Eiffel Activity/Lifecycle Events

and their relations to other events

in an orchestrated (non event-driven) pipeline execution,
e.g. using Jenkins Pipeline or Argo Workflow

Plot

- How should activity/lifecycle events (ActT, ActS, ActF, ActC) be linked to each other, when representing pipelines and pipeline steps?
- How should activity/lifecycle events be linked to other events, such as ArtC and CLM?
- How should other events, such as TSS and CD be linked to activity/lifecycle events?

Problem

- The Eiffel events were defined primarily with an *event-driven* pipeline in mind, where each pipeline step was triggered by an event emitted from another pipeline step
- Most CI systems today rely on an orchestrator, fed with a well-defined pipeline description, to trigger and execute each step of a pipeline
- The issue is primarily how to make best use of the CAUSE links, and secondarily of the CONTEXT links between the activity/lifecycle events
- Is there a reason to update the protocol with new link type(s) for activity/lifecycle events?

Concepts

- *Activity* events or *lifecycle* events, or both? Or *job* events? In this presentation we don’t make any difference between them, and we use the term *activity* events to cover all those semantics.

Why activity events?

- Activity events serve multiple purposes:
  - Describe what pipeline steps are executed in a given pipeline execution (often triggered by an SCM activity), including what is currently ongoing and what is already finished - Provide base for progress dashboards and follow-your-commit visualizations
  - Provide base for KPIs/measurements and allow monitoring of e.g. system performance, bottlenecks, queue times and execution durations

Event types

*with mandatory fields in **bold***

ActT – EiffelActivityTriggeredEvent
- **data.name**
- data.categories[]
- data.triggers[{**type**, description}]
- data.executionType
- links.CAUSE
- links.CONTEXT
- links.FLOW_CONTEXT

ActS – EiffelActivityStartedEvent
- data.executionUri
- data.liveLogs[{**name**, **uri**}]
- **links.ACTIVITY_EXECUTION**
- links.PREVIOUS_ACTIVITY_EXECUTION
- links.CAUSE
- links.CONTEXT
- links.FLOW_CONTEXT

ActF – EiffelActivityFinishedEvent
- **data.outcome**{**conclusion**, description}
- persistentLogs[{**name**, **uri**}]
- **links.ACTIVITY_EXECUTION**
- links.CAUSE
- links.CONTEXT
- links.FLOW_CONTEXT

ActC – EiffelActivityCanceledEvent
- data.reason
- **links.ACTIVITY_EXECUTION**
- links.CAUSE
- links.CONTEXT
- links.FLOW_CONTEXT

Activity Events Explained

- From Confidence Level Joining example:
  - A set of EiffelActivityTriggeredEvent, EiffelActivityStartedEvent, EiffelActivityFinishedEvent reporting on the lifecycle of two independent activity executions being caused by the publication of the artifact. Note that the EiffelActivityStartedEvents and EiffelActivityFinishedEvents are "dead ends" in the graph: in this example, nothing occurs as a direct cause of the activity starting or finishing (although such a setup is, of course, entirely possible) and the work being done within the activity refers directly to the EiffelActivityTriggeredEvent as its context. This shows how, in one sense, this type of lifecycle events is superfluous: for the core functionality of this example it is perfectly possible to cut them out and let the EiffelTestCaseStartedEvents refer directly to **ArtP1** as their cause. Activity events do provide important contextual information, however, as they allow monitoring of e.g. system performance, bottlenecks, queue times and execution durations. They also serve the purpose of clustering work; in this example, **ActT1** and **ActT2** may represent activities executed at wildly different times, at different locations and by different organizations.

CAUSE vs CONTEXT

- The docs says:
  - **CAUSE** (all events)
    - Legal targets: Any
    - Multiple allowed: **Yes**
    - Identifies a cause of the event occurring. SHOULD not be used in conjunction with **CONTEXT**: individual events providing **CAUSE** within a larger context gives rise to ambiguity. It is instead recommended to let the root event of the context declare **CAUSE**.
  - **CONTEXT** (all events)
    - Legal targets: EiffelActivityTriggeredEvent, EiffelTestSuiteStartedEvent
    - Multiple allowed: **No**
    - Identifies the activity or test suite of which this event constitutes a part.
  - **ACTIVITY_EXECUTION** (ActS, ActF, ActC)
    - Legal targets: EiffelActivityTriggeredEvent
    - Multiple allowed: **No**
    - Declares the activity execution that was started. In other words, EiffelActivityTriggeredEvent acts as a handle for the activity execution. This differs from **CONTEXT**. In **ACTIVITY_EXECUTION** the source carries information pertaining to the target (i.e. the activity started, finished or was canceled). In **CONTEXT**, on the other hand, the source constitutes a subset of the target (e.g. this test case was executed as part of that activity or test suite).

- An event can have multiple CAUSEs but only one CONTEXT
  - The CONTEXT of pipeline step activity events is the pipeline (linking to ActT of pipeline)
  - The CONTEXT of an event within a pipeline step is the pipeline step activity or test suite (linking to ActT of pipeline step)
- *It is recommended to let the root event of the context declare **CAUSE*** (see above on CAUSE)
  - This should be interpreted as the trigger event (ActT) is recommended to declare CAUSE, and events emitted within that activity should declare CONTEXT (linking to that ActT event)
  - ActS, ActF, ActC should not declare neither CAUSE nor CONTEXT, but instead ACTIVITY_EXECUTION linking to ActT of the activity
  - So, the big question is – for a pipeline step ActT event, how should it link to the pipeline activity (to ActT or ActS and with what link type), and to previous pipeline step activities (to ActT or ActF and with what link type)?
  - It's strange to consider a pipeline step being *triggered* by a previous step *triggering*, so if a ActT of a non-initial pipeline step should have a CAUSE it should be to the ActF of the previous step. The initial pipeline step of a pipeline could either be CAUSEd by ActT or ActS of the pipeline, or it could declare its CONTEXT to be ActT of the pipeline

Issues

1. How to link first pipeline step activity events to overall pipeline activity events?
2. How to link subsequent pipeline step activity events to previous pipeline step activity events?
3. How to link pipeline ActF event to last pipeline step events?
4. What about parallelly executed steps?
5. How to link other events to/from activity events?

For each issue:

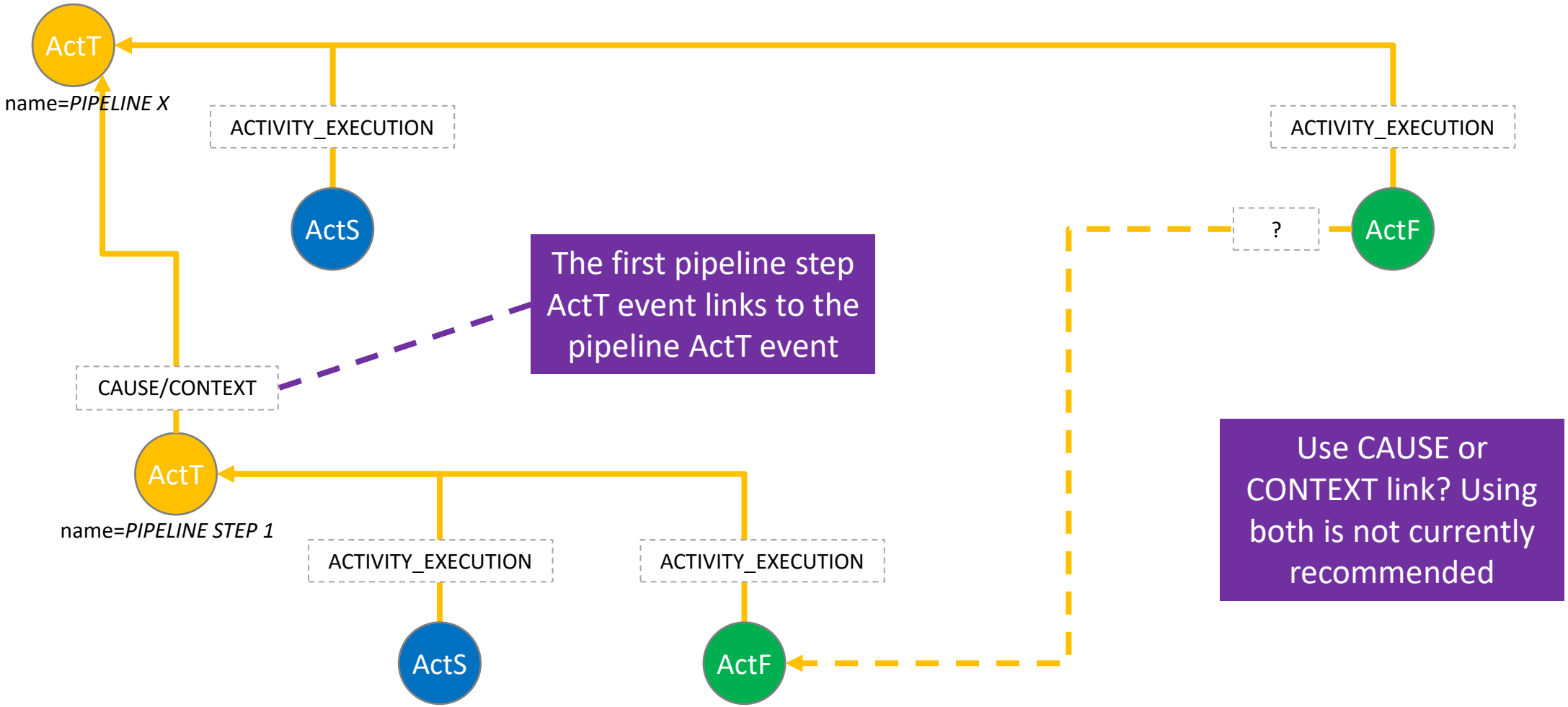
- Are there any concerns w.r.t. upstream/downstream searches? Is it hard to find certain events?

Issue 1
- Linking first pipeline step to pipeline

- A *pipeline step* is never triggered before the *pipeline* is triggered in a non event-driven context, i.e., the ActT of a pipeline step must have a timestamp higher than or equal to ActT of the pipeline
- A *pipeline step* never starts before the *pipeline* starts, i.e., the ActS of a pipeline step must have a timestamp higher than or equal to ActS of the pipeline
- Can a pipeline *start* before the first step is *triggered* or *started*, or should the *start* of the first step indicate that the pipeline has *started*?
- A *pipeline step* start could be CAUSEd by a *pipeline* being triggered, but then the pipeline step ActT should have the CAUSE link to pipeline ActT, and the pipeline ActS should be sent immediately before the pipeline step ActS

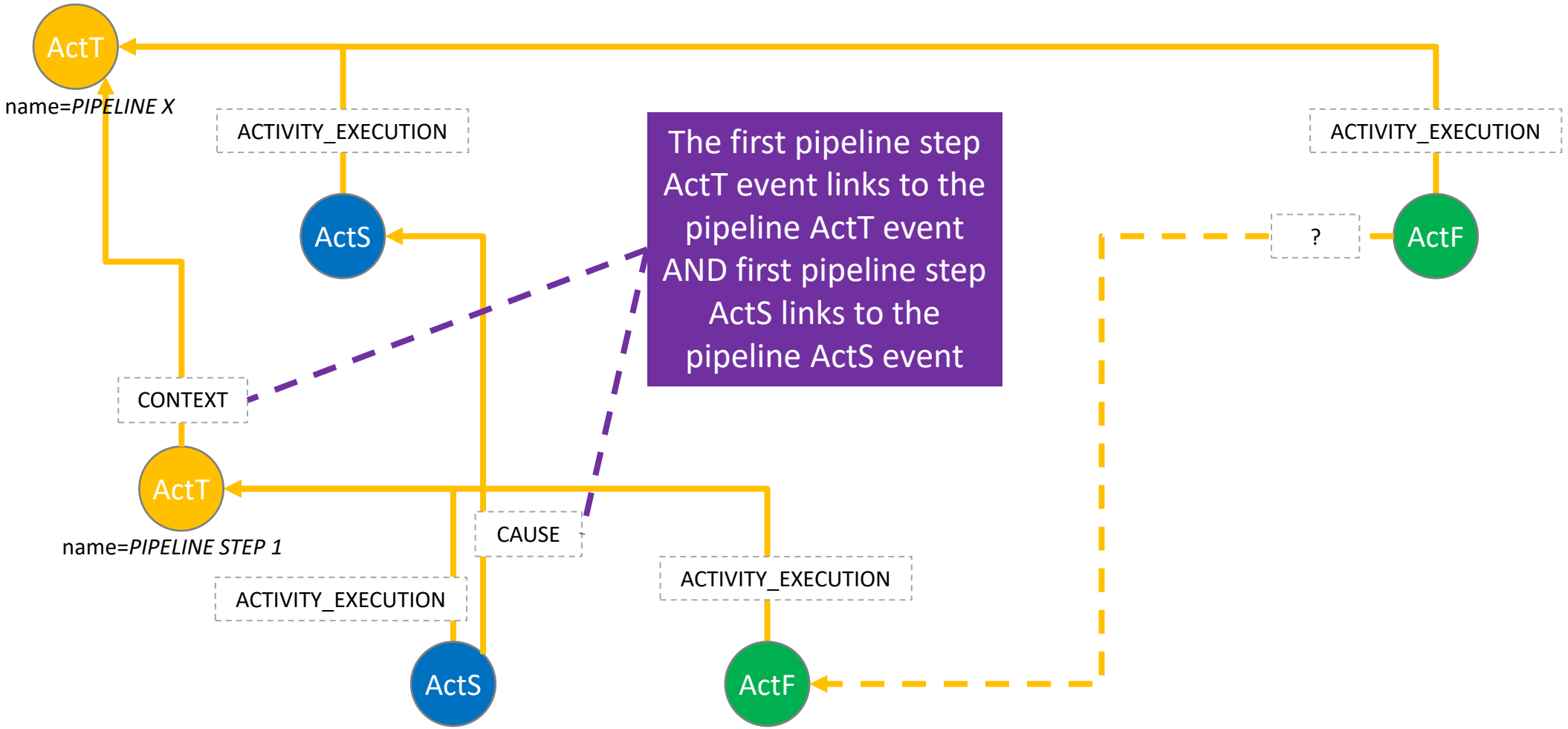
Alternative 1a

- First pipeline step triggered *before* pipeline started (using current spec)



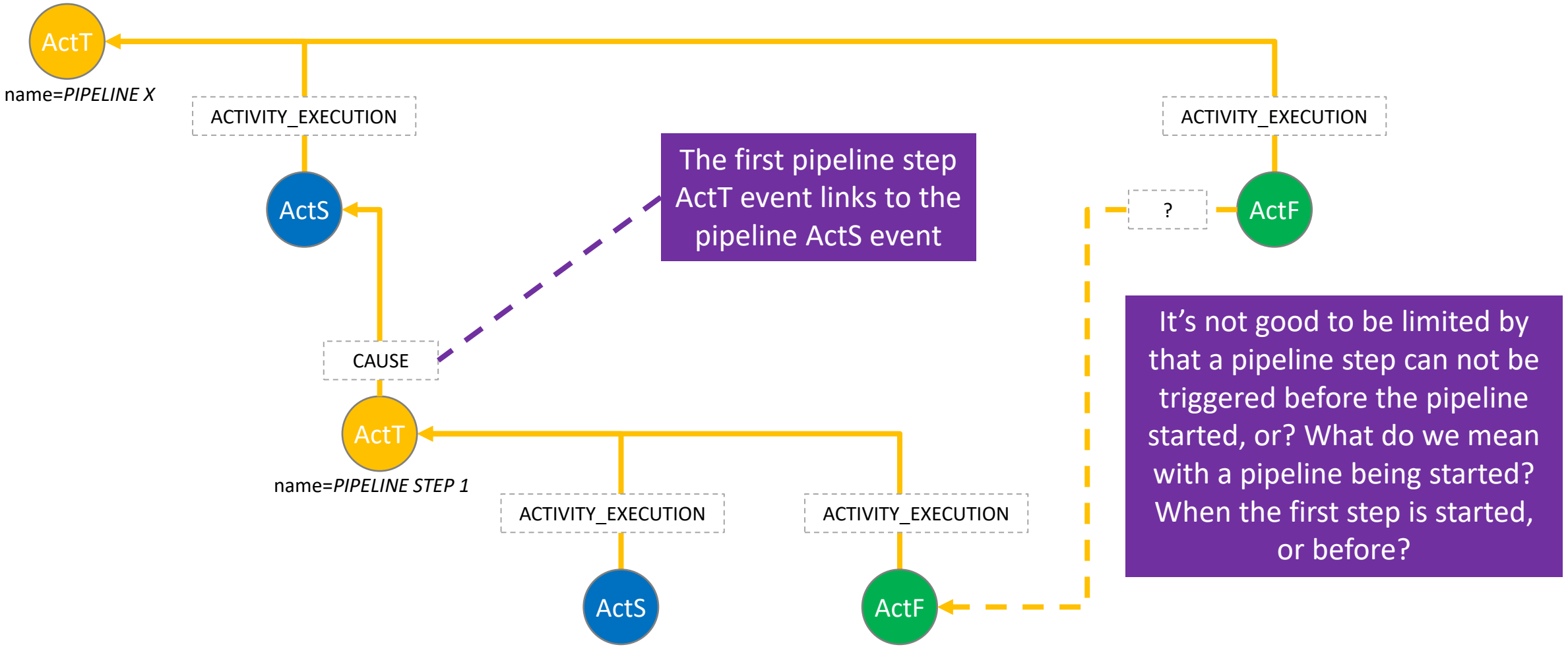
Alternative 1b

- First pipeline step triggered *before* pipeline started (using current spec)



Alternative 2

- First pipeline step triggered *after* pipeline started (using current spec)



Considerations

- General
  - Linking to ActT is more natural as all non-lifecycle events link to the ActT event of the surrounding activity (using CONTEXT link)
  - Alt 2 lacks the possibility to describe CONTEXT of first pipeline step, as it is not recommended to provide both CONTEXT and CAUSE from same event
- Trigger type
  - ActT holds a triggers.type field, which states what triggered the activity. For an event-driven activity flow, it should be set to “EIFFEL_EVENT”, but in this non event-driven kind it should have a different value. Setting it to OTHER is of limited use, so as long as there is no better legal value we should omit the triggers.type (and thereby the triggers object completely) from ActT
- Upstream/downstream searches
  - Alternative 1a/b – Upstream from first pipeline step finds ActT directly
  - Alternative 1b
    - Upstream from first pipeline step ActS finds pipeline ActS
    - Downstream from pipeline ActS can not find pipeline step ActT
  - Alternative 2 – Upstream from first pipeline step finds ActS directly, and ActT secondarily

Conclusion

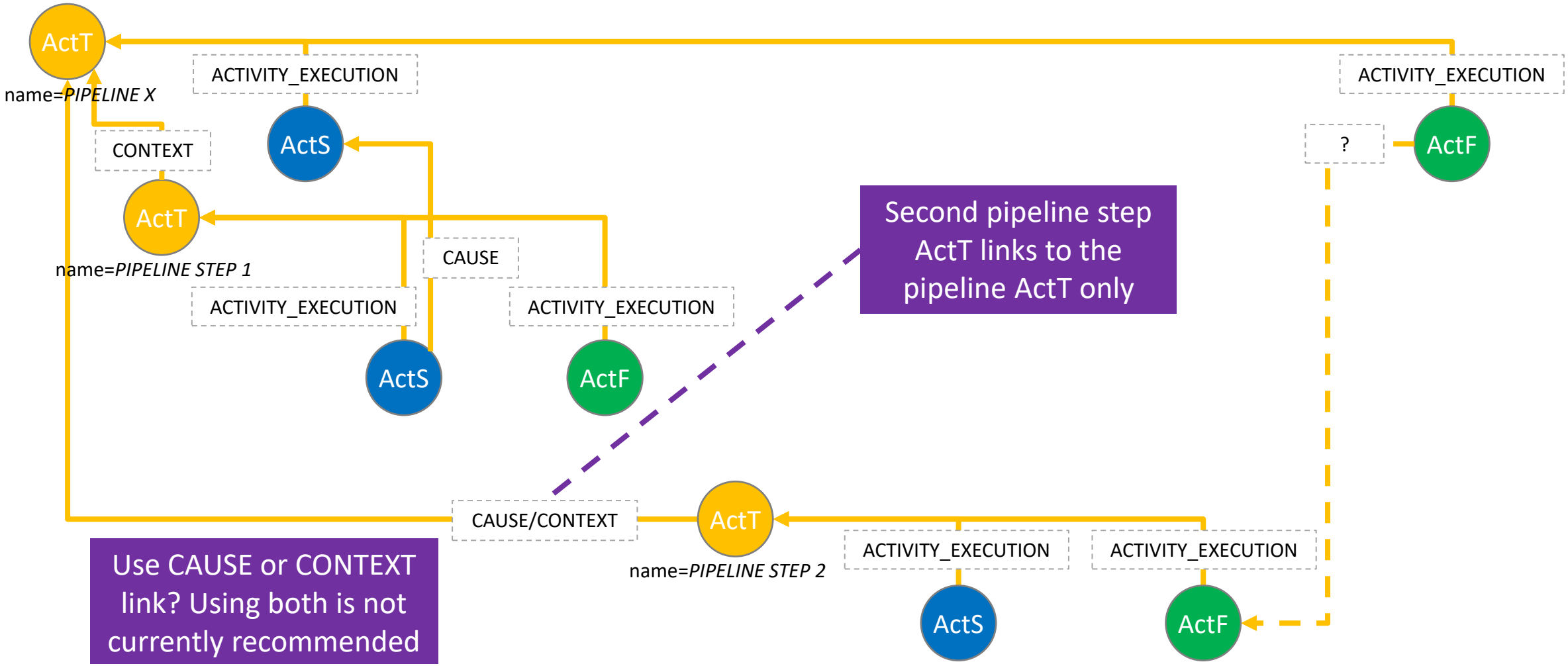
- We should recommend to use alternative 1b, as
  - It gives most flexibility for upstream/downstream searches

Issue 2
- Linking subsequent steps to previous steps

- Should the second pipeline step be considered CAUSEd by the first pipeline step finishing, or not? The orchestrator explicitly triggers the second step after the first is finished, but is that the same as saying that the first step finishing triggered second step?
- The second step ActS could be CAUSED by first step ActF, and second step ActT could have CONTEXT to pipeline ActT?

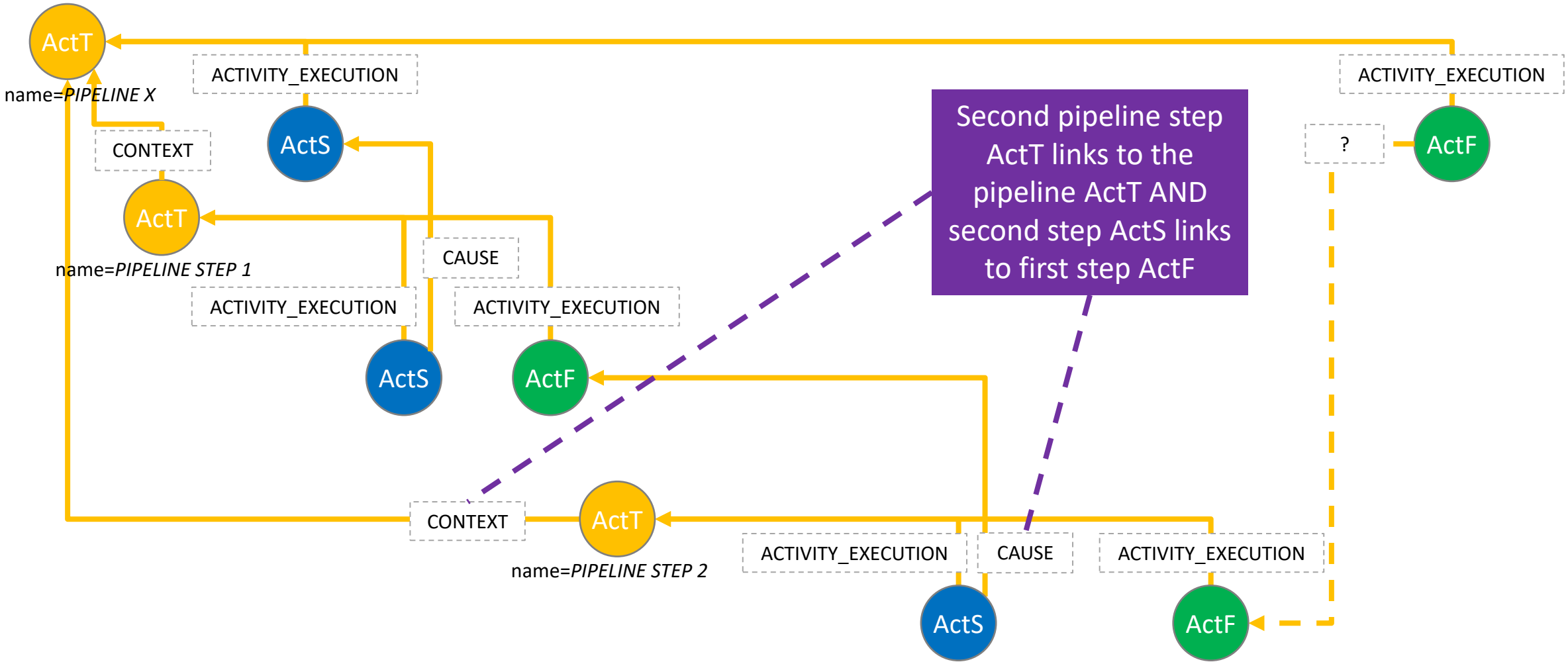
Alternative 1a

- Second pipeline step not CAUSEd by first step events (using current spec)



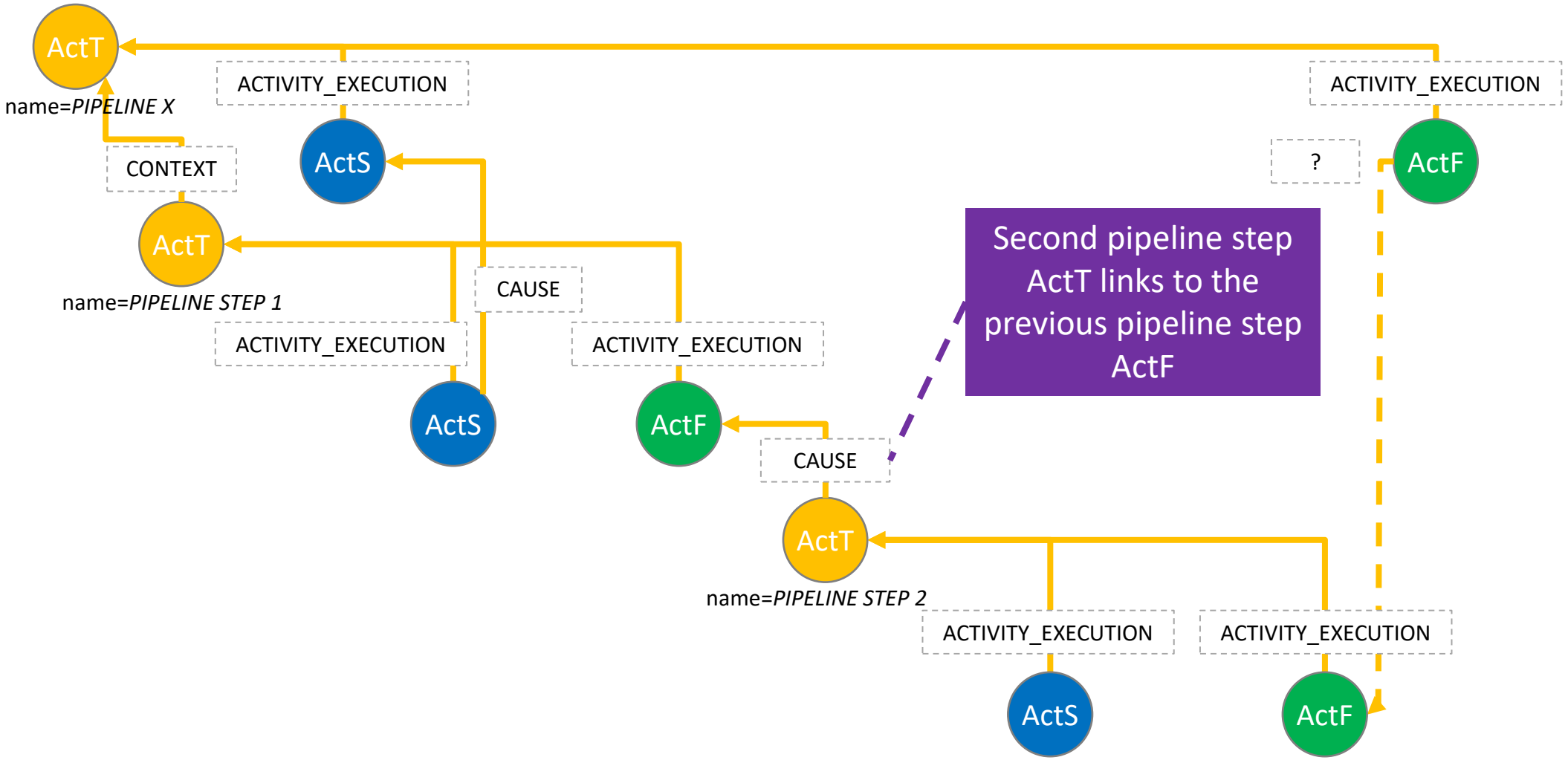
Alternative 1b

- Second pipeline step start CAUSEd by first step finishing, and with ActT CONTEXT to pipeline (using current spec)



Alternative 2

- Second pipeline step triggered by first pipeline step finishing
(using current spec)



Considerations

- Trigger type
  - See trigger type info in issue 1. We recommend not to use the triggers object in ActT for now
- General
  - Alternative 1b is similar to alternative 1b in issue 1
- Upstream/downstream searches
  - Alternative 1b gives most flexibility. Drawback is that first step can not be found by upstream search from second step ActT
  - All steps can be found by one simple downstream search on CONTEXT from pipeline ActT, but the come without order. To also get the order and causality, ACTIVITY_EXECUTION and CAUSE links need to be followed *as well*.

Conclusion

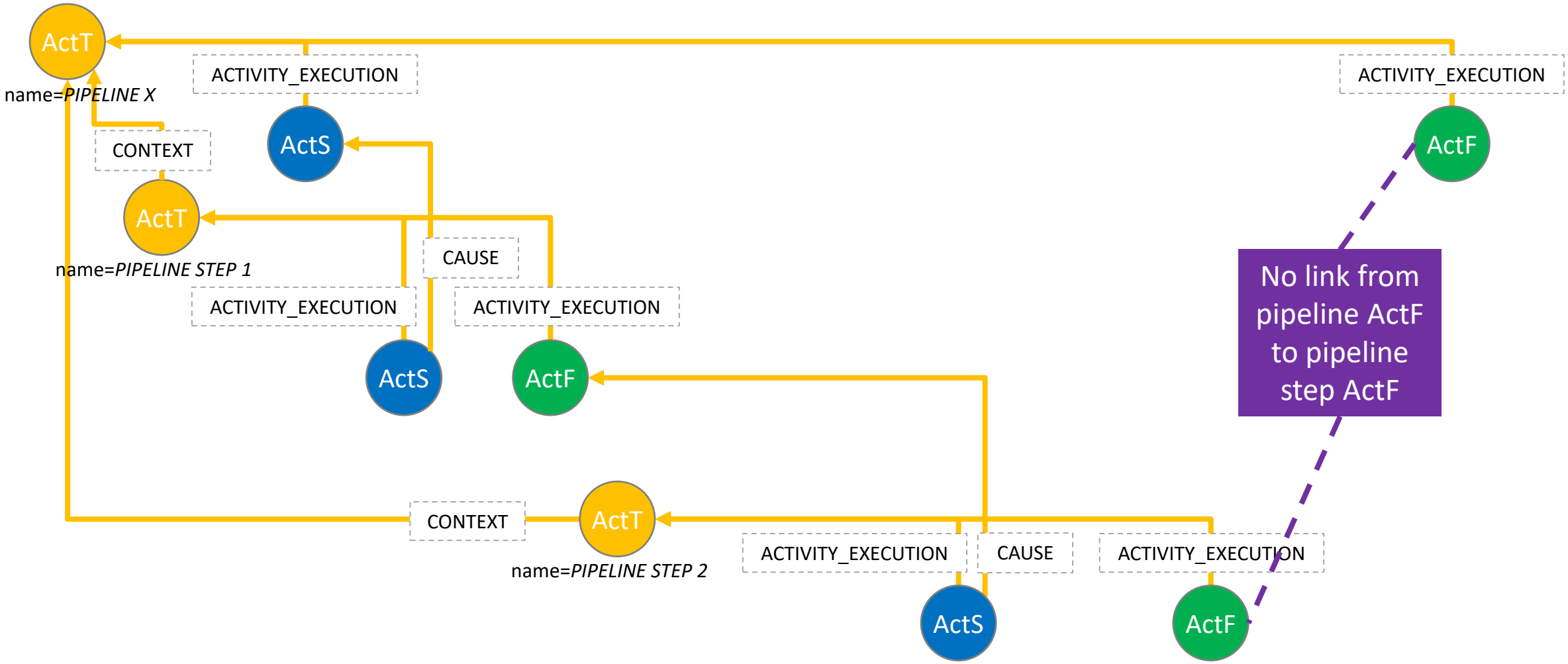
- We should recommend to use alternative 1b
  - It is similar to alternative 1b in issue 1 and no further drawbacks are seen

Issue 3
- Linking pipeline ActF to step events

- Should the pipeline ActF event link to the last pipeline step activity (ActT or ActF) and if so with what link type? CAUSE is probably the best fit, or should a new link type be added for this purpose?
- Generally, and this is for all issues mentioned here, a sub activity to any activity could be seen as something that happens in that activity. Whether that something sends activity events or not should not be a concern of the overall activity. Example:
  - A pipeline step performs some task that does not send any event and therefore ActF of that step has nothing to link to apart from its own ActT
  - If that task starts sending an event, e.g. CLM, why should the ActF all of a sudden be caused by that event?
  - And if that task grows bigger and starts sending its own activity events, should the overall activity bother?
  - I.e. finishing the overall activity should not be caused by the last sub activity finishing
- For this to be symmetrical, we should also consider if ActF for a step should link to any event sent within that step

Alternative 1

- Pipeline ActF has no upstream connection to the pipeline steps
(using current spec)



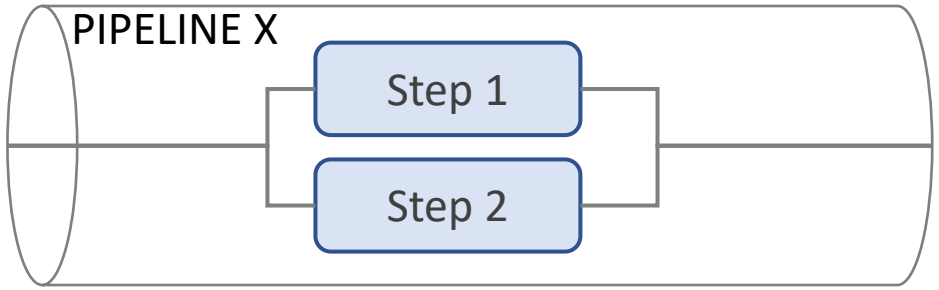
Considerations

- General
  - Only one alternative seems relevant, considering how pipeline steps can get new events added without the pipeline orchestrator knowing about it
- Upstream/downstream searches
  - Finding the pipeline steps from pipeline ActF is simple, as it contains the ACTIVITY_EXECUTION link to pipeline ActT. A simple downstream search from that ActT does it

Issue 4
- Parallel Pipeline Steps

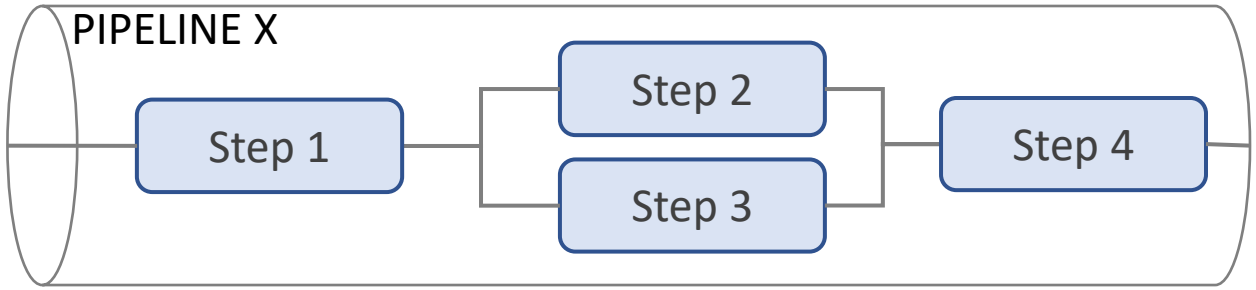
- Are there any considerations when dealing with parallel steps in a pipeline? What should their ActT events link to and with what link type?

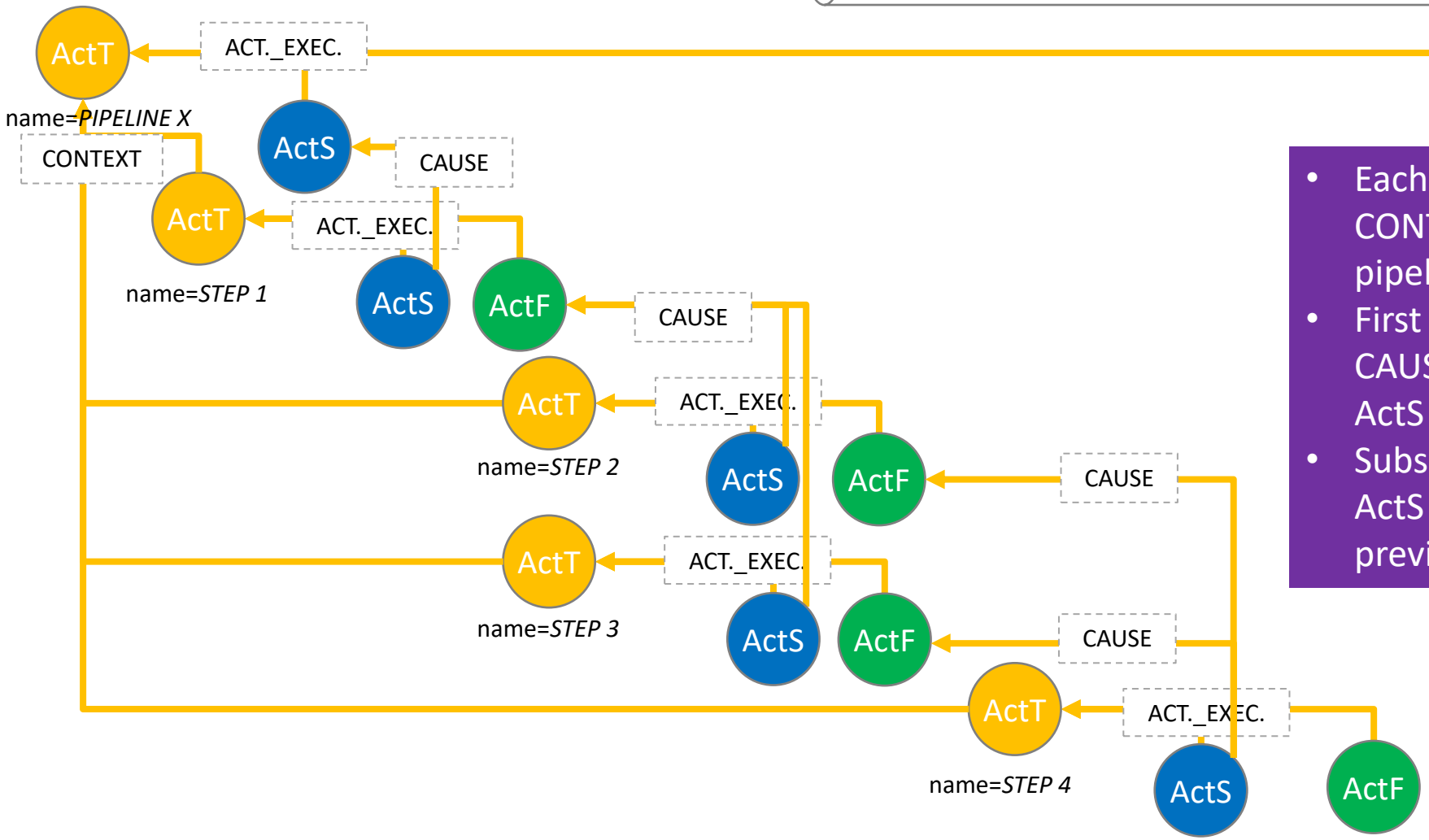
Alternative 1
- Simple Parallel Steps (using current spec)



ActT
name=*PIPELINE X*
ACTIVITY_EXECUTION
ActS
CONTEXT
ActT
name=*PIPELINE STEP 1*
ACTIVITY_EXECUTION
ActS
CONTEXT
ActT
name=*PIPELINE STEP 2*
ACTIVITY_EXECUTION
ActS
CAUSE
ACTIVITY_EXECUTION
ActF
CAUSE
ACTIVITY_EXECUTION
ActF
ACTIVITY_EXECUTION
ActF
Both steps CAUSEd by same event

Alternative 1
- Combined types





- Each step ActT has CONTEXT link to pipeline ActT
- First step ActS is CAUSEd by pipeline ActS
- Subsequent steps ActS are CAUSED by previous step ActF

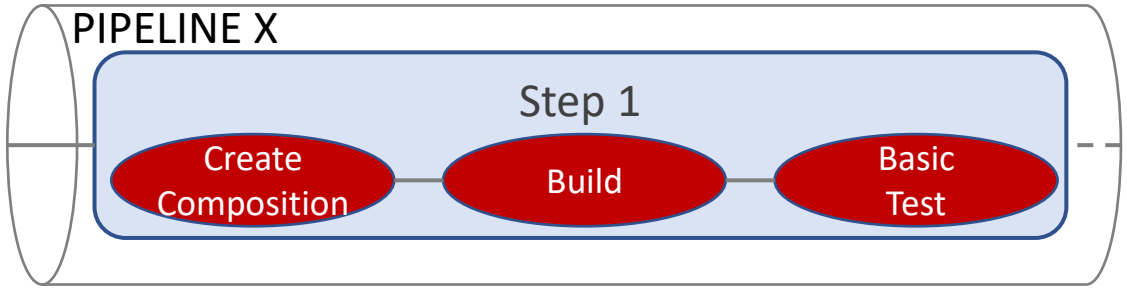
Considerations

- General
  - ..
- Upstream/downstream searches
  - …

Issue 5
- Linking other events to/from activities

- E.g. ArtC to ActT/ActS

Alternative 1
- Linking other events to/from activities



ActT
name=*PIPELINE X*
ACTIVITY_EXECUTION
ActS
CONTEXT
ActT
name=*PIPELINE STEP 1*
CAUSE
ACTIVITY_EXECUTION
ActS
CAUSE
Recommended, but optional CAUSE link
ACTIVITY_EXECUTION
ActF
ACTIVITY_EXECUTION
ActF
No link from pipeline step ActF to step internal events
CONTEXT
CD
COMPOSITION
CONTEXT
ArtC
ARTIFACT
CONTEXT
CLM

Considerations

- General
  - The CAUSE link from CD to pipeline step ActS is there for symmetrical reasons, comparing to how first pipeline step ActS is CAUSEd by pipeline starting. No CAUSE link from subsequent events in the same step for similar reason.
- Upstream/downstream searches
  - Finding the activity start/finish event from arbitrary event within an activity is done by downstream search on the ActT of the activity (which is available as CONTEXT in each event)

MoM from GitHub issue

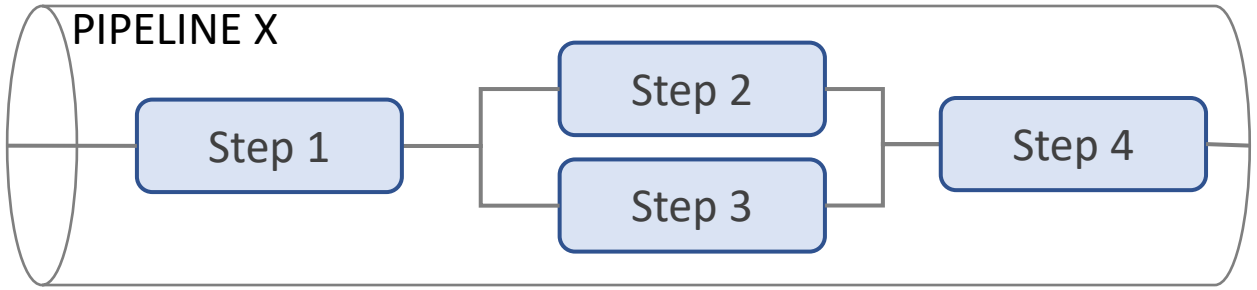
link

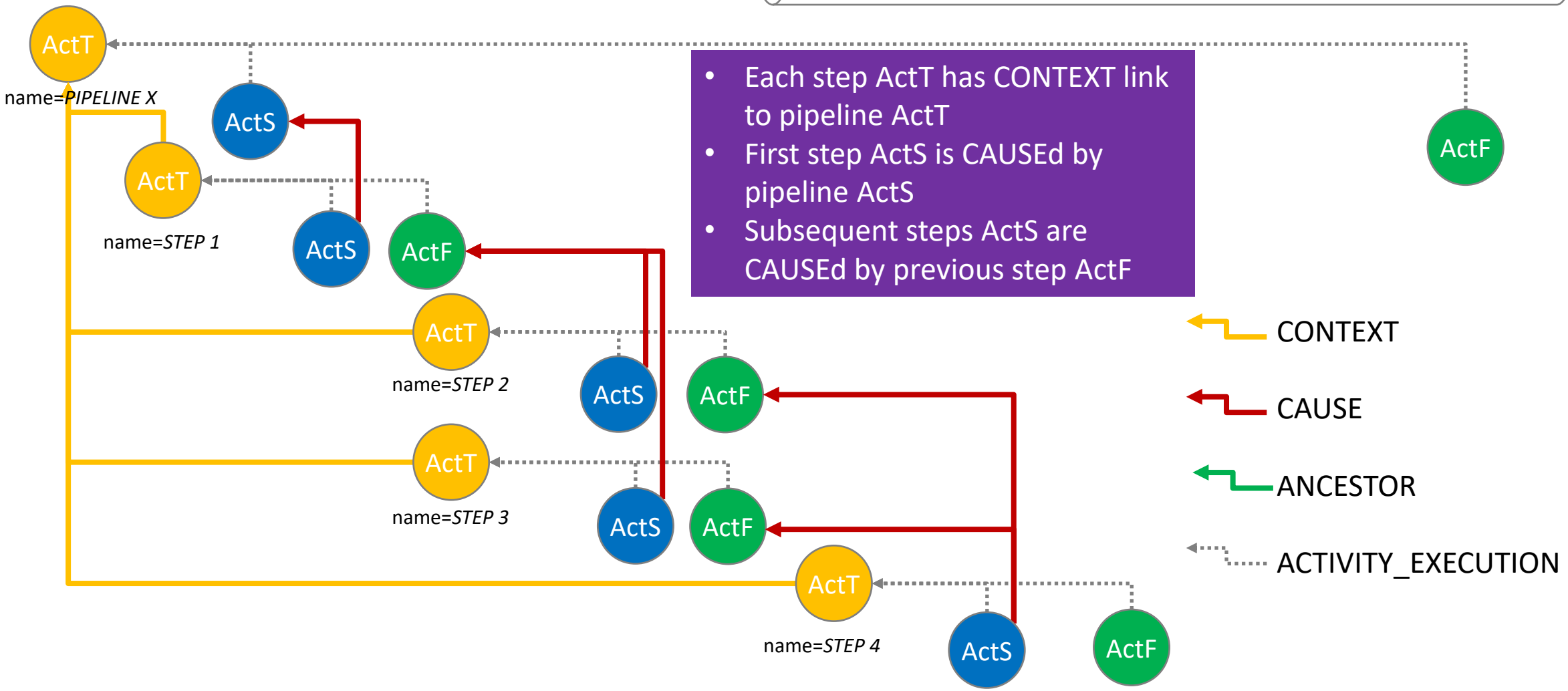
- CONTEXT links should be used when the source event is done as a part of the target event, i.e. it's a substep or subtask. In Emil's colorful example above the Step 1, ..., Step 4 activities should have a CONTEXT link to the Pipeline X activity.
- CAUSE links should be used when the target event is the cause of the source event. This *could* be true for Step 1 and Step 2, where Step 2 can be triggered because of Step 1's ActF. However, the CAUSE link shouldn't target Step 1's ActT.
- We see a need for another link type that conveys that two activities take place sequentially without any explicit causality. ANCESTOR and PREDECESSOR were floated as possible names.
- The end result would be that CONTEXT is used for horizontal/overlapping stacks of activities, CAUSE when there's a strict causality between events, and ANCESTOR/PREDECESSOR when activities have a non-causal relationship and take place sequentially.
- The current recommendation that CONTEXT and CAUSE shouldn't be used in the same source event should be removed or at least reworded. The purpose of the CAUSE event should be more clearly defined.

Additional output

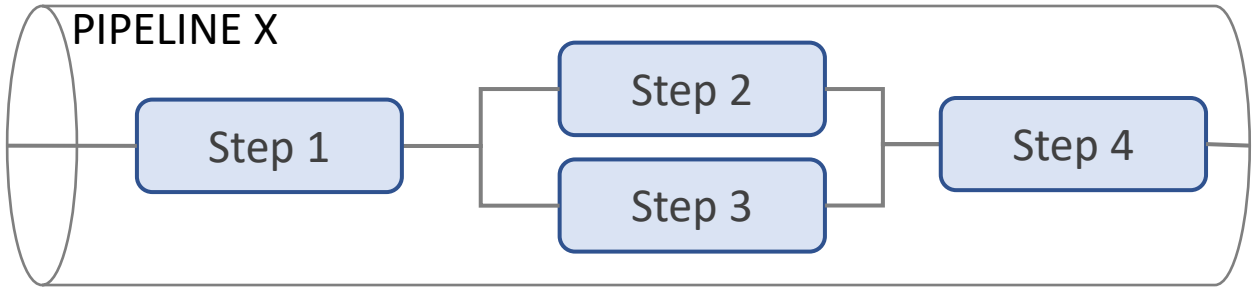
- CONTEXT and CAUSE represent different dimensions in the link structure and should therefore not be mutually exclusive

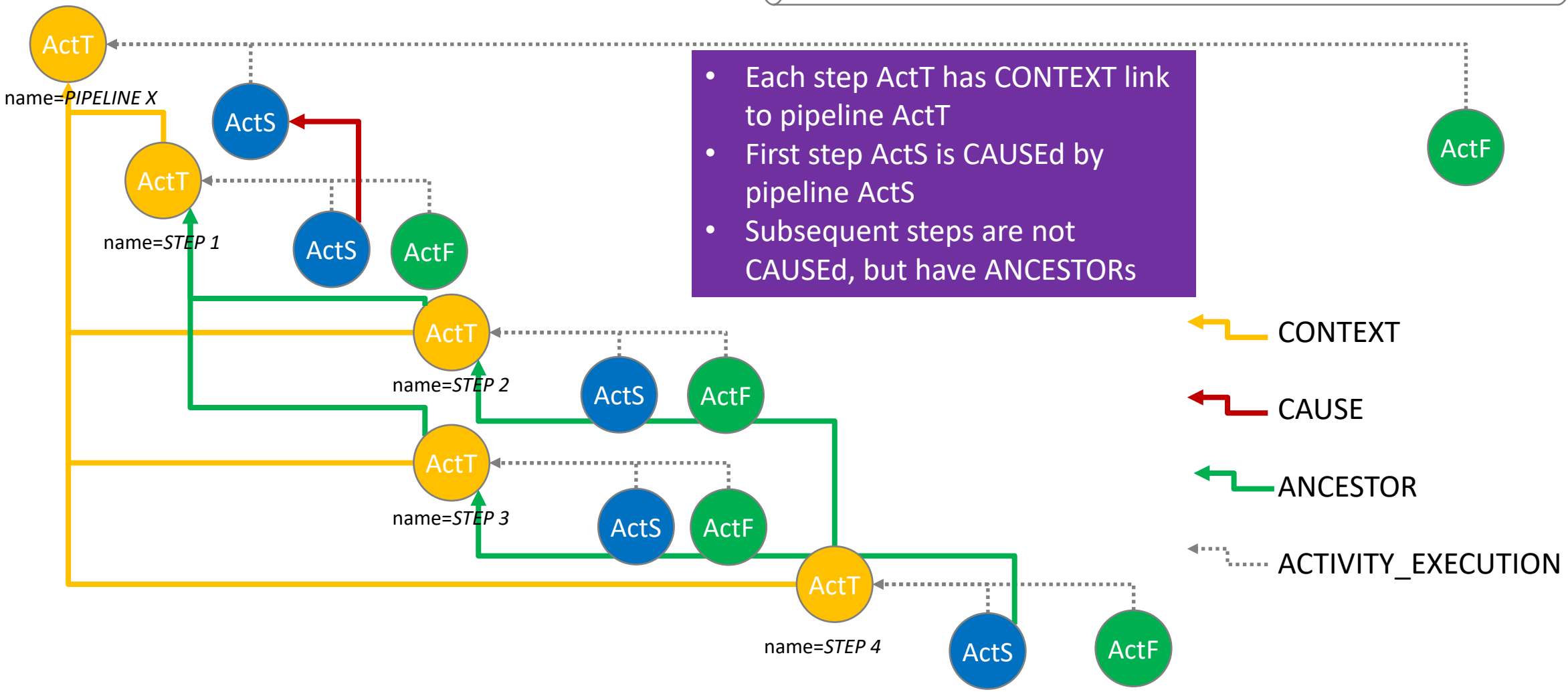
Example: All steps are event triggered





Example: All steps are explicitly triggered





Questions/concerns

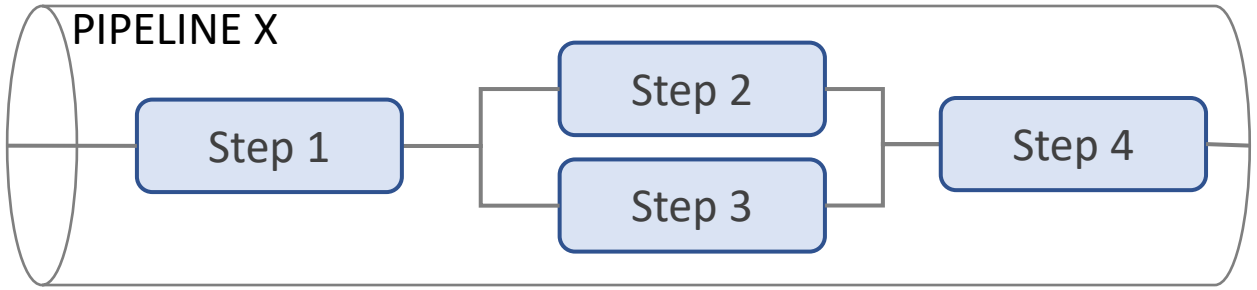
- Should the link type be ANCESTOR or PREDECESSOR or something else? We should not mix it with PREVIOUS…
  - PRECURSOR, ANTEDATE, AFOREGOER, ANTERIOR, ORIGIN, LINEAR_PREDECESSOR, CORRELATOR, INFLUENCER
  - **We decided to go for PRECURSOR**
- When using CAUSE links (targeting ActF), should we also use ANCESTOR links (targeting ActT) to be used to visualize pipeline graphs?
- When re-triggering a pipeline/step, what events should be sent and how should they be linked?

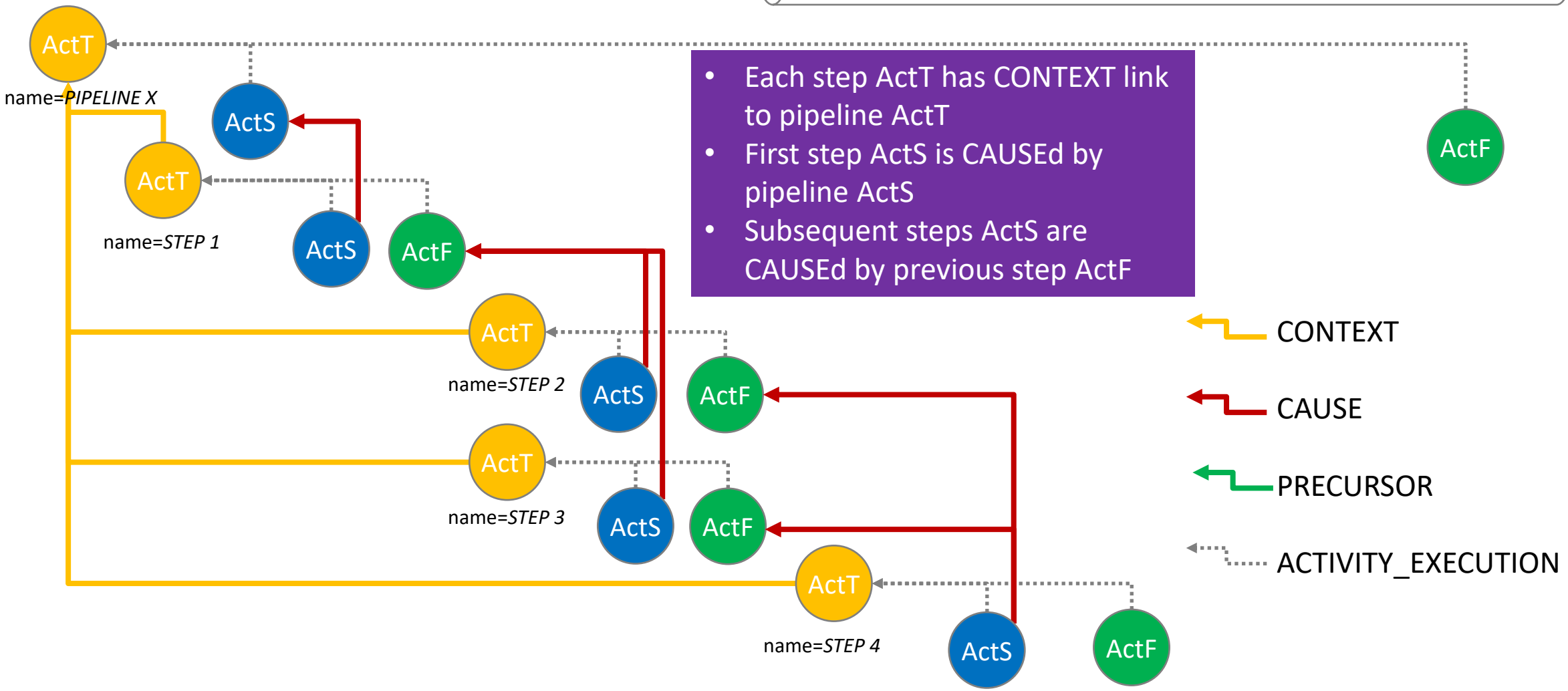
- A **causal relation** between two events exists if the occurrence of the first causes the other. The first event is called the cause and the second event is called the effect. A correlation between two variables does not imply causation. On the other hand, if there is a causal relationship between two variables, they must be correlated.
  https://www.varsitytutors.com/hotmath/hotmath_help/topics/correlation-and-causal-relation

To be updated!

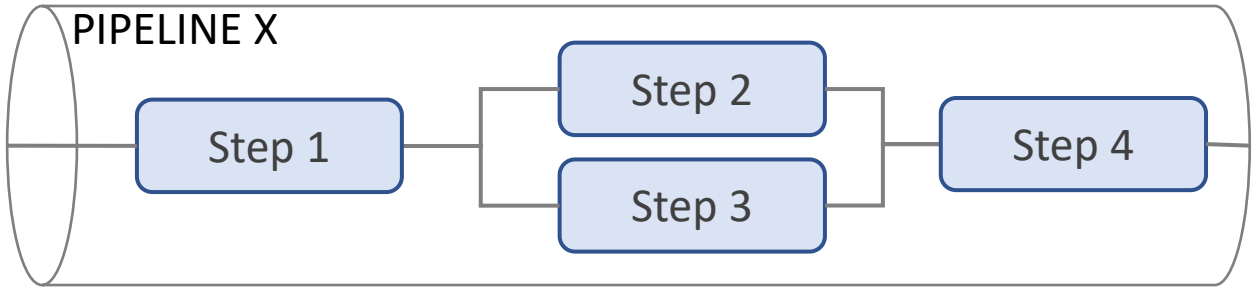
- Current CAUSE definition:
  Identifies a cause of the event occurring. SHOULD not be used in conjunction with CONTEXT: individual events providing CAUSE within a larger context gives rise to ambiguity. It is instead recommended to let the root event of the context declare CAUSE.
- Proposed CAUSE definition:
  Identifies a cause of the event occurring. SHOULD not be used in conjunction with CONTEXT: individual events providing CAUSE within a larger context gives rise to ambiguity. It is instead recommended to let the root event of the context declare CAUSE.
- Proposed ANCESTOR definition:
  Identifies a cause of the event occurring. SHOULD not be used in conjunction with CONTEXT: individual events providing CAUSE within a larger context gives rise to ambiguity. It is instead recommended to let the root event of the context declare CAUSE.

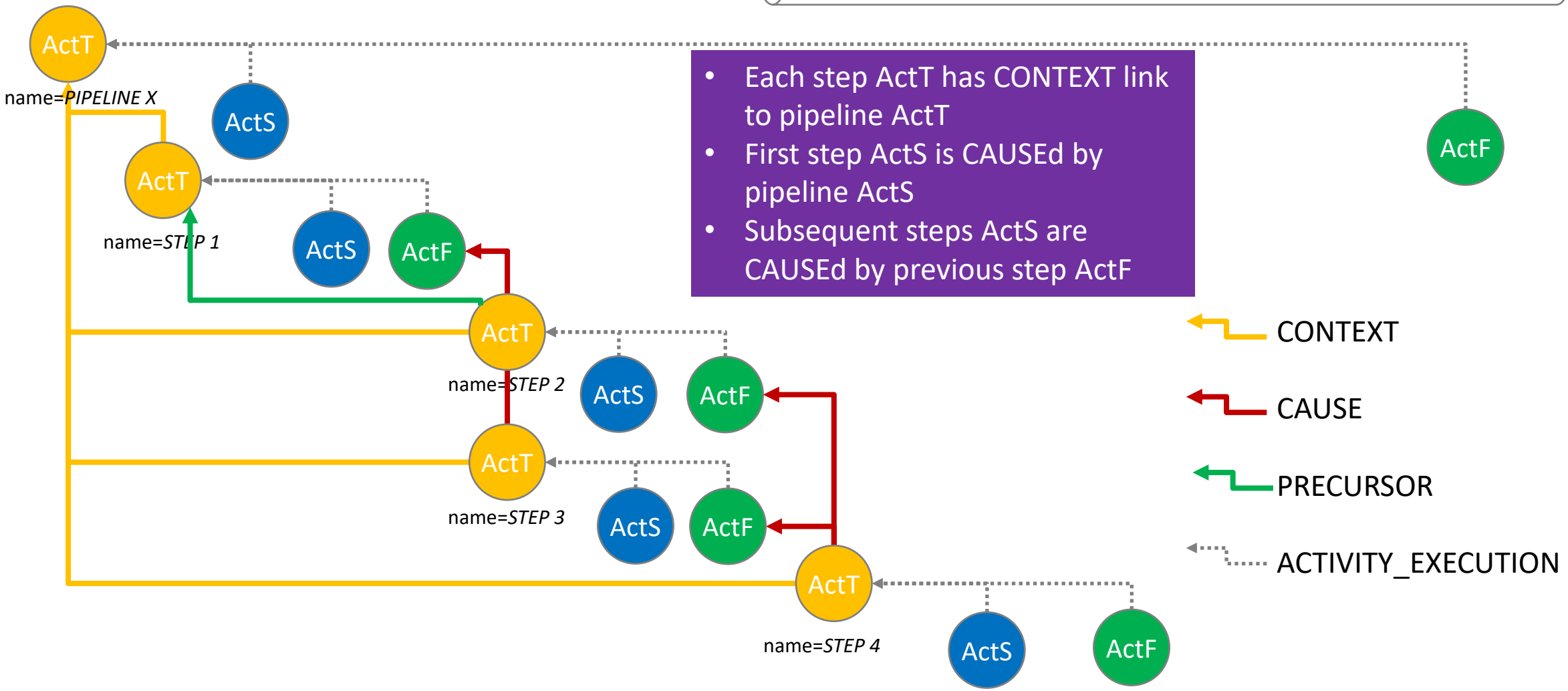
Example: All steps are event triggered



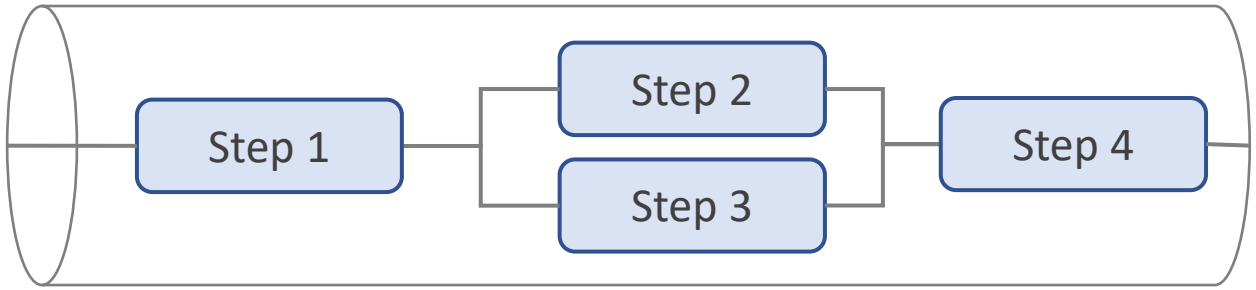


Example: All steps are event triggered

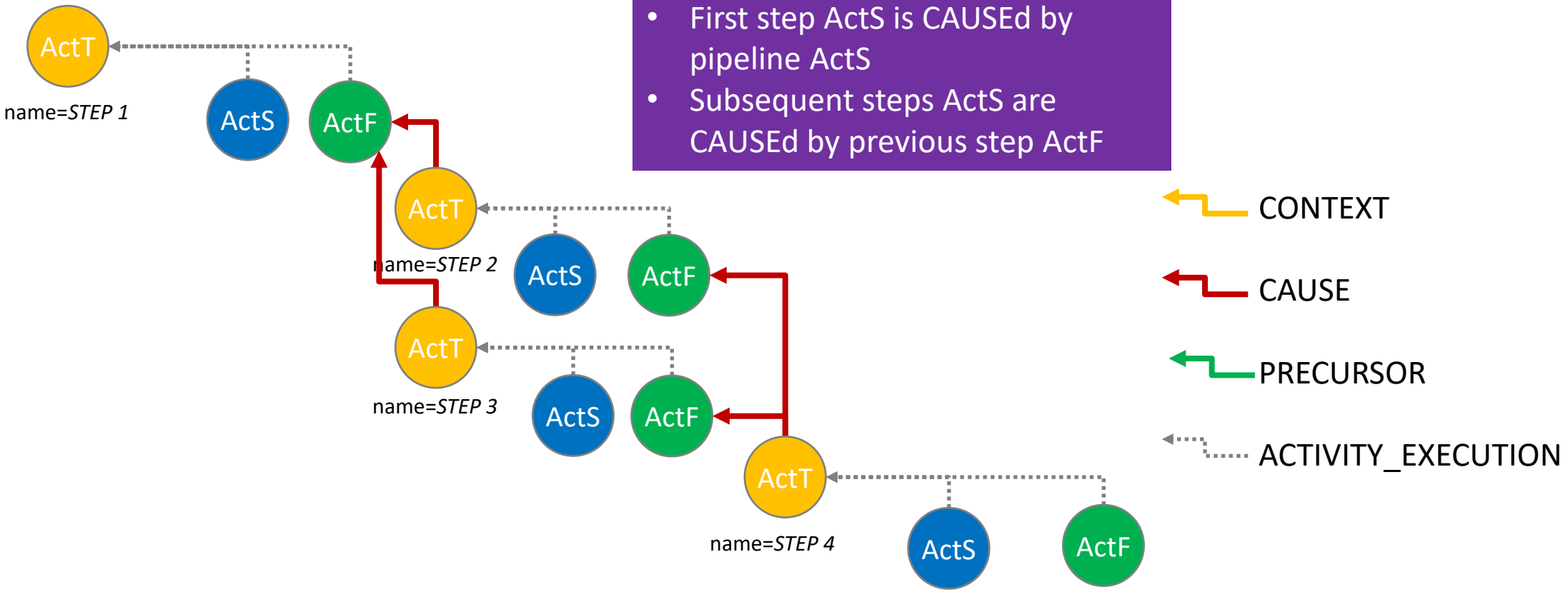




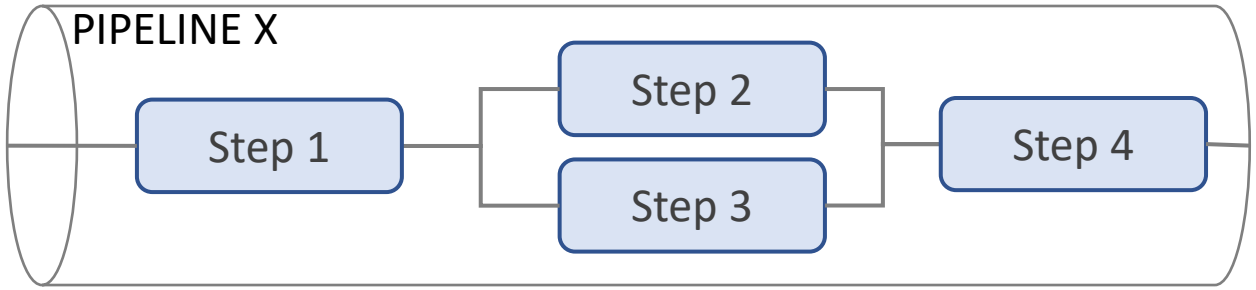
Example: All steps are event triggered

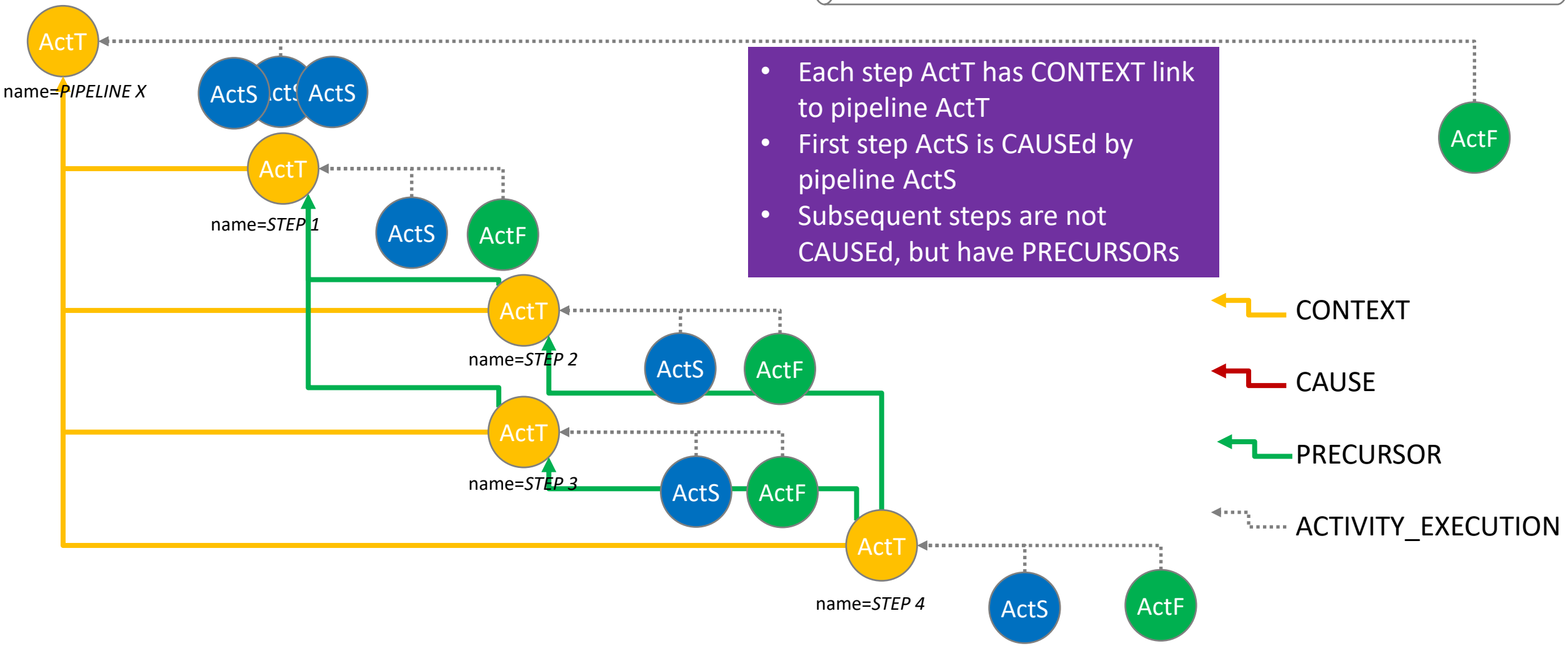


- Each step ActT has CONTEXT link to pipeline ActT
- First step ActS is CAUSEd by pipeline ActS
- Subsequent steps ActS are CAUSEd by previous step ActF

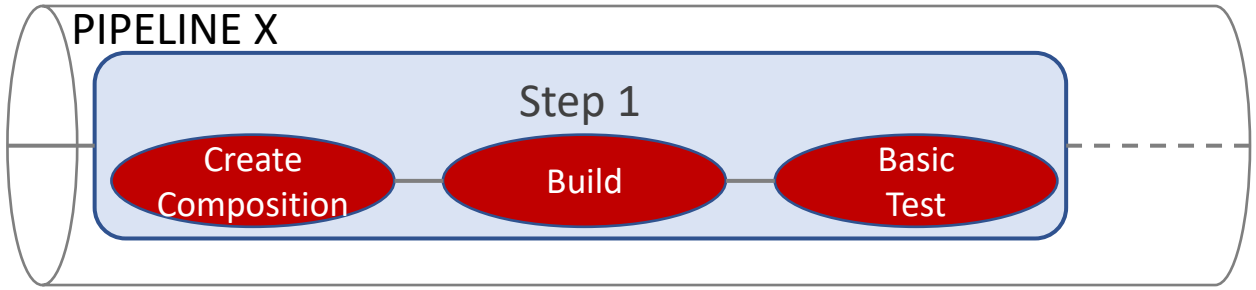


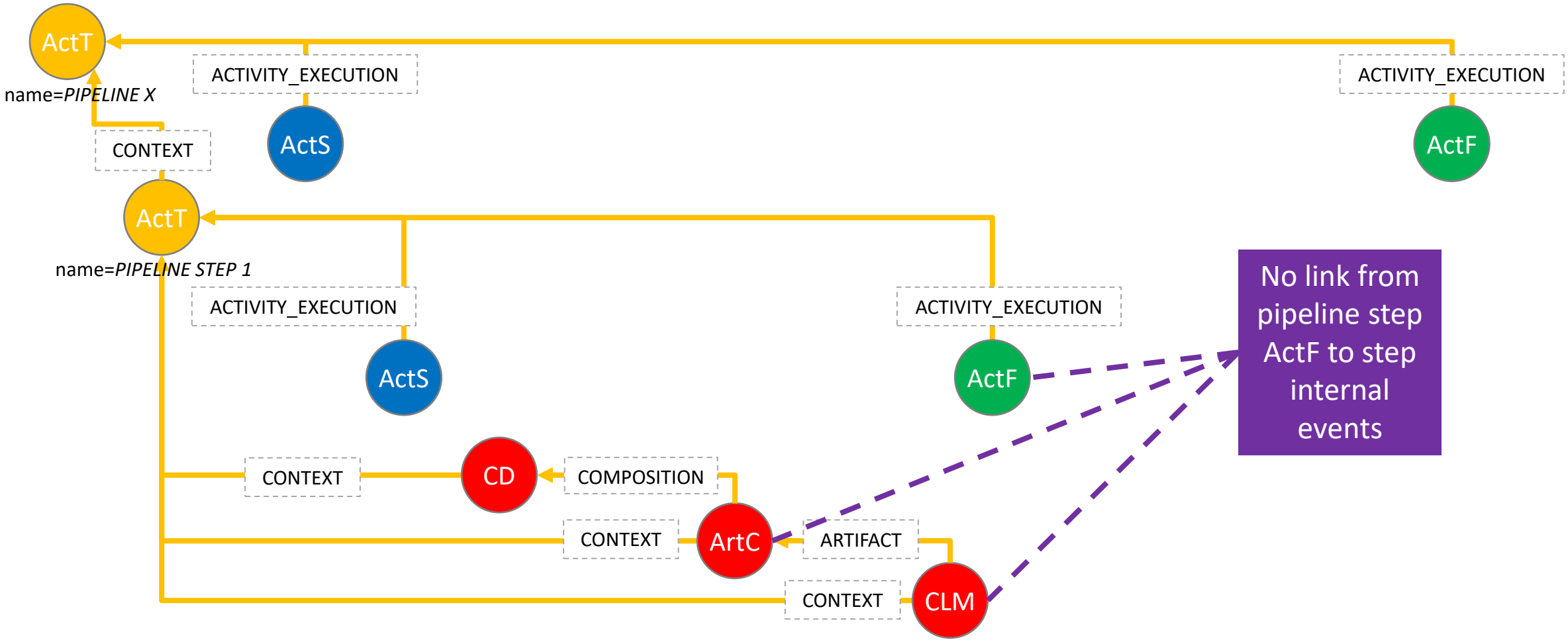
Example: All steps are explicitly triggered





Events in Activity (in pipeline step)

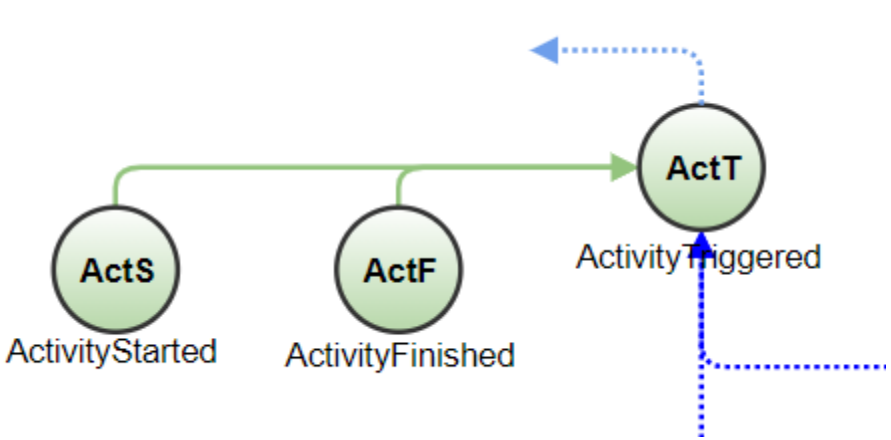


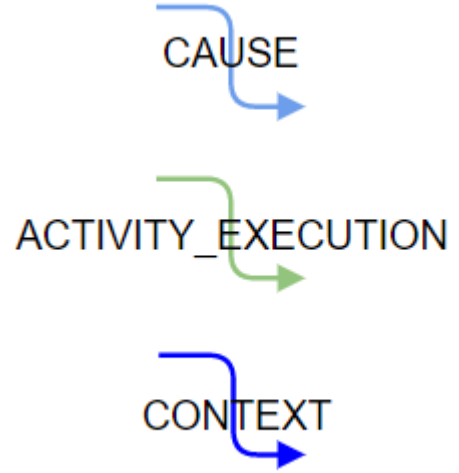


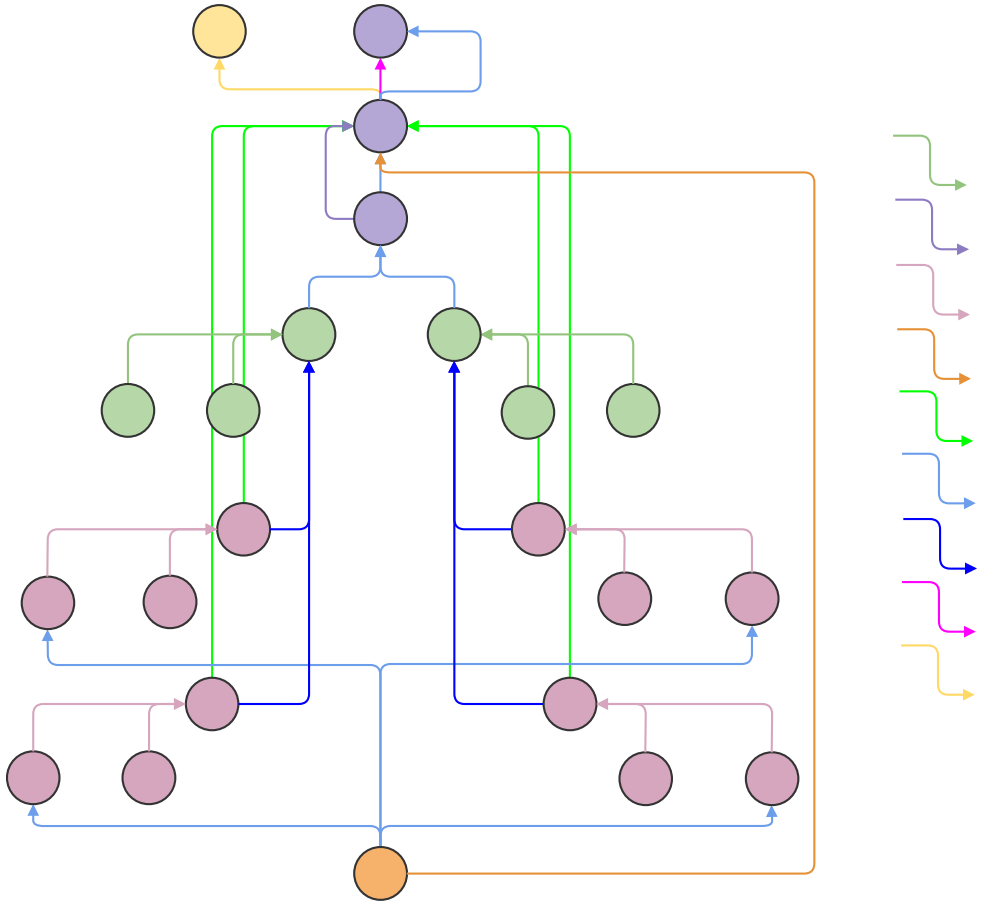
Current Examples

- Confidence Level Joining
  - Shows that TCT has a CONTEXT link to ActT
  - Shows that ActT is CAUSEd by and ArtP event
  - Shows that ActS and ActF has ACTIVITY_EXECUTION links to ActT
- Pipeline Monitoring
  - Shows that ArtC has a CONTEXT link to ActT
  - Shows that ActS and ActF has ACTIVITY_EXECUTION links to ActT
- Test Execution
  - Shows that TSS and TERCC has CONTEXT links to an ActT event
  - Shows that ActS and ActF has ACTIVITY_EXECUTION links to ActT

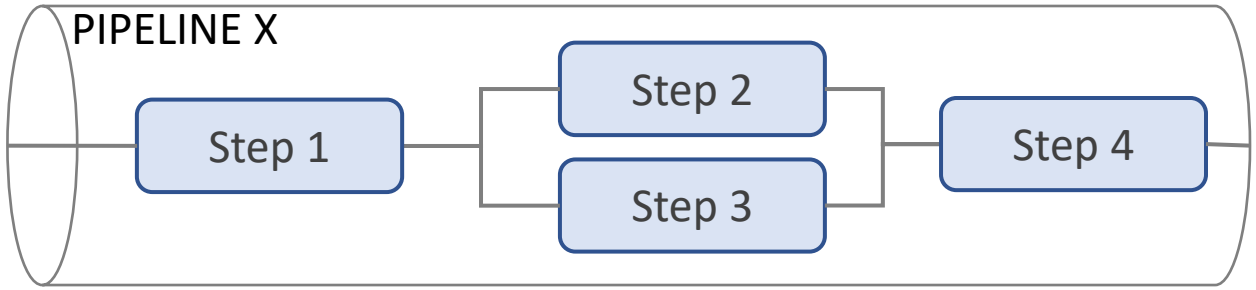
Example

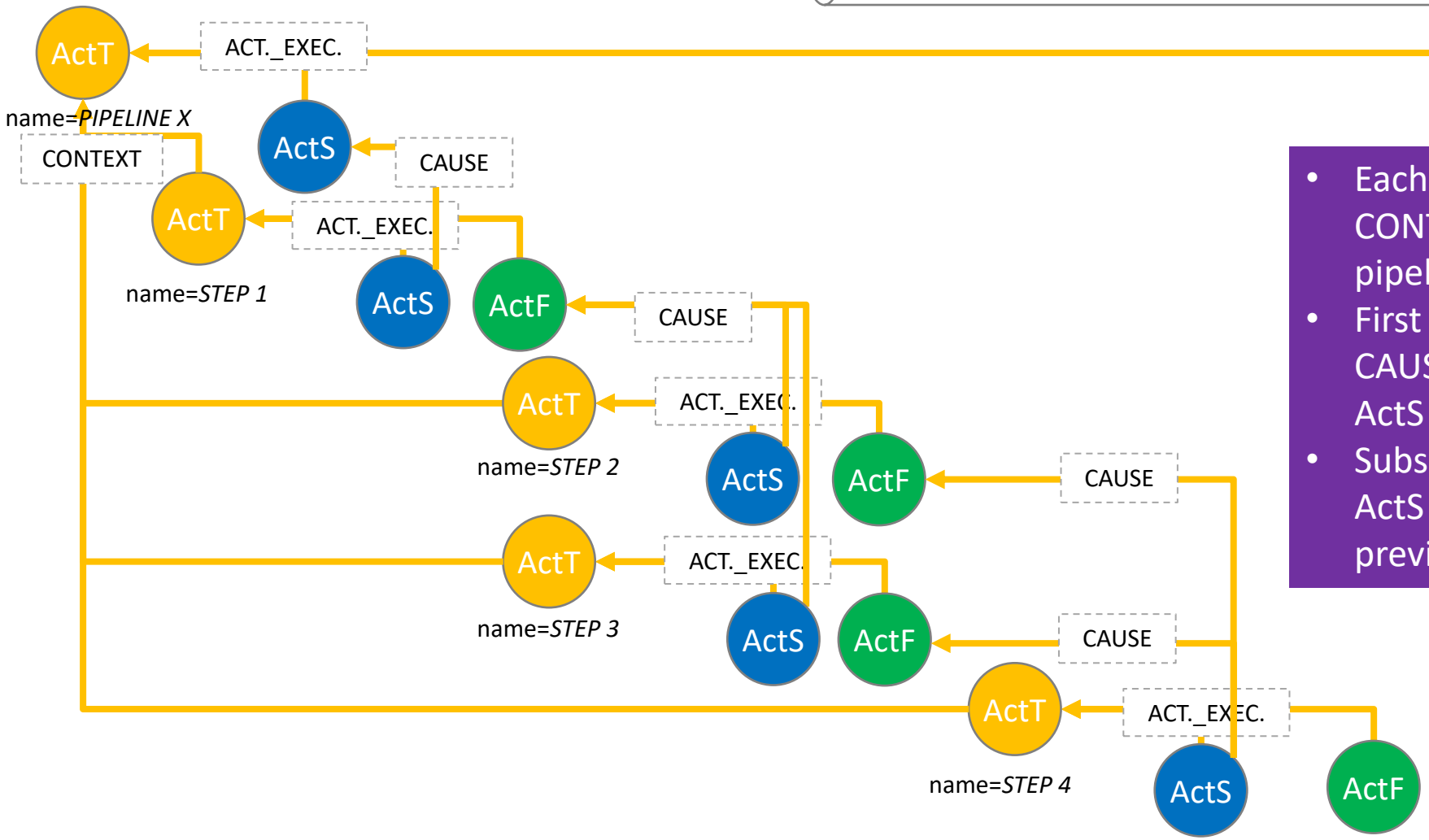

ActT
ActS
ActivityStarted
ActF
ActivityFinished


CAUSE
ACTIVITY_EXECUTION
CONTEXT



Pipeline Activities





- Each step ActT has CONTEXT link to pipeline ActT
- First step ActS is CAUSEd by pipeline ActS
- Subsequent steps ActS are CAUSED by previous step ActF

Events in Activity (in pipeline step)