UNIVERSITÉ DE LORRAINE

Loria
Laboratoire lorrain de recherche
en informatique et ses applications

An Exercise in Multi-modeling

Jean-Pierre Jacquot

IVOIRE - Bergamo

June 2024

Agenda

1.Motivation

2.Protocol

3.Implantation

4. Conclusion and future works

Motivation

System and Components in Event-B

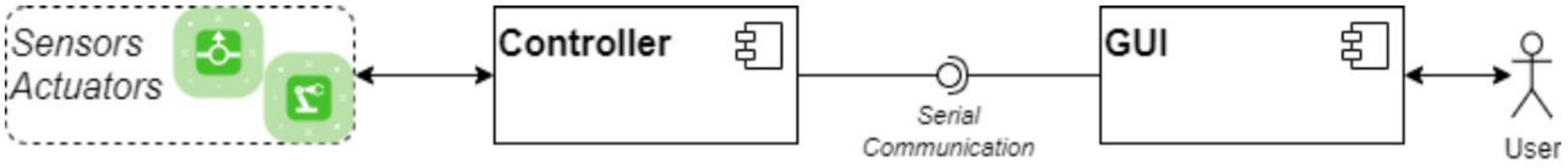


Figure 1.1: The high-level software architecture

Motivation

System and Components in Event-B

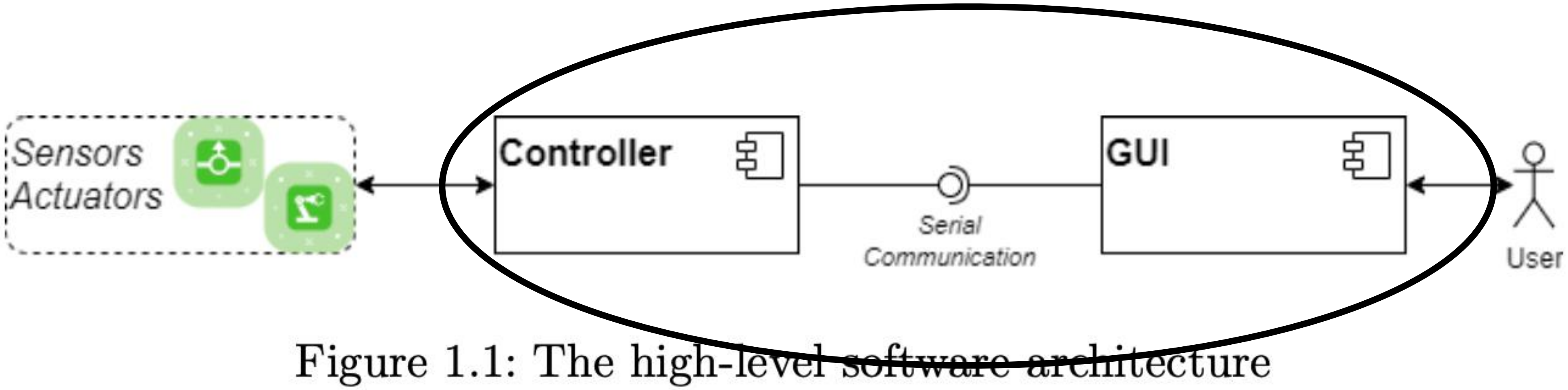


Figure 1.1: The high-level software architecture

Motivation

System and Components in Event-B

2 software components :

- loosely coupled
- invite independent modelling

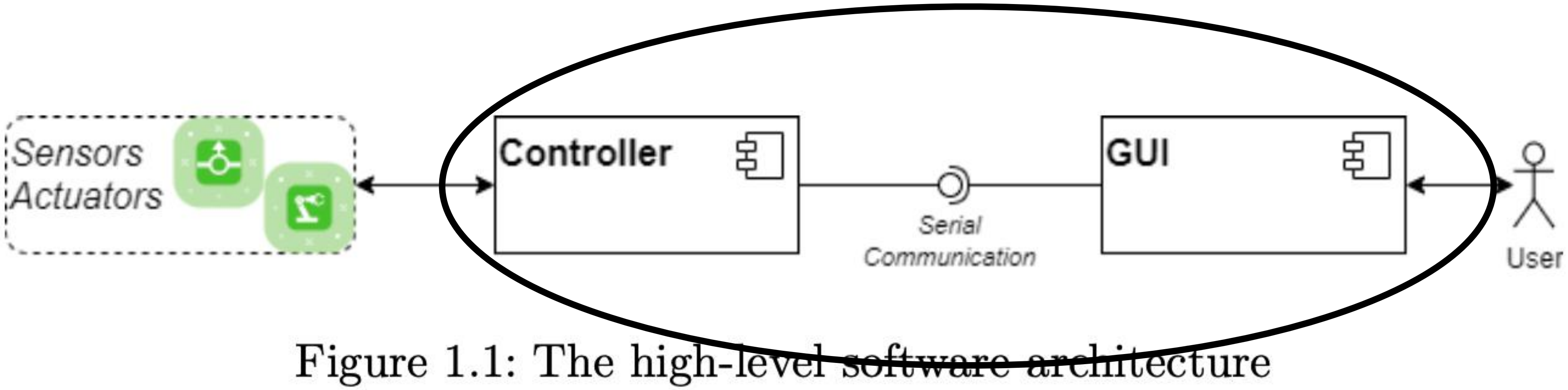
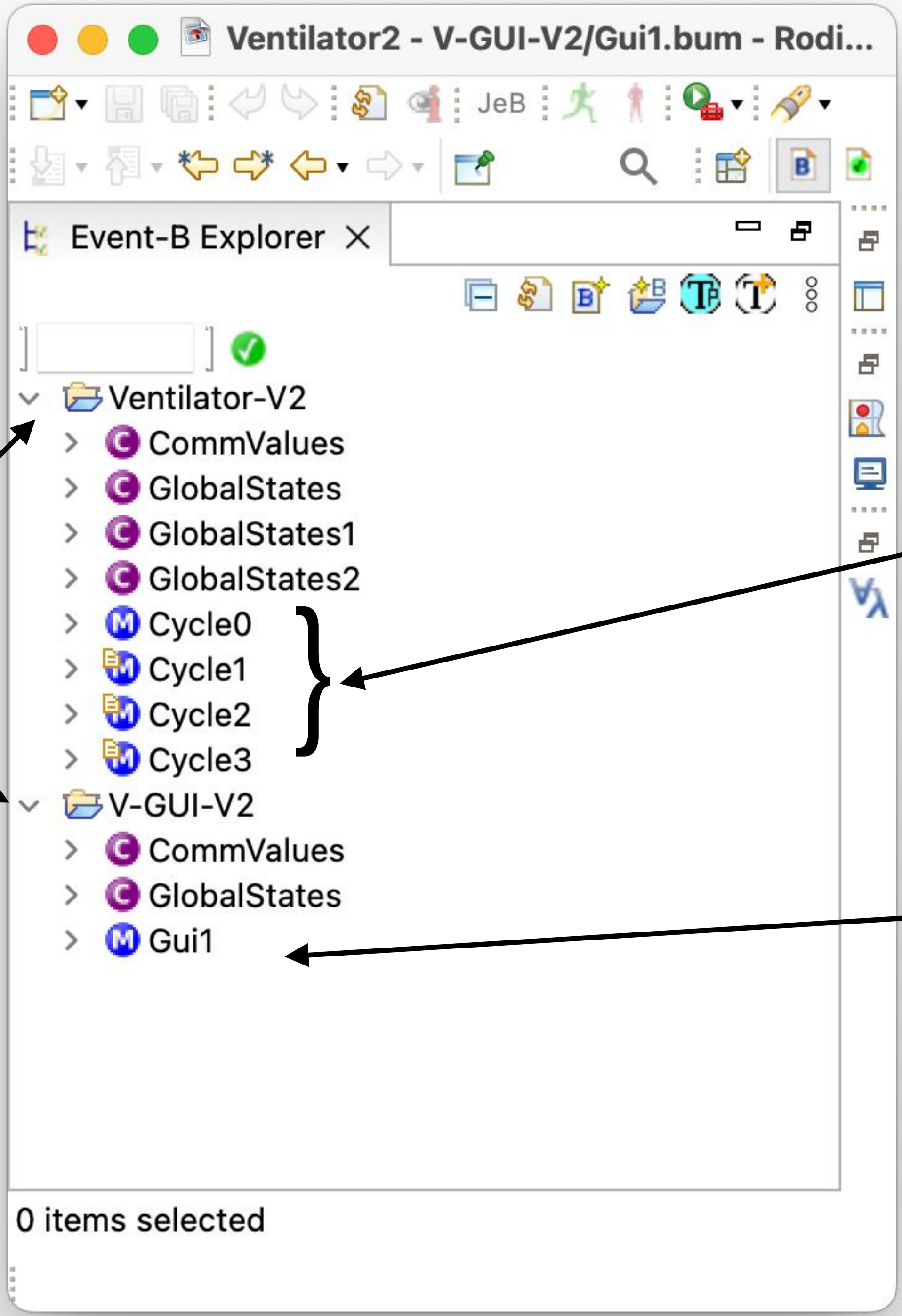


Figure 1.1: The high-level software architecture

Motivation

Modeling strategy

- Let’s do it!



Ventilator2 - V-GUI-V2/Gui1.bum - Rodi...

Event-B Explorer

- Ventilator-V2
  - CommValues
  - GlobalStates
  - GlobalStates1
  - GlobalStates2
  - Cycle0
  - Cycle1
  - Cycle2
  - Cycle3
- V-GUI-V2
  - CommValues
  - GlobalStates
  - Gui1

0 items selected

2 separate models

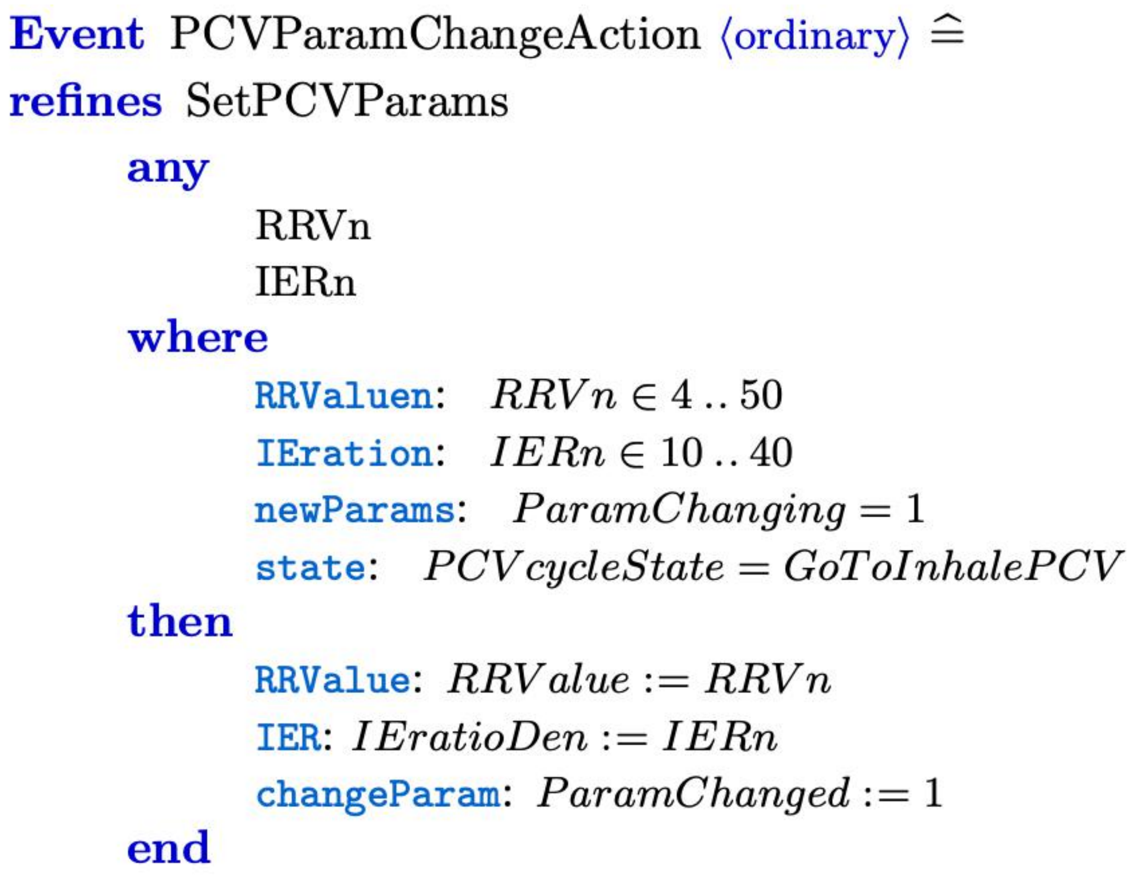
3 refinements

a very simple machine

Motivation

Event-B

- Event-B model: state + guarded transition

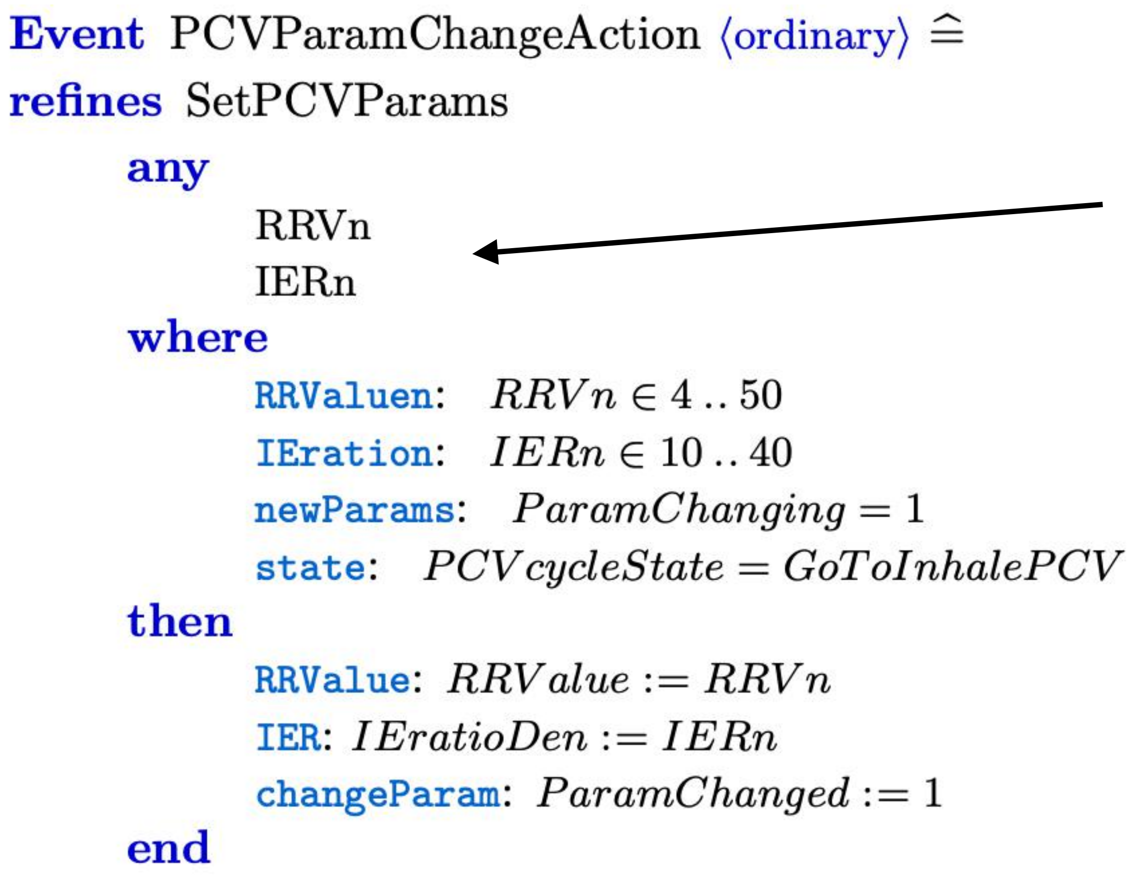


```
Event PCVParamChangeAction ⟨ordinary⟩ ≙
refines SetPCVParams
      any
            RRVn
            IERn
      where
            RRValuen:   $RRVn \in 4 \mathbin{..} 50$
            IEration:   $IERn \in 10 \mathbin{..} 40$
            newParams:   $ParamChanging = 1$
            state:   $PCVcycleState = GoToInhalePCV$
      then
            RRValue: $RRValue := RRVn$
            IER: $IEratioDen := IERn$
            changeParam: $ParamChanged := 1$
      end
```

Motivation

Event-B

- Event-B model: state + guarded transition



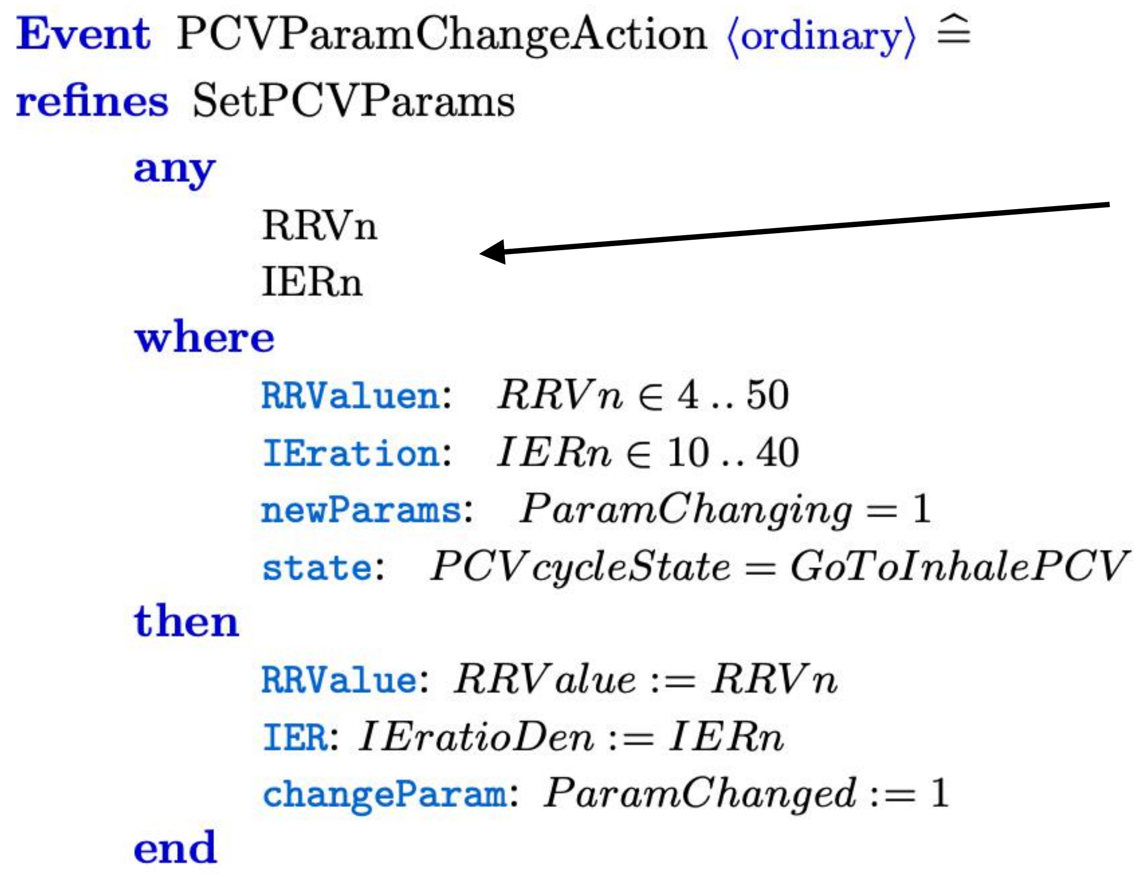
```
Event PCVParamChangeAction ⟨ordinary⟩ ≙
refines SetPCVParams
      any
            RRVn
            IERn
      where
            RRValuen:   RRVn ∈ 4 .. 50
            IEration:   IERn ∈ 10 .. 40
            newParams:   ParamChanging = 1
            state:   PCVcycleState = GoToInhalePCV
      then
            RRValue: RRValue := RRVn
            IER: IEratioDen := IERn
            changeParam: ParamChanged := 1
      end
```

event parameters

Motivation

Event-B

- Event-B model: state + guarded transition



```
Event PCVParamChangeAction ⟨ordinary⟩ ≙
refines SetPCVParams
    any
        RRVn
        IERn
    where
        RRValuen:  RRVn ∈ 4 .. 50
        IEration:  IERn ∈ 10 .. 40
        newParams:  ParamChanging = 1
        state:  PCVcycleState = GoToInhalePCV
    then
        RRValue: RRValue := RRVn
        IER: IERatioDen := IERn
        changeParam: ParamChanged := 1
    end
```

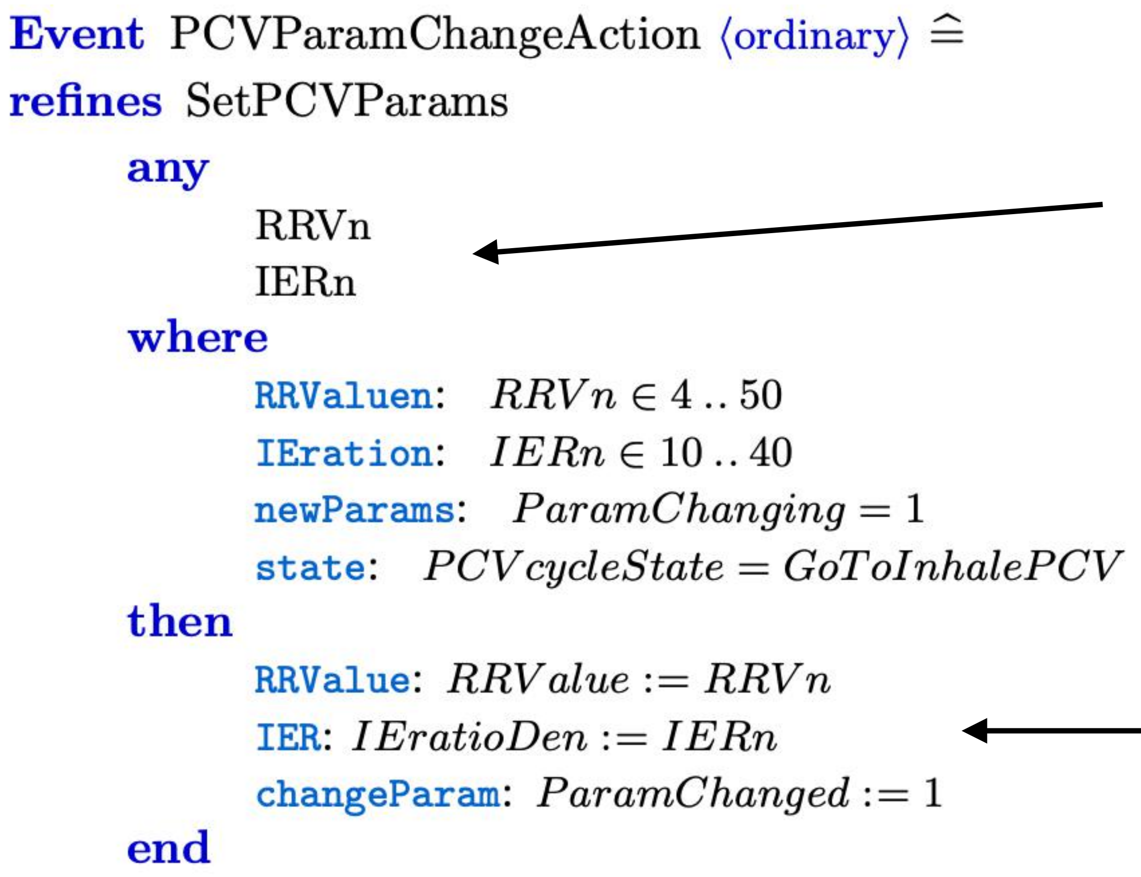
event parameters

guard on the state

Motivation

Event-B

- Event-B model: state + guarded transition



```
Event PCVParamChangeAction ⟨ordinary⟩ ≙
refines SetPCVParams
      any
            RRVn
            IERn
      where
            RRValuen:   RRVn ∈ 4 .. 50
            IEration:   IERn ∈ 10 .. 40
            newParams:   ParamChanging = 1
            state:   PCVcycleState = GoToInhalePCV
      then
            RRValue: RRValue := RRVn
            IER: IEratioDen := IERn
            changeParam: ParamChanged := 1
      end
```

event parameters

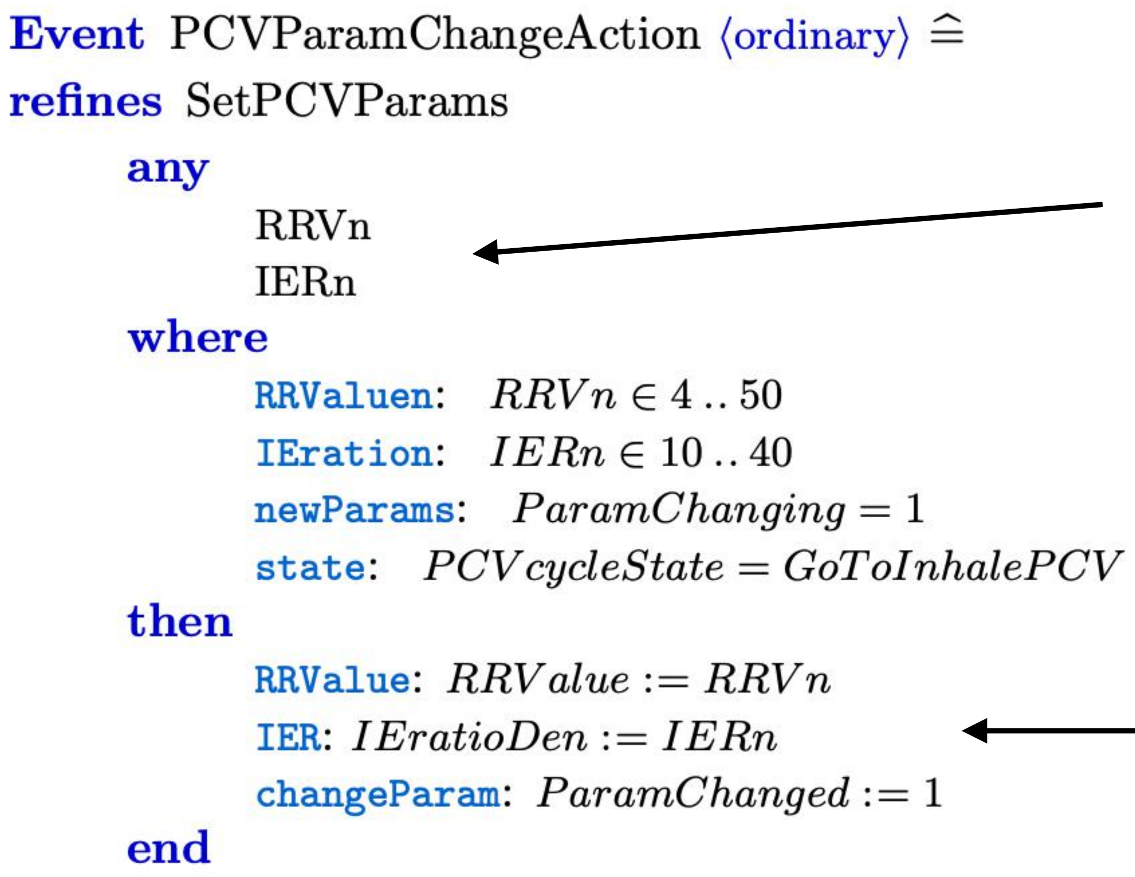
guard on the state

state modification

Motivation

Event-B

- Event-B model: state + guarded transition



```
Event PCVParamChangeAction ⟨ordinary⟩ ≙
refines SetPCVParams
      any
            RRVn
            IERn
      where
            RRValuen:   RRVn ∈ 4 .. 50
            IEration:   IERn ∈ 10 .. 40
            newParams:   ParamChanging = 1
            state:   PCVcycleState = GoToInhalePCV
      then
            RRValue: RRValue := RRVn
            IER: IEratioDen := IERn
            changeParam: ParamChanged := 1
      end
```

event parameters

guard on the state

state modification

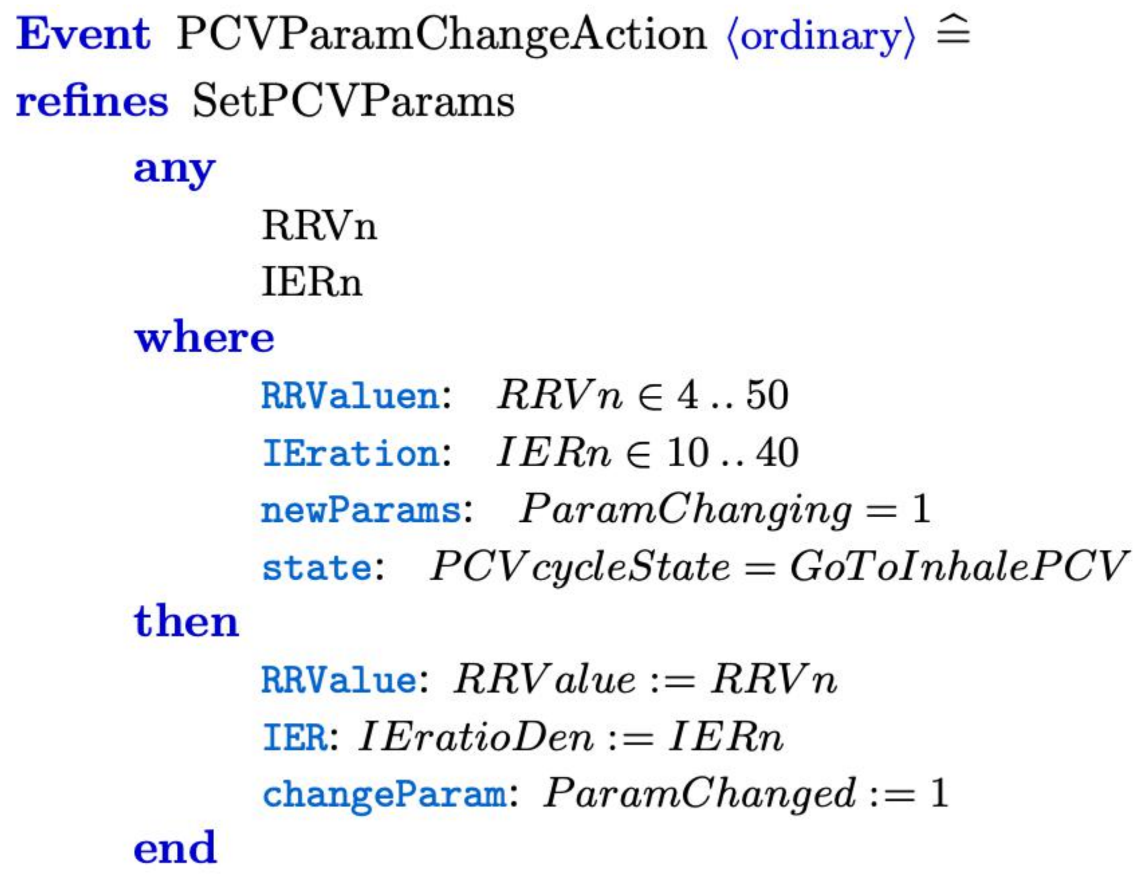
guard true => event can be trigged

triggered event => state is modified

Motivation

Event-B

- Event-B model: state + guarded transition

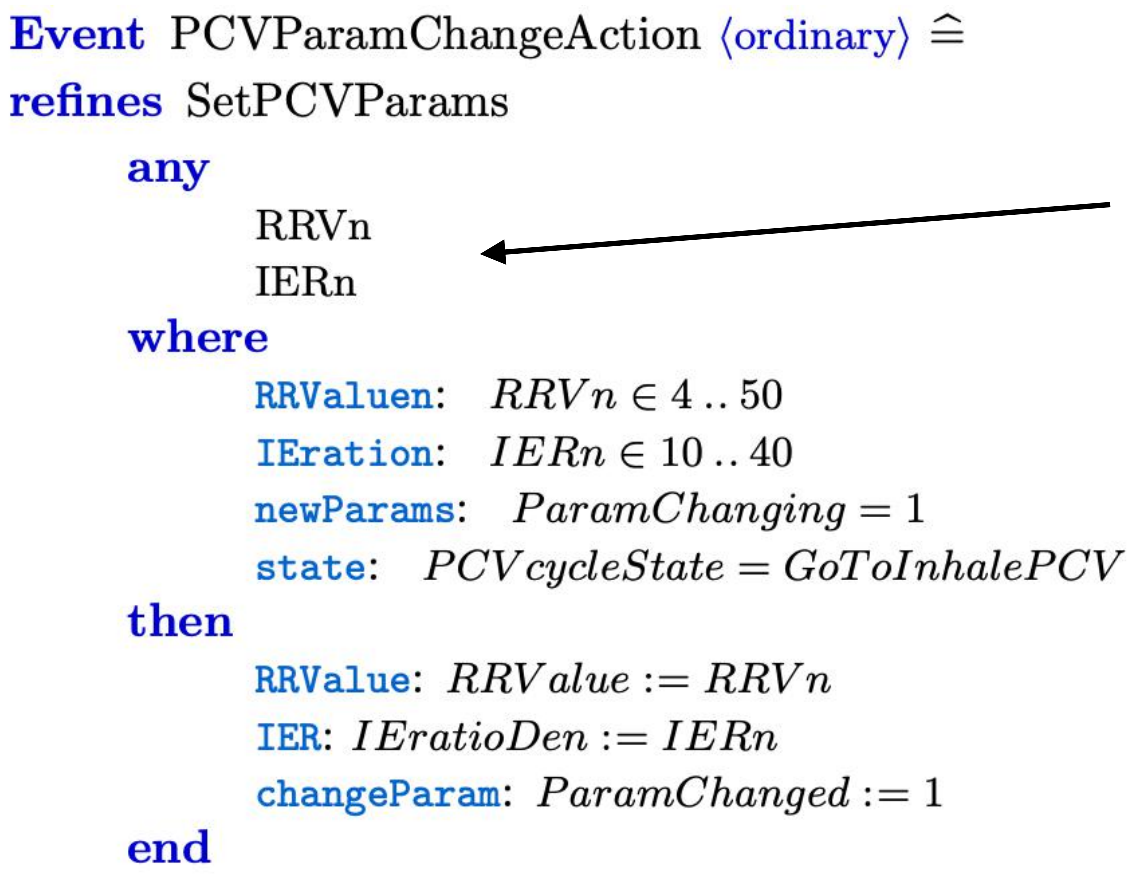


```
Event PCVParamChangeAction ⟨ordinary⟩ $\widehat{=}$
refines SetPCVParams
    any
        RRVn
        IERn
    where
        RRValuen:  $RRVn \in 4 \,..\, 50$
        IEration:  $IERn \in 10 \,..\, 40$
        newParams:  $ParamChanging = 1$
        state:  $PCVcycleState = GoToInhalePCV$
    then
        RRValue: $RRValue := RRVn$
        IER: $IEratioDen := IERn$
        changeParam: $ParamChanged := 1$
    end
```

Motivation

Event-B

- Event-B model: state + guarded transition



```
Event PCVParamChangeAction ⟨ordinary⟩ ≙
refines SetPCVParams
      any
            RRVn
            IERn
      where
            RRValuen:   RRVn ∈ 4 .. 50
            IEration:   IERn ∈ 10 .. 40
            newParams:   ParamChanging = 1
            state:   PCVcycleState = GoToInhalePCV
      then
            RRValue: RRValue := RRVn
            IER: IEratioDen := IERn
            changeParam: ParamChanged := 1
      end
```

Where the request to external values lies

Motivation

Validation

- How to validate the models?
  - Individually? not much point
  - together? we need to make them communicate!

- We need some protocol

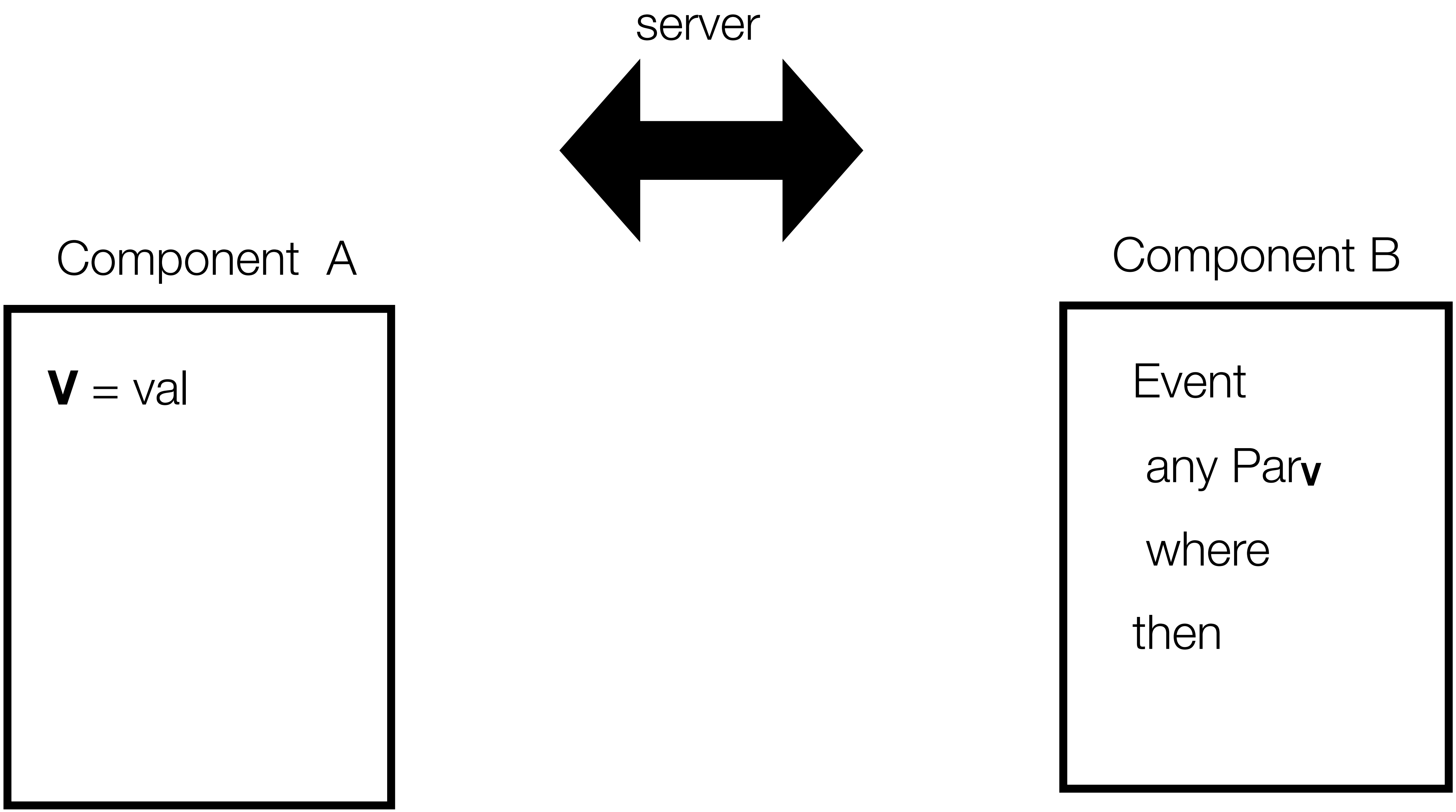
Protocol

Requirements

- No formal description of the architecture
- Strong opacity & loose coupling of components
- Compatibility with Event-B operational semantics
  - non determinism
  - no spurious modification of states
- Compatibility with existing tools
  - JeB in particular: no modification of the core

Protocol

Principles



server

Component A

V = val

Component B

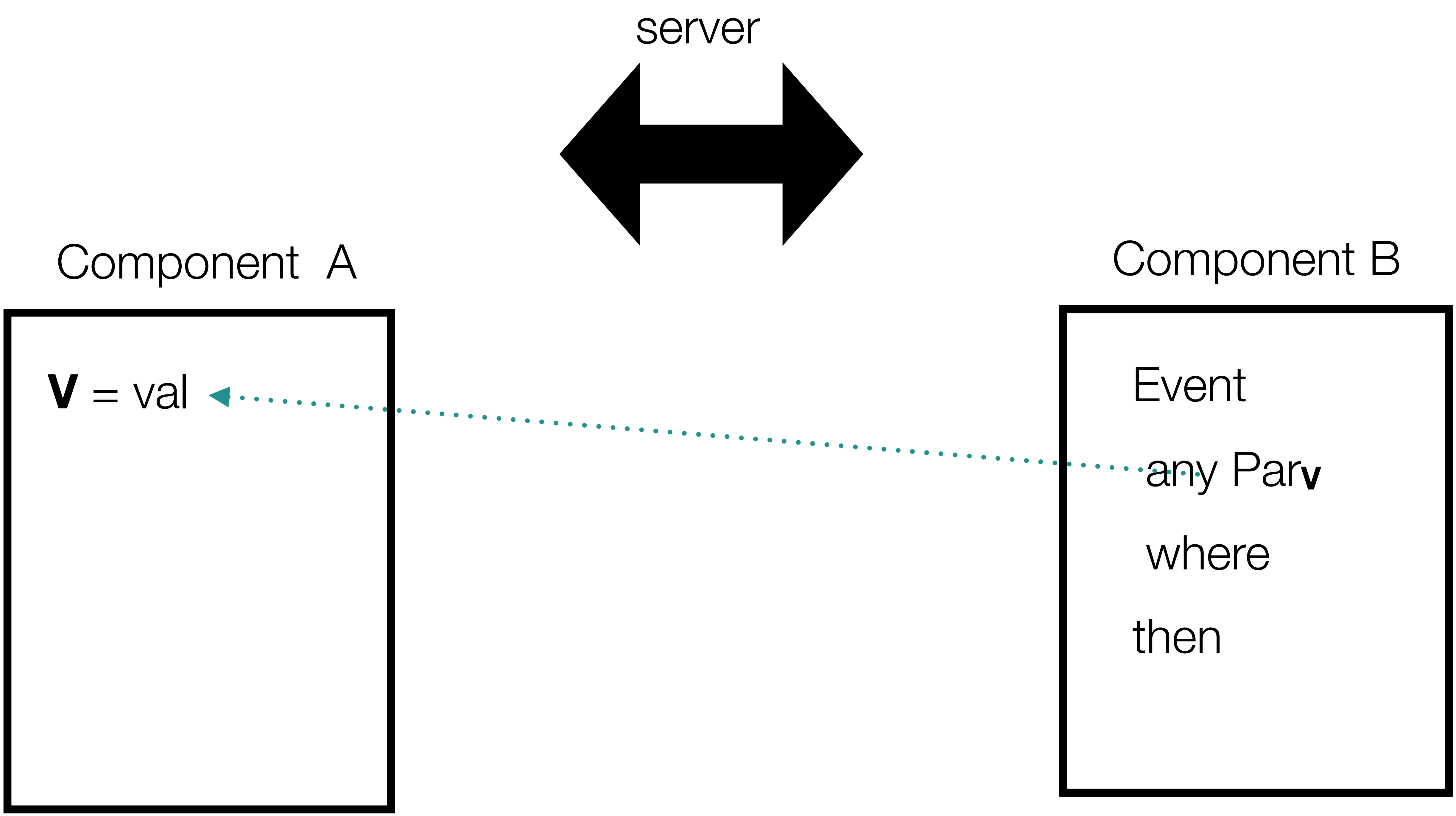
Event

any Par**v**

where

then

Protocol Principles



server

Component A

V = val

Component B

Event

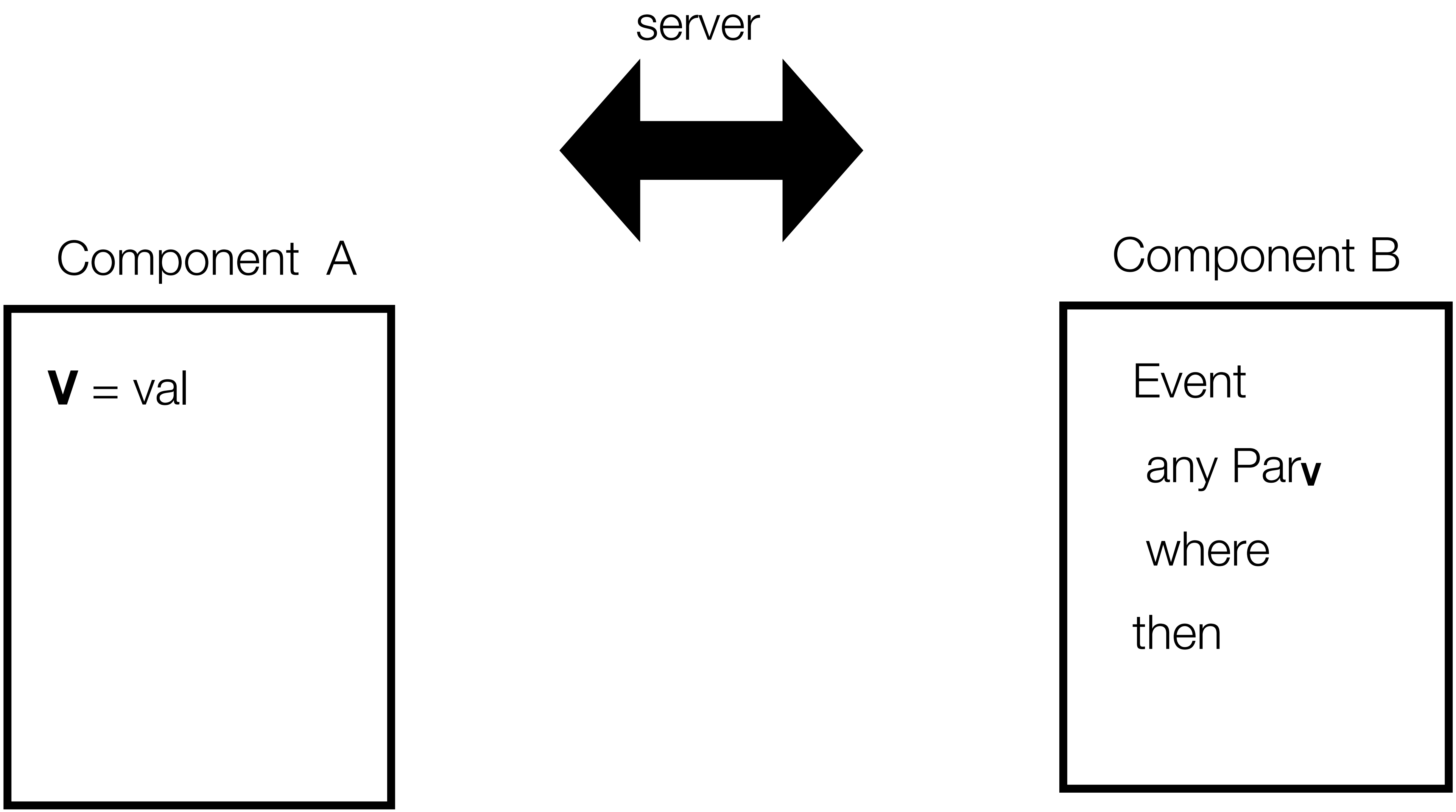
any Par**v**

where

then

Protocol

Principles



server

Component A

V = val

Component B

Event

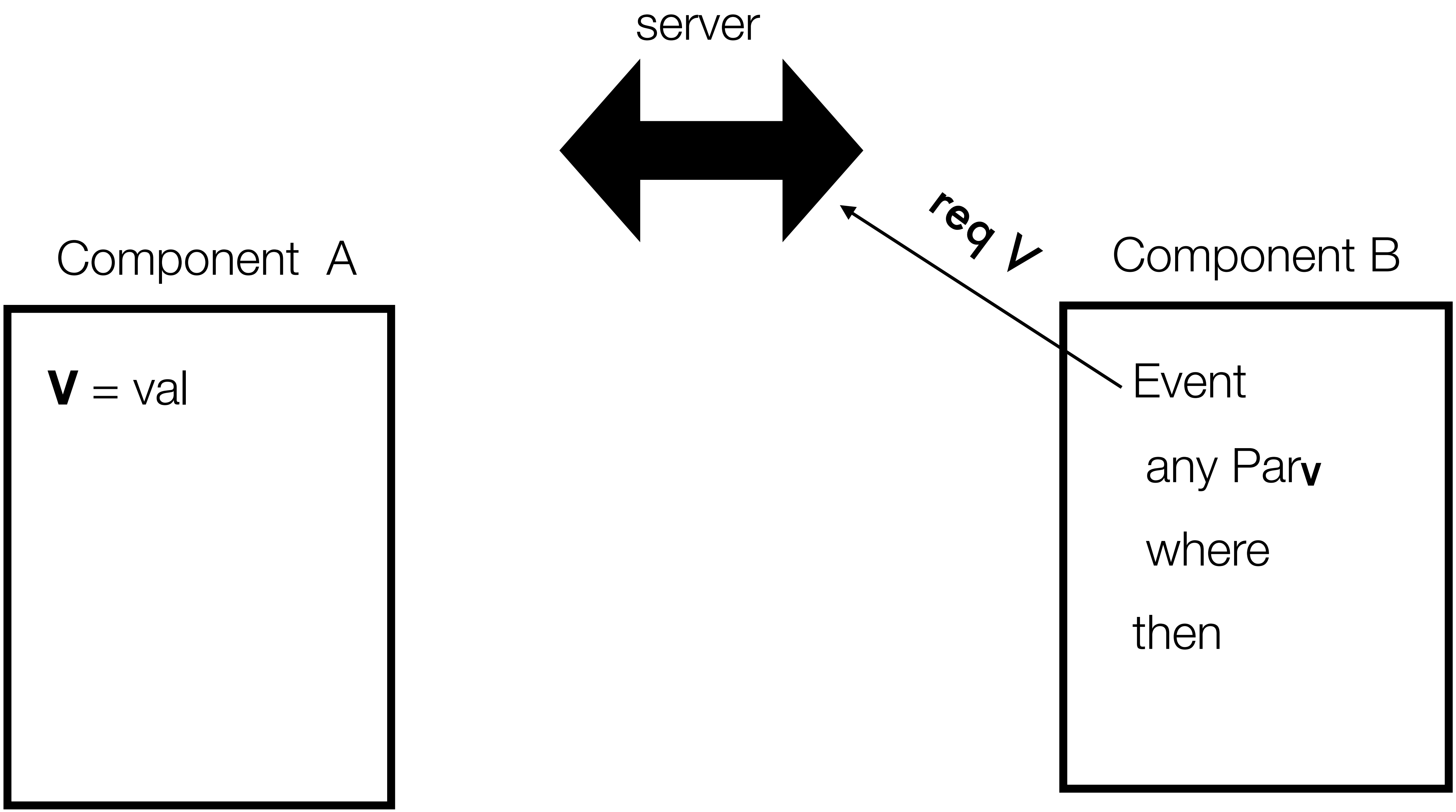
any Par**v**

where

then

Protocol

Principles



server

req V

Component A

V = val

Component B

Event

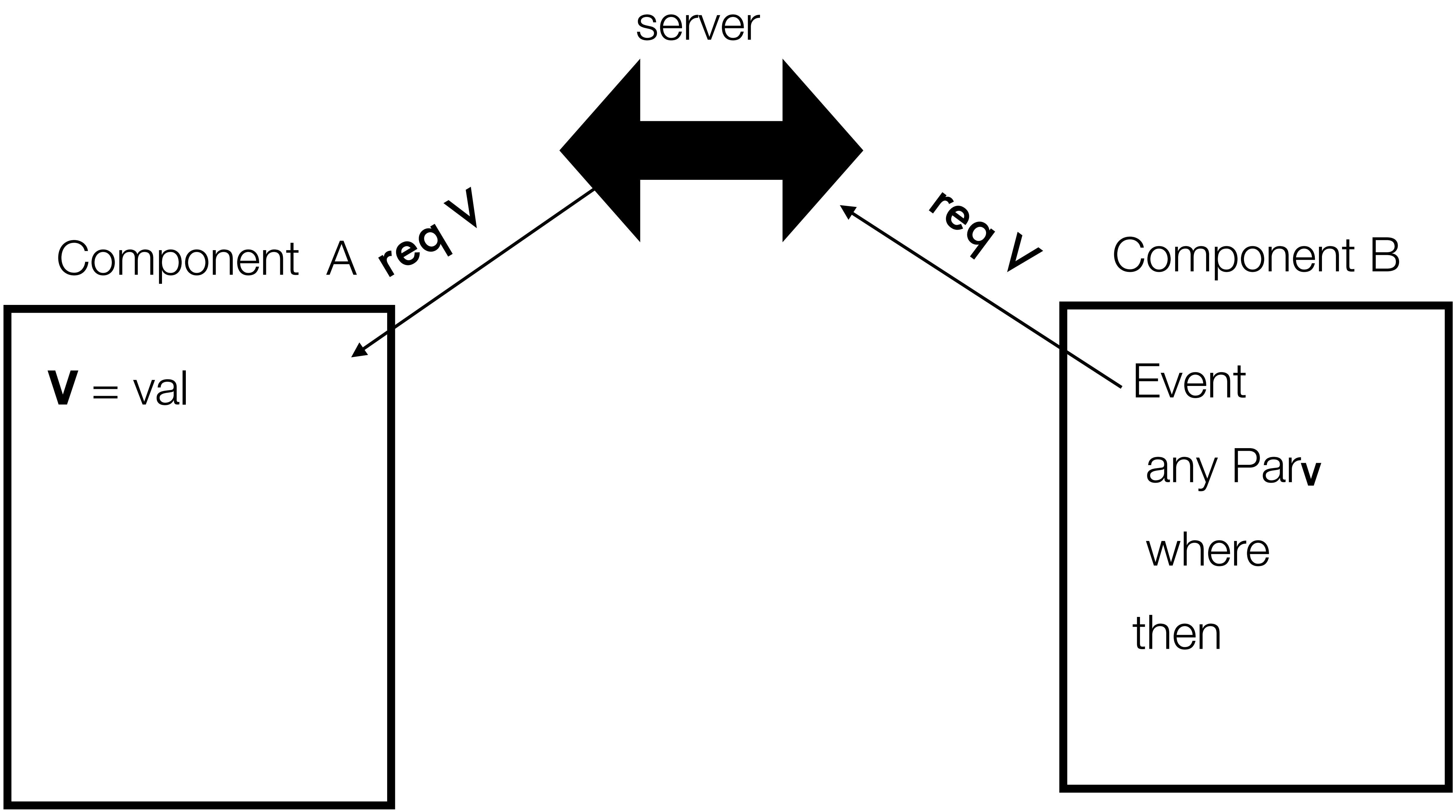
any Par**v**

where

then

Protocol

Principles



server

Component A

req V

V = val

req V

Component B

Event

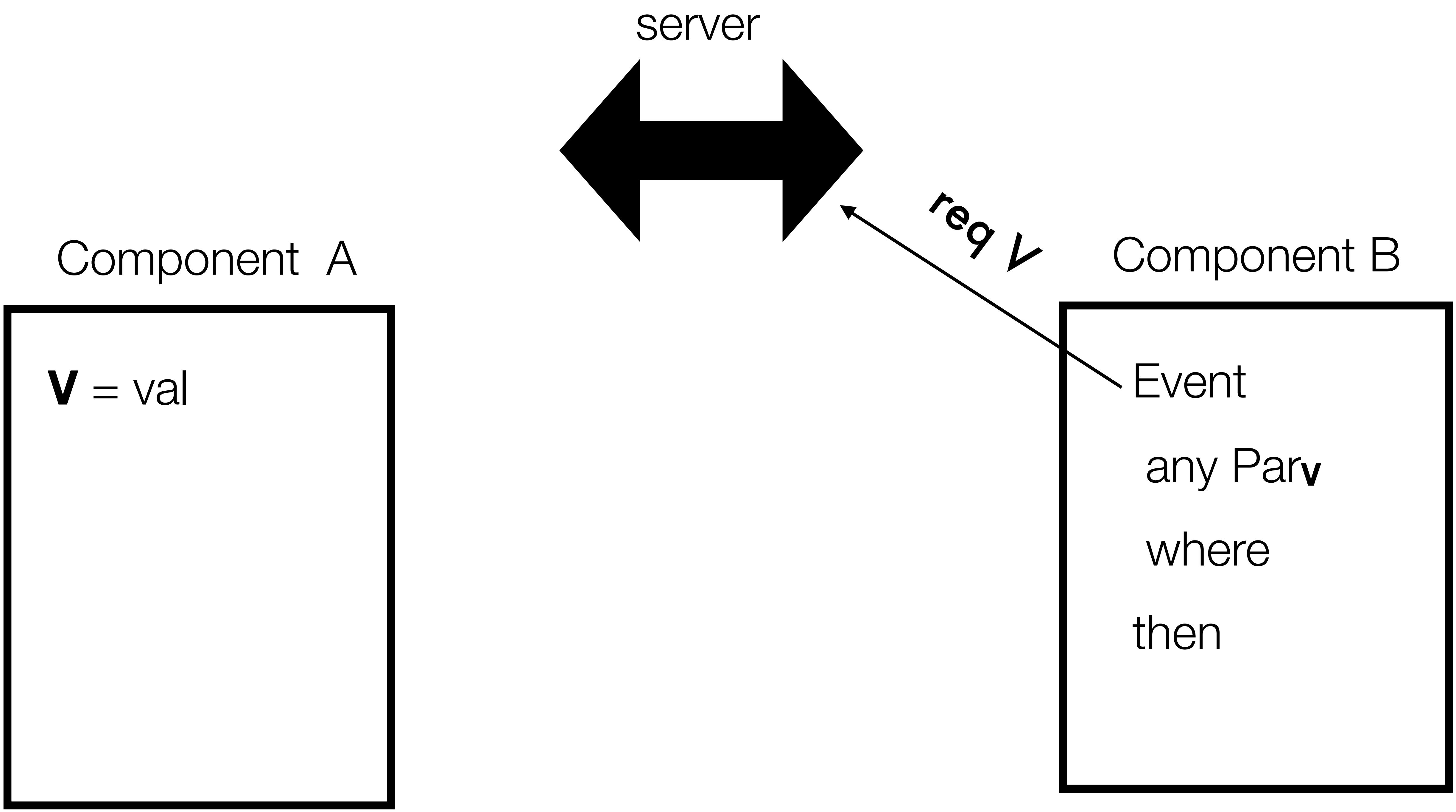
any Par**v**

where

then

Protocol

Principles



server

req V

Component A

V = val

Component B

Event

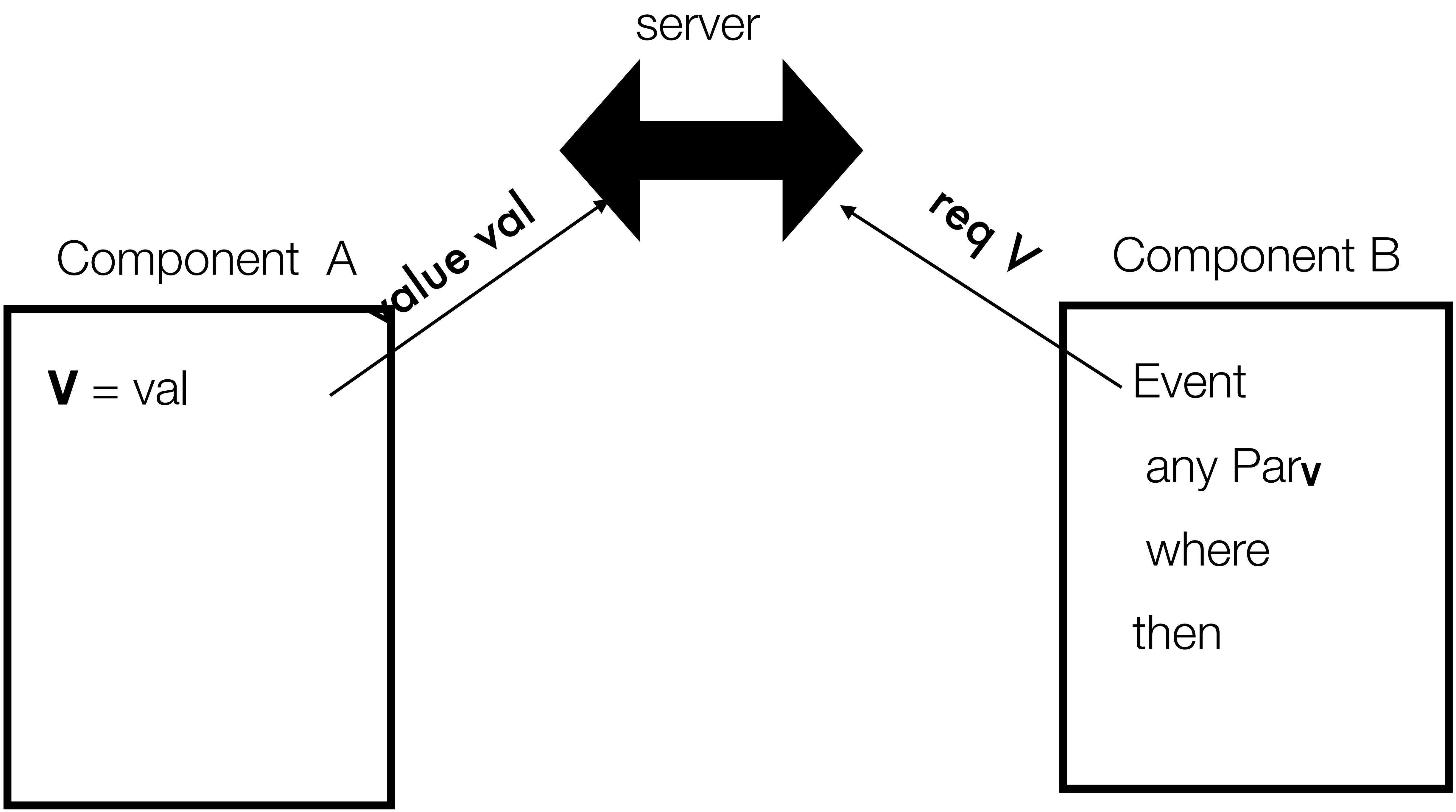
any Par$_{\mathbf{v}}$

where

then

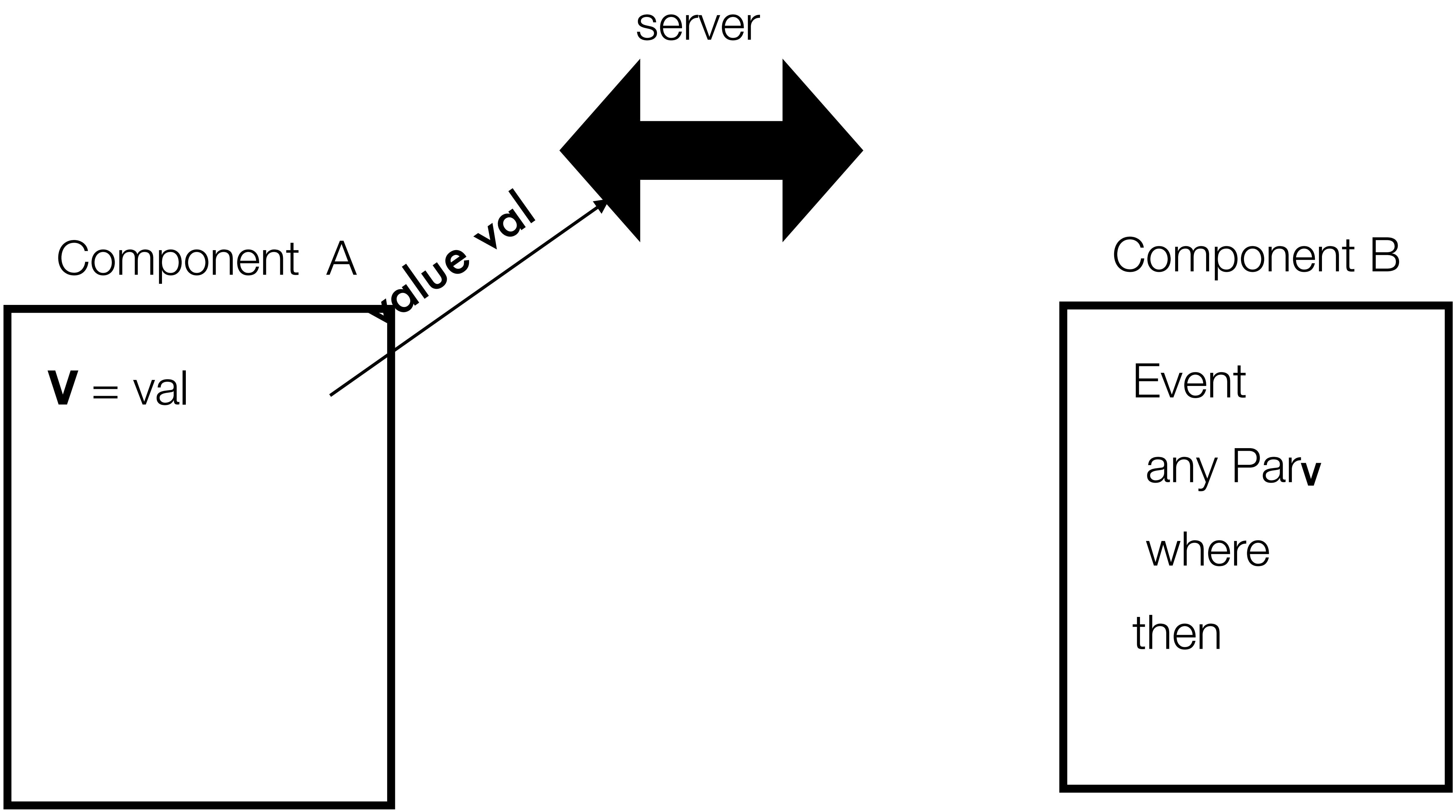
Protocol

Principles



Protocol

Principles



server

Component A

value val

V = val

Component B

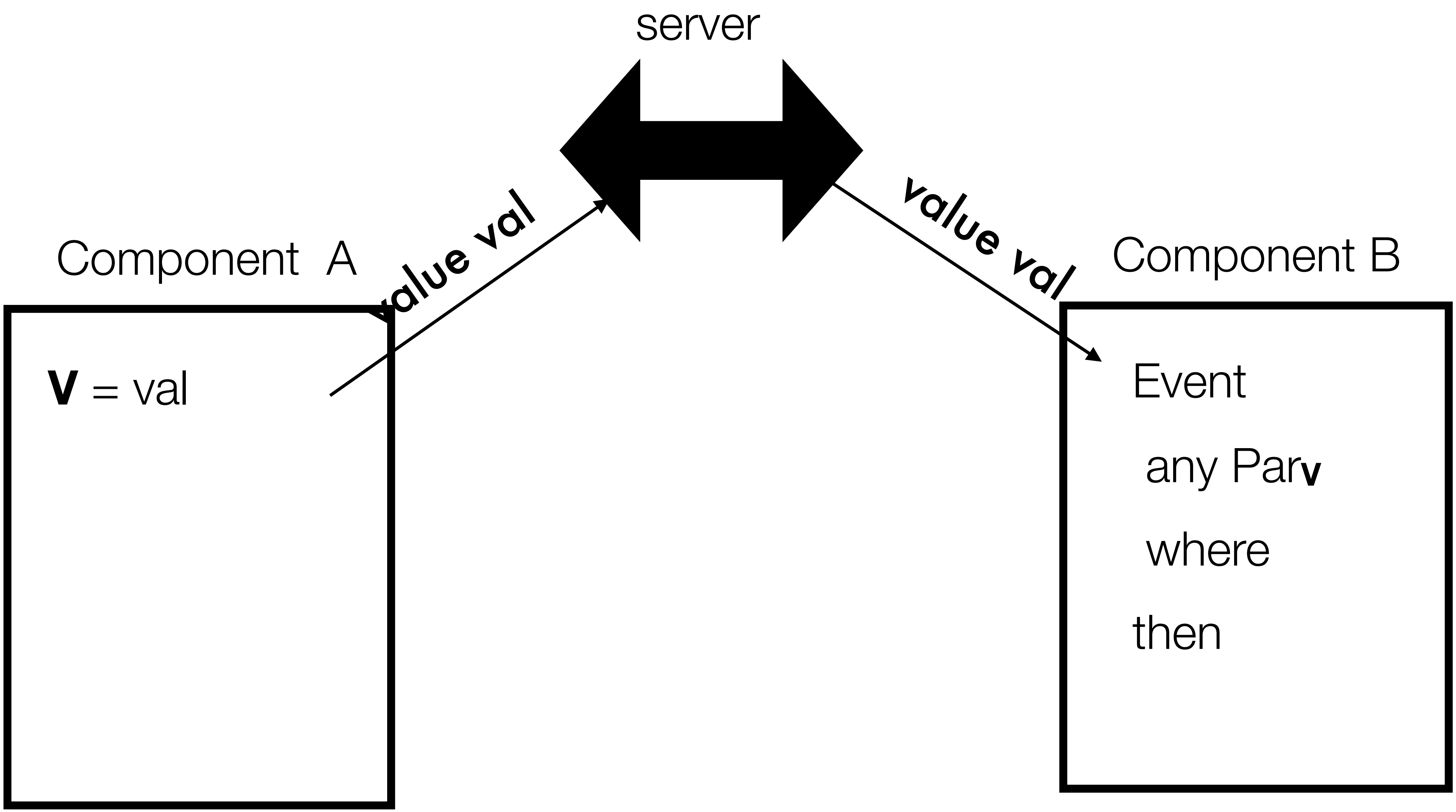
Event

any Par**v**

where

then

Protocol Principles



server

Component A

V = val

value val

value val

Component B

Event

any Par**v**

where

then

Protocole

Principles (2)

- Initialisation
  - `InitSimulation`: initialise the server
  - `register <v> <mg>`: register a variable `v` managed by model `mg`
- Requests
  - `request <v> <mr>` : model `mr` requests value of variable `v`

    the server forwards to the model managing `v`
  - `value <v> <val>` : model `mg` returns the value of `v`
    - the server forwards to the model requesting `v`

Protocole

Principles (3)

- Implicit architecture
  - the server forwards the request
- Added message
  - `update <v>` : emitted when a exported value is modified
  - comes from the non-determinism of Event-B
    - no guarantees the different parameters will be available at the same time
    - a need to inform when a requested value may be obsolete

Implantation

Executable models

- Two (strong) requirements :
  - no modification of JeB!
    - should only rely of the ability to (safely) add manually pieces of code
  - total conformity to the operational semantics
    - asynchronous and non-deterministic
    - strict separation of the model states

      no side-effect!

Implantation

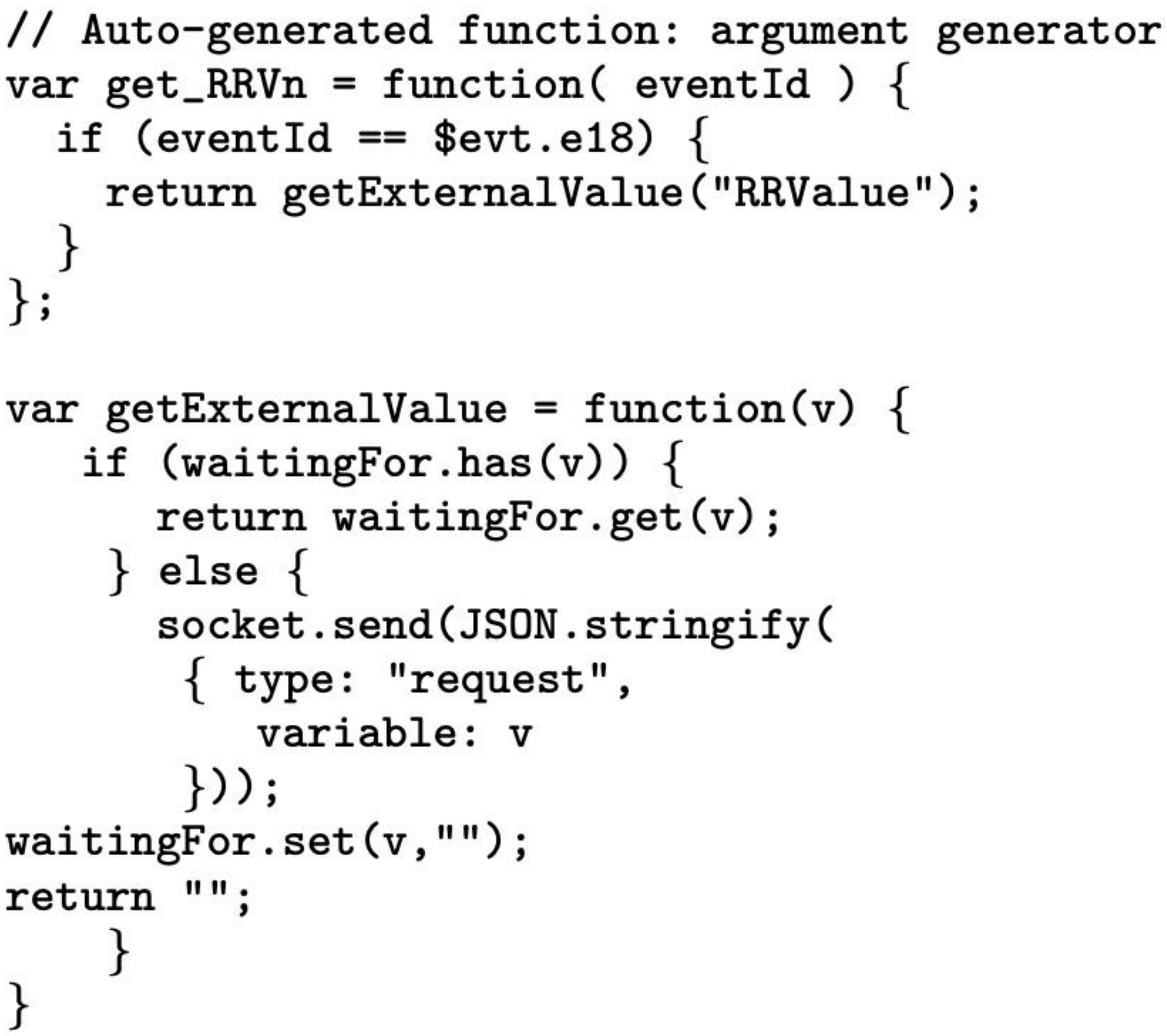
External values

- Paramètre d’un événement !

Implantation

External values

- Paramètre d’un événement !



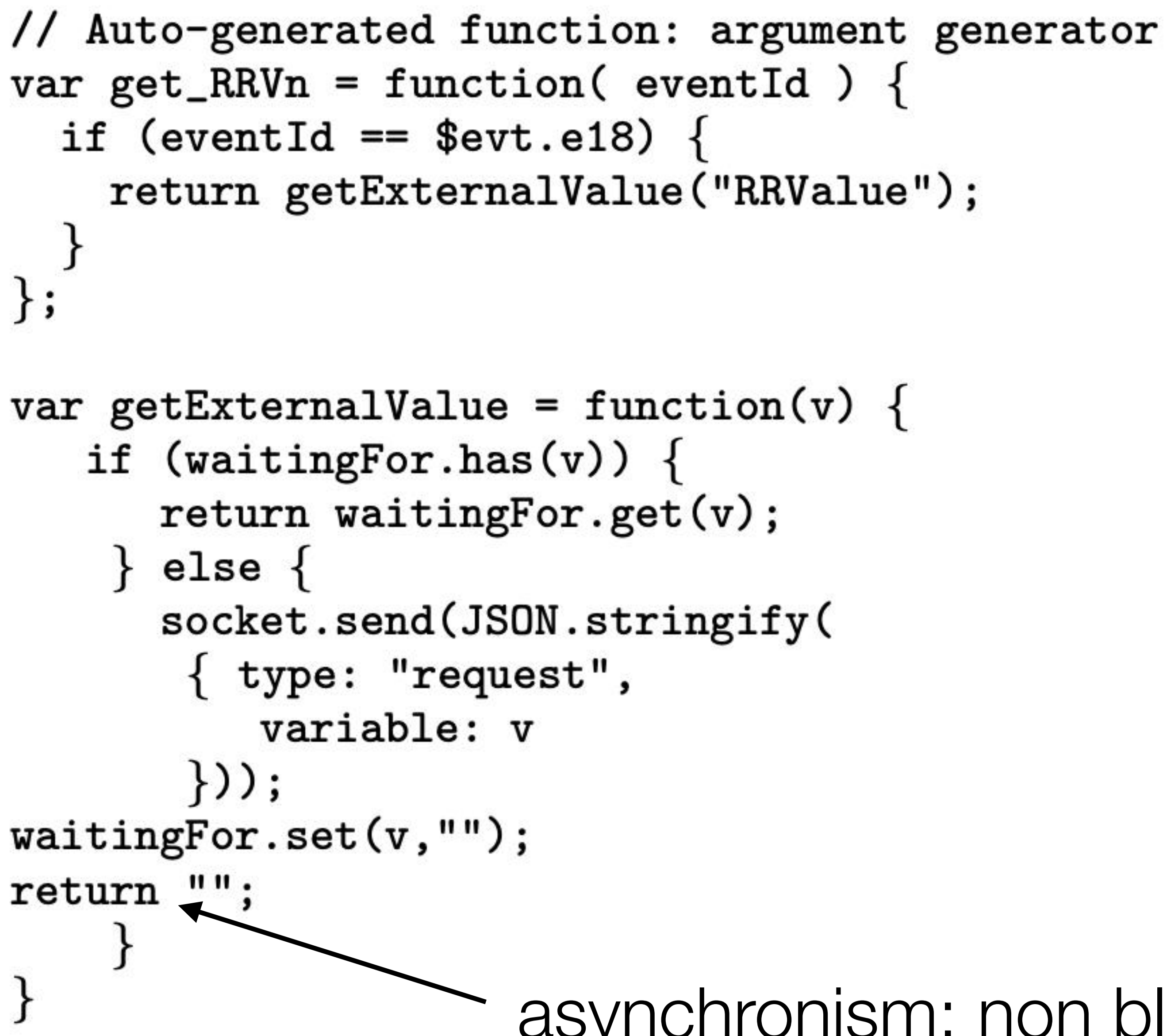
```
// Auto-generated function: argument generator
var get_RRVn = function( eventId ) {
  if (eventId == $evt.e18) {
    return getExternalValue("RRValue");
  }
};

var getExternalValue = function(v) {
   if (waitingFor.has(v)) {
      return waitingFor.get(v);
    } else {
      socket.send(JSON.stringify(
       { type: "request",
          variable: v
       }));
waitingFor.set(v,"");
return "";
    }
}
```

Implantation

External values

- Paramètre d’un événement !



```
// Auto-generated function: argument generator
var get_RRVn = function( eventId ) {
  if (eventId == $evt.e18) {
    return getExternalValue("RRValue");
  }
};

var getExternalValue = function(v) {
   if (waitingFor.has(v)) {
      return waitingFor.get(v);
     } else {
      socket.send(JSON.stringify(
       { type: "request",
          variable: v
       }));
waitingFor.set(v,"");
return "";
     }
}
```

asynchronism: non blocking return

Implantation

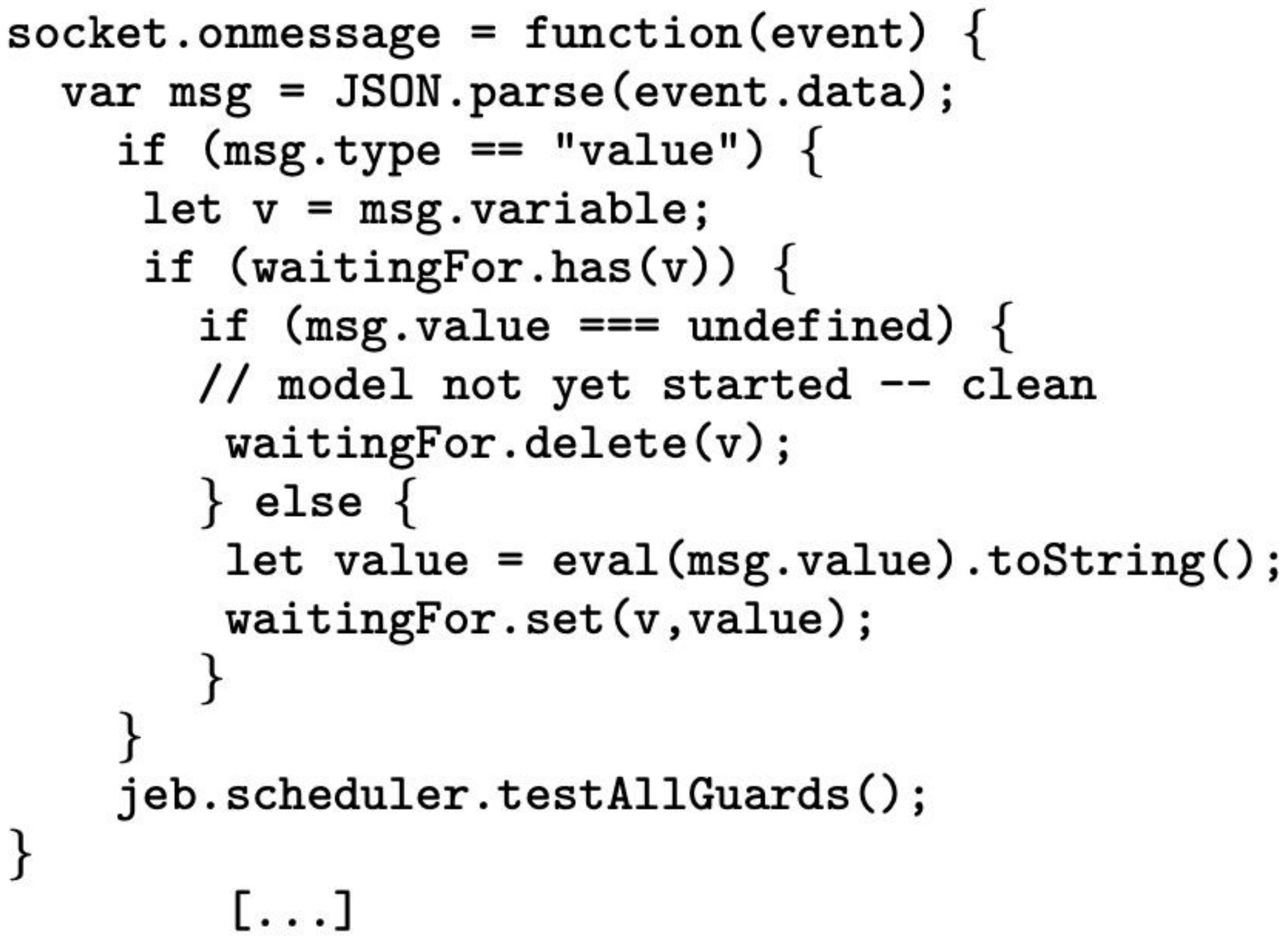
External values

- Paramètre d’un événement !

Implantation

External values

- Paramètre d’un événement !

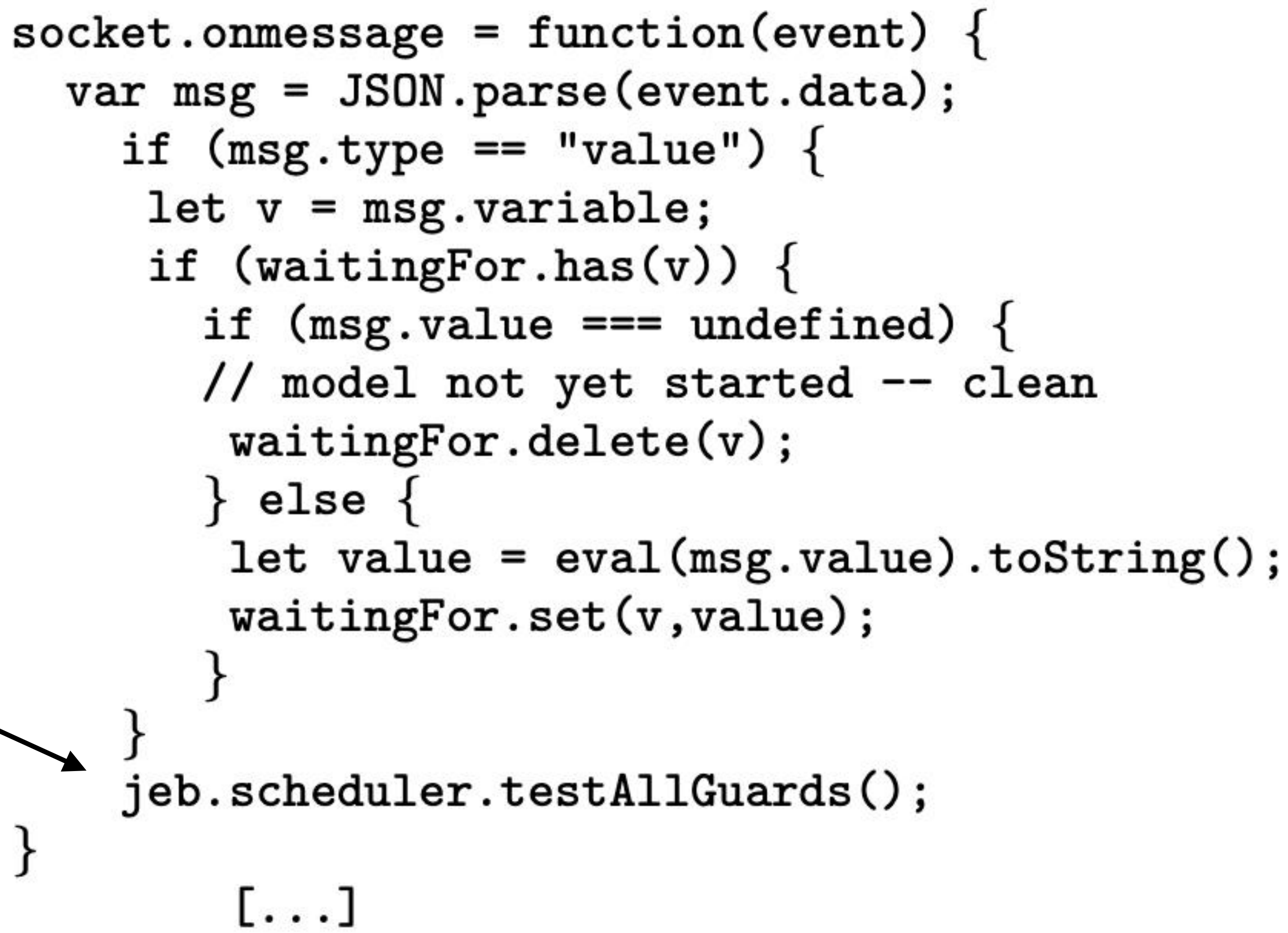


```
socket.onmessage = function(event) {
  var msg = JSON.parse(event.data);
    if (msg.type == "value") {
     let v = msg.variable;
     if (waitingFor.has(v)) {
       if (msg.value === undefined) {
       // model not yet started -- clean
        waitingFor.delete(v);
       } else {
        let value = eval(msg.value).toString();
        waitingFor.set(v,value);
       }
    }
    jeb.scheduler.testAllGuards();
}
        [...]
```

Implantation

External values

- Paramètre d’un événement !



```
socket.onmessage = function(event) {
  var msg = JSON.parse(event.data);
    if (msg.type == "value") {
     let v = msg.variable;
     if (waitingFor.has(v)) {
       if (msg.value === undefined) {
       // model not yet started -- clean
        waitingFor.delete(v);
       } else {
        let value = eval(msg.value).toString();
        waitingFor.set(v,value);
       }
    }
    jeb.scheduler.testAllGuards();
}
        [...]
```

new evaluation cycle

Implantation

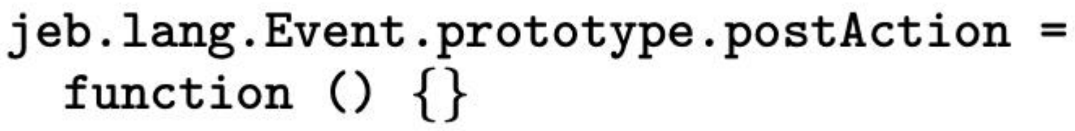
Consumption of values

- Several parameters => several receptions
    - consumption: only when the event is triggered
        - introduction of PostActions

Implantation

Consumption of values

- Several parameters => several receptions
  - consumption: only when the event is triggered
    - introduction of PostActions



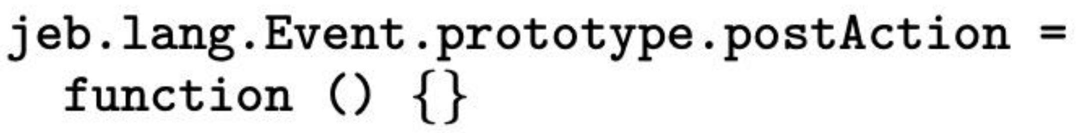
```
jeb.lang.Event.prototype.postAction =
  function () {}
```

new methods for event ``class”

Implantation

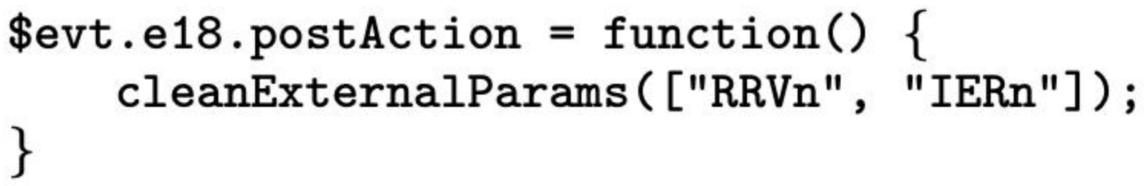
Consumption of values

- Several parameters => several receptions
  - consumption: only when the event is triggered
    - introduction of PostActions



```
jeb.lang.Event.prototype.postAction =
  function () {}
```

new methods for event ``class”



```
$evt.e18.postAction = function() {
    cleanExternalParams(["RRVn", "IERn"]);
}
```

postAction for event

`PCVParamsChangeAction`

Implantation

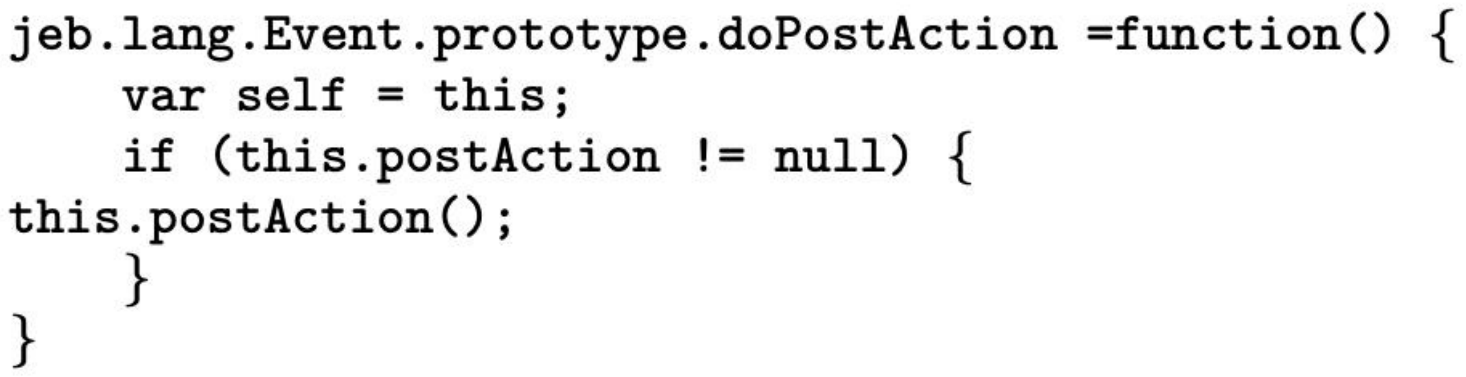
Consumption of values (2)

- Addition in the scheduler

Implantation

Consumption of values (2)

- Addition in the scheduler

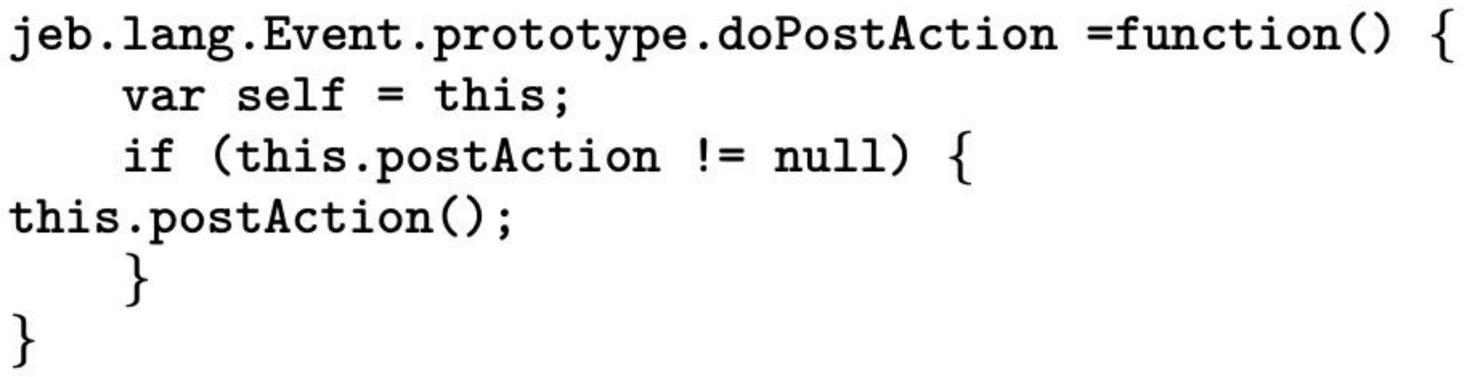


```
jeb.lang.Event.prototype.doPostAction =function() {
    var self = this;
    if (this.postAction != null) {
this.postAction();
    }
}
```

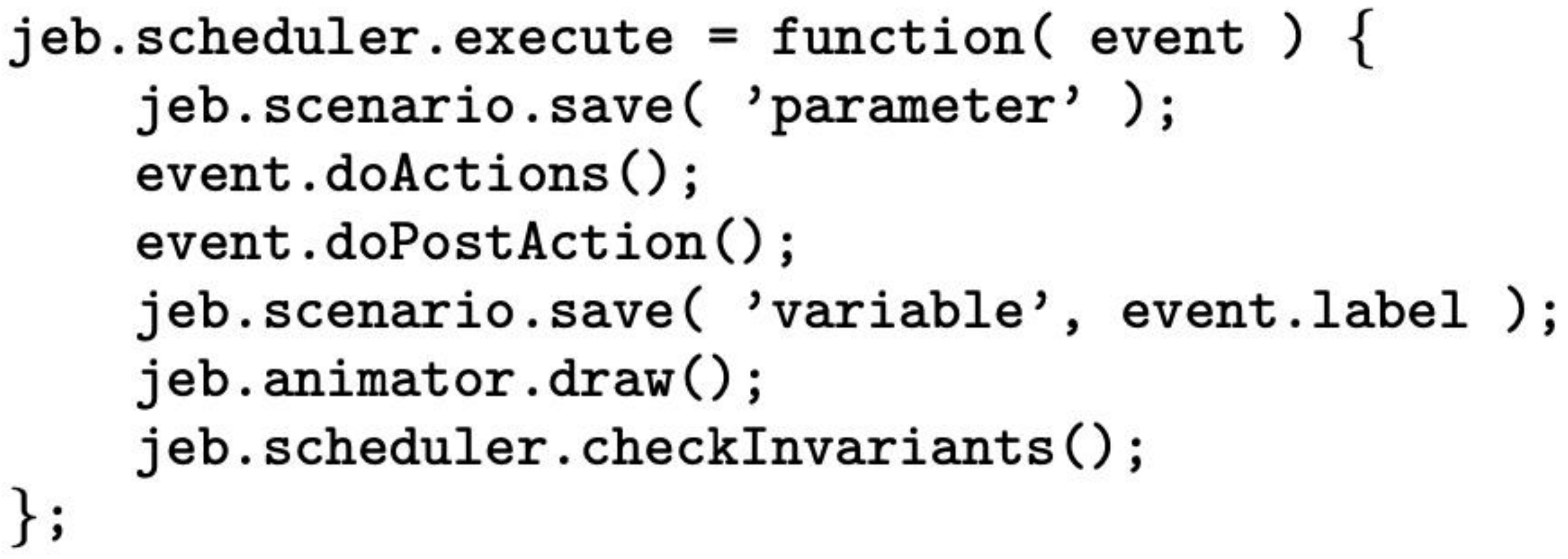
Implantation

Consumption of values (2)

- Addition in the scheduler



```
jeb.lang.Event.prototype.doPostAction =function() {
    var self = this;
    if (this.postAction != null) {
this.postAction();
    }
}
```

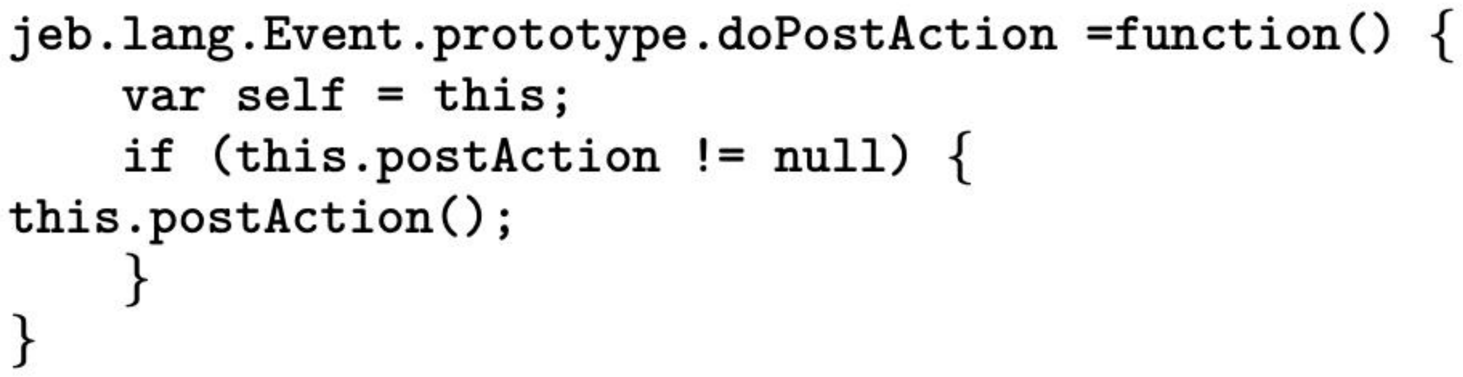


```
jeb.scheduler.execute = function( event ) {
    jeb.scenario.save( 'parameter' );
    event.doActions();
    event.doPostAction();
    jeb.scenario.save( 'variable', event.label );
    jeb.animator.draw();
    jeb.scheduler.checkInvariants();
};
```

Implantation

Consumption of values (2)

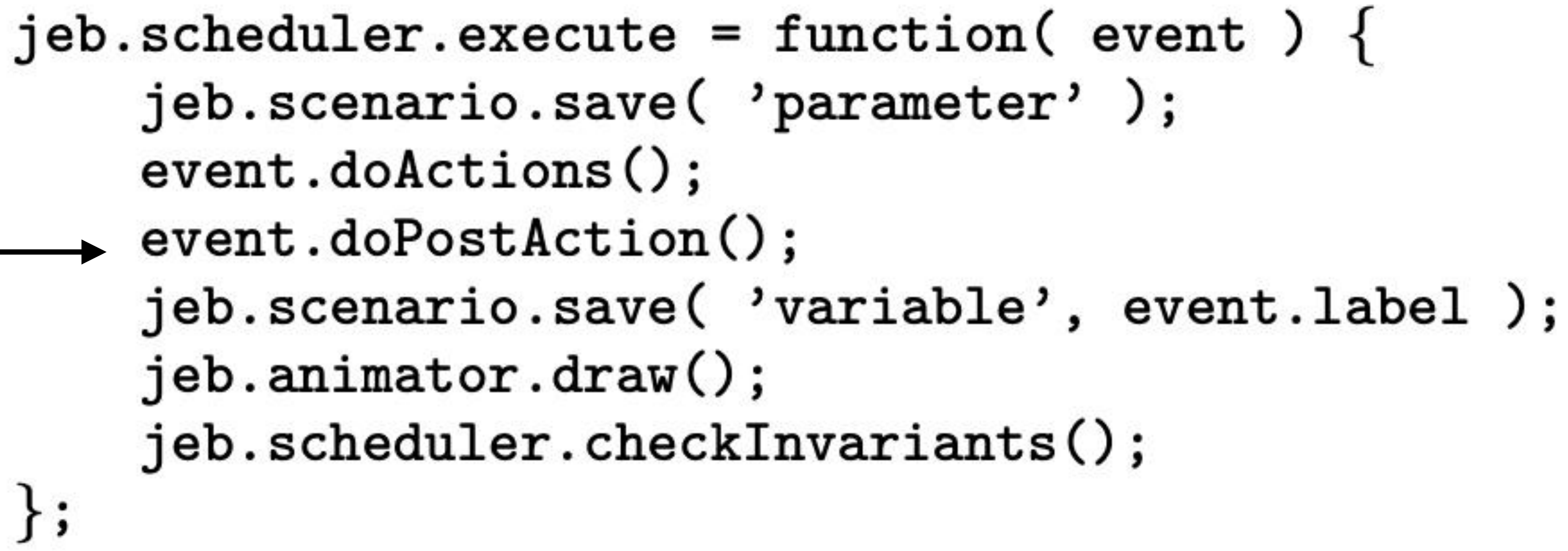
- Addition in the scheduler



```
jeb.lang.Event.prototype.doPostAction =function() {
    var self = this;
    if (this.postAction != null) {
this.postAction();
    }
}
```

add on in the function

which execute events



```
jeb.scheduler.execute = function( event ) {
    jeb.scenario.save( 'parameter' );
    event.doActions();
    event.doPostAction();
    jeb.scenario.save( 'variable', event.label );
    jeb.animator.draw();
    jeb.scheduler.checkInvariants();
};
```

Implantation

Websocket server

- Server on Node.js
    - three structures :
        - whereTo : (Variable -> managing model)
        - requestedBy : (Variable -> requesting model)
        - connectedSockets : ({connected models})
    - 80 lines of simple JavaScript code
        - initialisation
        - forward requests and values

Protocol

Added bonus

- The protocol is not restricted to Event-B models!

Protocol

Added bonus

- The protocol is not restricted to Event-B models!

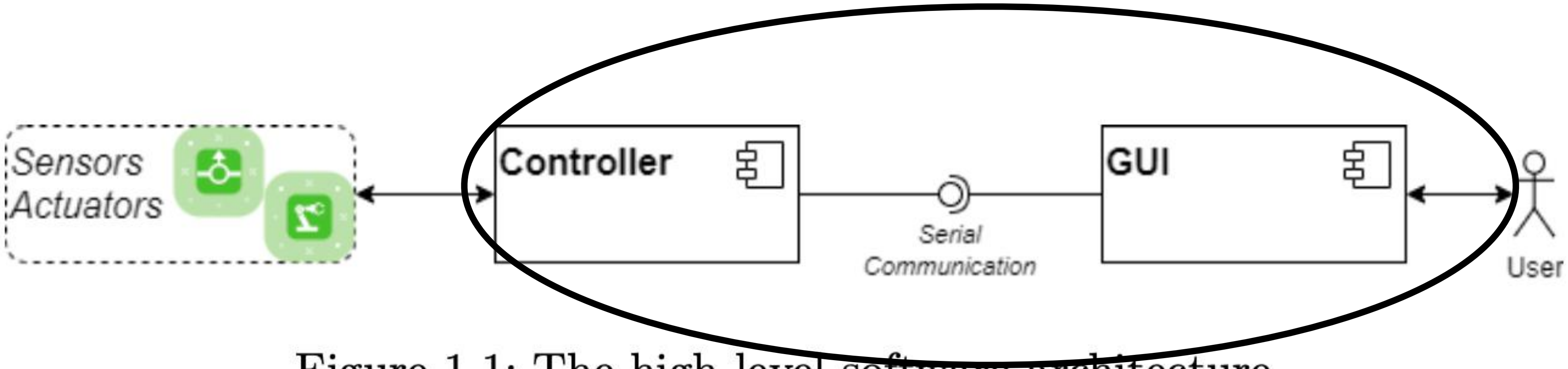


Figure 1.1: The high-level software architecture

Protocol

Added bonus

- The protocol is not restricted to Event-B models!

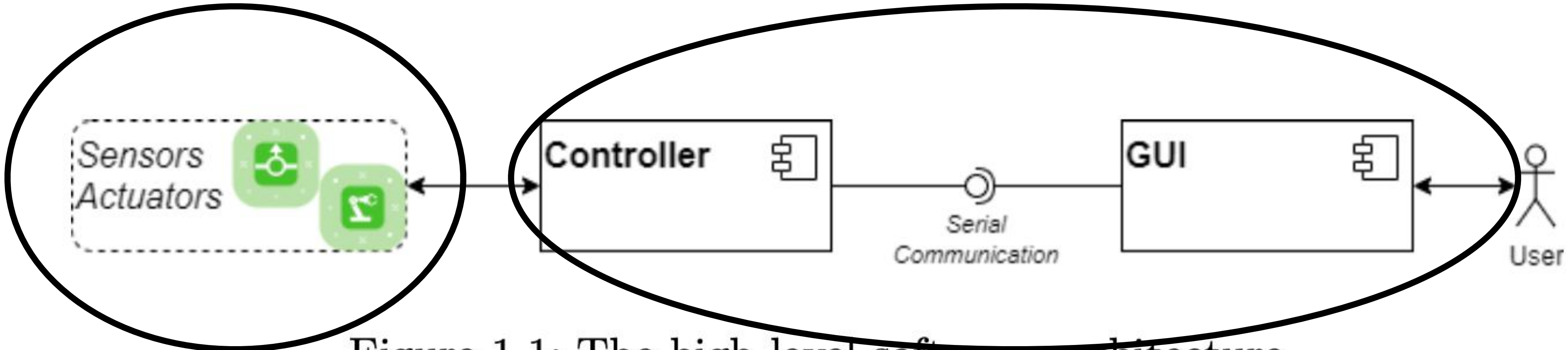


Figure 1.1: The high-level software architecture

Protocol

Added bonus

- The protocol is not restricted to Event-B models!

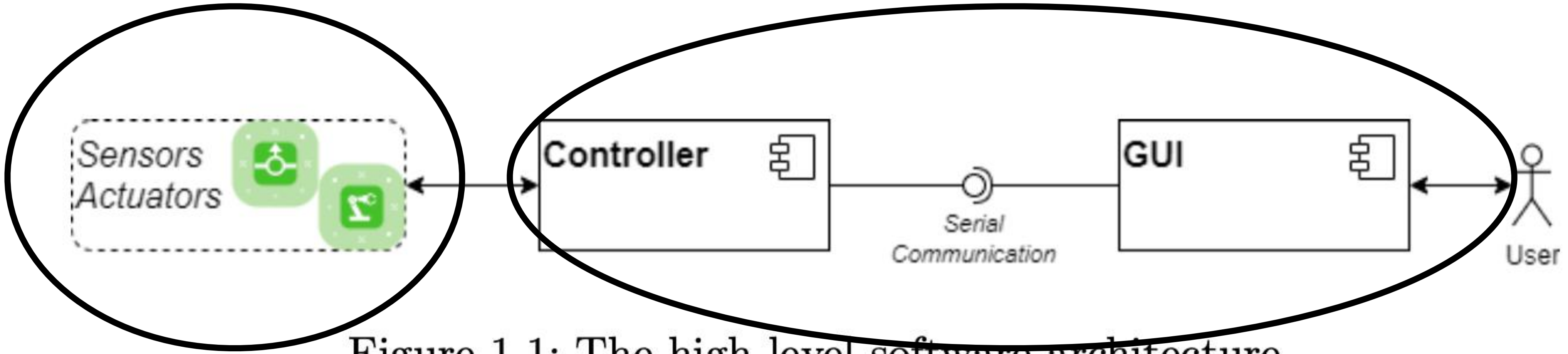
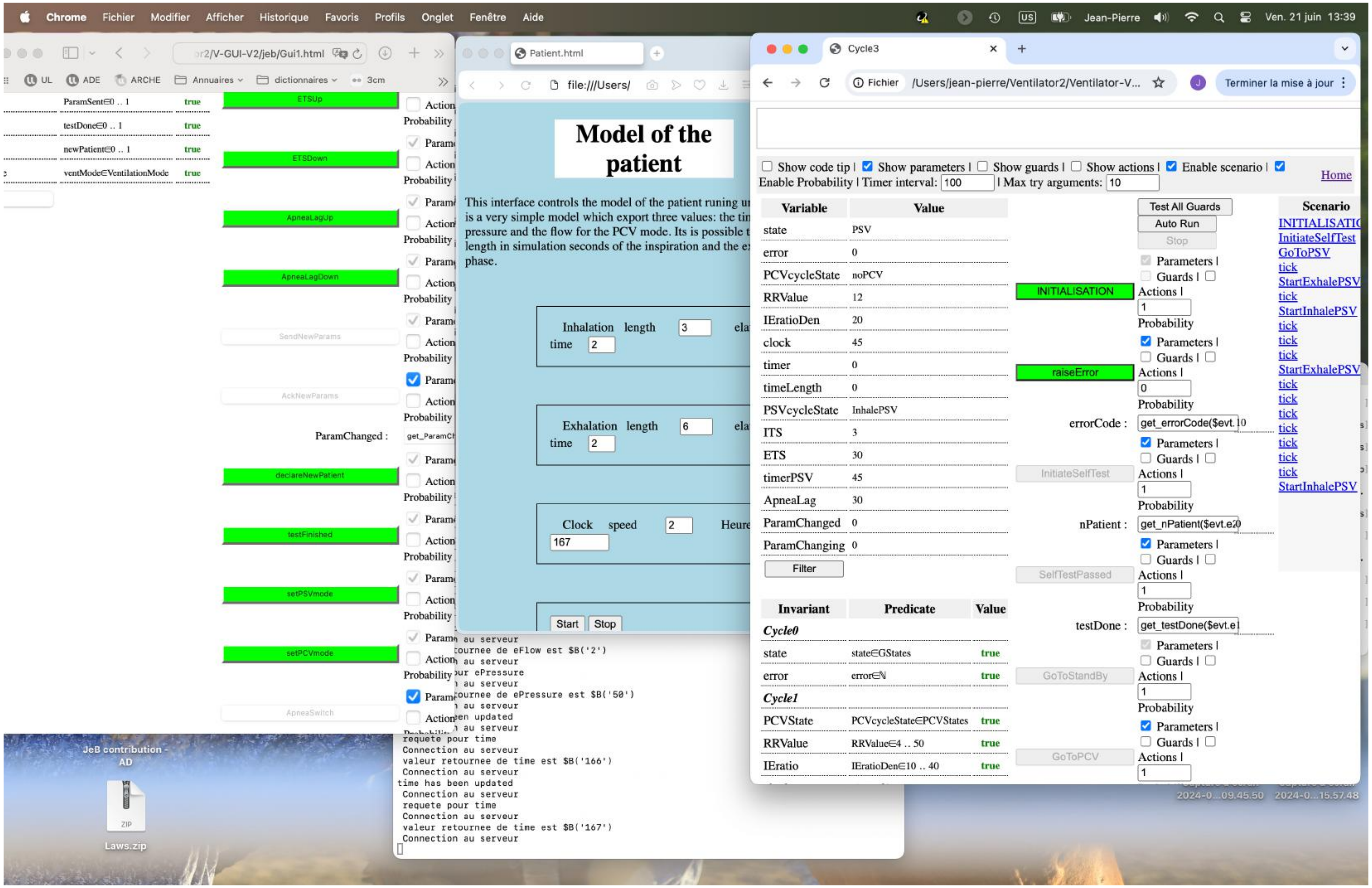


Figure 1.1: The high-level software architecture

- The `continuous” component can use the protocol too!

Implantation

Demo



	ParamSent∈0 .. 1	true
	testDone∈0 .. 1	true
	newPatient∈0 .. 1	true
	ventMode∈VentilationMode	true

ParamChanged : get_ParamCh

Model of the patient

This interface controls the model of the patient runing u[…] is a very simple model which export three values: the tin[…] pressure and the flow for the PCV mode. Its is possible t[…] length in simulation seconds of the inspiration and the e[…] phase.

Inhalation length 3 ela[…] time 2

Exhalation length 6 ela[…] time 2

Clock speed 2 Heure 167

```
         au serveur
        tournee de eFlow est $B('2')
         au serveur
        pur ePressure
         au serveur
        tournee de ePressure est $B('50')
         au serveur
        en updated
         au serveur
 requete pour time
 Connection au serveur
 valeur retournee de time est $B('166')
 Connection au serveur
time has been updated
 Connection au serveur
 requete pour time
 Connection au serveur
 valeur retournee de time est $B('167')
 Connection au serveur
```

☐ Show code tip | ☑ Show parameters | ☐ Show guards | ☐ Show actions | ☑ Enable scenario | ☑ Enable Probability | Timer interval: 100 | Max try arguments: 10

Variable	Value
state	PSV
error	0
PCVcycleState	noPCV
RRValue	12
IEratioDen	20
clock	45
timer	0
timeLength	0
PSVcycleState	InhalePSV
ITS	3
ETS	30
timerPSV	45
ApneaLag	30
ParamChanged	0
ParamChanging	0

Invariant	Predicate	Value
Cycle0		
state	state∈GStates	true
error	error∈ℕ	true
Cycle1		
PCVState	PCVcycleState∈PCVStates	true
RRValue	RRValue∈4 .. 50	true
IEratio	IEratioDen∈10 .. 40	true

INITIALISATION: ☑ Parameters | ☐ Guards | ☐ Actions | 1 Probability

raiseError: ☑ Parameters | ☐ Guards | ☐ Actions | 0 Probability
errorCode : get_errorCode($evt. 10

InitiateSelfTest: ☑ Parameters | ☐ Guards | ☐ Actions | 1 Probability
nPatient : get_nPatient($evt.e 0

SelfTestPassed: ☑ Parameters | ☐ Guards | ☐ Actions | 1 Probability
testDone : get_testDone($evt.e 1

GoToStandBy: ☑ Parameters | ☐ Guards | ☐ Actions | 1 Probability

GoToPCV: ☑ Parameters | ☐ Guards | ☐ Actions | 1

Scenario
INITIALISATIC
InitiateSelfTest
GoToPSV
tick
StartExhalePSV
tick
StartInhalePSV
tick
tick
tick
StartExhalePSV
tick
tick
tick
tick
tick
tick
tick
StartInhalePSV

Conclusion

Results

- Work on the exemple
  - with a (ultra-simplified) continuous model
- No modifications of JeB generation process
  - straight use of the get-parameter functions
  - extension of the event prototype
    - no modification of the semantics (so long as added fonctions do not mess with the state!)

Conclusion

Future works

- Scalability testing and performance improvement
  - optimisation of the communications (caching, grouping, …)
  - assessment of the scalability (how many models?)
- Formal assessment of compatibility with Event-B semantics
  - not mush risk with state modification
  - Slightly more concerned about non-determinism and asynchronous
    - no hidden condensing effect?

Conclusion

Future works (2)

- Verification and characterisation of the ``fidelity” property
  - does the observed behaviours conform to the specified ones?

- Possibility to verify the assembly
  - suggestion (O. Kouchnarenko): relating this to CSP-B ideas and formal framework
    - making this a valid formal strategy!

UNIVERSITÉ DE LORRAINE

Loria Laboratoire lorrain de recherche en informatique et ses applications

Thanks for your attention

Questions?