
Behavioural Contracts for Components

Cyril Carrez

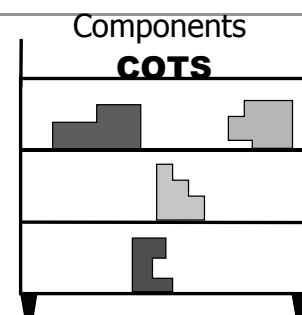
01/03/2004



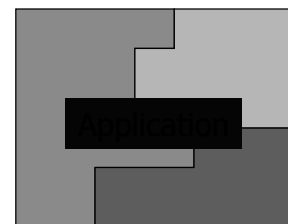
Design by Assembly

- ADL (90's)
 - components
 - connectors
 - configuration
- UML 2.0 (2003)

Classification
[Medvidovic & Taylor]

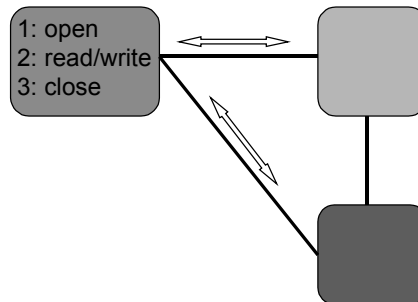


- Behavioural typing with explicit types
 - Regular types [Nierstrasz]
 - «non understood message» [Najm et al.]
- Contracts
 - Design by Contract [Meyer]
 - Classification [Beugnard et al.]
 - Syntactic / **behaviour** (pre/post) / **synchronisation** / QoS



Framework of the study

- Components
 - specification + code
- *Non uniform services*
- Dynamic links



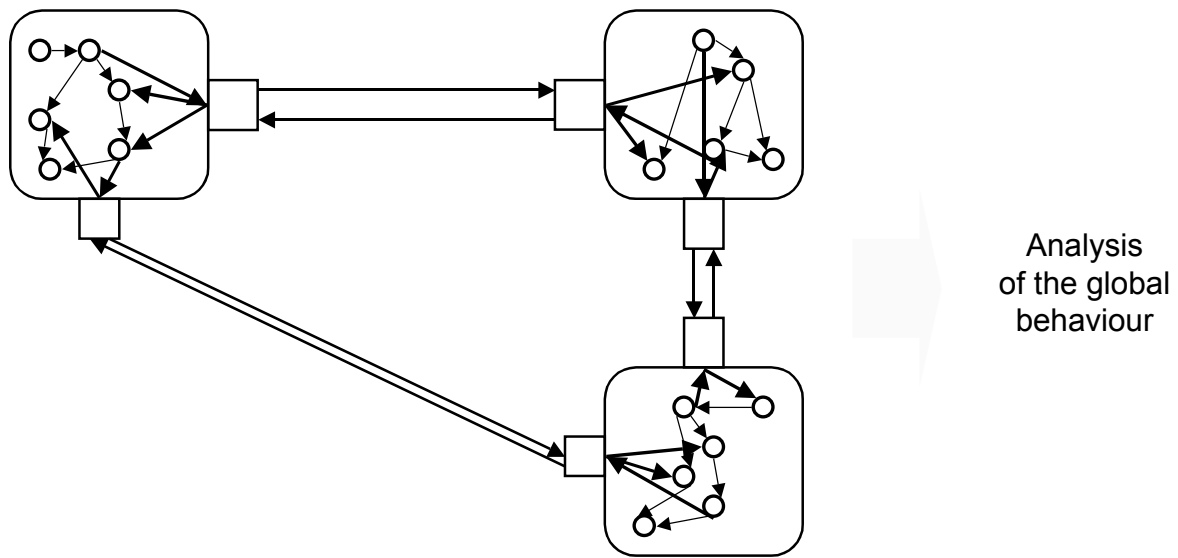
Objectives

- Safety properties: no external deadlock
- Liveness properties: messages will be consumed

Roadmap

- The approach
- Interface language
- Component semantics
- Contract respect
- Sound assembly
- Conclusion & Perspectives

Approaches: Darwin, Wright,...

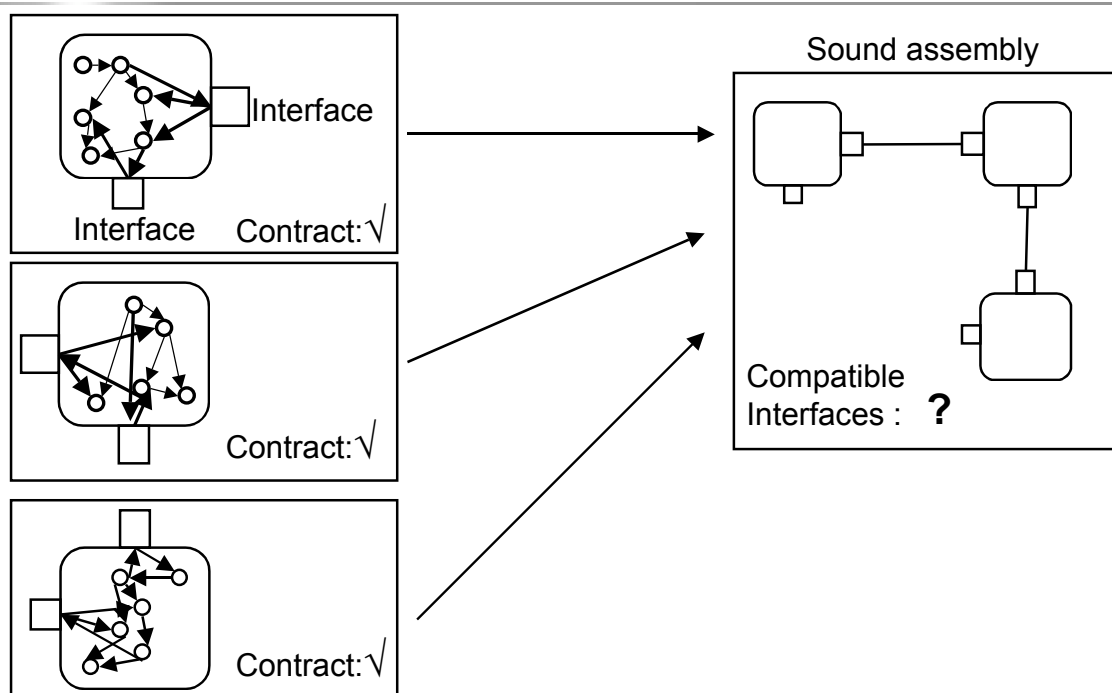


01/03/2004

Behavioural Contracts for Components

5

Our approach

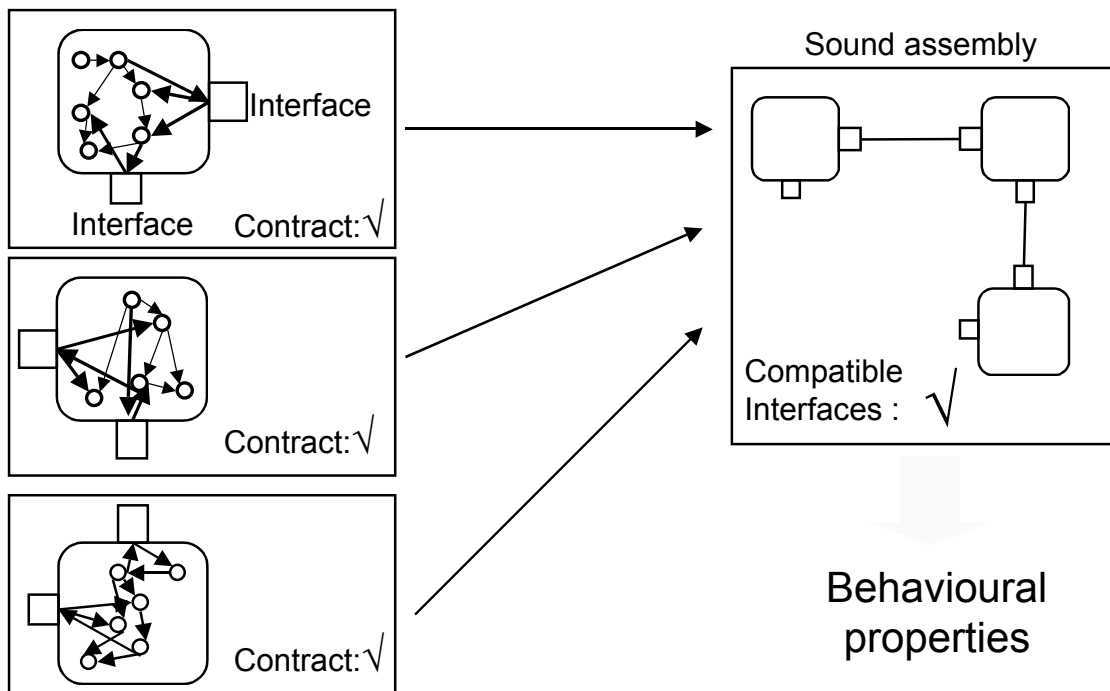


01/03/2004

Behavioural Contracts for Components

6

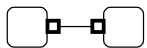
Our approach



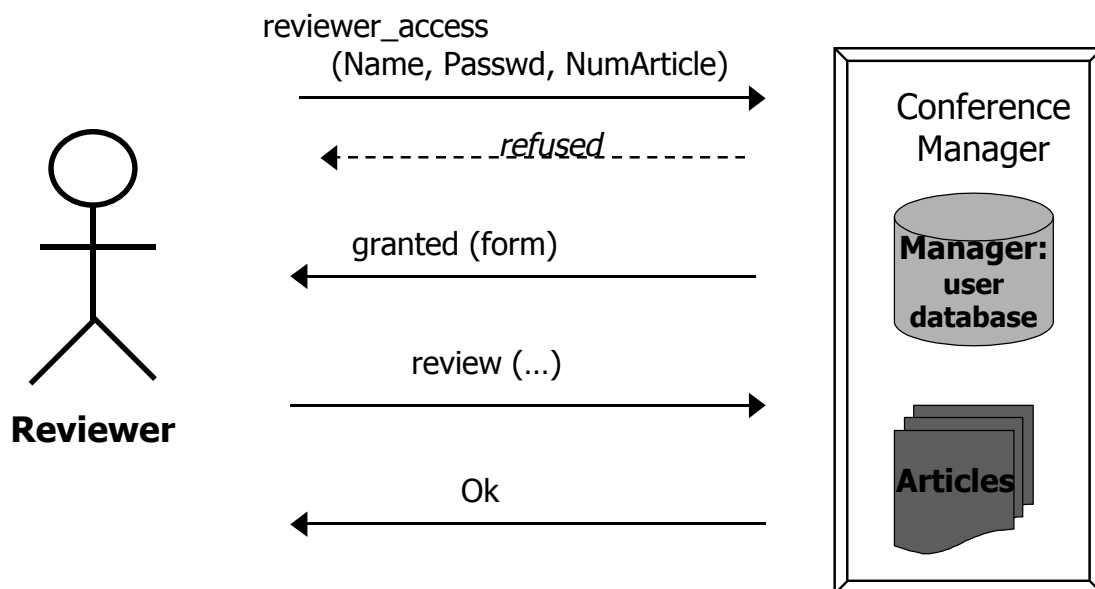
01/03/2004

Behavioural Contracts for Components

7



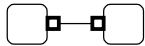
Interface types: example



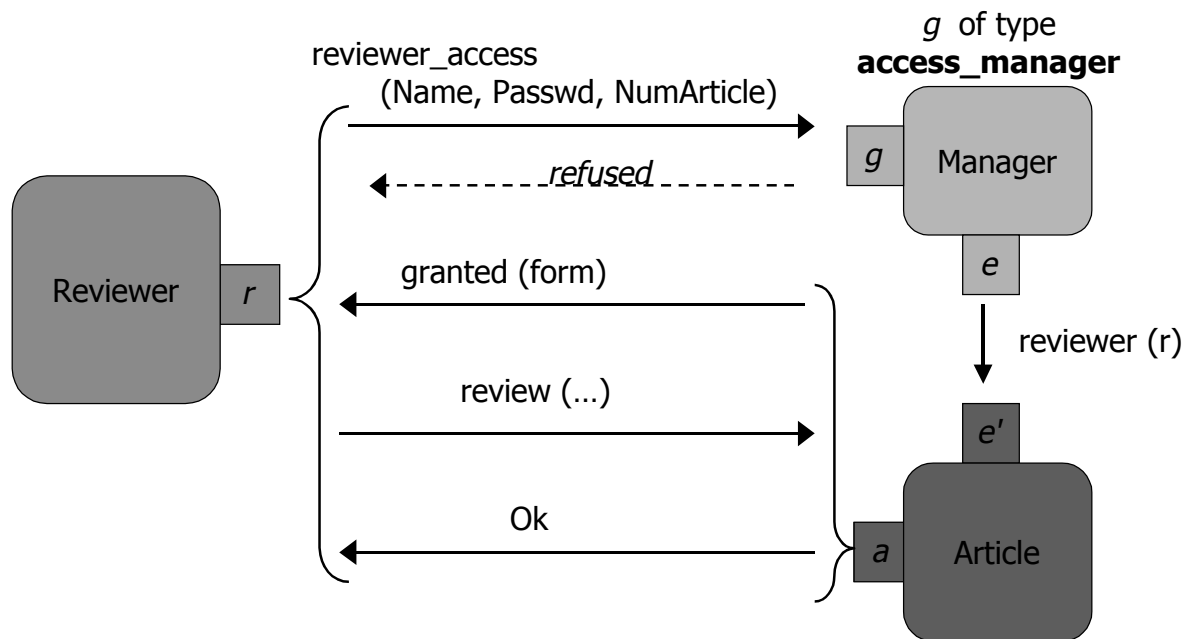
01/03/2004

Behavioural Contracts for Components

8



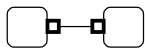
Interface types: example



01/03/2004

Behavioural Contracts for Components

9



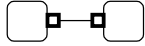
Example: Type access_manager

- **access_manager** =
 may ? [reviewer_access (string,string,integer);
 must ! [refused; **0**
 + granted (strings); **reviewer_manager**]]
- **reviewer_manager** =
 must ? [review (strings); **must** ! [Ok; **reviewer_manager_chg**
 + error; **reviewer_manager**]]
- **reviewer_manager_chg** =
 may ? [review (strings); **must** ! [Ok; **reviewer_manager_chg**
 + error; **reviewer_manager_chg**]]

01/03/2004

Behavioural Contracts for Components

10



Example: Type access_manager

allowed: you can send, I guarantee the reception

- **access_manager** =
 may ? [reviewer_access (string,string,integer);
 must ! [refused; **0**
 + granted (strings); **reviewer_manager**]]

You must send

obligation: I must send

- **reviewer_manager** =
 must ? [review (strings); **must !** [Ok; **reviewer_manager_chg**
 + error; **reviewer_manager**]]
- **reviewer_manager_chg** =
 may ? [review (strings); **must !** [Ok; **reviewer_manager_chg**
 + error; **reviewer_manager_chg**]]



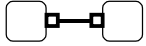
Compatibility: $Comp(I, J)$

| $J \backslash I$ | must ? | may ? | must ! | may ! | 0 |
|------------------|---------------|--------------|---------------|--------------|----------|
| must ? | | | ✓ | | |
| may ? | | ✓ | ✓ | ✓ | ✓ |
| must ! | ✓ | ✓ | | | |
| may ! | | ✓ | | | |
| 0 | | ✓ | | | ✓ |

$$Comp(mod_I ! [\Sigma_k M_k ; I_k], mod_J ? [\Sigma_l M_l ; J_l]) =_{\text{def}}$$

$$Comp_{\text{mod}}(mod_I !, mod_J ?) \wedge (\forall k, \exists l : Comp_{\text{msg}}(M_k, M_l) \wedge Comp(I_k, J_l))$$

$$Comp_{\text{msg}}(M_! (I_i), M_?(J_i)) =_{\text{def}} M_! = M_? \wedge \forall i, I_i \leq J_i$$



Compatibility: *Comp* (I, J)

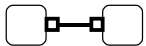
| | must ? | may ? | must ! | may ! | 0 |
|--------|--------|-------|--------|-------|---|
| must ? | | | √ | | |
| may ? | | √ | √ | √ | √ |
| must ! | √ | √ | | | |
| may ! | | √ | | | |
| 0 | | √ | | | √ |

- reviewer_manager =
must ? [review (strings); **must !** [Ok; reviewer_manager_chg
+ error; reviewer_manager]]
reviewer_manager_chg = **may ?** [...]
- enter_review =
must ! [review (strings); **must ?** [Ok; 0
+ error; enter_review]]

01/03/2004

Behavioural Contracts for Components

13



Compatibility: *Comp* (I, J)

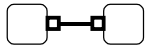
| | must ? | may ? | must ! | may ! | 0 |
|--------|--------|-------|--------|-------|---|
| must ? | | | √ | | |
| may ? | | √ | √ | √ | √ |
| must ! | √ | √ | | | |
| may ! | | √ | | | |
| 0 | | √ | | | √ |

- reviewer_manager =
must ? [review (strings); **must !** [Ok; reviewer_manager_chg
+ error; reviewer_manager]]
reviewer_manager_chg = **may ?** [...]
- enter_review =
must ! [review (strings); **must ?** [Ok; 0
+ error; enter_review]]

01/03/2004

Behavioural Contracts for Components

14



Compatibility: $Comp(I, J)$

| | must ? | may ? | must ! | may ! | 0 |
|--------|--------|-------|--------|-------|---|
| must ? | | | ✓ | | |
| may ? | | ✓ | ✓ | ✓ | ✓ |
| must ! | ✓ | ✓ | | | |
| may ! | | ✓ | | | |
| 0 | | ✓ | | | ✓ |

- reviewer_manager =
must ? [review (strings); **must !** [Ok; reviewer_manager_chg
+ error; reviewer_manager]]
- reviewer_manager_chg = **may ?** [...]
- enter_review =
must ! [review (strings); **must ?** [Ok; 0
+ error; enter_review]]

01/03/2004

Behavioural Contracts for Components

15



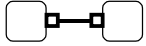
Subtyping: $T \leq S$

- Compatibility: sent message \leq received message
- receivings:
 - $mod ? M_1 + M_2 + M_3 \leq mod ? M_1 + M_2$
 - contra-variant: $M(I) \leq M(J) \Leftrightarrow J \leq I$
- sendings:
 - $mod ! M_1 \leq mod ! M_1 + M_2$
 - co-variant: $M(I) \leq M(J) \Leftrightarrow I \leq J$
- modalities:
 - **may ?** \leq **must ?** – **may ?** \leq **0** – **may ?** \leq **may !**
 - **must !** \leq **may !** – **0** \leq **may !**

01/03/2004

Behavioural Contracts for Components

16

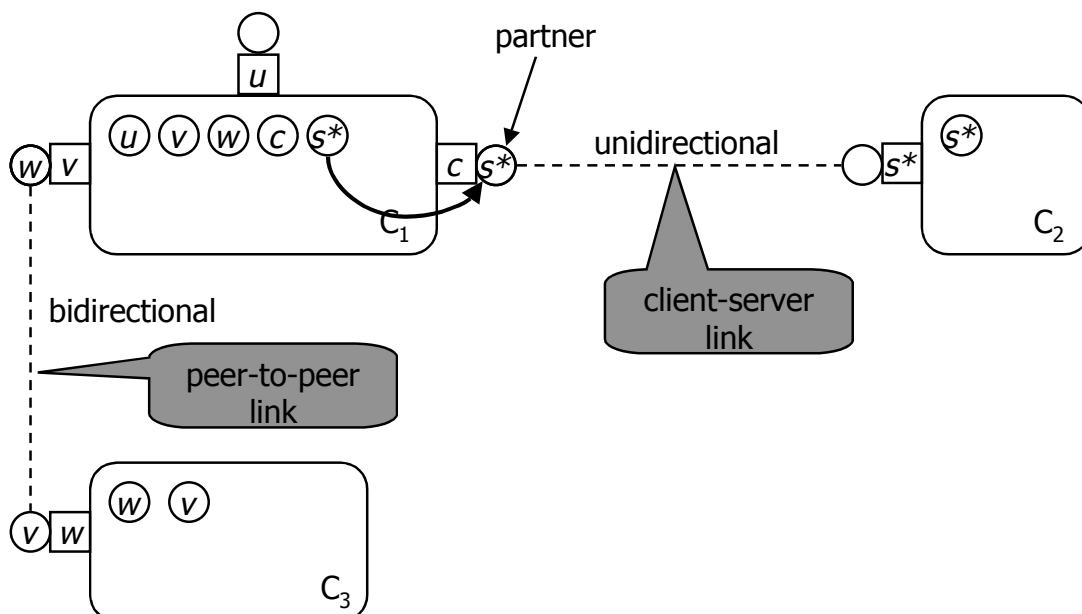


Properties of the subtypes

- \leq is transitive
- Subtype can replace super-type
 - $Comp(I, S) \ \& \ (T \leq S) \ \Rightarrow \ Comp(I; T)$
- Greater compatible super-type:
 - dual: $J^D =_{\text{def}} J$ with reversed sendings and receivings
 - $Comp(I, J) \Leftrightarrow I \leq J^D$
- Demonstrations
 - by induction on the structure of the types

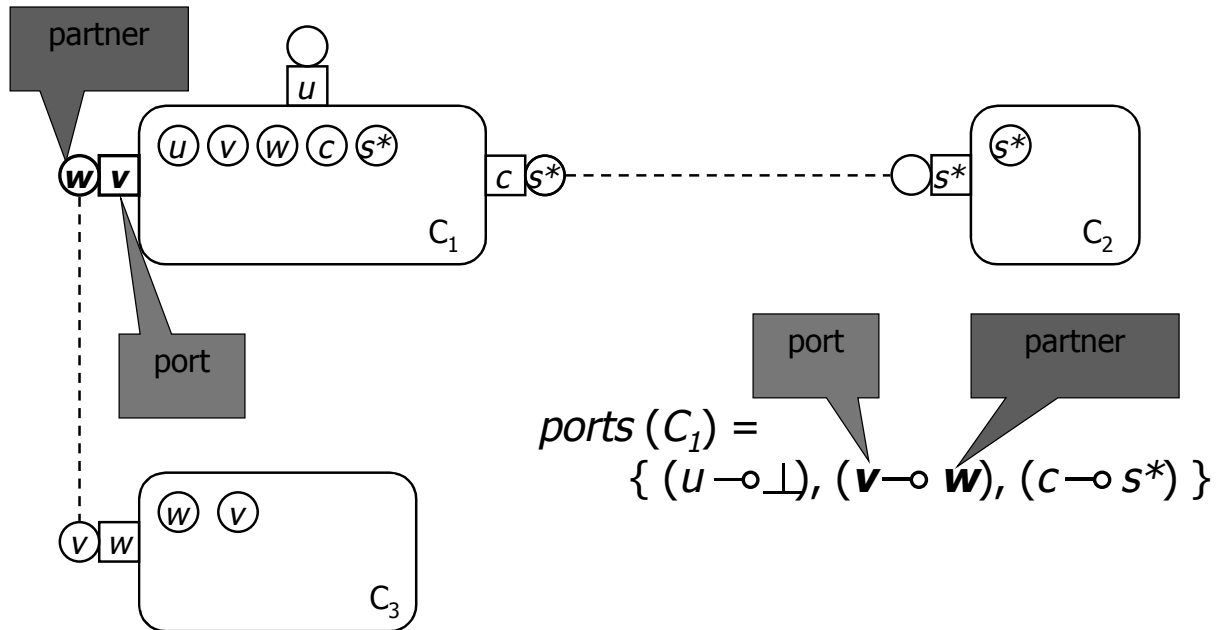


Component model





Component model



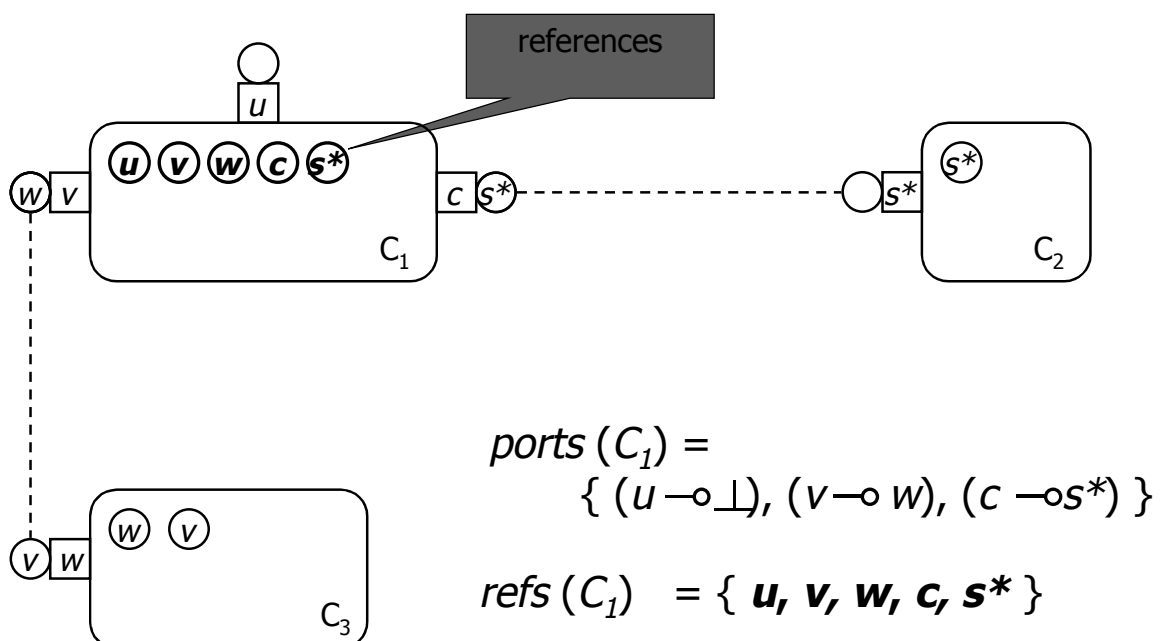
01/03/2004

Behavioural Contracts for Components

19



Component model



01/03/2004

Behavioural Contracts for Components

20



Component model: ports

- Model based on observation of ports
- State of a port : $u\rho^\sigma$

$$- \rho = \text{action} = \begin{cases} ! & u \text{ is in a sending state} \\ ? & u \text{ is in a receiving state} \\ \mathbf{0} & u \text{ has no action} \end{cases}$$

$$- \sigma = \text{activity} = \begin{cases} \mathbf{a} & u \text{ is active} \\ \mathbf{s} & u \text{ is suspended} \\ \mathbf{i} & u \text{ is idle} \end{cases}$$

- Example:
 - $u ?^{\mathbf{a}}$ = active in receiving $u !^{\mathbf{s}}$ = suspended in sending



Component model: threads

- Multi-threaded components
- Dependencies between ports: $x \succ \rightarrow y$
 - activity of x is suspended until y terminates or becomes idle
- A thread is a chain (*head, queue*)
 - *head*: current active port,
 - *queue*: ordered sequence of suspended ports
 - can dynamically grow/diminish

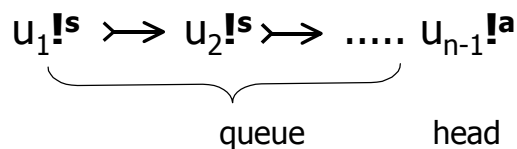
$$u_1 !^{\mathbf{s}} \succ \rightarrow u_2 !^{\mathbf{s}} \succ \rightarrow \dots u_{n-1} !^{\mathbf{s}} \succ \rightarrow u_n ?^{\mathbf{a}}$$

queue
head



Component model: threads

- Multi-threaded components
- Dependencies between ports: $x \succ \rightarrow y$
 - activity of x is suspended until y terminates or becomes idle
- A thread is a chain (*head, queue*)
 - *head*: current active port,
 - *queue*: ordered sequence of suspended ports
 - can dynamically grow/diminish

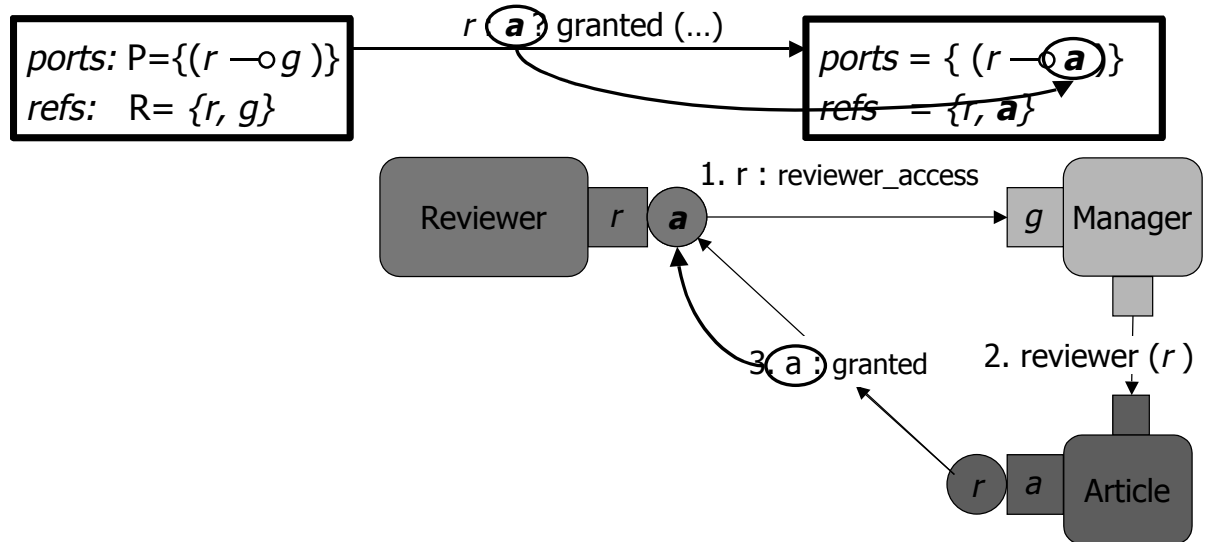


Component semantic

- Component: $B (P, R, T)$
 - state \swarrow
 - ports, references, threads \nwarrow
- Operational semantic
 - $B (P, R, T), Com \longrightarrow B' (P', R', T'), Com'$
 - async. com.: \downarrow
 - Fifo queues \downarrow
- 11 Rules:
 - creation / removal of ports
 - binding
 - (de)activation of ports (idle, active, suspended)
 - sending/receiving messages



Example: RECV for Reviewer component



$$T' = T[u\rho/u?] \quad R' = R \cup \{\text{refs}(\tilde{v}), u''\} - \{u' \mid (u \rightarrow o u') \wedge \text{peer}(u')\}$$

$$Com' = Com[u\rho] \quad \boxed{P' = P[u \rightarrow o u''] \text{ si } \text{peer}(u)}$$

$$B(P, R, T), Com \xrightarrow{u:u''?M(\tilde{v})} B'(P', R', T'), Com'$$



Some other rules

$$\text{C-BIND} \frac{P' = P[u \rightarrow o v]}{B(P, R, T), Com \xrightarrow{\text{bind}(u \rightarrow o v)} B'(P', R, T), Com} \quad \square$$

$$\square \triangleq (u \rightarrow o \perp) \wedge \boxed{T(u) = !^{a,i}} \wedge v \in R \wedge \boxed{(\text{peer}(v) \Rightarrow v \notin \text{CoDom}(P))}$$

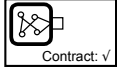
-only sending ports, not suspended
-peer reference is attached to 1 port

$$\text{C-ACTV} \frac{T' = T[u \mapsto v]}{B(P, R, T), Com \xrightarrow{\text{actv}(u \mapsto v)} B'(P, R, T'), Com} \quad T(u) = !^a \wedge \boxed{T(v) = !^i}$$

-a port cannot suspend on a receiving port

$$\text{C-SEND} \frac{\boxed{R' = R - \text{peer}(\tilde{v} \cup \{u\})} \quad T' = T[u\rho/u!] \quad Com' = Com[u' \triangleleft u : M(\tilde{v})]}{B(P, R, T), Com \xrightarrow{u:u'!M(\tilde{v})} B'(P, R', T'), Com'} \quad \Delta$$

-peer reference is private: known only to the partner



Component and contracts

Contractual component: $B(\dots), \tilde{C}$

– correct behaviour

$$\frac{\tilde{C} \xrightarrow{\alpha} \tilde{C}' \quad B(\dots) \xrightarrow{a} B'(\dots) \quad a : \alpha}{B(\dots), \tilde{C} \xrightarrow{a:\alpha} B(\dots), \tilde{C}'}$$

– unauthorized transition

$$\frac{\tilde{C} \not\xrightarrow{\alpha} \tilde{C}' \quad B(\dots) \xrightarrow{a} B'(\dots) \quad a : \alpha}{B(\dots), \tilde{C} \xrightarrow{a:\alpha} Error}$$

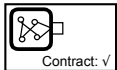
– missing required transition

$$\frac{\tilde{C} \xrightarrow{\alpha} \tilde{C}' \quad B(\dots) \not\xrightarrow{a} B'(\dots) \quad a : \alpha \text{ mod } (\alpha) = \mathbf{must}}{B(\dots), \tilde{C} \xrightarrow{a:\alpha} Error}$$

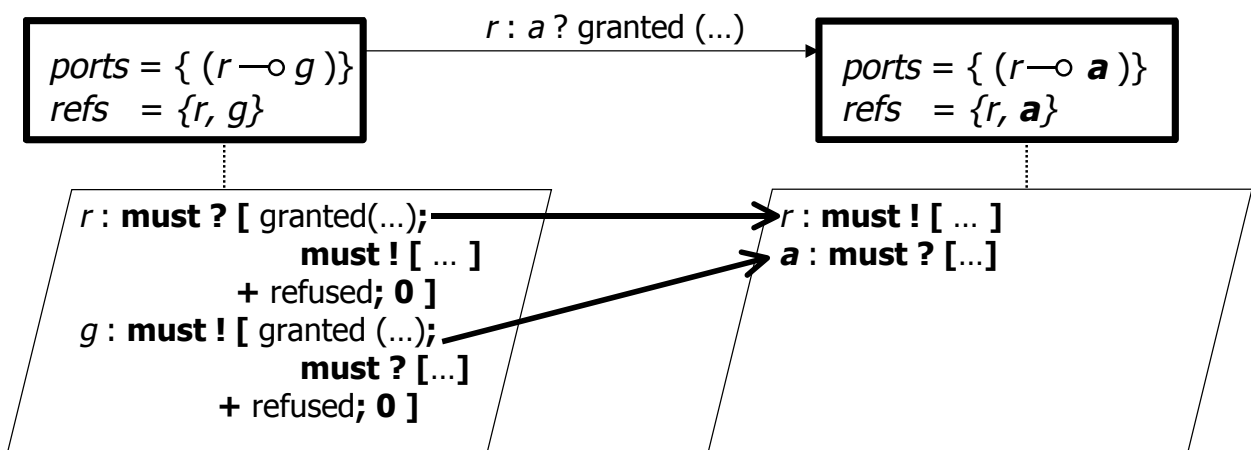
01/03/2004

Behavioural Contracts for Components

27



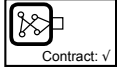
Example: RECV for Reviewer component



$$u:T \equiv \text{mod } ? M_{\Sigma}$$

$$u':T' \equiv \text{mod}' ! M'_{\Sigma} \quad B(P, R, T) \xrightarrow{u:u' ? m_k} B'(P', R', T')$$

$$(B(P, R, T), \tilde{C}) \xrightarrow{u:u' ? m_k} (B'(P', R', T'), \boxed{\tilde{C}[u:T_k/T, u':T'_k/u':T']} \Leftarrow \tilde{v}':\tilde{U}'_k) \blacktriangle$$



Some other rules

$$\text{BIND} \frac{u:T \quad v:S \quad B(P, R, T) \xrightarrow{\text{bind}(u-ov)} B'(P', R, T)}{(B(P, R, T), \tilde{C}) \xrightarrow{\text{bind}(u-ov)} (B'(P', R, T), \tilde{C})} \text{Comp}(T, S)$$

$$\text{BIND-ERR} \frac{u:T \quad v:S \quad B(P, R, T) \xrightarrow{\text{bind}(u-ov)} B'(P', R, T)}{(B(P, R, T), \tilde{C}) \rightarrow \text{Error}} \neg \text{Comp}(T, S)$$

$$\text{RECV-ERR} \frac{u:T \equiv \text{mod } ?[*]M_\Sigma \quad \forall k, B(P, R, T) \xrightarrow{u:u'/?m_k} B'(P', R', T')}{(B(P, R, T), \tilde{C}) \rightarrow \text{Error}}$$

$$\text{RECV-UN} \frac{u:T \equiv \text{mod } ? M_\Sigma \quad B(P, R, T) \xrightarrow{u:u'/?m_k} B'(P', R', T')}{(B(P, R, T), \tilde{C}) \xrightarrow{u:u'/?m_k} (B'(P', R', T'), \tilde{C}[u:T_k/T] \leftarrow \boxed{u':T_k^D}, \tilde{v}:\tilde{U}_k)} \blacktriangle \wedge (u \multimap \perp)$$

-RECV from unknown partner: take the greater type



Sound assembly of components

- Component honouring a contract
 - B is well-typed: $B(P, R, T), \tilde{C}$ never leads to *Error*
- Assembly of components:

$$\mathcal{A} = \{ (B_1(P_1, R_1, T_1), \tilde{C}_1), \dots, (B_n(P_n, R_n, T_n), \tilde{C}_n), \text{Com} \}$$

- reference closed
- only client/server and peer-to-peer bindings
- all ports are active and independent
- Sound assembly:
 - all components respect their contract
 - ports bound to each other are compatible



Properties

Soundness is maintained through evolution

– a sound configuration of components never leads to *Error*

$$\forall C : \mathcal{A} \longrightarrow * C, C \not\rightarrow Error$$

All the messages are eventually consumed

$$\forall u, v, i, M : (u \circ v) \in P_i, C \xrightarrow{u:v!M} C'$$

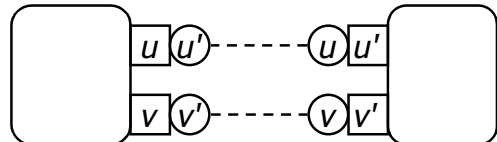
$$\Rightarrow \exists C'', C''' \text{ such that } C' \longrightarrow * C'' \xrightarrow{v:u?M} C'''$$



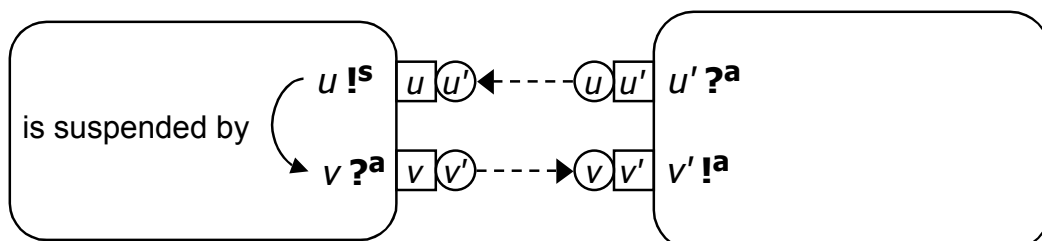
External deadlock

- During assembly : no verification of the global behaviour

- u and u' types are compatible
- v and v' types are compatible



- During execution :

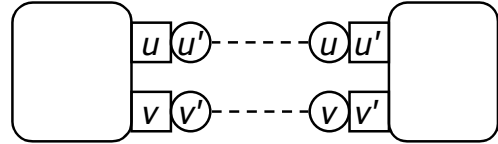




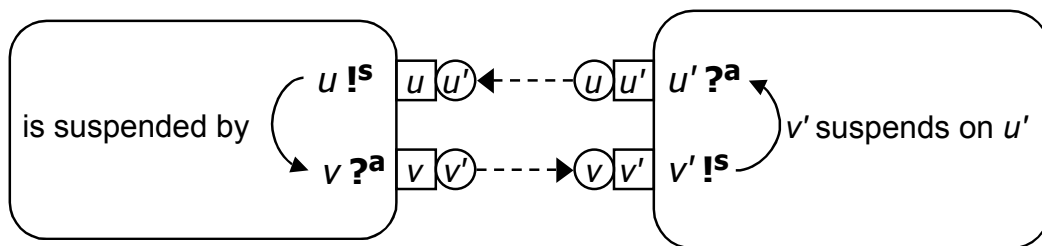
External deadlock

- During assembly : no verification of the global behaviour

- u and u' types are compatible
- v and v' types are compatible



- During execution :



01/03/2004

Behavioural Contracts for Components

33



Property: external deadlock freeness

- A port cannot suspend on a receiving port

- external deadlock:

$$- u \mathcal{S} v =_{\text{def}} u \xrightarrow{\quad} v \quad \vee \quad u \dashrightarrow v \quad (\dashrightarrow \text{external dependency})$$

- $\text{Ext_deadlock}(C) =_{\text{def}}$

$$\exists (u_i)_{1..n} \in C \text{ such that } \forall k < n : u_k \mathcal{S} u_{k+1} \wedge u_n \mathcal{S} u_1$$

- Demonstration (deadlock freeness):

- by induction & Reductio ad absurdum

01/03/2004

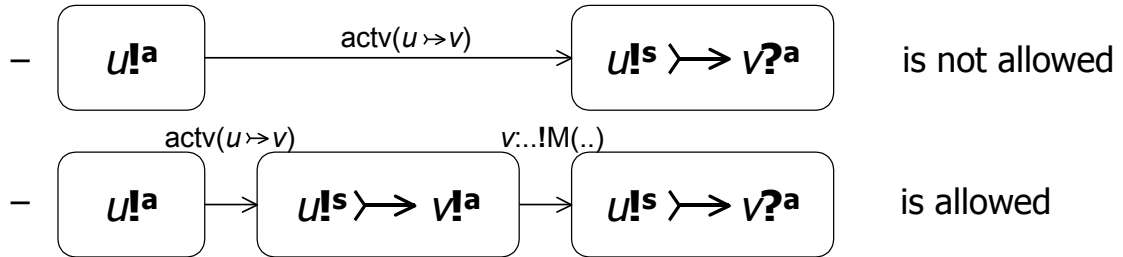
Behavioural Contracts for Components

34



Constraints on the component

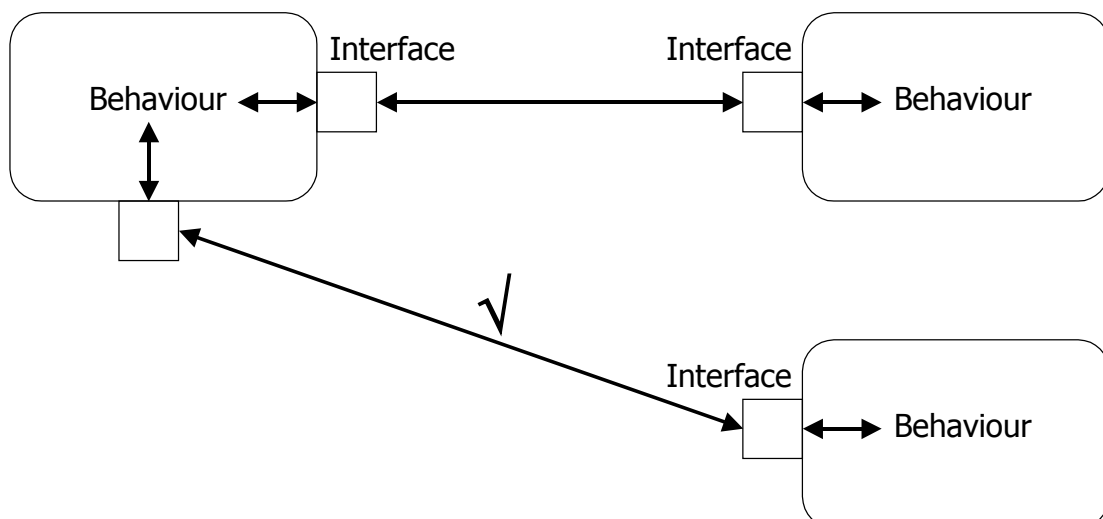
- a port cannot suspend on a receiving port:



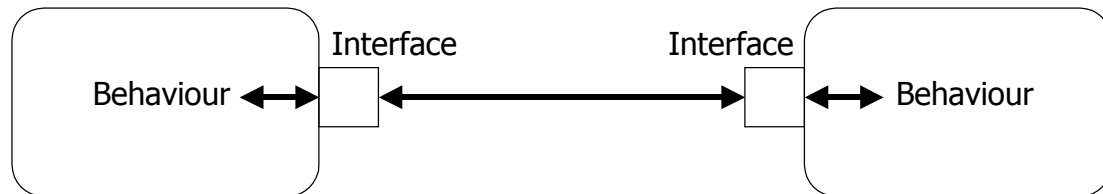
- a receiving port cannot be suspended: $u?^s$ forbidden
- bindings: only sending & (active or idle) ports: $u!^{a,i}$
- a 'must !' is not suspended by a 'may ?'
- unbind is not allowed
- [nonreentrant servers]

Application

- Sound extension of running application



Conclusion



- Contract conformance: \longleftrightarrow verification during compilation
- Compatible interfaces: \longleftrightarrow verification during deployment
- Properties of a sound assembly
 - safety: a configuration never leads to *Error*
 - safety: external deadlock freeness
 - liveness: all sent message are eventually consumed

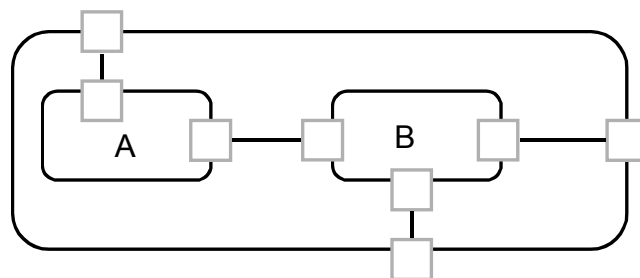
01/03/2004

Behavioural Contracts for Components

37

Perspectives

- Interfaces: infinite state machines
- Integration to existing component platforms
- UML Profile
- Composite components & delegation:



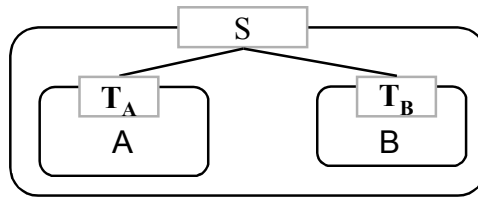
01/03/2004

Behavioural Contracts for Components

38

Future Work

- Application to UML2.0: multiple delegation



- Application to a language
- From interface contracts to component contracts
- Extension to timed interfaces
- Application to PATS!!

Get the slides!
www.cyril-carrez.net
www.item.ntnu.no/~carrez