Eiffel

An event protocol
for CI/CD pipelines

CD Foundation SIG Interoperability Feb 6th 2020

Emil Bäckmark

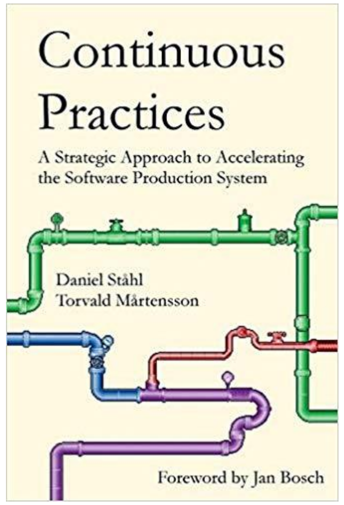
ERICSSON

Background

- How can we communicate about new artifacts becoming available?
- How can we reach out to others who are using a different platform or tools for their pipeline?
- How can we know that someone starts testing our product? And whether it is successful or not?
- How can we visualize our connected pipelines, independently of the tools or technology used for driving it?
- How can we trace our delivered product all the way back to the source code changes, across different pipeline tools?

Background

- Ericsson's journey with Eiffel was described in the CD Summit NA 2019.
- Eiffel presented on FOSDEM 2020 – Ericsson together with Axis Communications.
- The use cases for Eiffel and also Eiffel specifically has been discussed in academia (see refs on last slide) and in at least east one printed book.



Main Principle

“What to communicate is volitional; how to communicate it is not”

What you say is up to you,
but how you say it is not
(if you want to be understood)

Dialects are often understandable,
as long they don’t diverge to much

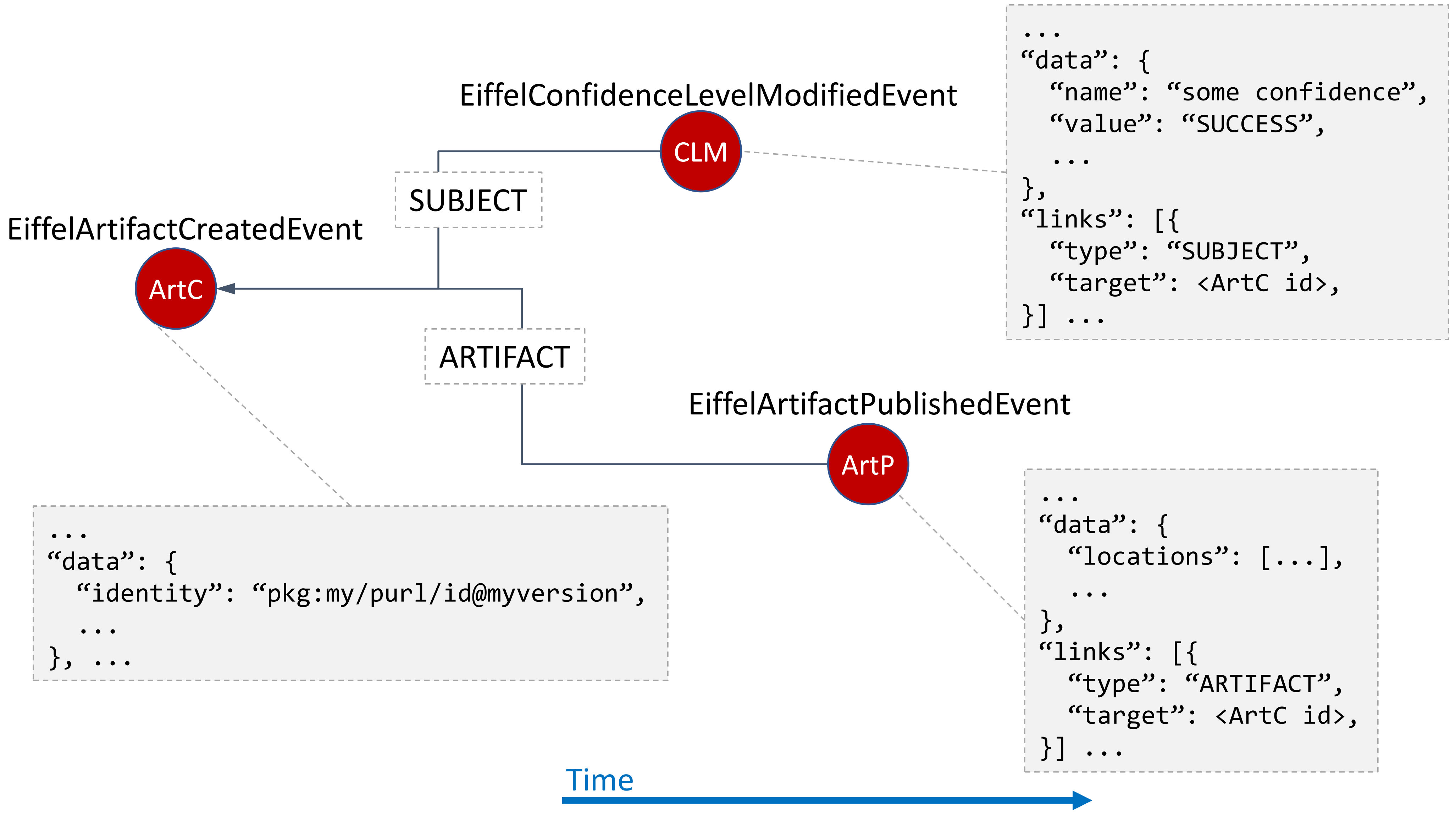
Event Syntax

```
{
  “meta”: {
    “id”: <UUID event id>,
    “type”: <event type>,
    “version”: <semver version>,
    “time”: <timestamp>,
    ...
},
  “data”: {
    ...
  },
  “links”: [
    ...
  ]
}
```

Event Families

- Artifact related events
- Test related events
- Source Code related events
- Activity related events
- and more

Artifact Related Events

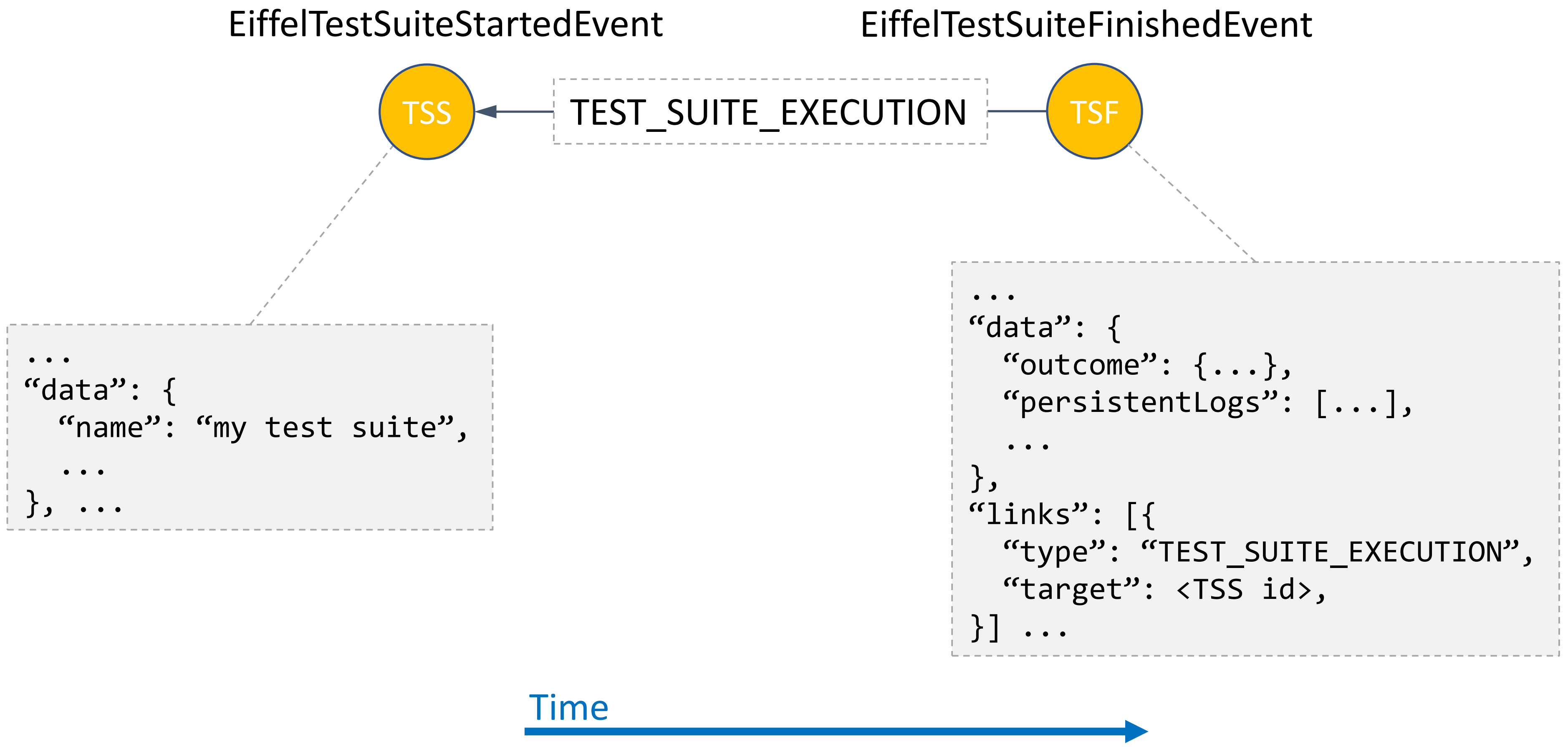

EiffelConfidenceLevelModifiedEvent
CLM
SUBJECT
EiffelArtifactCreatedEvent
ArtC
ARTIFACT
EiffelArtifactPublishedEvent
ArtP
Time

```
...
"data": {
  "name": "some confidence",
  "value": "SUCCESS",
  ...
},
"links": [{
  "type": "SUBJECT",
  "target": <ArtC id>,
}] ...
```

```
...
"data": {
  "identity": "pkg:my/purl/id@myversion",
  ...
}, ...
```

```
...
"data": {
  "locations": [...],
  ...
},
"links": [{
  "type": "ARTIFACT",
  "target": <ArtC id>,
}] ...
```

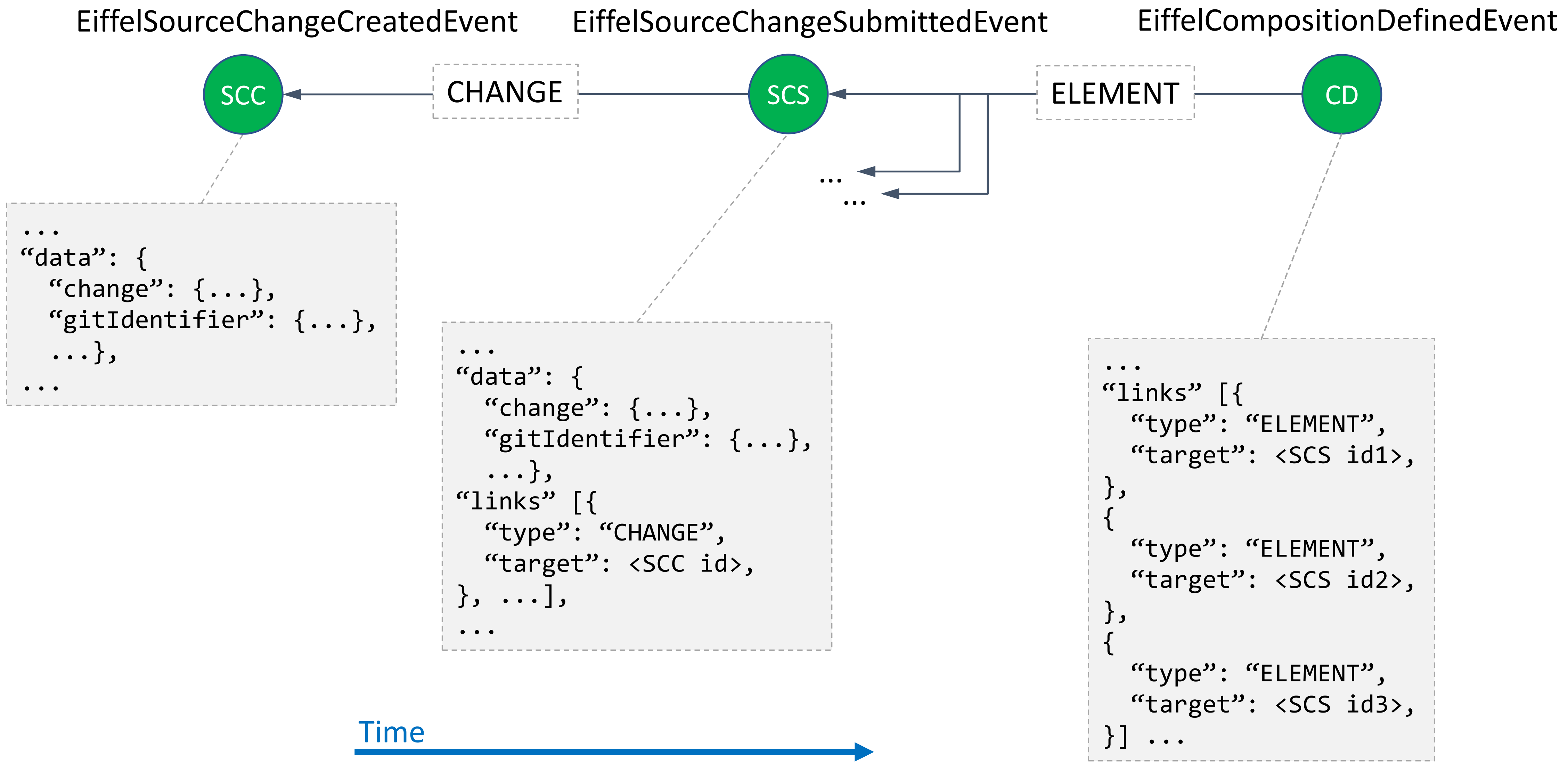
Test Related Events



```
...
"data": {
  "name": "my test suite",
  ...
}, ...
```

```
...
"data": {
  "outcome": {...},
  "persistentLogs": [...],
  ...
},
"links": [{
  "type": "TEST_SUITE_EXECUTION",
  "target": <TSS id>,
}] ...
```

Source Code Related Events

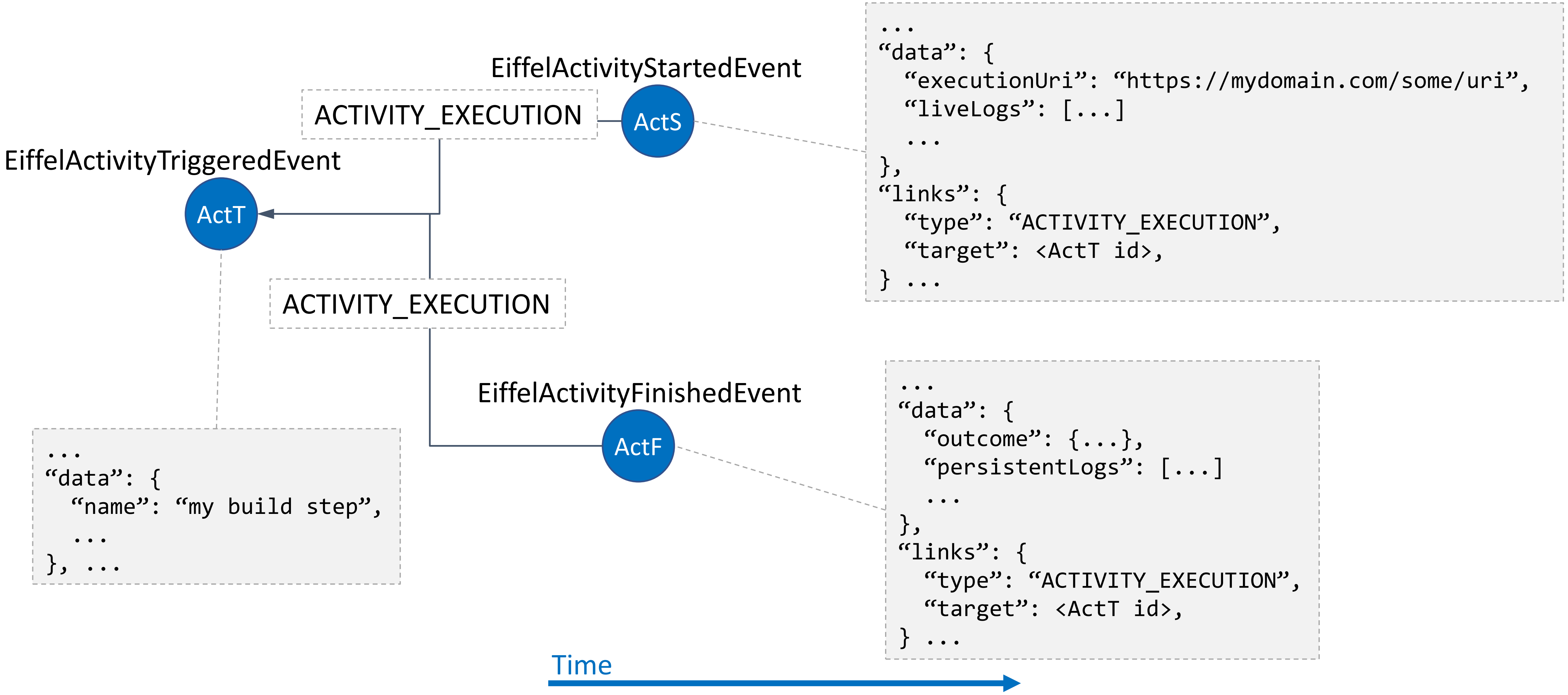

EiffelSourceChangeCreatedEvent
EiffelSourceChangeSubmittedEvent
EiffelCompositionDefinedEvent
SCC
CHANGE
SCS
ELEMENT
CD
...
...
Time

```
...
"data": {
  "change": {...},
  "gitIdentifier": {...},
  ...},
...
```

```
...
"data": {
  "change": {...},
  "gitIdentifier": {...},
  ...},
"links" [{
  "type": "CHANGE",
  "target": <SCC id>,
}, ...],
...
```

```
...
"links" [{
  "type": "ELEMENT",
  "target": <SCS id1>,
},
{
  "type": "ELEMENT",
  "target": <SCS id2>,
},
{
  "type": "ELEMENT",
  "target": <SCS id3>,
}] ...
```

Activity Related Events

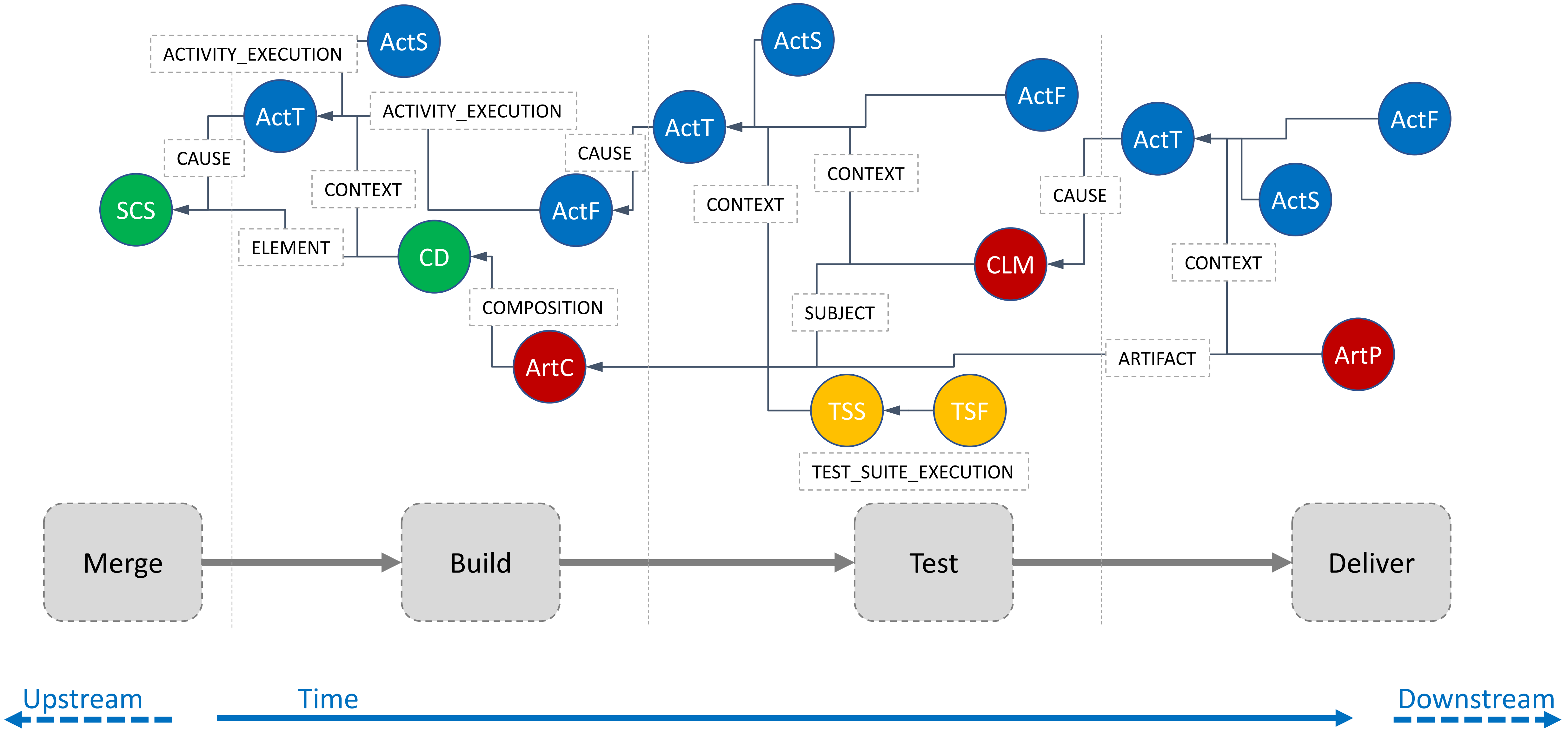

EiffelActivityStartedEvent
ACTIVITY_EXECUTION
ActS
EiffelActivityTriggeredEvent
ActT
ACTIVITY_EXECUTION
EiffelActivityFinishedEvent
ActF
Time

```
...
"data": {
  "executionUri": "https://mydomain.com/some/uri",
  "liveLogs": [...]
  ...
},
"links": {
  "type": "ACTIVITY_EXECUTION",
  "target": <ActT id>,
} ...
```

```
...
"data": {
  "name": "my build step",
  ...
}, ...
```

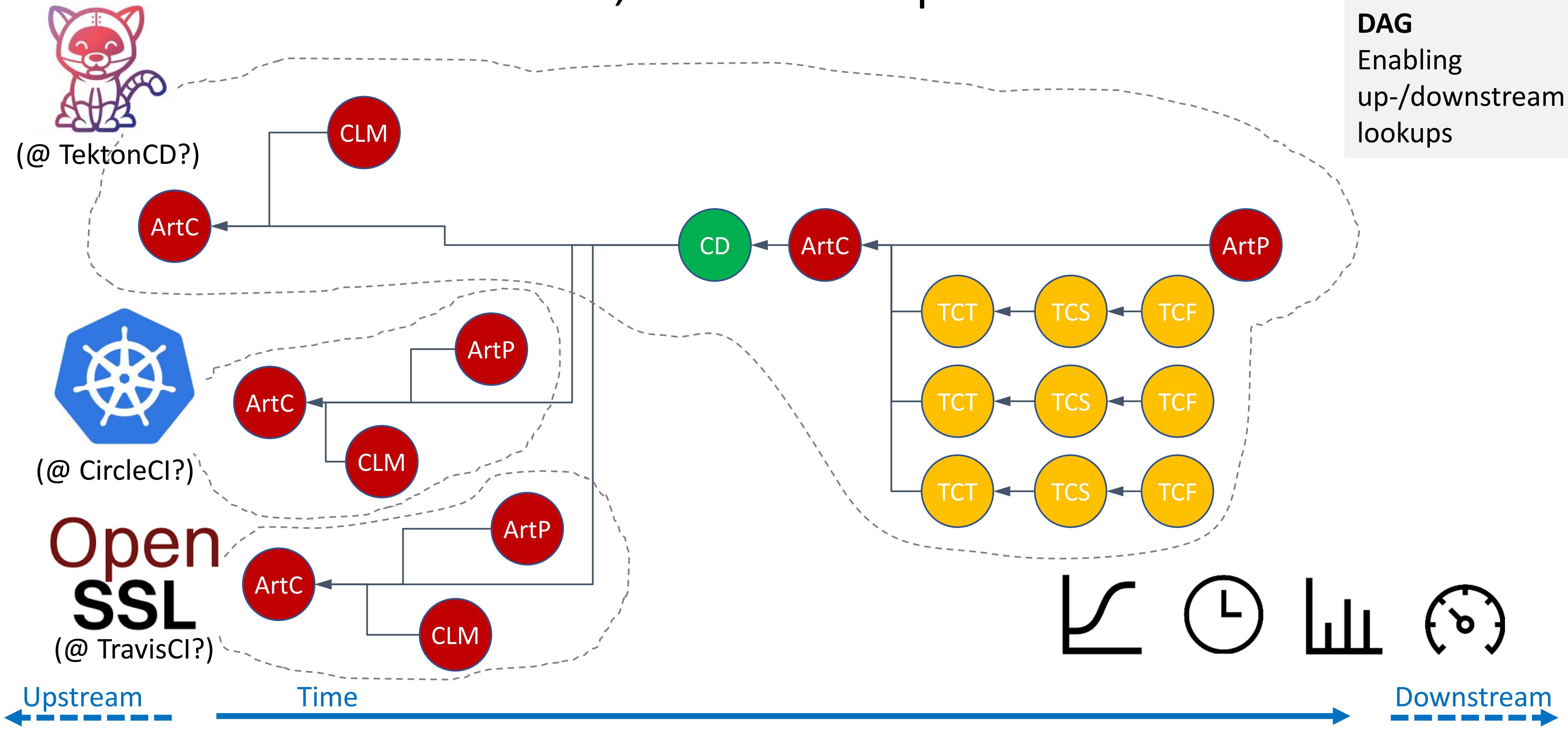
```
...
"data": {
  "outcome": {...},
  "persistentLogs": [...]
  ...
},
"links": {
  "type": "ACTIVITY_EXECUTION",
  "target": <ActT id>,
} ...
```

Combined Events



Combined Events, 2nd example

DAG
Enabling up-/downstream lookups

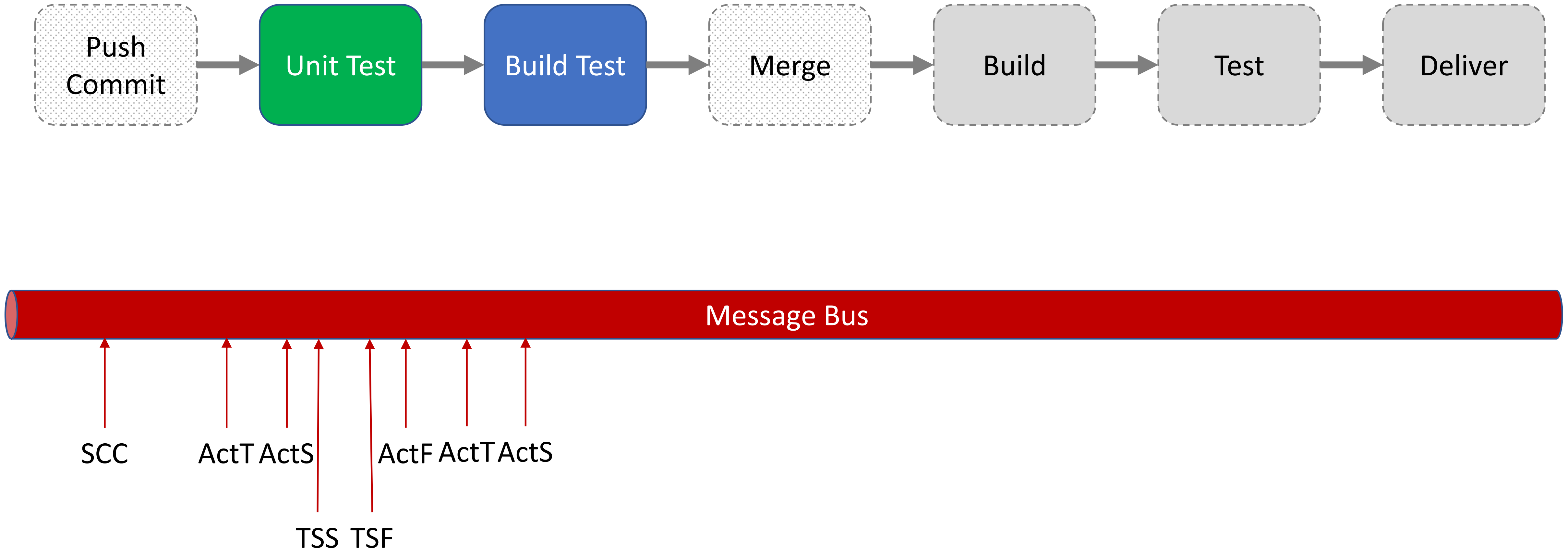


Use Cases

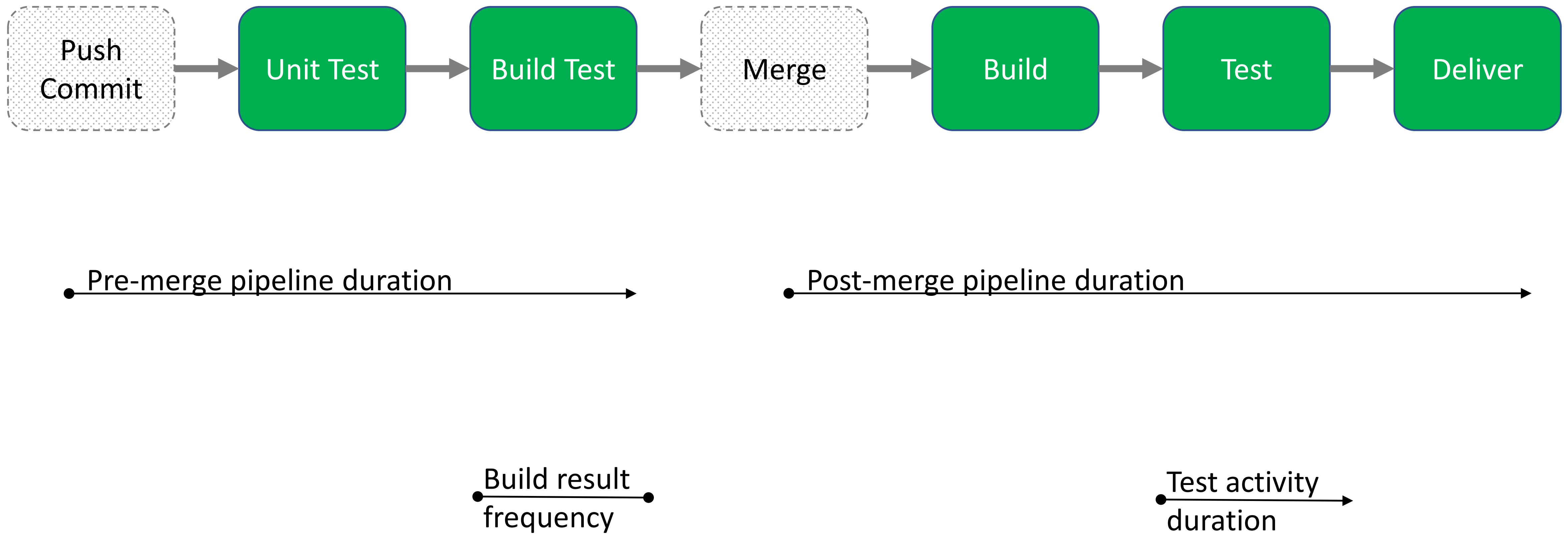
What can the Eiffel data be used for?

- Follow-my-commit visualization
- Pipelines measurements
    - Frequency & duration of
        - complete pipeline, pipeline of pipelines
        - pipeline steps/activities
        - Test suites / test cases
    - Test environment usage
- Dependency triggering (pipeline of pipelines) and traceability
- and more …

Follow-my-commit



Pipeline measurements



Pipelines of Pipelines

Communicating & tracing between pipelines

- Can be used to trigger and/or trace connected pipelines

Guiding Principles

“What to communicate is volitional; how to communicate it is not”

“Opinionated but open-minded”

Designed to encourage good practice, while still allowing diverse ways of working

Technology agnostic, flexible, scalable, traceable

See event design guidelines

Concerned about Security

What about CloudEvents?

This are *my* findings after some research on CloudEvents (no guarantees on correctness):

- At a first glance it looks quite similar:
  - It should be used to communicate *events* (descriptive, e.g. ‘this happened’), not *messages* (prescriptive, e.g. ‘do this’)
  - Structured as JSON data objects
  - A meta part and a data part in the JSON object
  - Independent of the communication channel (messaging system)
- But also a bit different:
  - CloudEvents does not define any specific data objects, as Eiffel does
  - CloudEvents does not have the concept of linked events
  - CloudEvents can handle binary data objects, Eiffel cannot
  - Eiffel explicitly states that data should not be duplicated in several events. CloudEvents does not care
- Eiffel could maybe become a so called ‘extension’ to CloudEvents
- The meta information in a CloudEvent is quite similar to that of Eiffel, but some adaptions need to be made to make Eiffel fully compatible with CloudEvents

What about Value-stream mapping/management?

- The area that the Eiffel protocol plays in could maybe be referred to as ‘Value-stream mapping’ or ‘Value-stream management’ in lean terms. But at the same time it might be like comparing apples and pears. Eiffel is not intended to describe hardware pipelines/deliveries.
- Value-stream mapping has the purpose to remove waste in the processes it covers. While Eiffel could be used for that as well, e.g. by keeping track of HW test environment usage, but that is probably not its primary use case for Eiffel.

Useful links

- The protocol – Eiffel @ GitHub
- The community – Eiffel Community @ GitHub.io
- Try it for yourself – Eiffel Easy2Use
- Quick intro (YouTube video) – What is Eiffel and why should I care?
  - More in-depth videos are also available on the same YouTube channel

Published Articles in Academia – related to Eiffel

- Ståhl, D., Hallén, K., & Bosch, J. (2016). Continuous integration and delivery traceability in industry: Needs and practices. In *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 68-72). IEEE.

- Ståhl, D., Hallén, K., & Bosch, J. (2017). Achieving traceability in large scale continuous integration and delivery: Deployment, usage and validation of the Eiffel framework. *Empirical Software Engineering*, *22*(3), 967-995.

- Ståhl, D., & Bosch, J. (2017). Cinders: The continuous integration and delivery architecture framework. *Information and Software Technology*, *83*, 76-93.

- Ståhl, D., & Bosch, J. (2018). Dynamic Test Case Selection in Continuous Integration: Test Result Analysis using the Eiffel Framework. *Analytic Methods in Systems and Software Testing*, *405*.

Eiffel