

# Towards a Verified Component Platform

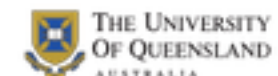
Matthew Fernandez, Ihor Kuz, Gerwin Klein, June Andronick

November 2013



**Australian Government**

## NICTA Funding and Supporting Members and Partners



# Trustworthy Systems

---



# Trustworthy Systems

---



# Trustworthy Systems

---

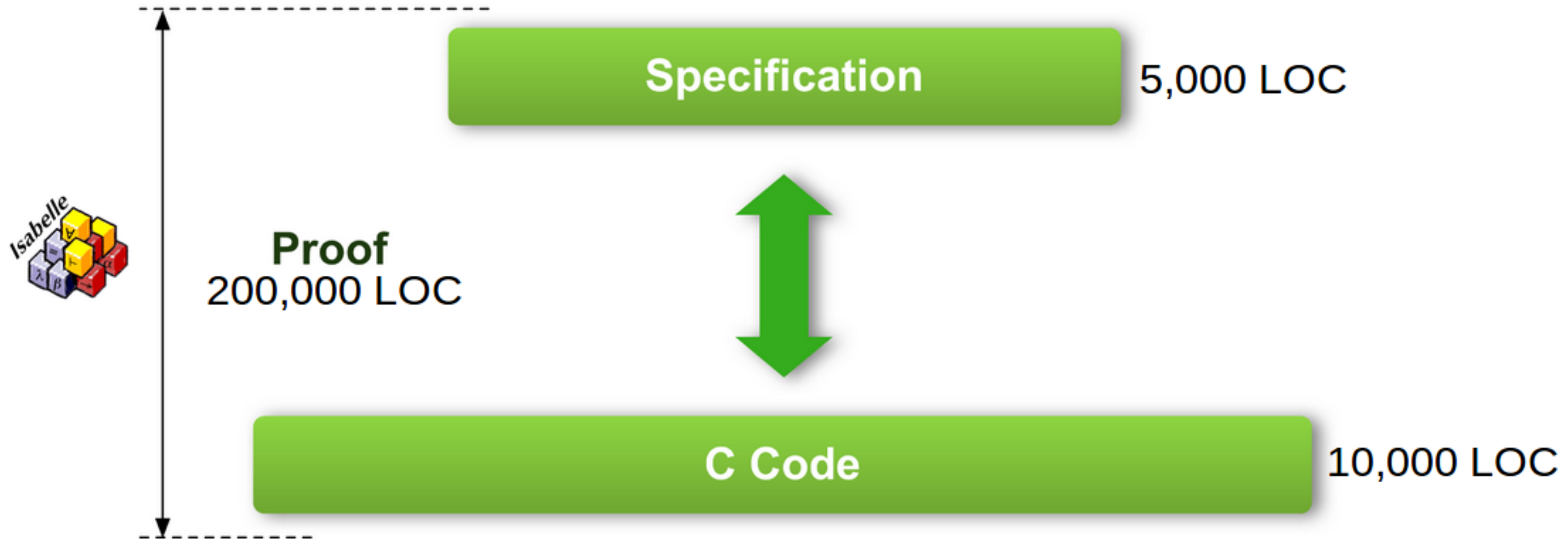
- Development processes
- Testing
- Configuration management
- Certification
- Formal verification



# The L4.verified Project



25 person years



# The L4.verified Project



25 person years

Specification

5,000 LOC



C Code

10,000 LOC



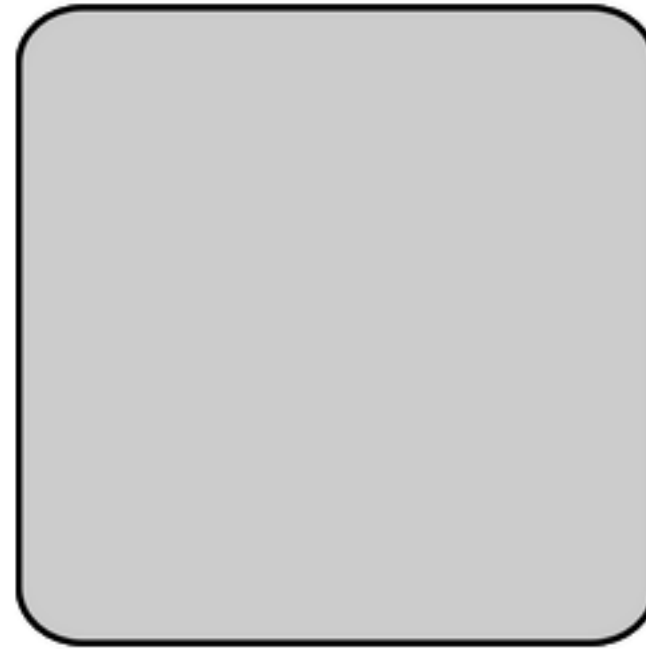
?

NICTA

1,000,000 LOC

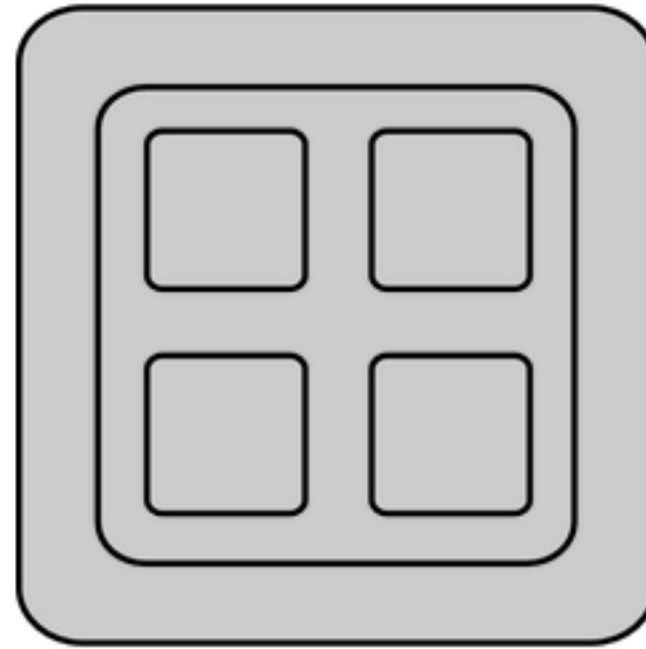
# Component-Based Development

---



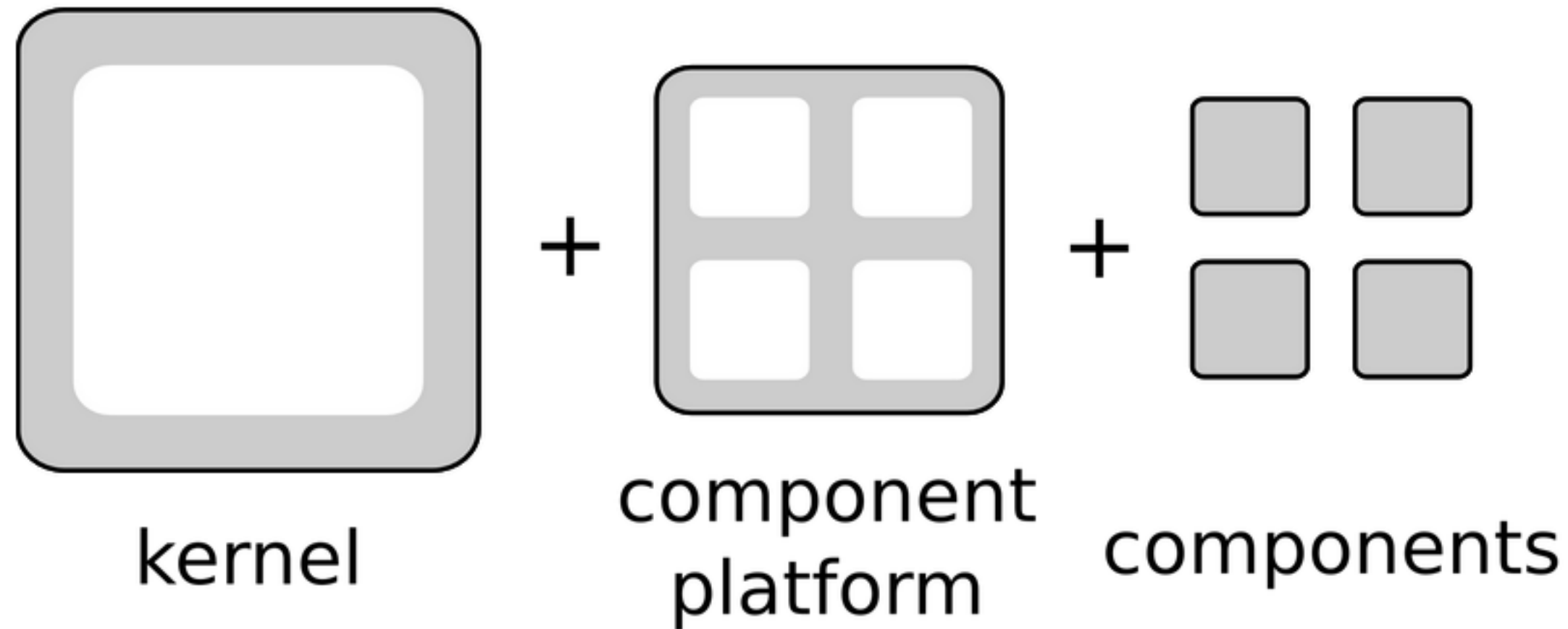
# Component-Based Development

---

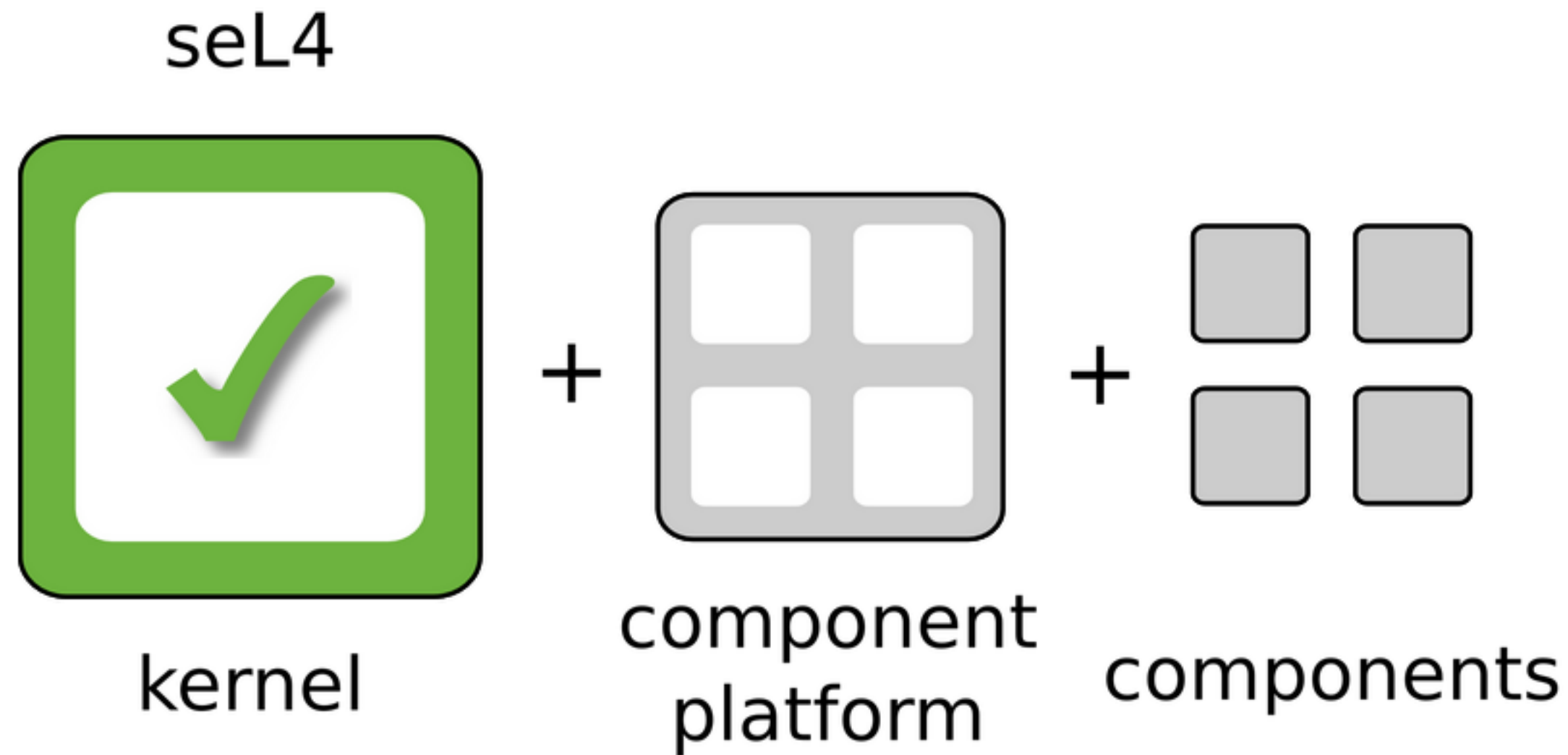


# Component-Based Development

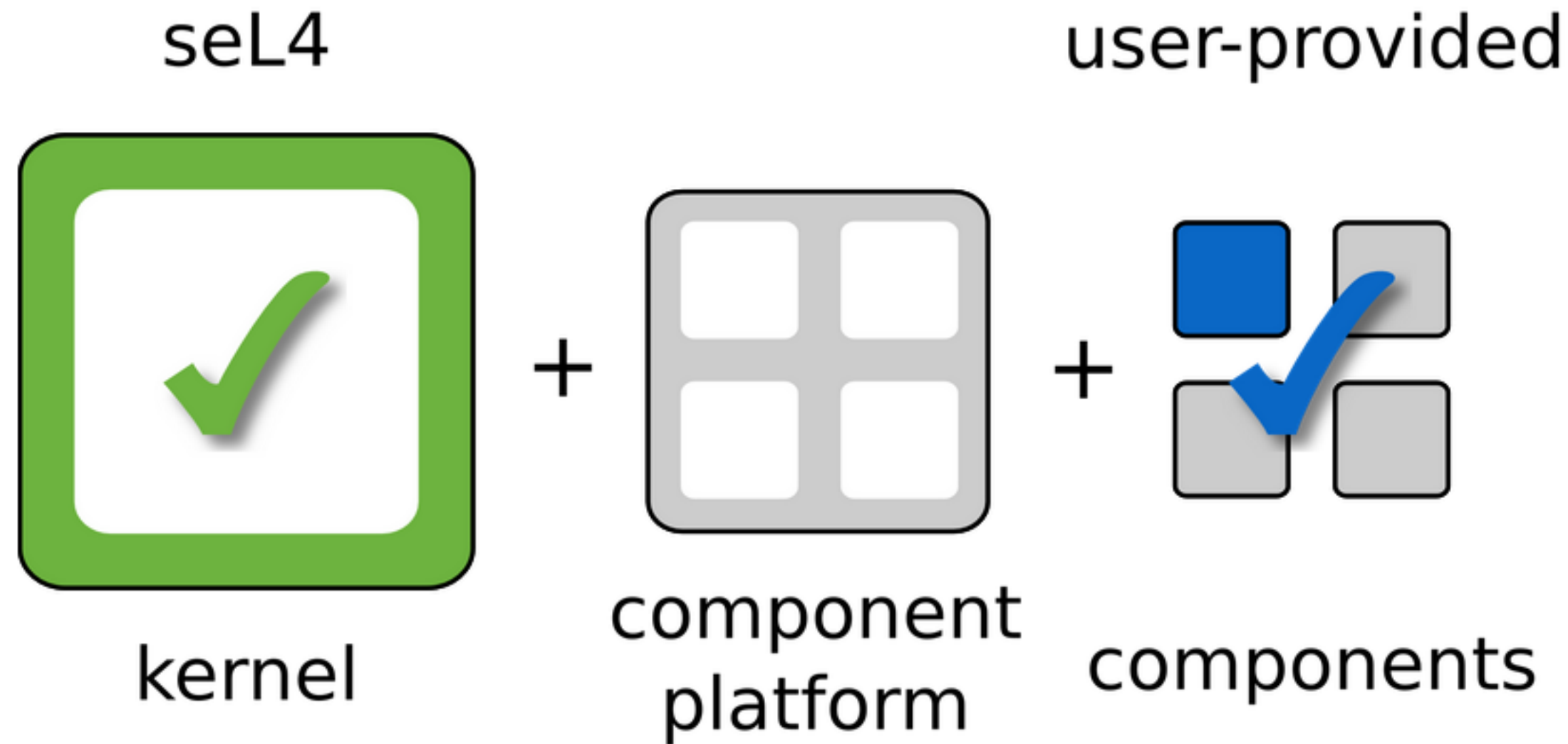
---



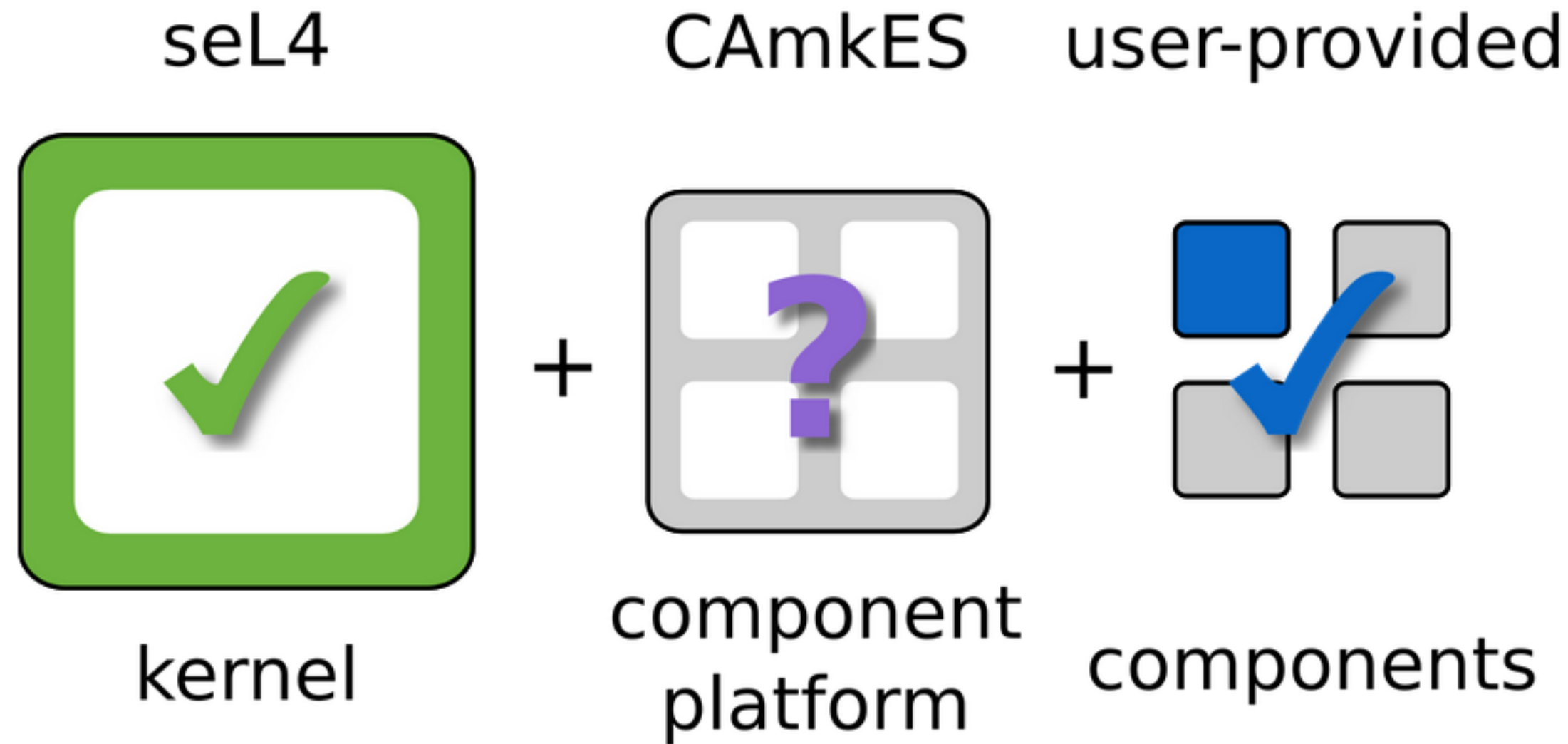
# Component-Based Development



# Component-Based Development



# Component-Based Development



# End-to-end Guarantees

---

architecture  
description



component  
code

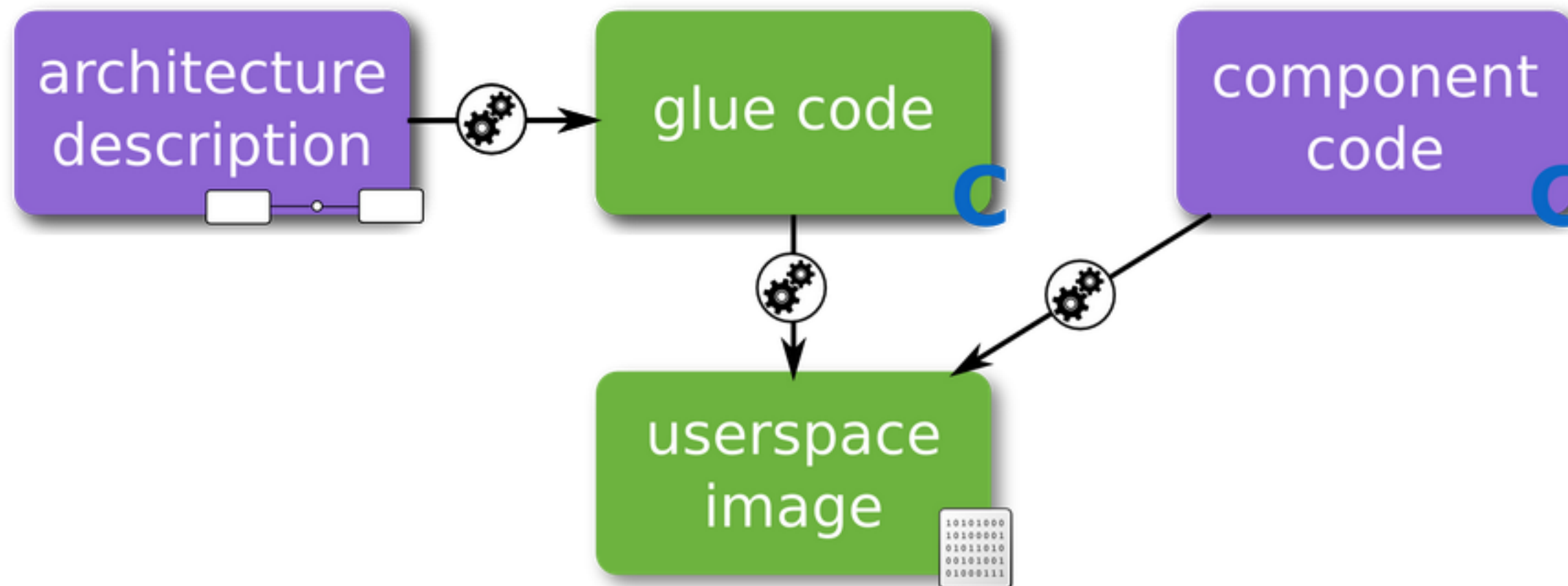


# End-to-end Guarantees

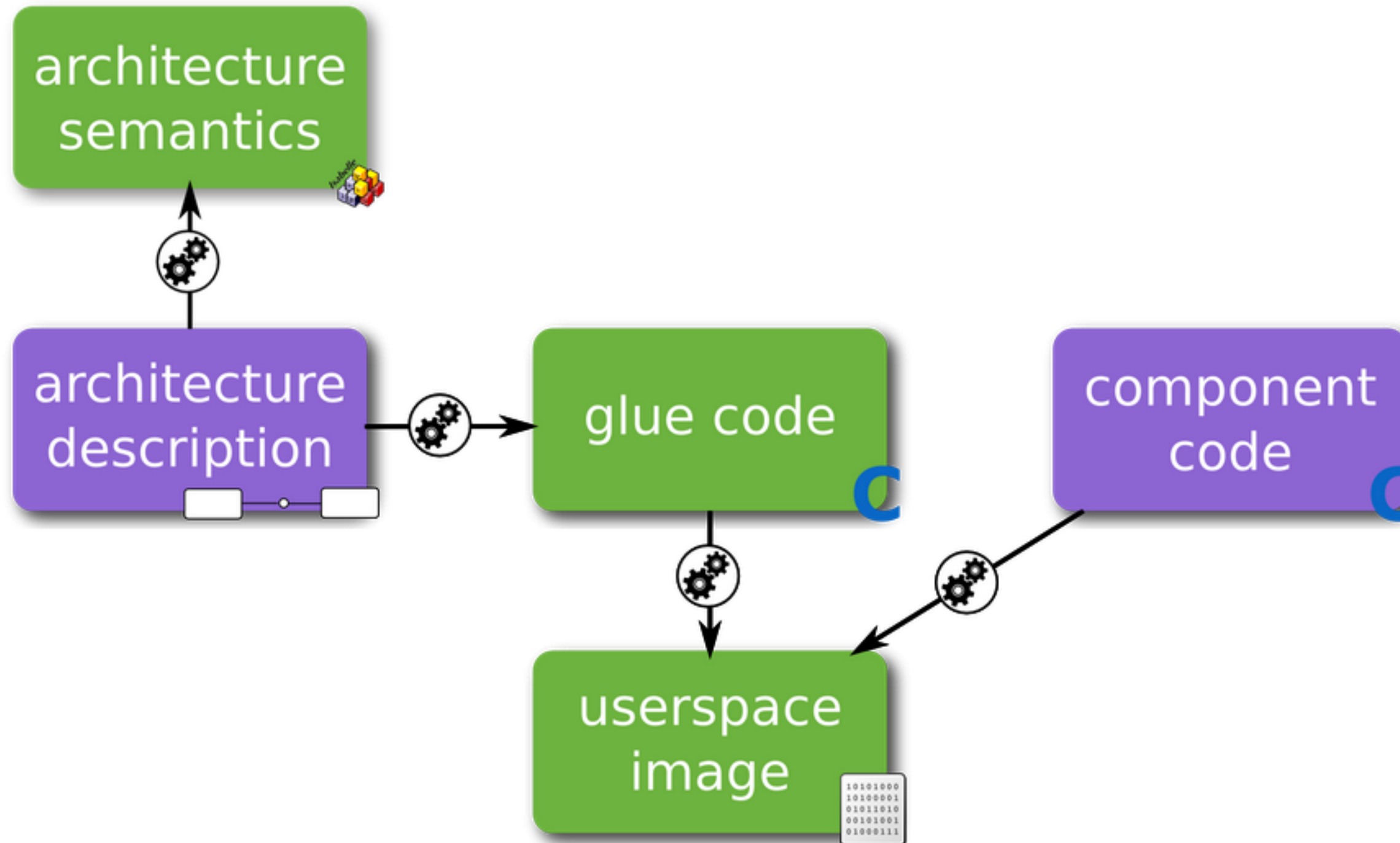
---



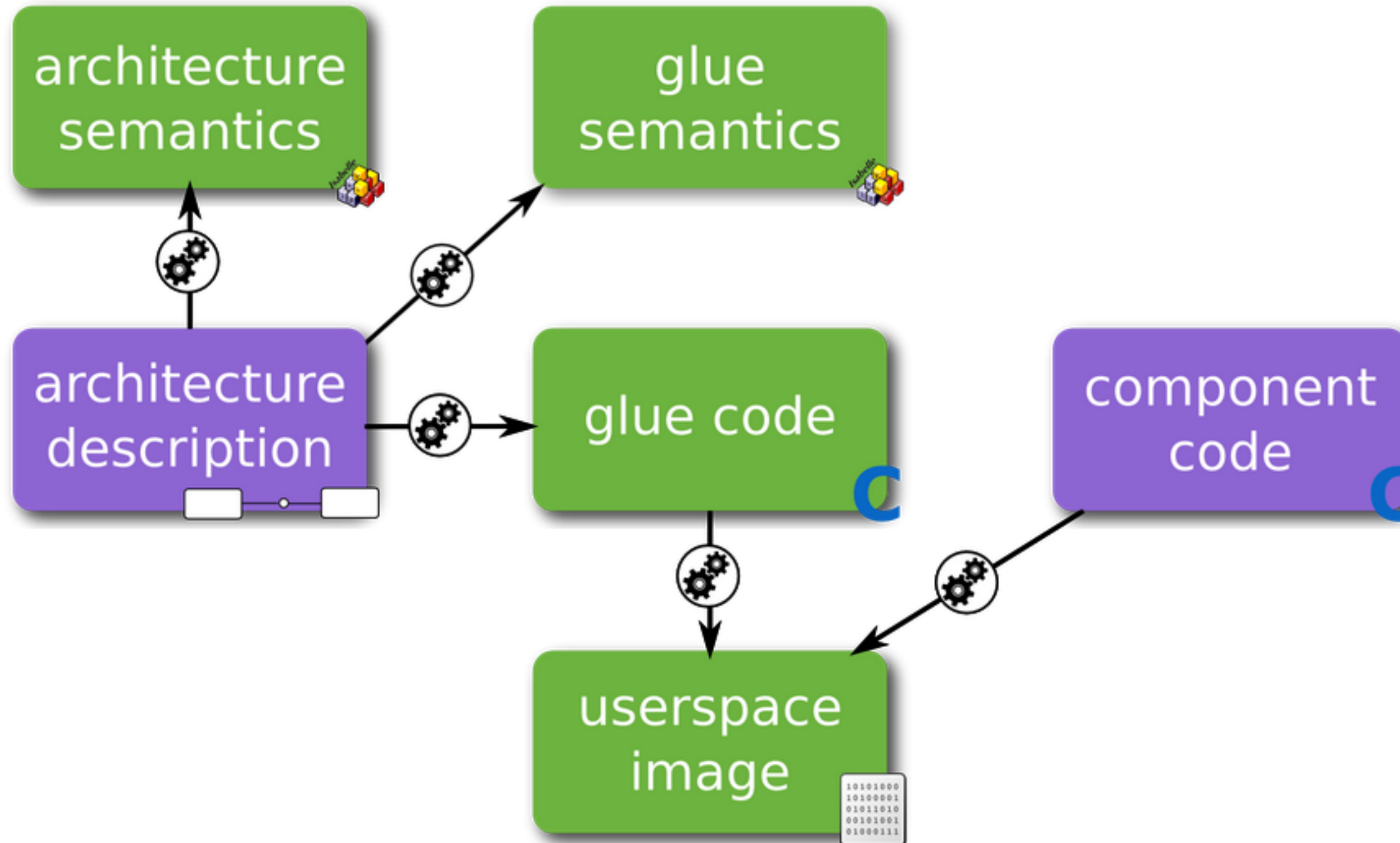
# End-to-end Guarantees



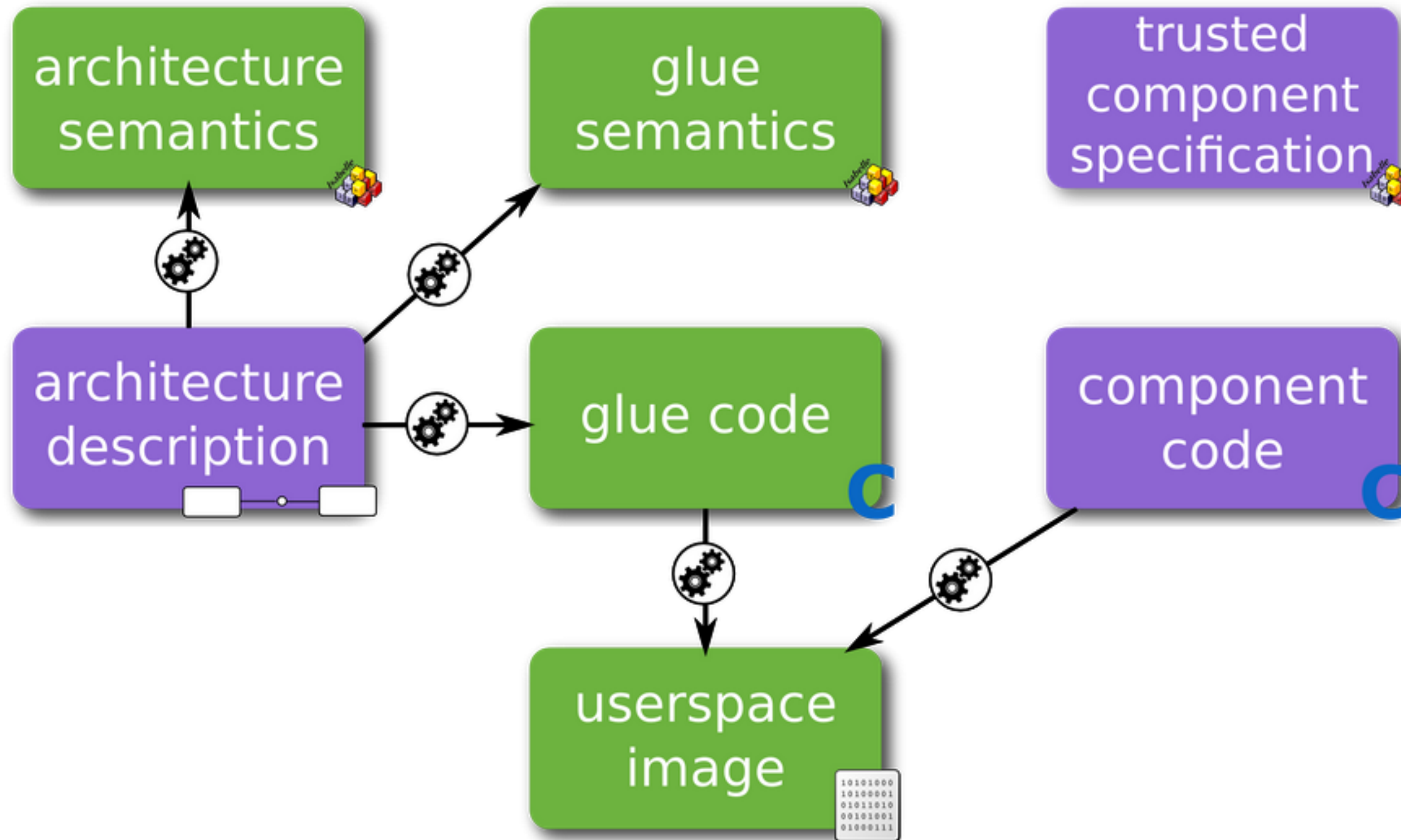
# End-to-end Guarantees



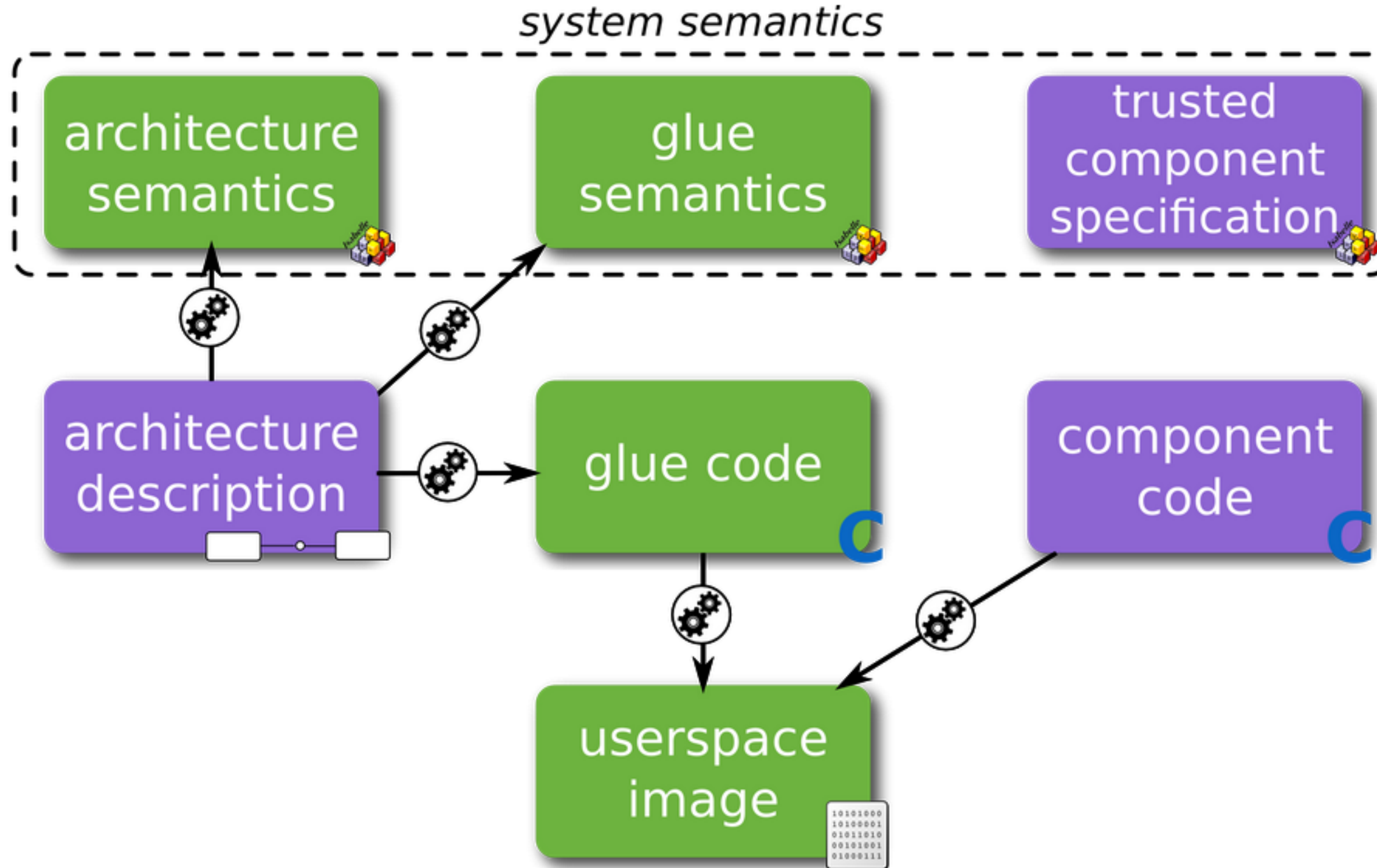
# End-to-end Guarantees



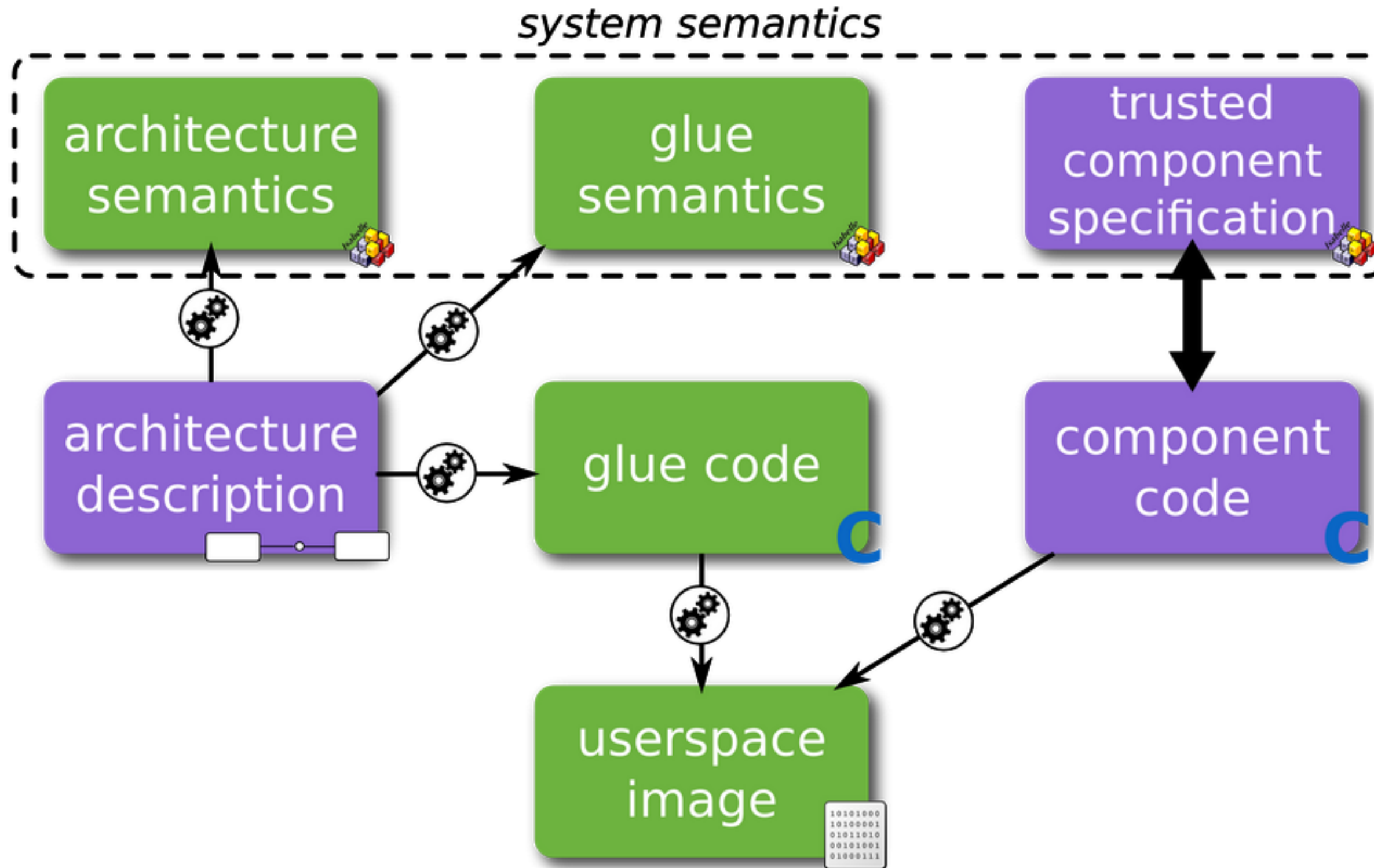
# End-to-end Guarantees



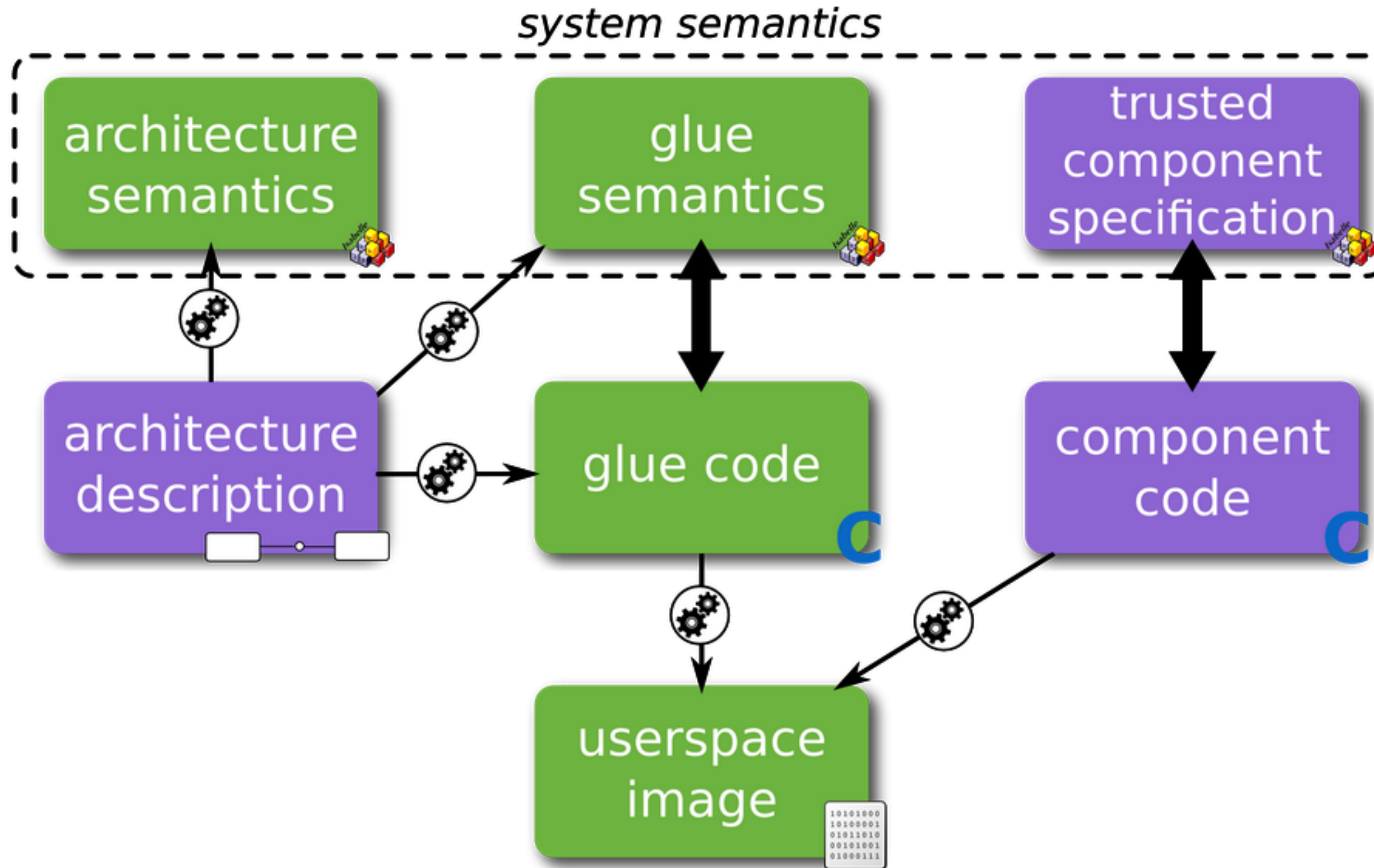
# End-to-end Guarantees



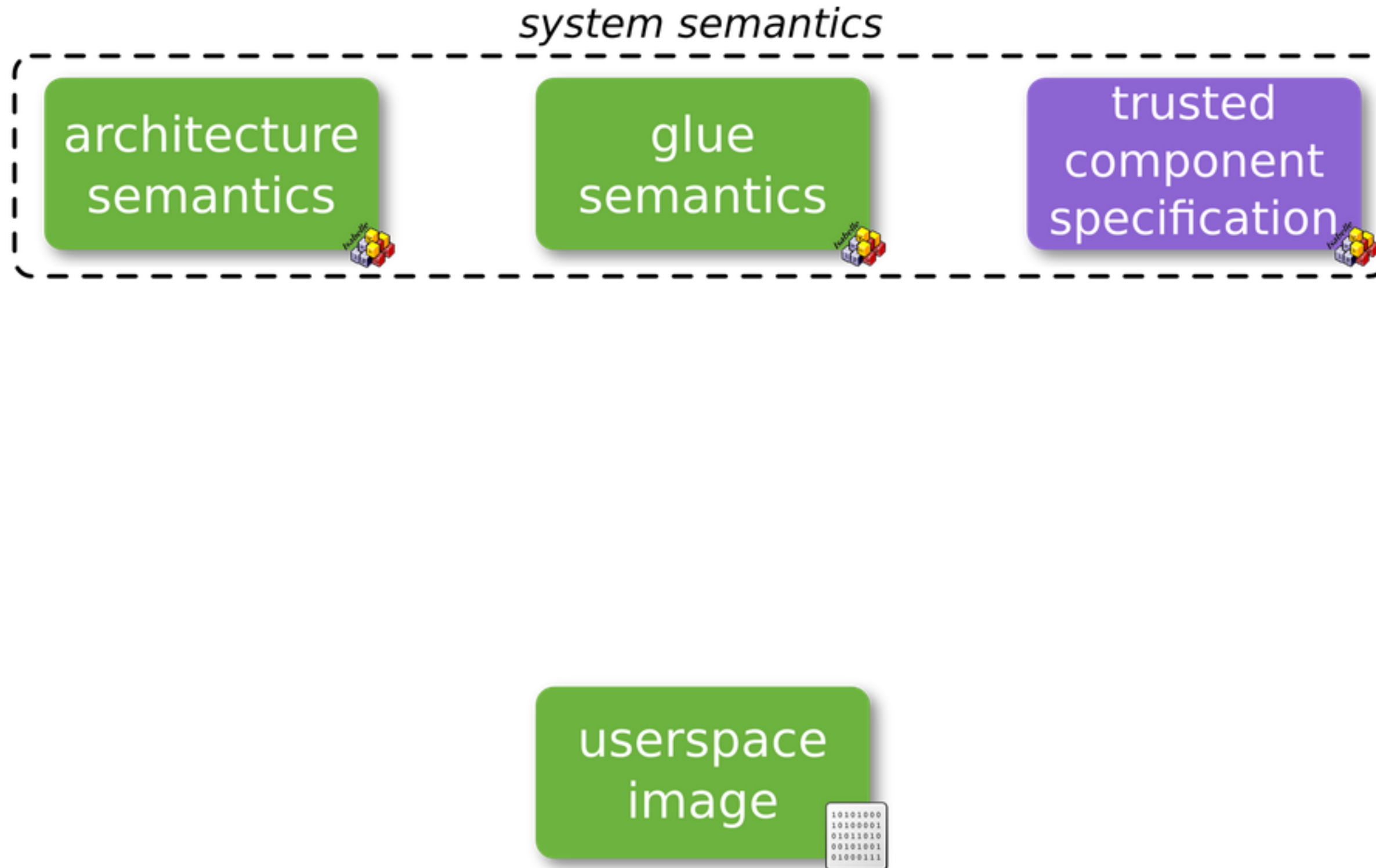
# End-to-end Guarantees



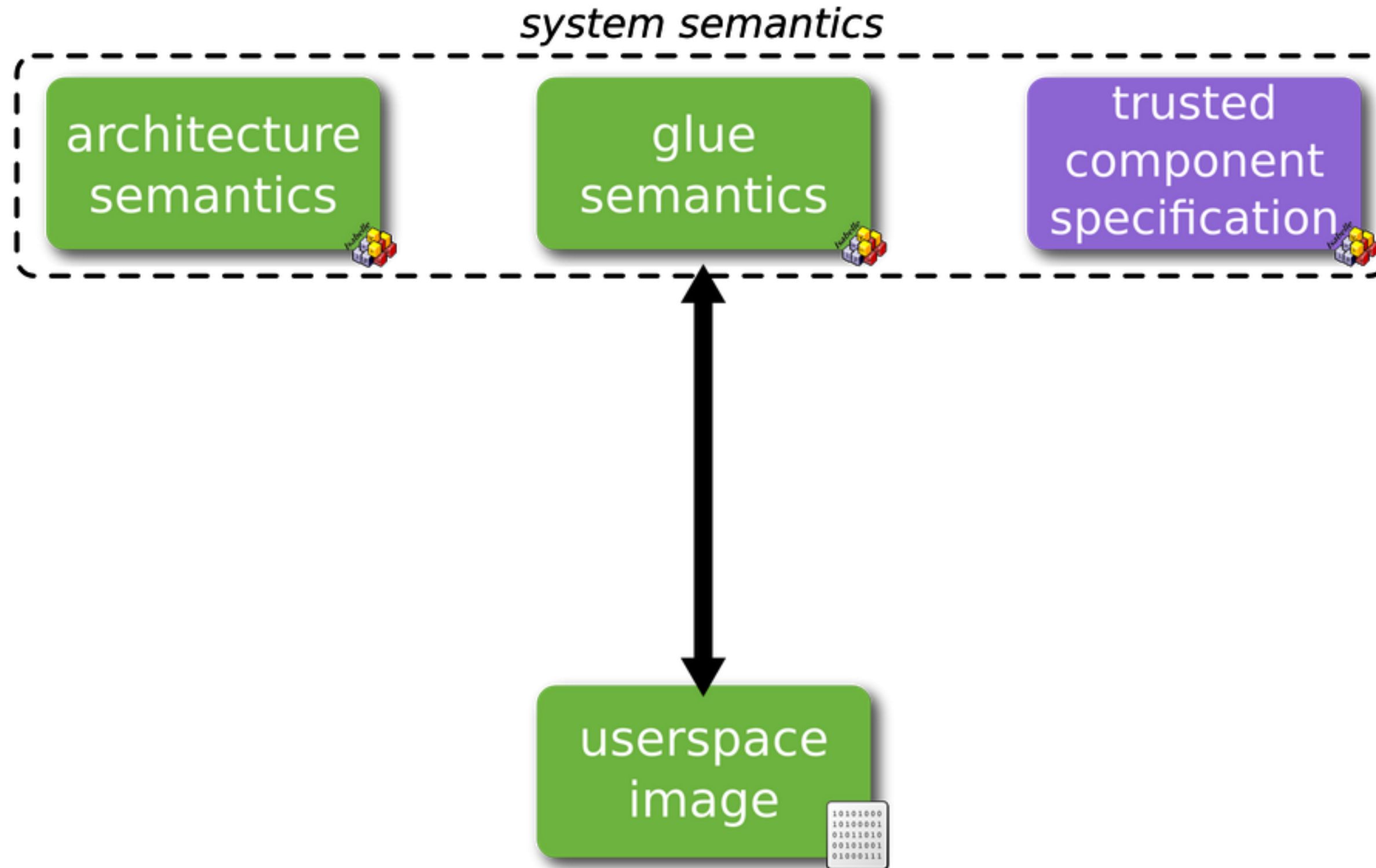
# End-to-end Guarantees



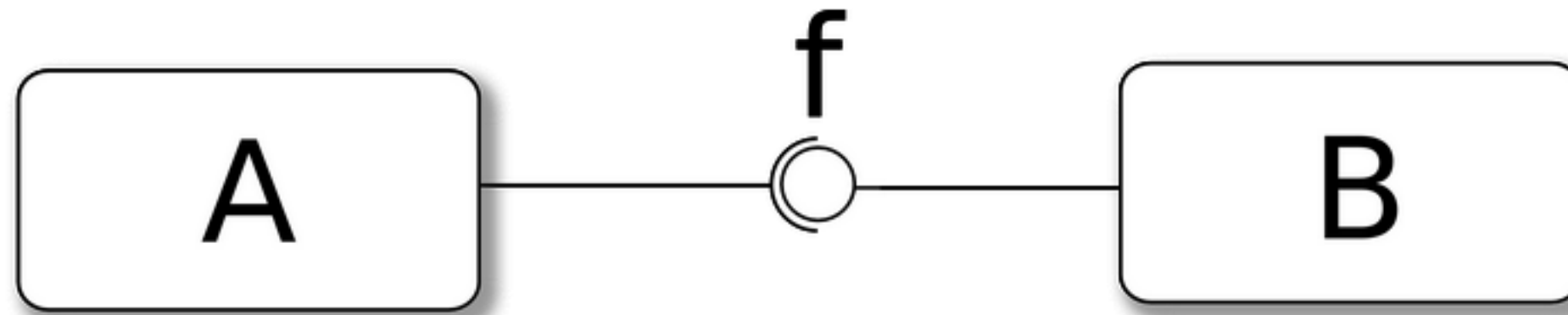
# End-to-end Guarantees



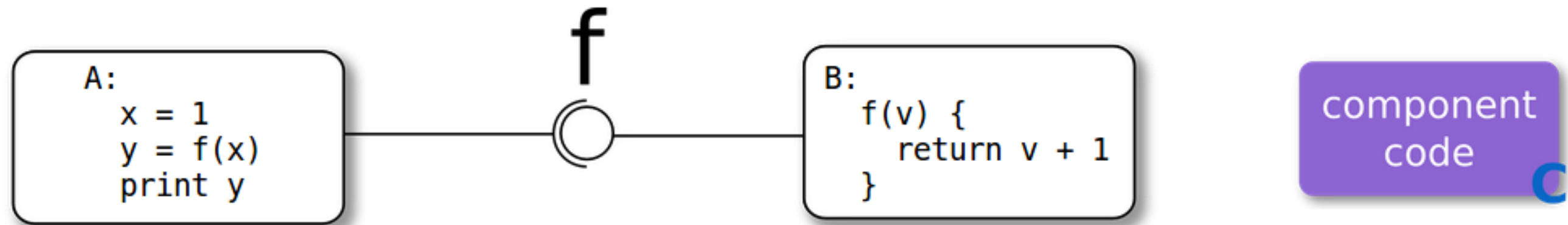
# End-to-end Guarantees



- Component platform for seL4
- Abstractions for procedures, events, shared memory
- Limited to static architectures



- Component platform for seL4
- Abstractions for procedures, events, shared memory
- Limited to static architectures



- Component platform for seL4
- Abstractions for procedures, events, shared memory
- Limited to static architectures

```
A:  
  x = 1  
  y = f(x)  
  print y
```

```
B:  
  f(v) {  
    return v + 1  
  }
```

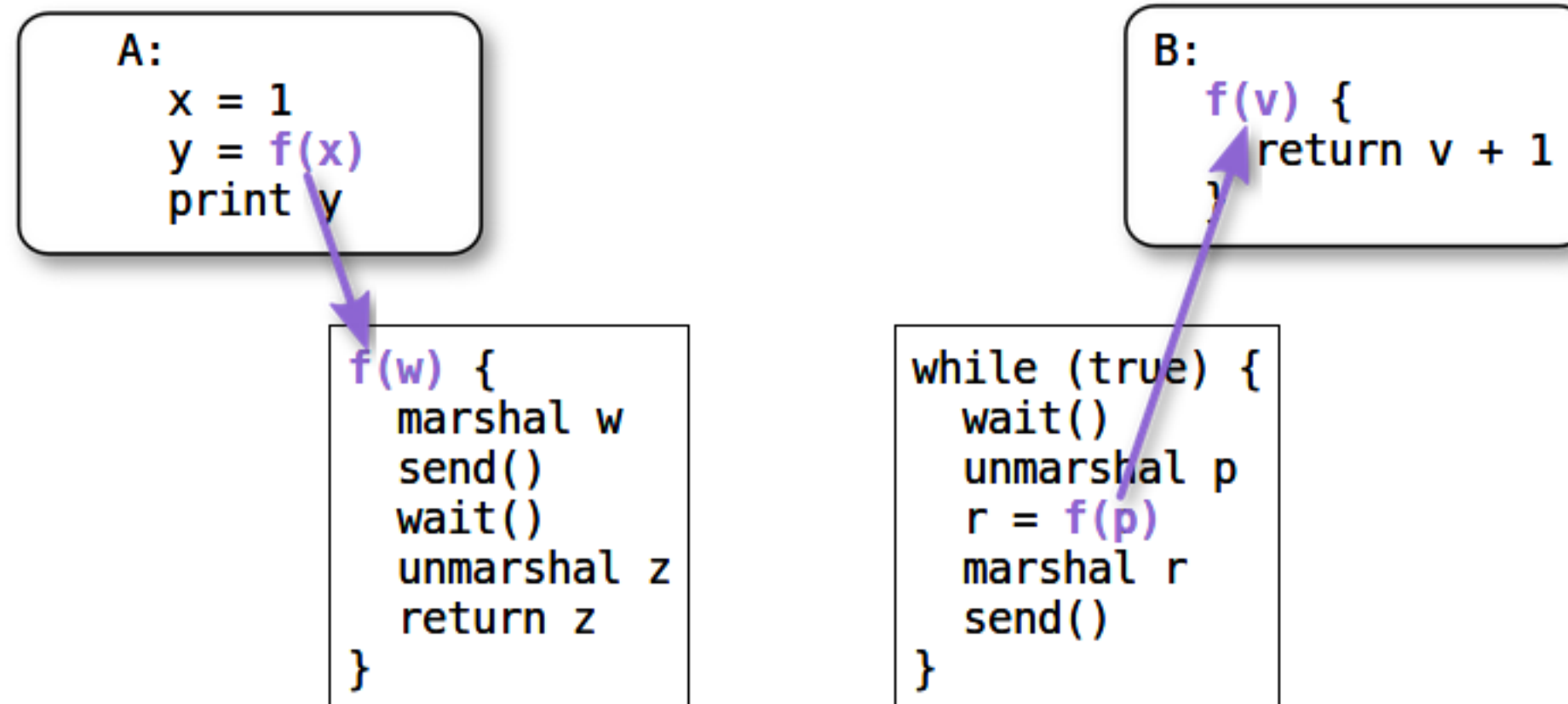
```
f(w) {  
  marshal w  
  send()  
  wait()  
  unmarshal z  
  return z  
}
```

```
while (true) {  
  wait()  
  unmarshal p  
  r = f(p)  
  marshal r  
  send()  
}
```

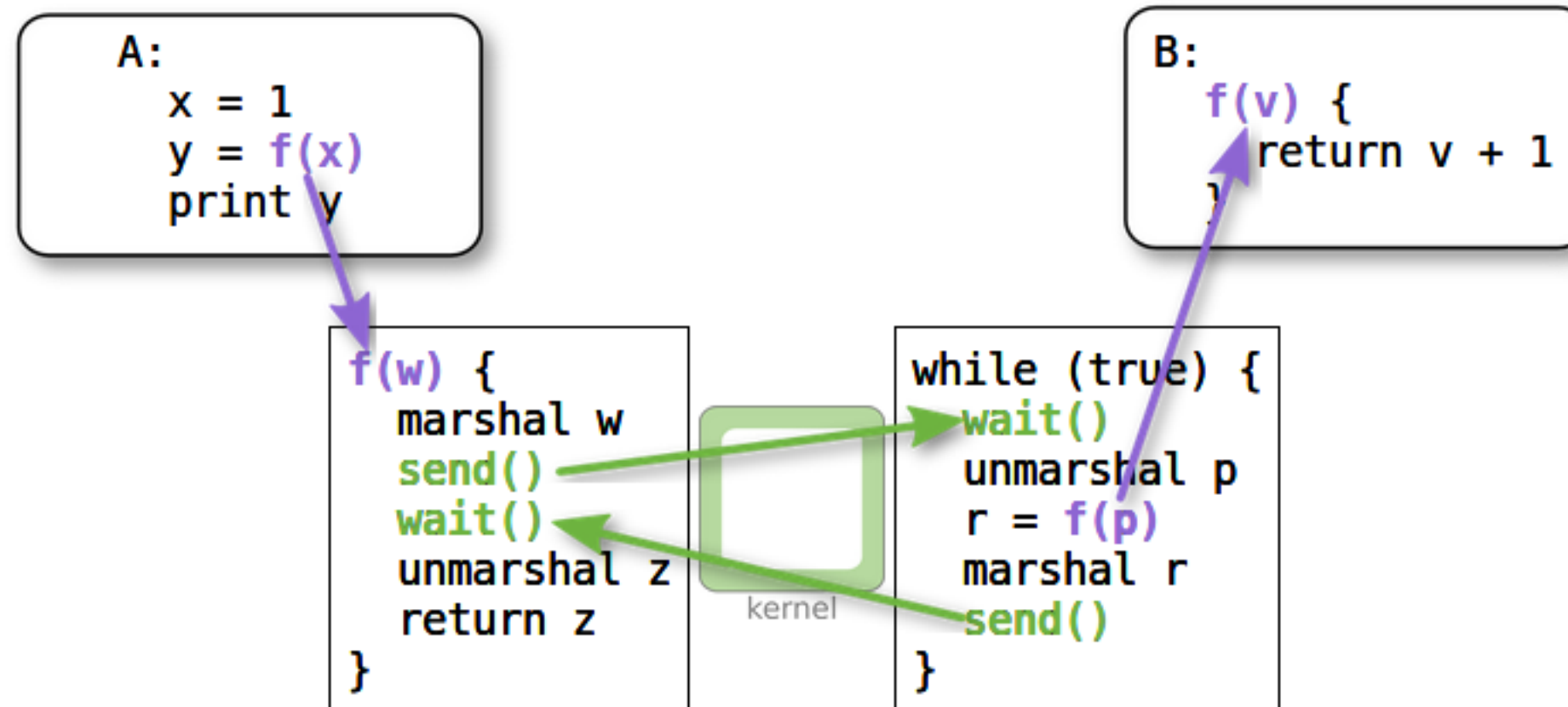
glue code



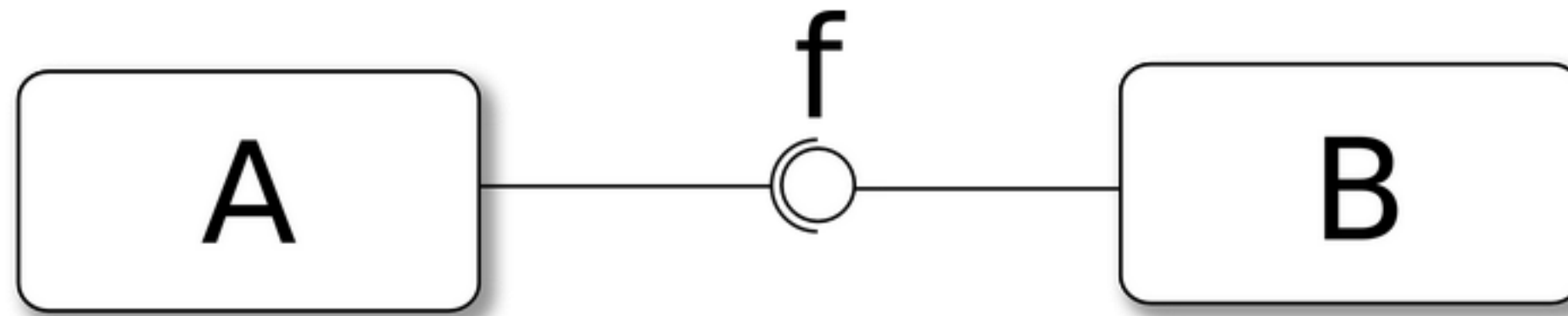
- Component platform for seL4
- Abstractions for procedures, events, shared memory
- Limited to static architectures

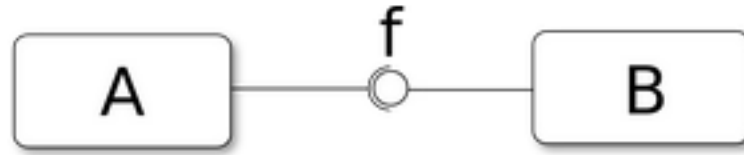


- Component platform for seL4
- Abstractions for procedures, events, shared memory
- Limited to static architectures



- Component platform for seL4
- Abstractions for procedures, events, shared memory
- Limited to static architectures

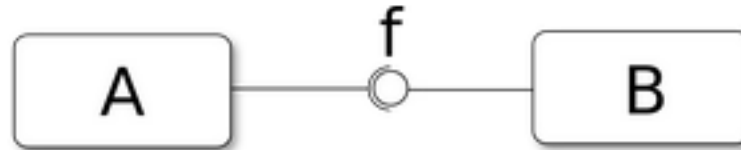




```
composition {  
  component Client A;  
  component Server B;  
  
  connection RPC f(from A.i, to B.j);  
}
```

# Architecture Semantics

architecture  
semantics



```
composition {  
  component Client A;  
  component Server B;  
  
  connection RPC f(from A.i, to B.j);  
}
```



```
definition  
  system :: composition  
where  
  system ≡ (  
    components = [('A', Client),  
                  ('B', Server)],  
    connections = [('f', (  
      conn_type = RPC,  
      conn_from = ('A', 'i'),  
      conn_to = ('B', 'j')  
    ))]  
  )
```



**UserStep**  $x = 1$

**SKIP** 

$s_1 ; ; s_2$

**IF**  $c$  **THEN**  $s_1$  **ELSE**  $s_2$

**WHILE**  $c$  **DO**  $s$

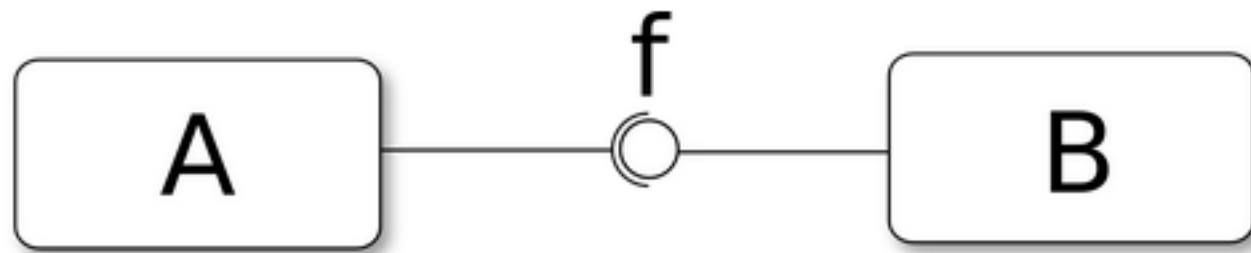
$s_1 \sqcup s_2$  

**Request** 

**Response** 

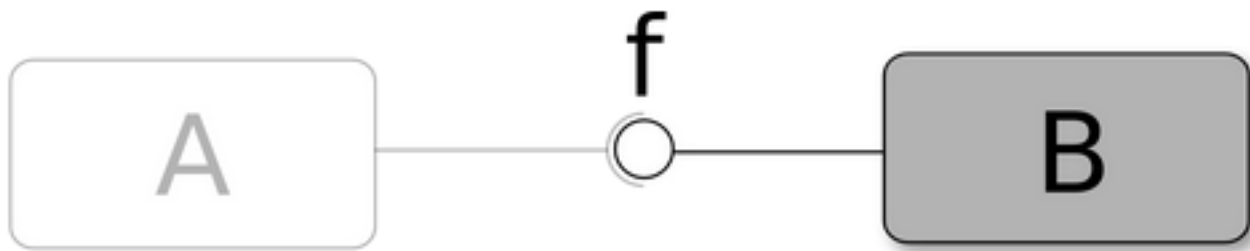
# Component Behaviour

---

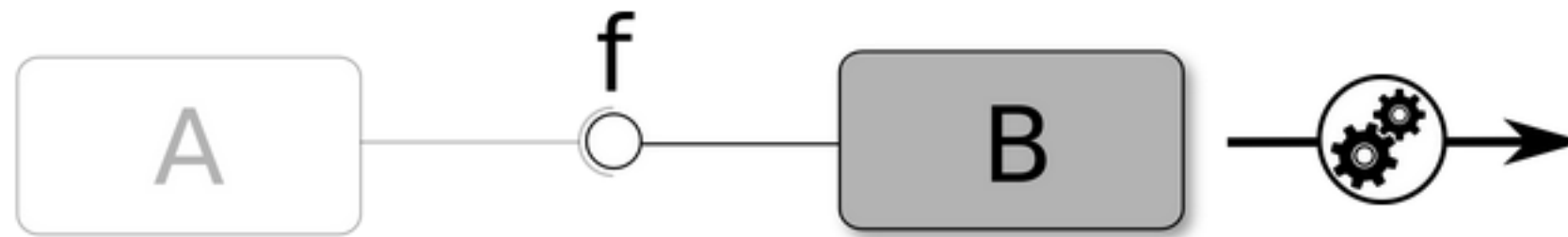


# Component Behaviour

---



# Component Behaviour



```
definition
  B_untrusted :: ...
where
  B_untrusted ≡
    WHILE True DO (
      UserStep
      ⊓ ArbitraryRequest f
      ⊓ ArbitraryResponse f
    )
```



glue  
semantics



## definition

Receive\_B\_f :: ...

## where

Receive\_B\_f fembed **f** fproject  $\equiv$

**Response** ... ;;

**f** ;;

**Request** ...



```
wait()  
unmarshal p  
r = f(p)  
marshal r  
send()
```

glue  
semantics



## definition

Receive\_B\_f :: ...

## where

Receive\_B\_f fembed f fproject  $\equiv$

**Response** ... ;;

f ;;

**Request** ...



```
wait()  
unmarshal p  
r = f(p)  
marshal r  
send()
```

glue  
semantics



## definition

Receive\_B\_f :: ...

## where

Receive\_B\_f fembed **f** fproject  $\equiv$

**Response** ... ;;

**f** ;;

**Request** ...



```
wait()  
unmarshal p  
r = f(p)  
marshal r  
send()
```

glue  
semantics



## definition

Receive\_B\_f :: ...

## where

Receive\_B\_f fembed **f** fproject  $\equiv$

**Response** ... ;;

**f** ;;

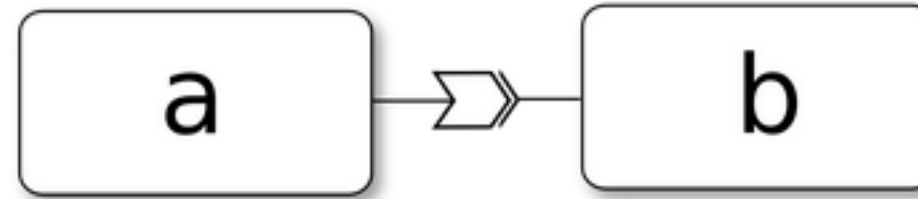
**Request** ...



```
wait()  
unmarshal p  
r = f(p)  
marshal r  
send()
```

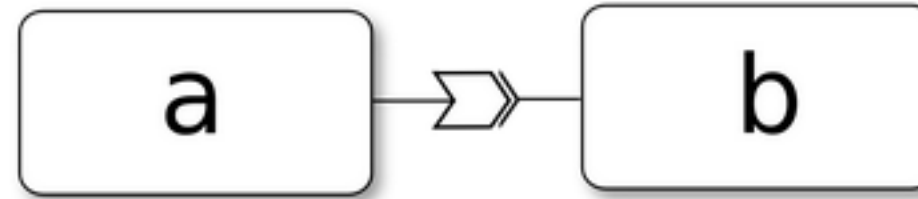
# Introduced Components

---

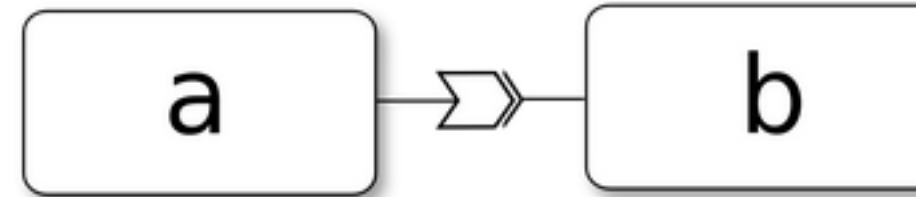


# Introduced Components

---

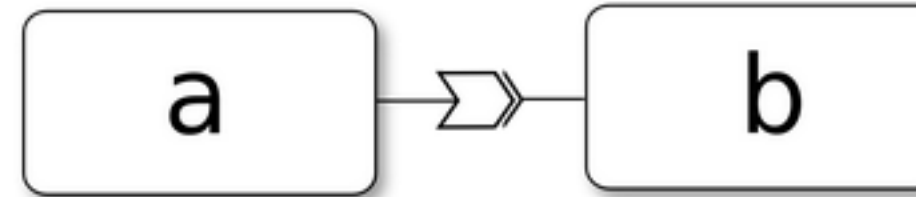


# Introduced Components



```
definition
  event :: ...
where
  event c ≡ WHILE True DO
    Response (λq s'. case s' of Event s ⇒
      (case q_data q of
        Set ⇒ ...
        | Poll ⇒ ...
        | _ ⇒ {})))
```

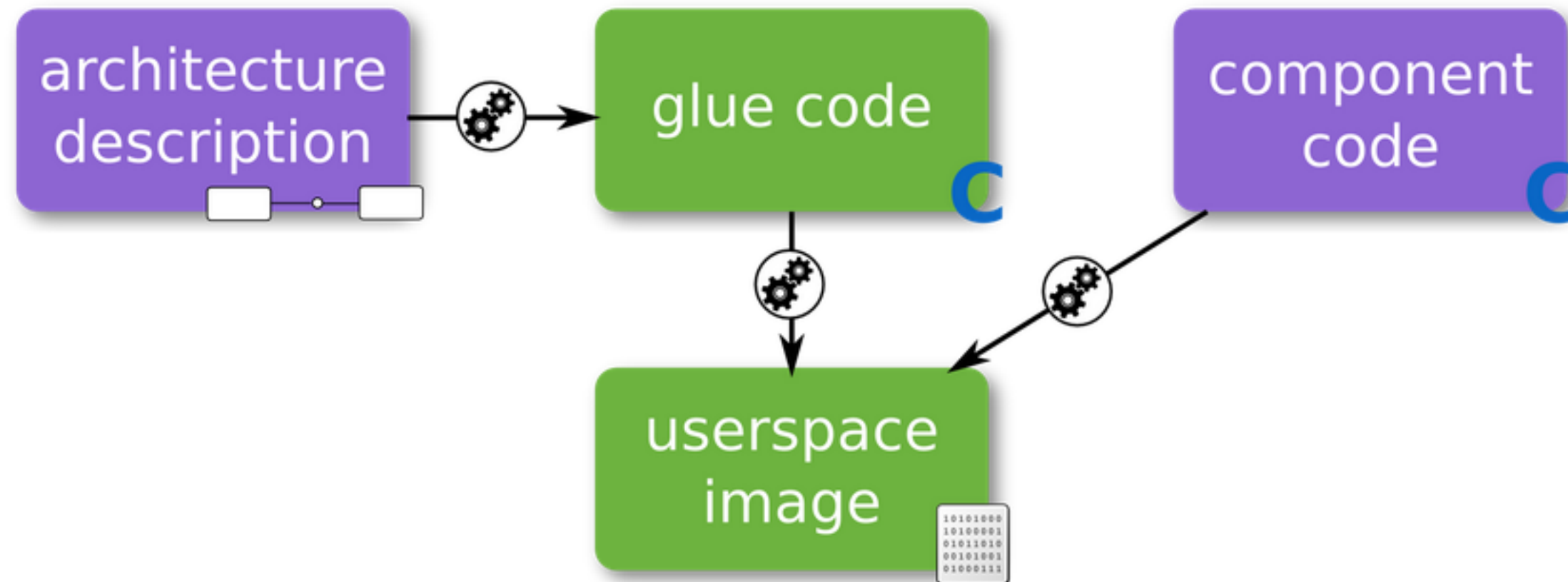
# Introduced Components



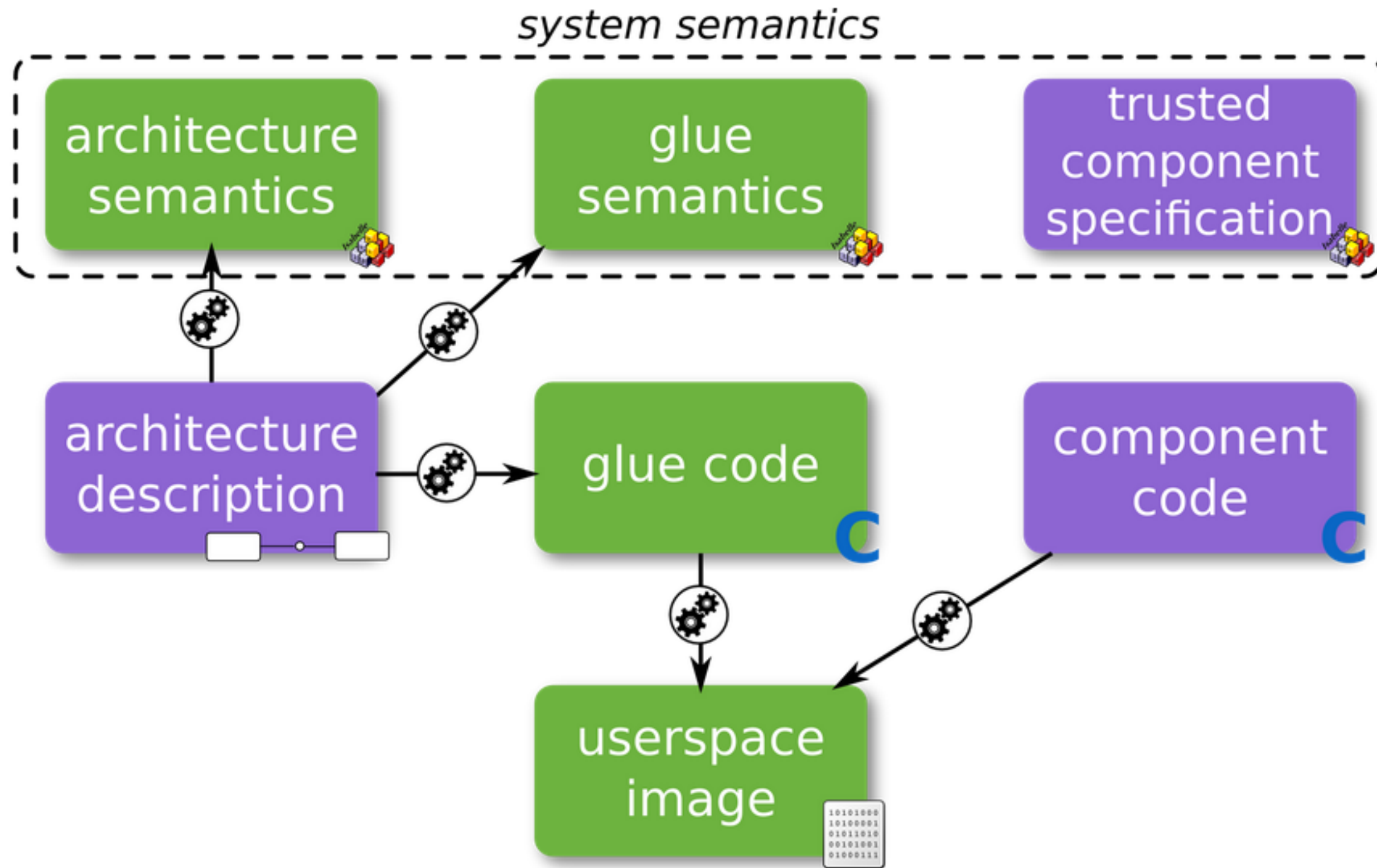
```
definition
  event :: ...
where
  event c ≡ WHILE True DO
    Response (λq s'. case s' of Event s ⇒
      (case q_data q of
        | Set ⇒ ...
        | Poll ⇒ ...
        | _ ⇒ {})))
```

```
definition
  memory :: ...
where
  memory c ≡ WHILE True DO
    Response (λq s'. case s' of Memory s ⇒
      (case q_data q of
        | Read addr ⇒ ...
        | Write addr val ⇒ ...
        | _ ⇒ {}))
```

# Towards a Verified Component Platform



# Towards a Verified Component Platform



# Towards a Verified Component Platform

