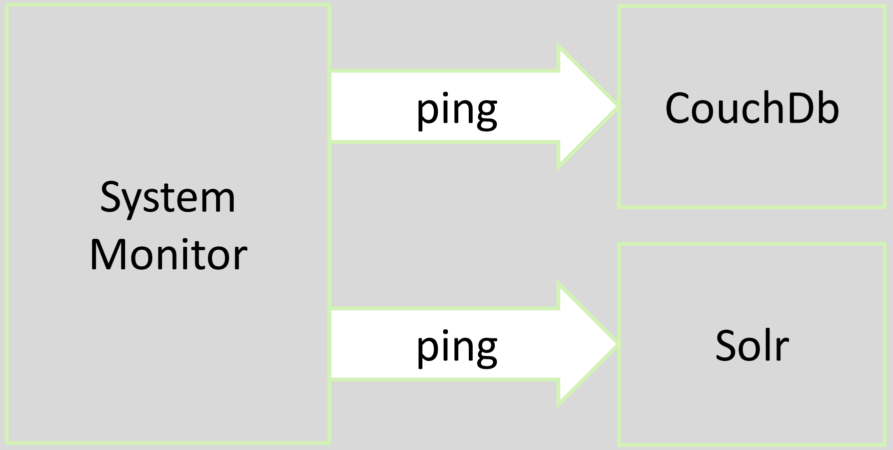
Artifact: Solr

Environment: Normal operation

Response: recover, log

Measure: 1s downtime

Availability – solution of monitoring



Availability – solution of monitoring (C4 model)

- System monitor can be an infrastructural node in the live deployment

```
model {
  deploymentEnvironment "Live" {
    deploymentNode "NODC UPAAS" {
      deploymentNode "NODC-upaas-monitor" {
        monitor = infrastructureNode "System monitor" "" "Zabbix"
      }
    }
  }
  monitor -> recordIndexInstance
  monitor -> recordStorageInstance
}
```

sampleworkspace10.dsl

Availability – solution of monitoring (C4 model)

- System monitor can be a container which reuses an existing solution, and which is then deployed.

```
model {
  systemMonitor = container "System Monitor" "" "Zabbix configuration"
  …
  deploymentNode "NODC-upaas-monitor" {
    deploymentNode "Zabbix" {
      monitor = containerInstance systemMonitor
    }
  }
}
```

sampleworkspace11.dsl

Availability – solution of monitoring (C4 model)

- System monitor can be a container with own implementation which is then deployed.

```
model {
  systemMonitor = container "System Monitor" "" ".Net“
  systemMonitorDb = container "System Monitor Database" "" "Relational database"
  …
  deploymentNode "NODC-upaas-monitor" {
    deploymentNode ".Net" {
      monitor = containerInstance systemMonitor
      monitorDb = containerInstance systemMonitorDb
    }
  }
}
```

sampleworkspace12.dsl

Availability – solution of recovery (C4 model)

- Database replication – given database system usually provides some replication features
- e.g. CouchDB - https://docs.couchdb.org/en/stable/replication/intro.html

Availability – solution of recovery (C4 model)

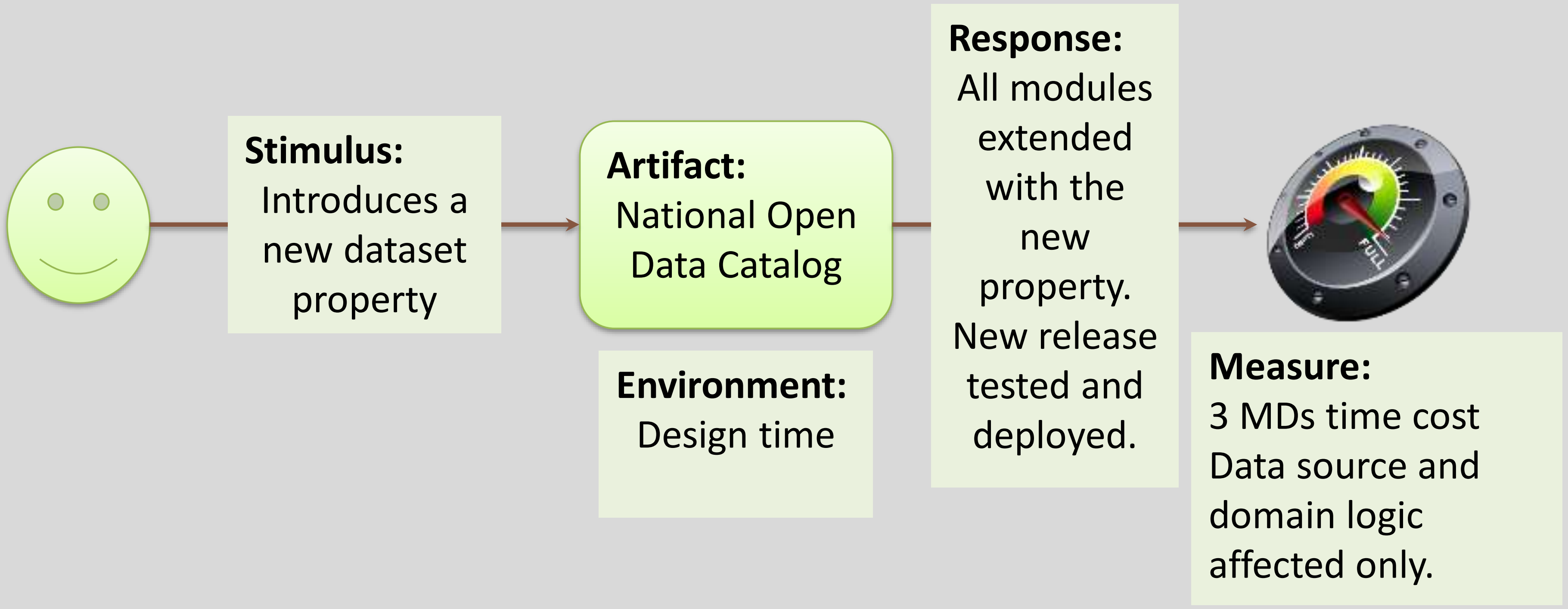
```
model {
  deploymentEnvironment "Live" {
    deploymentNode "NODC UPAAS" {
      deploymentNode "Apache CouchDB Cluster" {
        deploymentNode "NODC-upaas-storage" "" "Ubuntu 18.04 LTS" "" 2  {
          deploymentNode "Apache CouchDB" "" "Apache CouchDB 3.*"   {
            recordStorageInstance = containerInstance recordStorage
          }
        }
      }
    }
  }
}
```

sampleworkspace13.dsl

Modifiability

- Change is the only constant in the universe.
- It is ubiquitous in the software lifecycle.
- Our interest in modifiability centers on the cost and risk of making changes.

Modifiability - scenario



Stimulus:
Introduces a new dataset property

Artifact:
National Open Data Catalog

Environment:
Design time

Response:
All modules extended with the new property.
New release tested and deployed.

Measure:
3 MDs time cost
Data source and domain logic affected only.

Modifiability – solution of cohesion (C4 model)

- Component view of the NODC server container from our running example aims at increasing **cohesion**. How?
  - Separation of infrastructural responsibilities (APIs and gateways), business responsibilities (controllers) and model responsibilities.

Modifiability – solution of cohesion (C4 model)

- Component view of the NODC server container from our running example aims at decreasing **coupling**. How?
  - Layering of infrastructural responsibilities (APIs and gateways), business responsibilities (controllers) and model responsibilities.
  - Infrastructural layer cannot access domain model layer
  - Domain layer cannot access anything

Modifiability – intersystem coupling (C4 model)

- Consider a new situation – NODC is accessed by a government reporting system which reports on the current state of individual government organizations
- How to reduce coupling between NODC and the reporting system?
- NODC has a query interface to all metadata in the catalog.
  - Advantages? The reporting system can read anything.
  - Disadvantages? The reporting system creates couplings.

compare:
sampleworkspace14.dsl
sampleworkspace15.dsl

Performance

- performance is measure of how long it takes system to respond to events

Performance - scenario

Source: Gov Users

Stimulus: Search datasets - stochastically 60 txs per minute

Artifact: National open data catalog

Environment: Under normal operations

Response: Transactions are processed

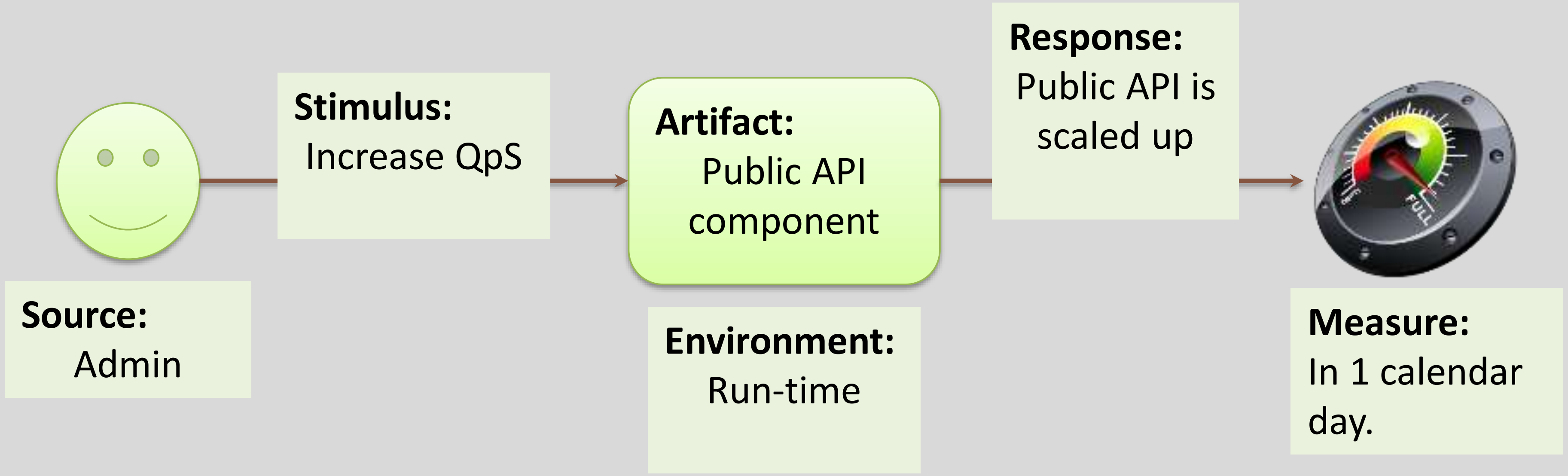
Measure: With average latency of 1 secs

sampleworkspace16.dsl (live deployment)

- prioritize events
    - events from normal and governmental users are on different queues
- increased resources tactic
    - more instances of NODC Server with a load balancer to serve governmental users

- Not related to the scenario:
    - Increased resources tactic
        - Metadata harvestor has 4 instances running in parallel
    - Better search algorithm, caching
        - Metadata index
    - Caching
        - Metadata storage

Scalability - scenario



Source: Admin

Stimulus: Increase QpS

Artifact: Public API component

Environment: Run-time

Response: Public API is scaled up

Measure: In 1 calendar day.

Scalability – solution of scaling (C4 model)

- redundancy + infrastructural nodes (i.e. load balancers)
- see previous sample workspaces for load balancing requests from end-user’s devices (container view)
- modifiability is related to but not the same as scalability (one of the next lectures)

Security

- measure of ability to protect data and information from unauthorized access

Security - scenario

Source:
Known external catalog

Stimulus:
Request to catalog high number of odd datasets to make harvesting service unavailable

Artifact:
Harvesting service

Environment:
open to internet

Response:
Harvesting other external catalogs not affected

Measure:
DoS detected during the attacked harvesting task,
external catalog recorded in the audit trail,
national open data catalog and local catalog administrators notified

sampleworkspace16.dsl

- #NODC_Container_View > Metadata Harvestor
  - logs to the Registry of local open data catalog
  - notifies NODC and LODC admins
- #Harvesting_Container_Security_Dynamic_View
  - suggests when the logging and notifications happen

Interoperability

- degree to which two or more systems can usefully exchange meaningful information via their interfaces in a given context.

Interoperability - scenario

Source:
Local Open Data Catalog

Stimulus:
Dataset records exchange

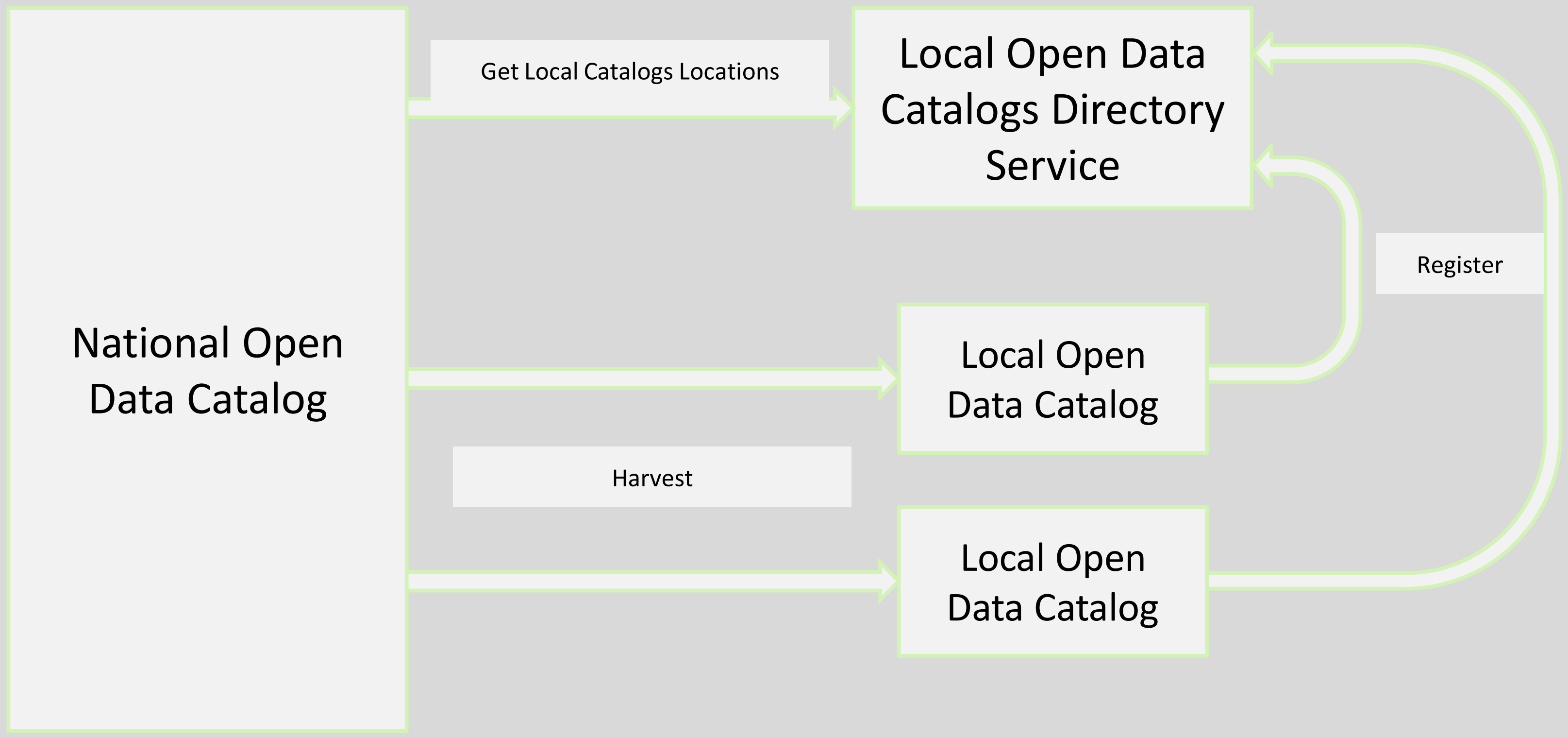
Artifact:
National Open Data Catalog

Environment:
LODC unknown prior to NODC run- time

Response:
NODC combines local dataset records with other LODCs and provides search features on top of them to users

Measure:
LODC information included with a probability of 100 %

Interoperability – directory service tactic


Local Open Data Catalogs Directory Service
Get Local Catalogs Locations
Register
National Open Data Catalog
Local Open Data Catalog
Harvest
Local Open Data Catalog

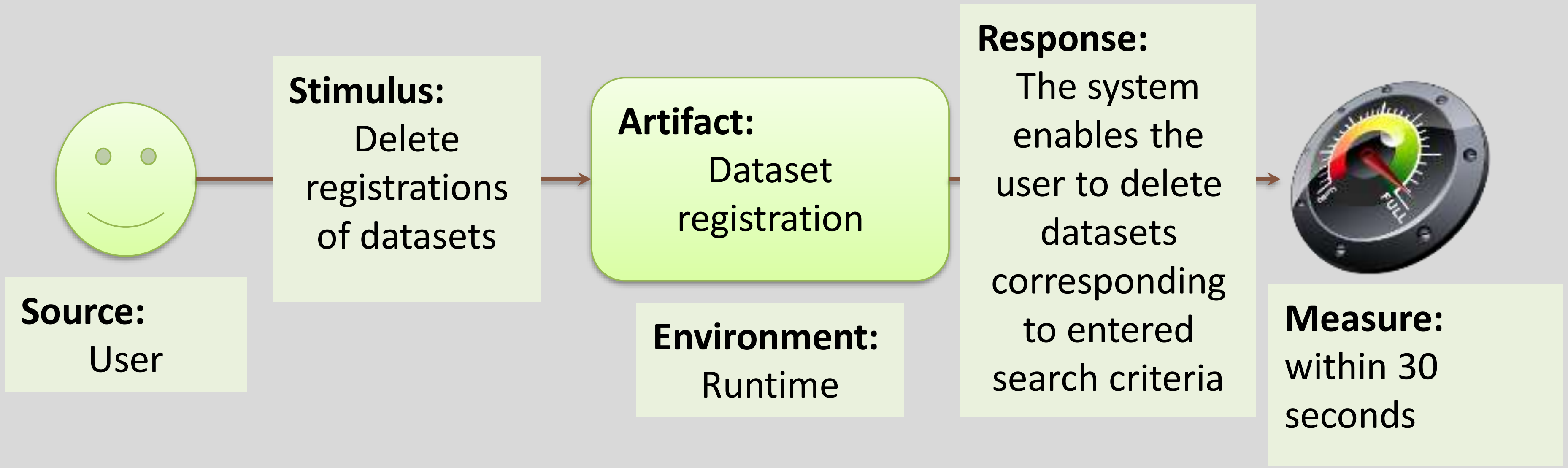
sampleworkspace16.dsl

- #NODC_Container_View > Registry of local open data catalogs is a directory service
- Specifies the DCAT-AP standard for exchanging metadata records between the Metadata Harvestor and LODCs to achieve syntactic and semantic interoperability

Usability

- concerned with how easy it is for the user to accomplish a desired task and the kind of user support the system provides

Usability - scenario



Source: User

Stimulus: Delete registrations of datasets

Artifact: Dataset registration

Environment: Runtime

Response: The system enables the user to delete datasets corresponding to entered search criteria

Measure: within 30 seconds

sampleworkspace16.dsl

- Think about the quality requirement w. r. t. our NODC architecture.
- Is it supported somehow? Is it real to implement it?

sampleworkspace16.dsl

- Possible answer : The current architecture does not support deleting datasets at all. This requirement is not about changing UI. The architecture itself disables such kind of requirement. It cannot be implemented without changing the architecture. It is possible to do it, but we would have to rebuild the basic building blocks of the system. It would cost a lot of resources and time.
- This situation can happen – that there are architectural reasons for not implementing some requirements which seem functional. But they are architectural since the architecture disables them.