Eiffel for beginners

Magnus Bäck, Axis Communications Emil Bäckmark, Ericsson Mattias Linnér, Ericsson



Proposed Outline

- (Emil) Present agenda
- (Magnus) Intro, 40m
- Coffee break
- (Emil) Ericsson use cases, 20m
 - OneRAN
 - Dependency updates
- (Magnus) Axis use cases, 20m
- Intro wrap-up
 - Emil to look in to preparing this
- (Mattias) Eiffel Community Landscape, 10m



Agenda

- Eiffel Introduction
- <Break>
- Eiffel Use Cases at Ericsson
- Eiffel Use Cases at Axis Communications
- (Eiffel Community Landscape)

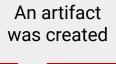


Prelude









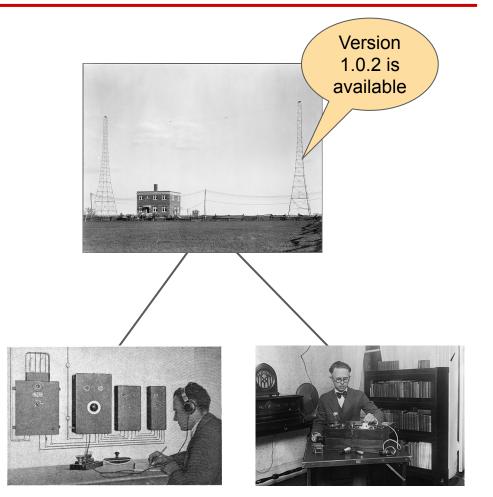
The tests passed

The artifact was deployed

time

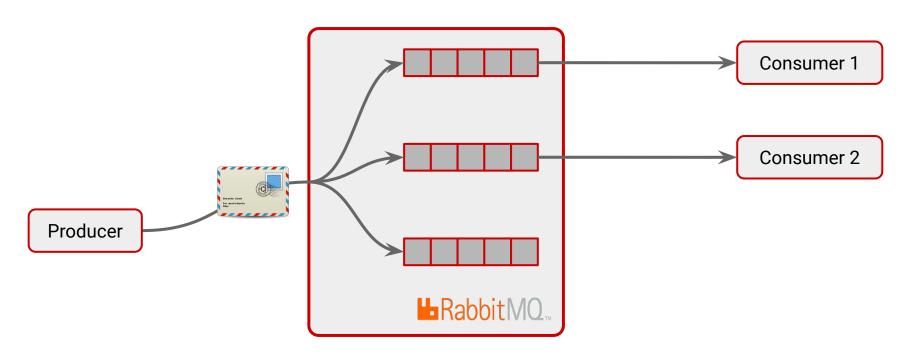






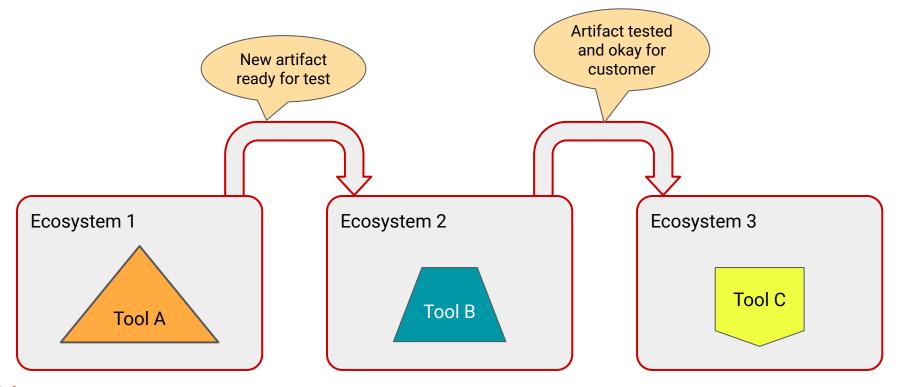


Asynchronous messaging





Using events to implement ecosystem interoperability





- Builds
- Test executions
- Promotions
- Deployments

Entities

- Source code
- Build artifacts
- Test results
- Tickets (issues)
- Environments



What does an Eiffel event look like?

```
Uniquely identifies this event
"meta":
  "type": "EiffelArtifactCreatedEvent"
                                                 Which schema?
  "version": "3.0.0",
  "time": 1631616812000
                                 Time event was generated
'data"
links"
                     Event-specific data
     This event's relationship to other events
```



- Builds
- Test executions
- Promotions
- Deployments

Entities

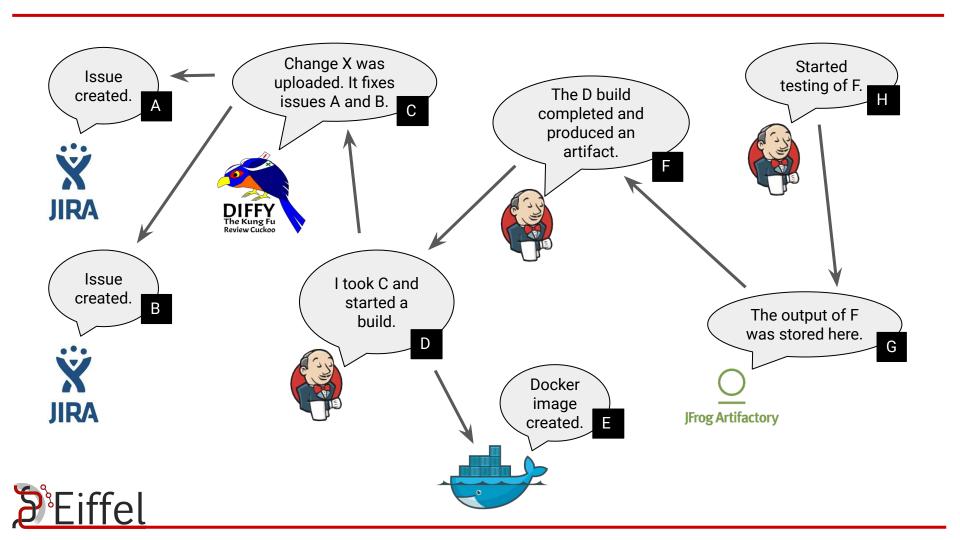
- Source code
- Build artifacts
- Test results
- Tickets (issues)
- Environments



Links describe relationships between events

```
"meta": {
                                             "id": "2342d8c0...",
"meta": {
  "id": "ad064dd7...",
                              CAUSE
                                           "links": [
                                               "target": "ad064dd7...",
"links": []
                                                "type": "CAUSE"
```





Walkthrough of Eiffel event types



Artifacts

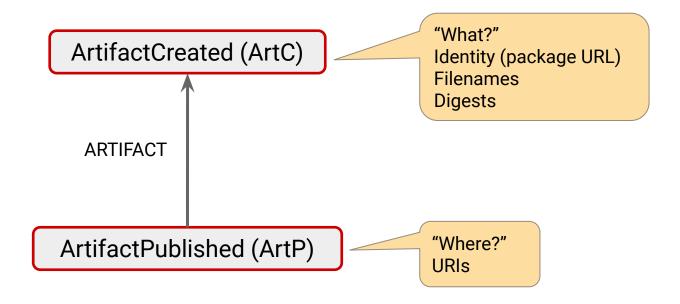
An Eiffel **artifact** is some kind of data that's produced. Often one or more files produced by a build system.

Artifacts are identified with a Package URL (https://github.com/package-url/purl-spec):

```
pkg:deb/debian/curl@7.50.3-1?arch=i386&distro=jessie
pkg:docker/cassandra@sha256:244fd47e07d1004f0aed9c
pkg:gem/ruby-advisory-db-check@0.12.4
pkg:maven/org.apache.xmlgraphics/batik-anim@1.9.1?packaging=sources
pkg:npm/foobar@12.3.1
pkg:pypi/django@1.11.1
pkg:generic/openssl@1.1.10g
```



Artifacts

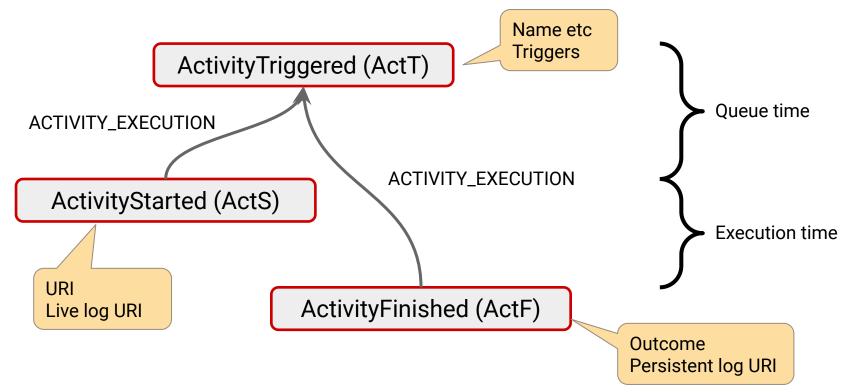




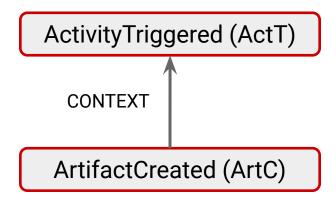
An Eiffel activity describes something that's taking place.

Activities typically don't carry meaning themselves, but they act as useful containers for artifacts and test executions.











Compositions

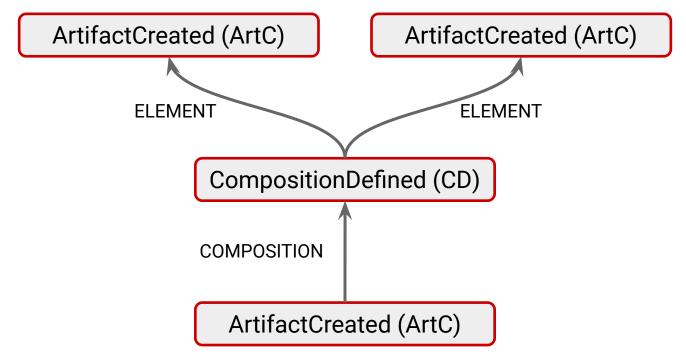
An Eiffel composition groups artifacts, source changes, or other compositions.

A composition has a name and optionally a version.

Artifacts use compositions to indicate what they were created from.



Compositions





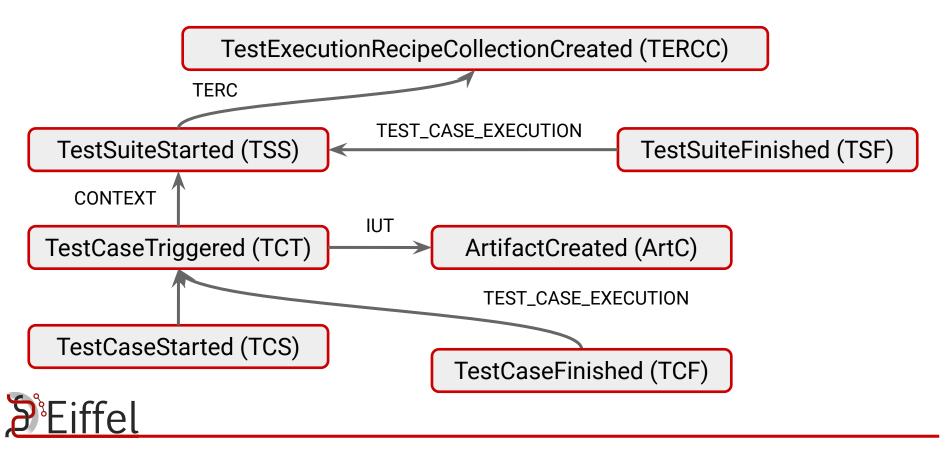
Tests

Eiffel defines various test-related events that describe what goes on during testing.

Like artifact creations tests are often connected to an Eiffel activity.



Tests



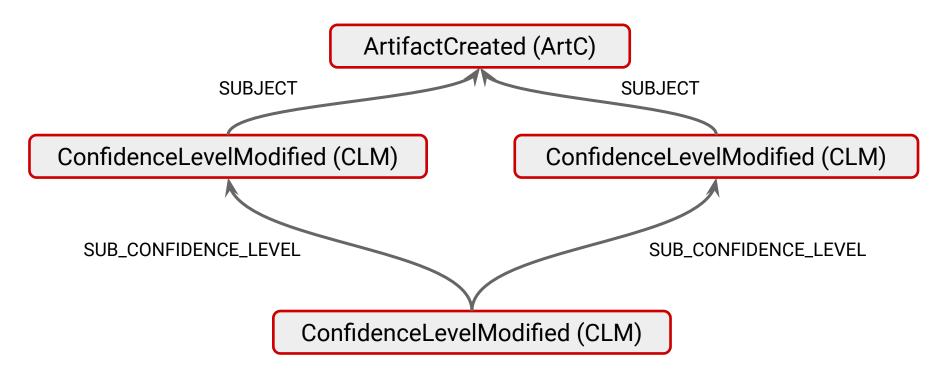
Confidence levels

An Eiffel **confidence level modification** declares that a set of Eiffel entities has been evaluated and found to achieve or not achieve a particular level of confidence for someone.

Example: "We're confident that baseline X is releasable".



Confidence levels





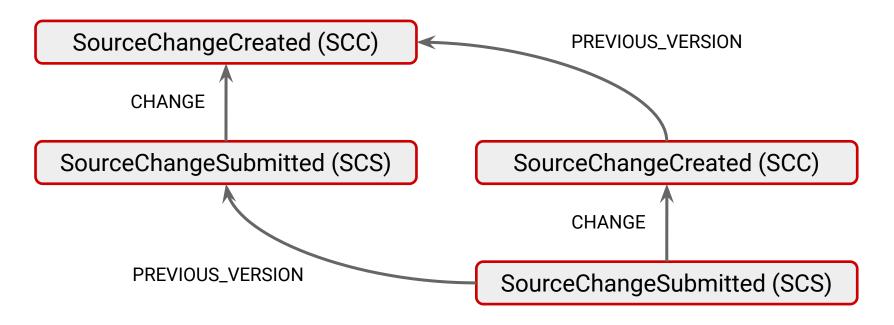
Source changes

Eiffel source change events describe what goes on in source code repositories.

Thanks to the links, source change events can represent the whole history of a repository.



Source changes





Ericsson use cases







Background and #1 Use Case provided by Kristofer Hallén, Ericsson





Ericsson Loves Events in CI/CD - since 2012 -



"There must be something we did right since we are still getting value from concepts defined more than ten years ago"

> CICD Architect Ericsson



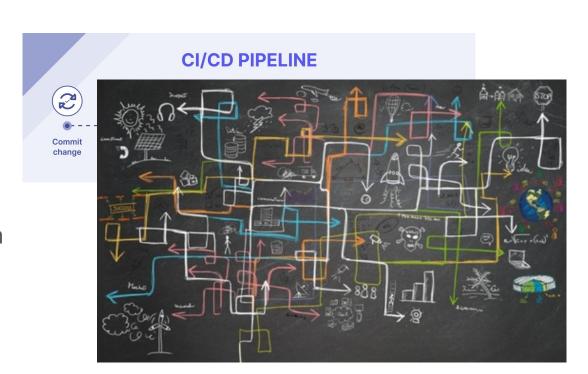
1,000,000s of Eiffel events per day

Connecting pipelines, triggering
Collecting product information
Providing data for visualization of
products and pipelines





Our world ...



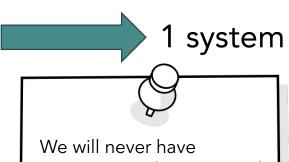
... is not like in the books





Our world ...

- 1,000s of software components
- 1,000s of commits per day
- 100,000s of test runs per day
- 1,000s of developers, on three continents
- From embedded to cloud platforms
- Radio signals
- ~40 development and CI/CD tool stacks.



We will never have everyone on the same tool stack.

Manual orchestration does not work.











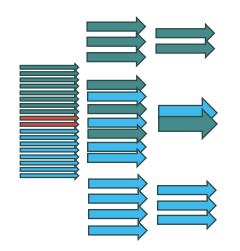




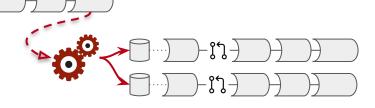
Ericsson Use Cases - Intro

S

- 1. SW Pipeline Assembly
 - Assemble Huge SW Pipelines in Days



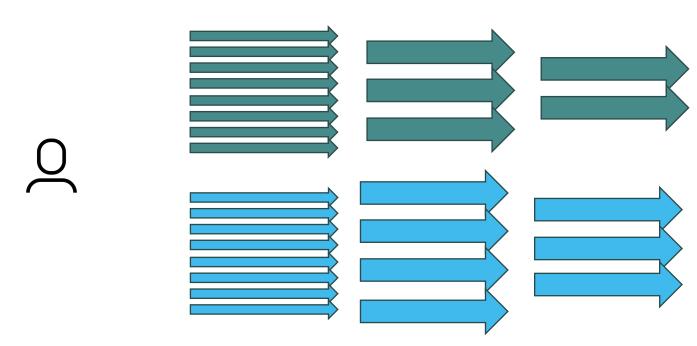
- 2. Dependency Updates
 - General purpose 'Dependabot'





Ericsson Use Cases - SW Pipeline Assembly The Starting Point



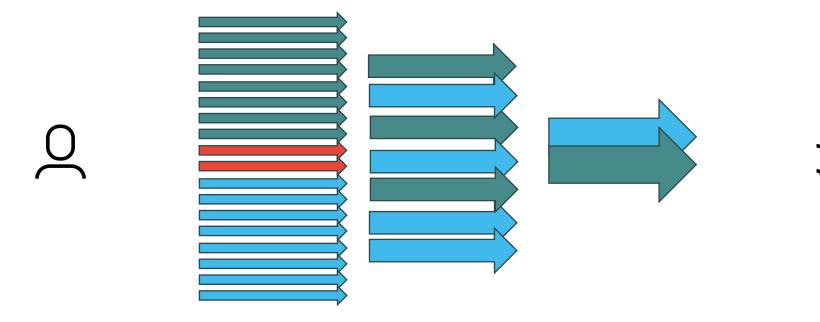






Ericsson Use Cases - SW Pipeline Assembly The Challenge



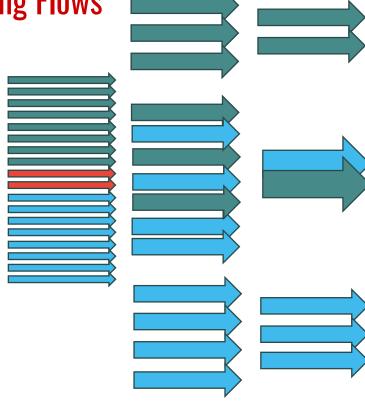
















Ericsson Use Cases - SW Pipeline Assembly Thanks to the Events!





Decoupled but Connected pipelines

Visualized pipeline progress

Collecting product information

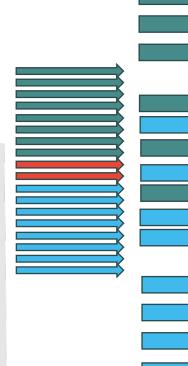
Collecting quality information



Fast

Cheap

Minimal impact on existing pipelines





Ericsson Use Cases - SW Pipeline Assembly Reflections



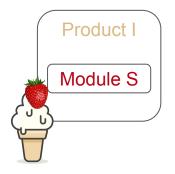


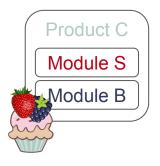
Infrastructure, best practice and schemas need central ownership

Some use cases are more important than others

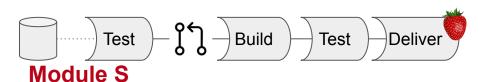


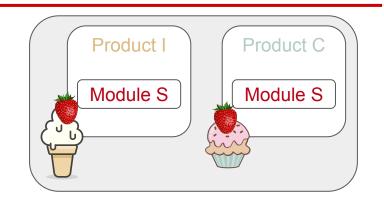






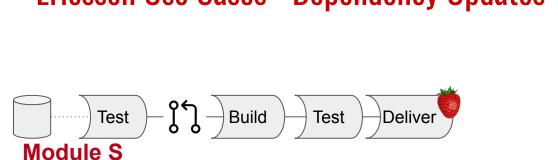


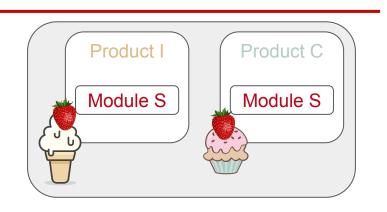


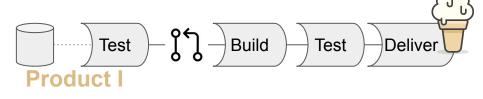


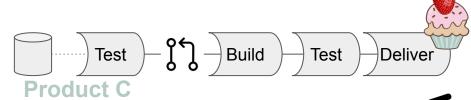












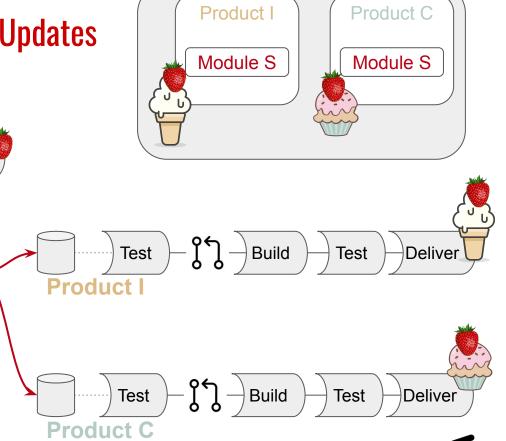




Test

Deliver

Build

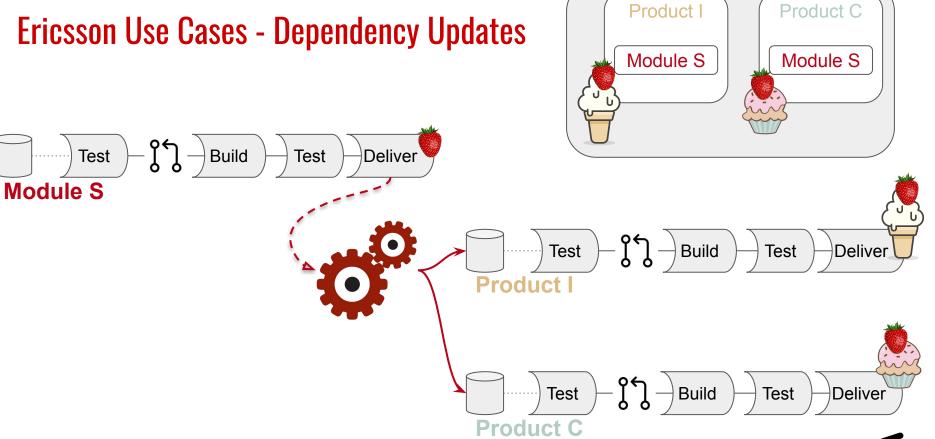




Test

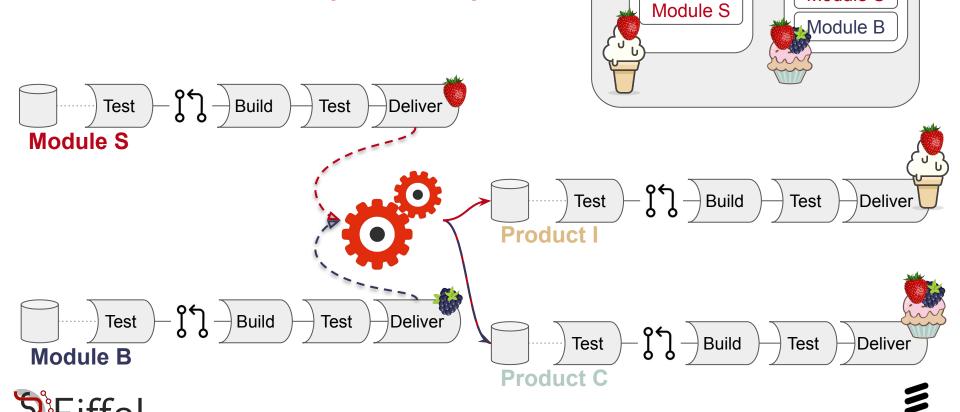
Module S











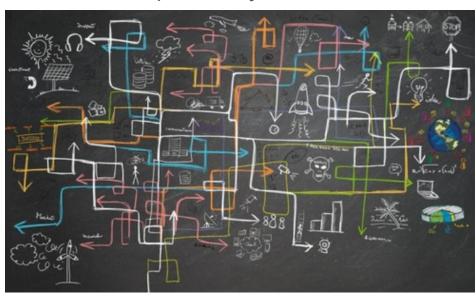
Product I

Product C

Module S

ERICSSON

What if the dependency tree looks like this?



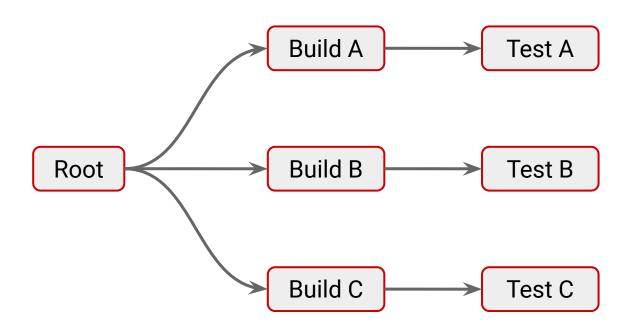




Axis use cases

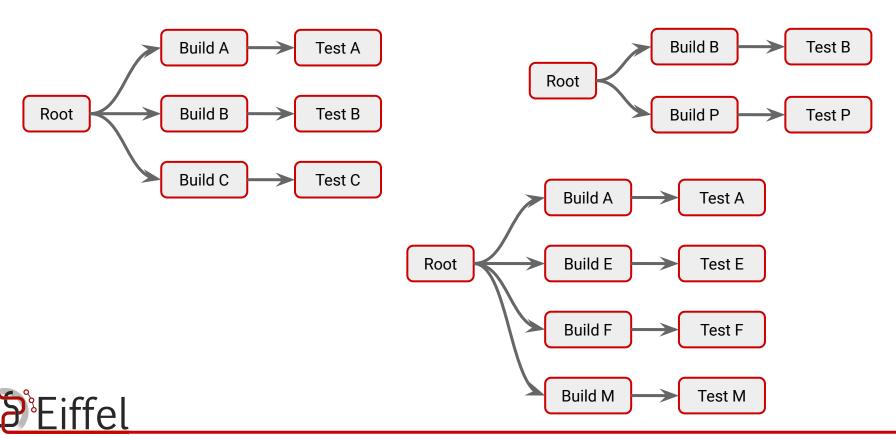


A simple pipeline for testing products running AXIS OS

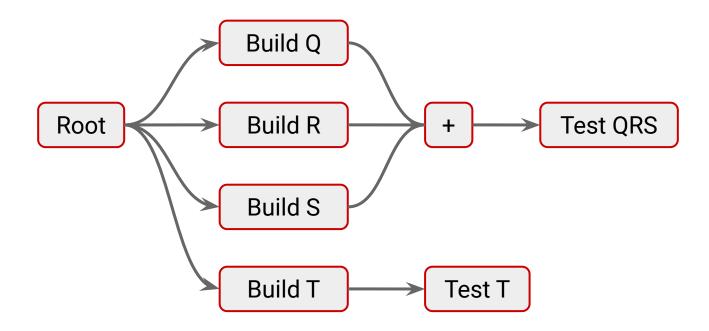




Actual pipelines for testing products running AXIS OS

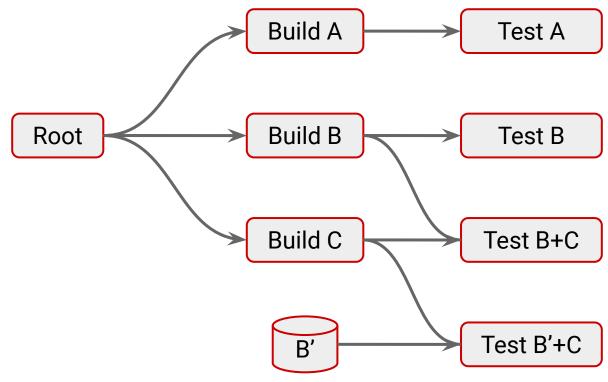


Fan-ins



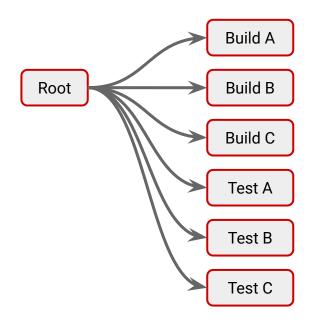


Testing systems of products



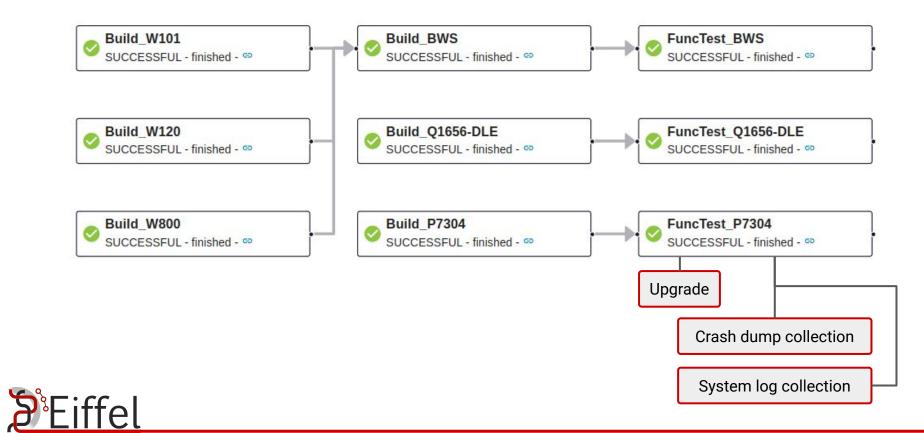


Solution: Implement the scheduling in a Jenkins pipeline





Solution: Eiffel-driven visualizer



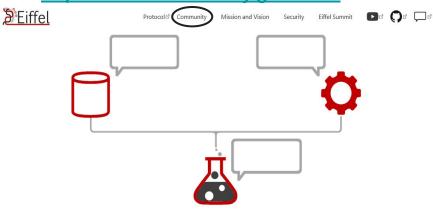
The Eiffel landscape

An overview of the projects in our landscape

Mattias Linnér - Ericsson

How to find the landscape

https://eiffel-community.github.io/



Eiffel

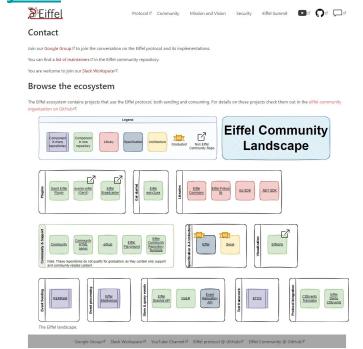
The Eiffel protocol enables technology agnostic communication for CI/CD eco systems.

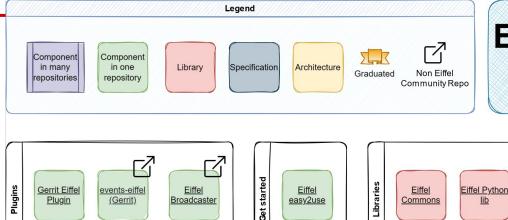
Eiffel is based on the concept of decentralized real time messaging providing traceability

and KPIs throughout your pipelines, across platforms.

Read more about our Mission and Vision

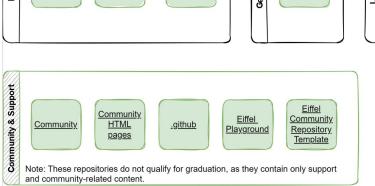
https://eiffel-community.github.io/community.html

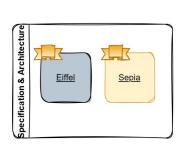




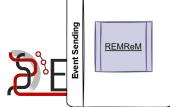


.NET SDK

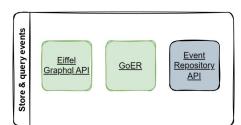


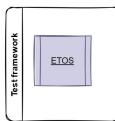




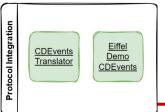








Go SDK



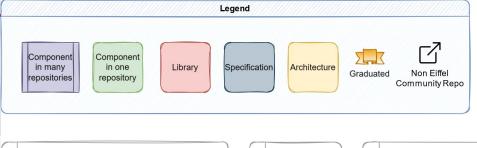
Lifecycle

- Sandbox stage is for the projects that are at the early stages of their development lifecycle such as newly created projects or the projects with heavy initial development going on.
- Graduated stage is for the projects that achieved production level functionality and quality. Eiffel Community expects projects in Graduated stage to be active, both within the community and towards their users.
- Archived stage is for the projects that do not expect further development or maintenance and may very well be in dormant state.

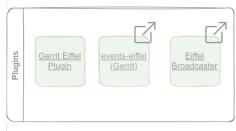
Stage Sandbox

Stage Graduated

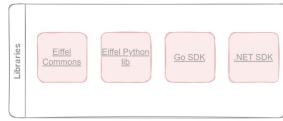
Stage Archived



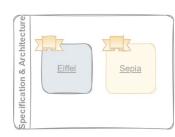


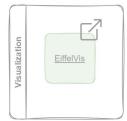












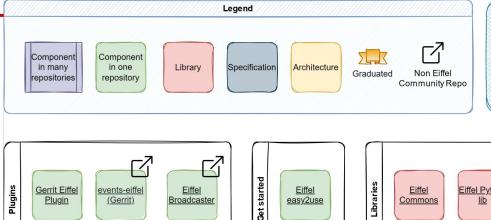




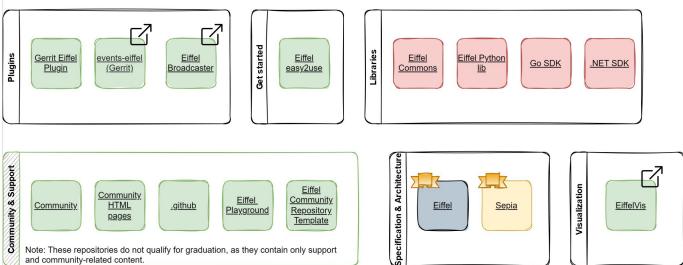






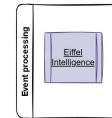


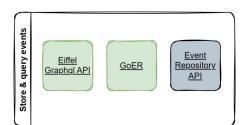


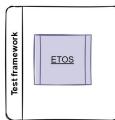


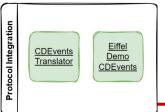


REMReM









SDKs

Project	Language	Multi-Editi on support	meta.security support	Comment
eiffel-remrem-semantics	Java	No	No	Not a full SDK
eiffel-pythonlib	Python	No	No	Untyped
eiffelevents-sdk-go	Go	Yes	Yes	
eiffelevents-sdk-dotnet	.NET	Yes	Yes	

Questions?