



# Coupler Contracts Annex



# Goal of the Annex

---

New Construct for Separation of Concerns

Syntax Checking

Extended Semantics Checking



# Goal of the new Construct

---

Separate Construction of Non-Functional Patterns

Reuse of Non-Functional Patterns



# Separation Mechanism: How

---

## Intercept Port Connections

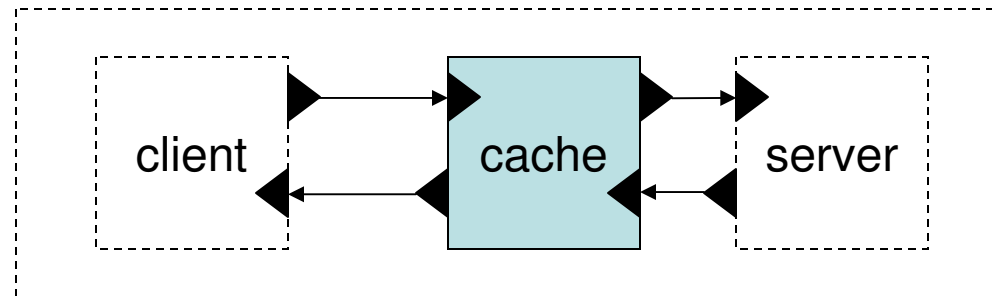
- Reroute the communication
- Add non-functional behavior to communication w/o interfering with components involved in the intercepted connection

## Interceptor (a.k.a. coupler)

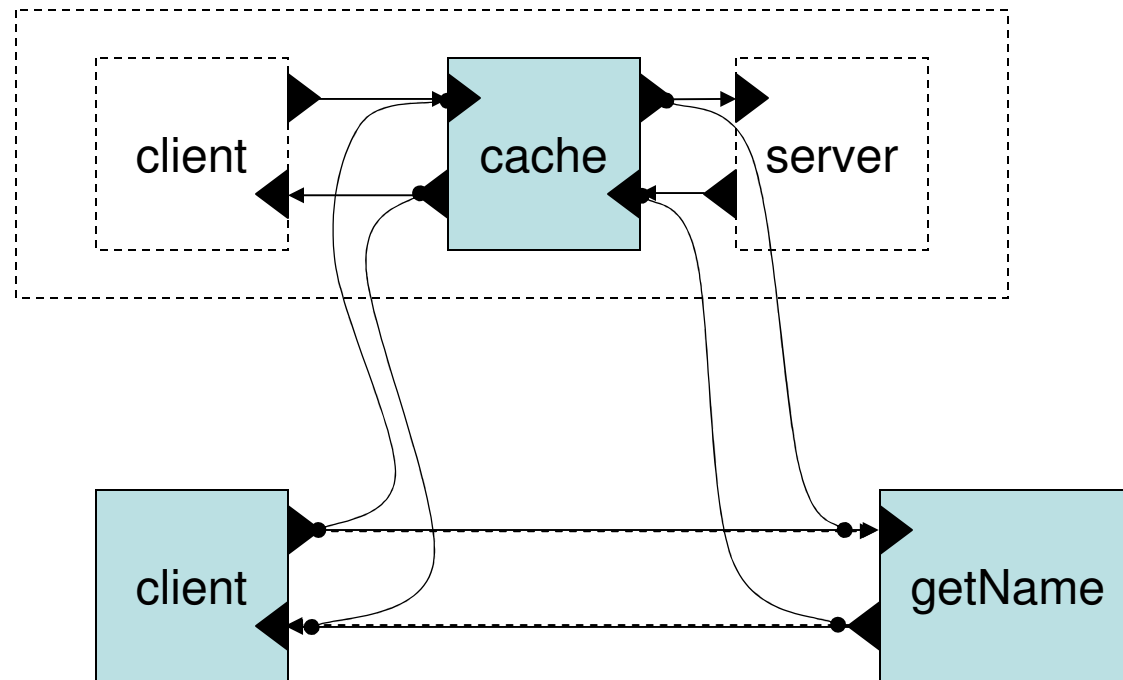
- Intercepts connections and defines the “coupling” of two components



# Sample Interceptor: Caching Pattern



# Interception



# What Syntax Checking

---

Intercepted Connection Exist

Interceptor Exist

Syntax

- `intercept(<connection>,<src interceptor port>,<dst interceptor port>);`



# Cache Interceptor AADL

**system** Cache

**features**

reqSrc: **in data port**;  
 reqDst: **out data port**;  
 respSrc: **in data port**;  
 respDst: **out data port**;

**end** Cache;

**system implementation** Cache.Impl

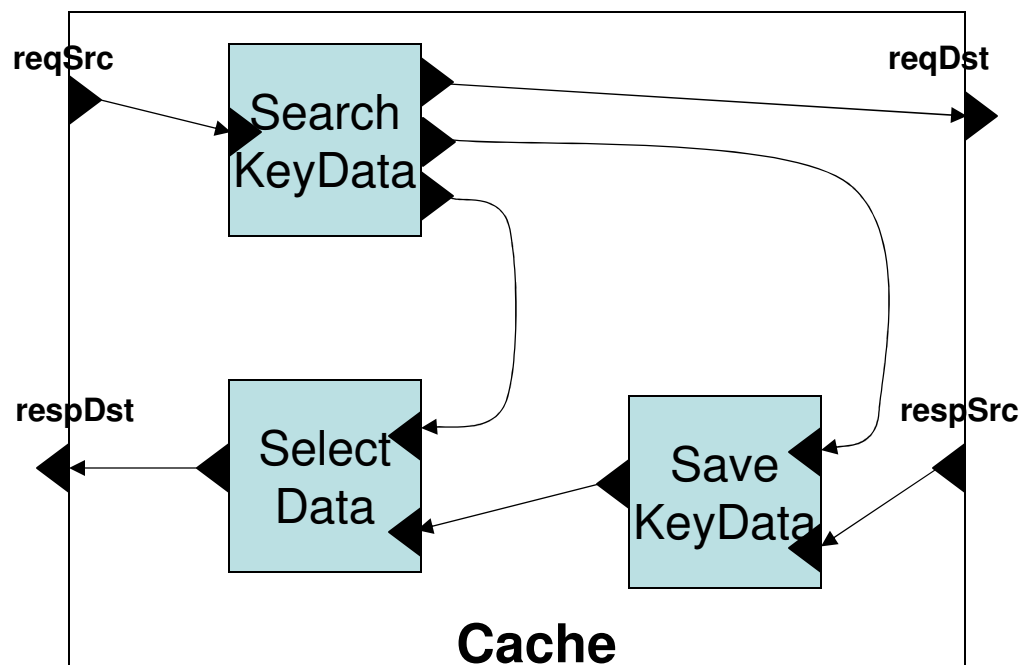
**subcomponents**

searchKDPair: **system** SearchKeyDataPair;  
 saveKDPair: **system** SaveKeyDataPair;  
 selData: **system** SelectData;

**connections**

c1: **data port** reqSrc -> searchKDPair.reqKey;  
 c2: **data port** searchKDPair.forwardKey -> reqDst;  
 c3: **data port** respSrc -> saveKDPair.dataToSave;  
 c4: **data port** searchKDPair.localData -> selData.localData;  
 c5: **data port** saveKDPair.forwardData -> selData.remoteData;  
 c6: **data port** selData.outData -> respDst;  
 c7: **data port** searchKDPair.keyToSave -> saveKDPair.keyToSave;

**end** Cache.Impl;



# AADL Interception Sample

```

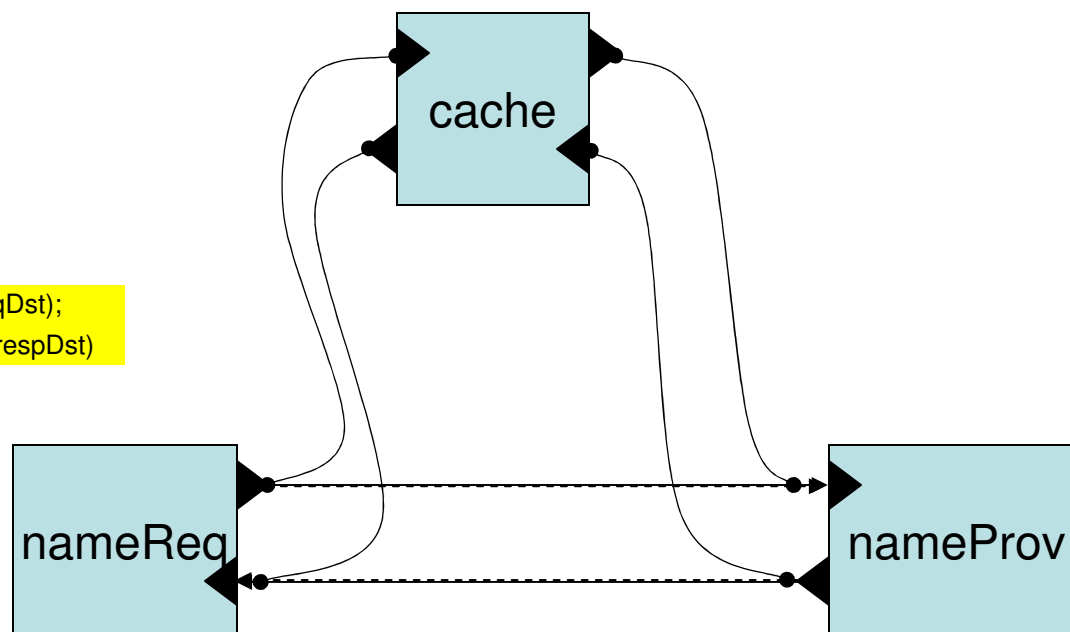
system implementation RequestingName.Impl
  subcomponents
    nameReq: system NameRequester.Impl;
    nameProv: system NameProvider.Impl;
  connections
    req: data port nameReq.reqName -> nameProv.reqName;
    resp: data port nameProv.respName -> nameReq.respName;
end RequestingName.Impl;

```

```

system implementation FinalSystem.Impl
  subcomponents
    logicView: system RequestingName.Impl;
    scaleView: system Cache.Impl;
  annex CouplerContracts {**
    intercept(logicView.req,scaleView.reqSrc,scaleView.reqDst);
    intercept(logicView.resp,scaleView.respSrc,scaleView.respDst)
  **};
end FinalSystem.Impl;

```



# What Semantic Check

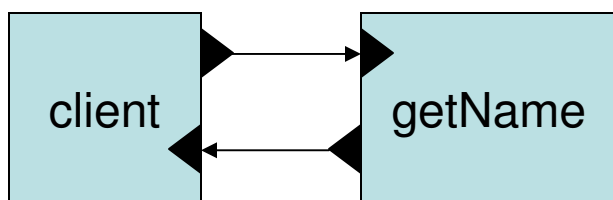
---

Assumptions about intercepted connection



# What Semantics Check

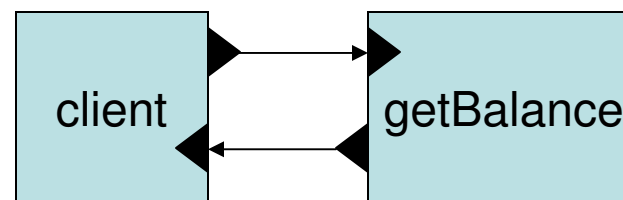
## Assumptions About Intercepted Connections



Cacheable

Every time I send the same request ID  
I receive the same *name*

Sending same request ID at different times  
may return different *balances*



NOT Cacheable



# What do I need for the semantic check

---

Encoding of Assumptions

Verification of Assumptions



# What do I need for the semantic check

---

## Encoding of Assumptions

- Design-by-contract!

## Verification of Assumptions



# What do I need for the semantic check

---

## Encoding of Assumptions

- Design-by-contract!
- New “interfacing” (interception)
  - Coupler Contracts

## Verification of Assumptions



# What do I need for the semantic check

---

## Encoding of Assumptions

- Design-by-contract!
- New “interfacing” (interception)
  - Coupler Contracts
- Additional constructs
  - Assumptions: *assumes*
  - Guarantees: *ensures*
- Additional Syntactic Verification

## Verification of Assumptions



# What do I need for the semantic check

---

## Encoding of Assumptions

- Design-by-contract!
- New “interfacing” (interception)
  - Coupler Contracts
- Additional constructs
  - Assumptions: *assumes*
  - Guarantees: *ensures*
- Additional Syntactic Verification

## Verification of Assumptions

- Non-functional assumptions:
  - behavior during execution



# What do I need for the semantic check

---

## Encoding of Assumptions

- Design-by-contract!
- New “interfacing” (interception)
  - Coupler Contracts
- Additional constructs
  - Assumptions: *assumes*
  - Guarantees: *ensures*
- Additional Syntactic Verification

## Verification of Assumptions

- Non-functional assumptions:
  - behavior during execution
- Temporal Logic!



# What do I need for the semantic check

---

## Encoding of Assumptions

- Design-by-contract!
- New “interfacing” (interception)
  - Coupler Contracts
- Additional constructs
  - Assumptions: *assumes*
  - Guarantees: *ensures*
- Additional Syntactic Verification

## Verification of Assumptions

- Non-functional assumptions:
  - behavior during execution
- Temporal Logic!
- Need External Tool:



# What do I need for the semantic check

---

## Encoding of Assumptions

- Design-by-contract!
- New “interfacing” (interception)
  - Coupler Contracts
- Additional constructs
  - Assumptions: *assumes*
  - Guarantees: *ensures*
- Additional Syntactic Verification

## Verification of Assumptions

- Non-functional assumptions:
  - behavior during execution
- Temporal Logic!
- Need External Tool: **Alloy!**



# What do I need for the semantic check

## Encoding of Assumptions

- Design-by-contract!
- New “interfacing” (interception)
  - Coupler Contracts
- Additional constructs
  - Assumptions: *assumes*
  - Guarantees: *ensures*
- Additional Syntactic Verification

## Verification of Assumptions

- Non-functional assumptions:
  - behavior during execution
- Temporal Logic!
- Need External Tool: **Alloy!**
  - Model port-based components that exchange tokens over time



# What do I need for the semantic check

## Encoding of Assumptions

- Design-by-contract!
- New “interfacing” (interception)
  - Coupler Contracts
- Additional constructs
  - Assumptions: *assumes*
  - Guarantees: *ensures*
- Additional Syntactic Verification

*Comp1.port[time1] <compare>*  
*Comp2.port[time2]*

## Verification of Assumptions

- Non-functional assumptions:
  - behavior during execution
- Temporal Logic!
- Need External Tool: **Alloy!**
  - Model port-based components that exchange tokens over time



# What do I need for the semantic check

## Encoding of Assumptions

- Design-by-contract!
- New “interfacing” (interception)
  - Coupler Contracts
- Additional constructs
  - Assumptions: *assumes*
  - Guarantees: *ensures*
- Additional Syntactic Verification

*Comp1.port[time1] <compare>*  
*Comp2.port[time2]*

## Verification of Assumptions

- Non-functional assumptions:
  - behavior during execution
- Temporal Logic!
- Need External Tool: **Alloy!**
  - Model port-based components that exchange tokens over time
- Need an Analysis Plug-in



# Cache Assumption

---

**system** Cache

**features**

reqSrc: **in data port**; reqDst: **out data port**;

respSrc: **in data port**; respDst: **out data port**;

**end** Cache;

**system implementation** Cache.Impl

...

**annex** CouplerContracts {\*\*

assumes no t1,t2:tick{(reqDst[t1]==reqDst[t2] &  
respSrc[t1]!=respSrc[t2])}

\*\*};

**end** Cache.Impl;



# NameProvider Guarantee

---

```
system implementation NameProvider.Impl
  annex CouplerContracts {**
    ensures all t1,t2:tick{(reqName[t1]==reqName[t2] =>
respName[t1]==respName[t2])}
    **};
end NameProvider.Impl;
```



# Cacheable / Not Cacheable

```
system implementation FinalSystem.Impl
subcomponents
  logicView: system RequestingName.Impl;
  scaleView: system Cache.Impl;
annex CouplerContracts {**
  intercept(logicView.req,
            scaleView.reqSrc,
            scaleView.reqDst);
  intercept(logicView.resp,
            scaleView.respSrc,
            scaleView.respDst)
  **};
end FinalSystem.Impl;
```

```
system implementation FinalSystem.Impl
subcomponents
  logicView: system RequestingBalance.Impl;
  scaleView: system Cache.Impl;
annex CouplerContracts {**
  intercept(logicView.req,
            scaleView.reqSrc,
            scaleView.reqDst);
  intercept(logicView.resp,
            scaleView.respSrc,
            scaleView.respDst)
  **};
end FinalSystem.Impl;
```





# Demo



# Summary of Extensions to AADL/OSATE



## Annex

- Encode interception construct
- Encode assumptions

## Analysis Plug-in

- Translate assumptions into Alloy model
- Verify the model
- Bring back results to OSATE designer





Questions?

