Coupling in Large Interactive Applications

Jean-Denis Lesage

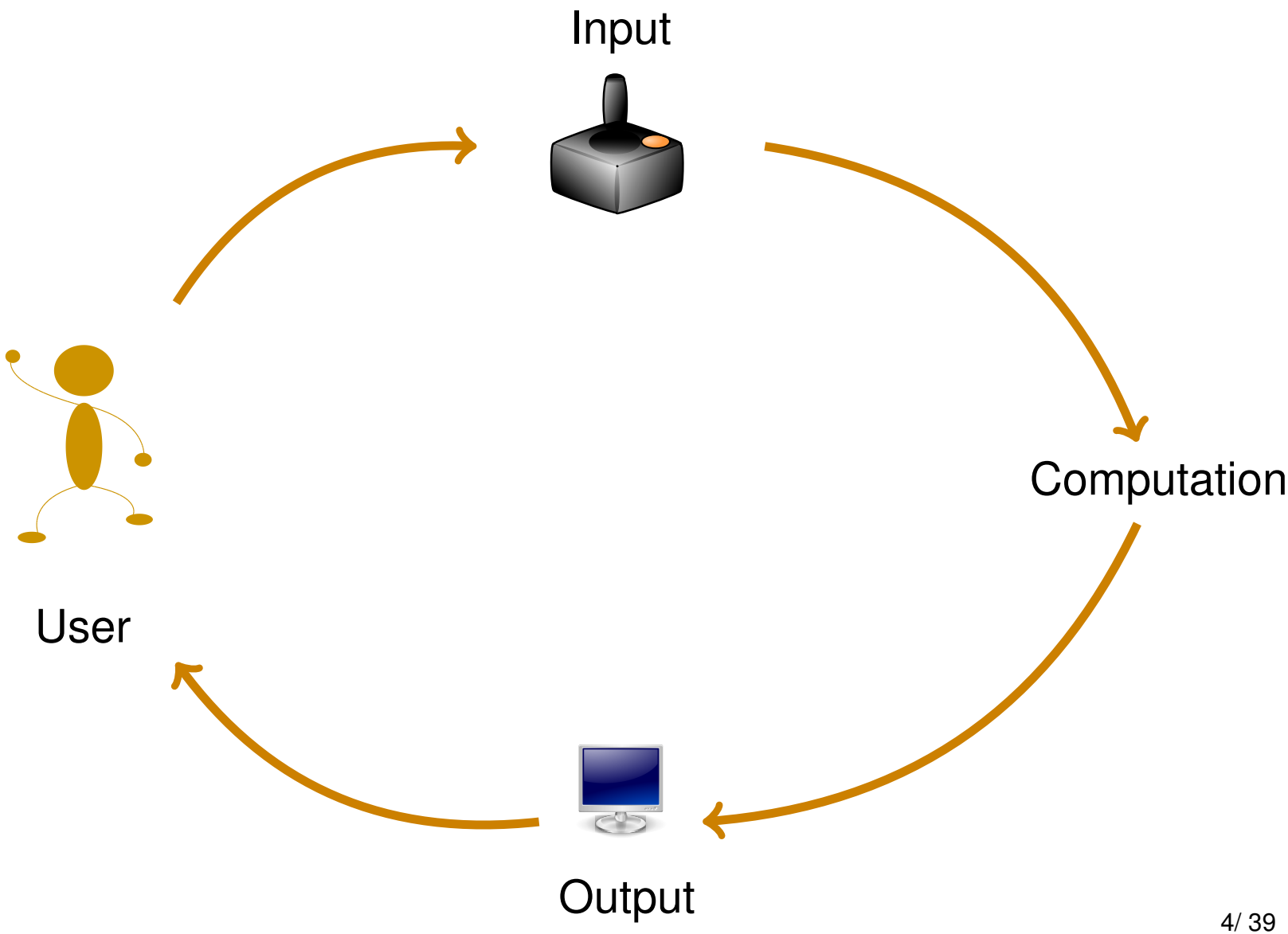
MOAIS

11/26/09

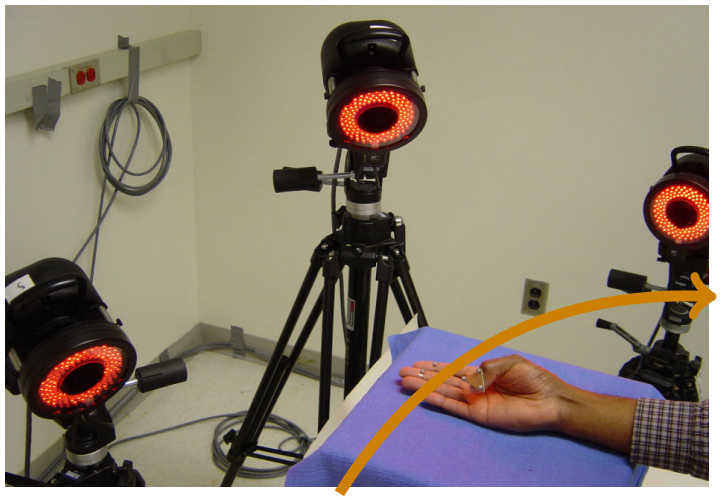
Outline

Outline

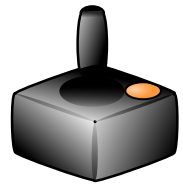
Interactive Applications

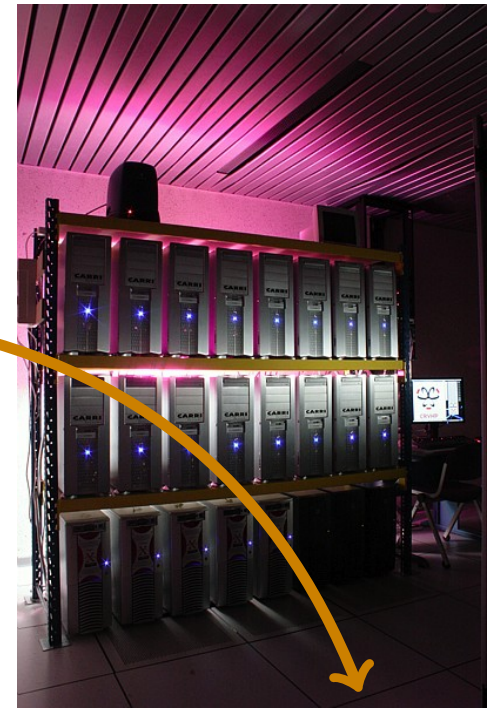


Large Interactive Applications

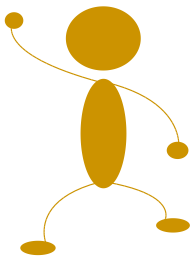


Input





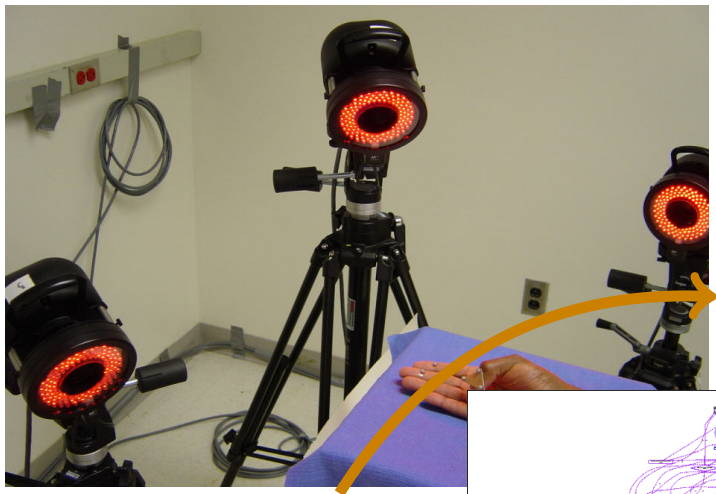
Computation



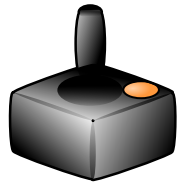
User

Output

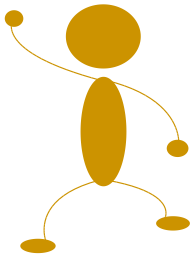
Large Interactive Applications



Input



Co tation



User

[?]

Output

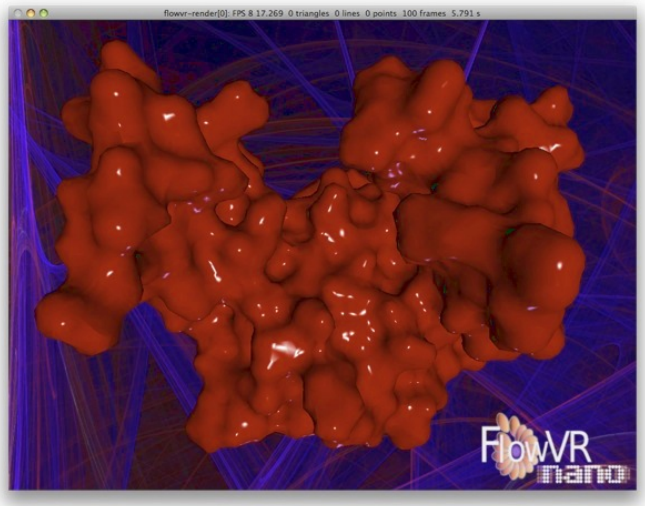
Examples of Large Interactive Applications

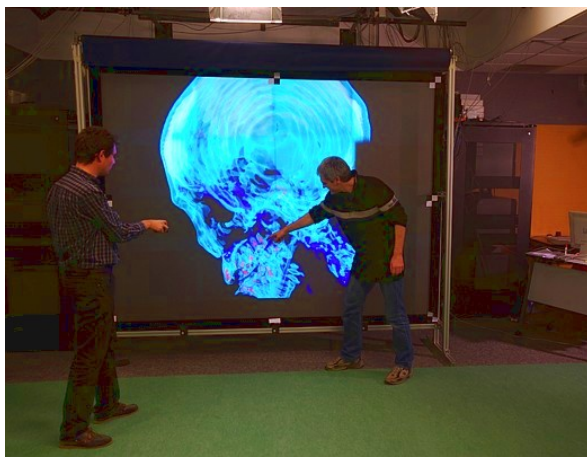
Metavers:
MMORPG, *Second Life*

Telepresence:
Interaction with distant user

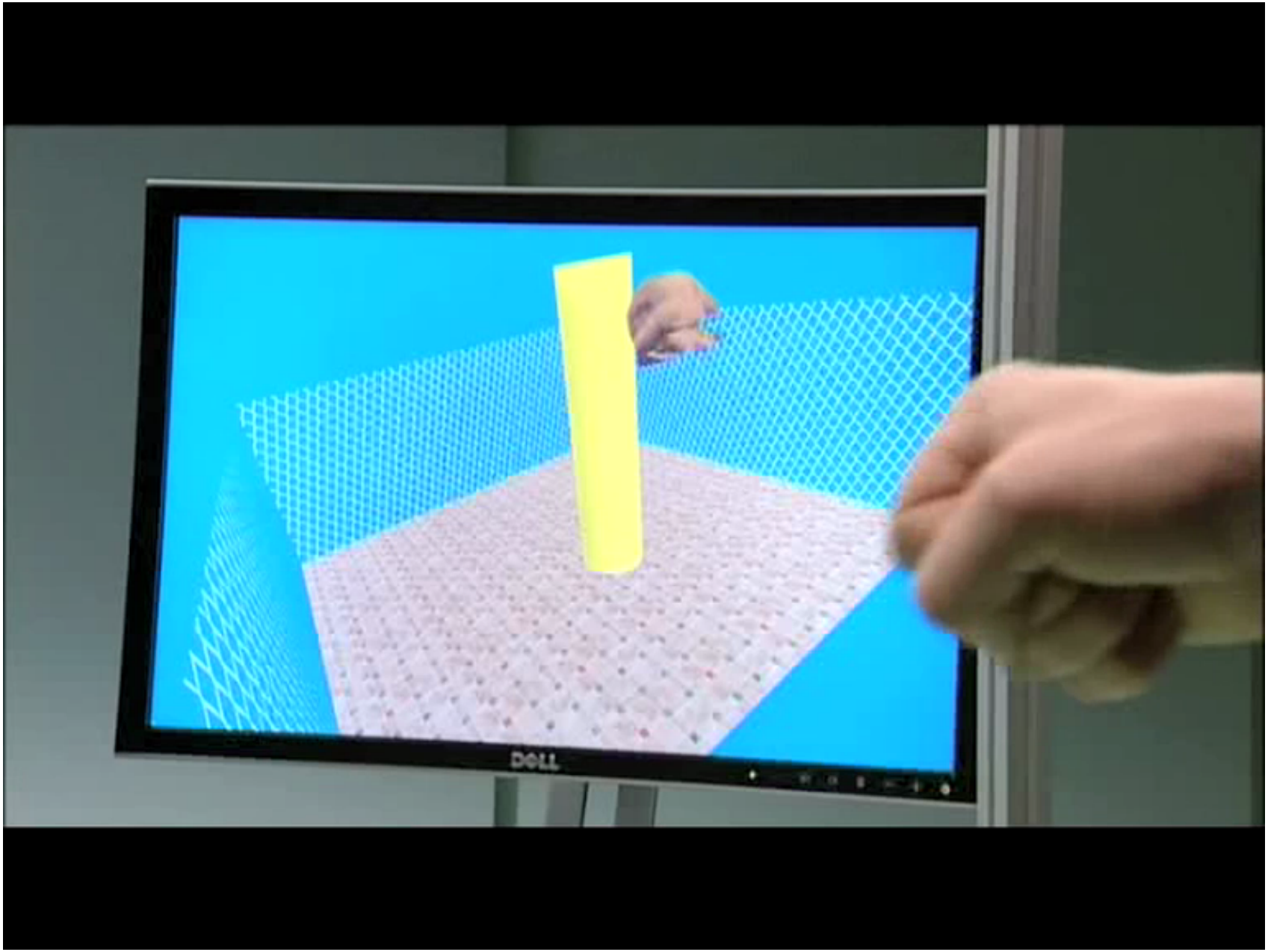
Computional Steering:
Interaction with complex simulation. Molecular simulation (300 K atoms)

Simulations:
Hercules earthquake simulation (5Hz, 12 billions elements)





Grimage Example [?]



Summary: Large Interactive Applications

Applications are

- **iterative:** a task is an endless loop
- **large:** hundreds or thousands of tasks
- **multi-frequency:** asynchronism
- **samplers:** can discard or duplicate data

Use Specific Hardwares

- specific I/O devices
- CPU, GPU, clusters, grids

Must Perform under Strong Performance Constraints

refresh rate (haptic: kHz), latency ($\leq$ 30 ms)

Problems Statements

Coupling in Large Interactive Applications

How to couple iterative tasks in a large dataflow graph?

Problems Statements

Coupling in Large Interactive Applications

How to couple iterative tasks in a large dataflow graph?

Questions:

- How to handle complexity?

Problems Statements

Coupling in Large Interactive Applications

How to couple iterative tasks in a large dataflow graph?

Questions:

- How to handle complexity?
- How to execute these applications under strong performance constraints ?

Problems Statements

Coupling in Large Interactive Applications

How to couple iterative tasks in a large dataflow graph?

Questions:

- How to handle complexity?
- How to execute these applications under strong performance constraints ?

Contributions

- Hierarchical components model
- Study of the impact of samplers on the global latency

Outline

Middlewares for Interactive Applications

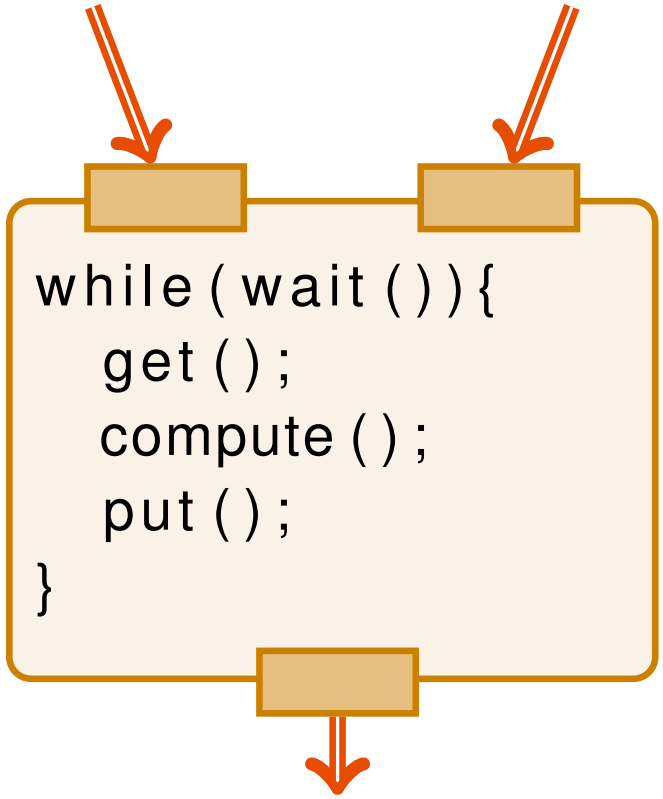
Middlewares

Scientific Visualization: SCIRun, IRISExplorer, VTK, ...

Virtual Reality: FlowVR, OpenMASK, Vista, ...

Architecture Description Language (ADL)

InTML, script languages, XML-based languages, ...



Goal: Modularity

- Multiple developers
- Long projects
- Maintainability
- Performance constraints

Modularity: Components Models

Component:

- Specified by interfaces. Re-usability.
- Iterative Task $\simeq$ Component

Hierarchical Component Model

Create new components by composing existing ones.
Example: Fractal norm. Implementations: Julia, ProActive, ...

Some Middlewares have a Component Support

SCIRun2 (CCA), VRJuggler (CORBA), ...
No hierarchy and a basic support of parallelism.

Modularity: Skeleton Programming

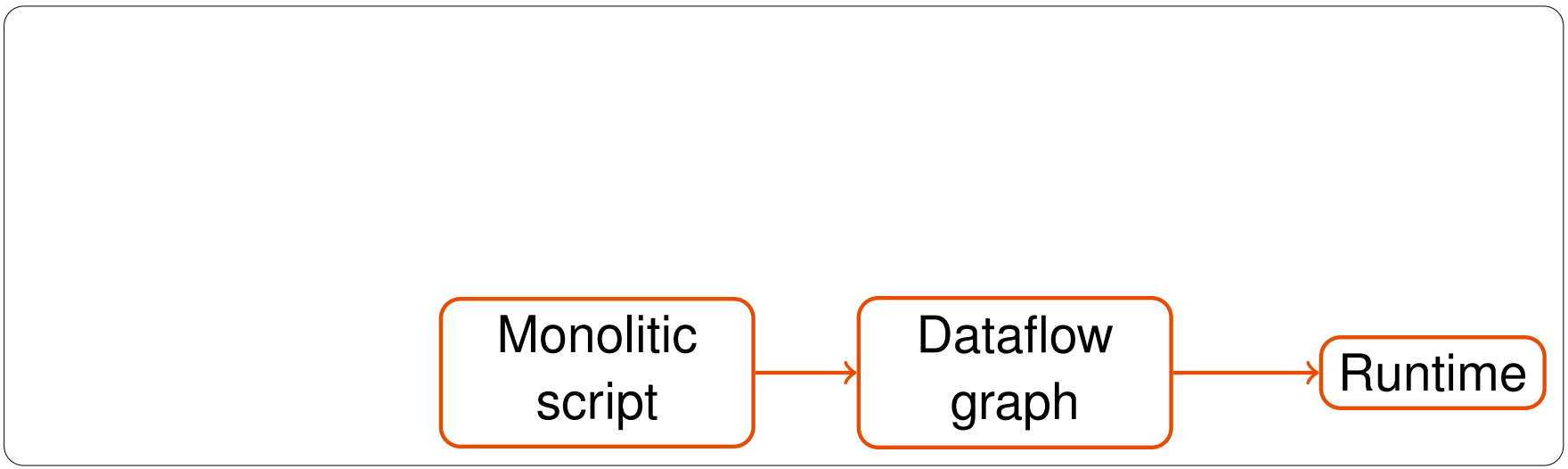
Main Idea

Provide efficient implementations of algorithms

- A software becomes a composition of skeletons.
- They often rely on formal models.
- A skeleton can be adapted to a target architecture.
- Skeletons libraries can be dedicated to a specific aera (Skipper-D...).
- They can support hierarchy.

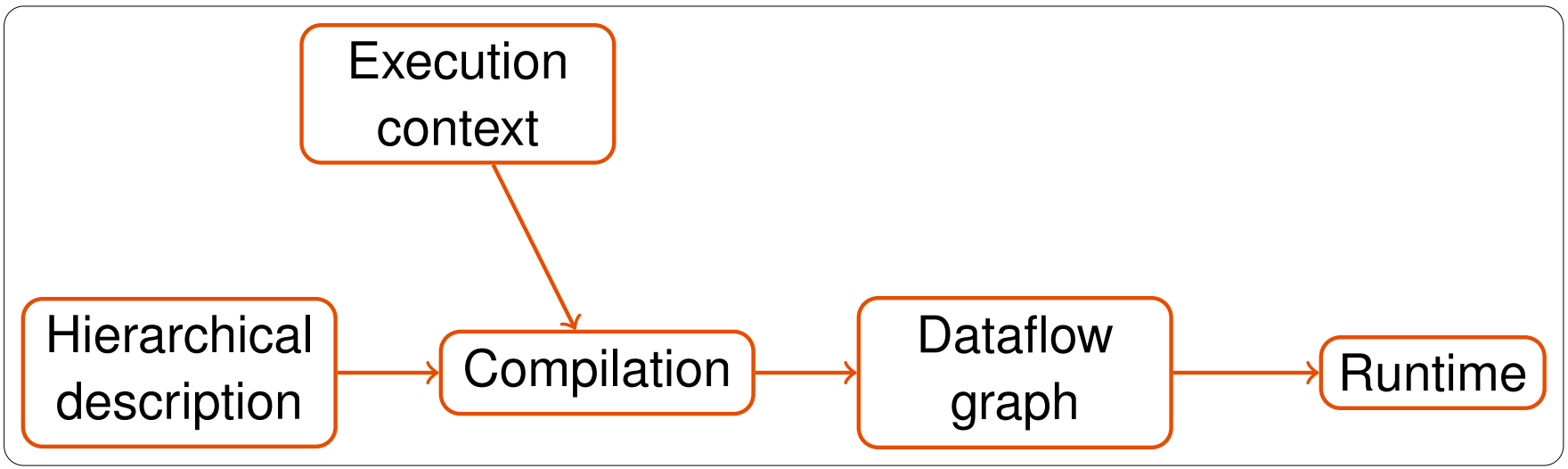
Our Contribution

Current application design process



Our Contribution

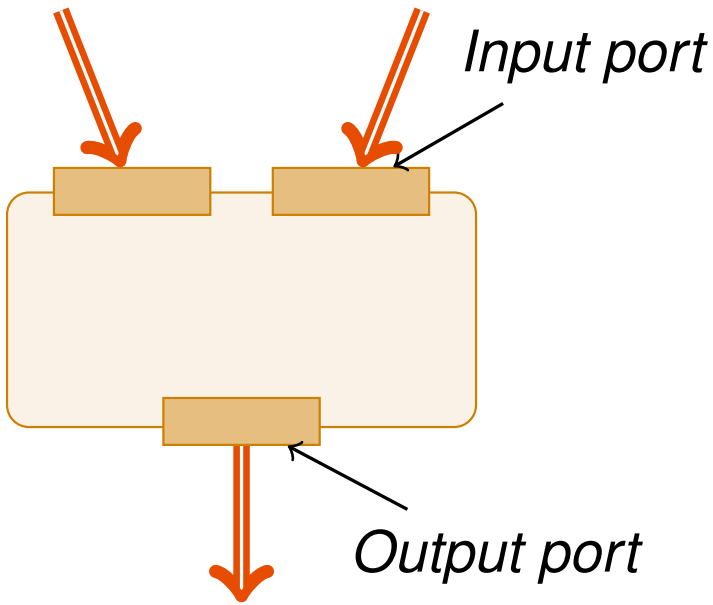
New application design process



Hierarchical Components Model

Interface:

Input and Output ports. Ports are FIFO queues.

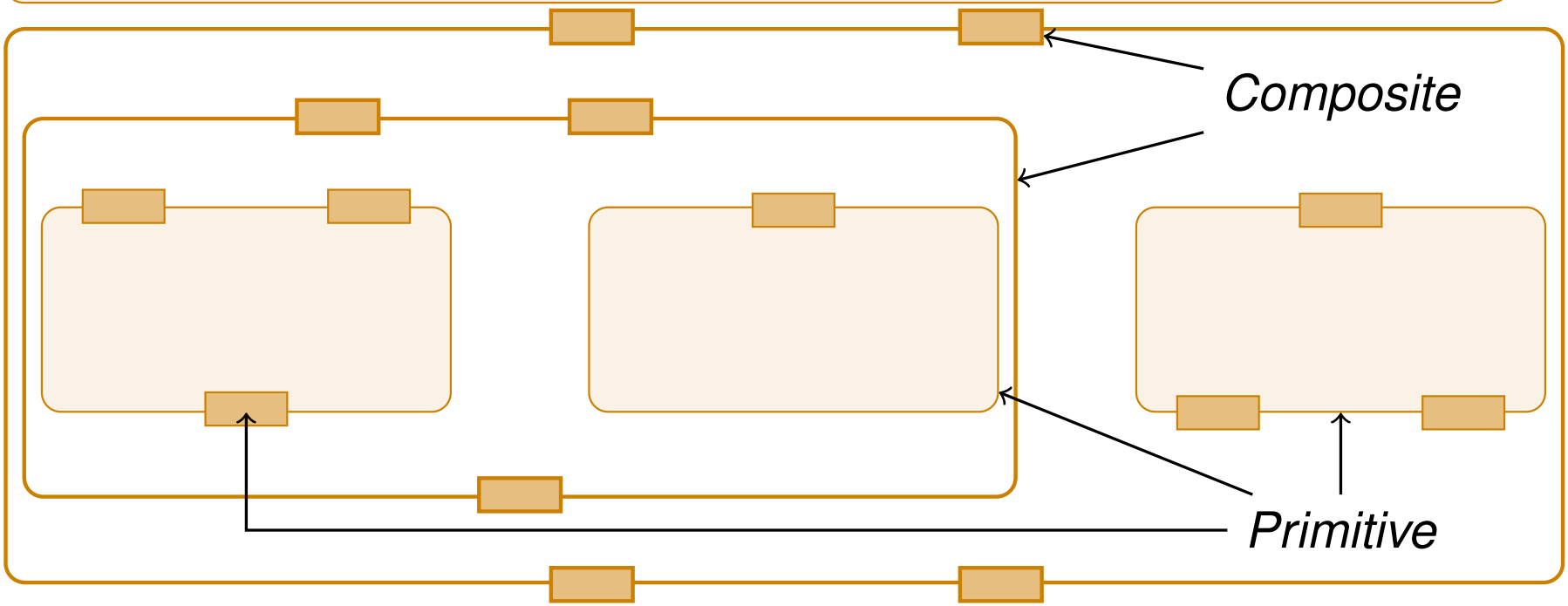


Hierarchical Components Model

Components:

Primitive Primitive components cannot contain any other component. Tasks, filters, connections are primitive components.

Composite Composite components contain other components (composite or primitive).

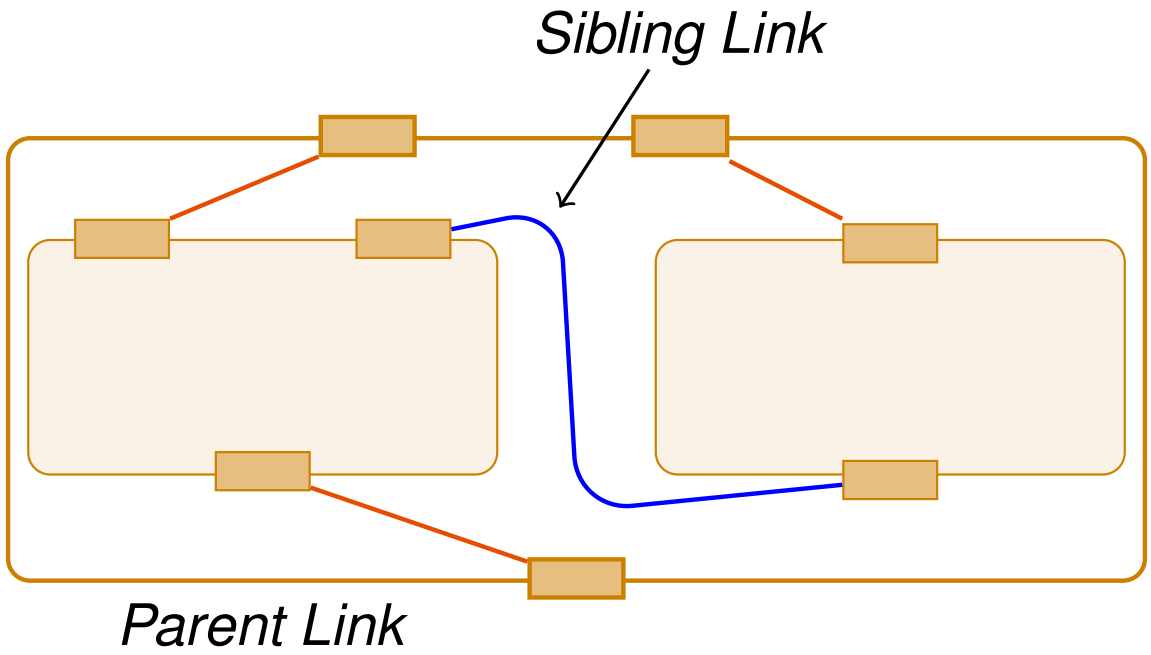


Hierarchical Components Model

Links:

Parent links the port from a composite component to one of its children's port.

Sibling links two ports of two components that have the same father.



Hierarchical Components Model

Controllers:

Introspection Controller reads component state.

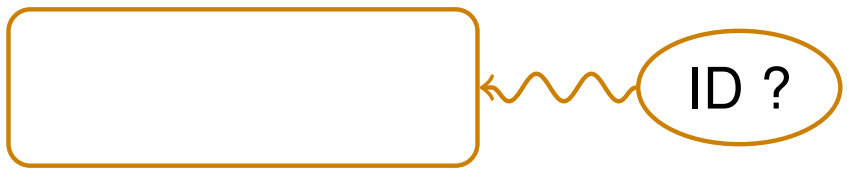
Configuration Controller changes component state.

Hierarchical Components Model

Controllers:

Introspection Controller reads component state.

Configuration Controller changes component state.

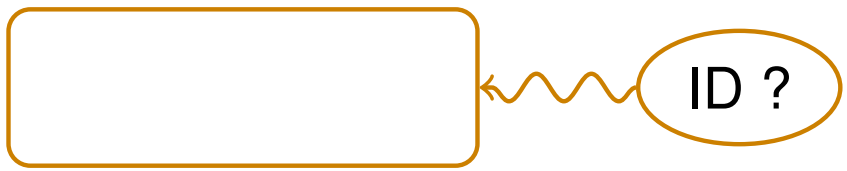


Hierarchical Components Model

Controllers:

Introspection Controller reads component state.

Configuration Controller changes component state.

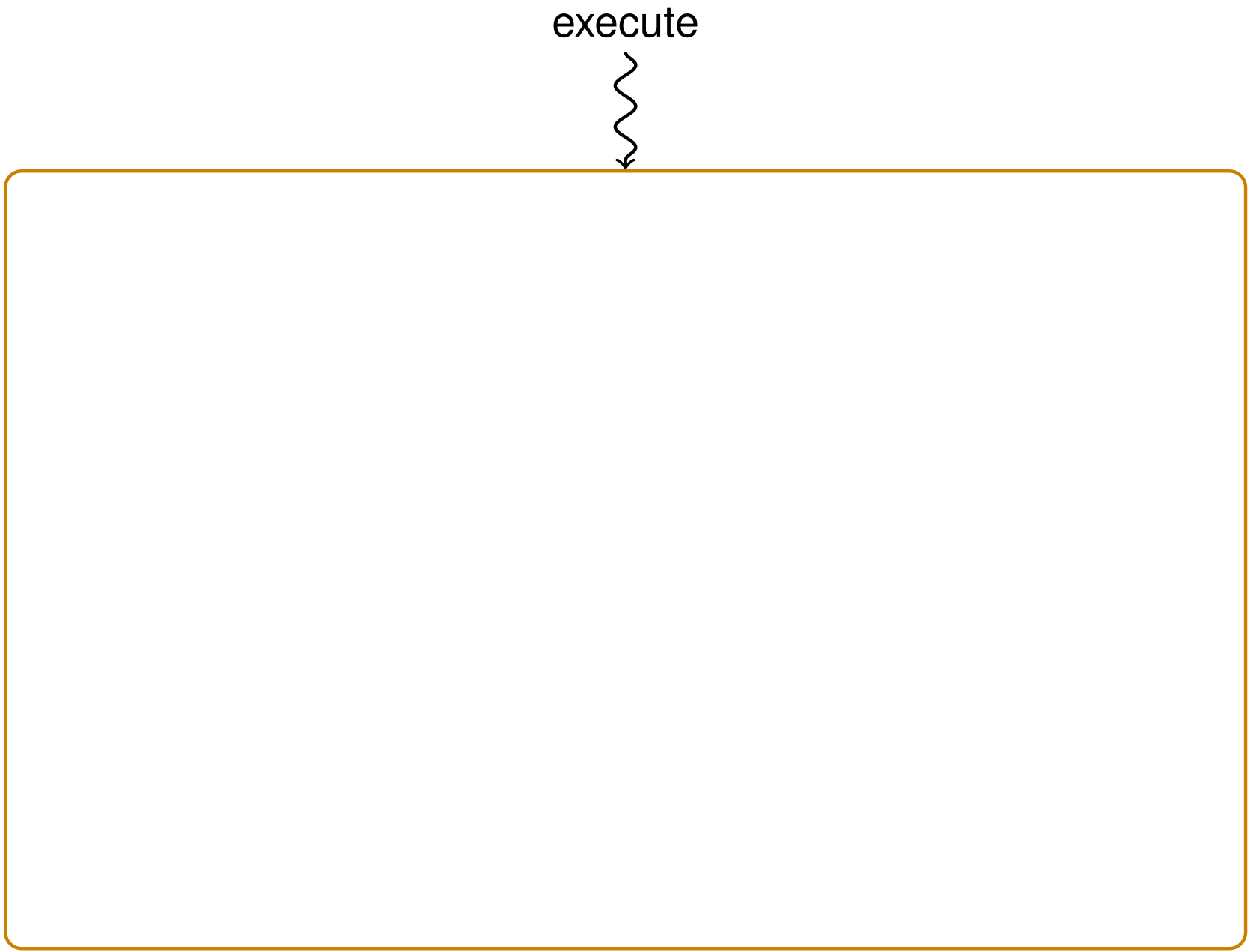


Examples:

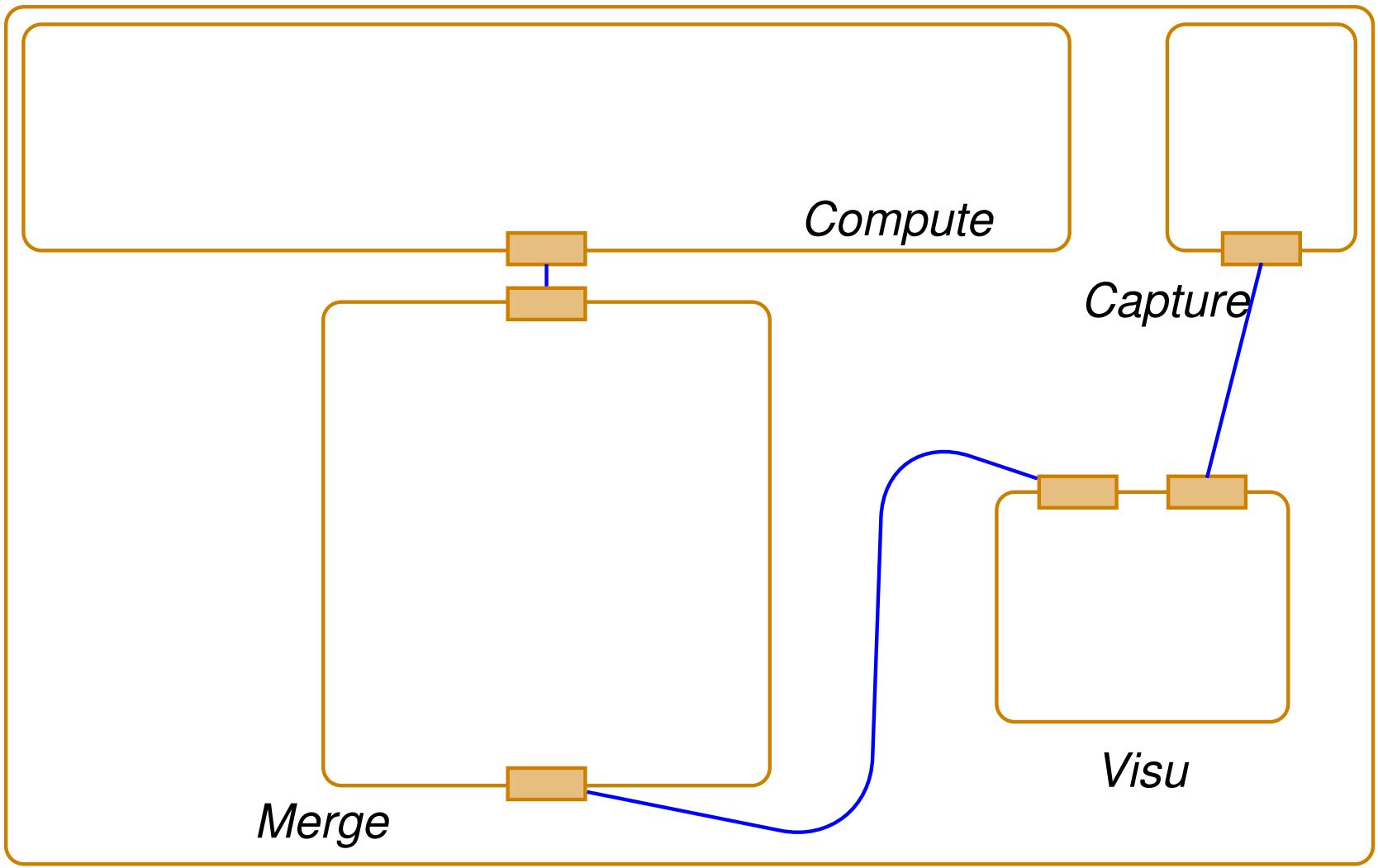
Introspection Controller Number of children, parameter file, architecture information. . .

Configuration Controller Addition of children, links or ports, mapping, deployment, . . .

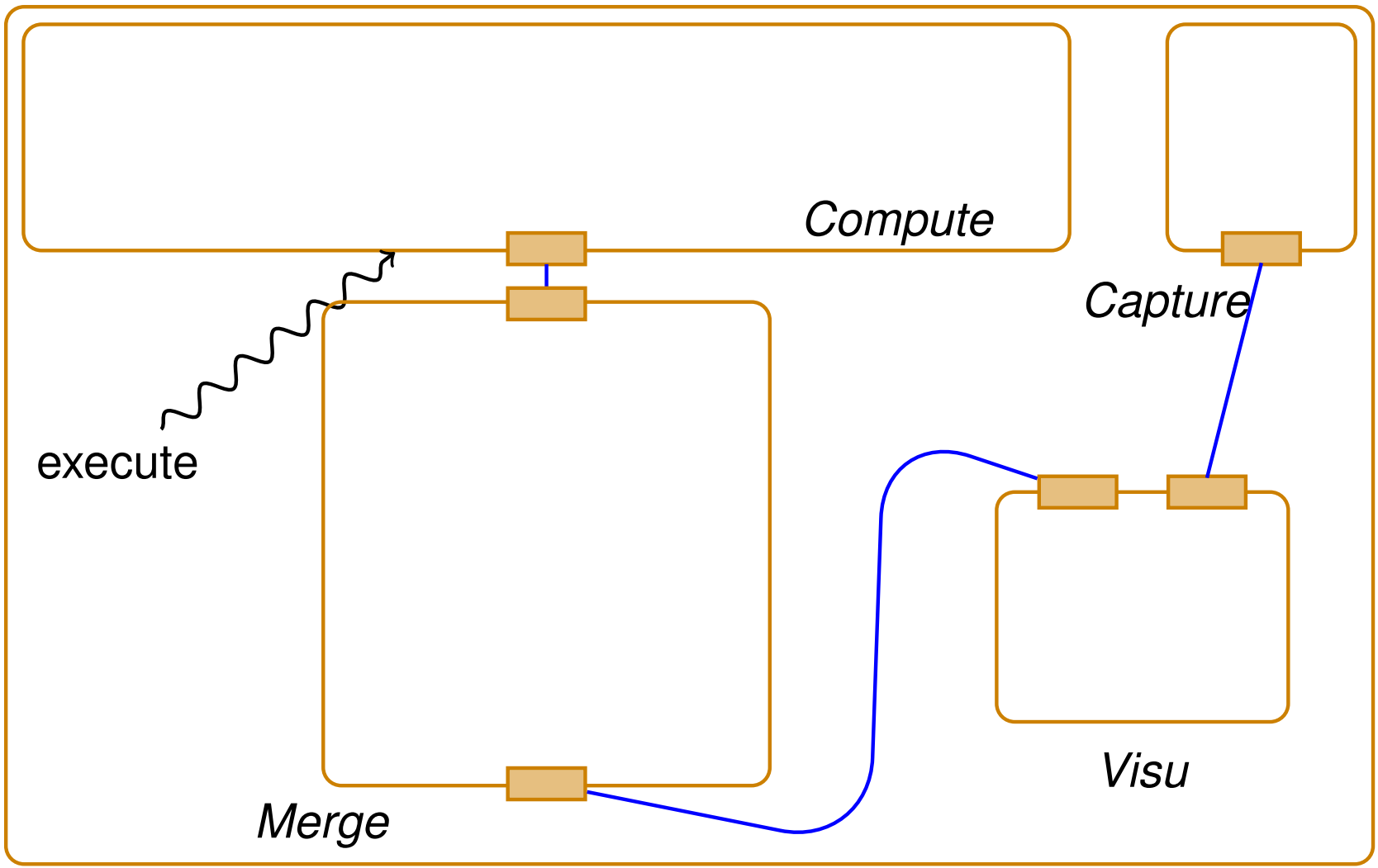
Example



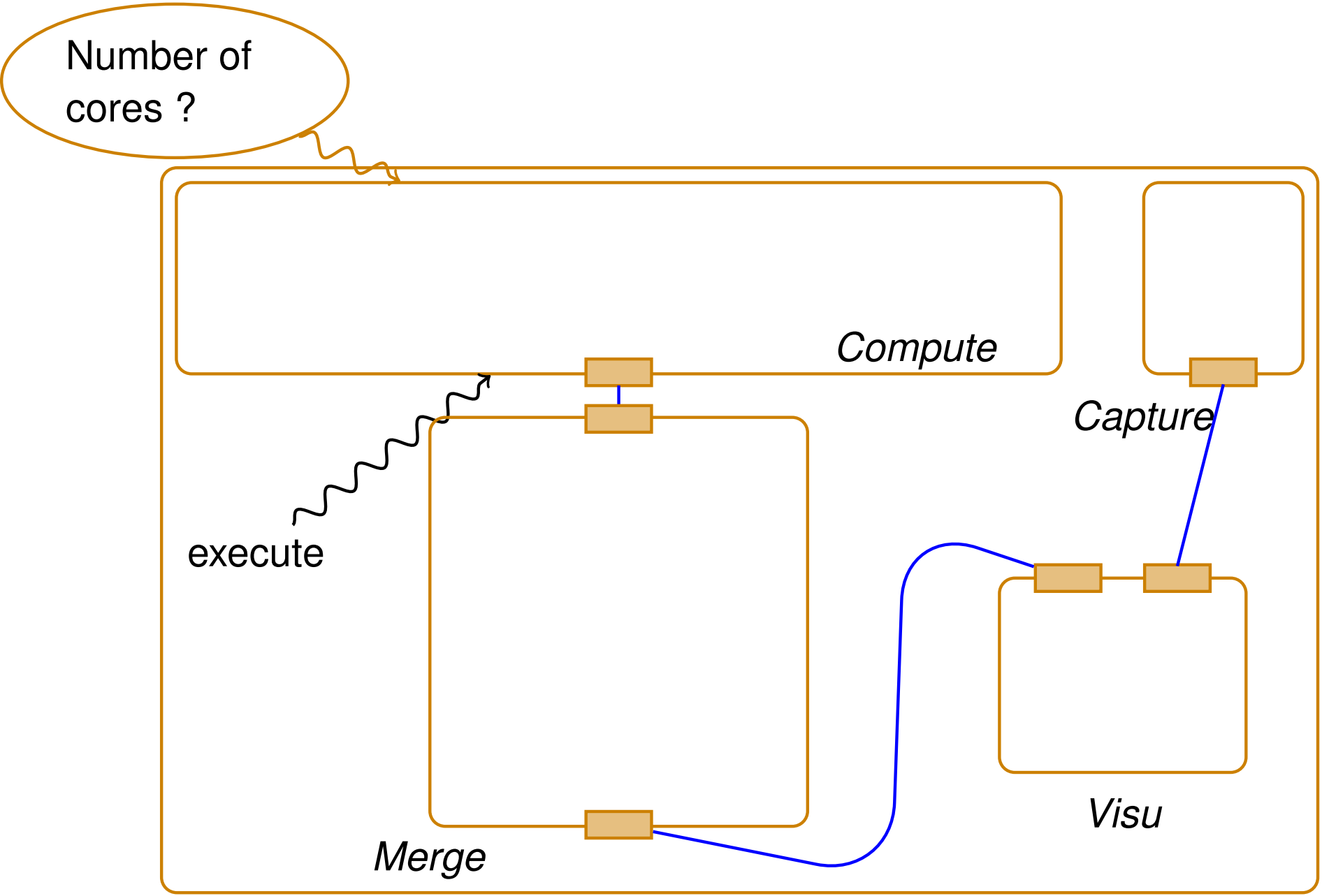
Example



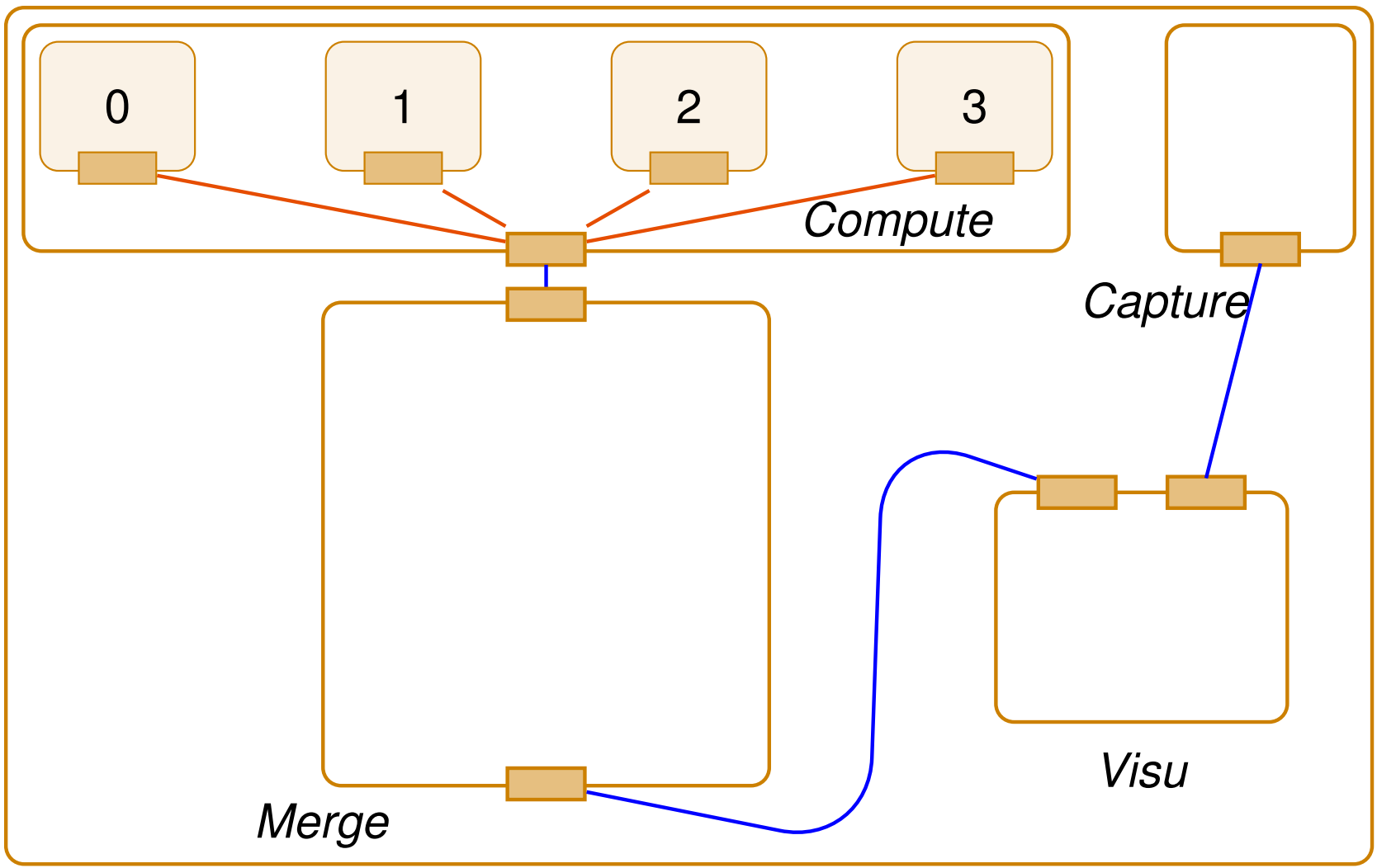
Example



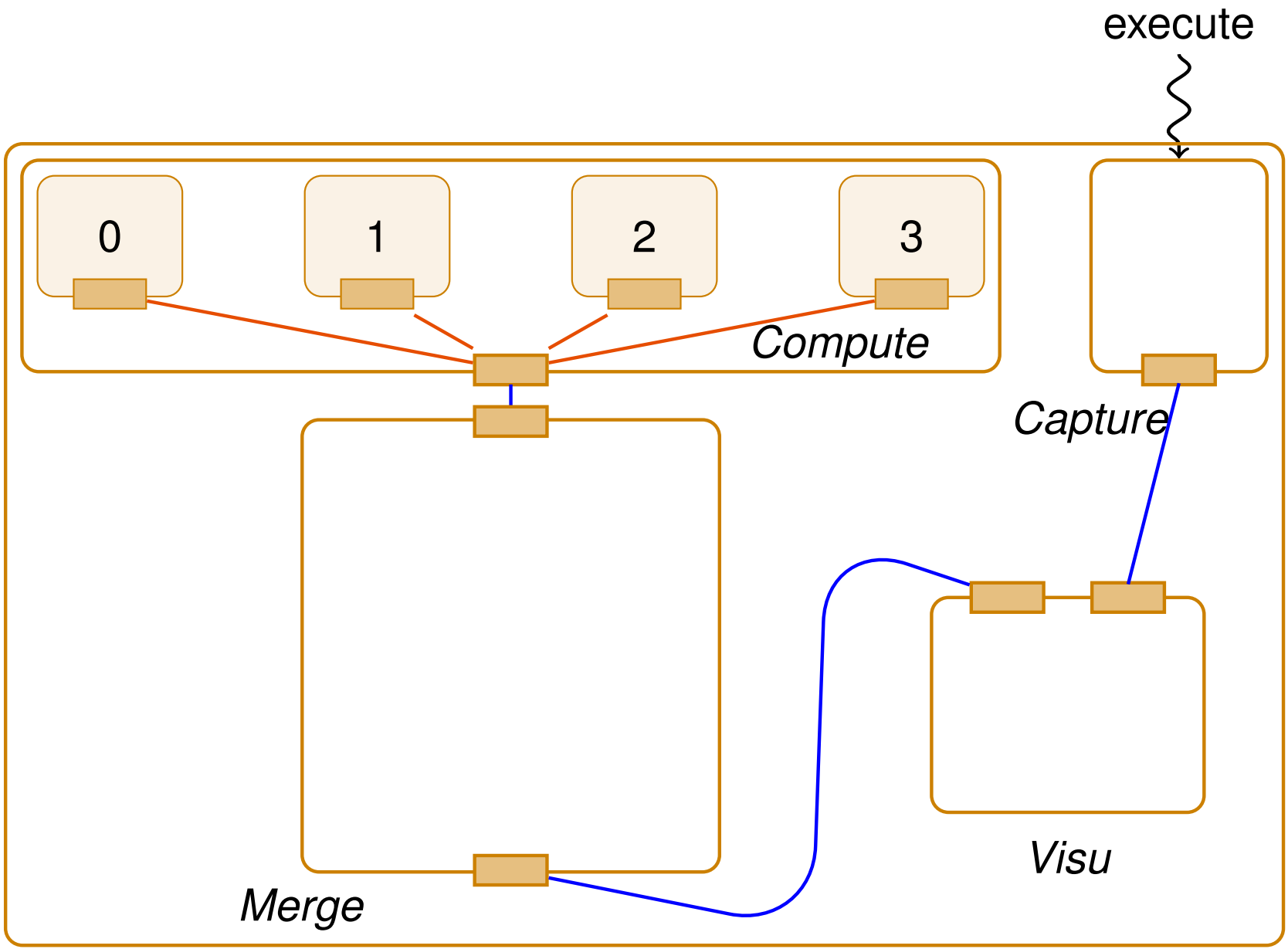
Example



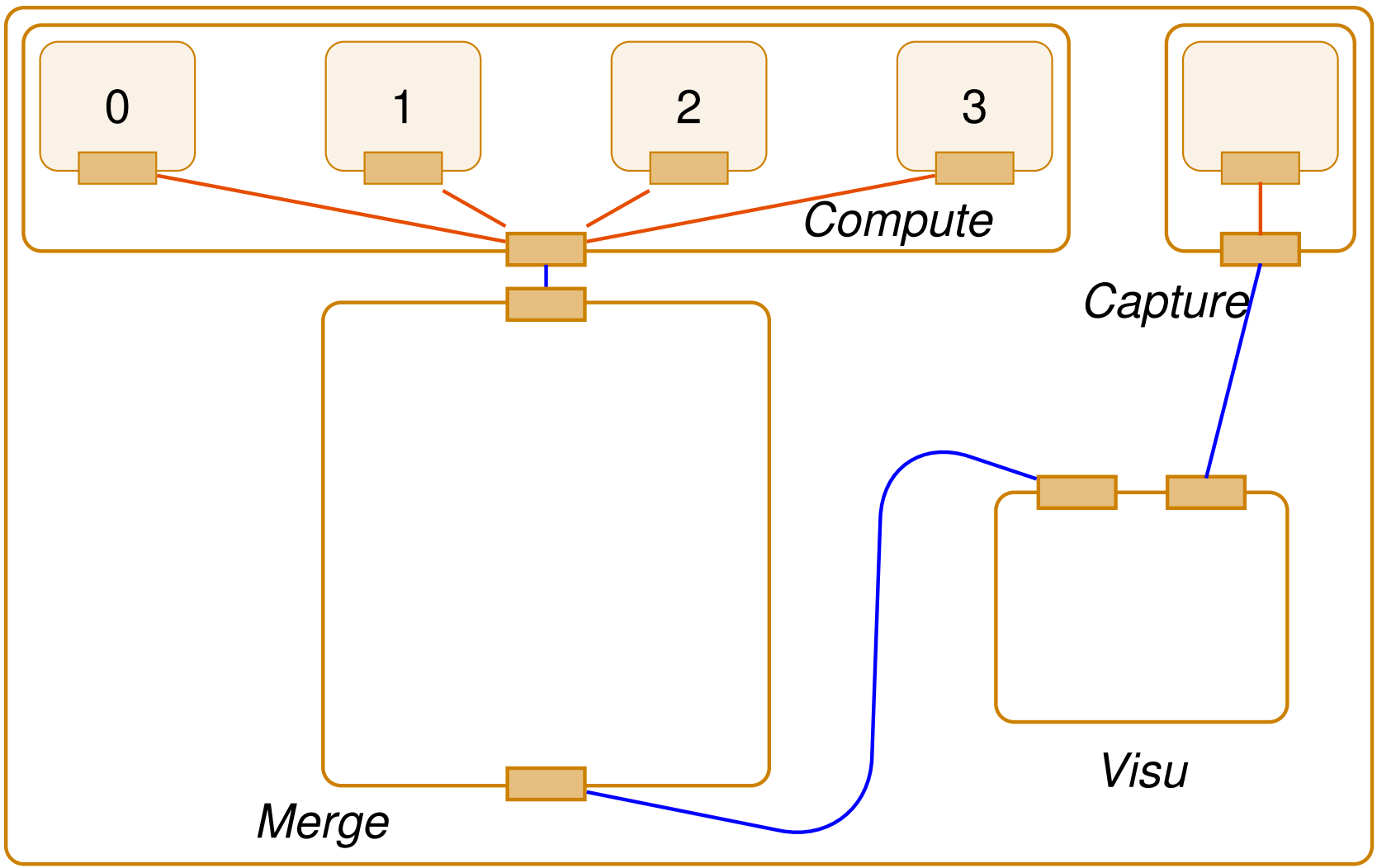
Example



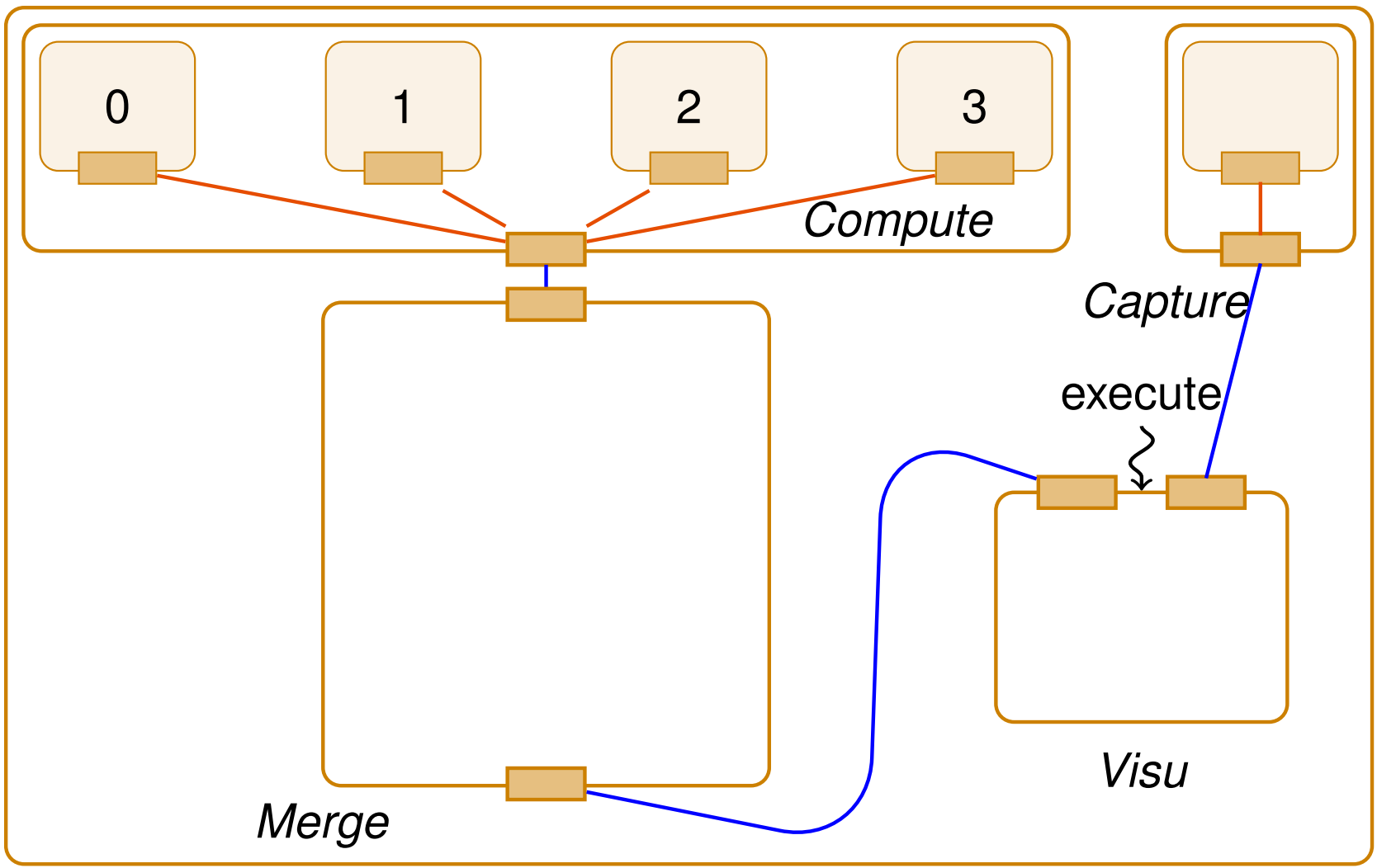
Example



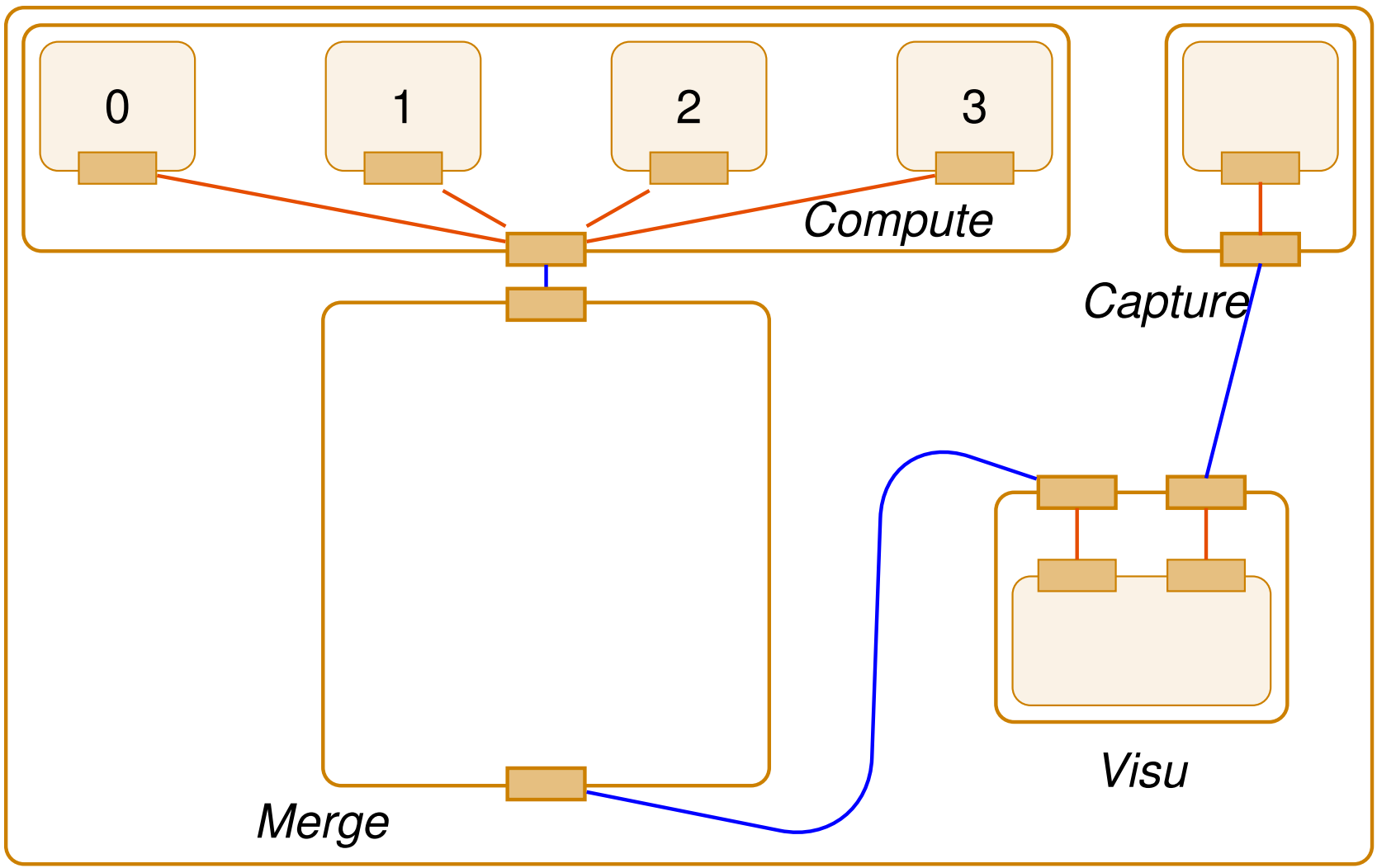
Example



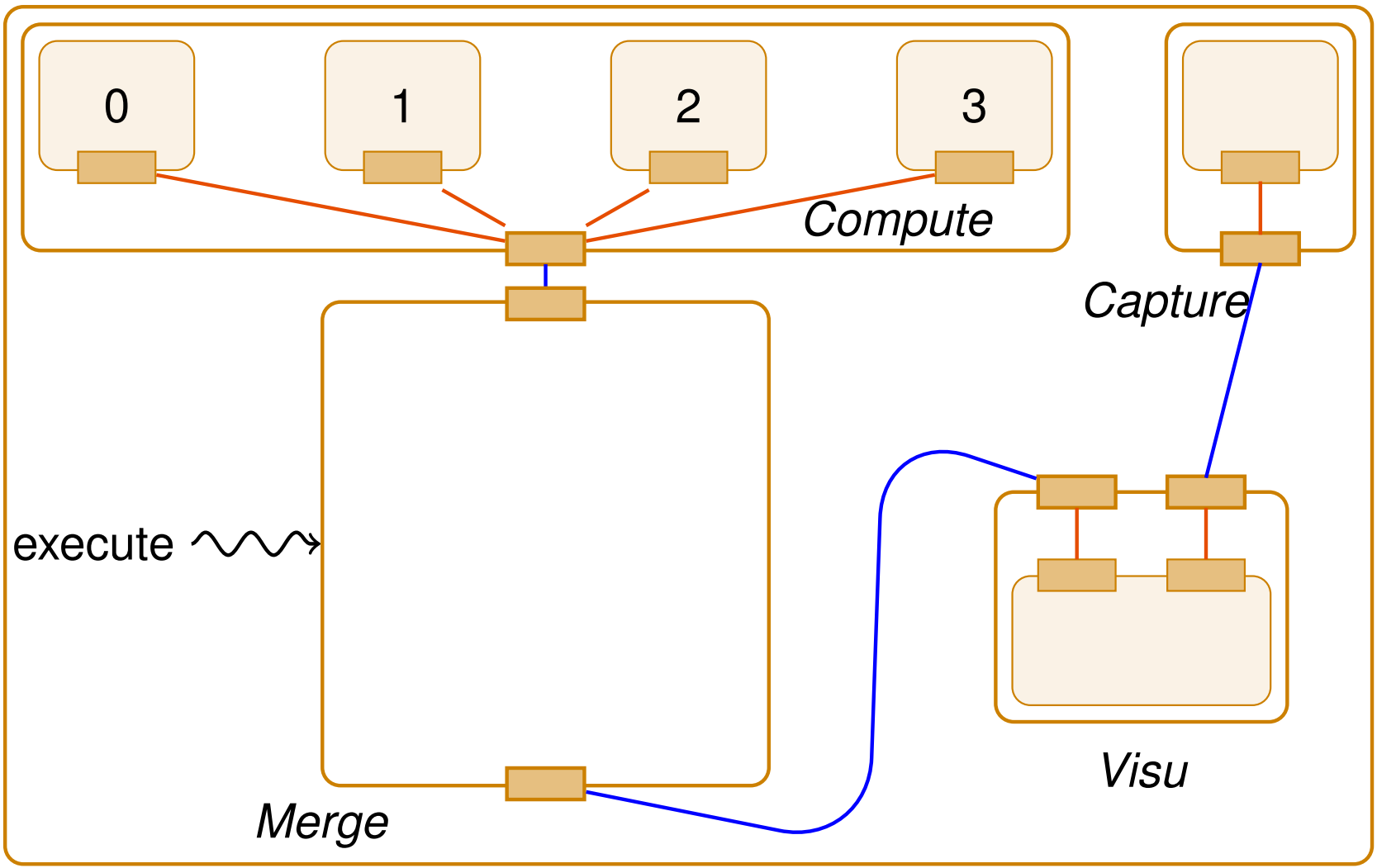
Example



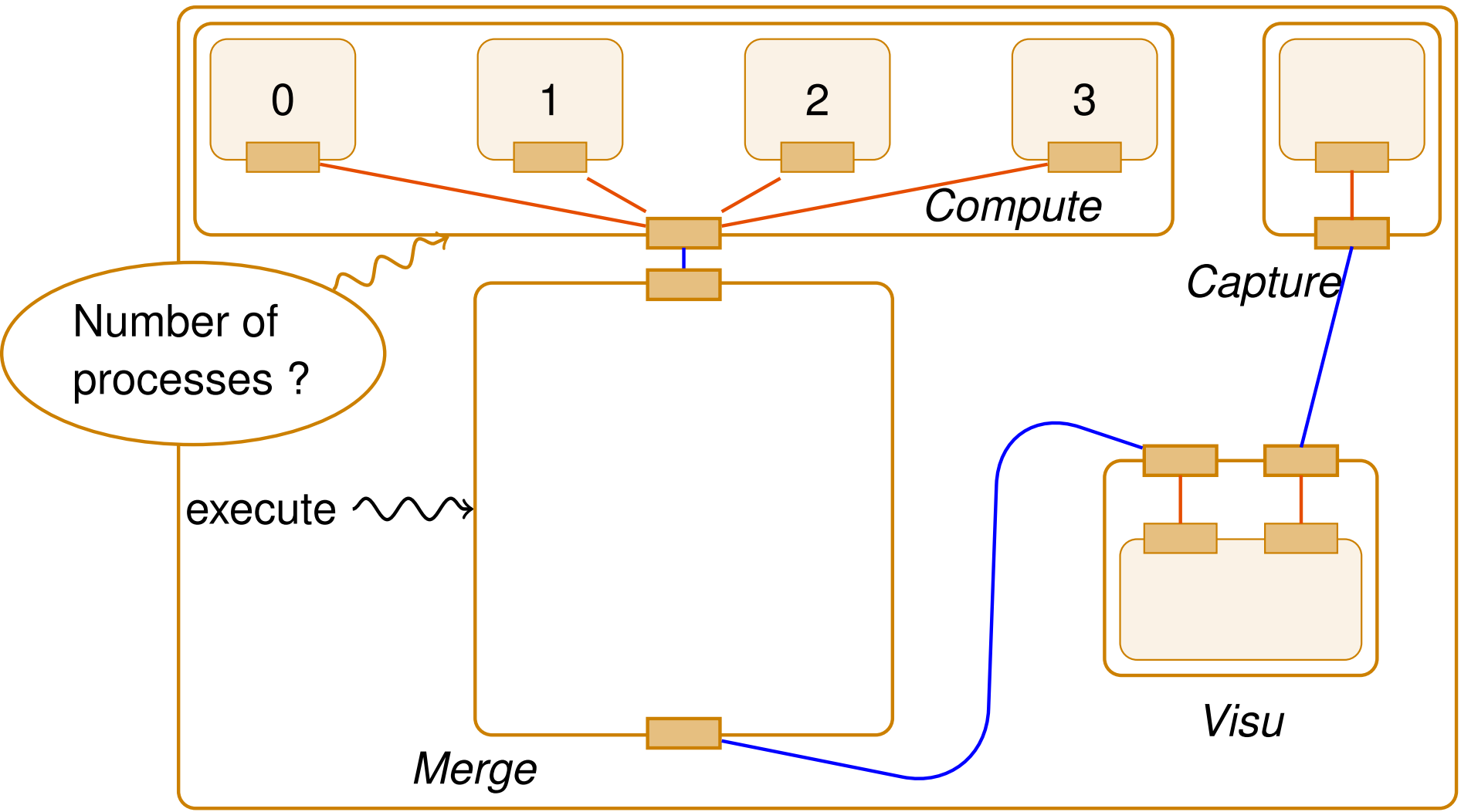
Example



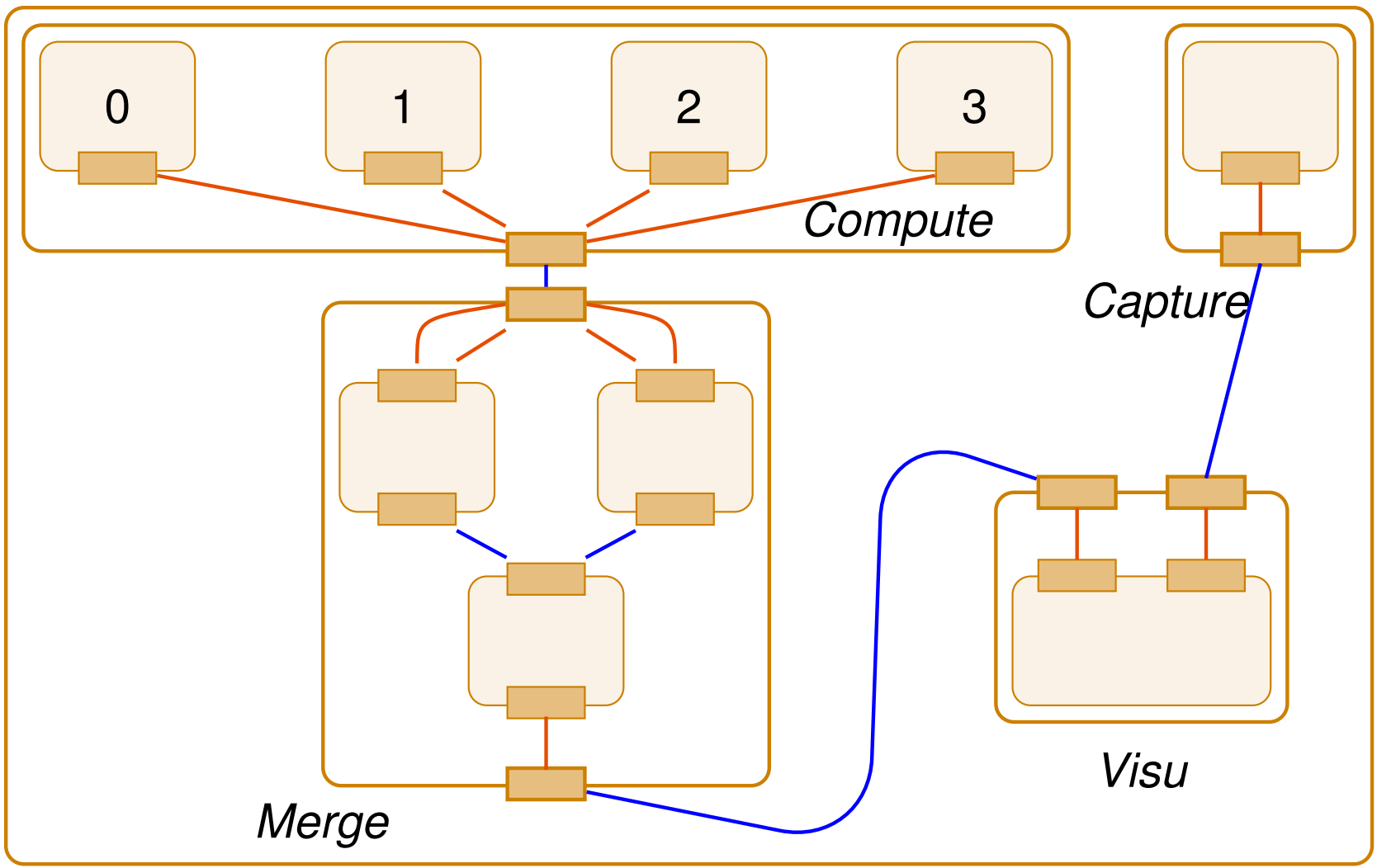
Example



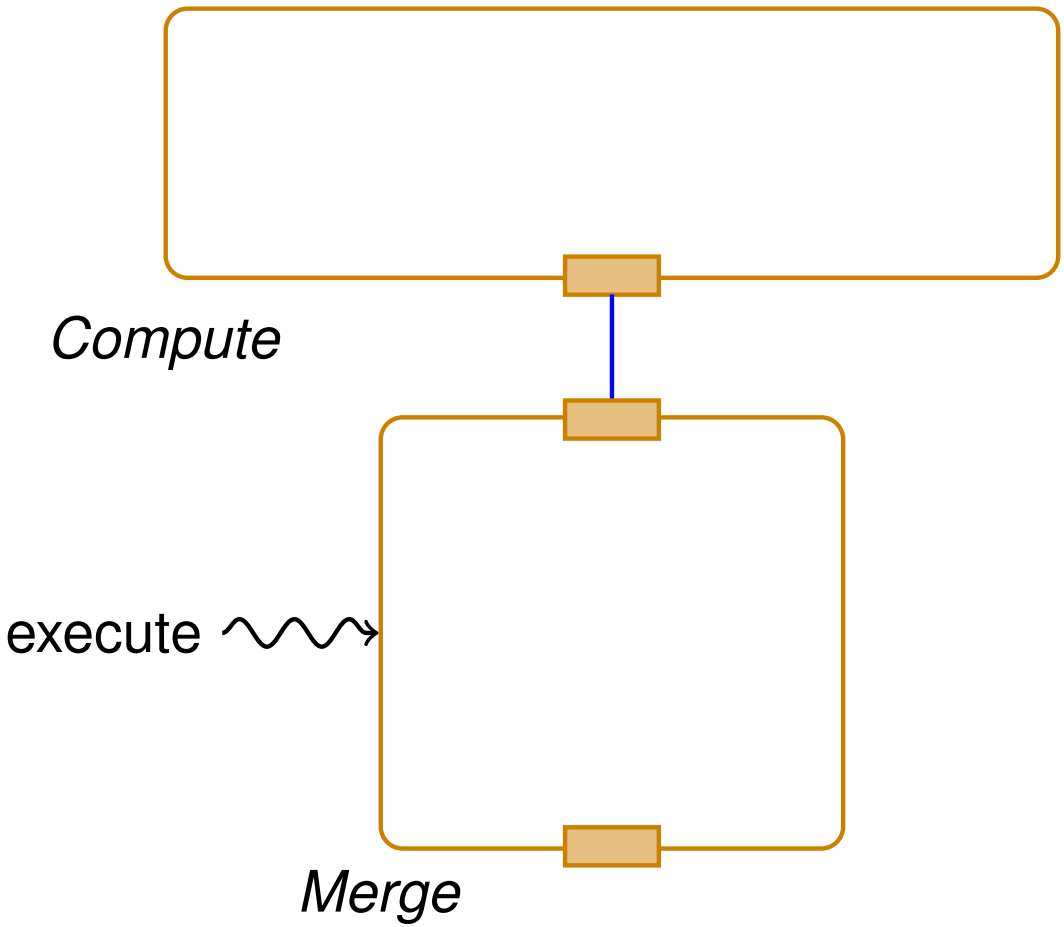
Example


0
1
2
3
Compute
Capture
Number of processes ?
execute
Merge
Visu

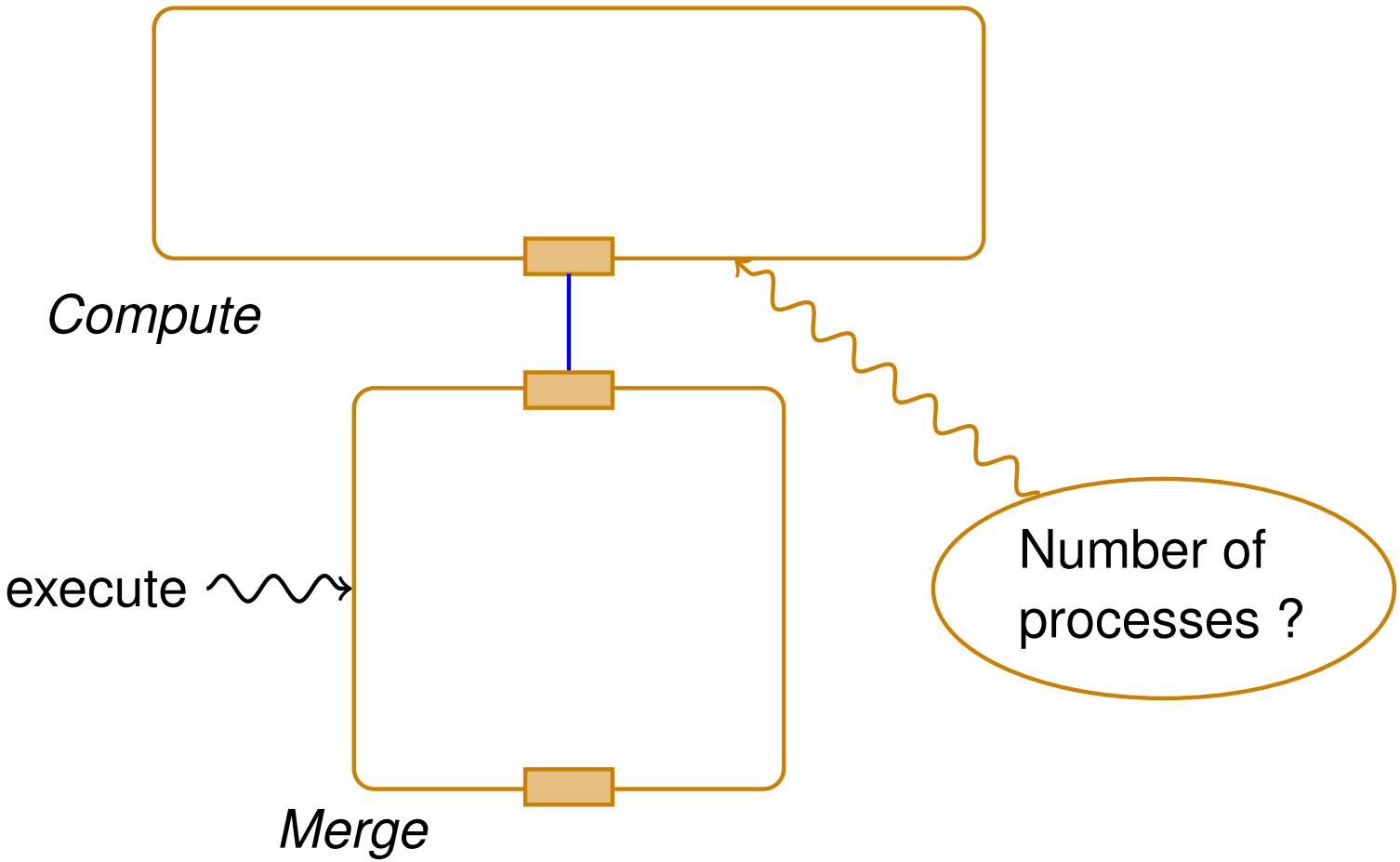
Example



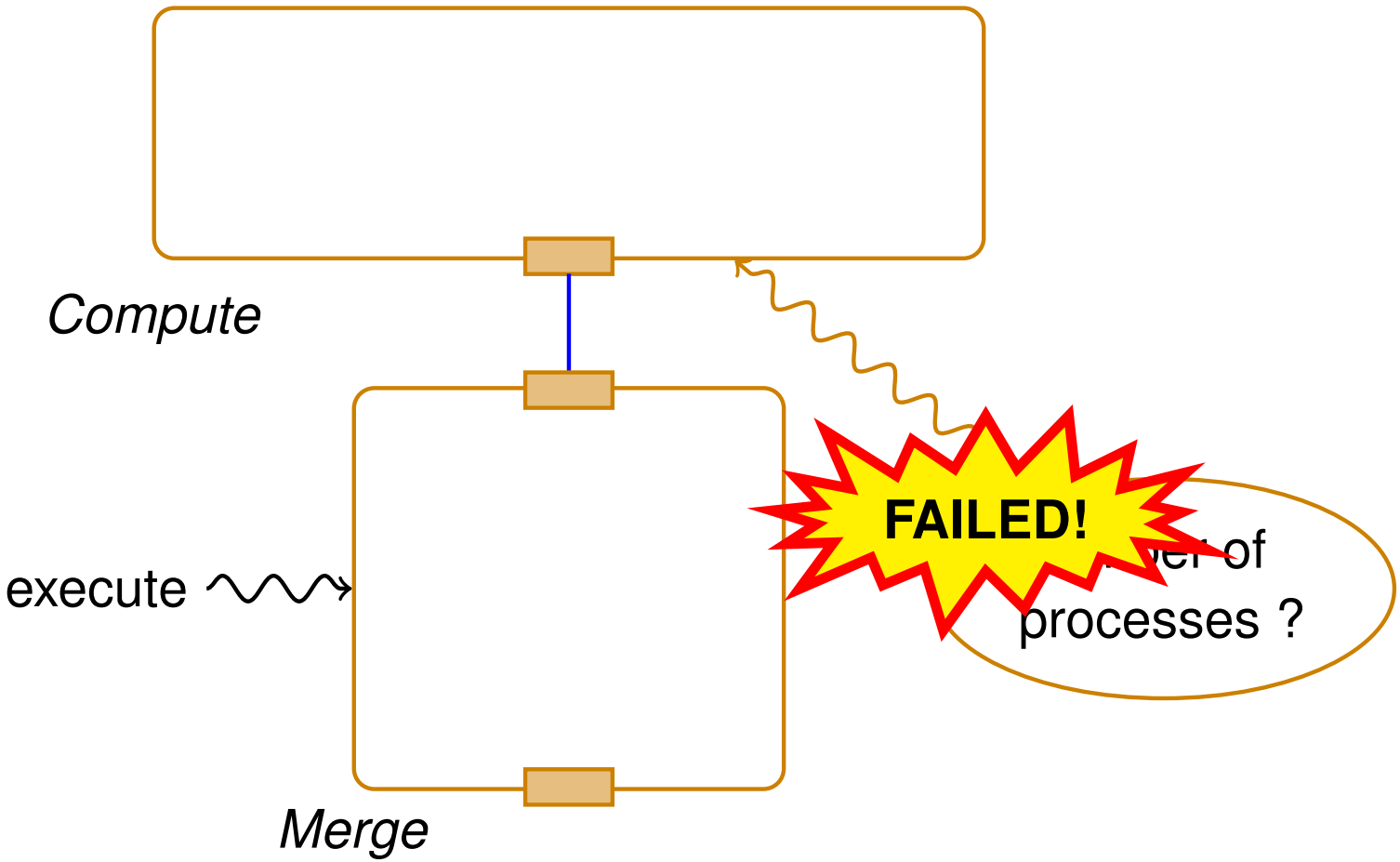
Acceptable Order



Acceptable Order



Acceptable Order



Traverse Algorithm: Motivations

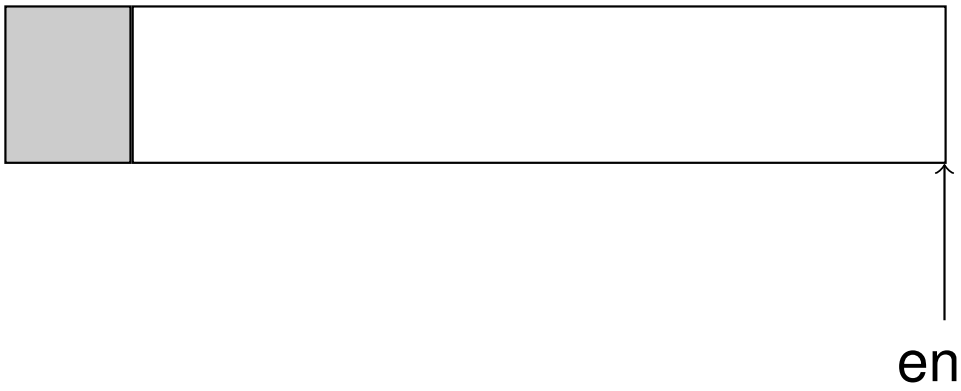
Goals:

- Controllers must be executed on all components
- Traverse must be performed in an acceptable order

Remarks:

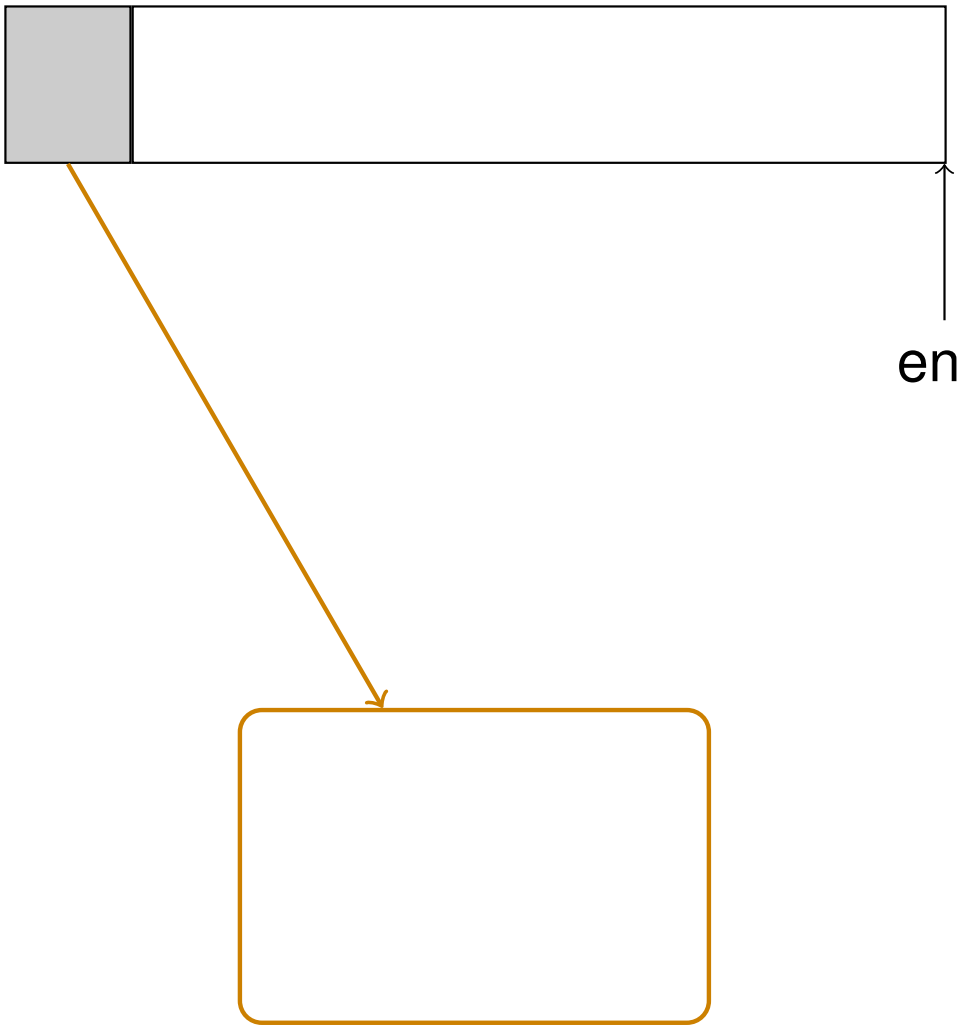
- An acceptable order may not exist (cycle dependencies).
- Express hundred constraints is error-prone and affect modularity.
- Use exceptions to find dependencies.

Traverse Algorithm [?]



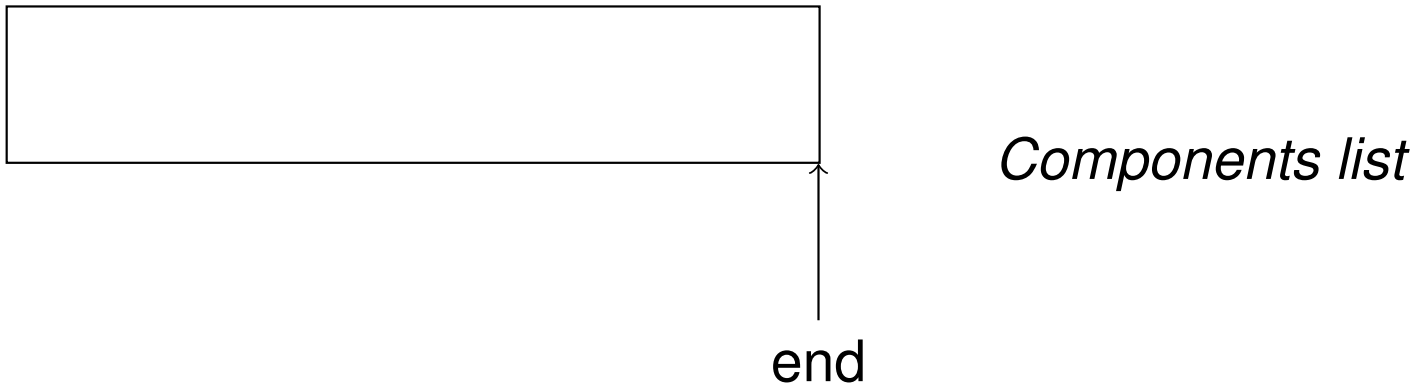
Components list

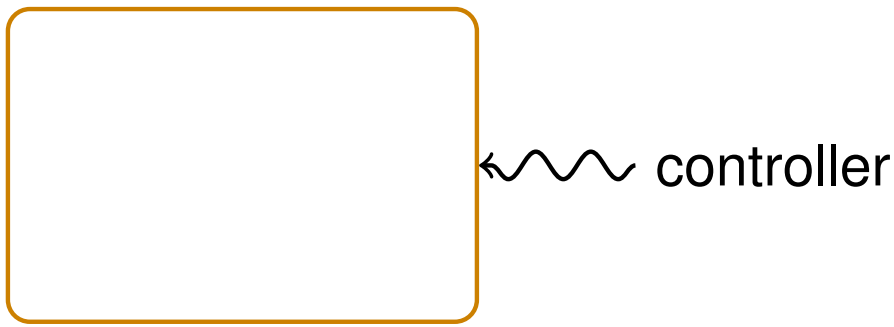
Traverse Algorithm [?]



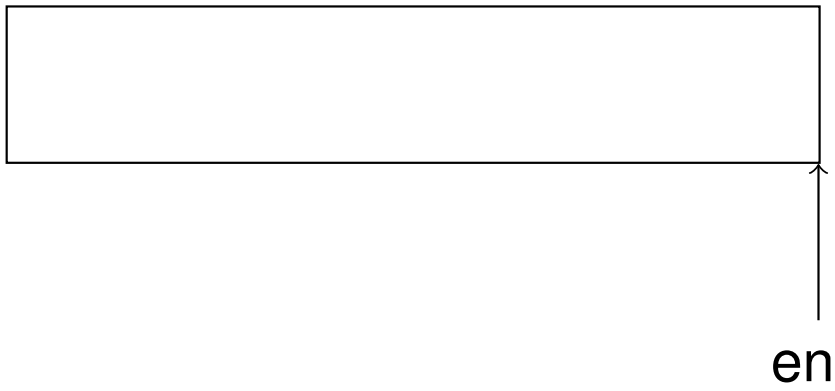
Components list

Traverse Algorithm [?]

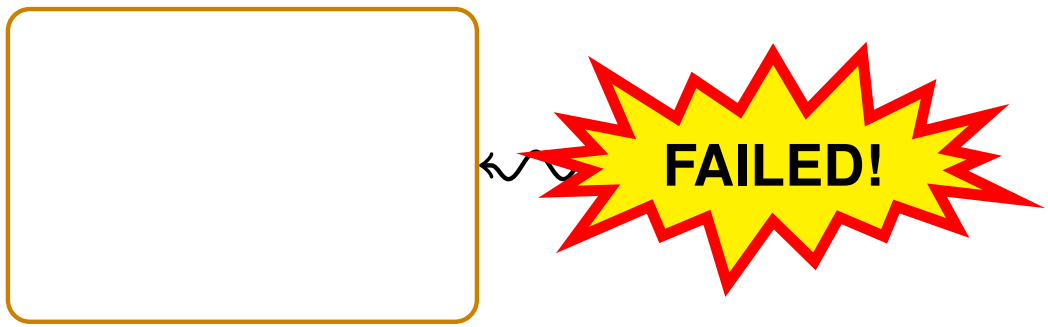




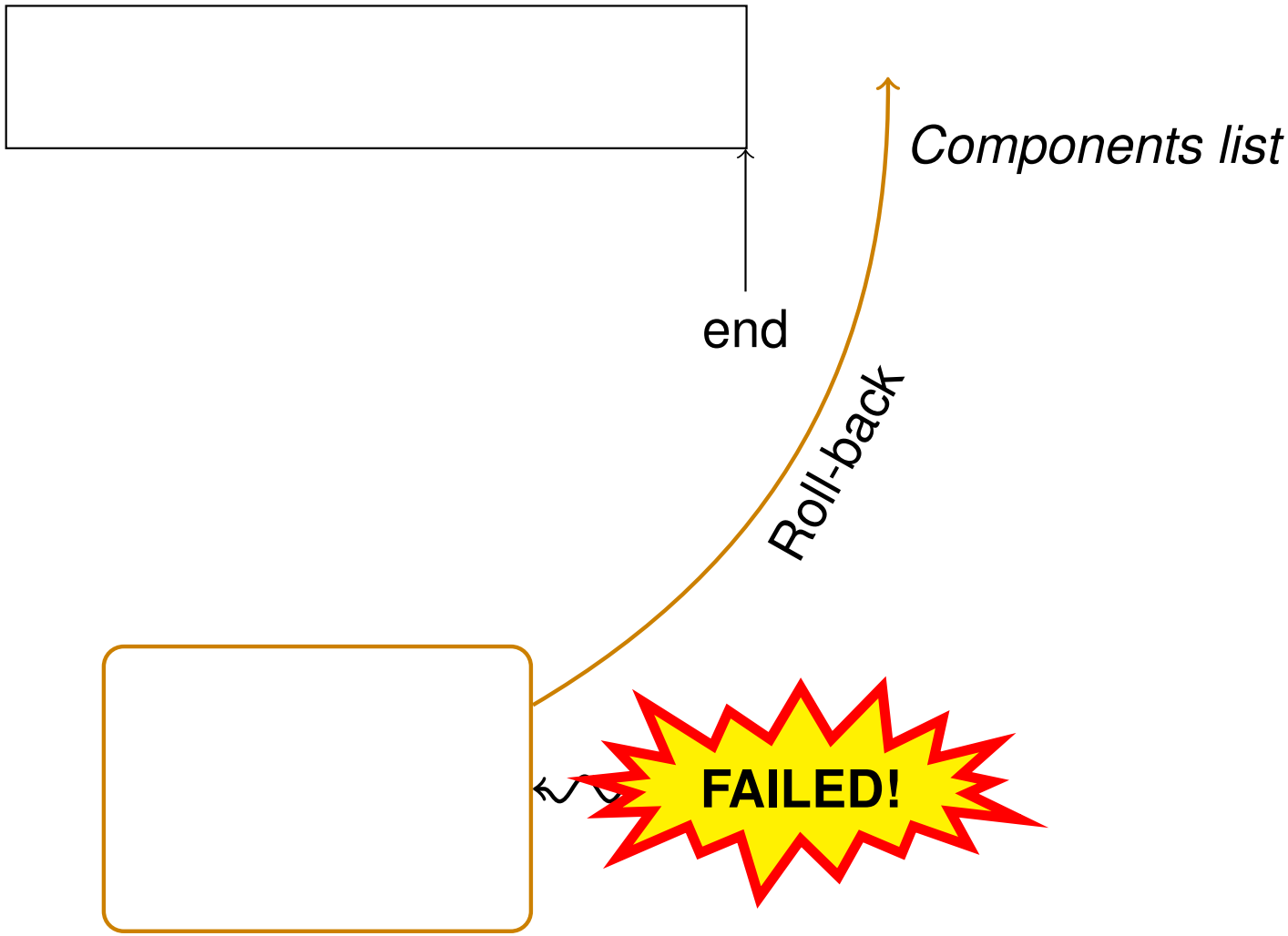
Traverse Algorithm [?]



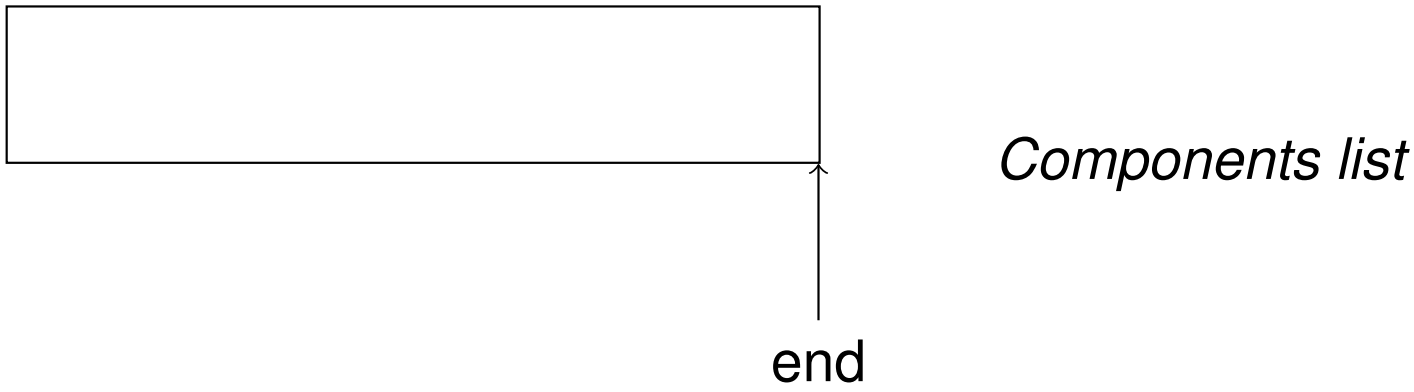
Components list

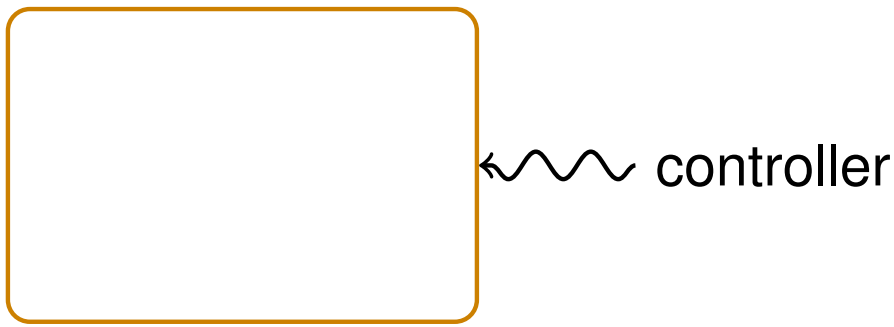


Traverse Algorithm [?]

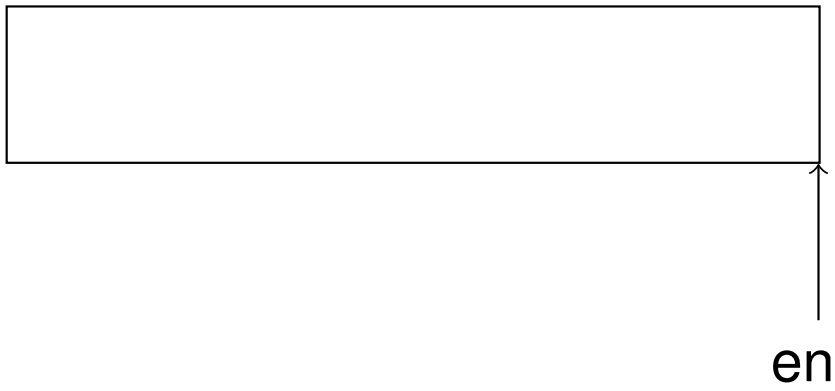


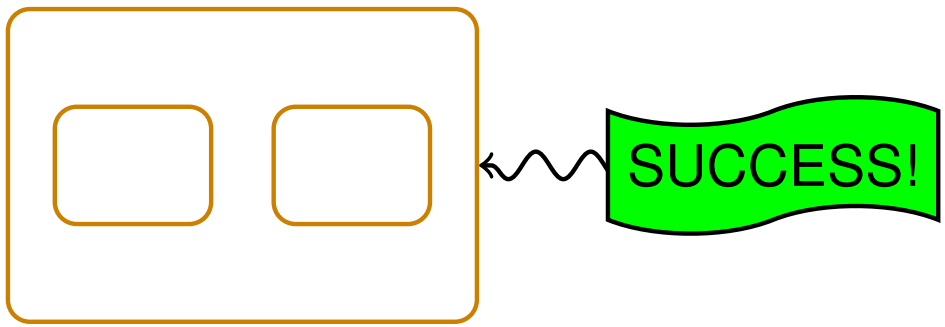
Traverse Algorithm [?]



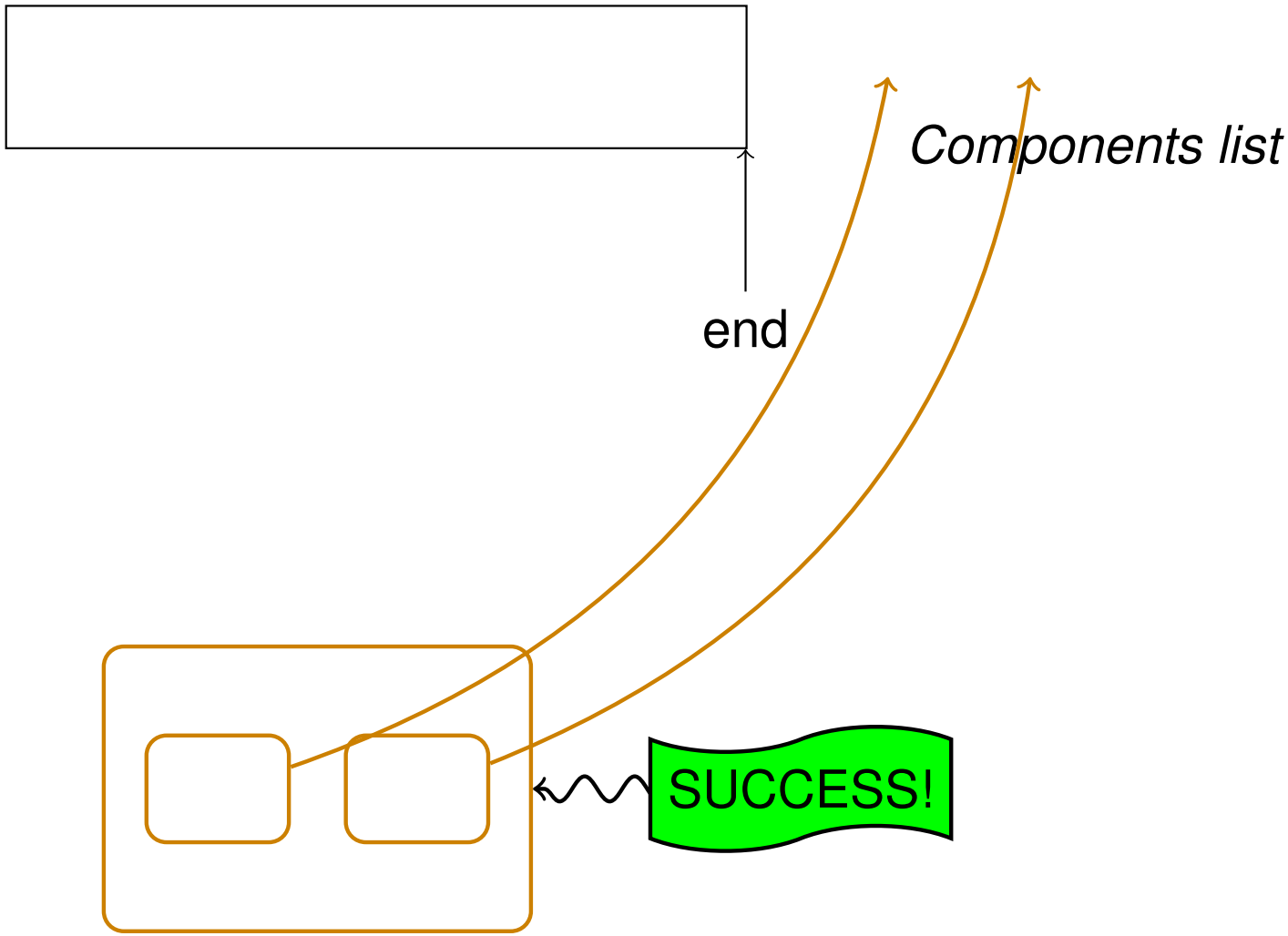


Traverse Algorithm [?]

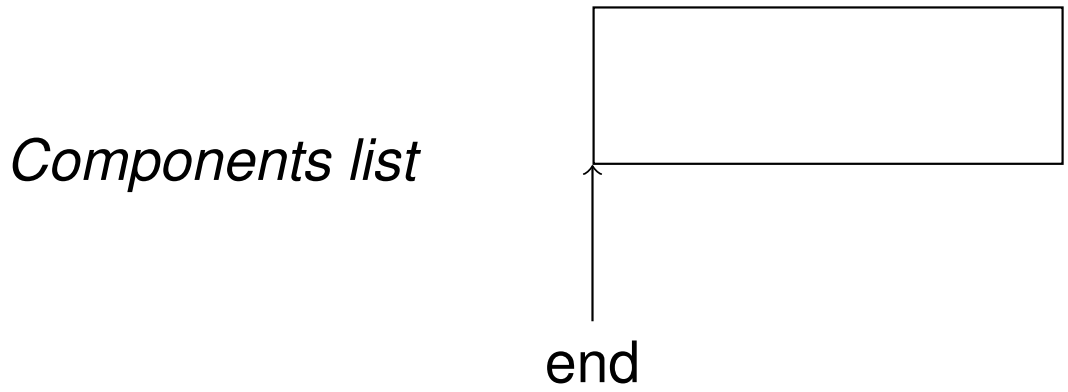




Traverse Algorithm [?]



Traverse Algorithm [?]



Traverse Algorithm [?]

Components list

Traverse Algorithm [?]

Complexity

$O(N^2)$ calls to controllers (N is the total number of components)

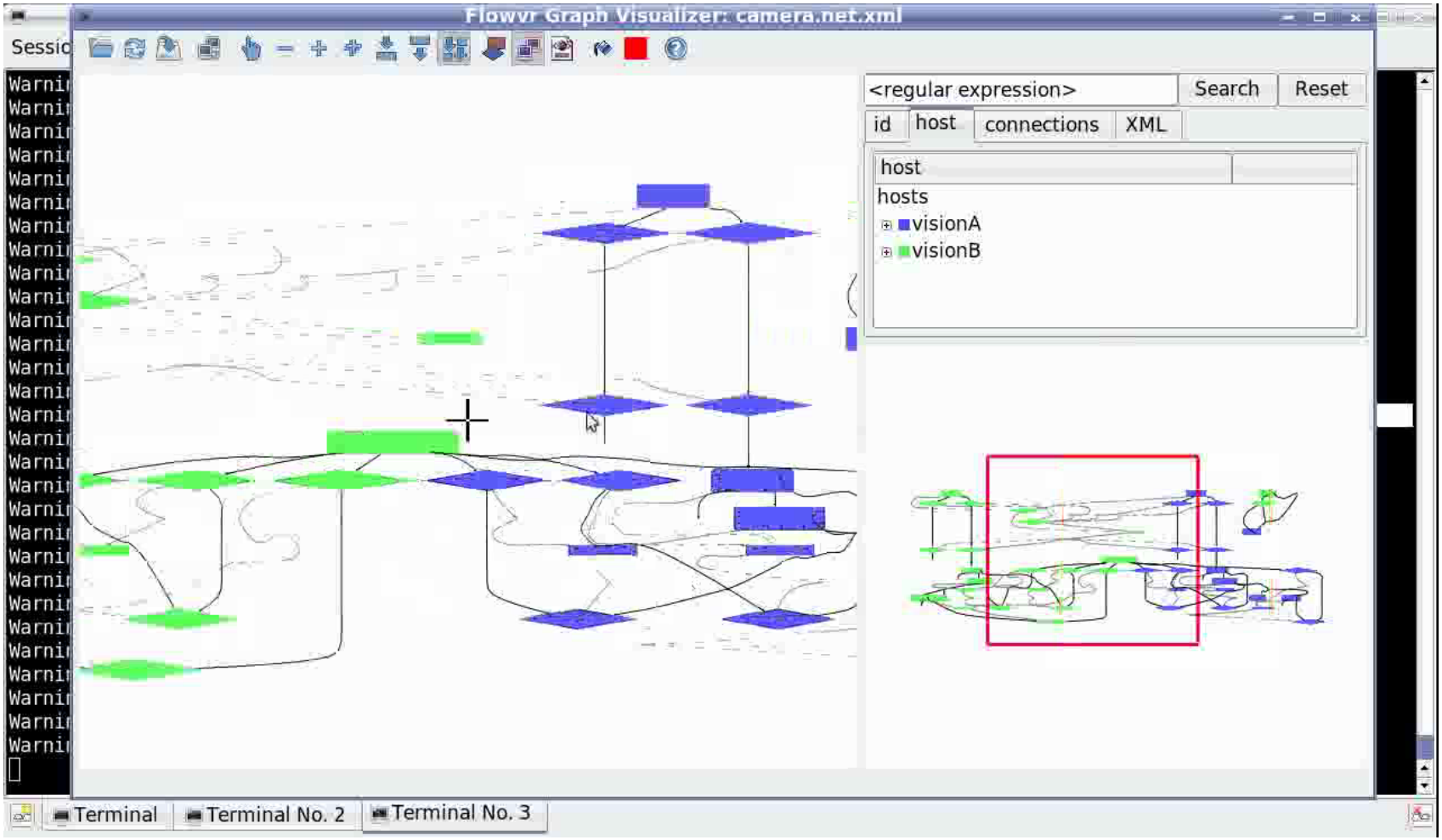
Existence of an Acceptable Order

It exists an acceptable order $\Leftrightarrow$ Components list is empty

Cyclic Dependencies Detection

Traverse algorithm provides the set of components in cyclic dependencies.

Integration on the Top of FlowVR



Outline

Samplers

Description

Tasks cannot run all at the same frequency: haptic versus simulation.

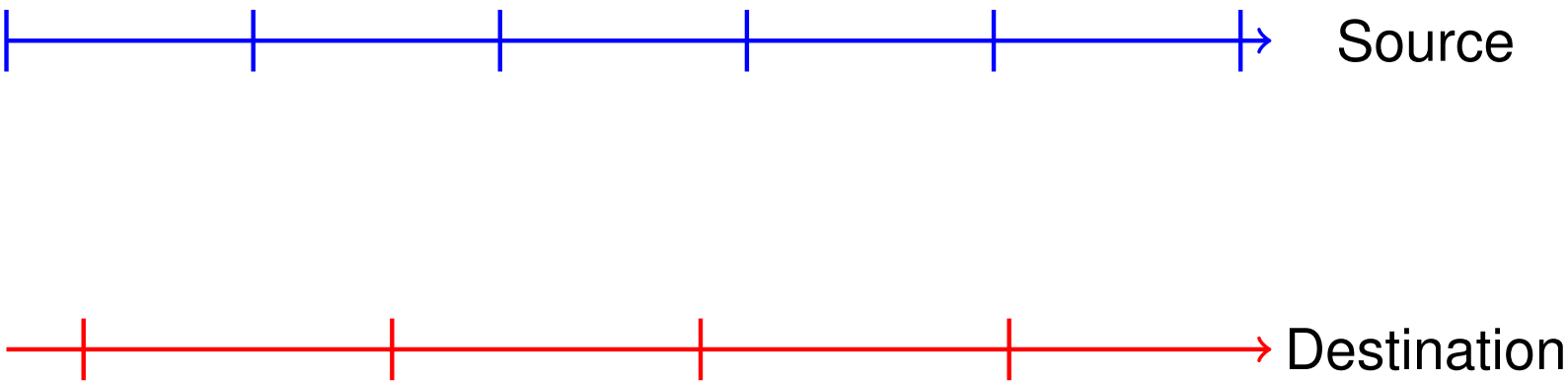
- Samplers adapt the dataflow rate.
- Samplers can discard or duplicate data.
- Example: *double-buffering*

Samplers

Description

Tasks cannot run all at the same frequency: haptic versus simulation.

- Samplers adapt the dataflow rate.
- Samplers can discard or duplicate data.
- Example: *double-buffering*

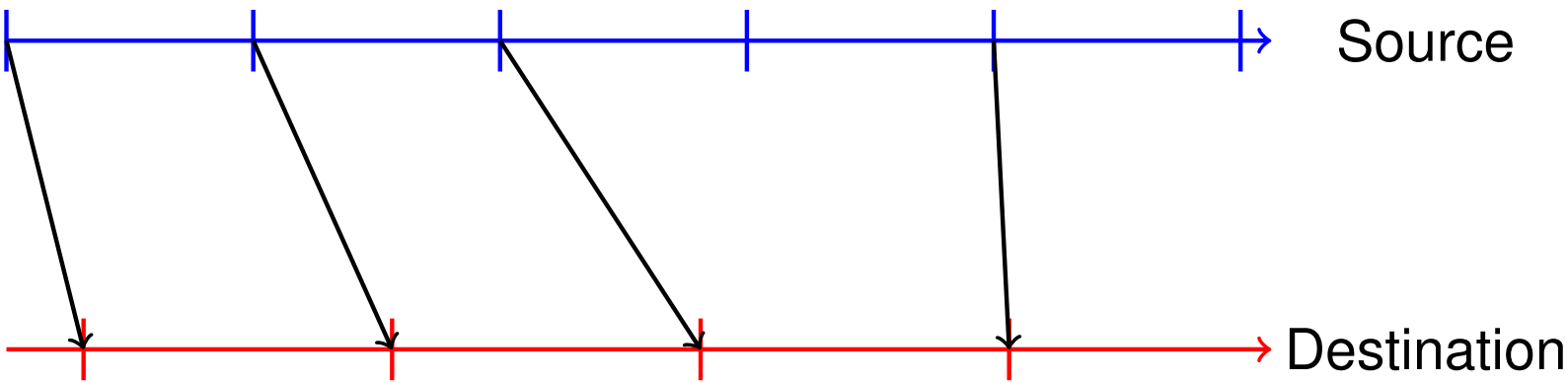


Samplers

Description

Tasks cannot run all at the same frequency: haptic versus simulation.

- Samplers adapt the dataflow rate.
- Samplers can discard or duplicate data.
- Example: *double-buffering*



Synchronization Lag [?]

Source of Latencies in an Application

Computation, data transferts, . . .

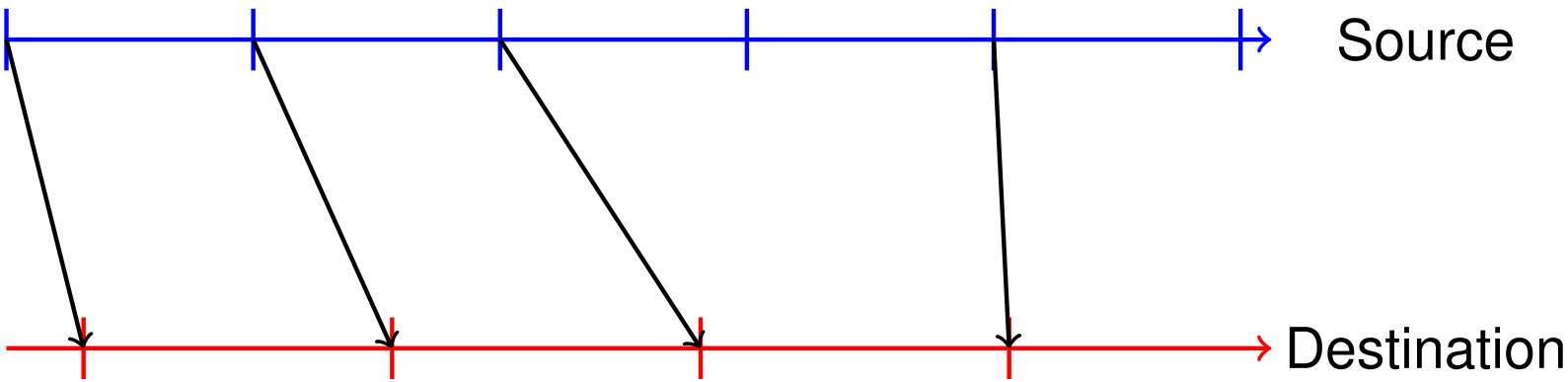
Synchronization Lag [?]

Source of Latencies in an Application

Computation, data transferts, . . .

Wloka observes a latency due to samplers

$\Rightarrow$ *Synchronization Lag*



Wloka's Observations

- Mean *Synchronization Lag*$=\frac{1}{2}.\frac{1}{f}$.
- Synchronization Lag is not constant.

Problem Statements

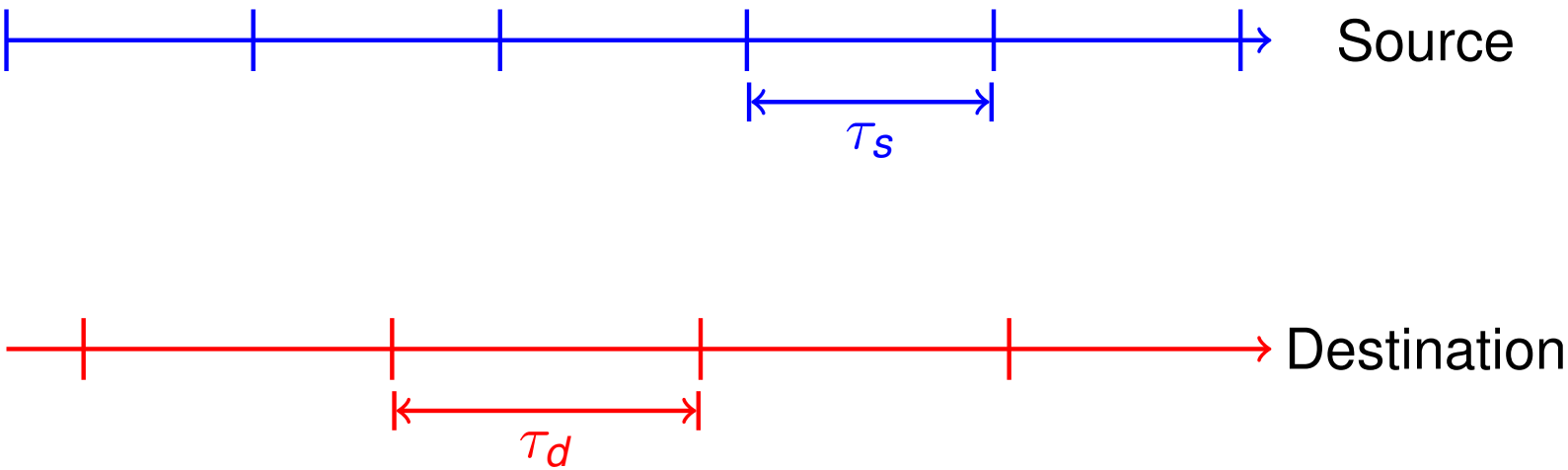
Latency and jitter have an important impact on interactivity.

The jitter is often more perturbing than an important constant latency.

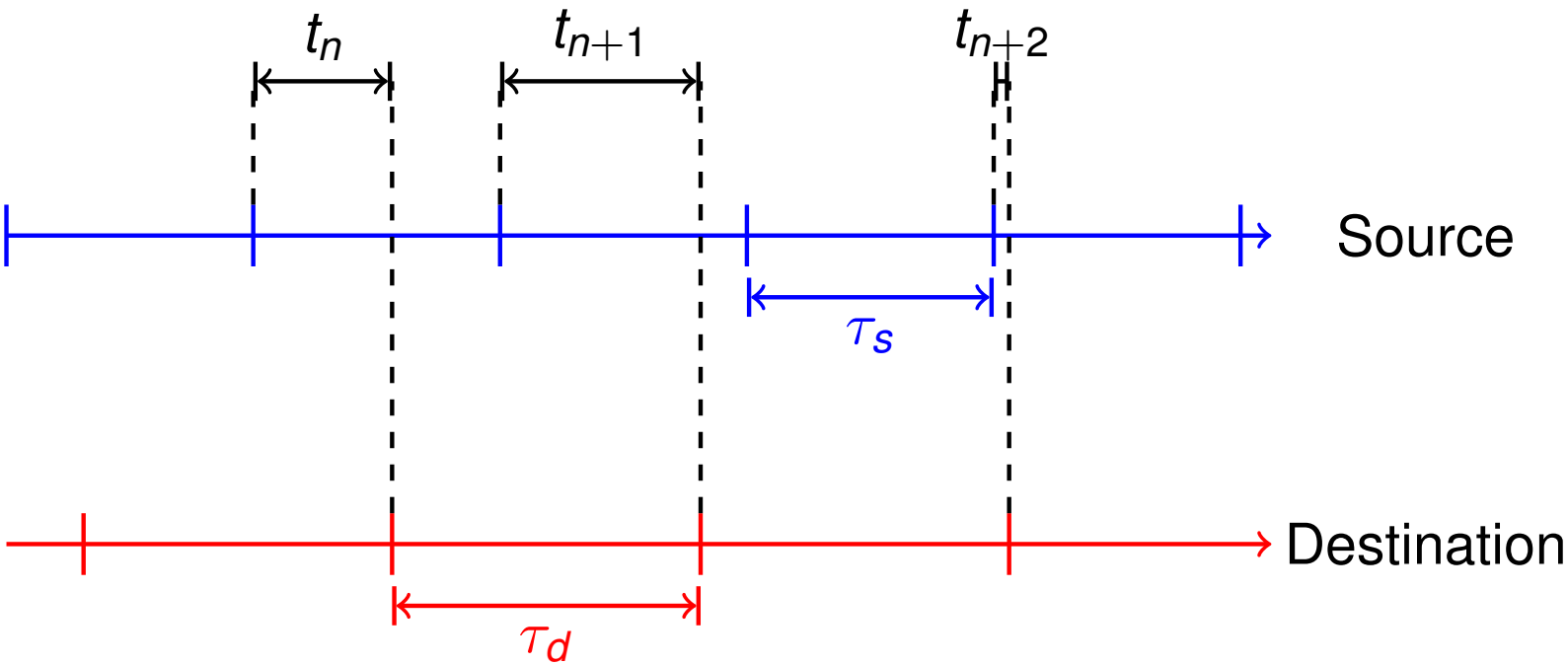
Combatting Latency in Interactive Applications

- Developers focus on latencies due to computations or data transfert.
- Developers ignore or even are not aware of *synchronization lag*

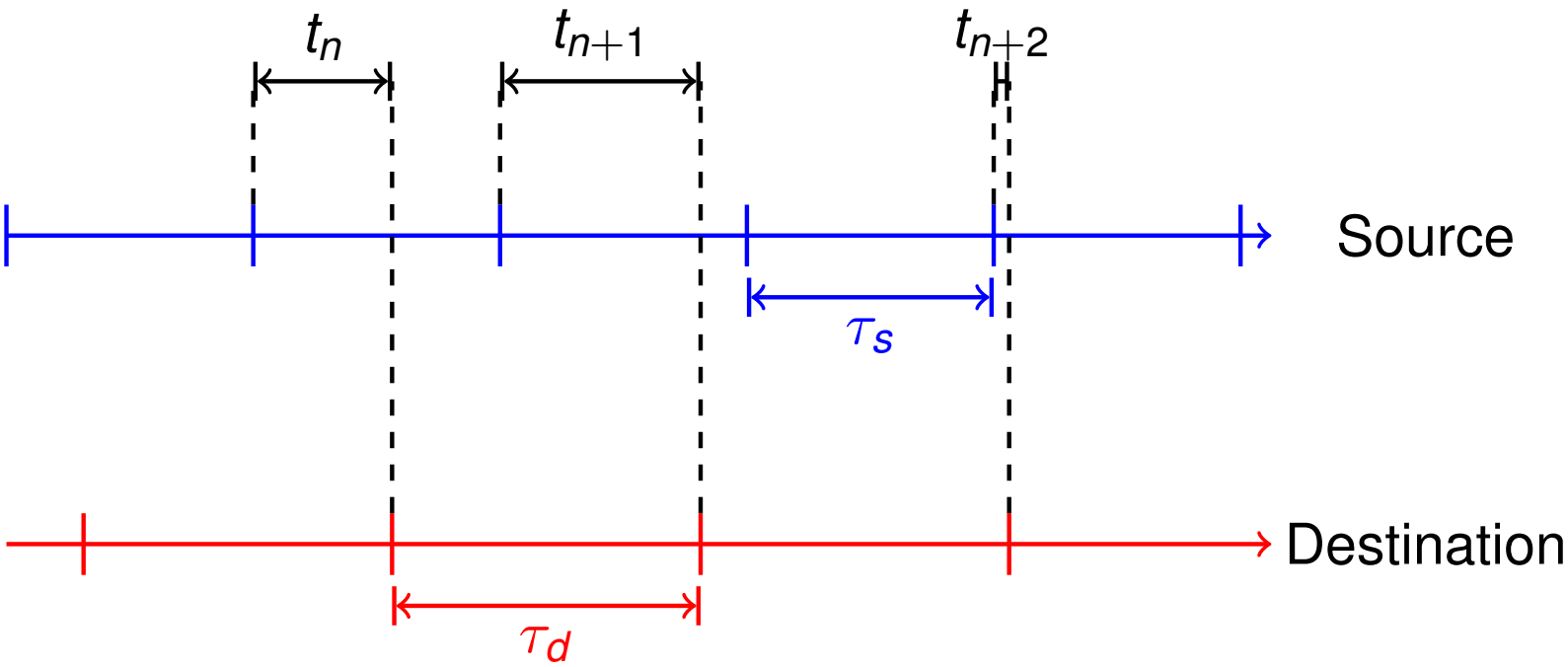
Synchronization Lag



Synchronization Lag



Synchronization Lag



Proposition

$\forall n, t_{n+1} = (t_n + \tau_d) \mod \tau_s$

Characterization of Synchronization Lag

Linear Congruential Pseudo-Random Generator

$$X_{n+1} = (a.X_n + c) \mod m$$

$\Rightarrow$ simulate a uniform distribution.

Idea:

Our expression looks like this pseudo-random generator.
Does the synchronization lag distribution fit the uniform distribution?

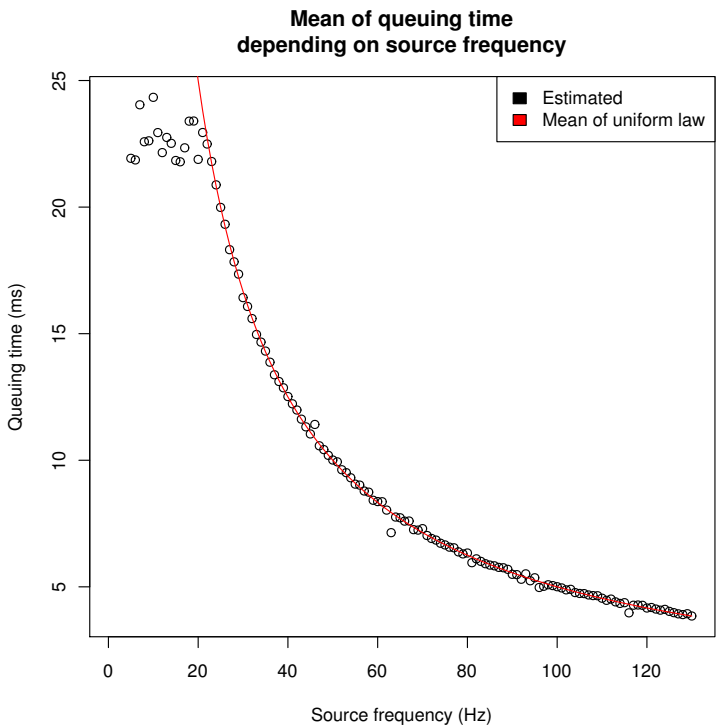
Test

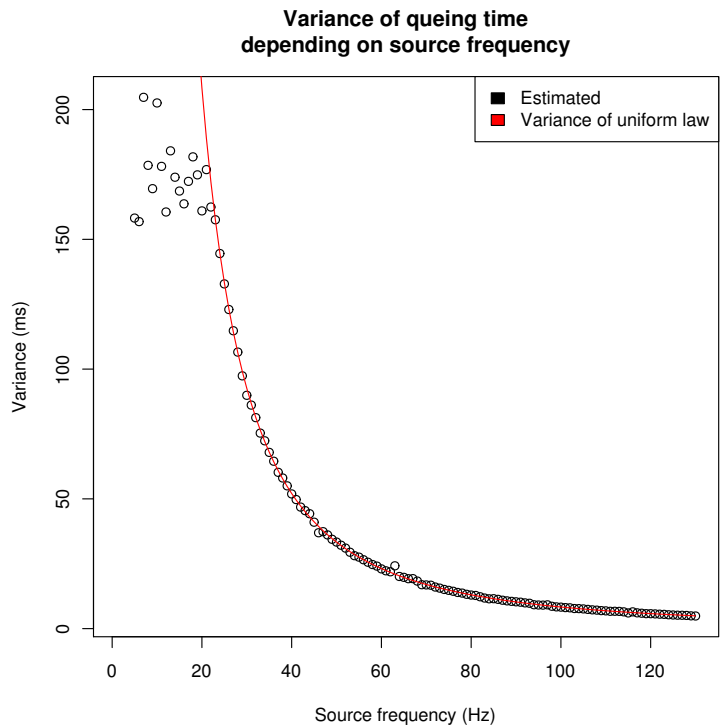
- Generate distributions using t_n expression
- Apply a statistical fit-of-goodness test
- *Result:* We can accept the hypothesis for most points

Mean and Variance of Synchronization Lag

If Synchronization Lag is $U(0, \tau_s)$ then

- $\left.\begin{array}{l} \mathrm{E}[t_n] = \frac{\tau_s}{2} = \frac{1}{2} \cdot \frac{1}{f_s} \\ t_n \in [0, \tau_s] \end{array}\right\}$ validate Wloka's observations
- $\mathrm{V}[t_n] = \frac{\tau_s^2}{12}$





Interpretations

First Idea:

Mean and Standard Deviation are proportional to τ_s.

Interpretations

First Idea:

Mean and Standard Deviation are proportional to τ_s.

Speed-up tasks!

Interpretations

First Idea:
Mean and Standard Deviation are proportional to τ_s.
Speed-up tasks!

But dispersion is constant :

$$C_v = \frac{\sqrt{\mathrm{V}[t_n]}}{\mathrm{E}[t_n]} = \frac{1}{\sqrt{3}}$$

Interpretations

First Idea:
Mean and Standard Deviation are proportional to τ_s.
Speed-up tasks!

But dispersion is constant :

$$C_v = \frac{\sqrt{\mathrm{V}[t_n]}}{\mathrm{E}[t_n]} = \frac{1}{\sqrt{3}}$$

Second Idea:
Statistical tests fail for some (τ_s, τ_d) values.
What is $\mathrm{E}[q_n]$ and $\mathrm{V}[q_n]$ in these cases ?

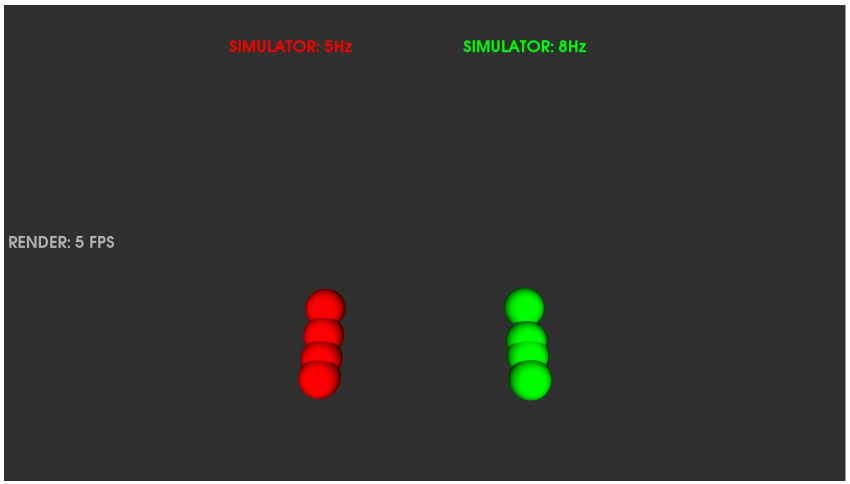
A Simple Condition to Remove Jitter

If $\tau_d \mod \tau_s = 0$ then

$$\begin{aligned} t_{n+1} &= (t_n + \tau_d) \mod \tau_s \\ \Rightarrow t_{n+1} &= t_n \end{aligned}$$

Mean and Variance Are

$$\begin{aligned} \mathrm{E}[t_n] &= t_0 \\ \mathrm{V}[t_n] &= 0 \end{aligned}$$



Future Works

Our Result

We should maximize frequencies under constraint $\tau_d \mod \tau_s = 0$

It is more complex than *“Go as fast as possible”*!

Future Works

Our Result

We should maximize frequencies under constraint $\tau_d \mod \tau_s = 0$
It is more complex than *"Go as fast as possible"*!

Questions:

- How measure τ_i online ?
- What is the impact of perturbations on the result ?
- Other sampling policies ?

Outline

Summary

Questions:

- How to handle complexity?
- How to execute these applications under strong performance constraints ?

Hierarchical Components Model

- Add modularity
- Compilation stage (tune the application)

Synchronization Lag

- Sampling is mandatory
- Few studies about sampling
- How to reduce sampling impact ?

My Experience with Large Interactive Applications

Grimage and ANR Dalia

Real-time 3D reconstruction.

Demonstrators: VRST08, Siggraph09, Fête de la Science (Grand Palais, Paris),...

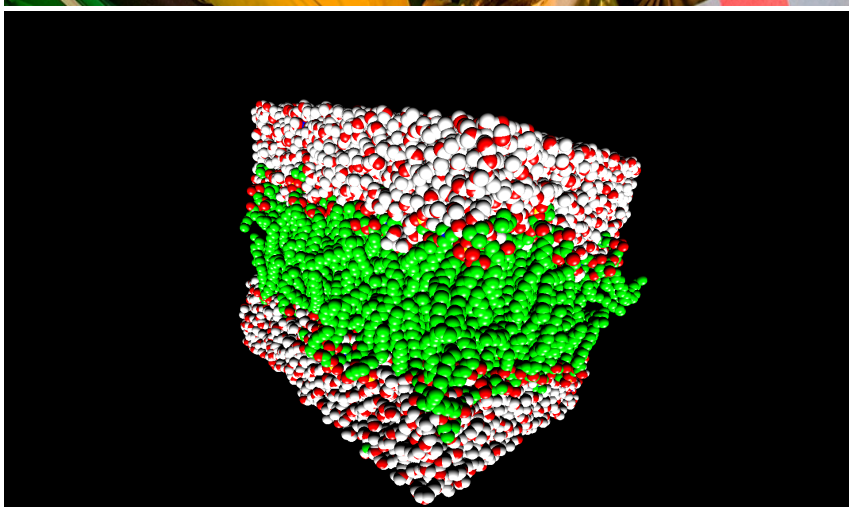
ANR FVNano

Real-time interactions with molecular simulation.

FlowVR

Middleware for Interactive Applications.

`http://flowvr.sf.net/`



Perspectives

Architecture will Become more and more Complex

Many-cores (*Larrabee*, *Fermi*, ...), clusters, grids, high-performance network

We should delegate complexity to middlewares

Perspectives

Architecture will Become more and more Complex

Many-cores (*Larrabee*, *Fermi*, ...), clusters, grids, high-performance network

We should delegate complexity to middlewares

Main Problem

Interactivity is a human feeling.

How a middleware can decide if an application is interactive ?

Perspectives

Architecture will Become more and more Complex

Many-cores (*Larrabee*, *Fermi*, ...), clusters, grids, high-performance network

We should delegate complexity to middlewares

Main Problem

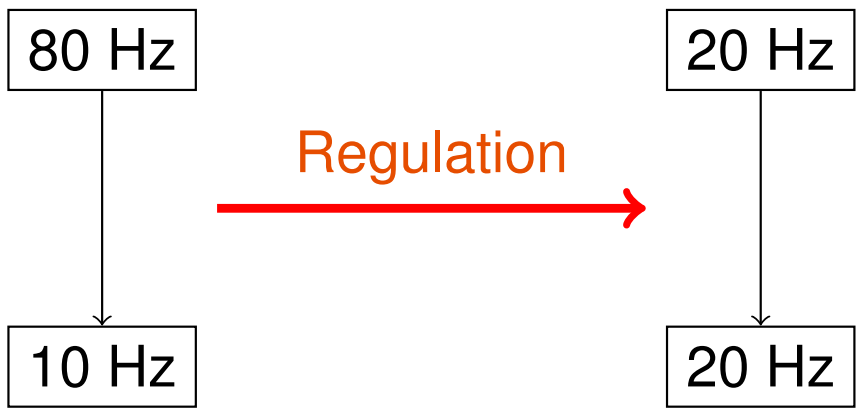
Interactivity is a human feeling.
How a middleware can decide if an application is interactive ?
It is an open question $\Rightarrow$ best-effort

The optimization problem is multi-criteria: latencies, frequencies, level of details, simulation accuracy, ...

Perspectives

Interactive Application Specific: Sampling

- Synchronization Lag
- Efficient sampling algorithms, prediction, dead-reckoning ?
- ($N \times M$ + Sampling) communications
- Regulation



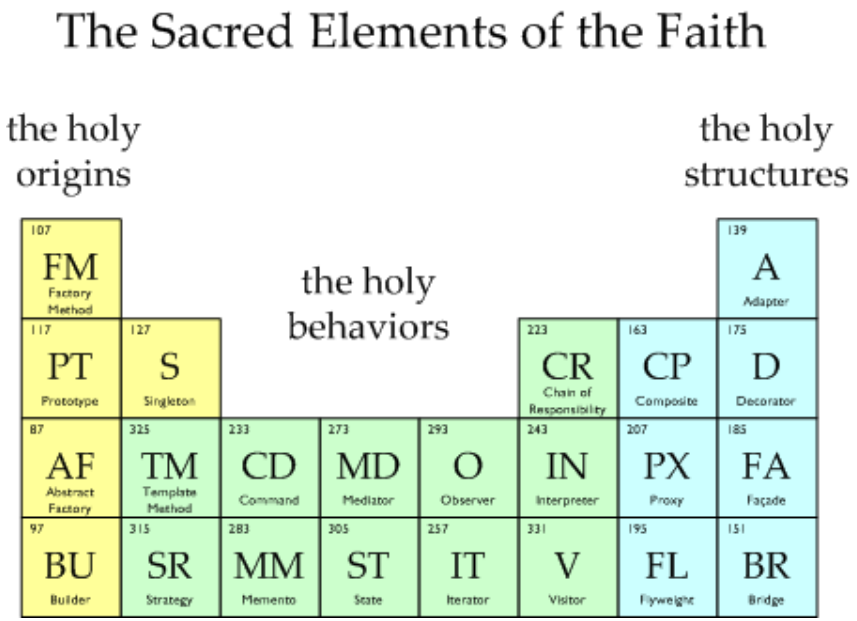
Thank You

Modularity: Design Patterns

Main Idea

Provide an efficient cookbook to developers

- They list well-known developers problem.
- They do not provide any implementation.
- There are some extensions for parallel computing.



A Feed-Back

Example of the Grimage Component

- 158 C++ lines
- 90% lines are related to communication

At high-level, few components and lot of ports and links.

A Benefit of the Hierarchical Structure

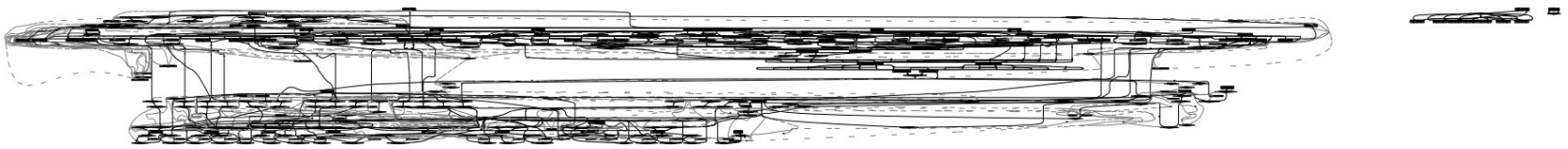


Figure: Simple layout algorithm

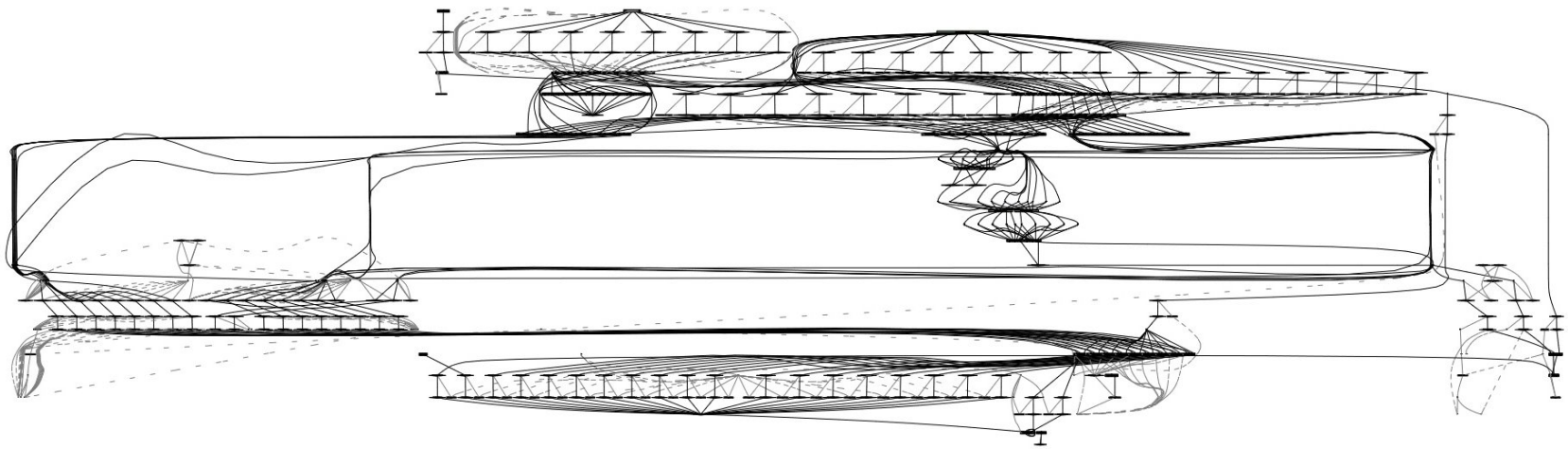


Figure: Layout algorithm relies on the hierarchical structure