An Object-Z / CSP Based Approach for the Specification of Architectural Connectors

Mourad Maouche

Philadelphia University Jordan

Mohamed Bettaz

MESRS Algeria

IFIP WG 1.3 Sierra Nevada, Spain, January 14 - 18, 2008

Objective

- Propose an Object-Z / CSP based approach for the specification of architectural connectors which are seen as explicit semantic entities.
- The approach has to support incremental development of specifications, and allow for verification of properties.

Background

- Start: Software architecture = collection of computational components together with a collection of connectors.
- Follow: Formal basis by R. Allen and D. Garlan, “A formal Basis for architectural connection”, connector definition rely on the definition of notions such as: component, port, role, glue, connector, attachment, etc.

Motivation

- Why more than one language?
  - Few specifications languages are suited for modeling all aspects of software architectures.
- Why Z and CSP?
  - Both of them have been advocated for specifying different aspects of software architectures.

Motivation (cont.)

- Why Object-Z / CSP?
  - Object-Z is a semi-graphical notation - visual appeal: suitable for representing system and software components in general (readability).
  - CSP is suitable for specifying the interactions between such components (conciseness).
- Both languages have common semantic basis (Object-Z classes might be given semantics of CSP processes): this enables using and / or developing unified method of refinement for the integrated notation.

The approach

- Roles, ports (refinements of roles) and glue, are seen as components.
- (Computational) components, roles, ports and glue are specified by Object-Z classes.
- Internal behavior of roles , glue, and ports (method execution) is governed by preconditions on adequate state variables.
- Behavior of the connector is specified by a parallel composition of roles and glue.
- Attachment of ports as roles is specified by a CSP process parameterized by Object-Z classes.

Similar Work

- **R. Allen and D. Garlan**: Using of CSP-like notation.
- **G. Abowd, R. Allen and D. Garlan**
  - Using Z.
  - No notion of glue.
  - ports and roles are specified as basic types (not as schemas).

Similar Work (cont.)

- **J.L.Fiadeiro et al.** : CommUnity
  - Components, glue, roles are CommUnity “component designs”
  - Ports are not defined explicitly. They are represented by input and output variables in the description of the components.
  - A connector is a finite set of connections with the same glue.
  - A connection consists of a glue, a role, a signature, and two category morphisms connecting a glue with a role.
  - The semantics of a connector is the colimit of the diagram formed by its connections.
  - A component (to be connected to a role) is seen as a refinement (according to CommUnity meaning) of this role.

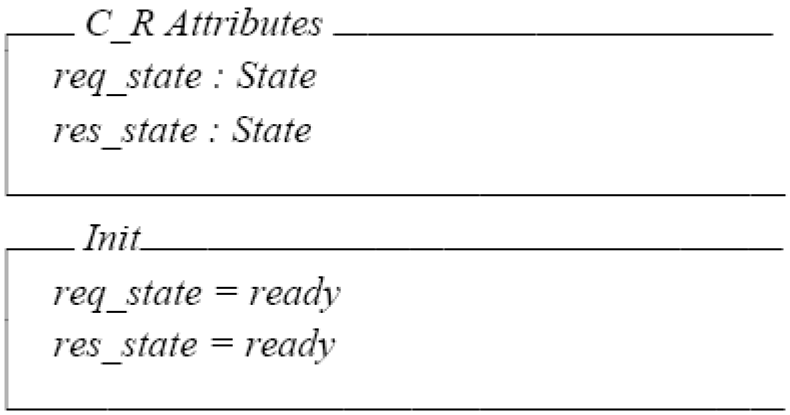
Example: A simple a client-server relationship

- Basic types: [*State, Request, Result, Invocation, Return*]
- *State == pending | ready*
- We suppose that the C-S_Connector handles only one service.

Roles

- The role describes the behavior that is expected of each of the interacting parts.

Client_Role: *The attributes*



C_R Attributes

req_state : State

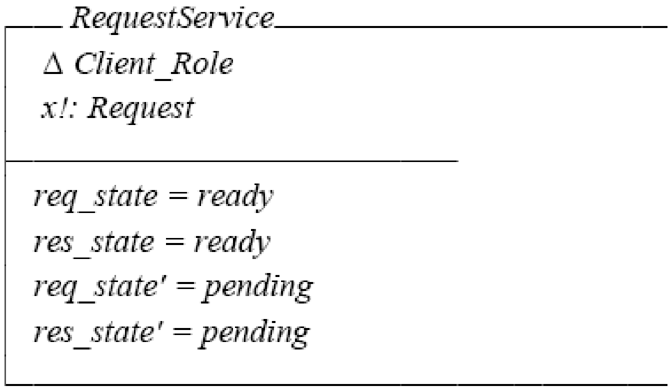
res_state : State

Init

req_state = ready

res_state = ready

Client_Role: *The methods*



RequestService

Δ *Client_Role*

x!: Request

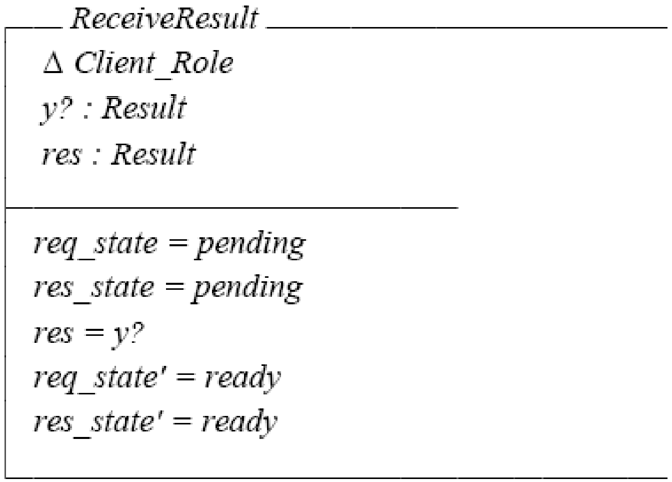
req_state = ready

res_state = ready

req_state′ = pending

res_state′ = pending

The client calls a service



ReceiveResult

Δ *Client_Role*

y? : Result

res : Result

req_state = pending

res_state = pending

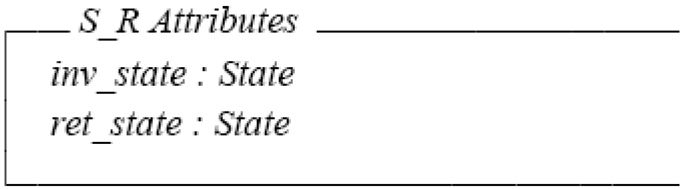
res = y?

req_state′ = ready

res_state′ = ready

The client receives the result

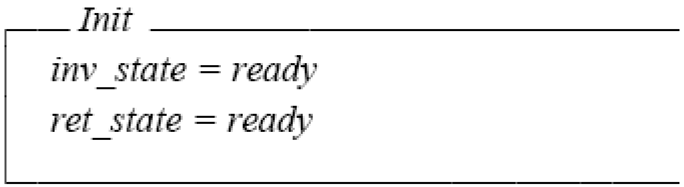
Server_Role: *The attributes*



S_R Attributes

inv_state : State

ret_state : State

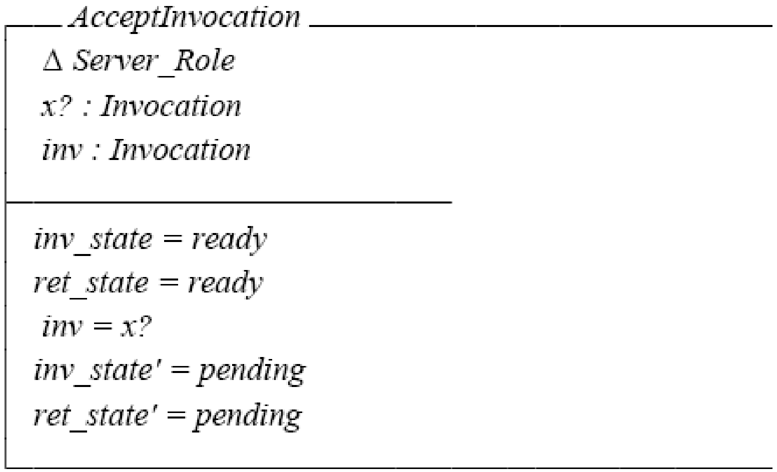


Init

inv_state = ready

ret_state = ready

Server_Role: *The methods*

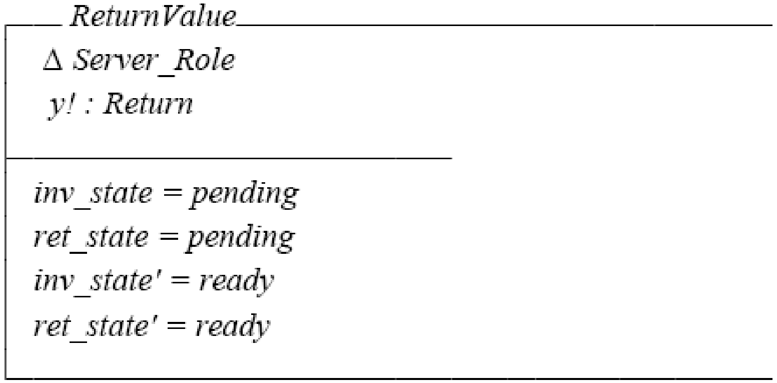


AcceptInvocation

Δ *Server_Role*
x? : Invocation
inv : Invocation

inv_state = ready
ret_state = ready
inv = x?
inv_state' = pending
ret_state' = pending

The server accepts the invocation



ReturnValue

Δ *Server_Role*
y! : Return

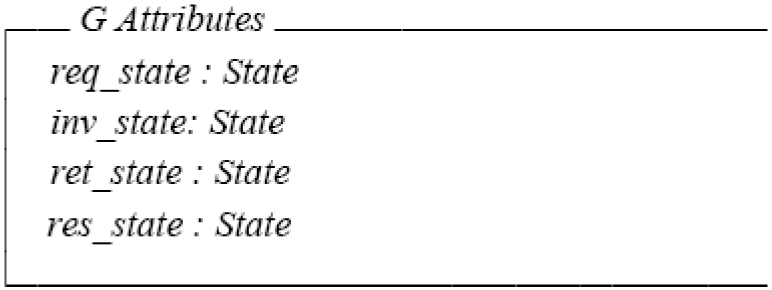
inv_state = pending
ret_state = pending
inv_state' = ready
ret_state' = ready

The server returns a value

Glue

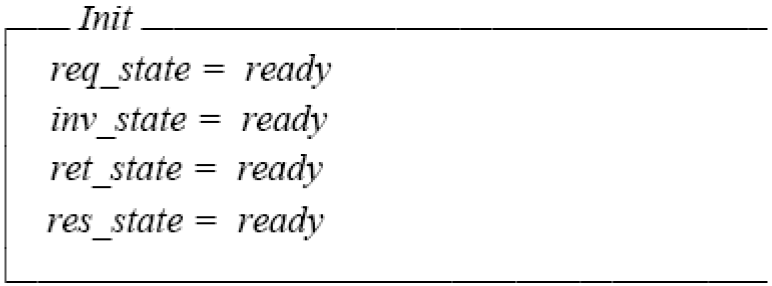
- The glue describes how the activities of the roles are coordinated.

Glue: *The Attributes*



G Attributes

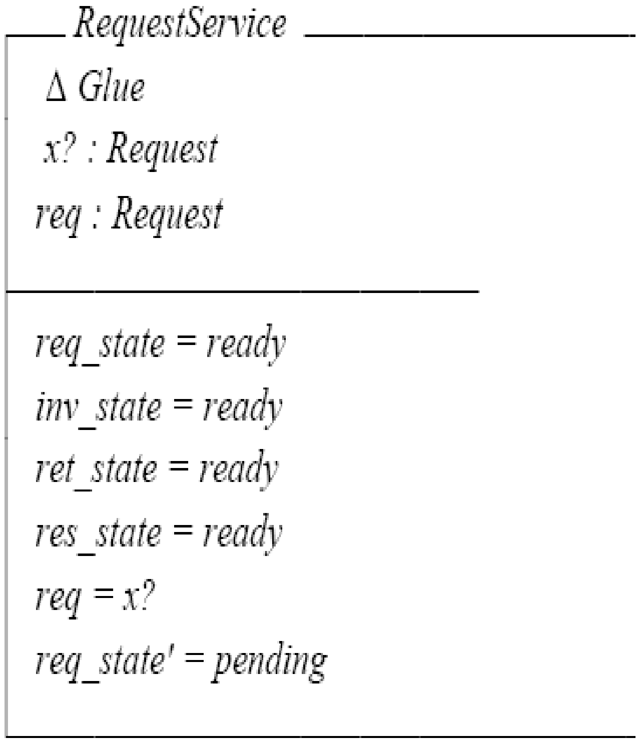
req_state : State
inv_state: State
ret_state : State
res_state : State



Init

req_state = ready
inv_state = ready
ret_state = ready
res_state = ready

Glue: *The methods*



RequestService

Δ *Glue*

$x? : Request$

$req : Request$

$req\_state = ready$

$inv\_state = ready$

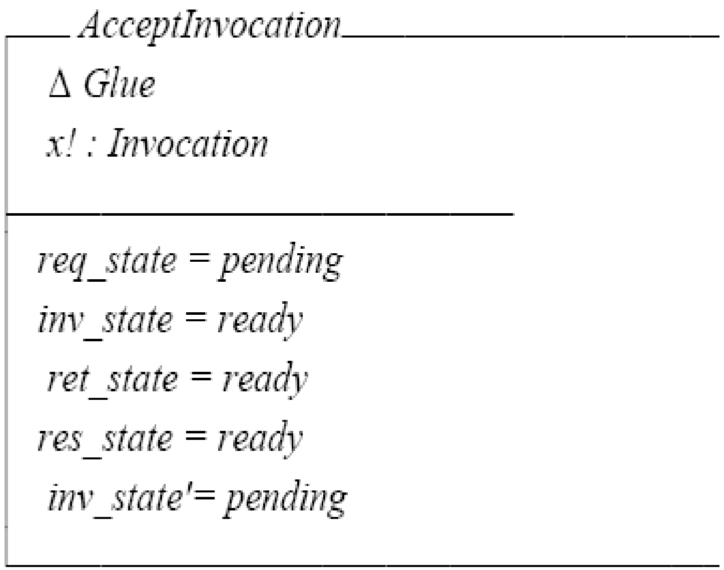
$ret\_state = ready$

$res\_state = ready$

$req = x?$

$req\_state' = pending$

The glue allows the client to call a service



AcceptInvocation

Δ *Glue*

$x! : Invocation$

$req\_state = pending$

$inv\_state = ready$

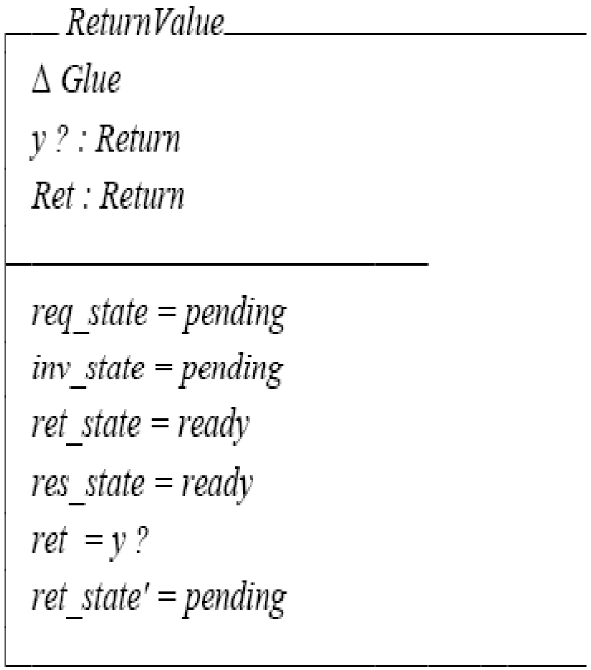
$ret\_state = ready$

$res\_state = ready$

$inv\_state' = pending$

The glue allows the server to accept an invocation

Glue: *The methods (cont.)*



ReturnValue

Δ *Glue*

y ? : Return

Ret : Return

req_state = pending

inv_state = pending

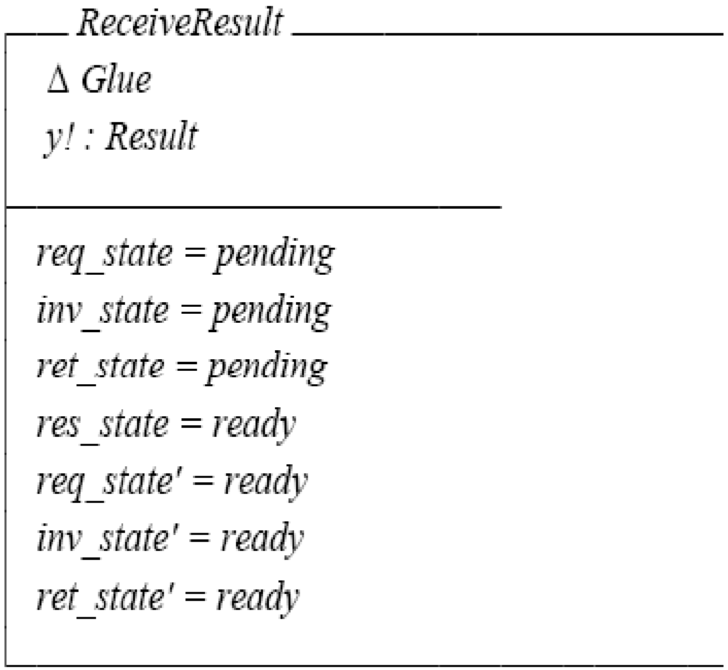
ret_state = ready

res_state = ready

ret = y ?

ret_state' = pending

The glue allows the server to return a value



ReceiveResult

Δ *Glue*

y! : Result

req_state = pending

inv_state = pending

ret_state = pending

res_state = ready

req_state' = ready

inv_state' = ready

ret_state' = ready

The glue allows the client to receive the result

C-S_Connector Behavior

- Parallel composition of roles and glue.

C-S_ConnectorBehaviour = Client_Role || Glue || Server_Role

Ports

- In our example ports are identical to roles, since our client server provides just one service.

Attachment of ports as roles

Attachement =

CS_ConnectorBehaviour [Client_Port / Client_Role ;
Server_Port / Server_Role]

Conclusion and future work

- Look for a unified method of refinement for the integrated notation (not necessarily process based).
- Tackle the problem of verification.

References

- R. Allen and D. Garlan, A Formal Basis for Architectural Connection, 1998
- G. Abowd, R. Allen and Garlan, Formalizing style to understand descriptions of software architectures, 1994
- M. Shaw and D. Galan, Formulations and Formalisms in Software Architecture, 1996
- G. Smith and J. Derrick, specification ,refinement and verification of concurrent systems – an integration of Object-Z and CSP, 2001
- G. Smith and J. Derrick, Refinement and Verification of Concurrent systems specified in Object-Z and CSP, 1997
- G. Smith and J. Derrick, Abstract Specification in Object-Z and CSP, 2002
- J. Derrick and G. Smith, Structural Refinement in Object-Z / CSP, 2000
- G. Smith, A Semantic Integration of Object-Z and CSP for the Specification of Concurrent Systems, 1998
- J.L. Fiadeiro, A. Lopes and M. Wermelinger, A Mathematical Semantics for Architectural Connectors, 2002
- T. McComb and G. Smith, Architectural Design in Object-Z, 2004
- T. Bures, Generating Connectors for Homogeneous and Heterogeneous Deployment, 2006