2023
Ingegneria del software: teoria parte 3

Giorgio Bruno

Dip. Automatica e Informatica
Politecnico di Torino
email: giorgio.bruno@polito.it

Argomenti

Notazione *BPN* (Business Process Notation) per la modellazione dei processi di business

Tipi di task

Scelte

Gestione del dataflow

Due scenari per i processi: centralizzato e distribuito (B2B)

Modelli di collaborazione

Modelli informativi delle collaborazioni

Notazione BPN

La notazione include tre tipi di modelli:

il modello di processo: contiene task di vario tipo con il dataflow delle entità gestite;

il modello informativo: definisce i tipi delle entità con attributi, relazioni e invarianti;

il modello delle collaborazioni, nel caso di processi B2B.

Modelli di processi

La struttura di base è quella delle reti di Petri, ma il significato di posti e transizioni è notevolmente esteso.

I posti contengono token che si riferiscono ad entità presenti nel sistema informativo sottostante. Per questo motivo la notazione include un modello informativo.

Le transizioni rappresentano **task** che possono essere eseguiti dal sistema (automatici) o dalle persone coinvolte nel processo (partecipanti).

I partecipanti sono divisi in *ruoli* e i task umani indicano il ruolo richiesto.

Gli effetti dei task sulle entità sono indicati da *post-condizioni*; i vincoli sui task da *pre-condizioni*.

I task sono suddivisi in vari tipi sulla base dei loro effetti sulle entità.

Scenari

1. Centralizzato: un'organizzazione definisce un processo al quale i partner possono partecipare eseguendo direttamente dei task. I ruoli comprendono quindi persone appartenenti all'organizzazione (ruoli interni) oppure ai partner (ruoli esterni).
2. B2B (Business to Business): varie organizzazioni interagiscono tra loro mediante messaggi secondo protocolli di collaborazione. Tali protocolli sono definiti in modelli di collaborazione.

Scenario centralizzato

Procedimento di analisi

Esempi: gestione di ordini, di proposte, di conferenze

Procedimento di analisi in un contesto centralizzato

Analisi dei requisiti

Definizione del modello informativo

- entità (contestuali, di processo, ruoli)
- attributi
- relazioni (categorie), relazioni derivate
- invarianti

Definizione del processo

Esempio: gestione ordini

Requisiti

Un'azienda tratta gli ordini immessi dai suoi clienti. Un ordine si riferisce ad un tipo di prodotto.

Gli ordini hanno un importo; se è <= 1000, l'ordine è accettato automaticamente, altrimenti è esaminato dall'accountMgr (ruolo interno) associato al cliente. L'account manager può accettarlo o respingerlo e il risultato è scritto nello stato dell'ordine. Poi il cliente legge l'ordine per conoscerne lo stato.

Clienti, account manager e tipi di prodotto sono già registrati nel sistema informativo.

Tipi di entità: Cliente, Ordine, Tipo (di prodotto), AccountMgr.

Task: Cliente: immette ordine, legge ordine.

AccountMgr: accetta o respinge ordine (esamina ordine).

System: accetta ordine.

I *nomi dei task* sono generalmente costituiti da un verbo e un complemento oggetto (che può anche essere omesso).

Analisi dei requisiti

Un'azienda tratta gli ordini immessi dai suoi clienti. Un ordine si riferisce ad un tipo di prodotto. Gli ordini hanno un importo.

Un ordine è collegato necessariamente al cliente che l'ha immesso e al tipo al quale si riferisce.

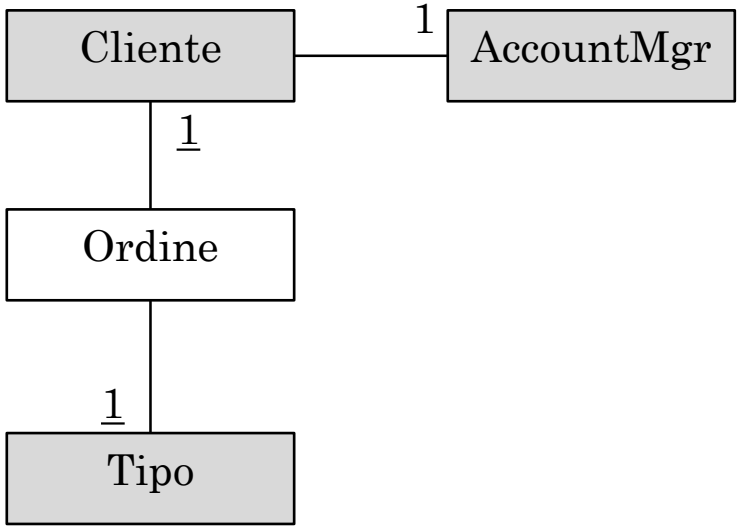
Un ordine è trattato dall'accountMgr del cliente.

Gli ordini hanno un importo; se è <= 1000, l'ordine è accettato automaticamente, altrimenti è esaminato dall'accountMgr associato al cliente.

Modello informativo

In grigio le entità contestuali

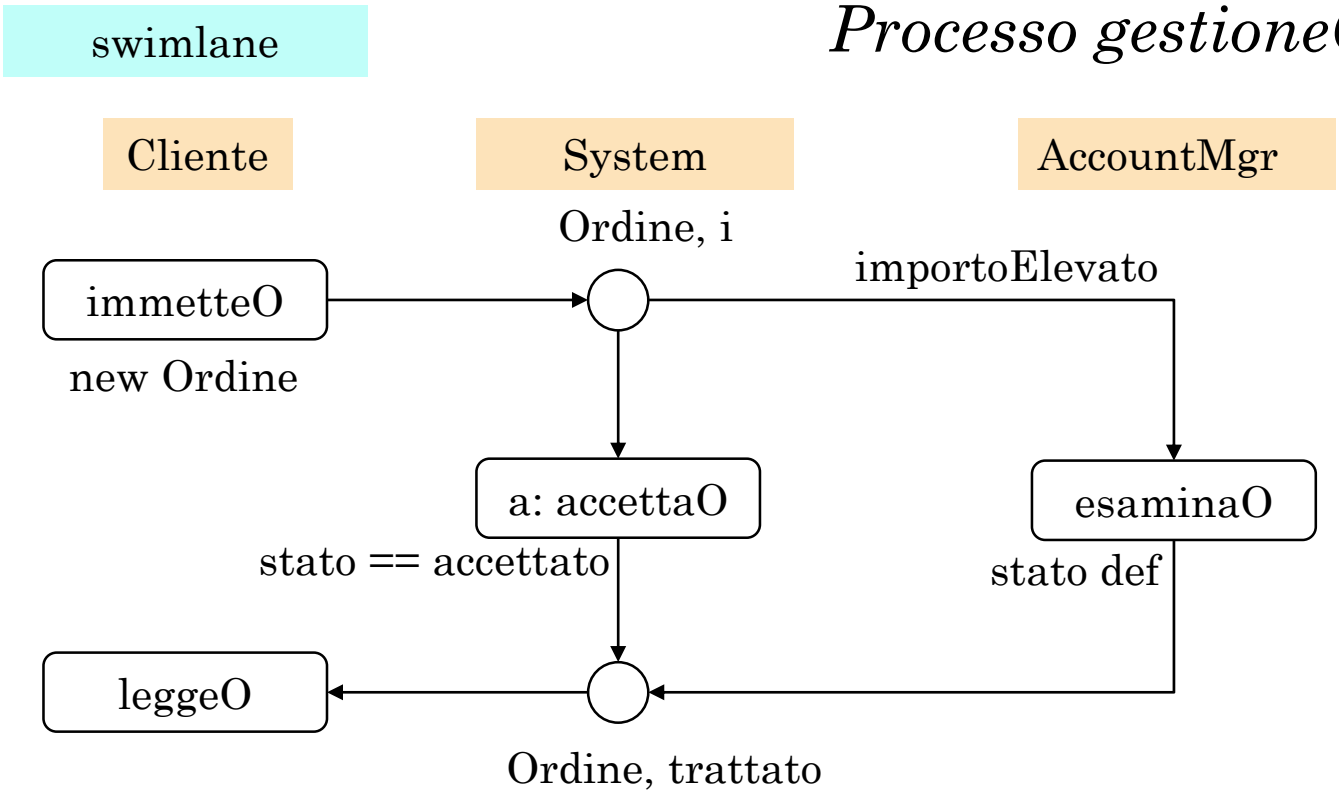
Le uniche entità generate dal processo sono gli ordini.



Attributi
Ordine: int importo; stato (accettato, respinto);
boolean importoElevato = importo > 1000. // serve nel processo

Nota: inizialmente lo stato è null, poi potrà assumere i valori accettato o respinto.

Processo gestioneOrdini

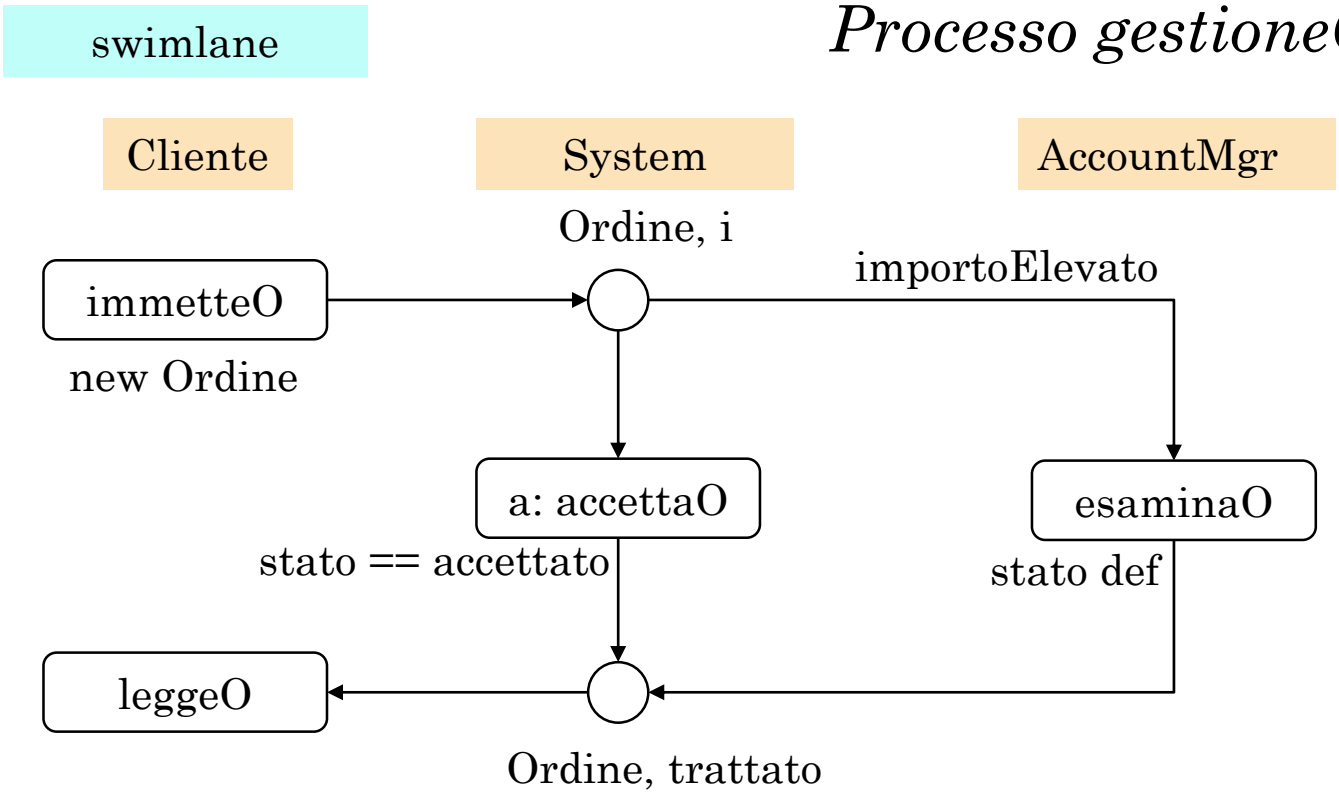


Le swimlane sono partizioni verticali del modello: il ruolo indicato nella swimlane è quello dei task umani contenuti. Il task a:accettaO è automatico.

La *condizione booleana di routing* importoElevato stabilisce quale dei due task (esaminaO, accettaO) sarà eseguito.

Il processo è *singleton*: una sola istanza può gestire tutti gli ordini.

Processo gestioneOrdini



Gli effetti dei task sono espressi mediante post-condizioni, introdotte da *post*, che si può omettere se non vi sono pre-condizioni (introdotte da *pre*).

La post *stato def* precisa che l'effetto dell'esecuzione del task esaminaO è di valorizzare lo stato dell'ordine; lo stato conterrà il valore accettato o respinto.

Tipi di task

entry task

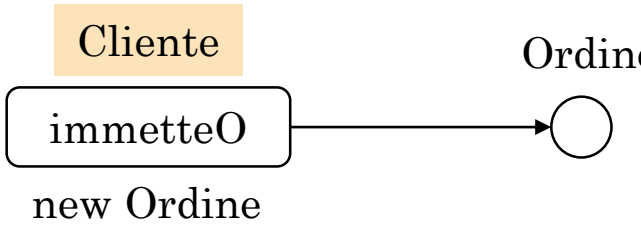
exit task

task passante

scelta semplice strutturata

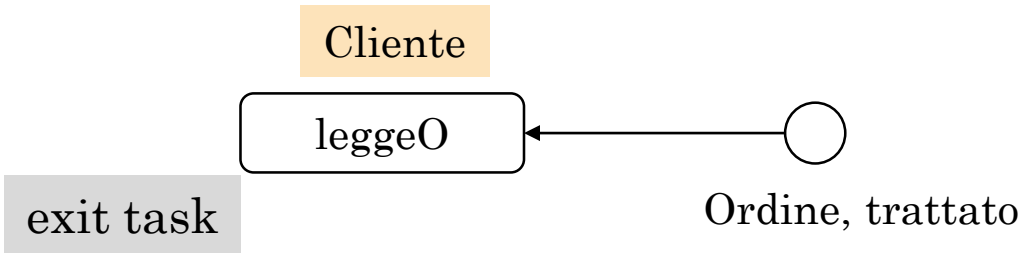
task multi-performer

Tipi di task: entry task, exit task



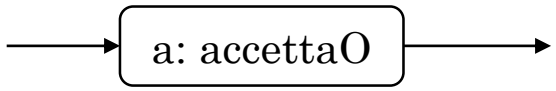
entry task

Un entry task non ha input ed immette una nuova entità nel dataflow.
Il task immetteO può essere eseguito da ciascun cliente quando lo desidera, a meno che non ci sia una pre-condizione.
Un esempio di pre-condizione è il seguente:
il n. di ordini del cliente con stato null deve essere minore di 3
pre: [cliente.ordini(stato == null)] < 3.



exit task

Un exit task non ha output: toglie entità dal dataflow.



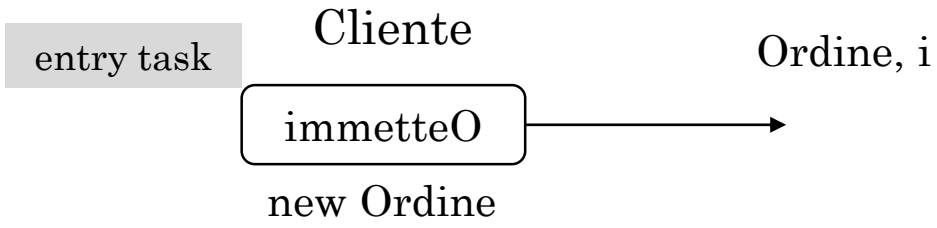
task automatico

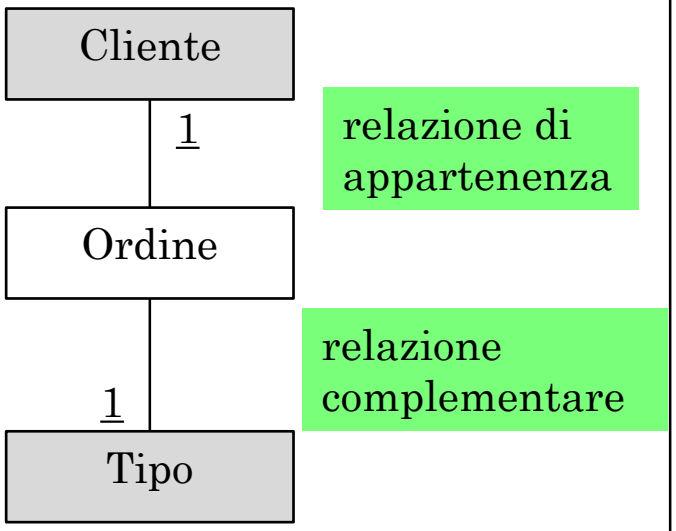
Analisi del testo

Un'azienda tratta gli ordini immessi dai suoi clienti. Un ordine si riferisce ad un tipo di prodotto. Gli ordini hanno un importo.

Interpretazione

Un ordine è collegato necessariamente al cliente che l'ha immesso e al tipo al quale si riferisce.





L'effetto di **immetteO** è la generazione di un nuovo ordine nello stato iniziale (i). *Come conseguenza delle relazioni obbligatorie*, l'ordine è collegato ad un cliente e ad un tipo di prodotto: il cliente è l'esecutore del task, mentre il tipo di prodotto è scelto durante l'esecuzione del task. Anche l'importo, essendo un attributo obbligatorio, è definito durante l'esecuzione del task.

Ordine: int importo.

Chi esamina l'ordine?

Un ordine è trattato dall'account mgr del cliente.

La regola standard di *BPN* per la determinazione del performer di un task è la seguente:

il performer è determinato in base ad una relazione diretta o derivata con l'entità o le entità di input.

Il task *esaminaO* è eseguito dall'account mgr del cliente che ha inserito l'ordine di input. In altre parole, un ordine di input è assegnato per il trattamento *esaminaO* all'account mgr del cliente che l'ha inserito.

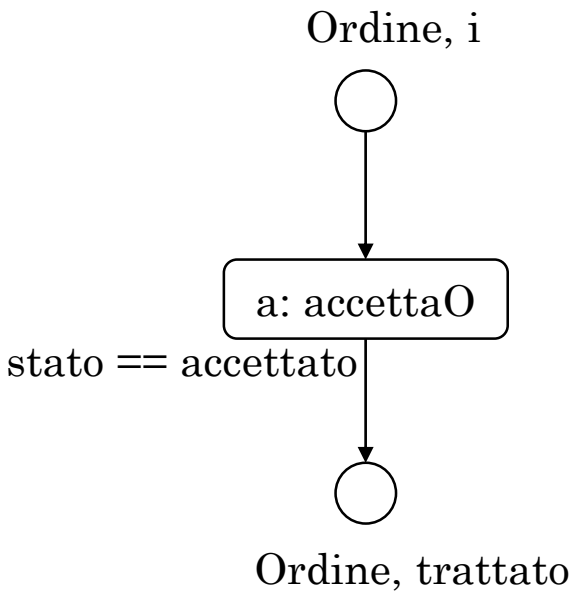
Il performer si ottiene in base alla relazione Ordine – AccountMgr.

Non c'è una relazione diretta quindi occorre inserire la *relazione derivata* seguente: Ordine – AccountMgr = Ordine – Cliente – AccountMgr

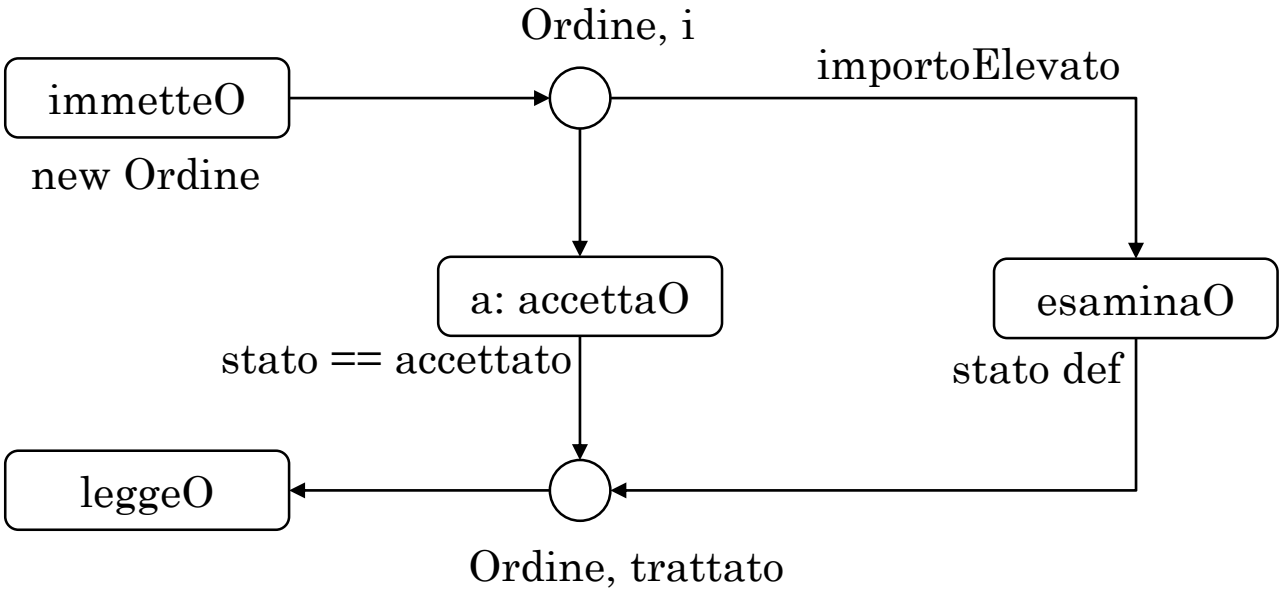
Se il percorso dal tipo dell'input al ruolo del task è univoco, si può considerare implicita.

Tipi di task: task passante

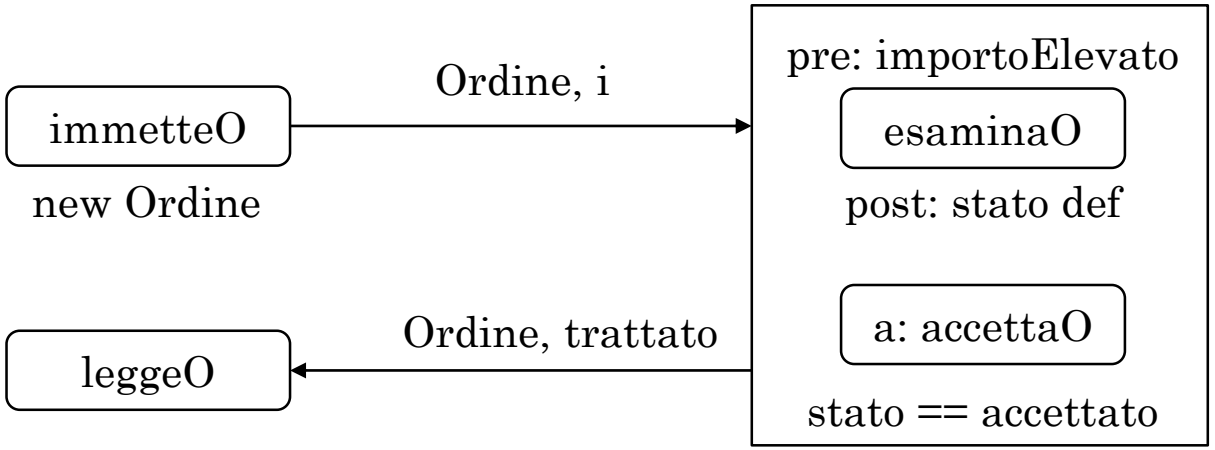
Un task passante ha un solo input e un solo output che sono dello stesso tipo. I task accettaO ed esaminaO sono passanti.



Scelte strutturate o non strutturate



scelta non strutturata



scelta automatica strutturata

Scelte semplici

Scelte semplici non strutturate: simili a free choice con l'aggiunta di condizioni di routing.

Scelte semplici strutturate:

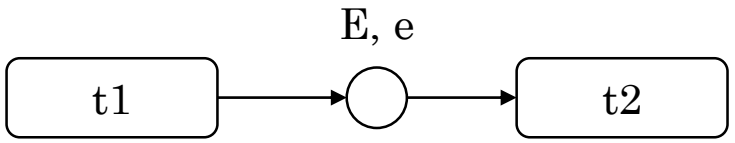
I task sono racchiusi in un blocco che ha un solo input e può avere un solo output. L'input del blocco è l'input dei task interni.

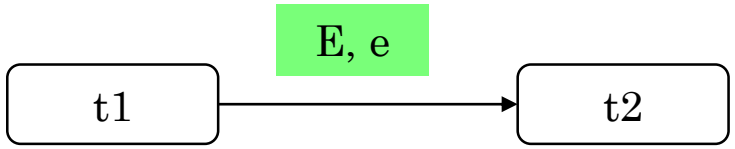
I task possono avere posti di output differenti: se hanno lo stesso posto di output basta un link dal blocco al posto.

Le scelte semplici possono essere automatiche o umane.

Le scelte possono avere un nome, ed es. decisione su ordine.

Posti assorbiti nei link





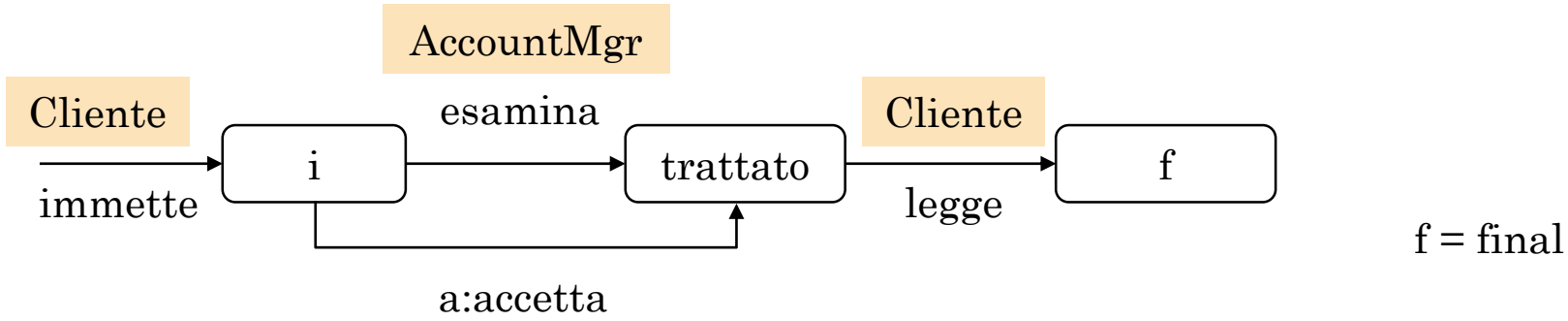
Se il posto *E,e* ha un solo input e un solo output, si può assorbire nel link diretto tra i due task. L'etichetta del posto diventa l'etichetta del link.

Life cycle degli ordini

Il processo gestisce il *life cycle* degli ordini: gli stati o **stadi** (**stages**) mostrati sono *i* (iniziale) e *trattato*.

Il life cycle si può rappresentare con un modello di stato.

Modello di stato di Ordine



Nota: si possono aggiungere condizioni di routing e post-condizioni.

Process engine

I processi sono eseguiti da un'entità software chiamata process engine, che interpreta le loro descrizioni.

Il process engine interagisce con le entità software che implementano i task: web activities (spesso modellate con use cases) per i task umani e servizi per i task automatici.

I task umani sono gestiti tramite **work list**: ogni utente del sistema ha la propria work list che è analoga ad un'agenda. Ogni voce (entry) corrisponde ad un task che l'utente può o deve eseguire.
Con il click su una entry si attiva una web activity che implementa il task scelto ed interagisce con il process engine.

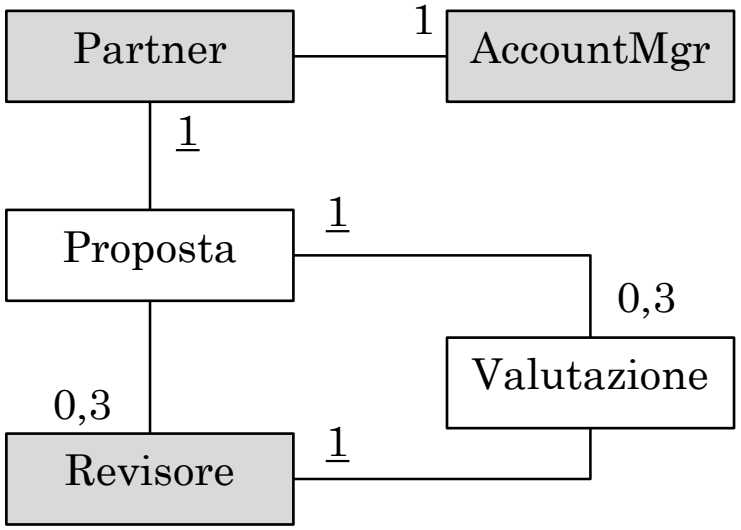
Esempio: gestione Proposte

Un'organizzazione tratta le proposte inserite dai partner. Una proposta è esaminata dall'account manager relativo al partner: può accettarla, respingerla o sottoporla a 3 revisori che sceglie. I revisori forniscono una valutazione nel range 1..10.
In modo automatico, se la media è >= 6 la proposta è accettata, altrimenti è respinta.
L'esito della proposta (accettata o respinta) è scritto nello stato che il partner può poi leggere.

Nel sistema informativo sono registrati partner, revisori e accountMgr (con i legami con i partner).

Ruoli: Partner, AccountMgr, Revisore.
Entità: Proposta, Valutazione.

Modello informativo



Categorie delle relazioni:

appartenenza: Partner - Proposta, Revisore - Valutazione

causa-effetto: Proposta - Valutazione.

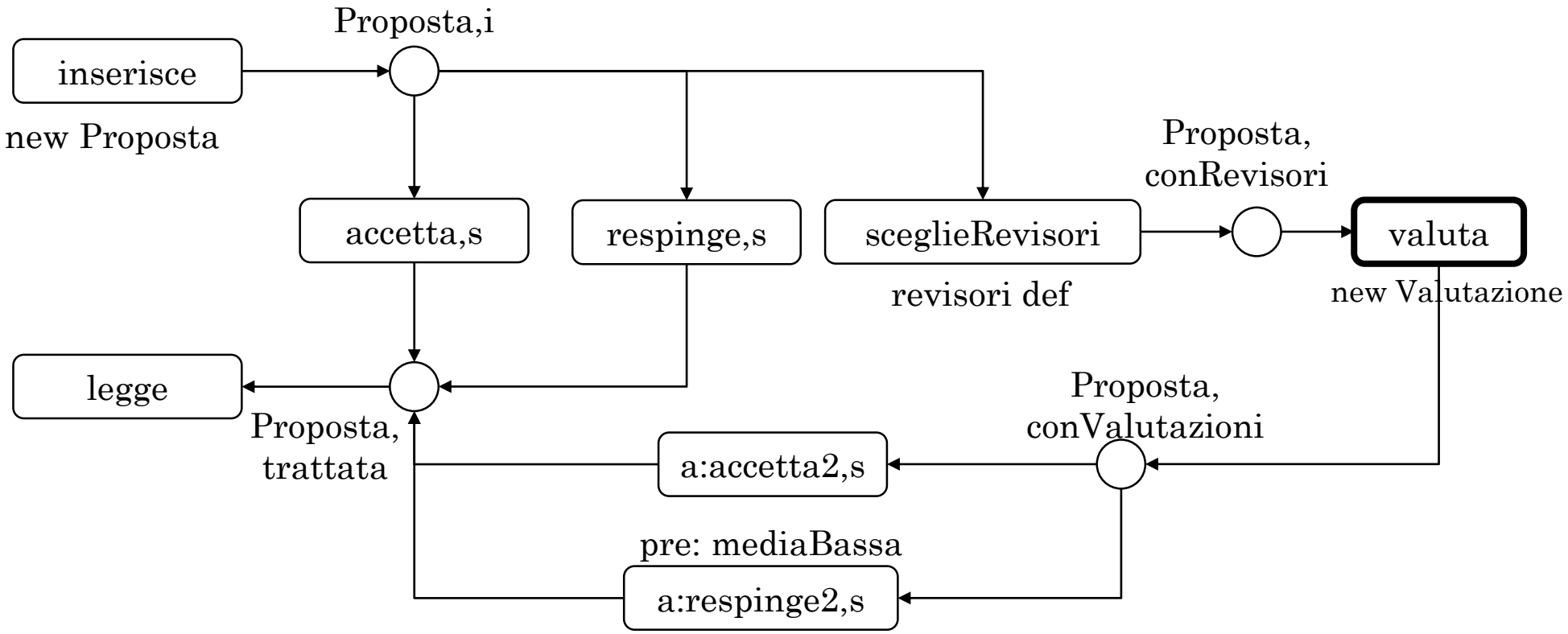
assegnazione: Proposta - Revisore.

Attributi
Proposta: stato (accettata, respinta); // inizialmente null
boolean mediaBassa = valutazioni.average(val) < 6. //attr. derivato; serve nel processo

Valutazione: int val (1..10).

Perché 0,3?

Processo



Scelta umana: *Proposta, i* ha 3 task in uscita dei quali è l'unico posto di input e non ci sono condizioni di routing o pre-condizioni.
Proposta, conValutazioni introduce invece una scelta automatica.
Ci sono *due scelte non strutturate*.

Modifica dello stato

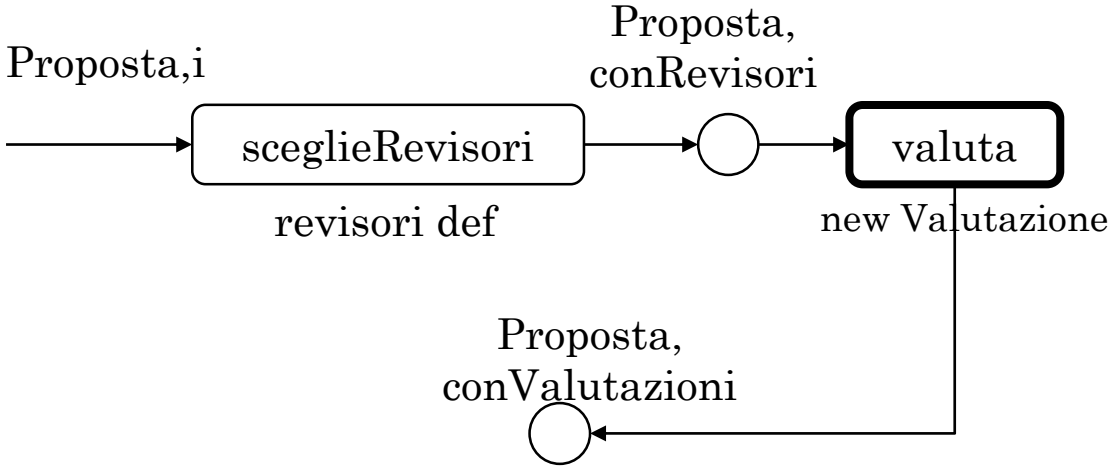
accetta, s equivale a accetta

stato == accettata

Proposta: stato (accettata, respinta); // inizialmente null

Se il nome di un task è seguito da **s** separata da virgola, il task modifica lo stato dell'entità trattata; il valore dello stato è quello corrispondente (a senso) al nome del task.

Analisi dei task

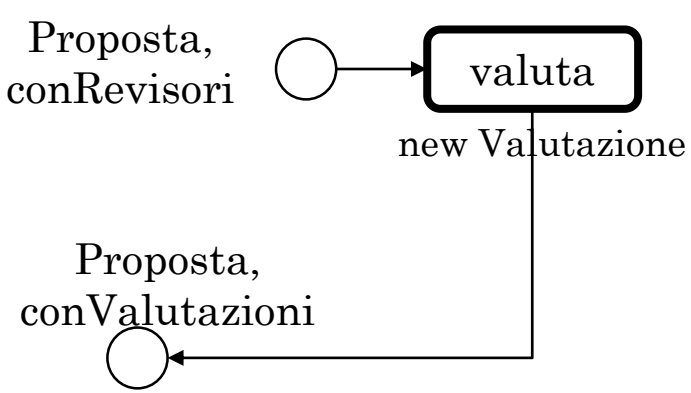


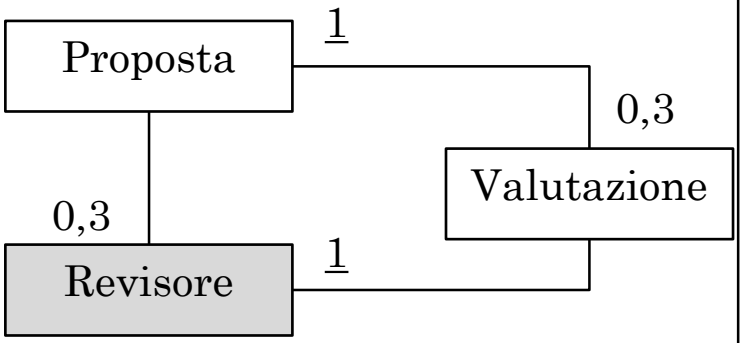
I task sono tutti passanti.

Il task *sceglieRevisori* associa la proposta a 3 revisori come indicato dalla molteplicità 3 nel modello informativo. La scelta dei revisori è fatta dal performer del task.

Il task **valuta** è multi-performer in quanto è assegnato a 3 revisori (quelli collegati alla proposta). continua …

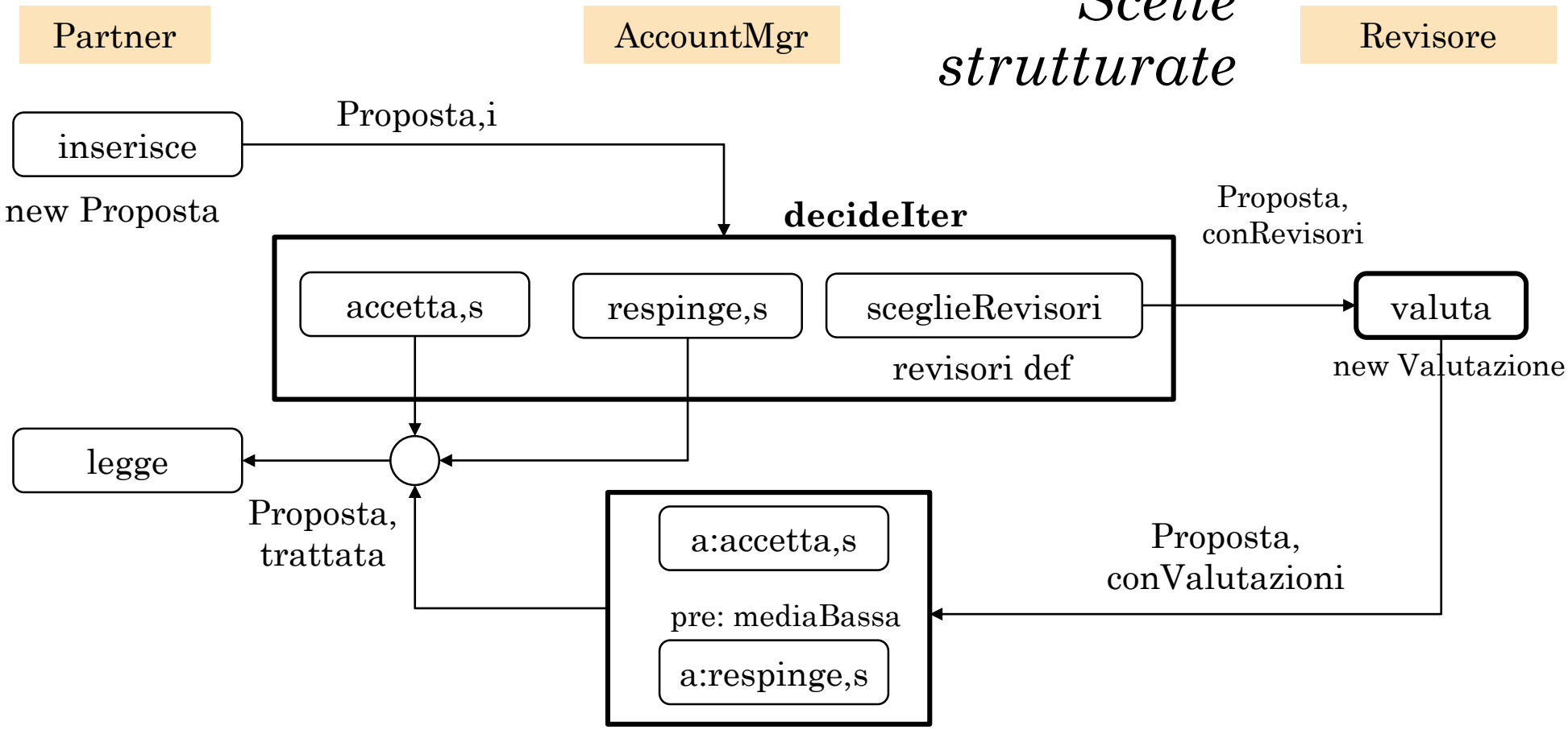
Task multi-performer





Il task **valuta** è multi-performer in quanto è assegnato a 3 revisori (quelli collegati alla proposta).
Ogni revisore riceve la stessa proposta e produce la propria valutazione.
Il fatto che l'output sia una proposta (e non una valutazione) indica che la produzione di una singola valutazione non è un evento per il processo mentre lo è la disponibilità di tutte.
La molteplicità può essere indicata qualitativamente con un bordo più spesso (opzionale).
IMPORTANTE: la nuova valutazione è collegata alla proposta e al revisore come indicato dalle relazioni obbligatorie di causa/effetto e appartenenza nel modello informativo.
Un task multi-performer produce nuove entità relative a quella di input.

Scelte strutturate



Una scelta umana e una automatica.

Tipi di task e scelte

task di timeout

scelta semplice

scelta composta

scelta composta con restrizione

task composto

task composto con restrizione

riduttore

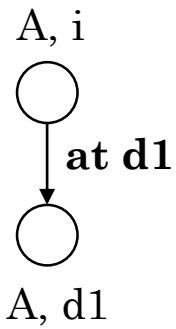
fork

join

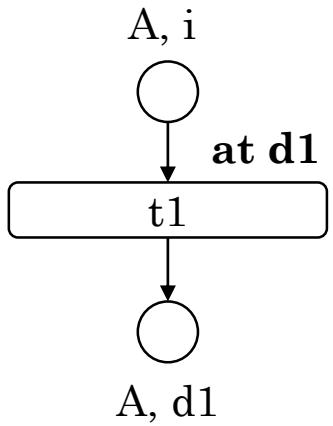
join/fork

task partecipativo

Task di timeout



task di timeout embedded

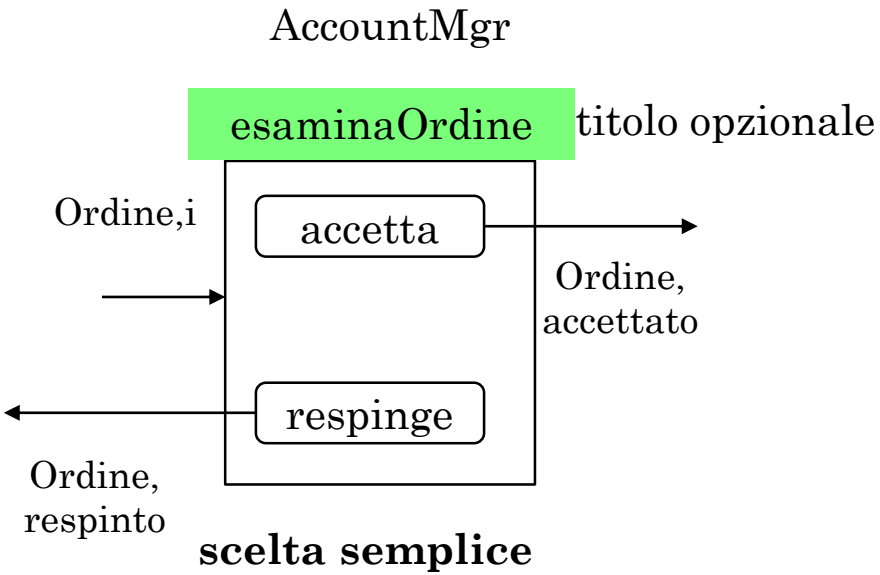


task di timeout esplicito

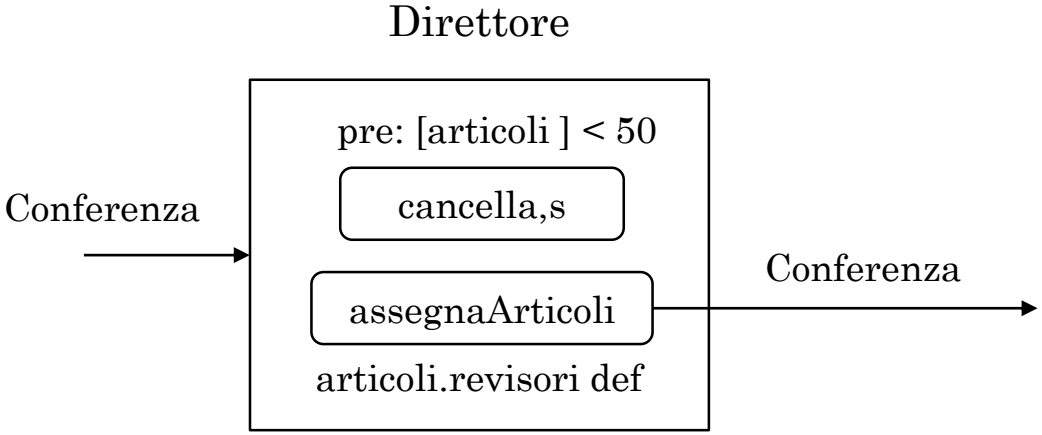
A: Date d1.

Alla scadenza dell'attesa (attributo d1 delle entità A), l'entità A è spostata dal task di timeout al posto successivo.

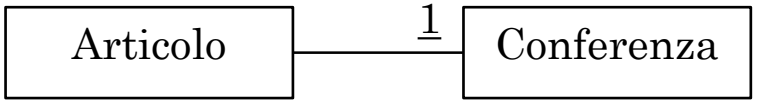
Scelta semplice



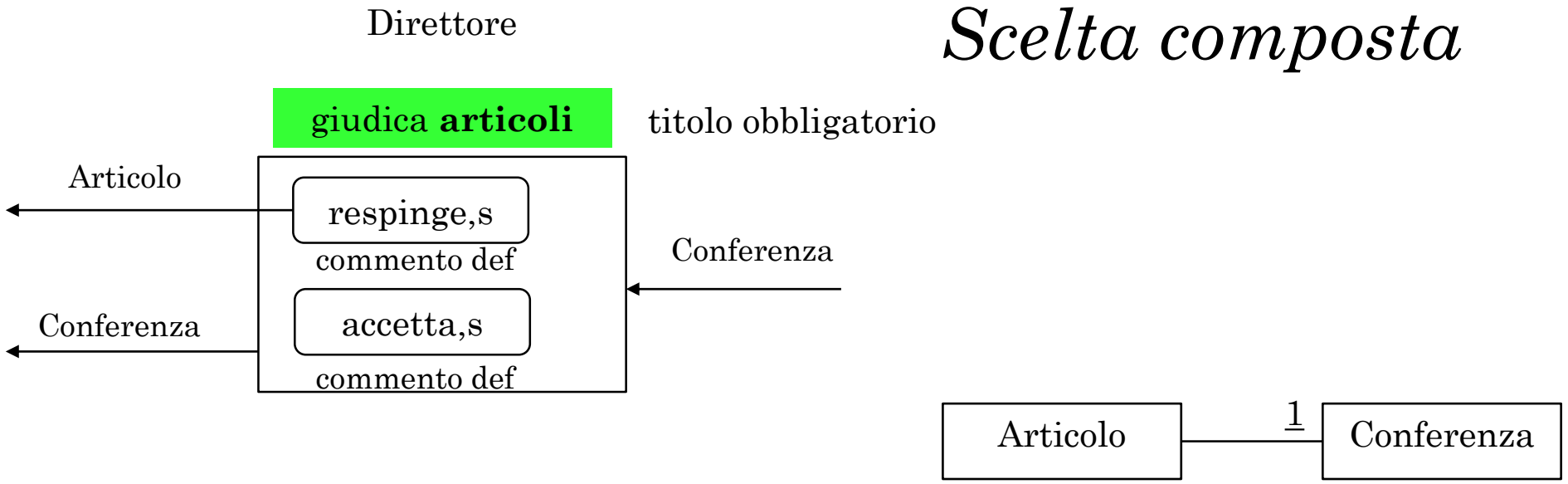
scelta semplice



scelta semplice automatica



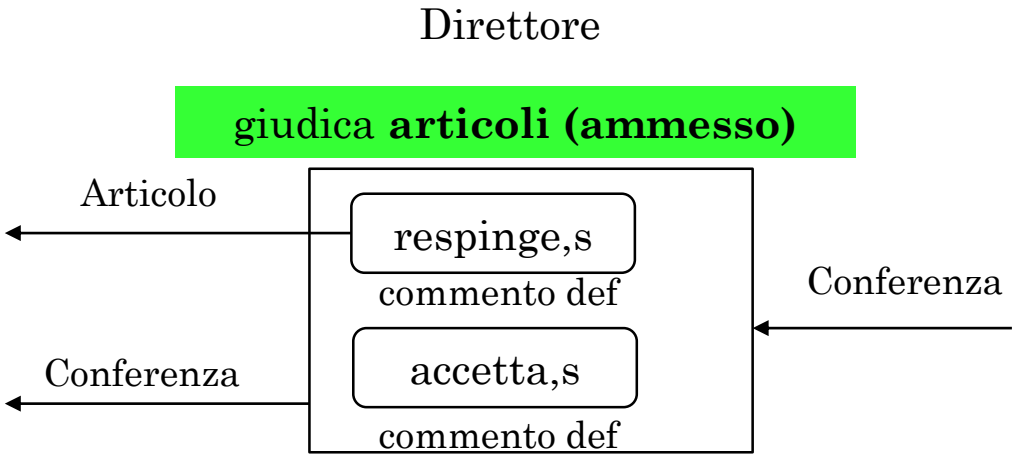
Scelta composta

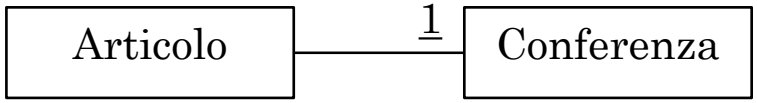


La scelta è composta perché non agisce sull'entità di input (in questo caso una conferenza) ma su entità (in questo caso gli articoli) ad essa collegate (entità *subordinate*). Il nome della scelta include il nome delle entità da trattare.
Gli input dei task interni sono le entità subordinate. I task interni possono avere output.
Una scelta composta ha un solo input e può avere un output.

Attributi: Articolo: String titolo, String commento.

Scelta composta con restrizioni



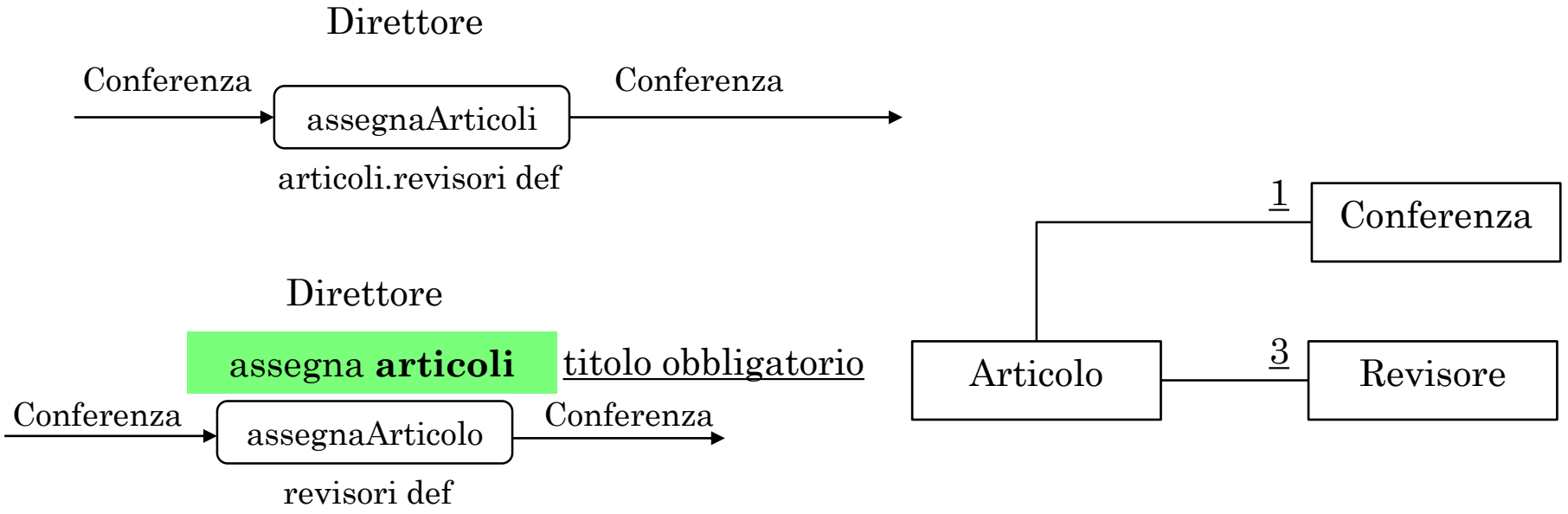


Articolo: stato (ammesso, accettato, respinto);
boolean ammesso = stato == ammesso.

Nell'esempio la scelta riguarda soltanto gli articoli ammessi.
La restrizione è definita mediante un'espressione booleana tra parentesi.

task semplice

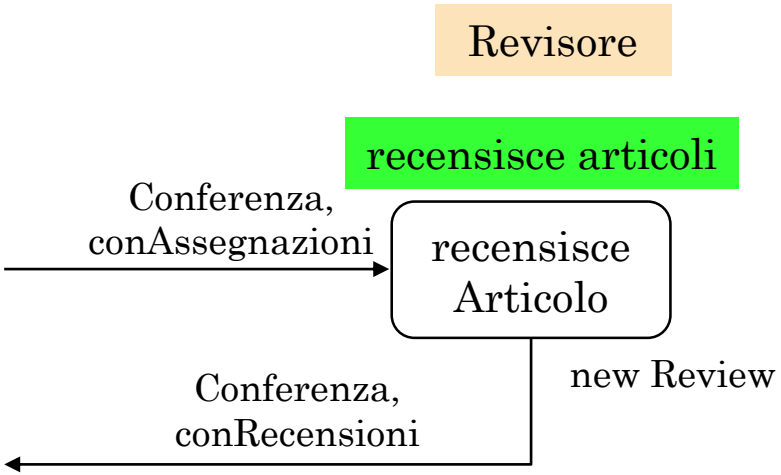
Task composto associativo

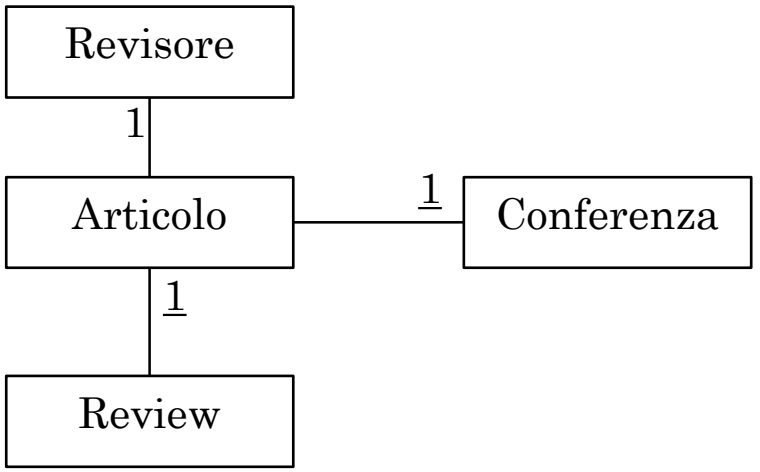


task composto associativo

Task composto: per ogni articolo della conferenza è eseguito il task semplice assegnaArticolo. Il nome del task composto (*assegna articoli*) è scritto sopra il task semplice: esso include l'attributo associativo *articoli* che indica quali elementi della conferenza vanno trattati.

Task composto generativo

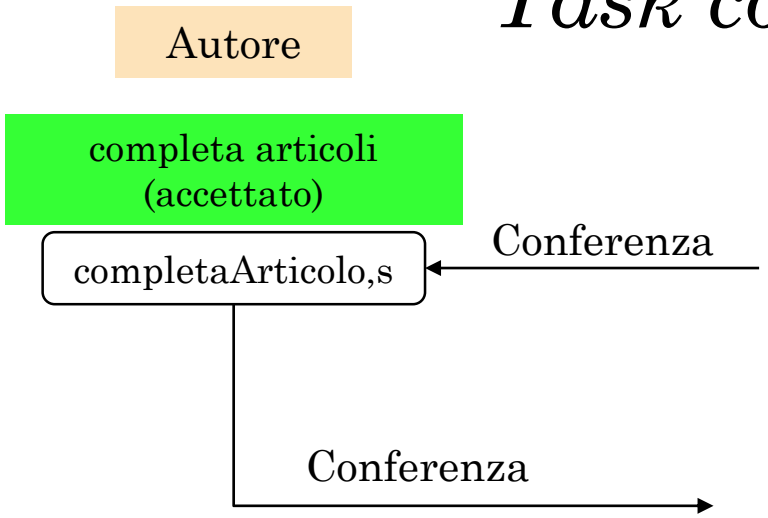




Si suppone che ad ogni articolo sia associato un revisore.
Il task *recensisce articoli* è **composto**; per ogni articolo della conferenza (che è l'input del task composto) è eseguito il task *recensisceArticolo* il quale genera una review per l'articolo di input.
Un task composto include nel nome del task il nome delle entità da trattare.

Le review si possono raggiungere attraverso gli articoli della conferenza.

Task composto con restrizione



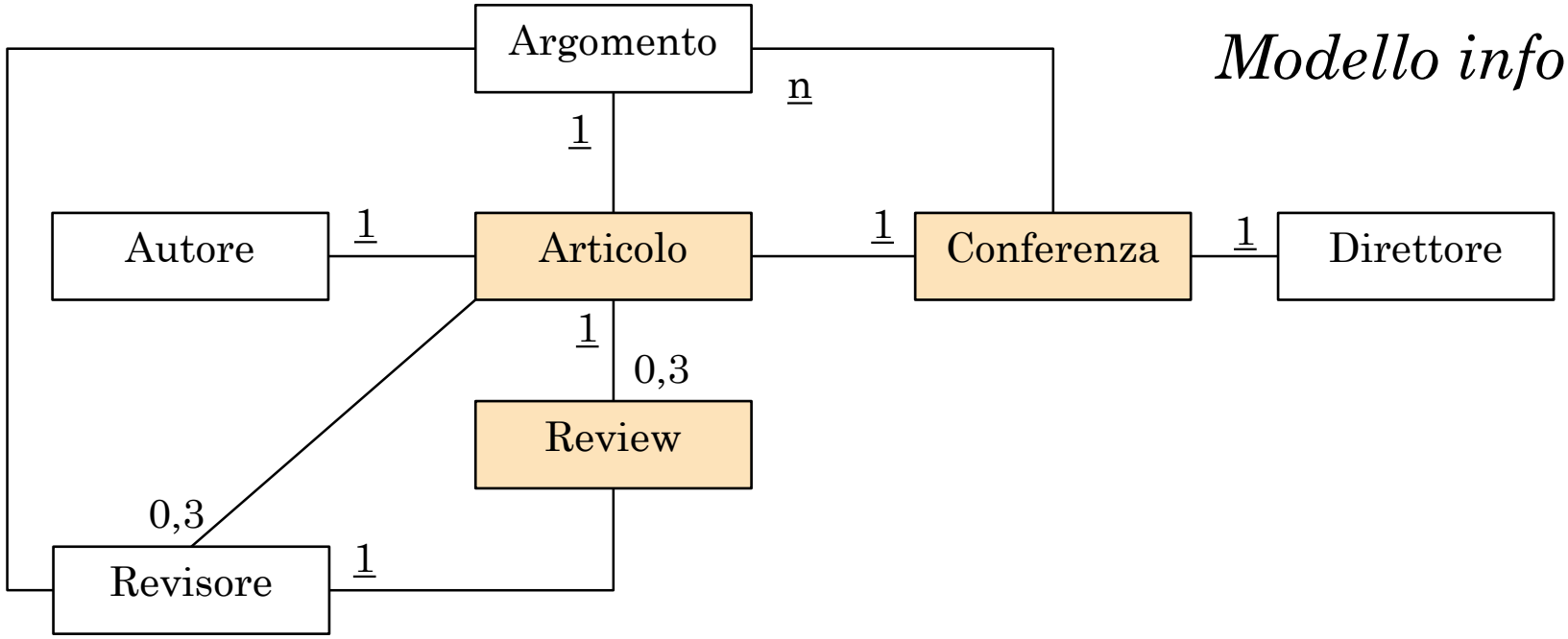
Il task *completa articoli* non riguarda tutti gli articoli della conferenza ma soltanto quelli accettati.

La restrizione è definita mediante un'espressione booleana tra parentesi.

Esempio: gestione Conferenze

Il processo opera in un'organizzazione di gestione di conferenze. Nel sistema informativo sono registrati direttori, autori, revisori e argomenti. I revisori sono collegati agli argomenti di loro interesse.
I direttori possono introdurre nuove conferenze: una nuova conferenza è associata ad alcuni argomenti (scelti dai direttori). Inoltre una conferenza contiene la deadline d1.
Prima della scadenza d1 gli autori possono sottoporre articoli (in versione preliminare) alla conferenza; gli articoli hanno un solo autore e si riferiscono ad un solo argomento della conferenza (1).
Al tempo d1, se non ci sono almeno 50 articoli, il direttore cancella la conferenza, altrimenti assegna gli articoli a 3 revisori (per articolo). I revisori sono interessati all'argomento dell'articolo (1). I revisori forniscono le review degli articoli.
Quando tutte le review sono disponibili, il direttore per ogni articolo decide se accettarlo o respingerlo e inoltre scrive un commento; ne deve accettare almeno 20.
Per gli articoli respinti gli autori leggono i commenti e per quelli accettati forniscono la versione completa. Dopo che tutte le versioni complete sono state fornite, il direttore conferma la conferenza.
La conferenza ha uno stato che indica se è cancellata o confermata.
Gli articoli hanno uno stato che indica se l'articolo è accettato, respinto, completo.
(1). Si definisca un invariante.

Modello informativo



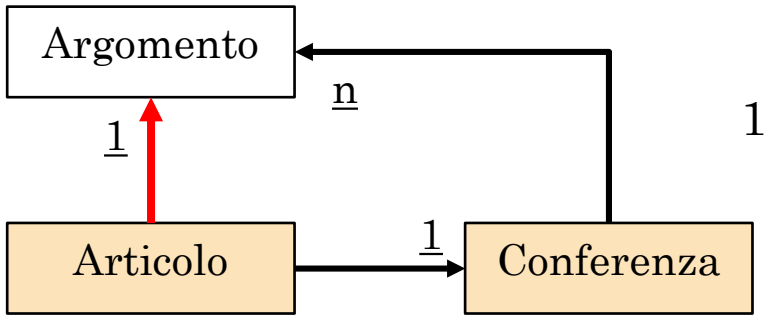
Attributi

Conferenza: Date d1; stato (cancellata, confermata).
Articolo: String commento, stato (accettato, respinto, completo);
boolean respinto = stato == respinto; boolean accettato = stato == accettato;
boolean completo = stato == completo.

Invarianti

L'argomento di un articolo fa parte di quelli della conferenza:
articolo.argomento in articolo.conferenza.argomenti.
I revisori di un articolo sono interessati all'argomento dell'articolo:
articolo.revisori in articolo.argomento.revisori

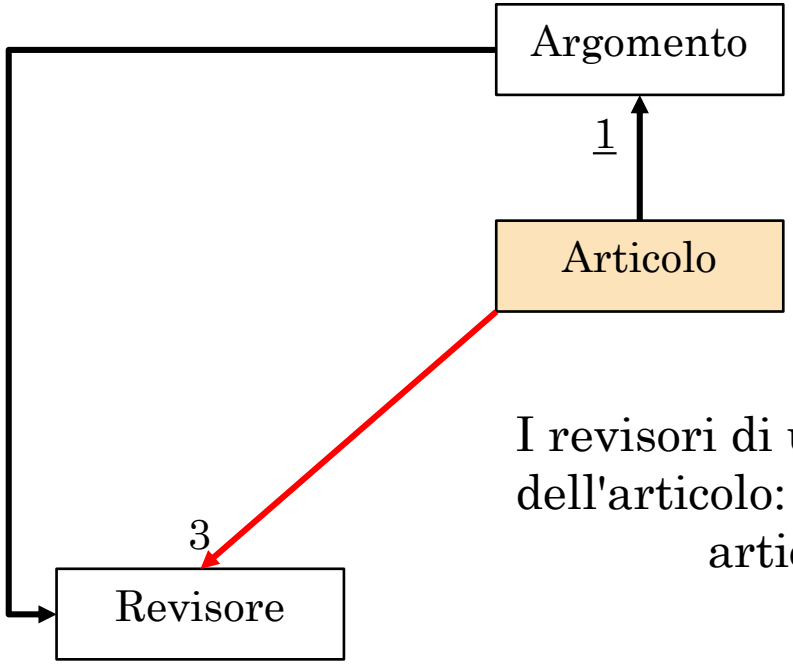
Invarianti



1 in n

L'argomento di un articolo fa parte di quelli della conferenza:

articolo.argomento in articolo.conferenza.argomenti.



m in n

I revisori di un articolo sono interessati all'argomento dell'articolo:

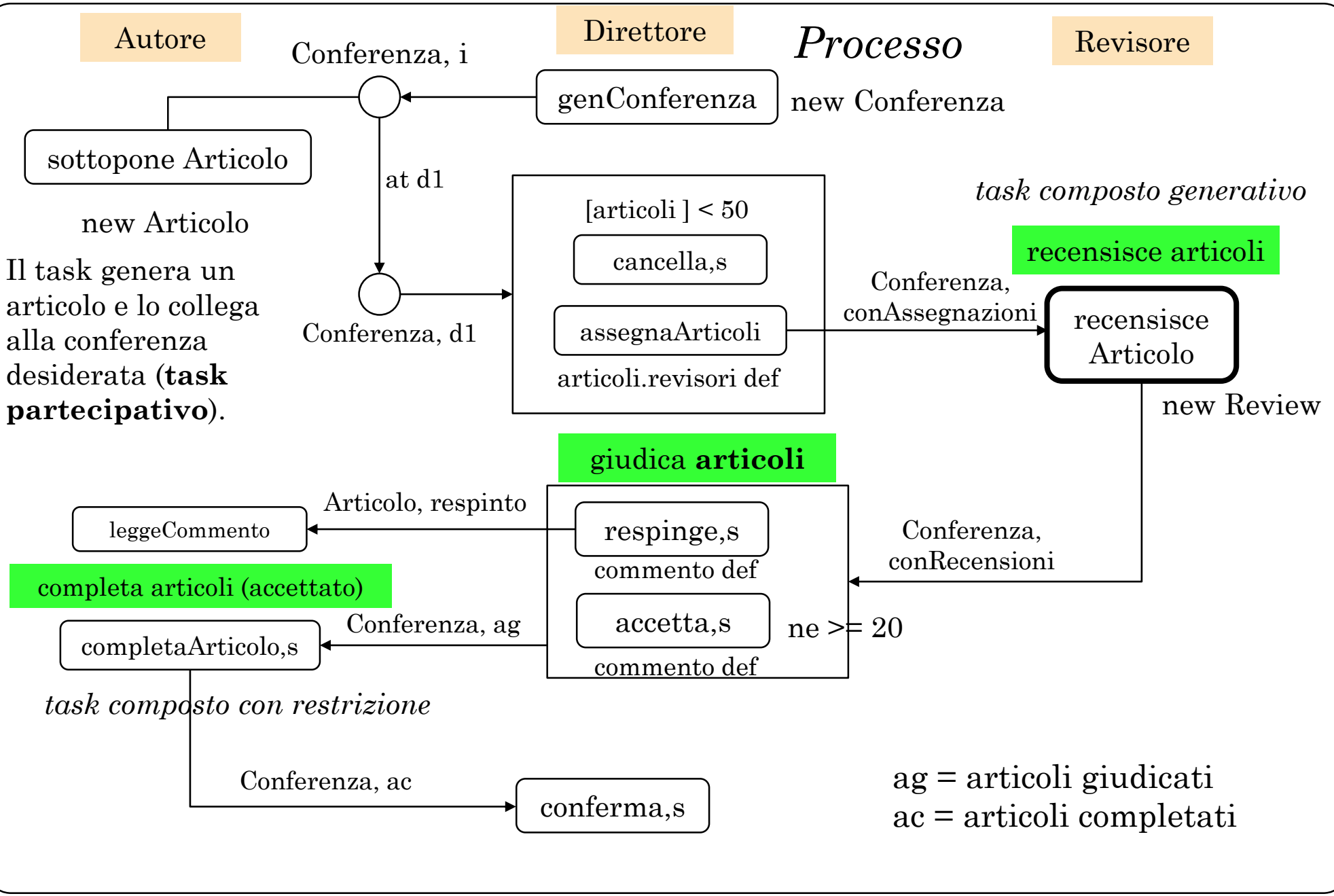
articolo.revisori in articolo.argomento.revisori

Note

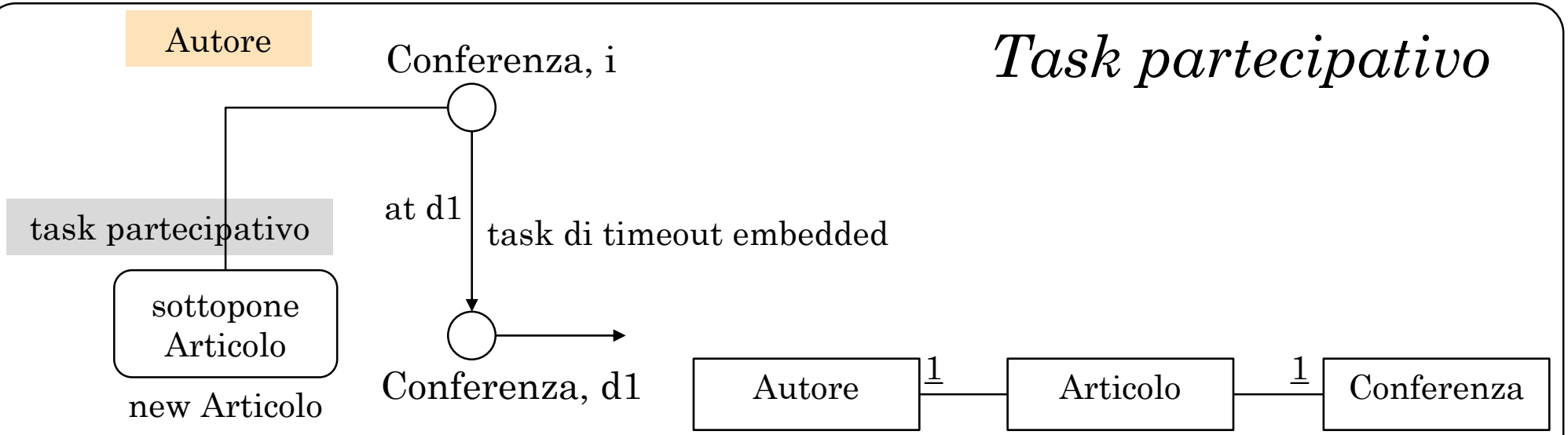
articolo.argomento in articolo.revisori.argomenti

è sbagliato!

i due percorsi sono di tipo 1.1 e 1.n.n (basta che uno solo dei 3 revisori sia competente sull'argomento)

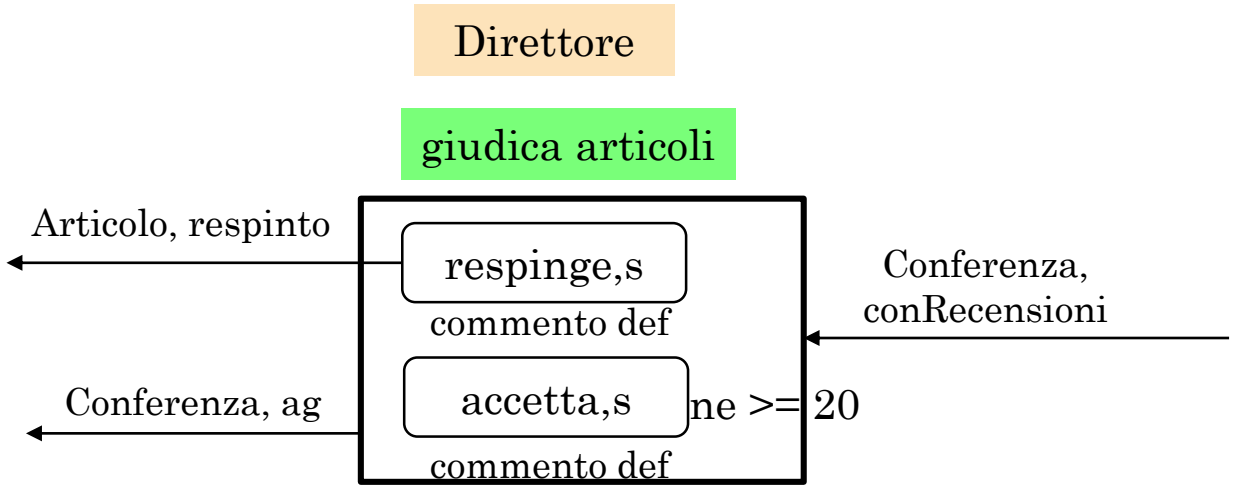

Autore
Direttore
Processo
Revisore
Conferenza, i
genConferenza
new Conferenza
sottopone Articolo
new Articolo
at d1
Conferenza, d1
[articoli] < 50
cancella,s
assegnaArticoli
articoli.revisori def
task composto generativo
recensisce articoli
Conferenza, conAssegnazioni
recensisce Articolo
new Review
Il task genera un articolo e lo collega alla conferenza desiderata (task partecipativo).
giudica articoli
Articolo, respinto
leggeCommento
respinge,s
commento def
Conferenza, conRecensioni
completa articoli (accettato)
Conferenza, ag
completaArticolo,s
accetta,s
commento def
ne >= 20
task composto con restrizione
Conferenza, ac
conferma,s
ag = articoli giudicati
ac = articoli completati

Task partecipativo



Gli autori possono sottoporre articoli ad una conferenza fino alla scadenza d1 (attributo della conferenza). Un **task partecipativo** associa una nuova entità ad un'entità esistente in base alla relazione partecipativa tra il tipo della nuova entità e quello dell'entità esistente (Articolo – Conferenza).
Il collegamento è senza freccia per indicare che il task non toglie la conferenza dal posto; quindi vari autori possono sottoporre contemporaneamente i loro articoli.
Alla scadenza d1 il task di timeout sposta la conferenza nello stato d1 e quindi non è più possibile sottoporre altri articoli alla conferenza.

Vincoli sul numero di esecuzioni



l direttore deve accettare almeno 20 articoli. Questo vincolo si può esprimere con il numero di esecuzioni del task *accetta*: **ne** >= 20, dove **ne** significa *numero di esecuzioni*.

Effetti dei task e dataflow

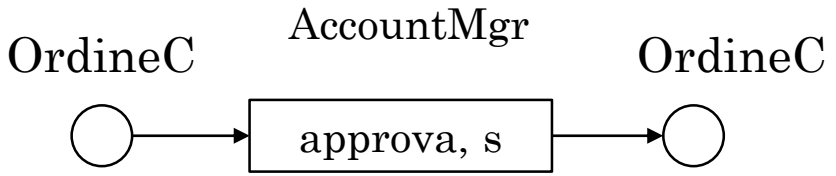
I task, mediante le *post-condizioni*, producono vari effetti sul sistema informativo. I principali effetti sono: modificativo, associativo, generativo, aggregativo, sincronizzativo.

Il *dataflow* è costituito dalle entità che fluiscono nel processo; è rappresentato dai posti ed è determinato dai task sulla base dei posti di output.

Dal punto di vista del dataflow i task si possono suddividere in vari tipi: task passanti, aggregativi, riduttori, fork, join, join/fork, mapping task.

Le prossime slide presentano degli esempi.

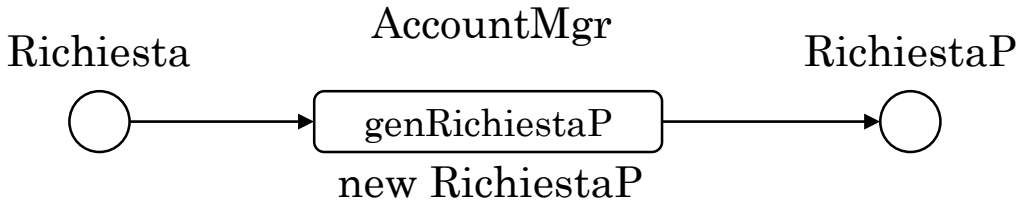
Task con 1 posto di input e 1 posto di output



OrdineC = OrdineCliente

Task passante: l'entità di output è quella di input.

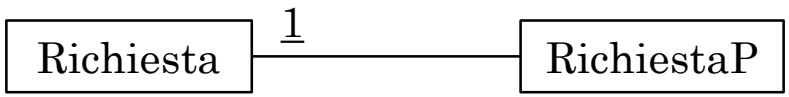
L'effetto del task è modificativo.



Mapping task: l'entità di output è collegata a quella di input.

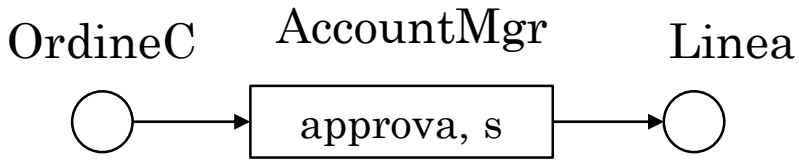
L'effetto del task è generativo.

Modello informativo



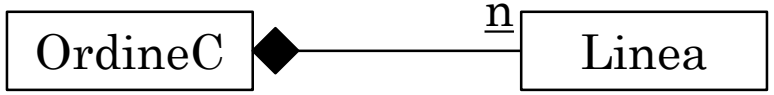
RichiestaP = RichiestaPreventivo

Task con 1 posto di input e 1 posto di output

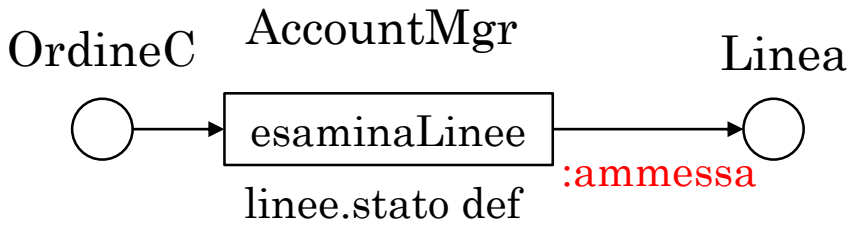


L'effetto del task è modificativo.

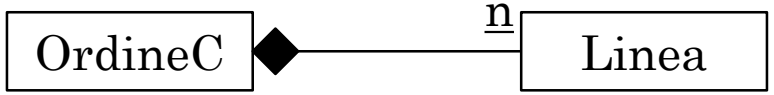
Modello informativo



Mapping task: nel posto di output entrano le linee collegate all'entità di input.



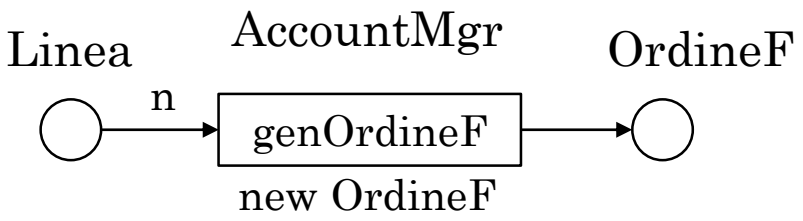
Modello informativo



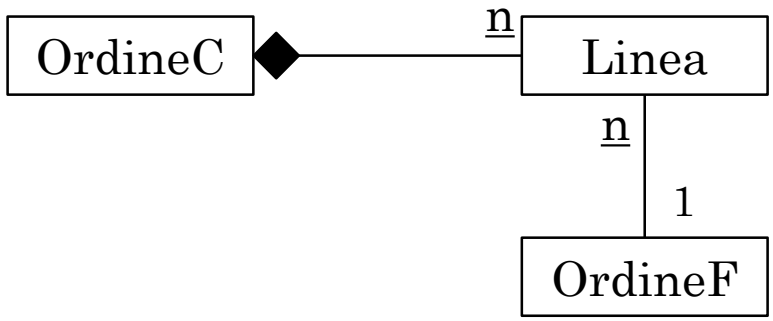
Linea: stato (ammessa, respinta);
boolean ammessa = stato == ammessa.

Mapping task con restrizione: nel posto di output entrano soltanto le linee ammesse collegate all'entità di input. La restrizione è effettuata mediante l'attributo derivato *ammessa* preceduto da **:**.

Task con 1 posto di input e 1 posto di output



Modello informativo



L'effetto del task è aggregativo.

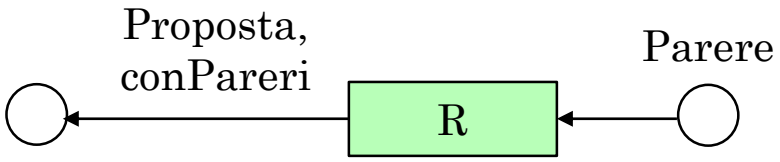
OrdineF = OrdineFornitore

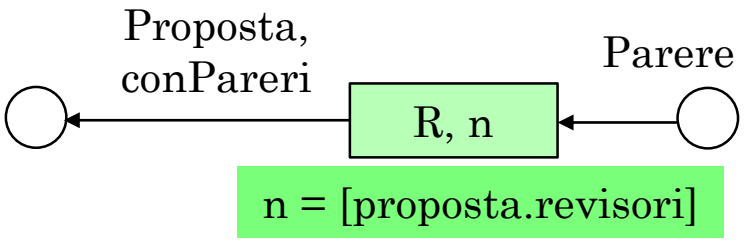
Una linea è associata a 1 OrdineF nell'ipotesi che tutte le linee entrino in ordiniF.

Task aggregativo. Nel posto di uscita entra una nuova entità collegata alle entità di input.

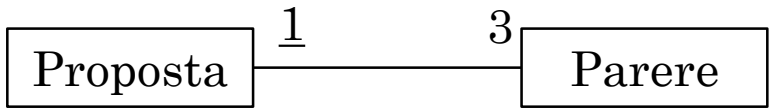
Task riduttore

Ha un posto di input (ad es. di tipo A) e uno di output (ad es. di tipo B). Quando tutte le entità A relative alla stessa entità B sono presenti nel posto di input, il riduttore (task standard) le toglie e immette nel posto di output l'entità B. Se il numero di entità A non è noto a priori, è necessaria una formula. Il nome del task inizia con R.





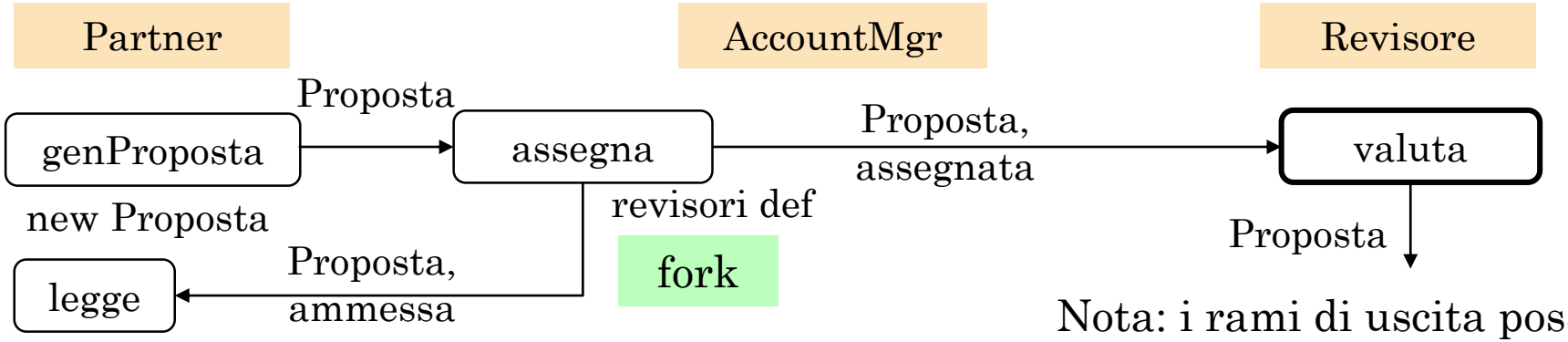
Modello informativo



Formula per indicare il n. di pareri attesi (tanti quanti sono i revisori della proposta).

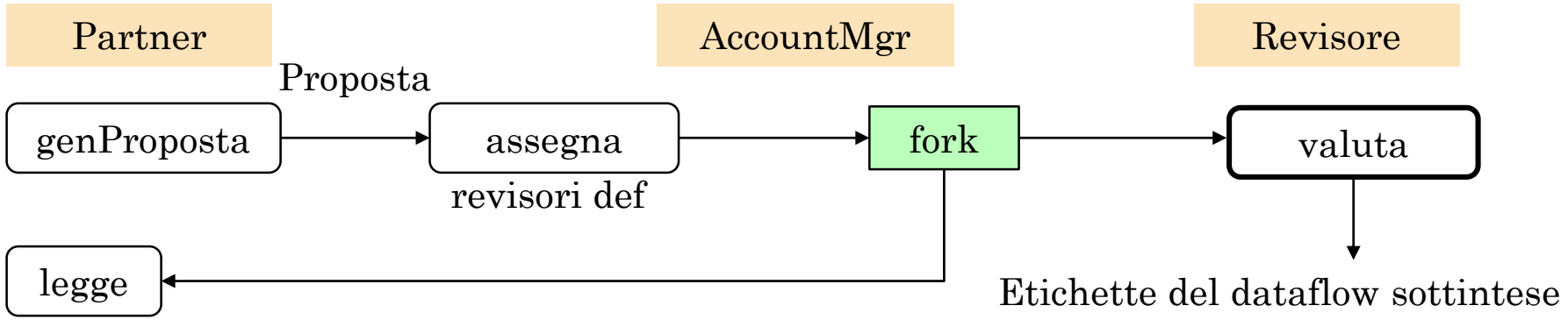
Fork: 1 posto di input e 2 o più posti di output

Esempio: La proposta generata da un partner dopo essere stata associata a 3 revisori è inviata ad essi e al partner.



Nota: i rami di uscita possono avere un effetto passante o di mapping.

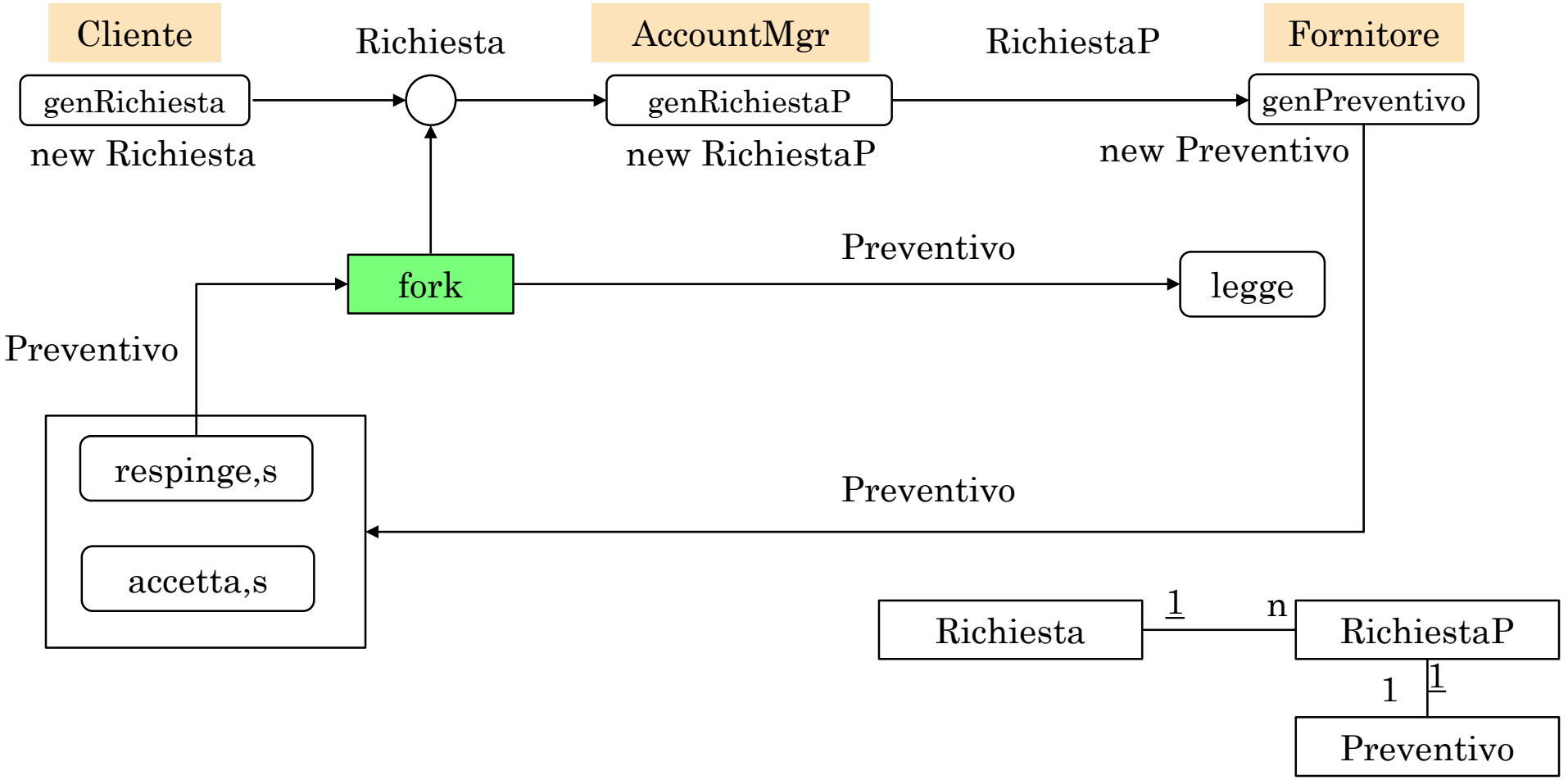
Si può usare anche un task fork automatico



Etichette del dataflow sottintese

Fork

Nell'esempio, il ramo *fork - Richiesta* esegue un *mapping*: nel posto di output entra la richiesta associata al preventivo. Il ramo *fork - Preventivo* è *passante*.



Join

Un task join ha due o più posti di input e un solo posto di output.

Effetti di un join:

aggregativo: il join aggrega entità di input in una nuova entità;

associativo: il join associa entità di input;

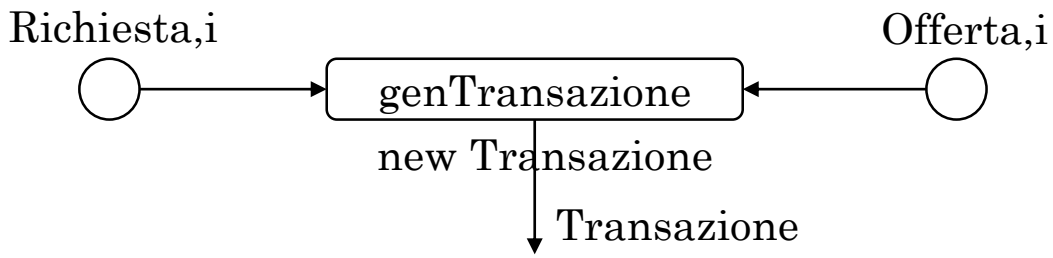
sincronizzativo: il join tratta entità di input correlate (aspetta che siano presenti nei posti di input).

Se ha due o più posti di output è un task join/fork.

I rami di uscita possono avere un effetto passante o di mapping.

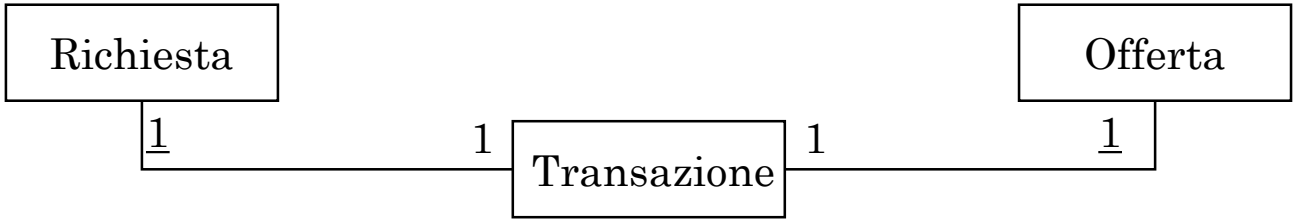
Join

Esempio: La richiesta proveniente da un cliente è combinata con l'offerta proveniente da un fornitore per generare una transazione.



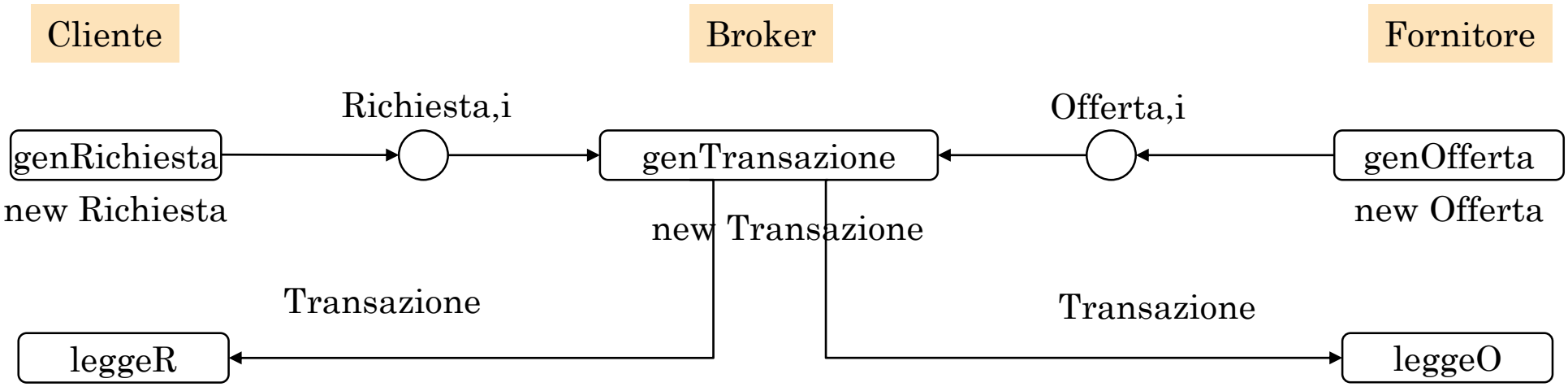
Effetto aggregativo

Nel posto di uscita entra la nuova entità di tipo Transazione (effetto mapping)



Con pesi sugli archi la transazione può combinare varie richieste con 1 offerta o viceversa.

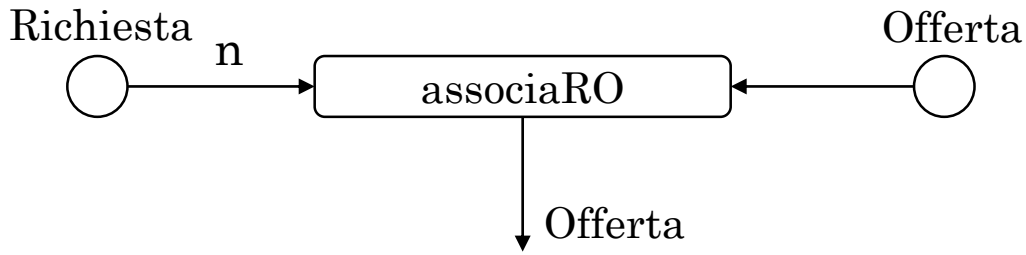
Join / Fork



La transazione deve essere inviata sia al cliente sia al fornitore.

Join

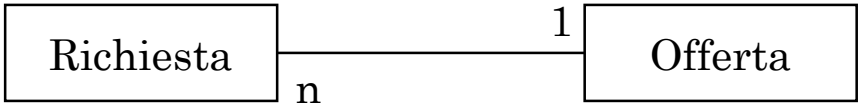
Il task associaRO associa varie richieste ad un'offerta. L'effetto del task è associativo.



associaRO: post: richieste.offerta == offerta.

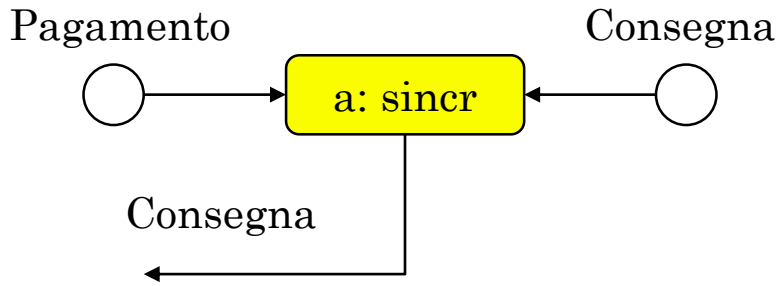
Effetto associativo

Il ramo di uscita è passante; nel posto di output entra l'offerta di input.



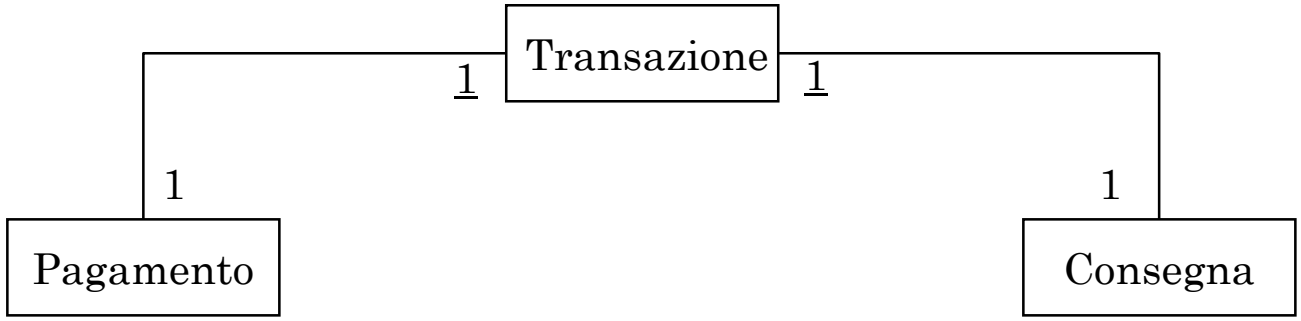
Join

Il task sincr sincronizza un pagamento e una consegna collegati alla stessa transazione ed emette la consegna. Il task ha un effetto sincronizzativo.



Il ramo di uscita è passante; nel posto di output entra la consegna di input.

sincr: pre: pagamento.transazione == consegna.transazione.



B2B Systems

Modelli:

Collaborazioni con relativi modelli informativi

Modello informativo del processo considerato

Modello del processo

B2B systems

A B2B (Business-to-Business) system is made up of a number of organizations that cooperate to achieve a common purpose. Organizations can be denoted by the role they play, e.g., buyer, seller, and distributor.

Organizations cooperate through their processes, which interact by sending/receiving messages according to predefined protocols represented by collaboration models.

Collaboration models establish what messages two parties must exchange so as to achieve the intended purpose. They are binary communication models and should be defined before the implementation of the actual business processes begins.

Assumptions

Messages are strongly related to business entities.

We assume that an output message has one specific entity as its primary source in the sender, and an input message mainly results in the generation or update of one specific entity in the recipient.

In fact, it is common opinion that *communication in business collaborations is about aligning the information systems of the partners*.

Although the parties have distinct information systems, they can share the definition of a number of entities called *global entities*. Messages can carry references to global entities.

For modeling purposes we assume that the mapping between business entities and messages can be performed automatically.

Collaboration models

A collaboration model (CM) defines the way two parties have to interact in order to achieve a common goal. It is a *type* in that it is a description of how a number of similar collaboration occurrences will take place.

CMs are based on UML sequence diagrams.

Esempi di collaborazioni tra un buyer e un seller

Il buyer manda al seller una richiesta di offerta relativa ad un tipo di prodotto offerto dal seller. Il seller risponde con un'offerta. Il buyer può accettare o rifiutare l'offerta (comunicando la decisione al seller). I tipi di prodotti sono noti al buyer.

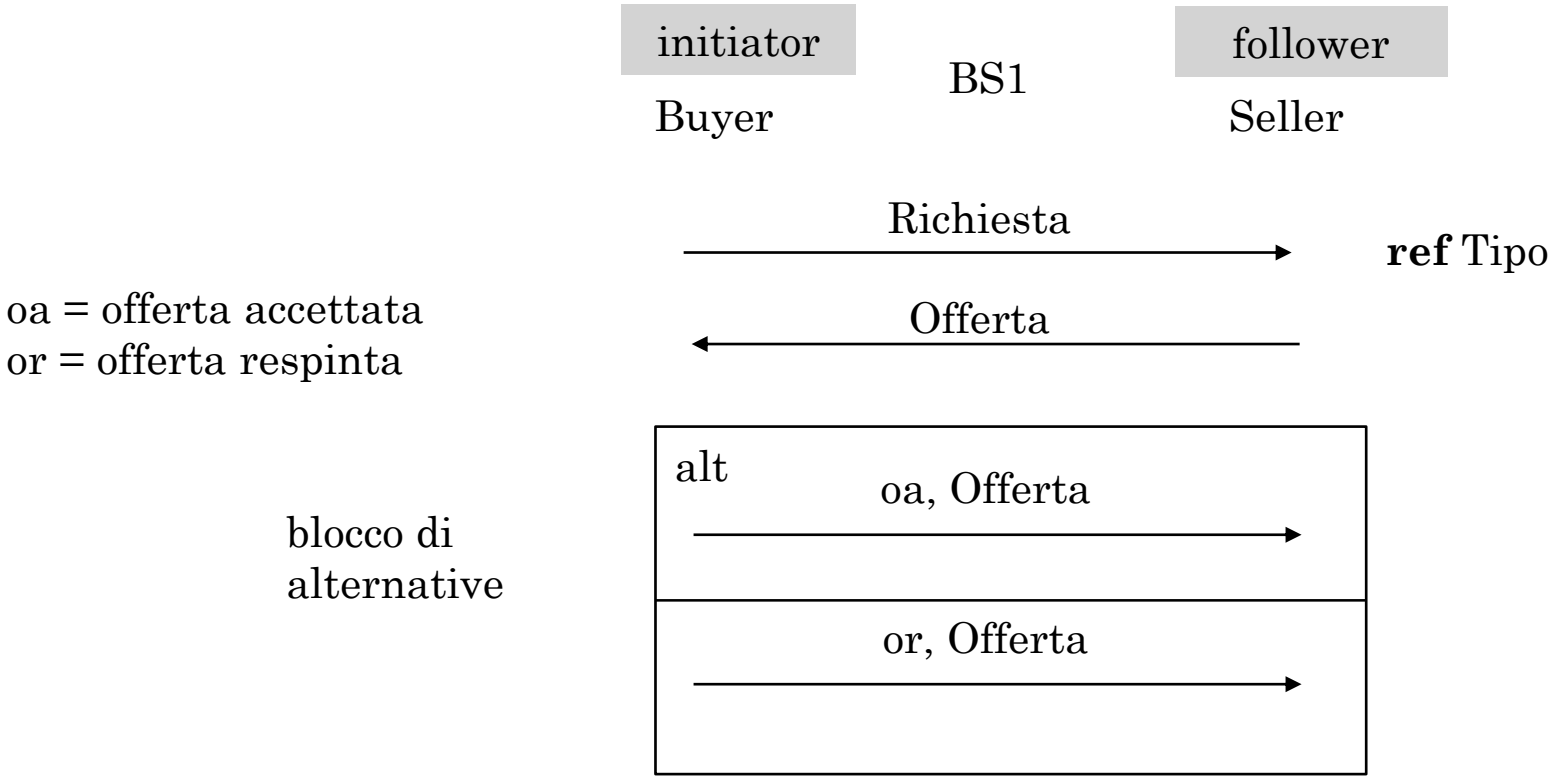
Variante

La richiesta include due deadline, d1 and d2. L'offerta va inviata entro d1 e l'accettazione entro d2, altrimenti la collaborazione si considera conclusa. Se l'accettazione non è inviata nei termini stabiliti l'offerta si considera rifiutata.

I due modelli di collaborazione si chiamano BS1 e BS2.

Collaborazione buyer-seller BS1

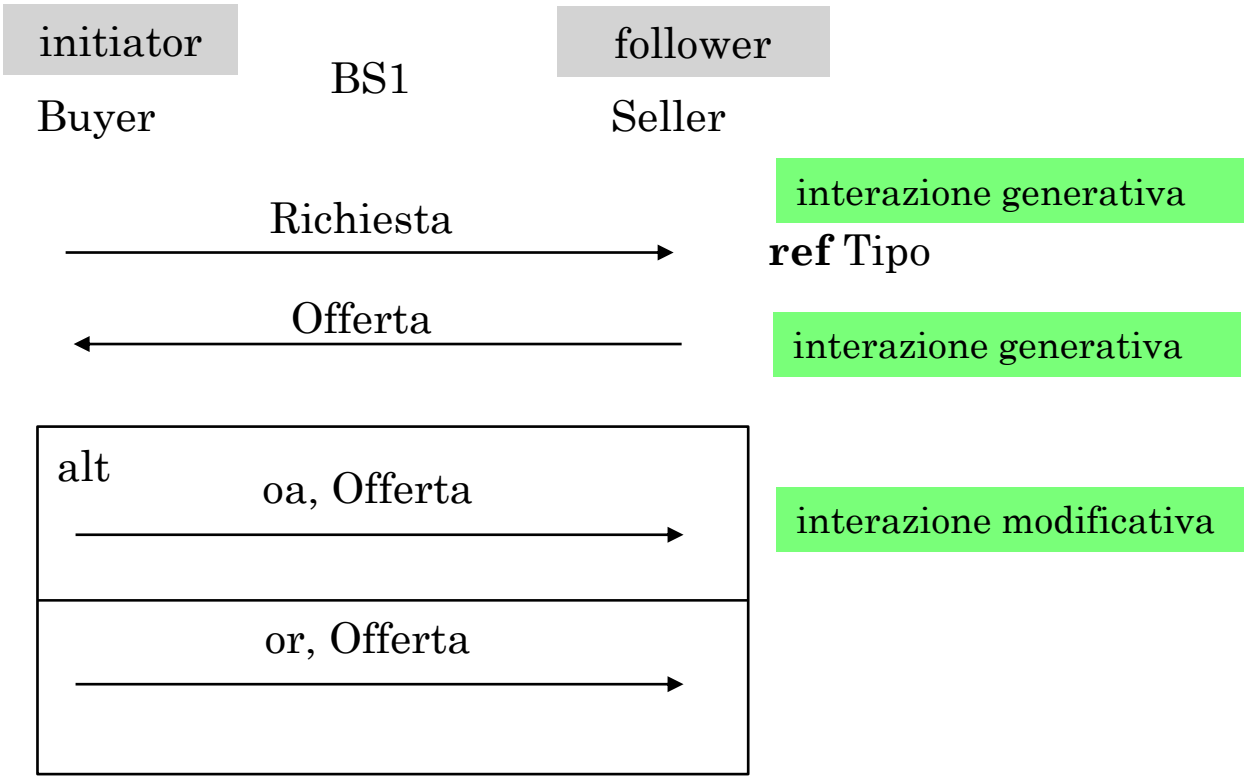
Il buyer manda al seller una richiesta di offerta relativa ad un tipo di prodotto offerto dal seller. Il seller risponde con un'offerta. Il buyer può accettare o rifiutare l'offerta.



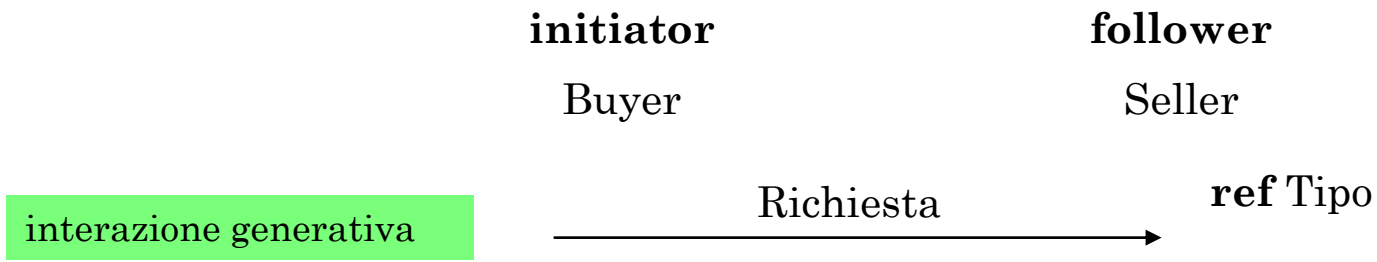
La keyword ref introduce entità note agli interlocutori; il payload del messaggio include il loro identificativo (id).

Collaborazione buyer-seller BS1

Il buyer manda al seller una richiesta di offerta relativa ad un tipo di prodotto offerto dal seller. Il seller risponde con un'offerta. Il buyer può accettare o rifiutare l'offerta.



Note



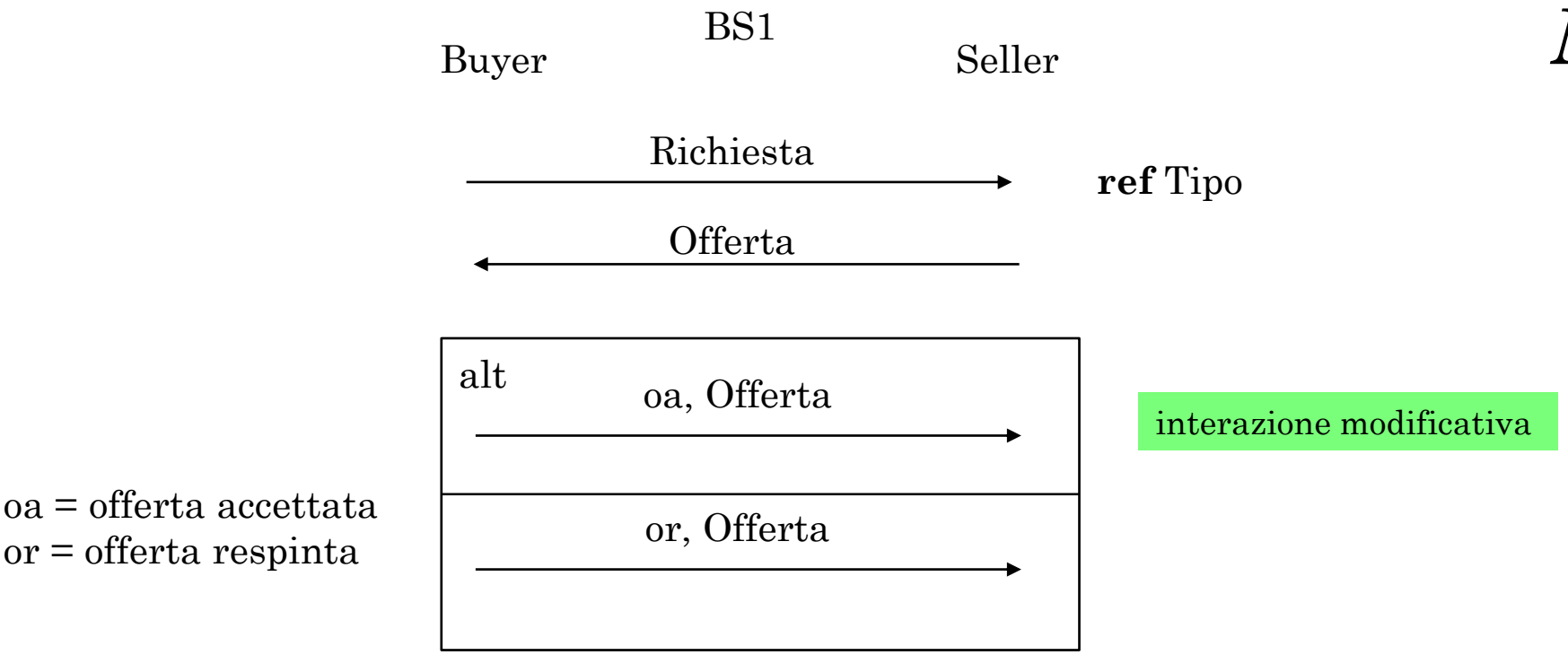
Le interazioni generative trasportano nuove entità (con gli attributi indicati nel modello di collaborazione) mediante messaggi.
All'inizio della modellazione gli attributi sono spesso sottintesi.
Un messaggio include un header e un **payload**. Il payload contiene gli attributi dell'entità.

Il **nome dell'interazione**, come Richiesta, è il nome del tipo dell'entità trasportata.

Il ricevente genera nel sistema informativo una nuova entità con le informazioni contenute nel payload dell'interazione.

Nota: l'interazione iniziale di una collaborazione è sempre generativa.

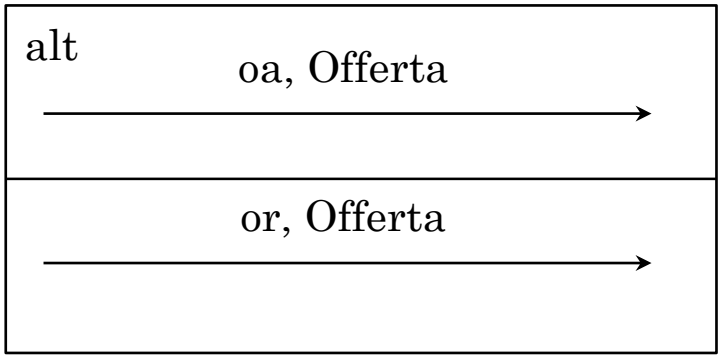
Note



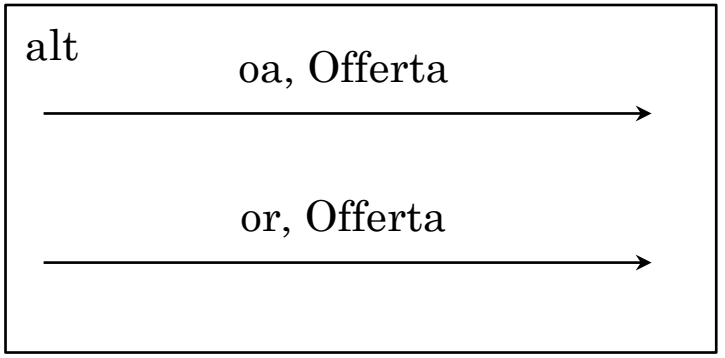
Un'interazione modificativa riguarda un'entità già trasmessa in precedenza e contiene delle modifiche agli attributi che il mittente vuole rendere note al destinatario.
Il primo nome è il nome dell'interazione (spesso una sigla) e ne indica il significato; il secondo nome è il nome del tipo dell'entità.

Gli attributi modificati sono spesso sottintesi.

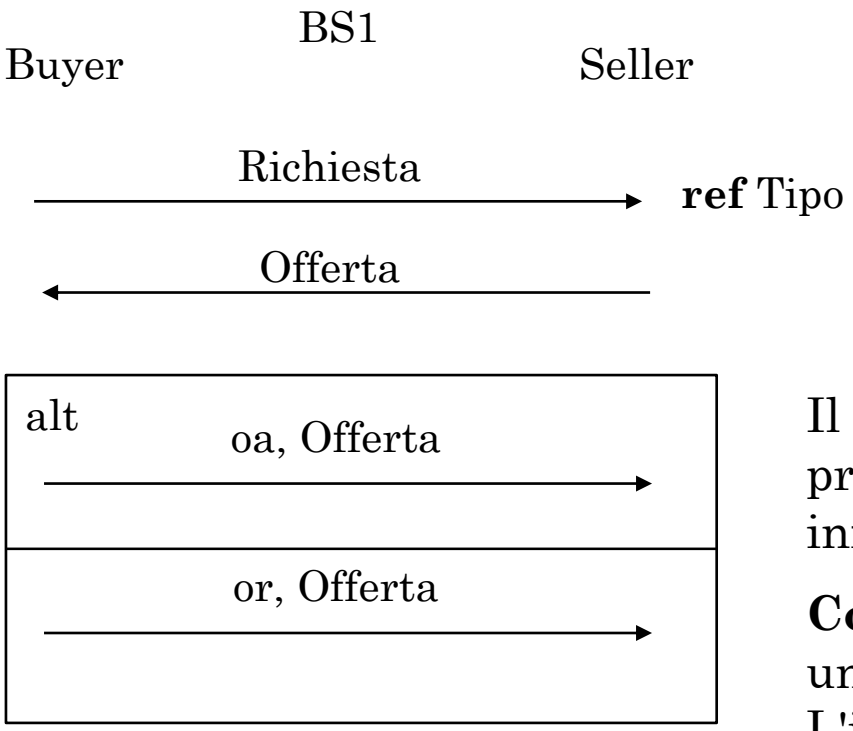
Forma compatta del blocco alt

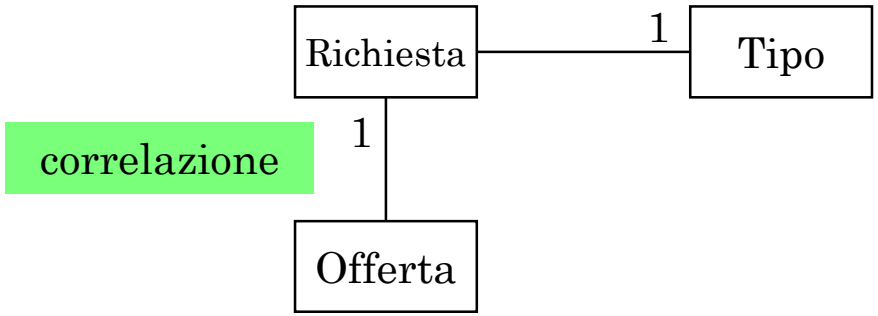


Se le alternative sono interazioni singole



Modello informativo della collaborazione buyer-seller BS1



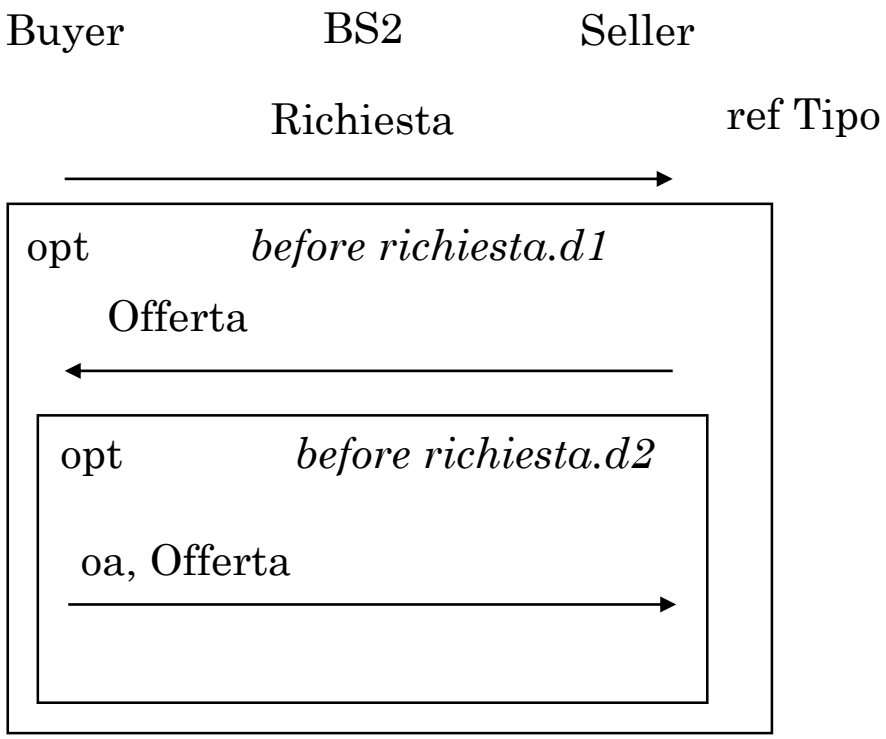


Il modello informativo di una collaborazione prefigura una porzione del modello informativo degli interlocutori.

Correlazione tra Richiesta e Offerta: un'offerta è l'effetto di una richiesta.
L'interazione ← Offerta
include l'id della richiesta a cui si riferisce l'offerta.

Variante: offerta e accettazione opzionali (BS2)

La richiesta include due deadline, d1 and d2. L'offerta va inviata entro d1 e l'accettazione entro d2, altrimenti la collaborazione si considera conclusa.

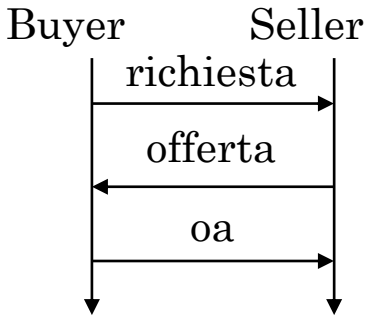


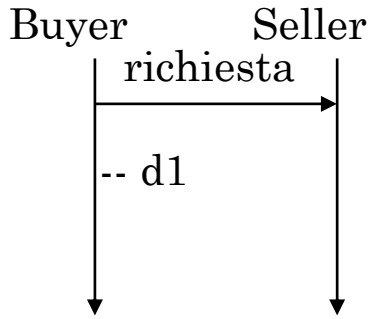
Un blocco opzionale può essere saltato. Il primo blocco è saltato se l'offerta non è ricevuta prima della scadenza indicata nell'attributo d1 della richiesta.

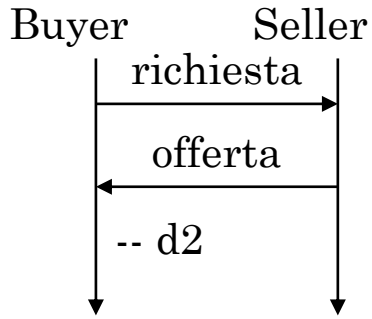
Attributi
Richiesta: Date d1; Date d2.

Nota: la collaborazione termina con oa, oppure a d1 (se non è inviata l'offerta), oppure a d2 (se non è eseguita oa). Gli attributi d1 e d2 sono visibili nell'istanza di collaborazione.

Scenari


Buyer
Seller
richiesta
offerta
oa


Buyer
Seller
richiesta
-- d1

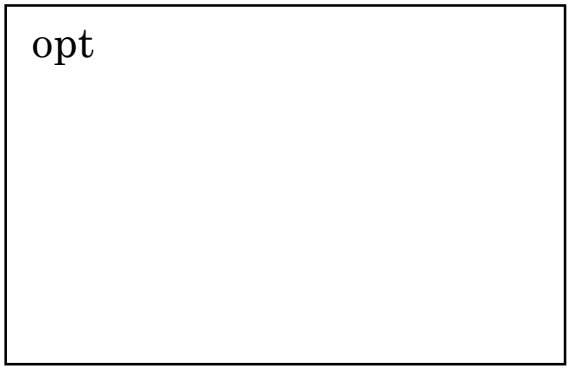

Buyer
Seller
richiesta
offerta
-- d2

UML sequence diagrams

The sequence of interactions is given by the vertical arrangement. Interactions are grouped in blocks (also called *fragments).*

The control flow is defined by means of *control constructs* (opt, alt, loop, break, continue, end). The default control construct is opt.

Sequence diagrams can be used for illustrative purposes (traces) or normative ones.



Optional block: the interactions included are optional. The option may be *time-driven, event-driven or data-driven*.

Alternatives block: it is made up of a number of alternatives (blocks). Only the interactions of one block take place.

Struttura delle collaborazioni

Una *sequenza collaborativa* è una sequenza di interazioni e/o blocchi (opt, alt, loop).
Un blocco opt contiene una sequenza collaborativa; analogamente un loop.
Un blocco alt è diviso in sezioni, ciascuna delle quali contiene una sequenza collaborativa.

Una *collaborazione* è una sequenza collaborativa il cui primo elemento è un'interazione generativa.

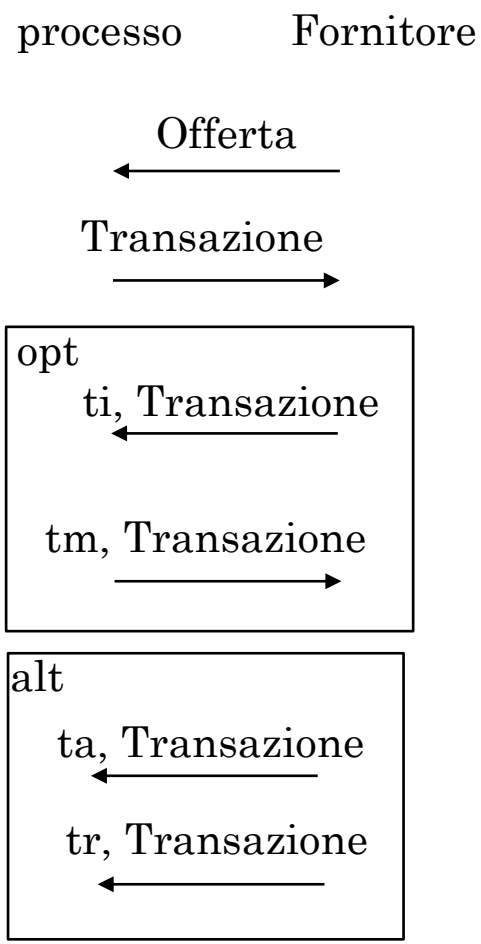
Le sezioni di un blocco alt non possono avere le interazioni iniziali con direzioni diverse, per non ingenerare confusione.

Blocchi opt e loop

Un blocco opt può essere event-driven, time-driven o data-driven.

Un loop ha una sezione che può essere eseguita varie volte, solitamente in modo sequenziale. Se le istanze della sezione sono eseguite in parallelo, il loop è seguito dalla **keyword par**.

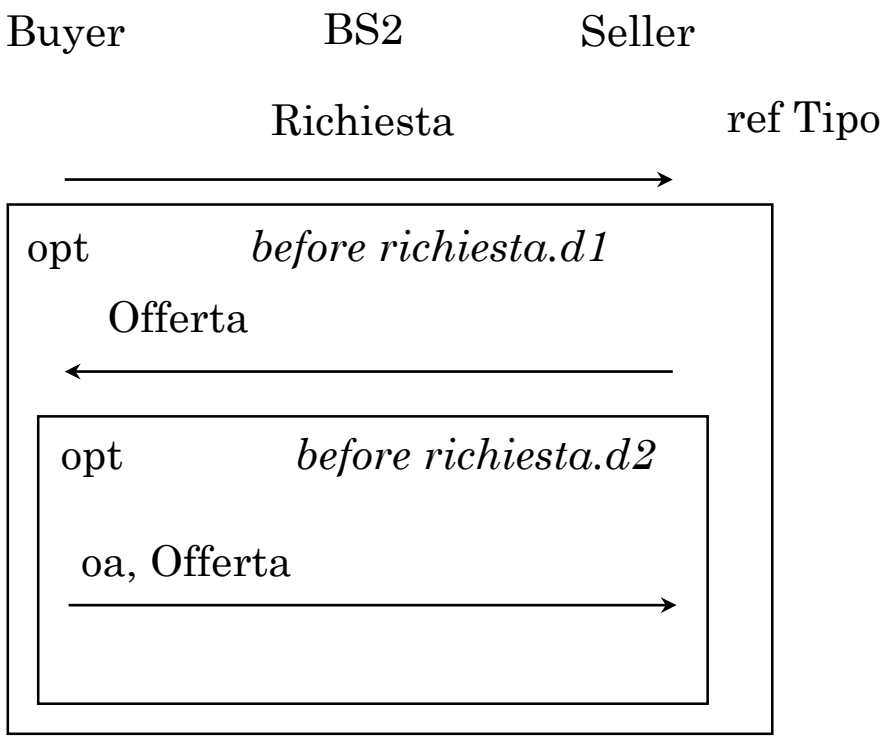
Blocco opt event-driven



Dopo Transazione possono essere eseguite ti oppure ta. Nel secondo caso il blocco opt è saltato.

tr,ta,ti,tm = transazione respinta, accettata, incompleta, modificata

Blocco opt time-driven



Attributi
Richiesta: Date d1; Date d2.

Se l'offerta non è *inviata/ricevuta* entro il termine fissato nell'attributo d1 della richiesta, il blocco opt è saltato.

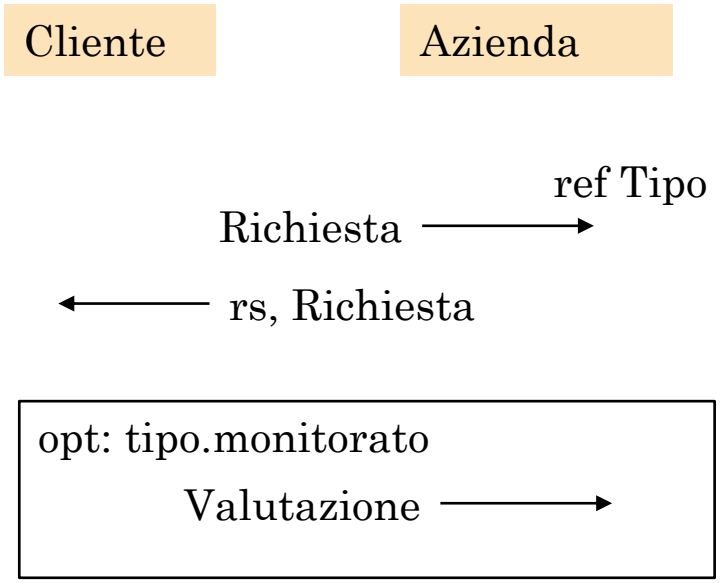
Blocco opt data-driven

Un'azienda riceve richieste di servizio dai suoi clienti. Una richiesta si riferisce ad un tipo di servizio. Un tipo di servizio include l'attributo booleano monitorato.

Eseguito il servizio, l'azienda comunica al cliente che la richiesta è stata soddisfatta e nel caso in cui il tipo sia monitorato il cliente invia la sua valutazione del servizio ottenuto.

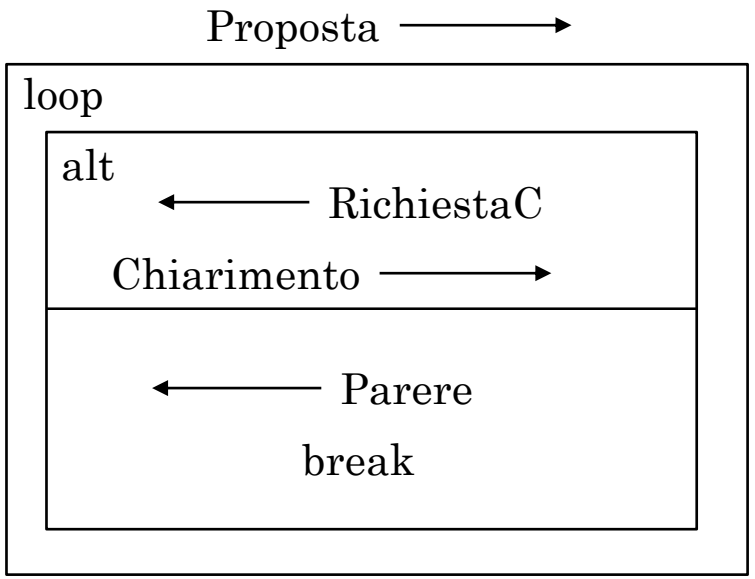
rs = richiesta soddisfatta

Il blocco opt è eseguito se è vera la condizione che segue la keyword opt; altrimenti è saltato.



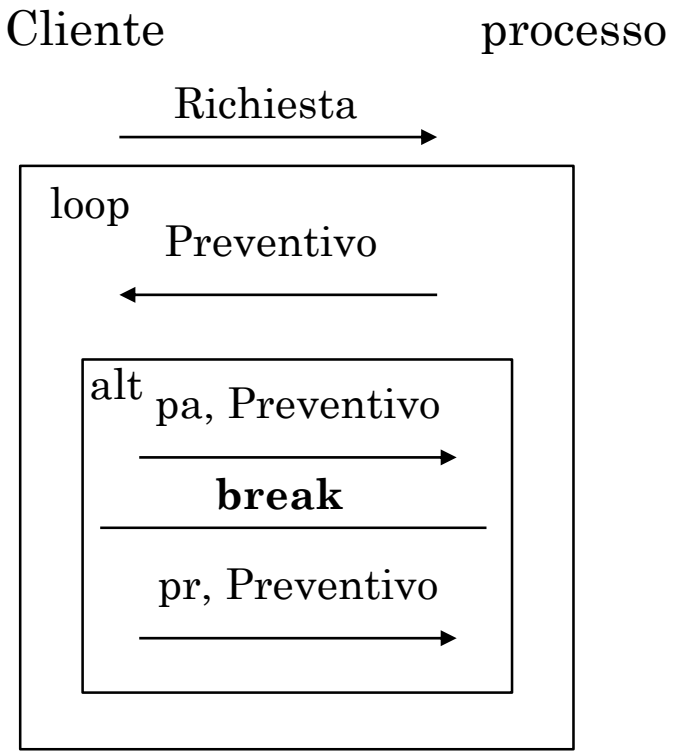
Attributi
Tipo: boolean monitorato.

Loop



La prima sezione del loop, ripetibile, può non essere eseguita. Con il comando break si esce dal loop. Se il loop è l'ultimo blocco, termina la collaborazione.

I = Initiator, F = Follower



pa, pr = preventivo accettato, rifiutato

L'interazione Preventivo è eseguita almeno una volta.

L'operatore par

```
Studio        processo
-> Progetto
loop par
  alt
     <- RichiestaC
     -> rs, RichiestaC
  ---
     <- pa, Progetto
     break
```

```
rs = richiesta soddisfatta
pa = progetto accettato
```

loop par indica un loop con iterazioni concorrenti.
Uno studio può ricevere contemporaneamente varie richieste di chiarimento (RichiestaC) relative ad un progetto.
Il loop termina quando lo studio riceve il messaggio *pa*.
Il messaggio pa è inviato quando non ci sono più interazioni RichiestaC e rs.

Uso dei modelli di collaborazione

Le organizzazioni interagiscono mediante processi interni che usano protocolli di collaborazione basati su modelli di collaborazione.

I prossimi esempi riguardano il processo di un buyer e quello di un seller che interagiscono sulla base dei modelli di collaborazione precedenti.

Il processo contiene dei task d'interazione che eseguono le interazioni con i collaboratori. Tali task compaiono soltanto nelle swimlane delle collaborazioni.

Queste swimlane contengono soltanto task d'interazione.

Task d'interazione

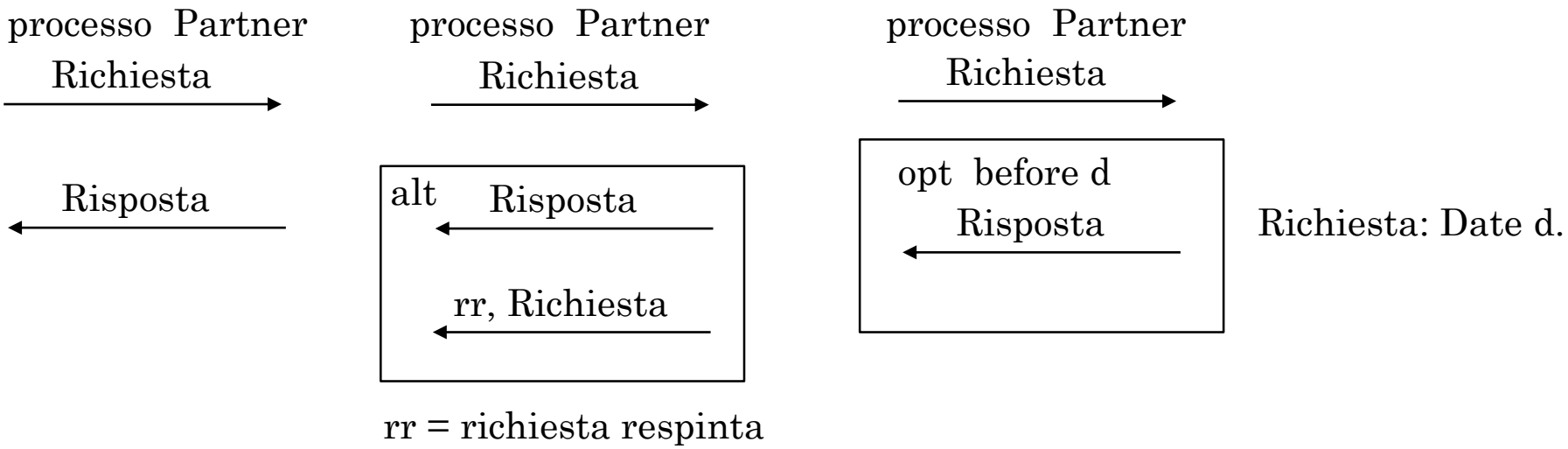
I task d'interazione, indicati con taskI, si dividono in task di output, task di input e task doppi (di output e di input). I loro nomi si riferiscono ad interazioni di output, di input e di output e input rispettivamente.

Esempio 1

Il processo considerato invia un richiesta ad un partner che

1. risponde con una risposta;
2. risponde con una risposta o dichiara respinta la richiesta;
3. può rispondere con una risposta entro la scadenza d della richiesta (quindi la risposta è opzionale).

3 modelli di collaborazione



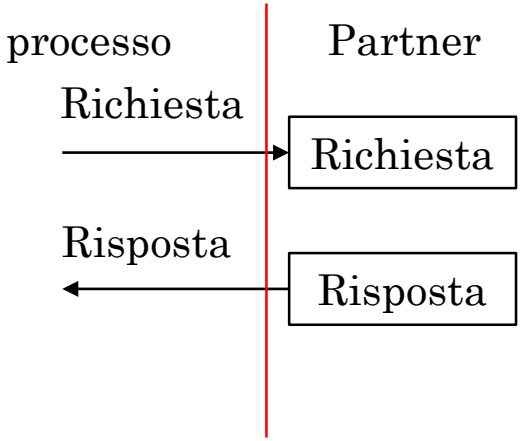
rr = richiesta respinta

La richiesta è collegata ad un solo partner.

swimlane del processo

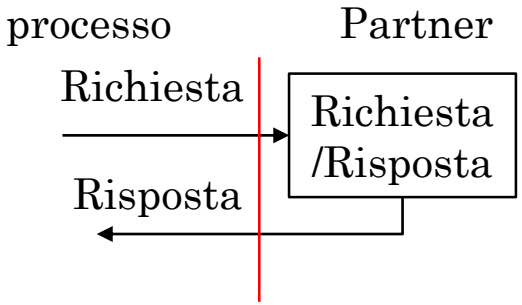
swimlane del partner (contiene i taskI relativi al partner)

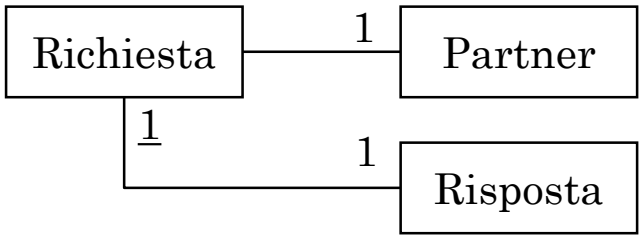
Esempio 1 a



TaskI singoli (di output e di input)

L'effetto del taskI Risposta è la generazione di una nuova risposta associata alla richiesta.





TaskI doppio; in output la richiesta e in input la risposta associata alla richiesta.

Esempio 1 b

swimlane

processo Partner

Richiesta → Richiesta

Risposta ← Risposta

Richiesta ← rr

3 taskI singoli

swimlane

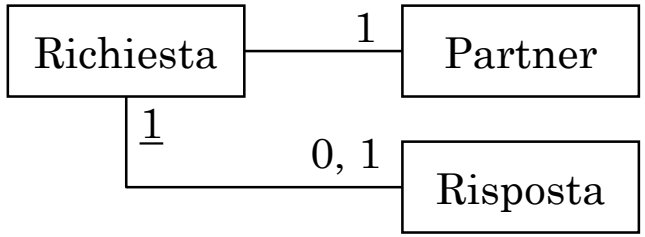
processo Partner

Richiesta → Richiesta/ Risposta,rr

Risposta ←

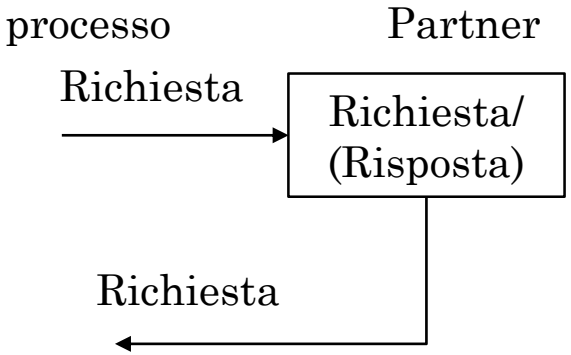
rr → Richiesta,respinta

TaskI doppio con **risposta multipla**.

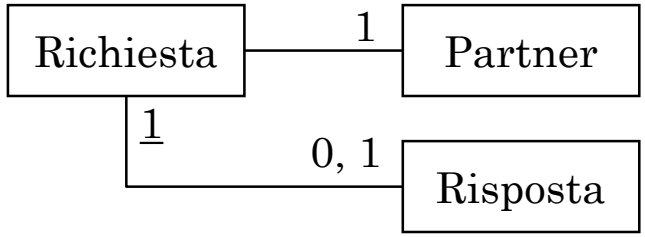


Esempio 1 c

swimlane



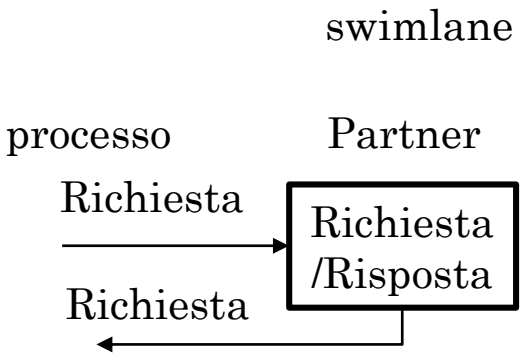
TaskI doppio con **risposta opzionale** tra parentesi. L'output è la richiesta perché la risposta potrebbe non essere stata inviata. La risposta è collegata alla richiesta.

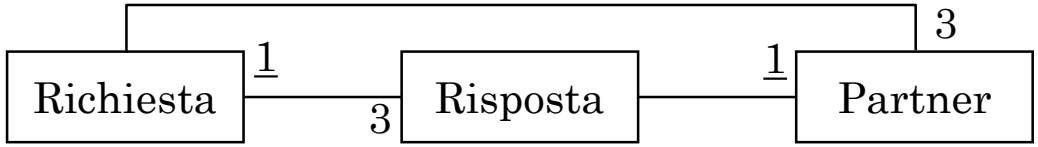


Richiesta: Date d.

Esempio 2 a

Il processo interagisce con 3 partner collegati alla richiesta.

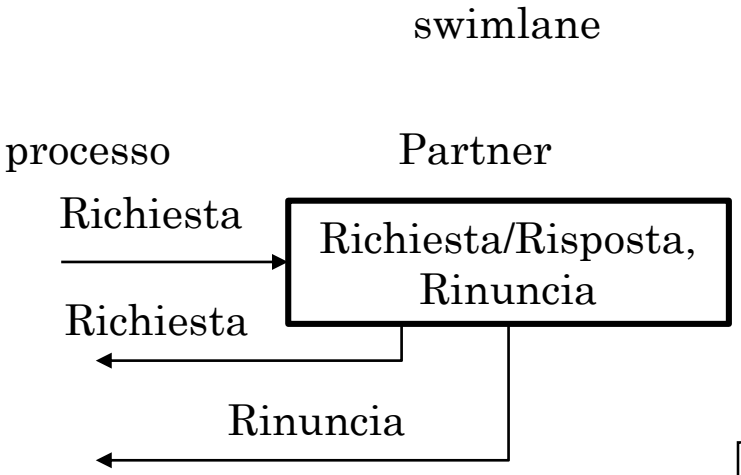




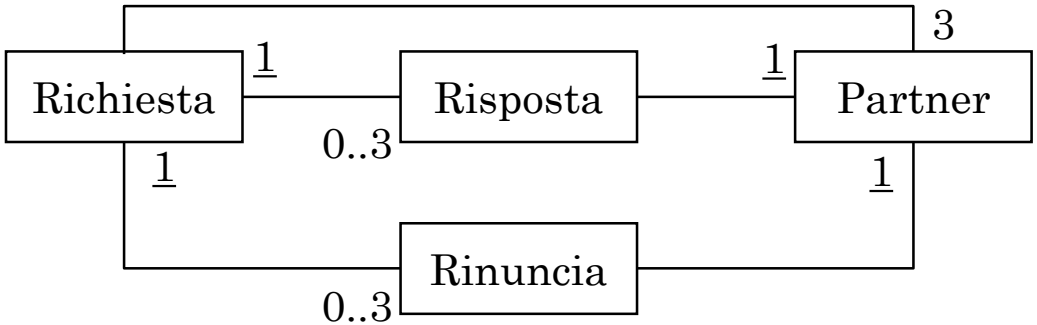
TaskI doppio multi-destinatario;
in output la richiesta alla quale sono associate le risposte.

Esempio 2 b

Variante: il partner risponde con una risposta o con una rinuncia.

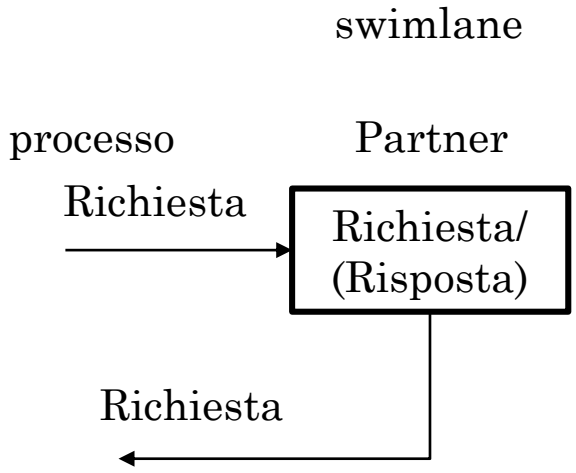


TaskI doppio multi-destinatario; in output la richiesta alla quale sono associate le risposte e le rinunce.



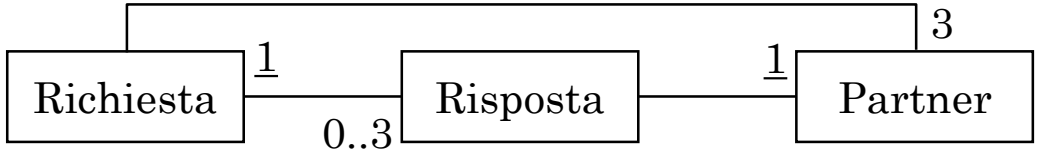
Esempio 2 c

Le risposte sono opzionali.



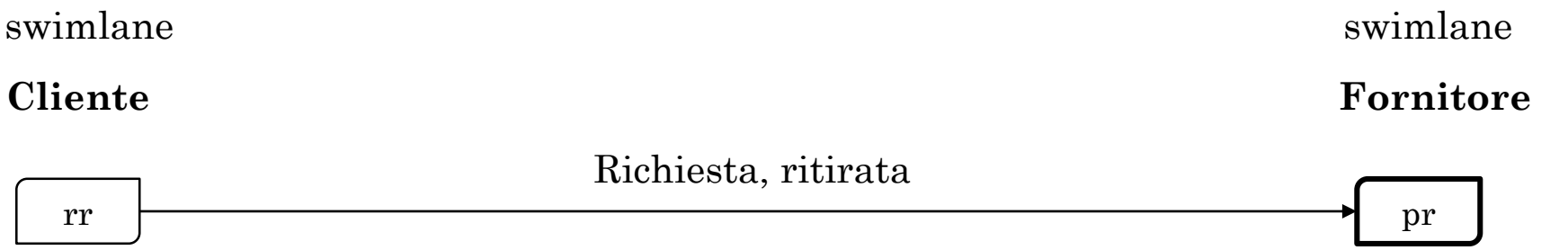
TaskI doppio multi-destinatario con risposta opzionale; alla scadenza indicata nella richiesta emette la richiesta alla quale sono associate le risposte pervenute.

Richiesta: Date d.



Mapping tra task d'interazione

Il cliente ha inviato una richiesta ad un fornitore che ha risposto con un preventivo.
Il cliente ritira la richiesta. Il taskI rr emette la richiesta ritirata. Il taskI pr invia il preventivo respinto al mittente. Il processo esegue un mapping da richiesta a preventivo.



rr = richiesta ritirata
pr = preventivo respinto

Buyer e seller

Modelli informativi e processi

Buyer1

Il modello di collaborazione è BS1.

Nel buyer, ogni funzionario può emettere una richiesta.

Nel sistema informativo sono registrati i funzionari, i seller, i tipi di prodotti e i tipi trattati dai seller.

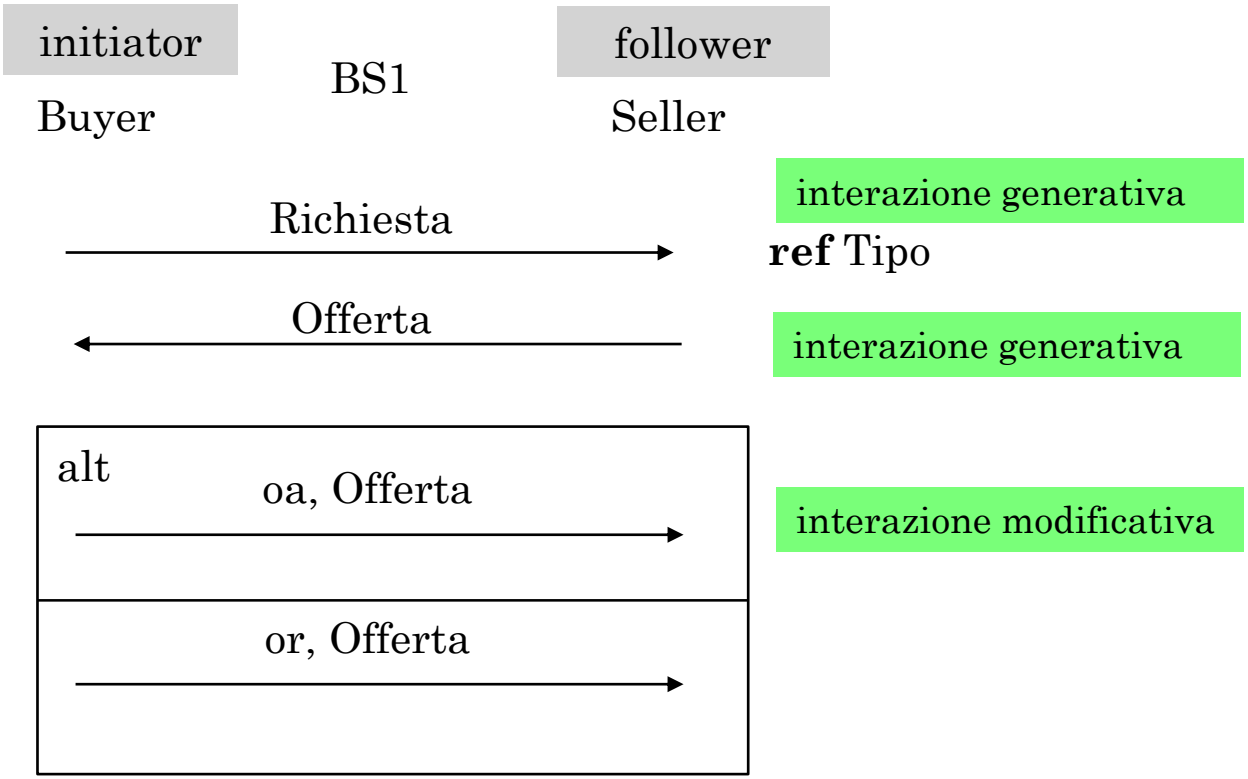
L'ufficio acquisti associa una richiesta a 1 seller idoneo (si usi un invariante per stabilire il vincolo). Poi il processo invia la richiesta al seller che risponde con un'offerta.

L'ufficio acquisti può accettare o respingere l'offerta. Il funzionario legge la richiesta trattata.

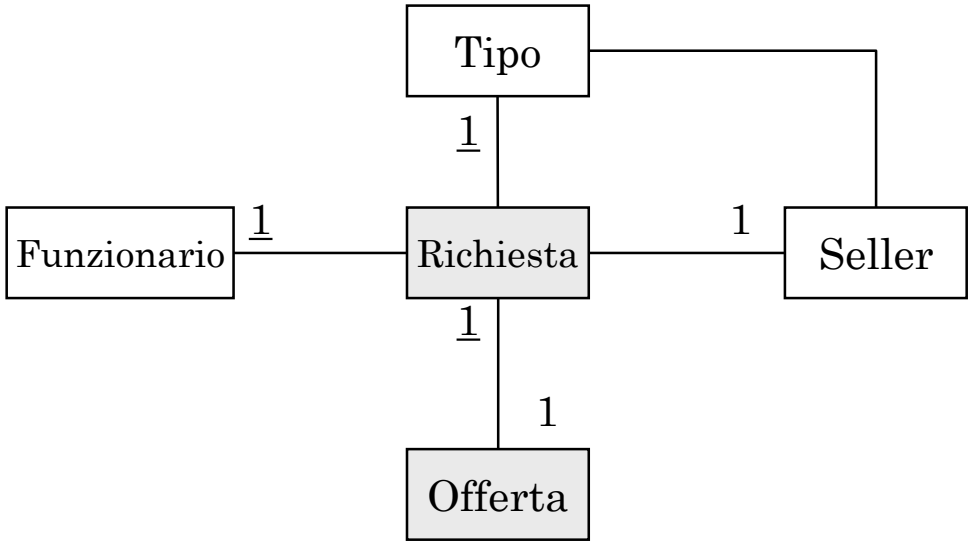
Le offerte hanno uno stato (accettata, respinta).

Collaborazione buyer-seller BS1

Il buyer manda al seller una richiesta di offerta relativa ad un tipo di prodotto offerto dal seller. Il seller risponde con un'offerta. Il buyer può accettare o rifiutare l'offerta.

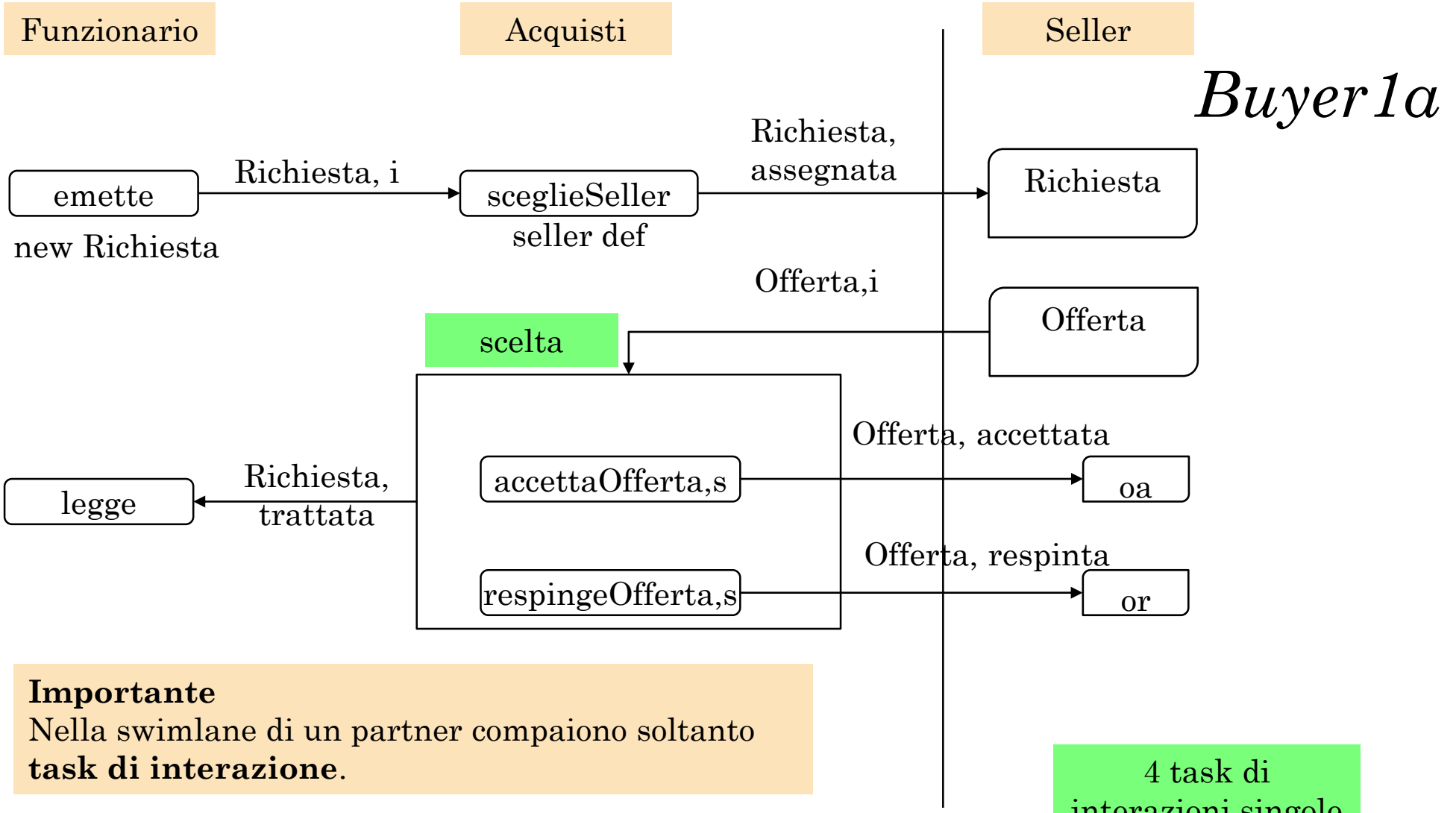


Modello informativo del Buyer1



Offerta: stato (accettata, respinta).

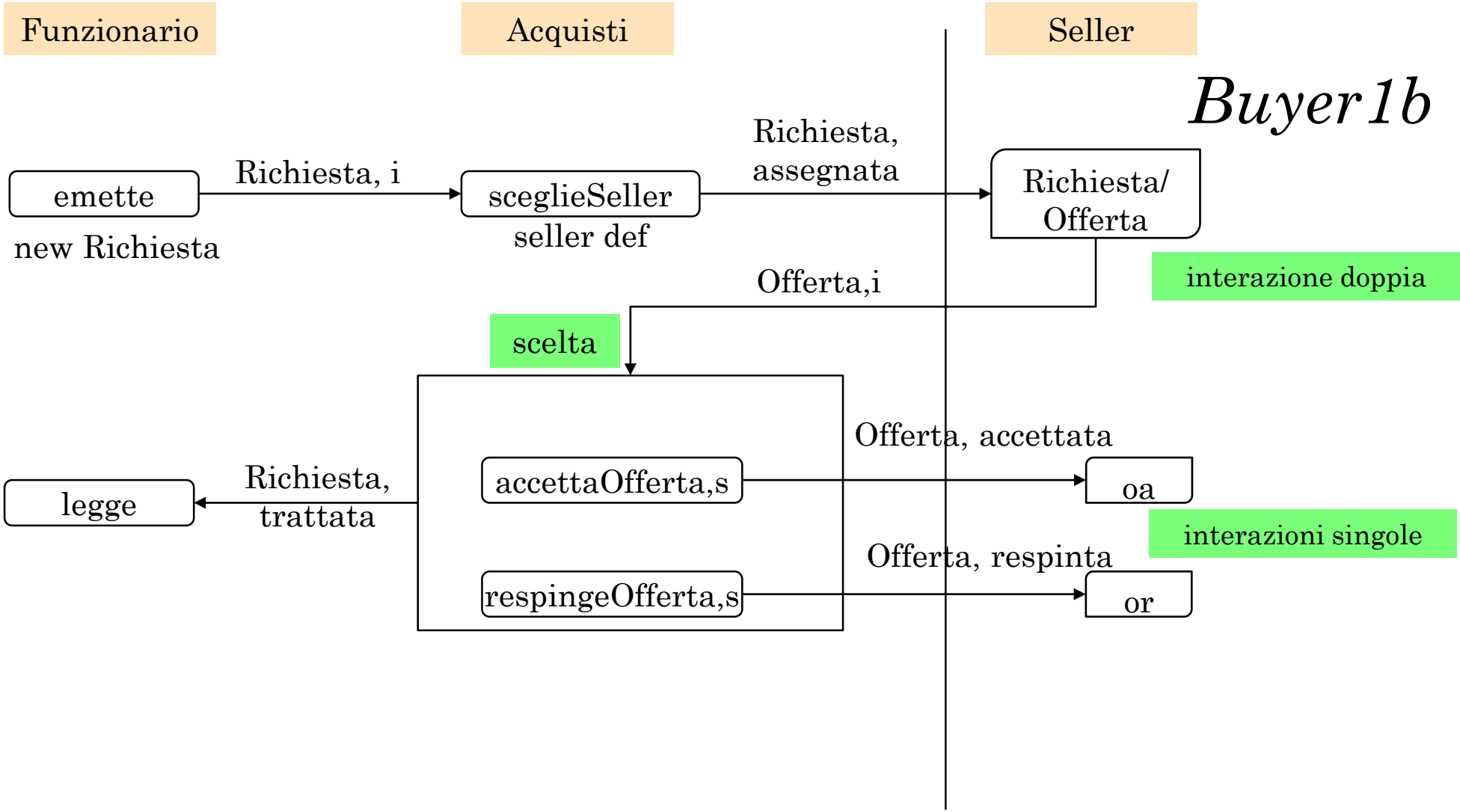
Invariante:
richiesta.seller in richiesta.tipo.sellers.



Importante
Nella swimlane di un partner compaiono soltanto **task di interazione**.

4 task di interazioni singole

Nota: il task d'interazione Offerta genera un'offerta e la collega alla richiesta.



Nota: il task d'interazione doppia genera un'offerta e la collega alla richiesta.

Buyer 2

Il modello di collaborazione è BS1.

Buyer 2a

Nel buyer, ogni funzionario può emettere una richiesta.

Nel sistema informativo sono registrati i funzionari, i seller, i tipi di prodotti e i tipi trattati dai seller.

L'ufficio acquisti associa una richiesta a 3 seller idonei (si usi un invariante per stabilire il vincolo). Poi il processo invia la richiesta ai seller. Ricevute le offerte, l'ufficio acquisti ne accetta una e respinge le altre. Il funzionario legge la richiesta trattata.

Le offerte hanno uno stato (accettata, respinta).

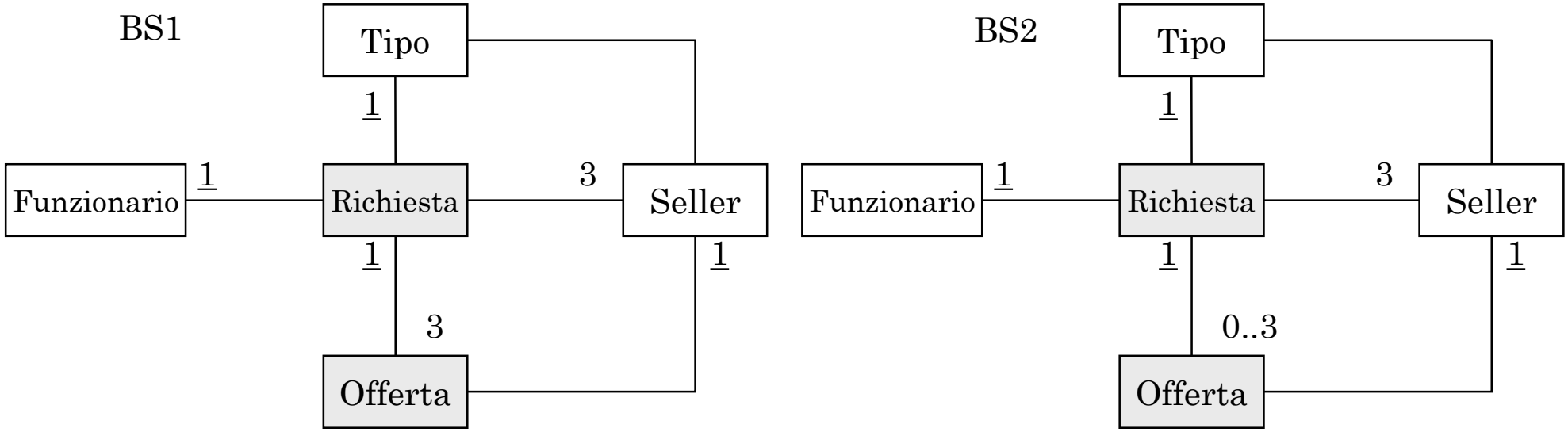
Variante

Buyer 2b

Buyer con il modello di collaborazione BS2

Ricevute le offerte (se ce ne sono), l'ufficio acquisti ne accetta una e respinge le altre o le respinge tutte.

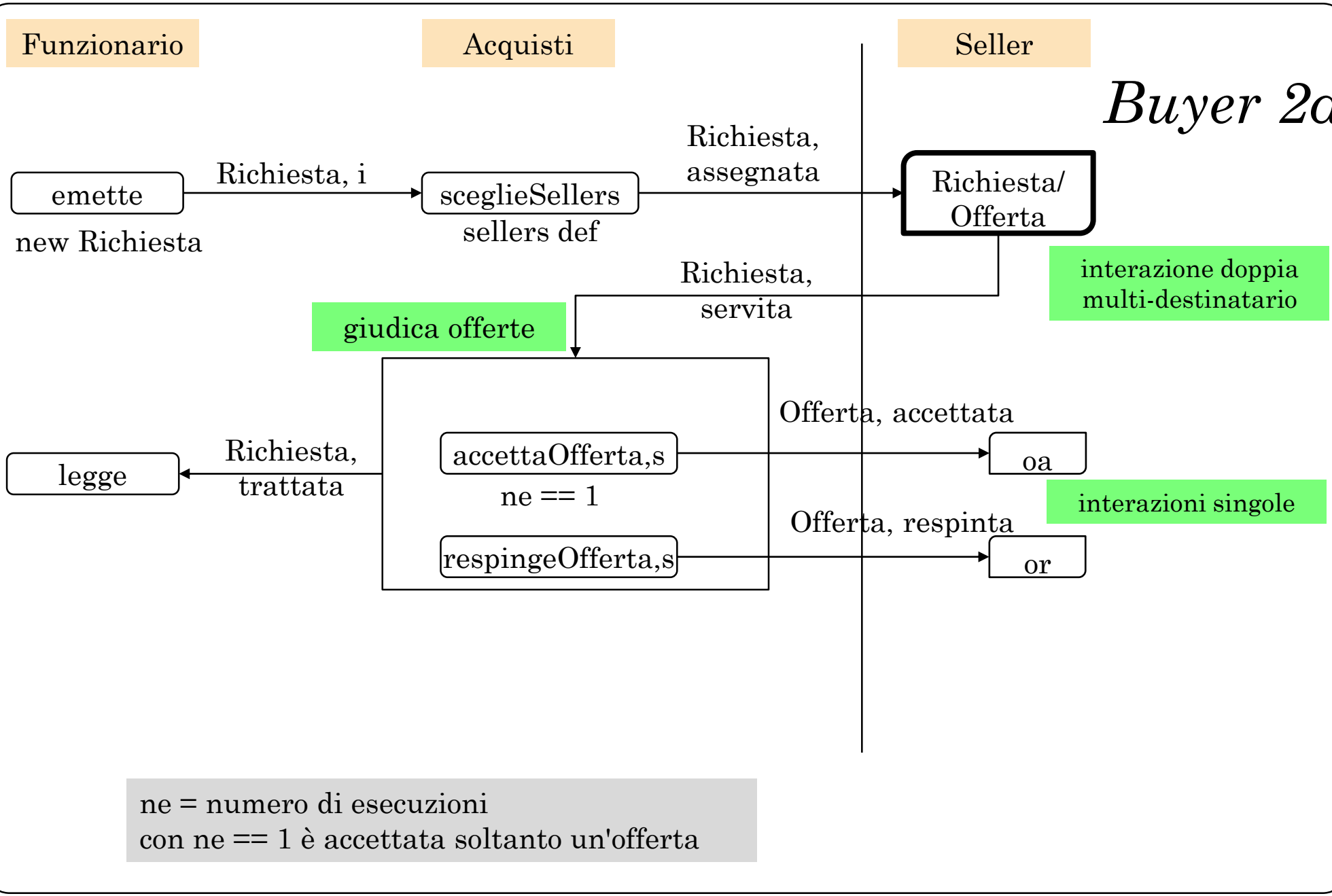
Modello informativo del Buyer 2



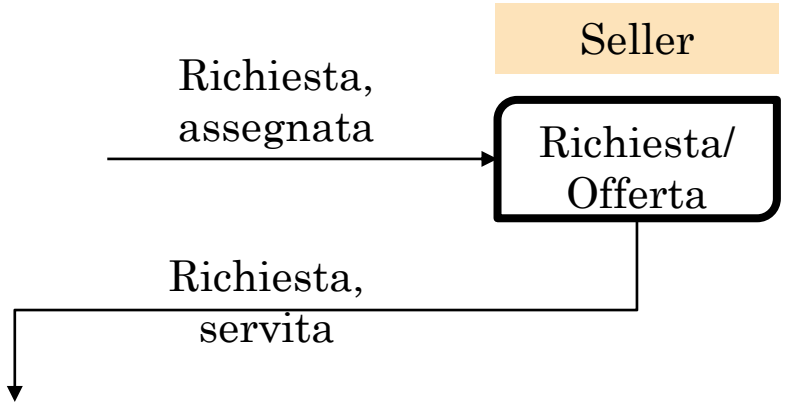
Offerta: stato (accettata, respinta).

Invariante:
richiesta.sellers in richiesta.tipo.sellers.

Nota: con BS2 una richiesta è collegata a 0..3 offerte.
Richiesta: Date d1.


Funzionario
Acquisti
Seller
Buyer 2a
emette
new Richiesta
Richiesta, i
sceglieSellers
sellers def
Richiesta, assegnata
Richiesta/ Offerta
interazione doppia multi-destinatario
Richiesta, servita
giudica offerte
accettaOfferta,s
ne == 1
respingeOfferta,s
Offerta, accettata
oa
Offerta, respinta
or
Richiesta, trattata
legge
interazioni singole
ne = numero di esecuzioni
con ne == 1 è accettata soltanto un'offerta

Task di interazione doppia multi-destinatario



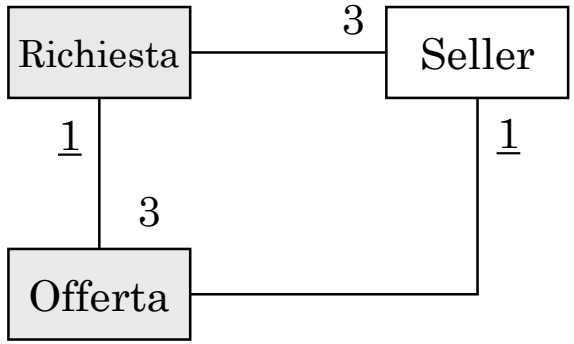
Il tipo del destinatario è indicato dal nome della swimlane (Seller).

Quali sono i destinatari effettivi? Quelli collegati direttamente o indirettamente (con una relazione derivata) alla richiesta.
In questo caso sono 3. Quindi il task invia la richiesta a 3 seller e riceve le loro offerte. Si può ispessire il bordo.

Che cosa consegna il task al processo?
Potrebbe consegnare le offerte man mano che arrivano o tutte in blocco.
Dipende dal tipo dell'uscita: in questo caso consegna la richiesta con le 3 offerte collegate.

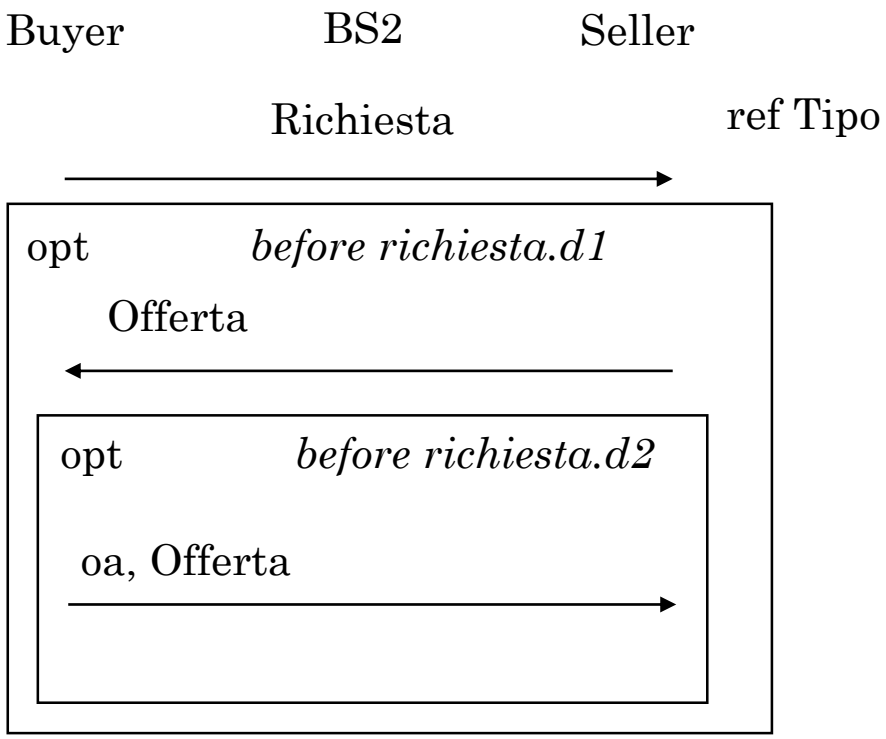
Task di interazione doppia multi-destinatario

Nel modello di collaborazione l'offerta segue la richiesta. Il seller invia un'offerta come risposta ad una richiesta: un'offerta è correlata ad una richiesta. La correlazione è basata sugli identificativi delle entità.
Sulla base del modello informativo, il task d'interazione quando riceve un messaggio Offerta genera una nuova entità Offerta e la collega alla richiesta correlata all'offerta e al seller che ha inviato l'offerta. Per stabilire questo comportamento, nel modello informativo ci sono le relazioni obbligatorie Offerta – Richiesta, Offerta – Seller.



Variante: offerta e accettazione opzionali (BS2)

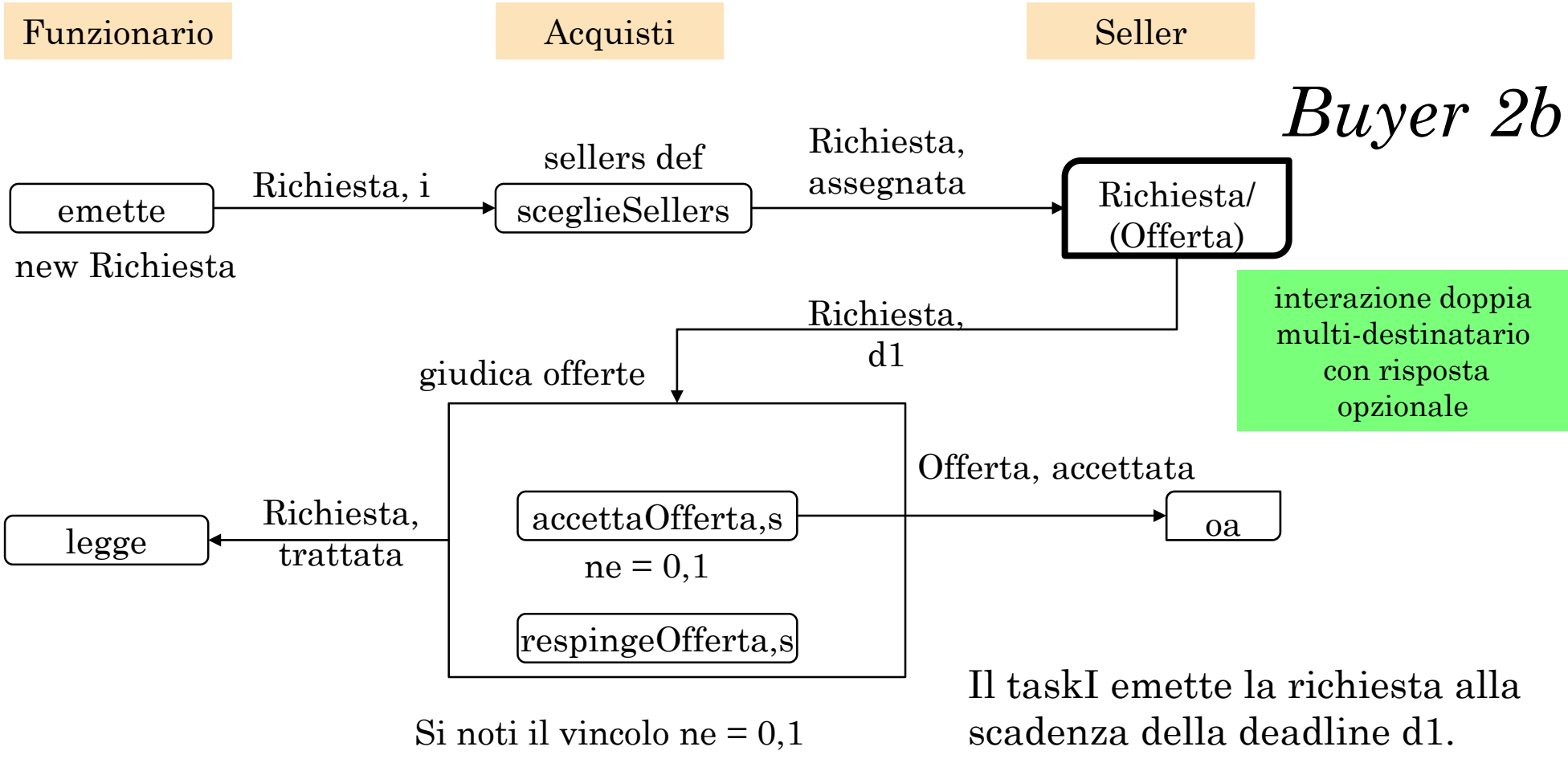
La richiesta include due deadline, d1 and d2. L'offerta va inviata entro d1 e l'accettazione entro d2, altrimenti la collaborazione si considera conclusa.



Un blocco opzionale può essere saltato. Il primo blocco è saltato se l'offerta non è ricevuta prima della scadenza indicata nell'attributo d1 della richiesta.

Attributi
Richiesta: Date d1; Date d2.

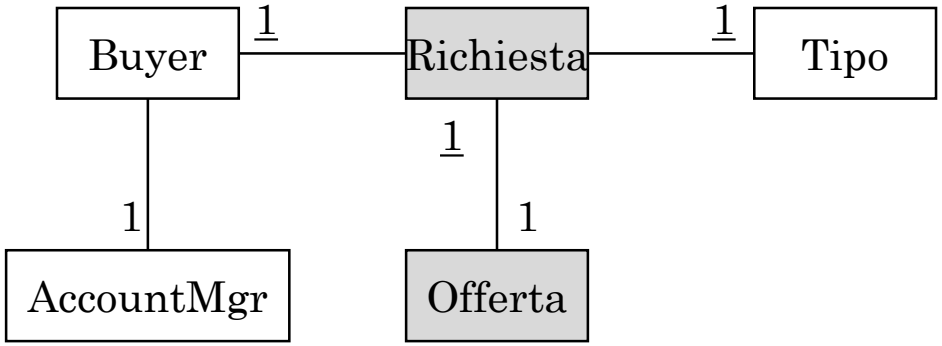
Nota: la collaborazione termina con oa, oppure a d1 (se non è inviata l'offerta), oppure a d2 (se non è eseguita oa). Gli attributi d1 e d2 sono visibili nell'istanza di collaborazione.



Il taskI emette la richiesta alla scadenza della deadline d1.
La richiesta in output di *giudica offerte* può non avere alcuna offerta, può averne una accettata o nessuna accettata.

Modello informativo del Seller

Nel seller, la richiesta è indirizzata all'account manager del buyer, il quale produce l'offerta e la invia al buyer (tramite un task di interazione). L'offerta potrà essere accettata o respinta.

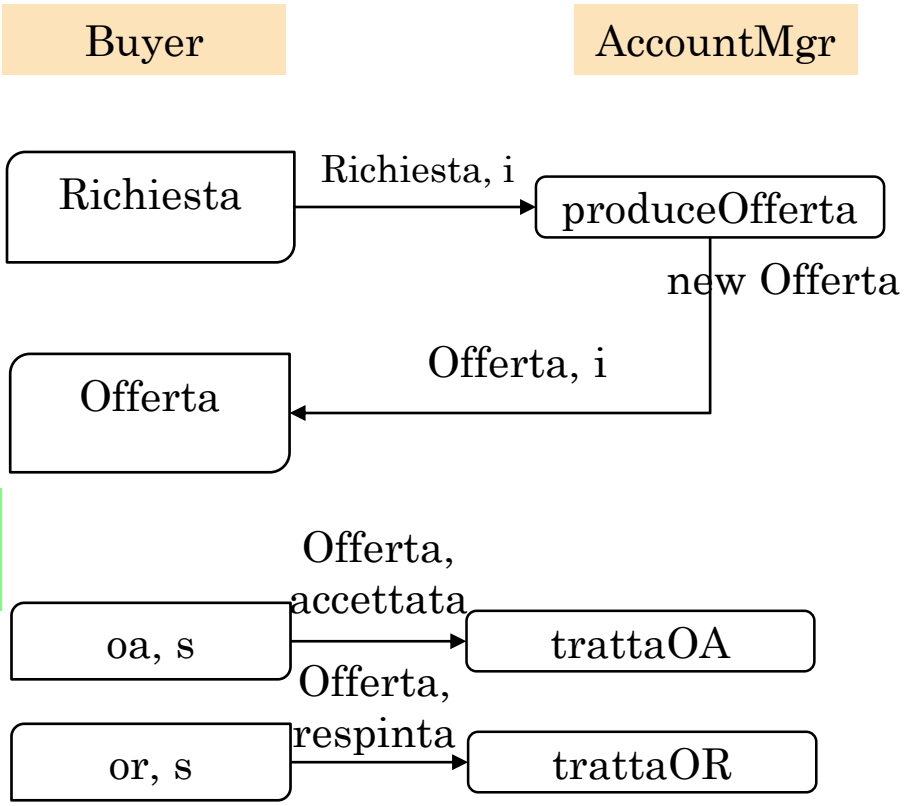


Offerta: stato (accettata, respinta).

Con BS2 l'offerta è opzionale (0..1).
Se l'offerta è stata inviata, lo stato dell'offerta sarà accettata o null.

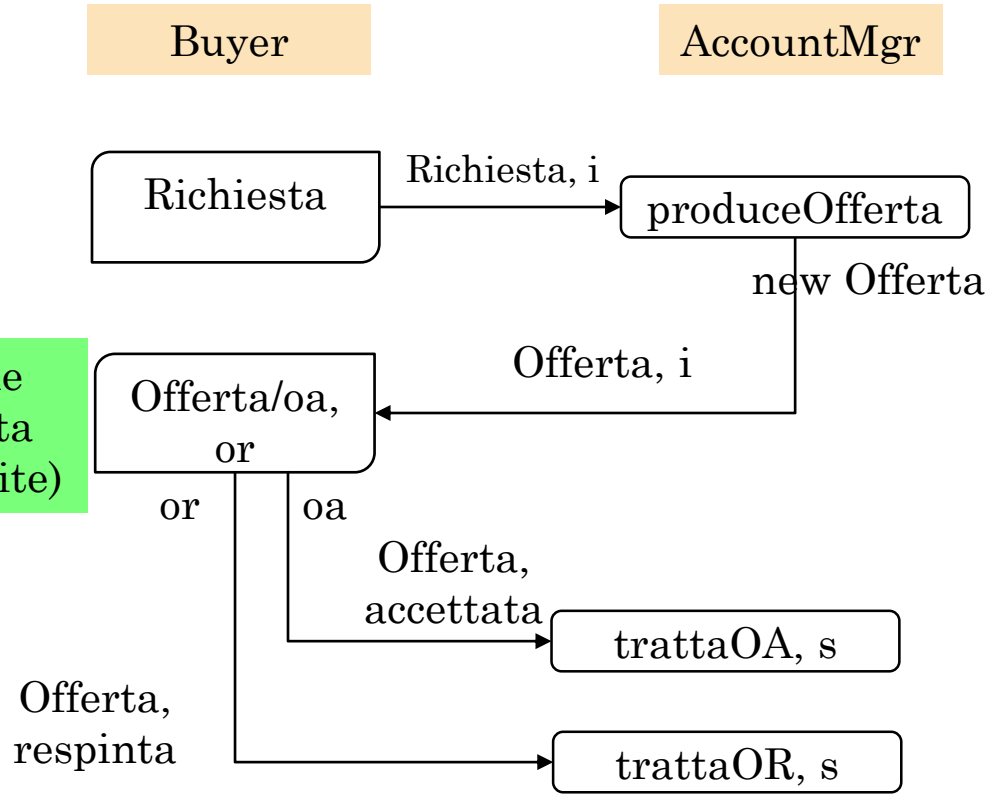
Una richiesta è collegata all'entità che rappresenta il mittente. Ecco il motivo della relazione obbligatoria Richiesta – Buyer.
L'offerta è una risposta ad una richiesta: ecco il motivo della relazione obbligatoria Offerta – Richiesta.
L'autore dell'offerta è determinata dalla relazione derivata
Offerta – AccountMgr = Offerta – Richiesta – Buyer – AccountMgr.

Seller 1a (BS1)



Il task d'interazione **Richiesta** genera una nuova richiesta collegata al buyer e al tipo di prodotto desiderato. Il task **produceOfferta** genera un'offerta collegata alla richiesta: la relazione Richiesta – Offerta esprime sia un rapporto di causalità sia un rapporto di correlazione.
I task d'interazione *oa* e *or*, seguiti da *s* definiscono lo stato dell'offerta (accettata o respinta).

Seller 1b (BS1)

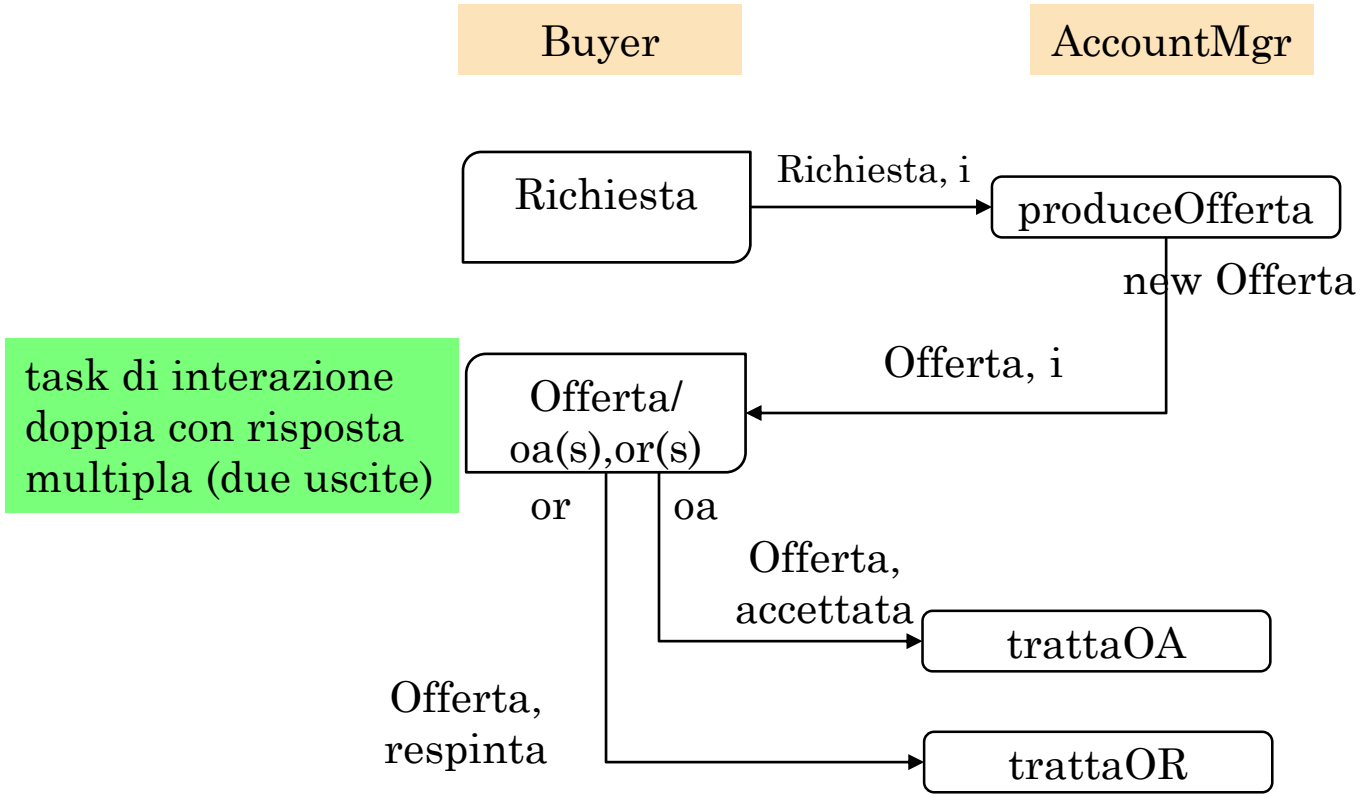


task di interazione doppia con risposta multipla (due uscite)

I due task definiscono lo stato dell'offerta.

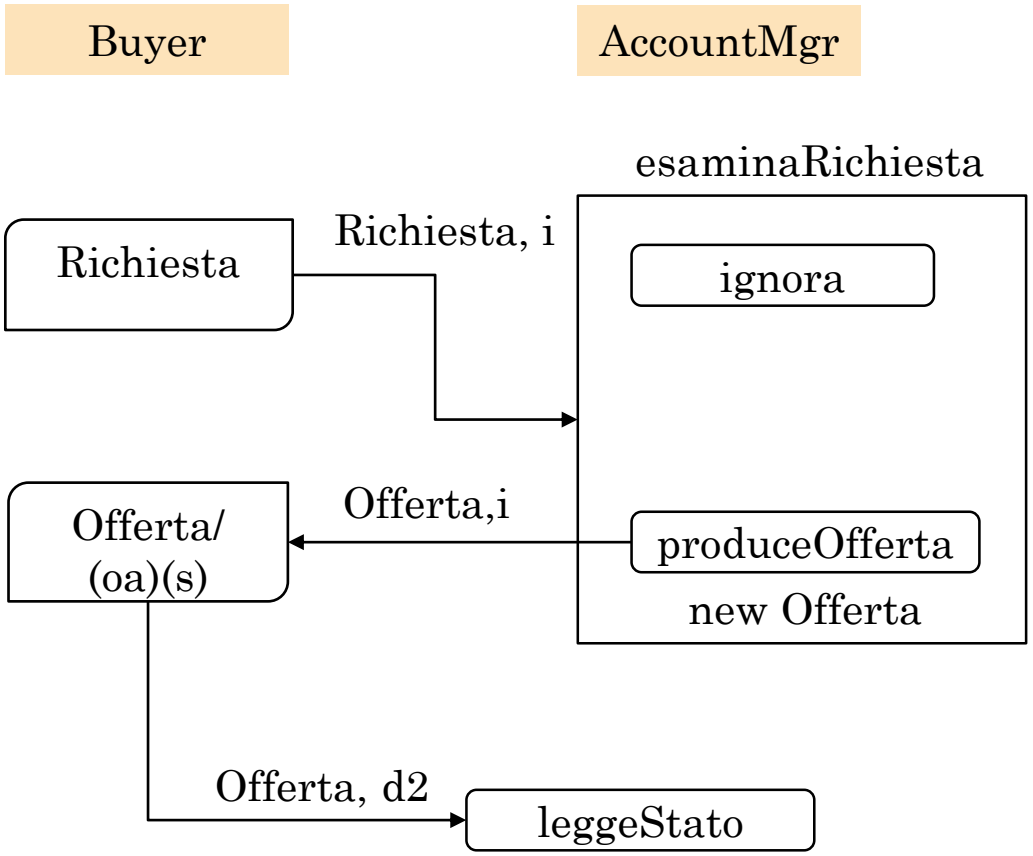
Il **task di interazione doppia** invia l'offerta con l'identificativo della richiesta correlata; in base alla risposta emette la stessa offerta dall'uscita *oa* o dall'uscita *or* e quindi l'offerta si troverà nel posto *accettata* o nel posto *respinta*.

Variante sulla definizione dello stato dell'offerta



L'aggiunta di (s) alle interazioni entranti *oa* e *or* permette al task d'interazione di definire lo stato dell'offerta prima di immetterla in una delle due uscite.

Seller (BS2)



L'account mgr può ignorare la richiesta.

Il task d'interazione doppia emette l'offerta il cui stato può essere null oppure accettata. Lo stato è null se l'interazione *oa* non è stata eseguita; in caso contrario lo stato è accettata.
La clausola *(s)* indica lo stato dell'offerta.

Collaborazione con loop

Riduttore

Gestione Proposte B2B

Un'organizzazione riceve proposte dai partner. Nel sistema informativo dell'organizzazione sono registrati i partner, i revisori e gli account manager insieme con le associazioni con i partner.

Una proposta è trattata dall'account manager relativo al partner: può accettarla, respingerla o assegnarla a 3 revisori. Nei primi 2 casi, il processo comunica al partner che la proposta è stata accettata o respinta, nel terzo caso invia la proposta ai revisori.
Un revisore può chiedere un chiarimento e l'account manager glielo fornisce; oppure può esprimere il suo parere sulla proposta. Ricevuto un chiarimento, può chiederne un altro, e così via finché non esprime il suo parere sulla proposta. Il parere può essere favorevole o sfavorevole.

Ricevuti i 3 pareri, l'account manager accetta la proposta se ci sono almeno due pareri favorevoli, altrimenti la respinge.
Il processo comunica al partner che la proposta è stata accettata o respinta.

La proposta ha come stato finale *accettata o respinta*.

Note

Un revisore può chiedere un chiarimento e l'account manager glielo fornisce

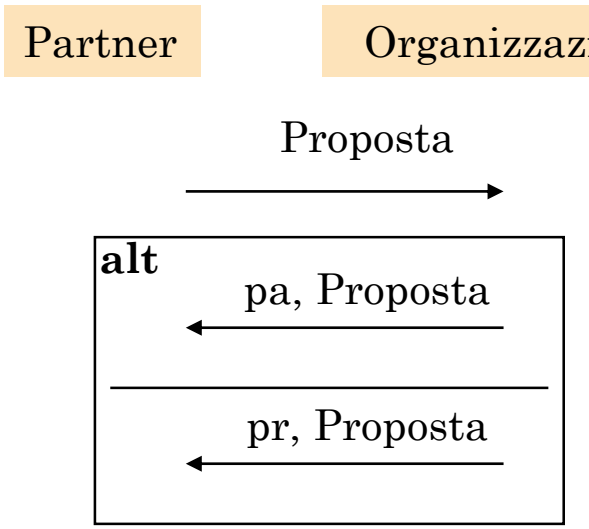
significa

Un revisore può inviare una richiesta di chiarimento e l'account manager glielo fornisce

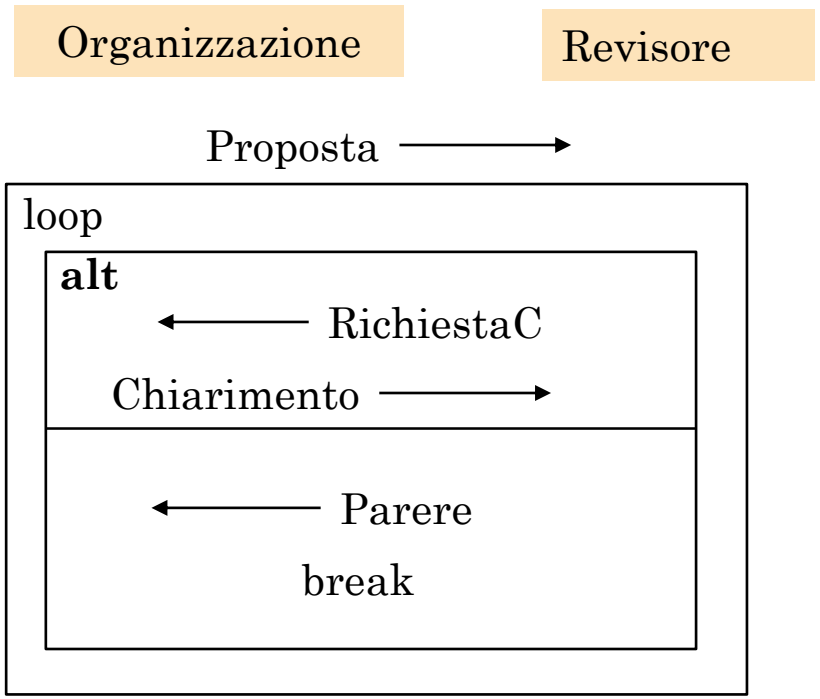
cioè l'account manager produce un chiarimento e il processo lo invia al revisore.

L’esecutore di un task non invia né riceve direttamente un’entità: l’invio e la ricezione avvengono mediante task di interazione.

Collaborazioni



pa = proposta accettata
pr = proposta respinta

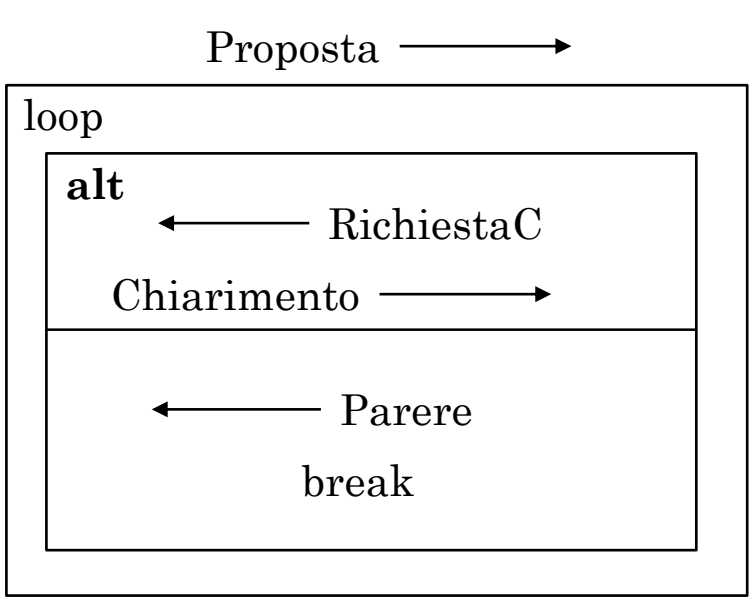


Al posto di break si può usare end.

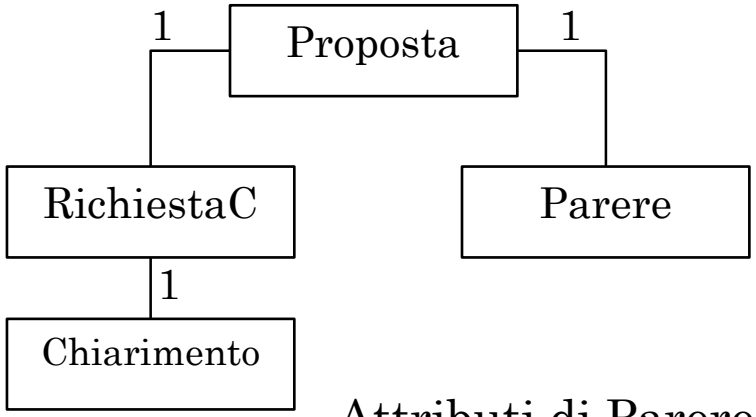
Attributi di Parere: stato (f, sf). f = favorevole, sf = sfavorevole

Nota: il break porta all'elemento (interazione o blocco) successivo al loop o, se questo manca, al termine della collaborazione.

Modello informativo della collaborazione



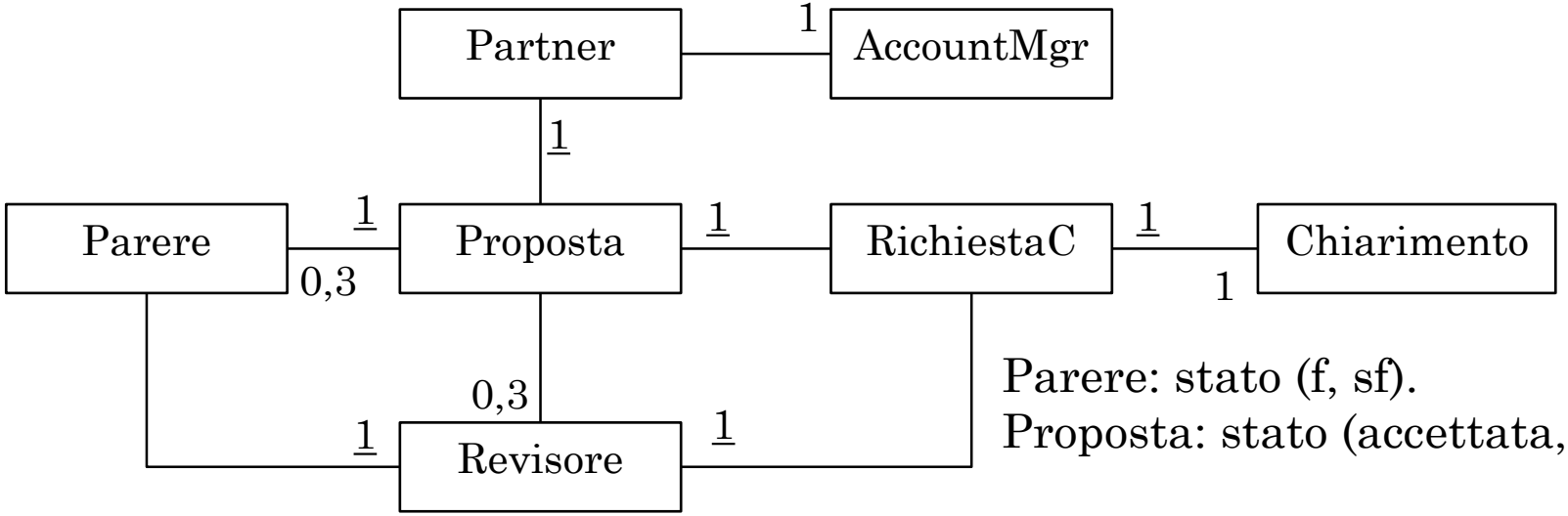
Ha una struttura ad albero, in cui compaiono i tipi delle entità delle interazioni.
Un tipo è correlato a quello che lo precede nell'albero. Nell'esempio, RichiestaC e Parere sono correlati a Proposta, Chiarimento a RichiestaC.
Negli esempi trattati nel corso, ciascun tipo ha un unico predecessore, eccetto quello dell'interazione iniziale che non ne ha.



Albero delle correlazioni
Proposta
RichiestaC Parere
Chiarimento

Attributi di Parere: stato (f, sf). f = favorevole, sf = sfavorevole

Modello informativo del processo



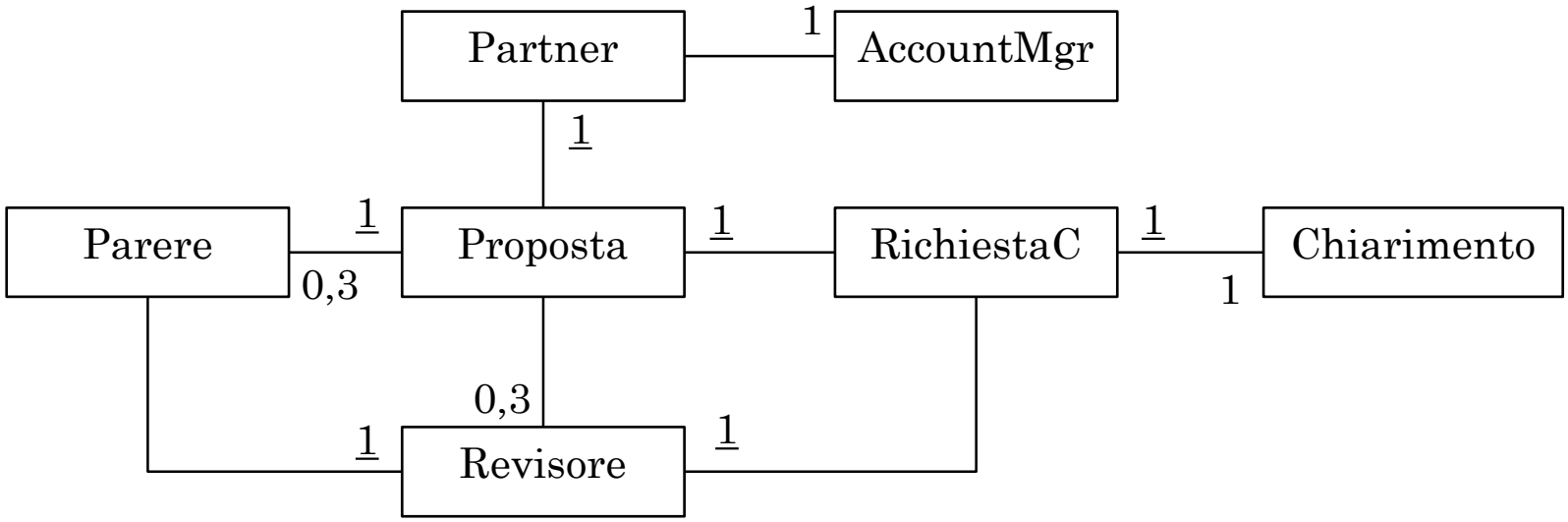
Parere: stato (f, sf).
Proposta: stato (accettata, respinta).

Albero delle correlazioni

Il modello informativo include l'albero delle correlazioni e aggiunge i legami con i partecipanti al processo.

Albero delle correlazioni
Proposta
RichiestaC Parere
Chiarimento

Relazione derivata tra Proposta e AccountMgr



Una proposta è trattata dall'account mgr del partner che l'ha inviata.
Non c'è una relazione diretta tra Proposta e AccountMgr, quindi si dovrebbe definire la relazione derivata Proposta – AccountMgr = Proposta – Partner – AccountMgr .

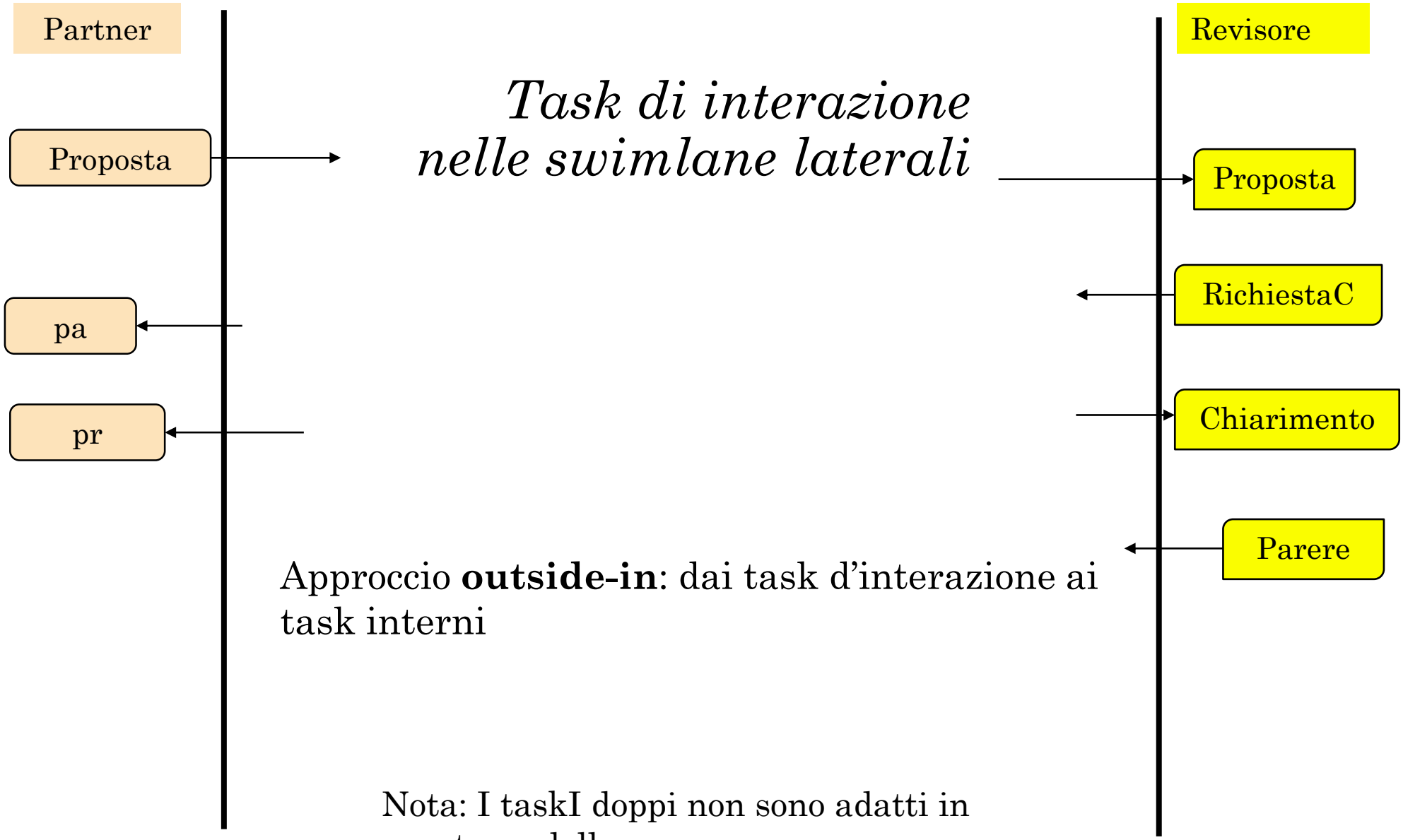
Per semplicità la relazione derivata si può considerare implicita.

Struttura del processo

Ai lati le swimlane delle collaborazioni: contengono soltanto task d'interazione.

Al centro le swimlane dei ruoli con i task interni del processo.

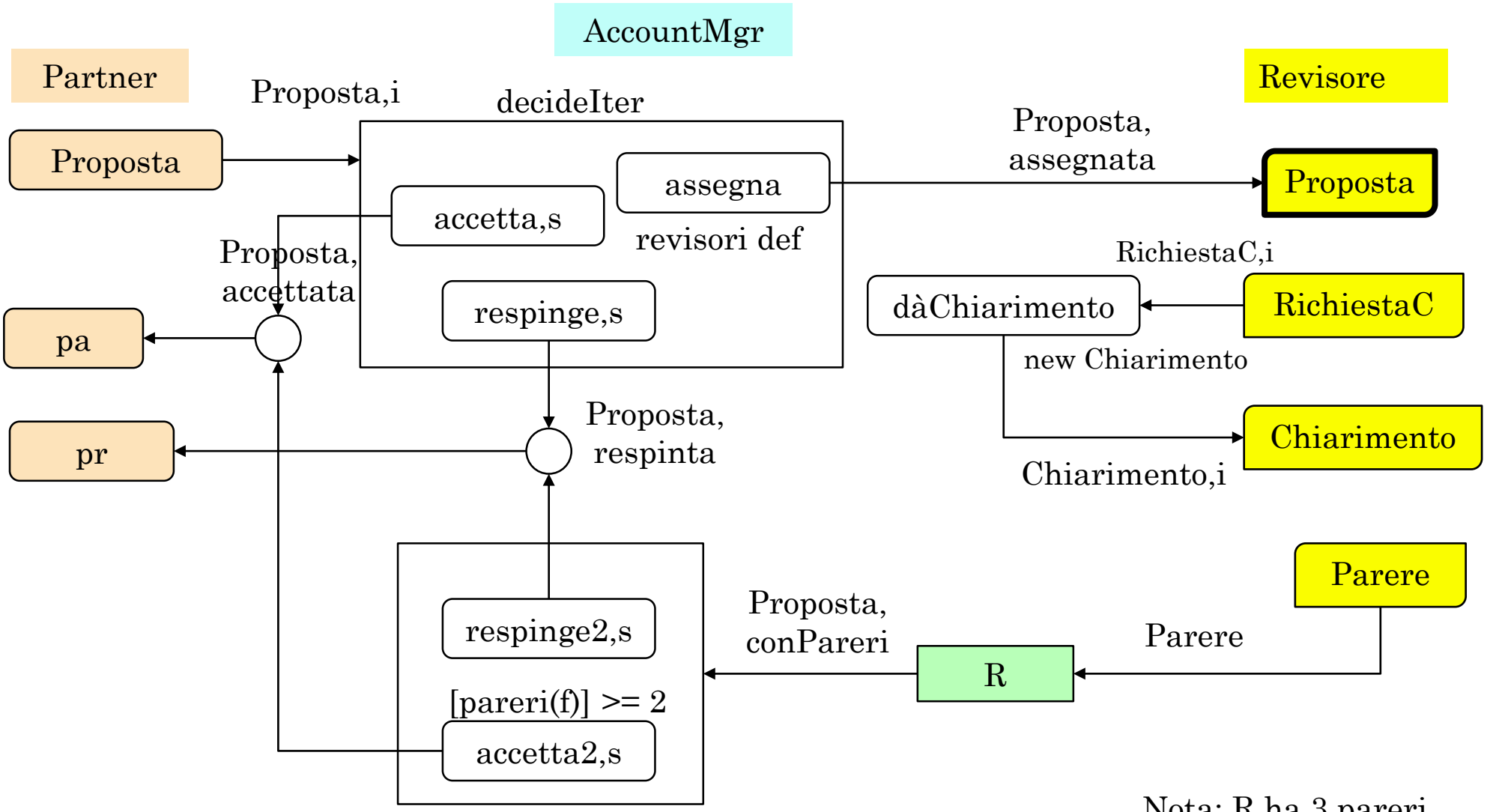
Task di interazione nelle swimlane laterali



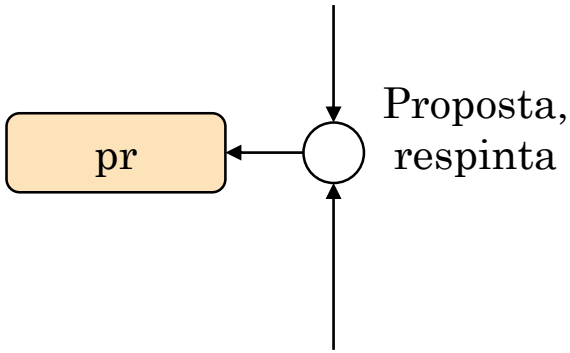
Approccio **outside-in**: dai task d'interazione ai task interni

Nota: I taskI doppi non sono adatti in questo modello.

Processo


AccountMgr
Partner
Revisore
Proposta,i
decideIter
Proposta
assegna
Proposta,
assegnata
Proposta
accetta,s
revisori def
Proposta,
accettata
RichiestaC,i
respinge,s
dàChiarimento
RichiestaC
pa
new Chiarimento
Proposta,
respinta
pr
Chiarimento
Chiarimento,i
Parere
respinge2,s
Proposta,
conPareri
Parere
R
[pareri(f)] >= 2
accetta2,s
Nota: R ha 3 pareri

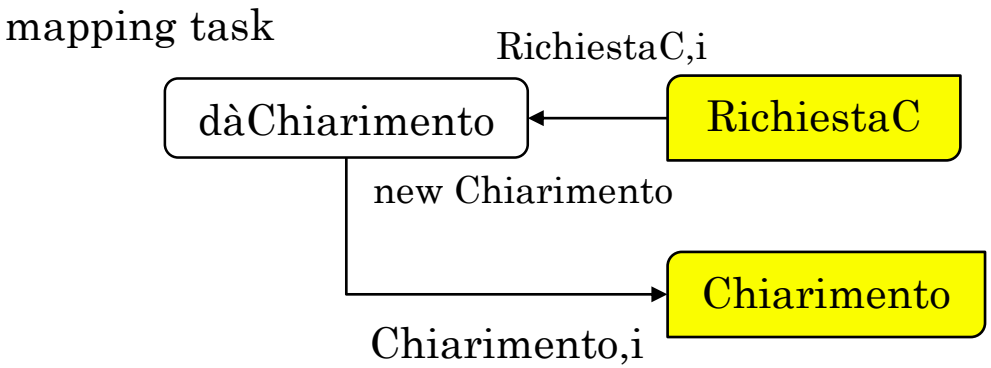
Nota



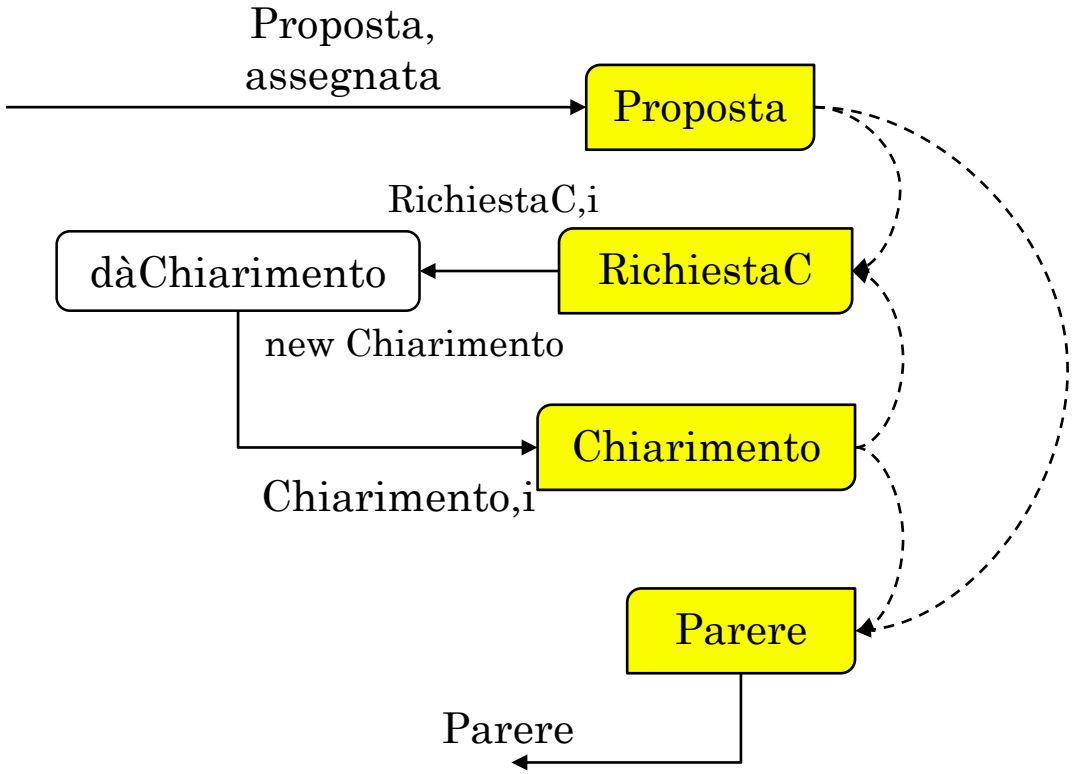
La confluenze sono necessarie perché i task di interazione hanno un solo ingresso.

Modifica del dataflow

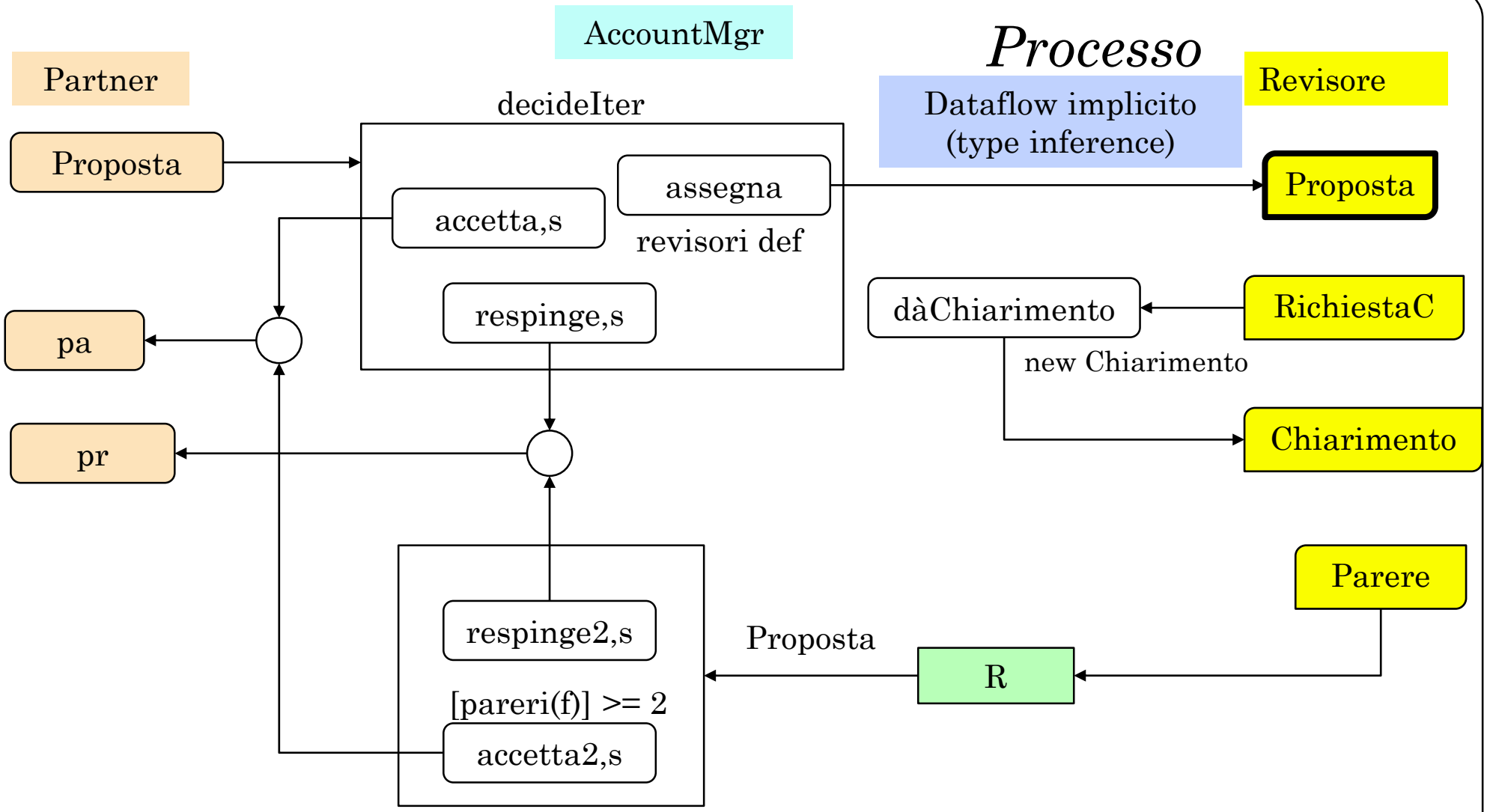
L'output del task non è l'entità di input ma la nuova entità Chiarimento come indicato nel posto di uscita (inglobato nel link).



Processo

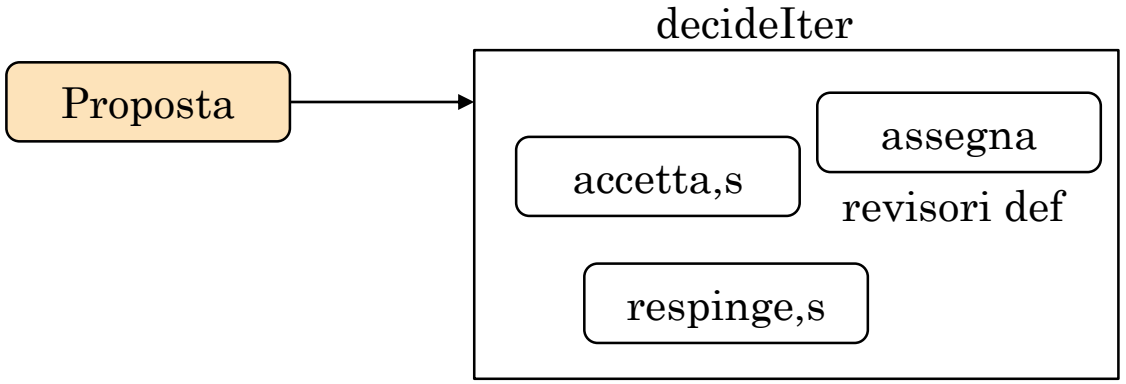


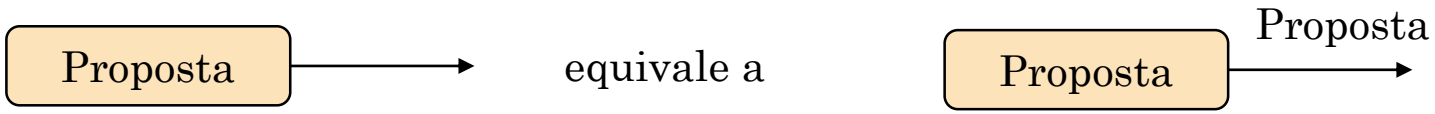
Il modello appare sconnesso, ma in realtà le connessioni sono date dal modello collaborativo.



Il dataflow si può determinare sulla base degli estremi dei link. Si perde però l'informazione sugli stadi (stage) dei lifecycle.

Dataflow implicito

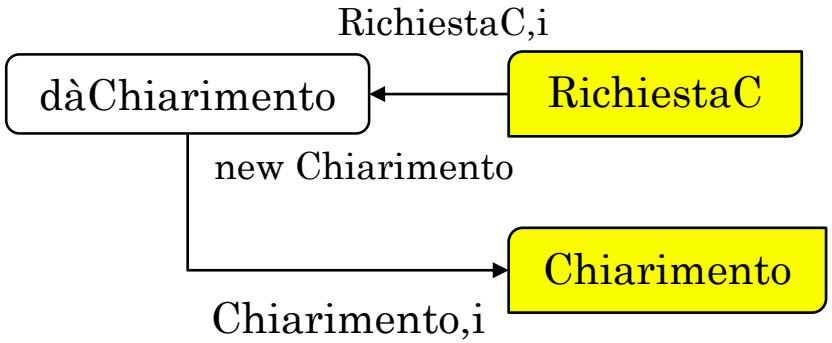


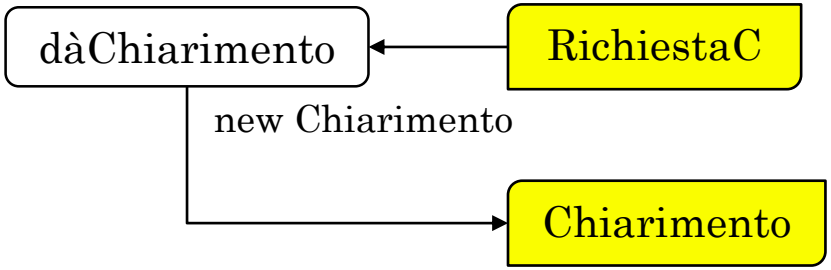


Type inference
Il task d'interazione Proposta riguarda un'interazione generativa e quindi l'output è una proposta.

I tre task sono passanti quindi il tipo di output è lo stesso dell'input (Proposta).

Dataflow implicito



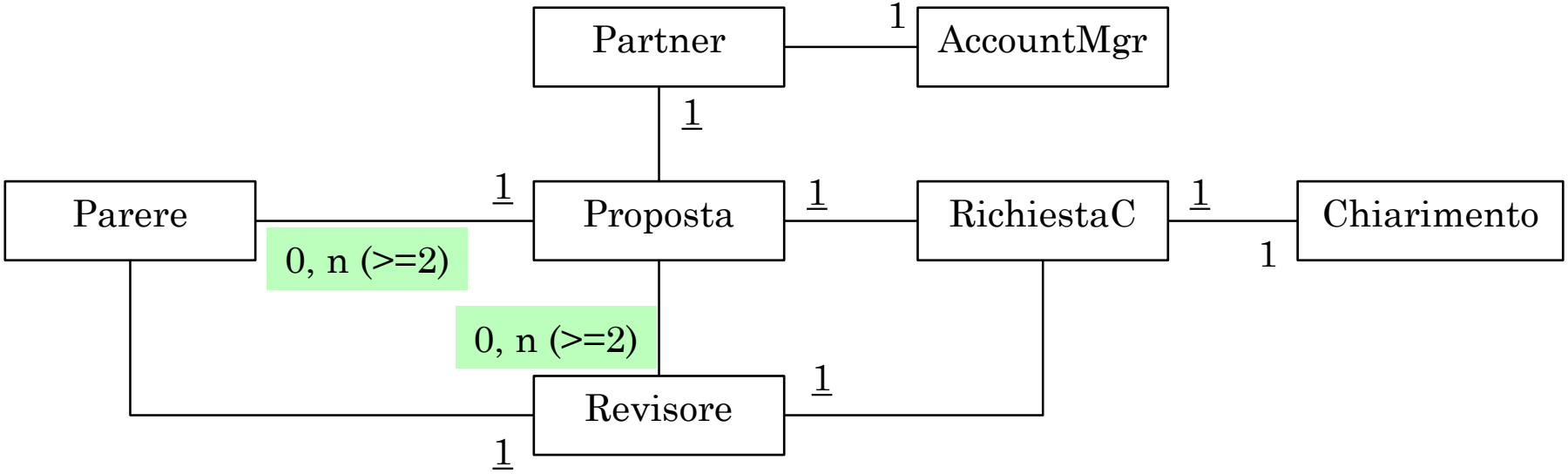


oppure con mapping implicito

Variante 1

Il numero dei revisori non è prestabilito, ma è scelto dall'account manager in base alla proposta; i revisori devono essere almeno 2.

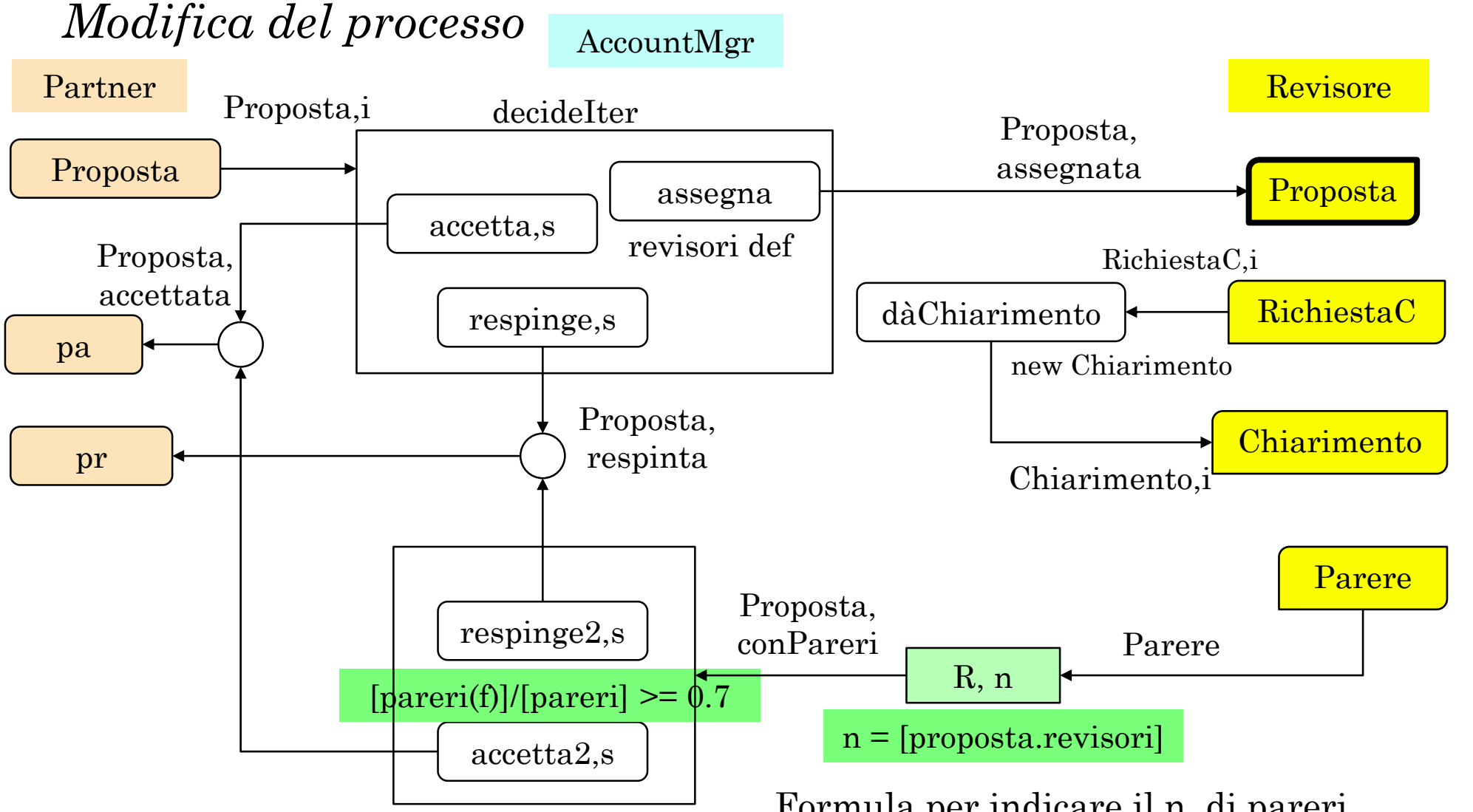
Una proposta è accettata se il n. di pareri favorevoli è >= del 70%.



Modifica del modello informativo

Parere: stato (f, sf).
Proposta: stato (accettata, respinta).

Modifica del processo



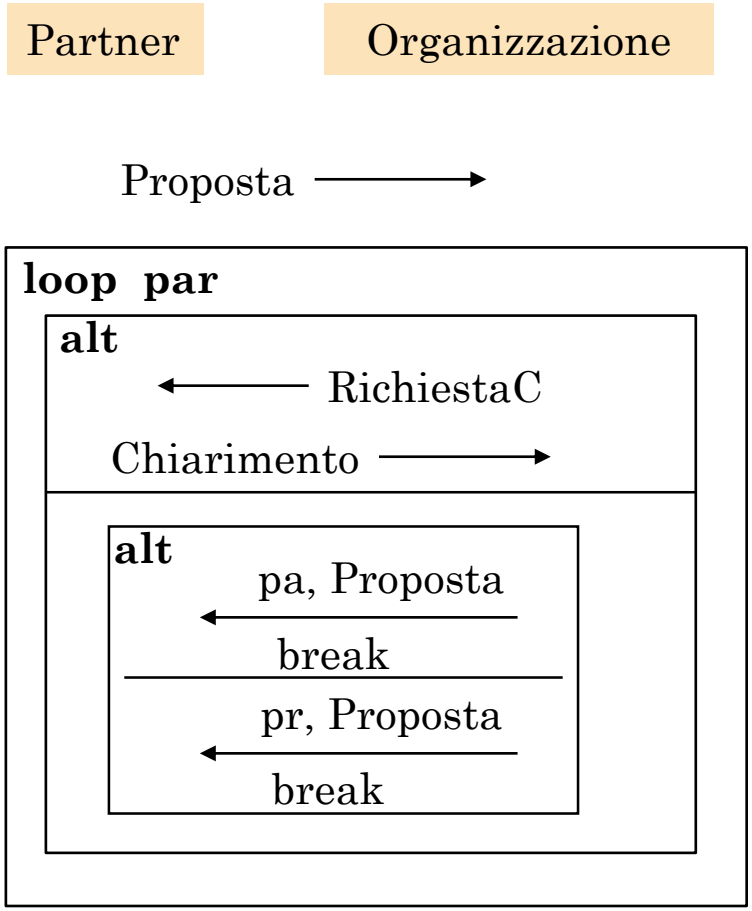
Formula per indicare il n. di pareri attesi; n è il numero dei revisori della proposta.

Variante 2

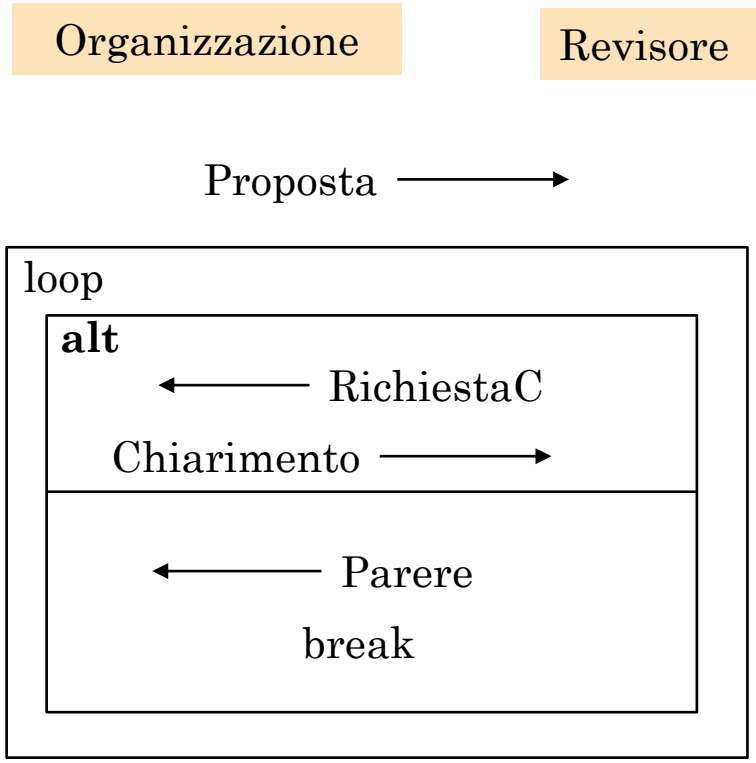
Le richieste di chiarimento dei revisori non sono evase direttamente dal processo ma sono rinviate al partner della proposta.

Le richieste di chiarimento arrivano al partner senza un ordine preciso e pertanto vanno gestite in parallelo; serve quindi un loop con la variante par.

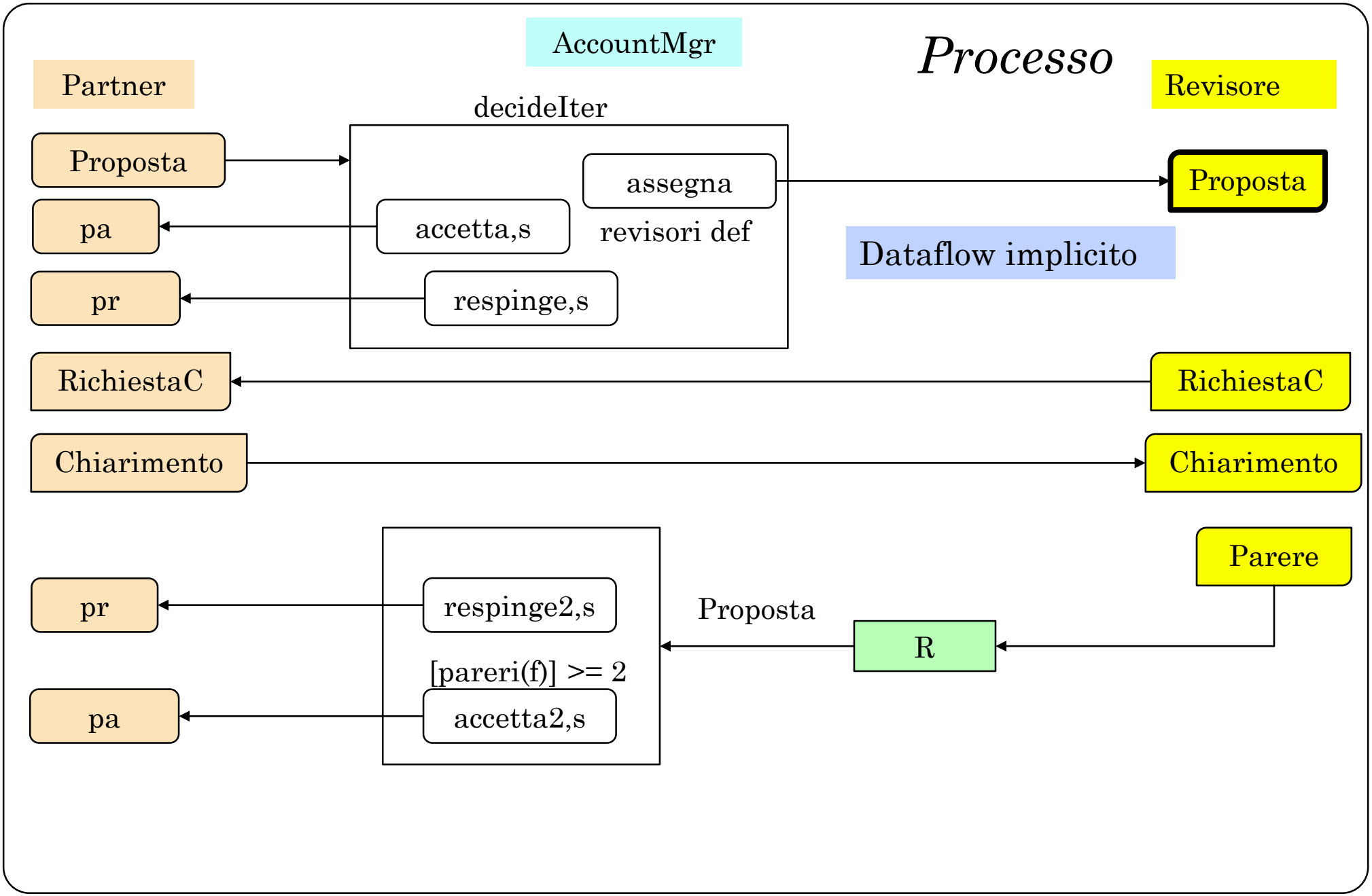
Collaborazioni



pa = proposta accettata
pr = proposta respinta



Attributi: Parere: stato (f, sf).


AccountMgr
Processo
Partner
Revisore
decideIter
Proposta
assegna
Proposta
pa
accetta,s
revisori def
Dataflow implicito
pr
respinge,s
RichiestaC
RichiestaC
Chiarimento
Chiarimento
Parere
pr
respinge2,s
Proposta
R
[pareri(f)] >= 2
pa
accetta2,s

Propagazione di un'entità da una collaborazione ad un'altra

Mapping implicito

Interazione partecipativa

Gestione proposte d'acquisto

Il sistema B2B offre proposte d'acquisto a clienti e fornitori.

Nel sistema informativo sono presenti clienti, fornitori, tipi (di prodotti), gestori (persone di staff). I fornitori sono collegati ai tipi forniti.

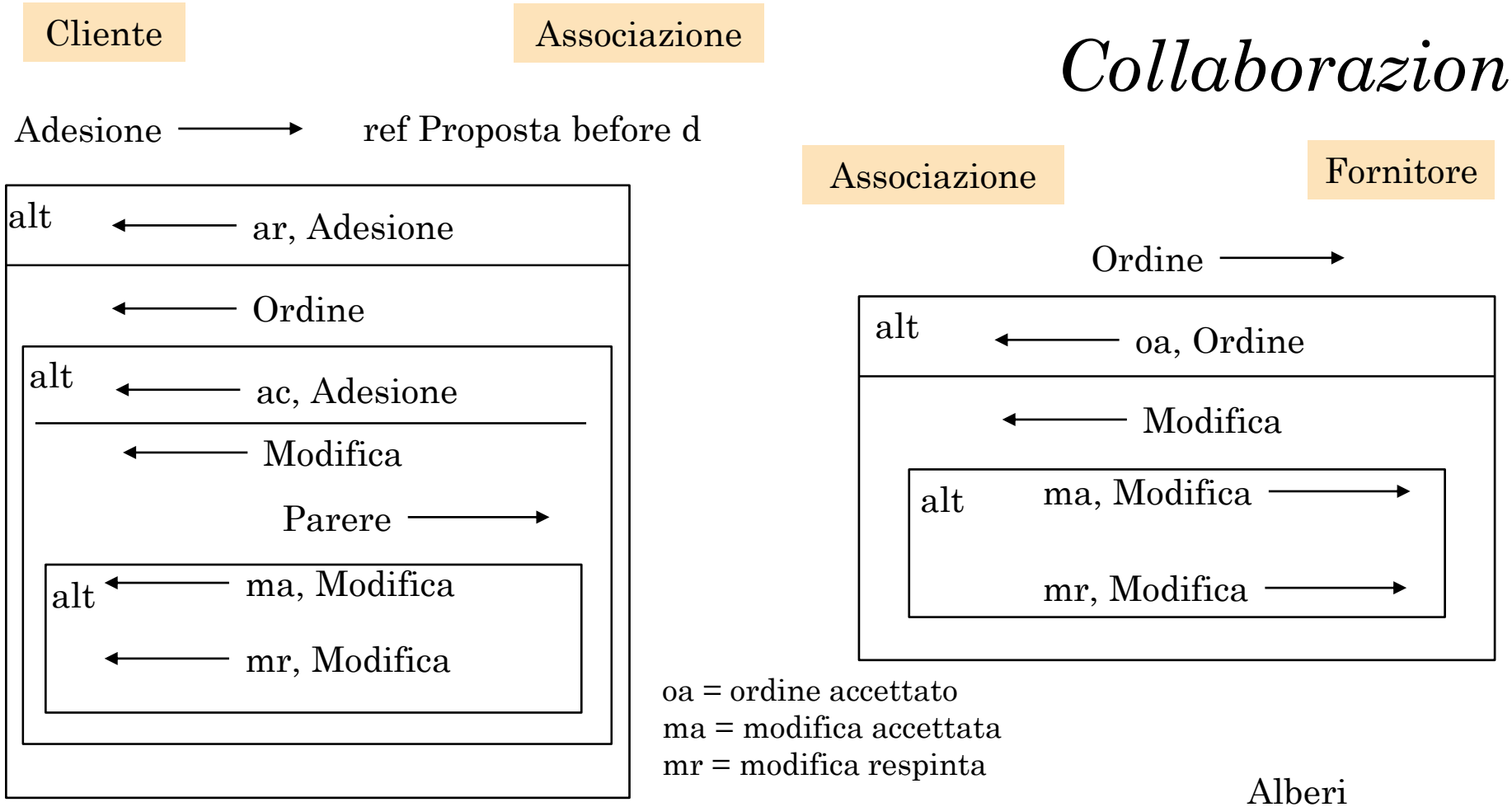
I gestori possono generare proposte (d'acquisto): una proposta è relativa ad un gestore, ad un tipo di prodotto e ha una scadenza d.

I clienti interessati ad una proposta possono inviare un'adesione entro la scadenza. Se le adesioni sono almeno 5, il gestore conferma le adesioni e genera un ordine collegato alla proposta e ad un fornitore idoneo per il tipo della proposta (1). Il processo manda l'ordine al fornitore e ai clienti. Se le adesioni sono meno di 5, il gestore respinge le adesioni.

Il fornitore può accettare l'ordine o proporre una modifica all'ordine. Nel primo caso il processo informa i clienti che le loro adesioni sono confermate. Nel secondo caso invia la modifica ai clienti per ottenere il loro parere che può essere favorevole o sfavorevole. Se tutti i pareri sono favorevoli, il processo accetta la modifica e informa sia il fornitore sia i clienti che la modifica è stata accettata; altrimenti respinge la modifica e informa sia il fornitore sia i clienti che la modifica è stata respinta.

(1) Si esprima il vincolo con un invariante.

Collaborazioni



oa = ordine accettato
ma = modifica accettata
mr = modifica respinta

Attributi: Parere: stato (f, sf).

ar = adesione respinta
ac = adesione confermata

ma = modifica accettata
mr = modifica respinta

Alberi

→ Adesione - Proposta
← Ordine
← Modifica
→ Parere

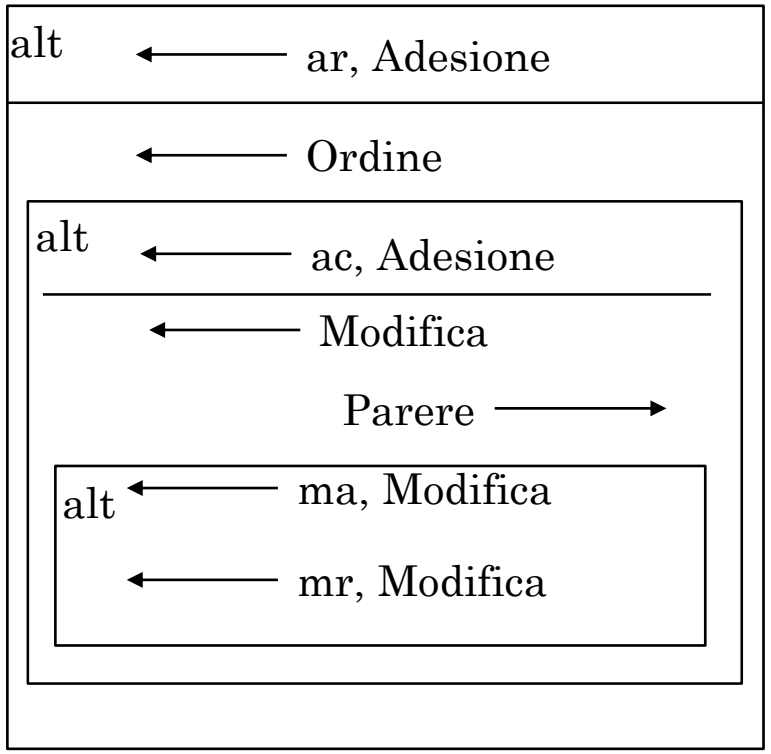
→Ordine
←Modifica

Collaborazioni

Cliente

Associazione

Adesione ⟶ **ref Proposta before d**



Interazione partecipativa che va collegata ad un'entità del processo entro la scadenza indicata nell'entità. Se la scadenza è già passata, la collaborazione fallisce.

Quando il processo riceve un'adesione la collega alla proposta, tipicamente nello stato i (iniziale).

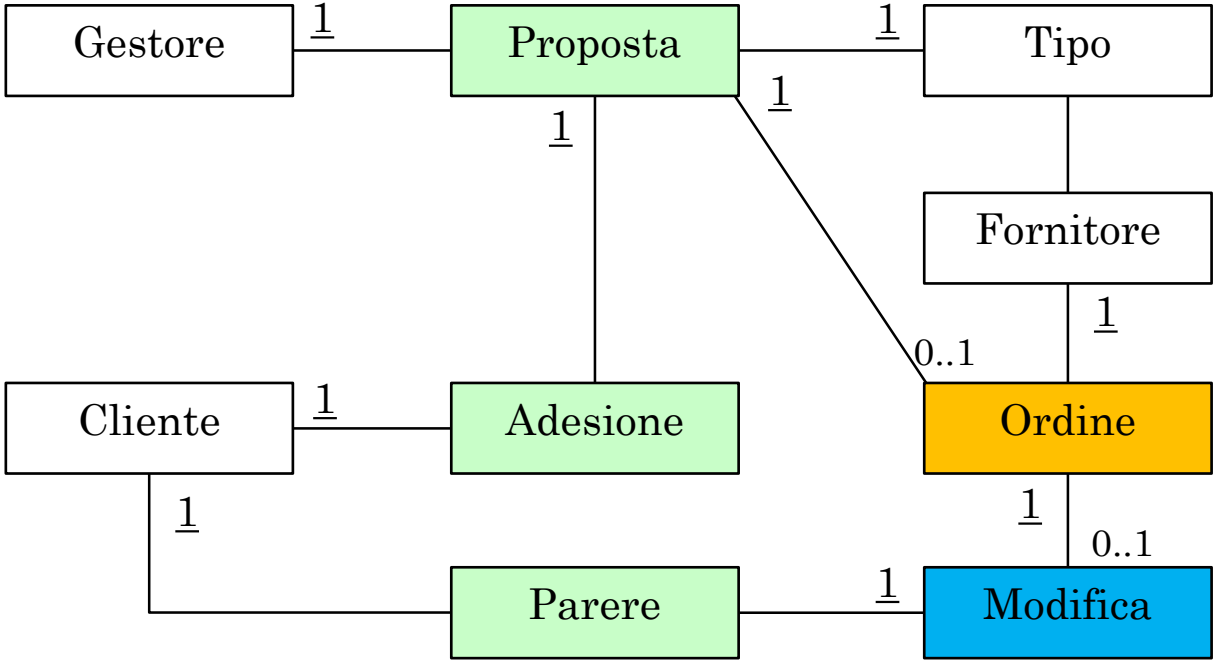
ar = adesione respinta
ac = adesione confermata

ma = modifica accettata
mr = modifica respinta

Alberi

→ Adesione - Proposta
← Ordine
← Modifica
→ Parere

→Ordine
←Modifica



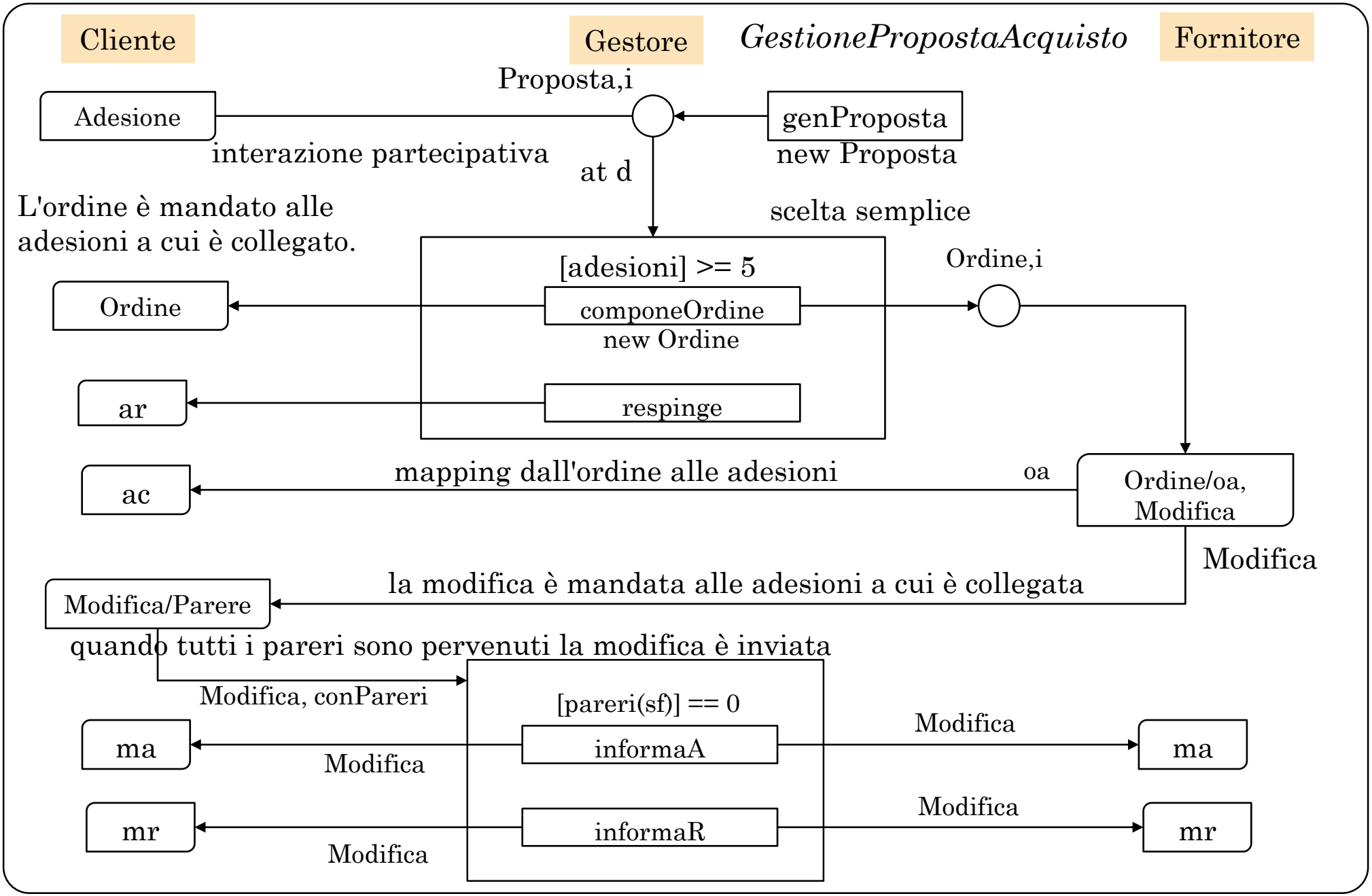
Modello informativo

Rettangoli bianchi per i tipi di contesto.

Attributi: Parere: stato (f, sf).

Invariante: ordine.proposta.tipo in ordine.fornitore.tipi.

Collegamenti
L'ordine è collegato alle adesioni: Ordine – Proposta – Adesione.
La modifica è collegata alle adesioni: Modifica – Ordine – Proposta – Adesione.


Cliente
Gestore
GestionePropostaAcquisto
Fornitore
Proposta,i
Adesione
genProposta
interazione partecipativa
new Proposta
at d
L'ordine è mandato alle adesioni a cui è collegato.
scelta semplice
[adesioni] >= 5
Ordine,i
Ordine
componeOrdine
new Ordine
ar
respinge
ac
mapping dall'ordine alle adesioni
oa
Ordine/oa, Modifica
Modifica
Modifica/Parere
la modifica è mandata alle adesioni a cui è collegata
quando tutti i pareri sono pervenuti la modifica è inviata
Modifica, conPareri
[pareri(sf)] == 0
ma
Modifica
informaA
Modifica
ma
mr
Modifica
informaR
Modifica
mr

Interazioni con payload strutturato

Task aggregativo

Payload con attributi

Gestione ordini

Un distributore riceve ordini di libri dai clienti. Un ordine cliente si compone di linee ciascuna delle quali si riferisce ad un libro e contiene il numero di copie desiderato.

Nel sistema informativo sono registrati clienti, editori e libri. Ciascun libro è associato ad un editore. Ogni cliente è gestito da un account manager (AccountMgr). Un AccountMgr può respingere o accettare un ordine cliente. Se lo respinge, il processo informa il cliente, se l'accetta le linee sono rese *disponibili*.

Un AccountMgr può aggregare varie linee *disponibili* per produrre un ordine editore: tutti i libri indicati dalle linee devono essere trattati dall'editore al quale è diretto l'ordine (1). Il processo invia l'ordine all'editore che risponde con il messaggio *oe* (ordine eseguito): allora le linee dell'ordine risultano *servite*. Un riduttore dichiara *servito* un ordine cliente quando tutte le sue linee si trovano tra quelle *servite*. L' AccountMgr completa un ordine cliente servito e il processo lo invia al cliente con l'interazione *oe*.

(1) Si esprima il vincolo con un invariante.

Gestione ordini

In generale,

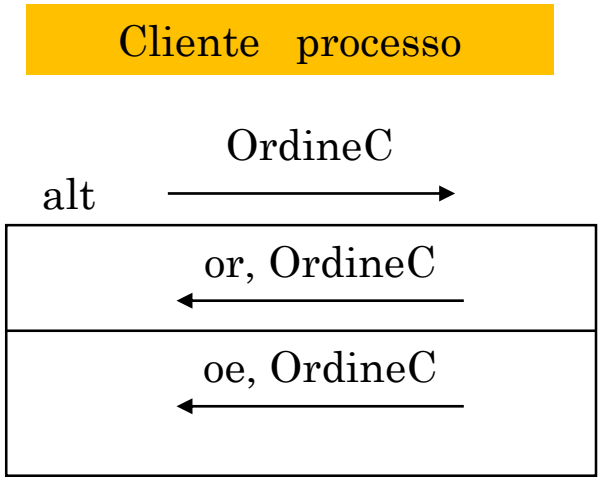
le linee di una collaborazione Cliente entrano in n collaborazioni Editore

e

le linee incluse in una collaborazione Editore provengono da n collaborazioni Cliente.

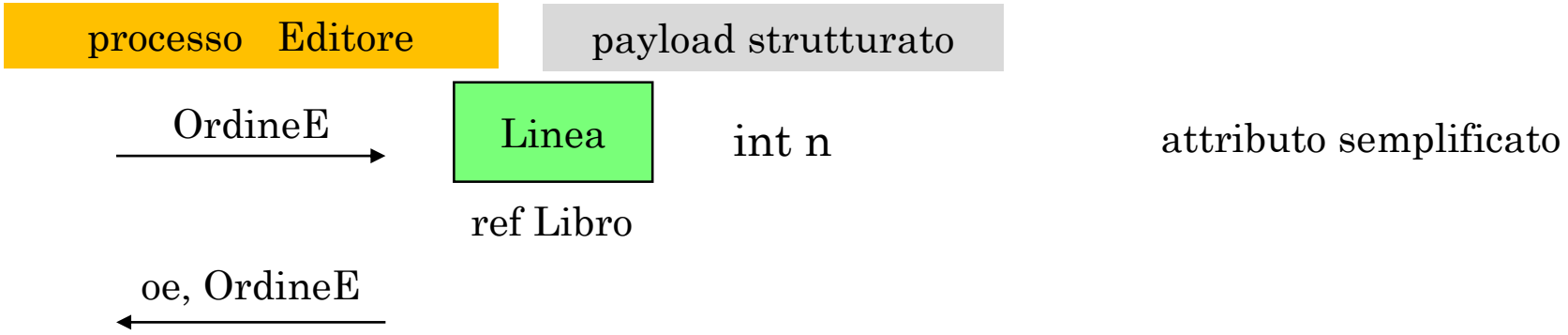
Il rapporto tra i due tipi di collaborazione è m,n.

Collaborazioni



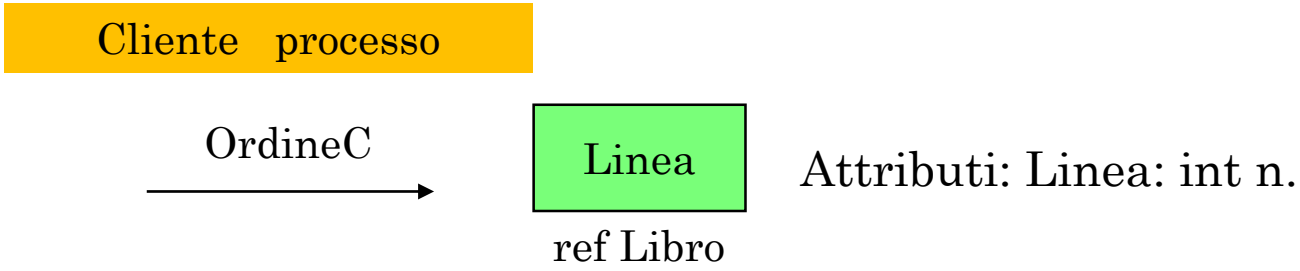
or = ordine respinto
oe = ordine eseguito

Albero
OrdineC - Linea



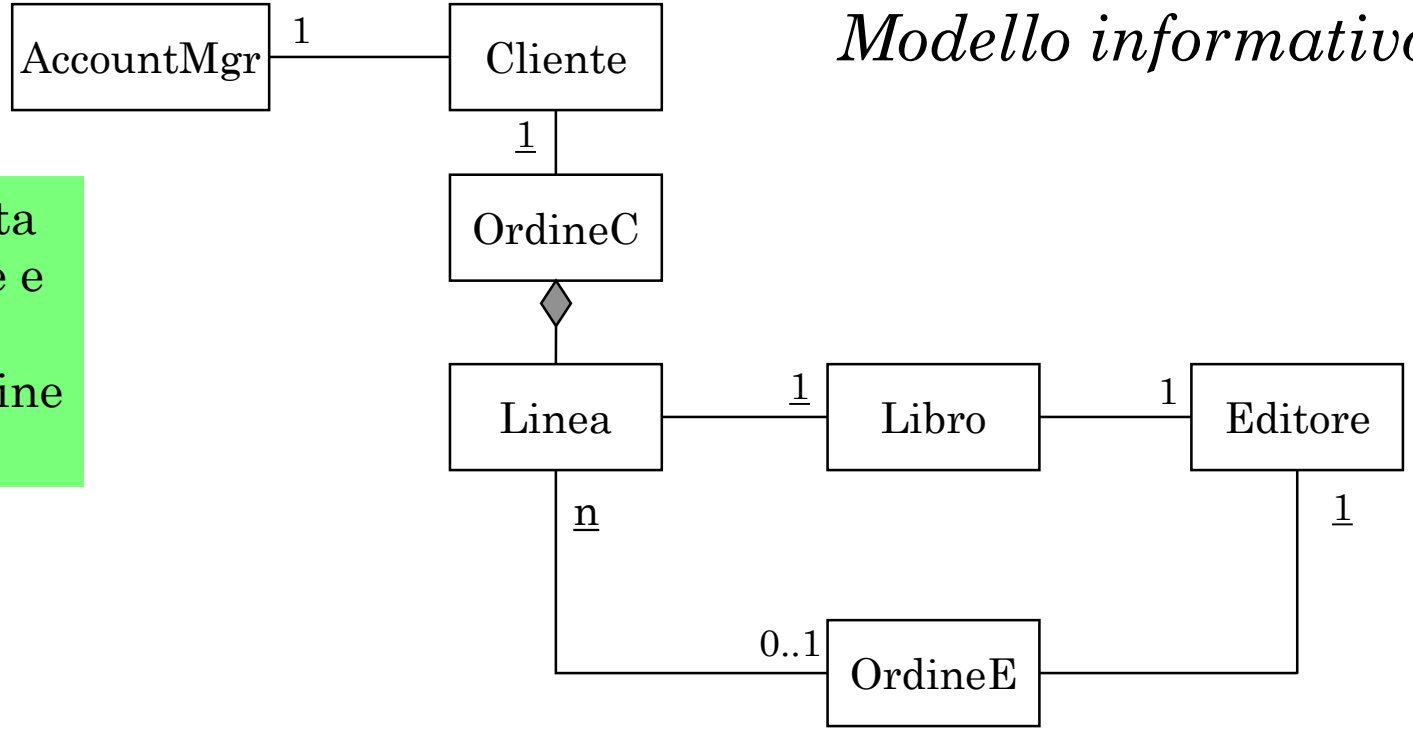
Albero
OrdineE - Linea

Payload strutturato



L'interazione trasmette un ordineC con le sue linee; con ogni linea trasmette anche il riferimento ad un libro.

Modello informativo



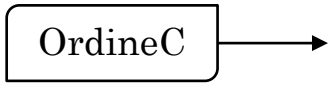
Una linea è associata ad un ordine cliente e può anche essere associata ad un ordine editore.

Linea: int n.

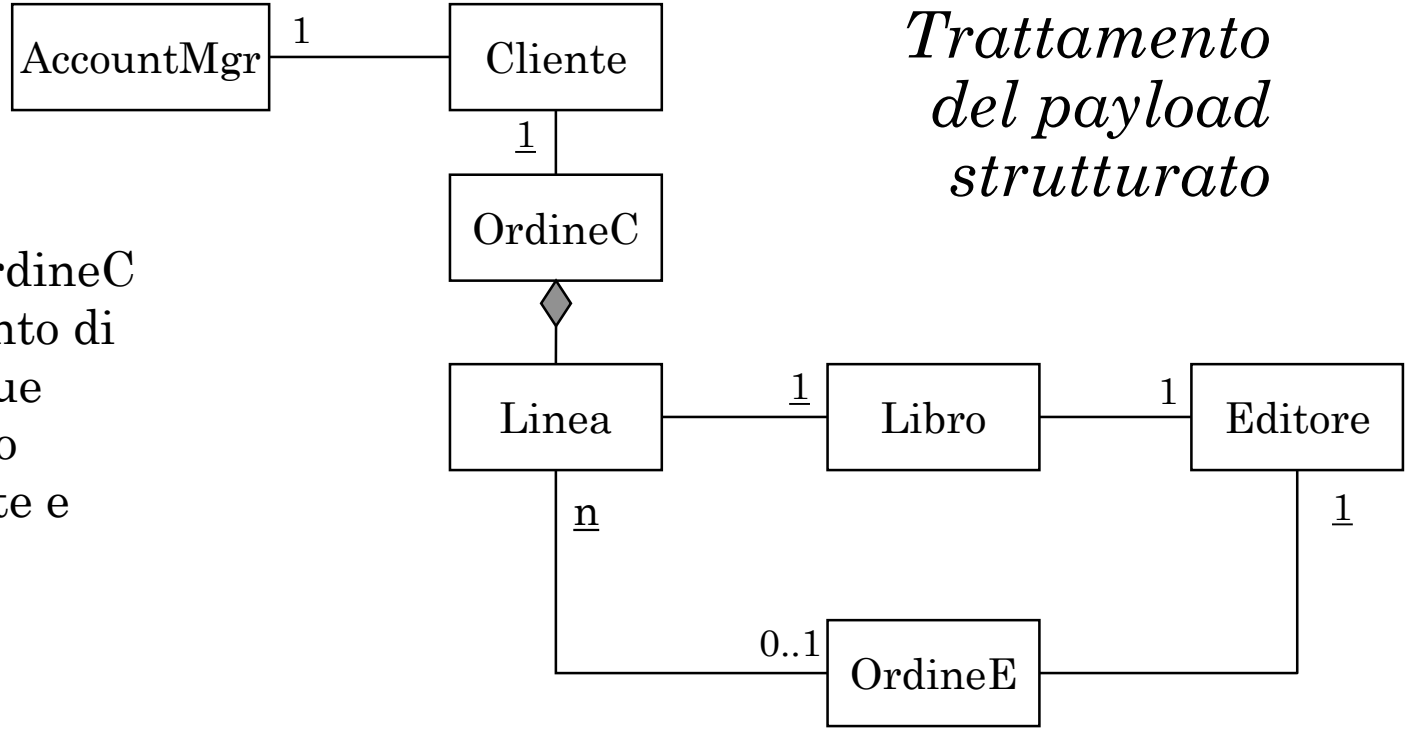
Invariante:
ordineE.editore == ordineE.linee.libro.editore.

Nota: una linea non è collegata a nessun ordineE se l'ordineC a cui appartiene è stato respinto.

Trattamento del payload strutturato

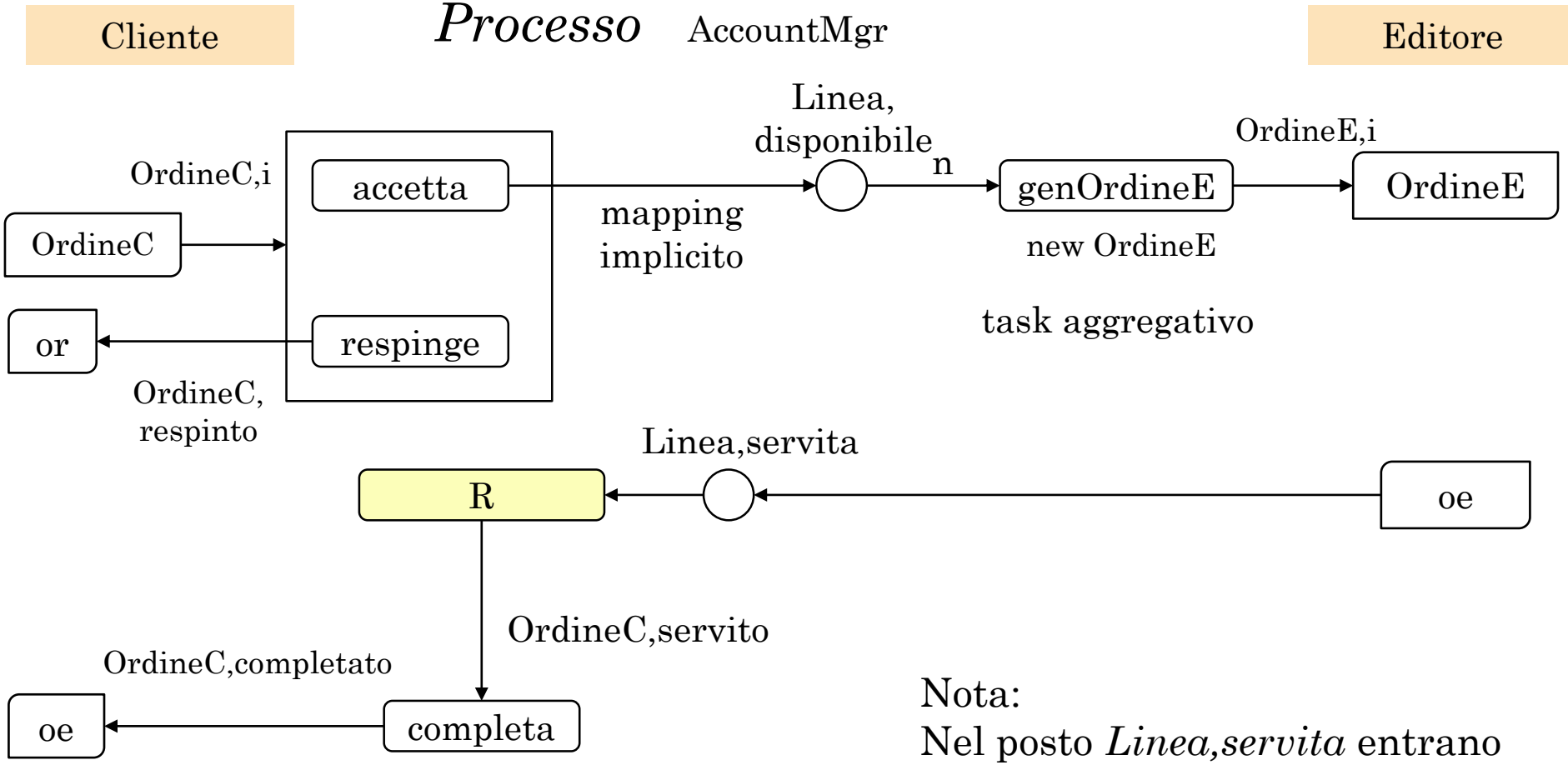


La ricezione di un ordineC comporta l'inserimento di un ordineC e delle sue linee, il collegamento dell'ordineC al cliente e delle linee ai libri.



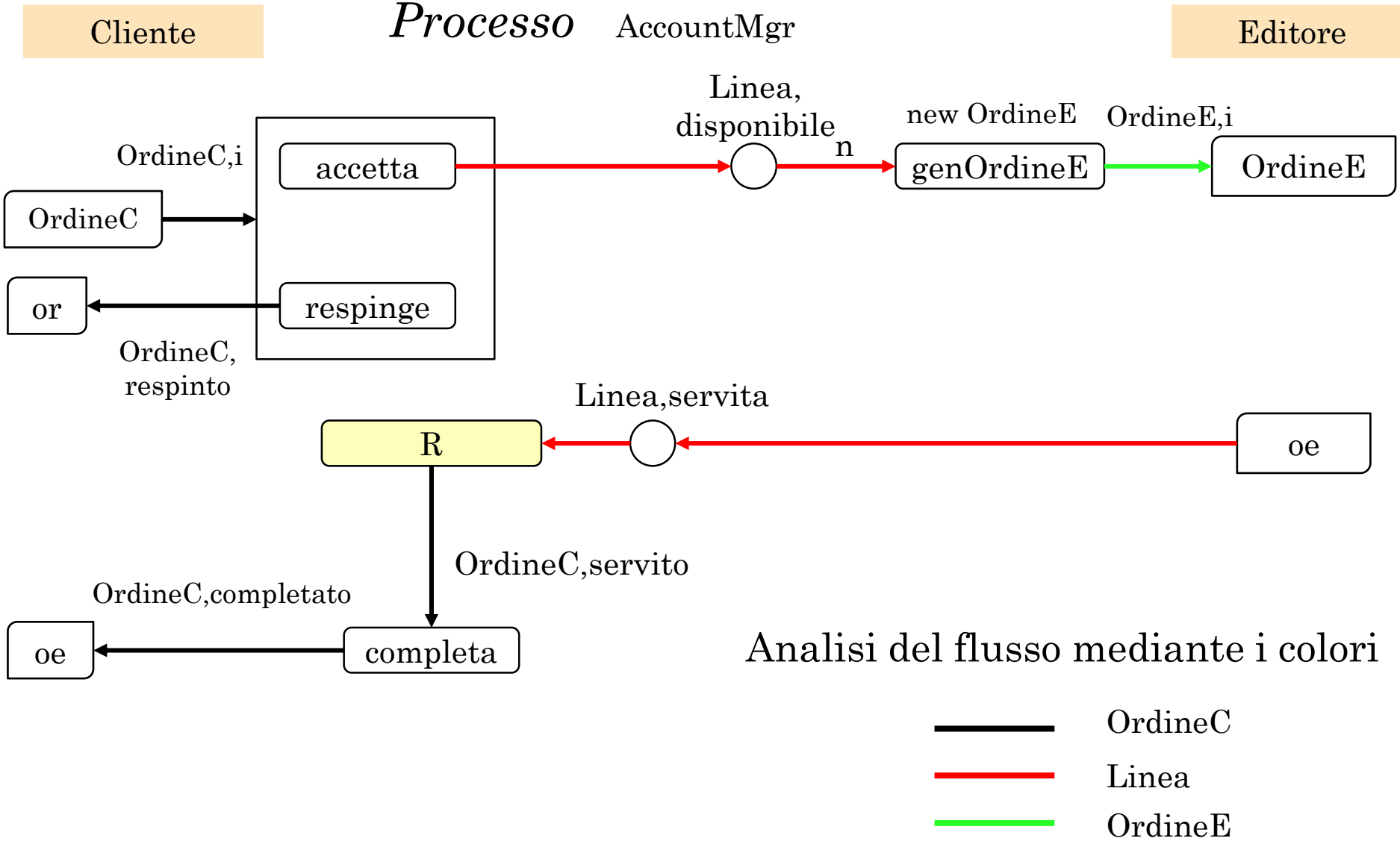
L'invio di un ordineE comporta l'invio dell'ordine e delle sue linee (con i rif. ai libri).

Linea: int n.



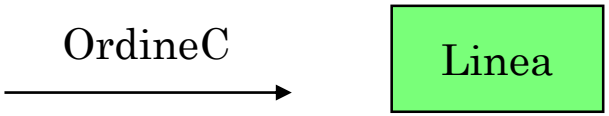
Nota:
Nel posto *Linea,servita* entrano le linee associate all'ordineE.

Quando nel posto *Linea,servita* si trovano tutte le linee relative allo stesso ordineC, il riduttore emette l'ordineC. Le linee sono già collegate ad un ordineC.

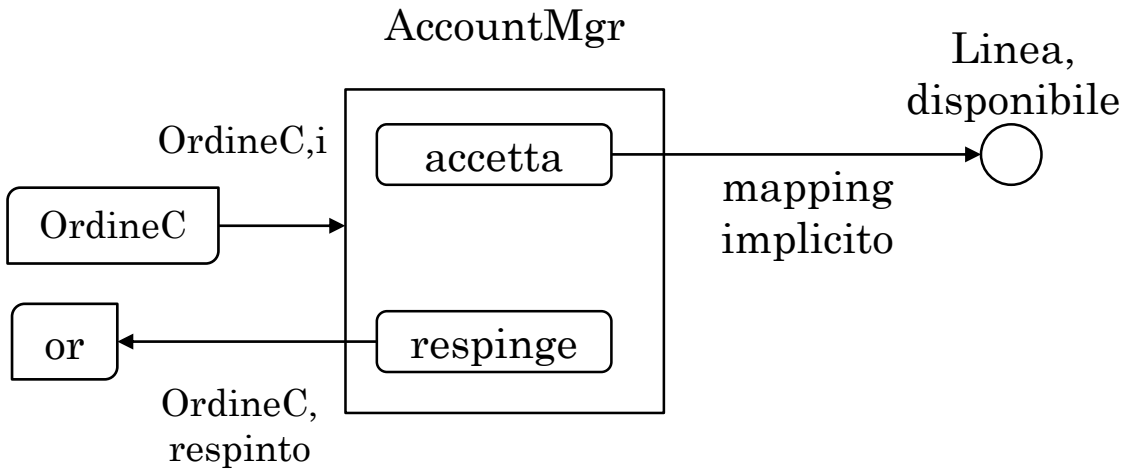

Cliente
Processo
AccountMgr
Editore
Linea,
disponibile
n
new OrdineE
OrdineE,i
OrdineC,i
accetta
genOrdineE
OrdineE
OrdineC
or
respinge
OrdineC,
respinto
Linea,servita
R
oe
OrdineC,servito
OrdineC,completato
oe
completa
Analisi del flusso mediante i colori
OrdineC
Linea
OrdineE

Cliente processo

Variante



L'account manager respinge l'ordineC se il numero medio di copie è < 2.
Esempio
[ordineC.linee.nCopie.average] < 2



respinge: pre: [ordineC.linee.nCopie.average] < 2

Task join/fork

Task di timeout

Brokerage

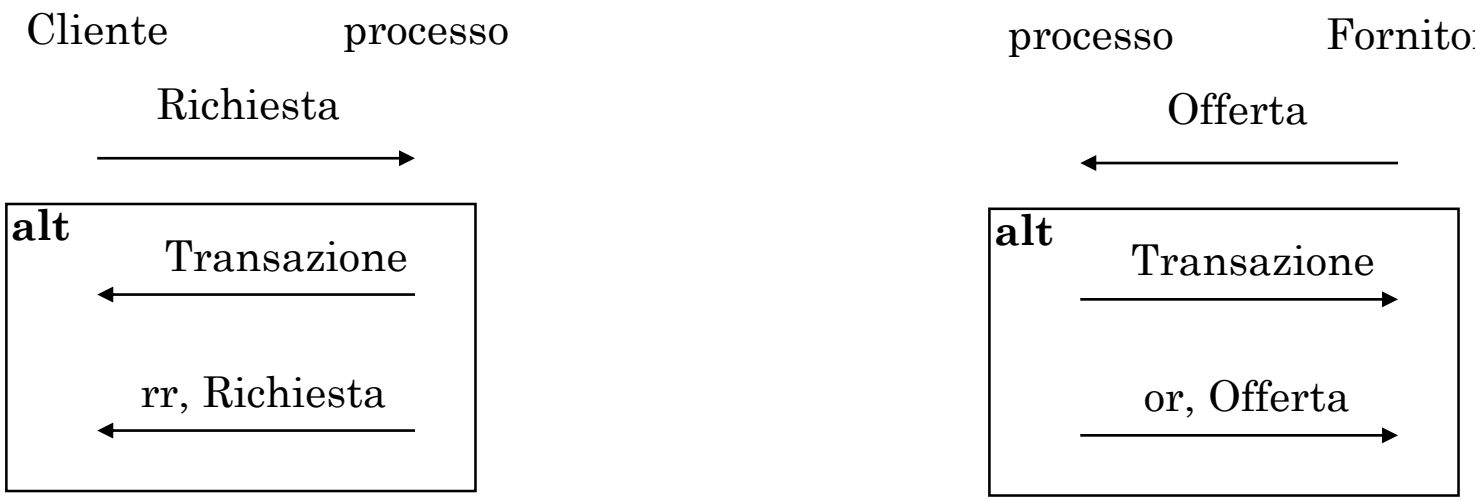
Un'agenzia di intermediazione riceve richieste da clienti e offerte da fornitori. Richieste e offerte si riferiscono ad un tipo di prodotto e hanno una data di scadenza (d).
Nel sistema informativo dell'agenzia sono registrati clienti, fornitori e tipi di prodotti; inoltre ogni tipo di prodotto è assegnato ad un broker (ruolo di staff).

Il broker sceglie una richiesta e un'offerta relative allo stesso tipo (1) per generare una transazione (legata alla richiesta e all'offerta). Il processo la invia ai mittenti della richiesta e dell'offerta.

Se una richiesta (o un'offerta) scade prima che sia inclusa in una transazione, è rinviata al mittente come respinta.

(1) Si esprima il vincolo con un invariante.

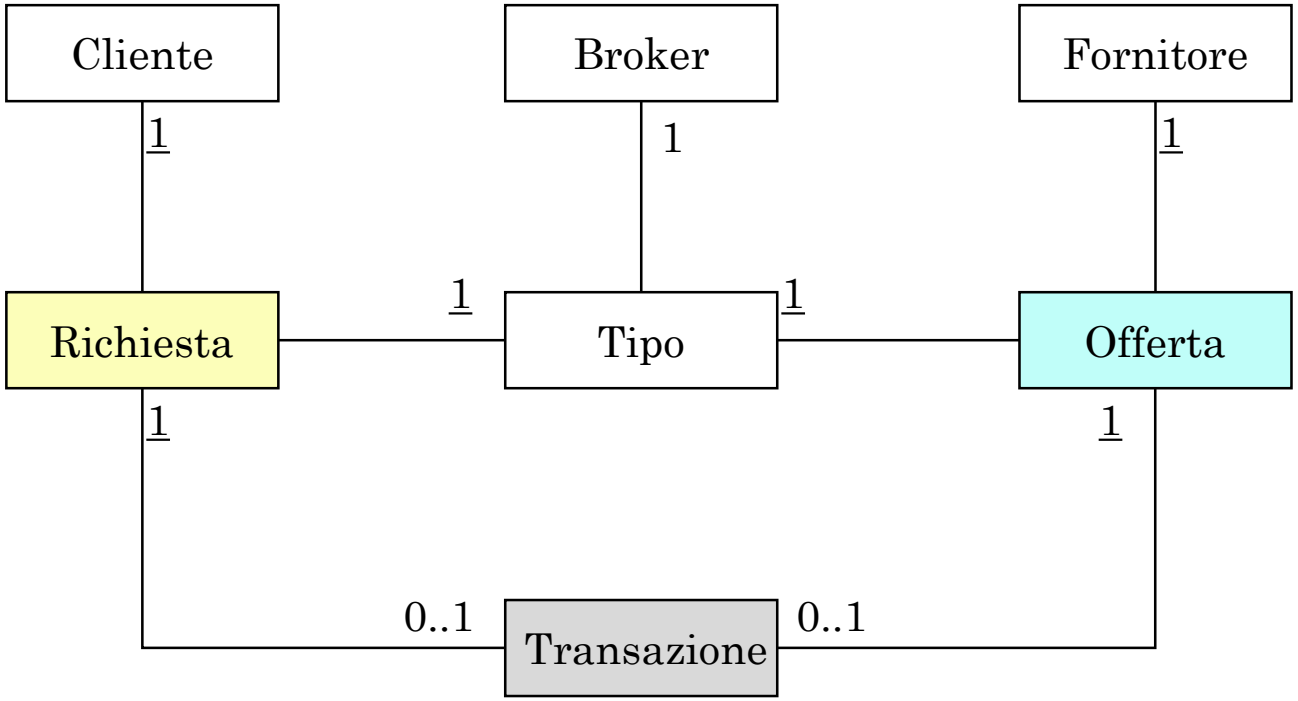
Collaborazioni



rr = richiesta respinta
or = offerta respinta

Attributi
Richiesta: Date d.
Offerta: Date d.

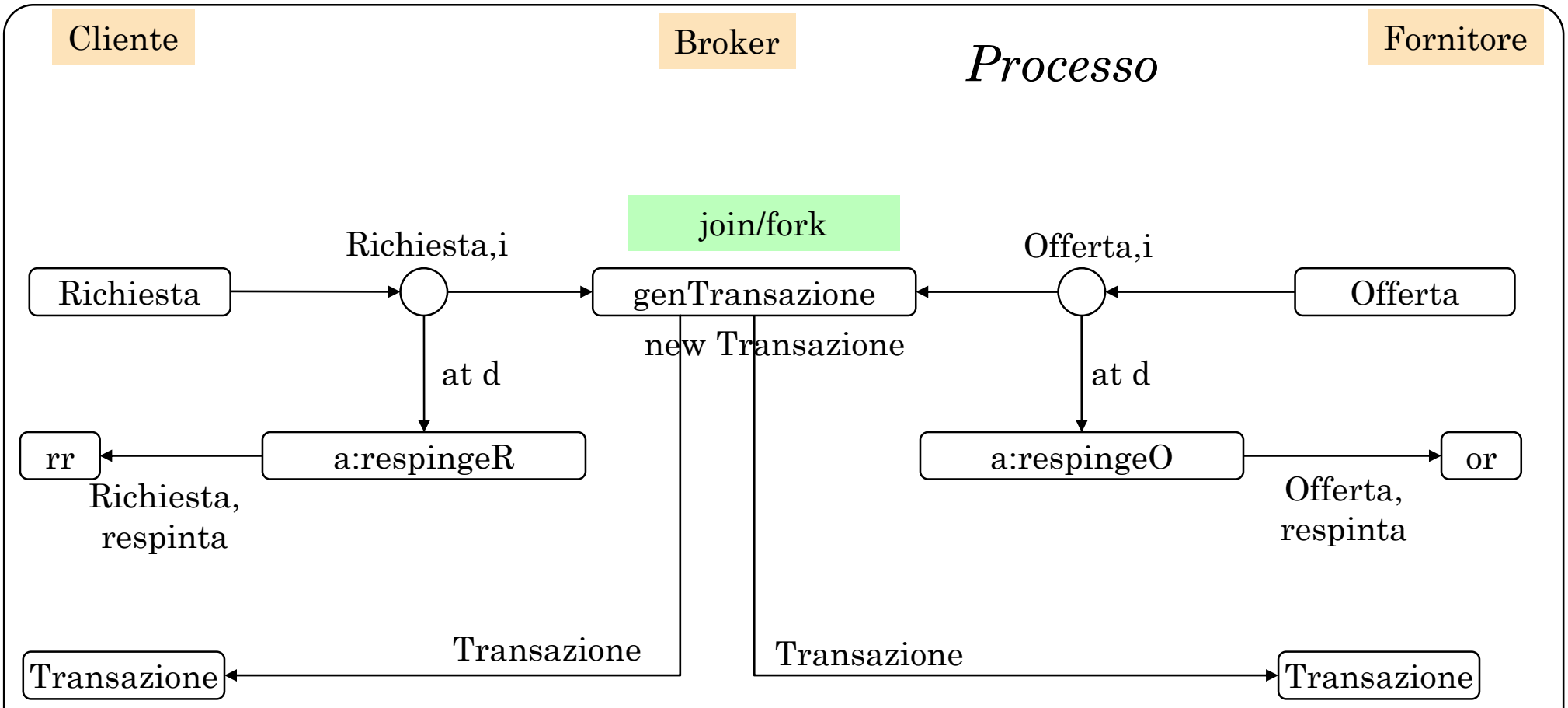
Information model



Rettangoli bianchi per i tipi di contesto.

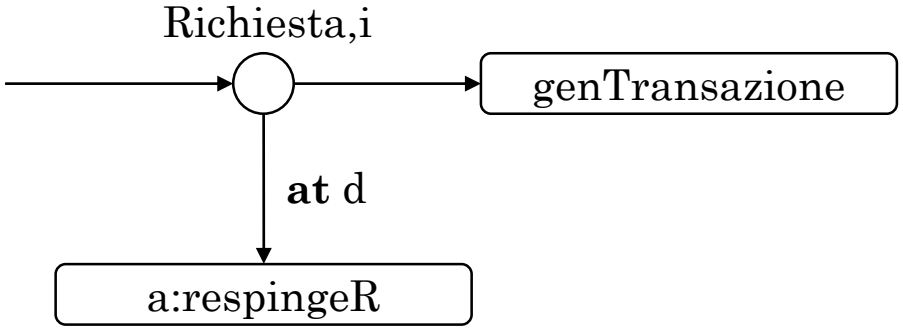
Richiesta: Date d.
Offerta: Date d.

Invariante
transazione.richiesta.tipo == transazione.offerta.tipo



Se una richiesta r al tempo r.d è ancora nel posto *Richiesta, i*, allora è presa dal task respingeR (task di timeout). Analogamente per un'offerta.

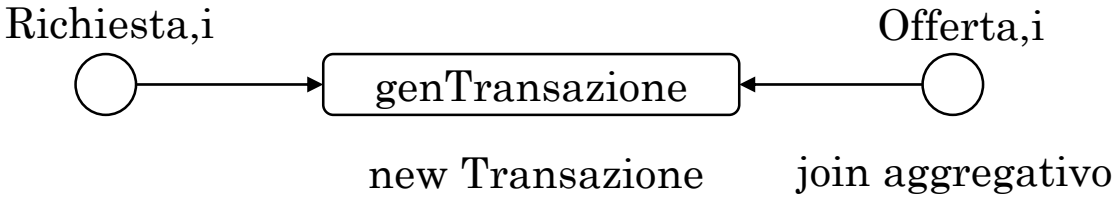
Task di timeout



Un task di timeout ha un unico input link con la clausola **at** che si riferisce ad un attributo (di valore temporale) delle entità di input.

Nell'esempio, se una richiesta r al tempo r.d è ancora nel posto *Richiesta, i*, allora è presa dal task respingeR (task di timeout). Analogamente per un'offerta.

Commento



Gli input del task sono una richiesta e un'offerta scelti tra quelli presenti nei posti di input. La post-condizione new Transazione indica l'effetto: è generata una nuova transazione collegata alla richiesta e all'offerta scelte. Questi legami esprimono il rapporto di causalità e sono conformi alle relazioni obbligatorie nel modello informativo.

L'invariante garantisce la congruenza degli input.

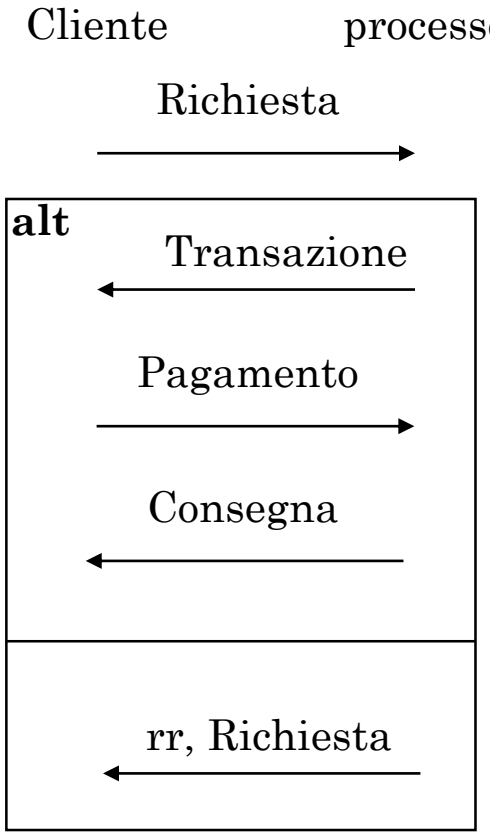
Senza invariante si dovrebbe inserire la pre-condizione:
richiesta.tipo == offerta.tipo
dove richiesta e offerta denotano gli input scelti (si usa il nome della classe con l'iniziale minuscola).

Variante

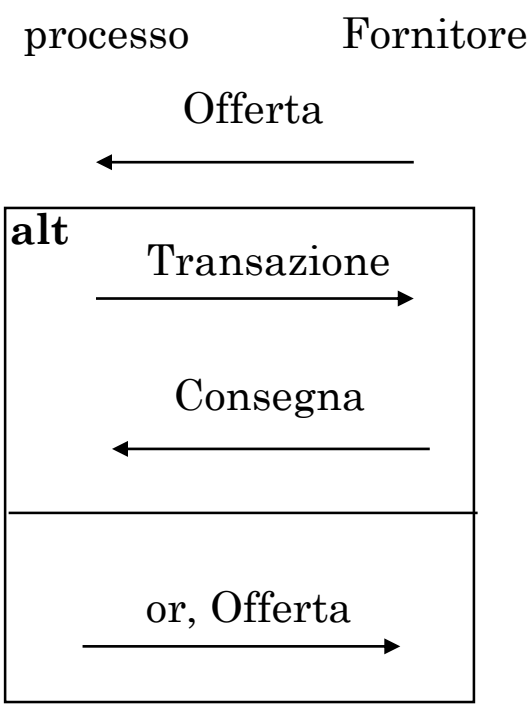
Il broker sceglie una richiesta e un'offerta relative allo stesso tipo per generare una transazione (legata alla richiesta e all'offerta). Il processo la invia ai mittenti della richiesta e dell'offerta. Il cliente risponde con un pagamento e il fornitore con una consegna.

Il processo, quando trova una consegna e un pagamento che si riferiscono alla stessa transazione, manda la consegna al cliente.

Nuove collaborazioni



Albero
Richiesta
Transazione
Pagamento
Consegna



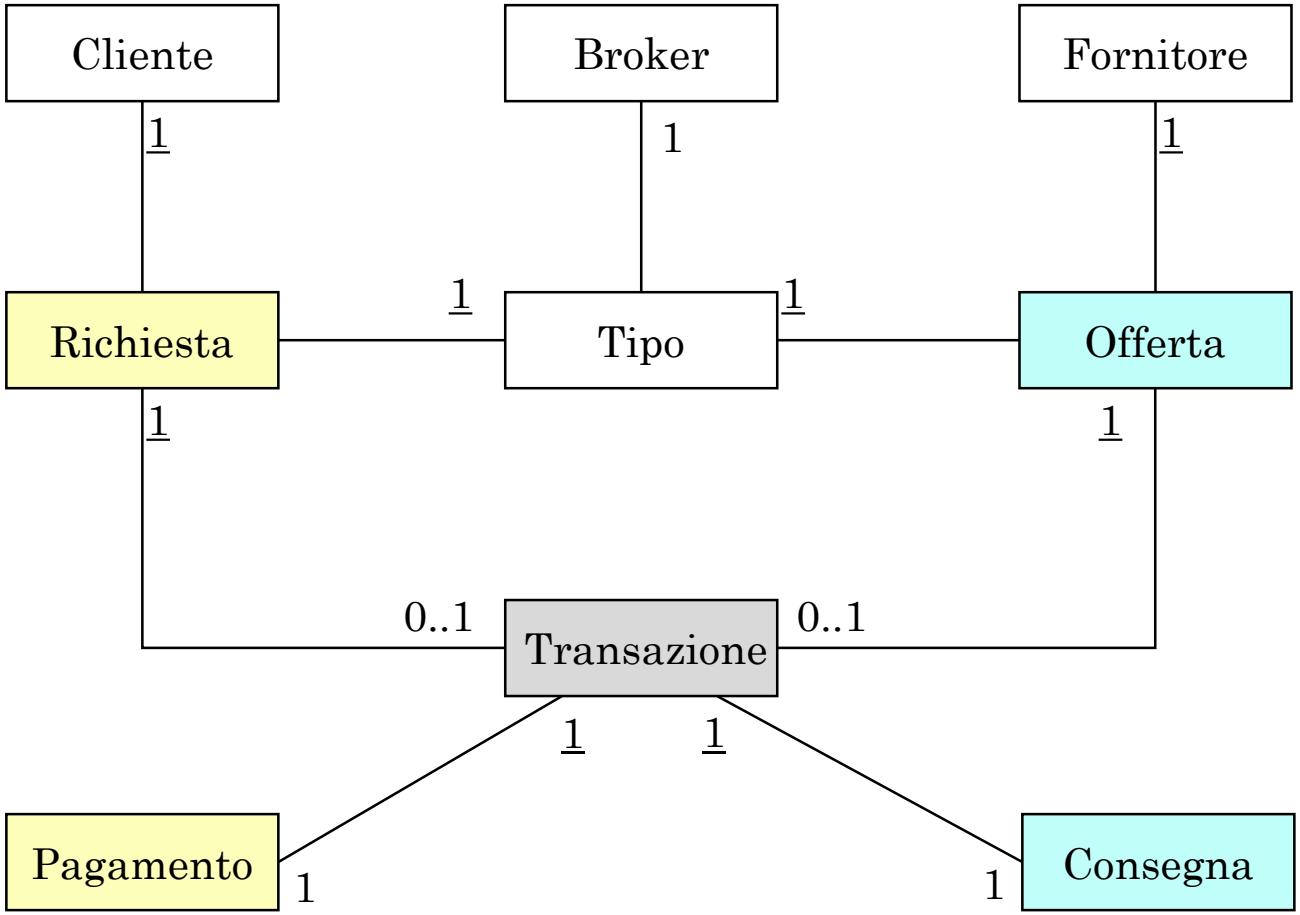
Albero
Offerta
Transazione
Consegna

rr = richiesta respinta

Attributi
Richiesta: Date d.
Offerta: Date d.

or = offerta respinta

Nuovo Information model

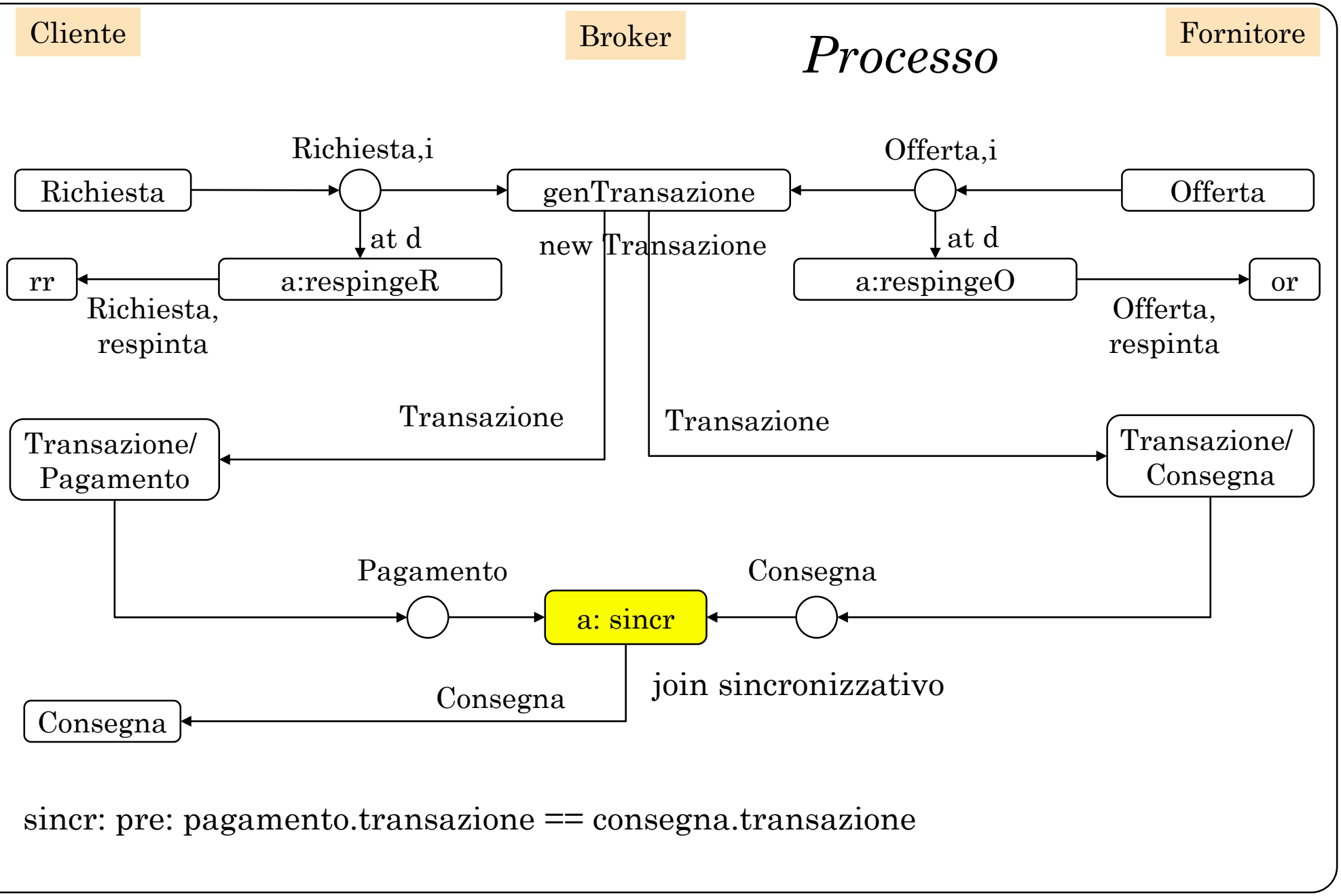


Albero
Richiesta
Transazione
Pagamento
Consegna

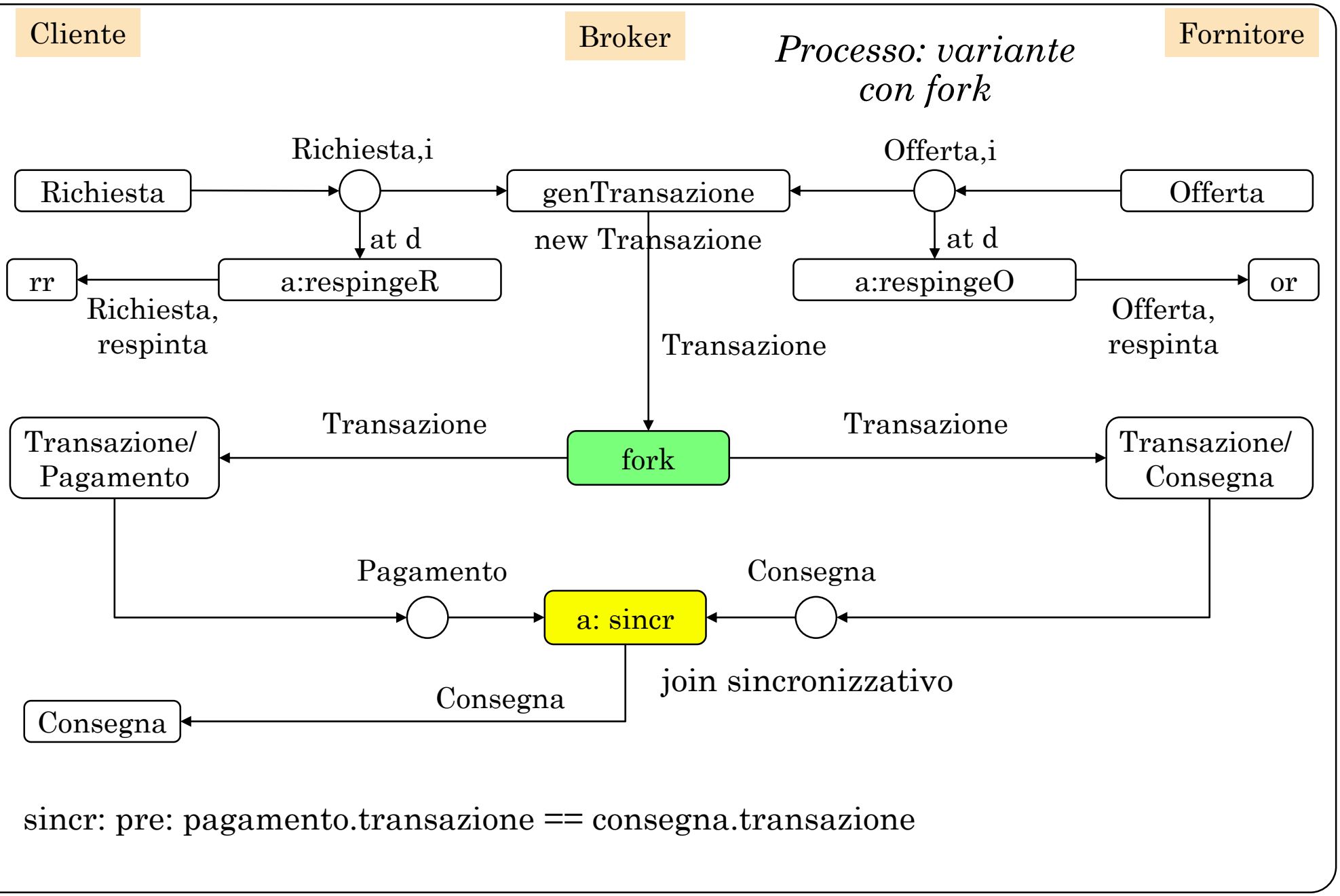
Albero
Offerta
Transazione
Consegna

Richiesta: Date d. Offerta: Date d.
Invariante: transazione.richiesta.tipo == transazione.offerta.tipo.

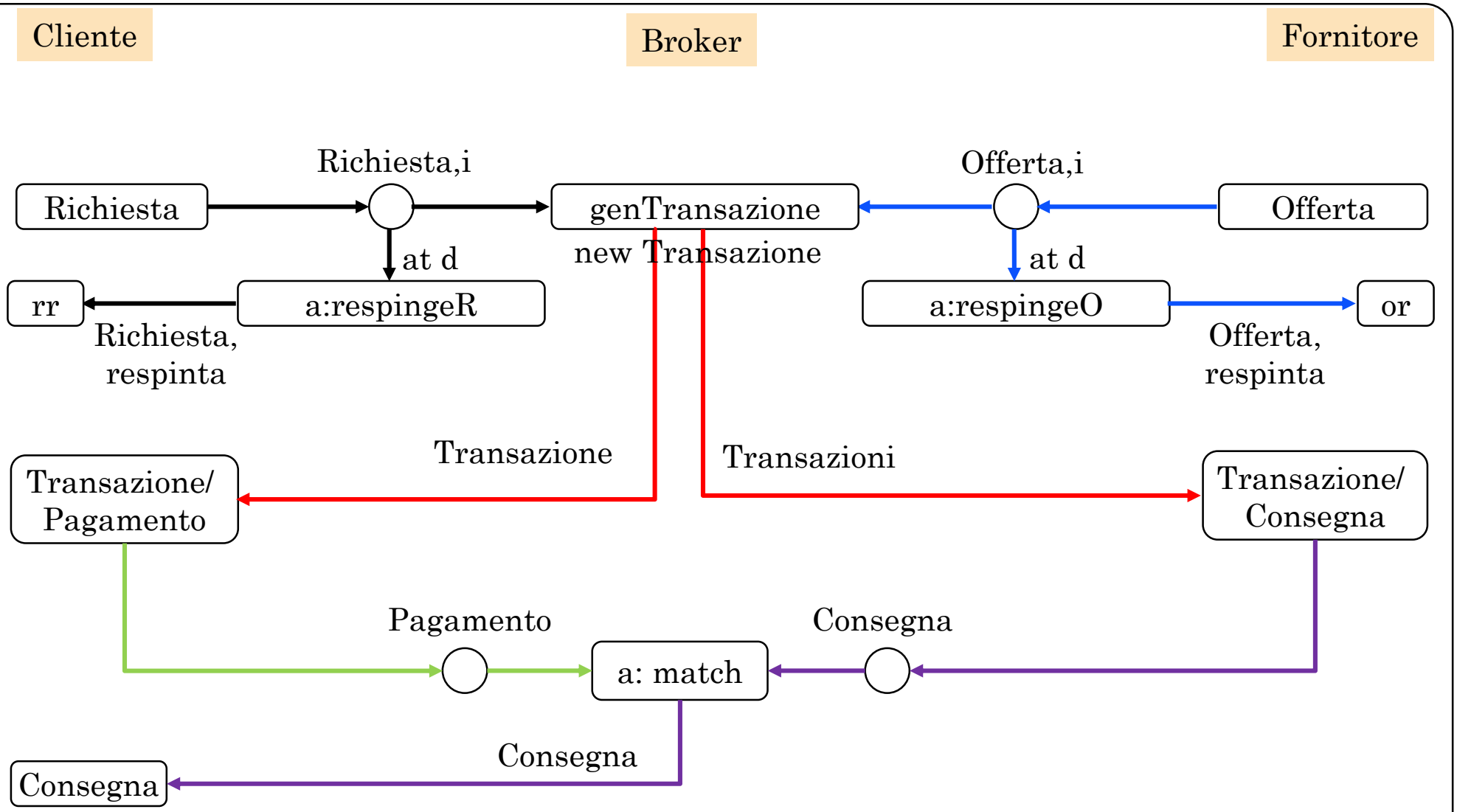
La consegna passa dal fornitore al cliente. La consegna è correlata al pagamento mediante la relazione derivata Consegna – Transazione – Pagamento.



sincr: pre: pagamento.transazione == consegna.transazione



sincr: pre: pagamento.transazione == consegna.transazione



Dataflow con colori

Collaborazione con voto relativo ad un elemento del payload strutturato

Gestione offerte di lavoro

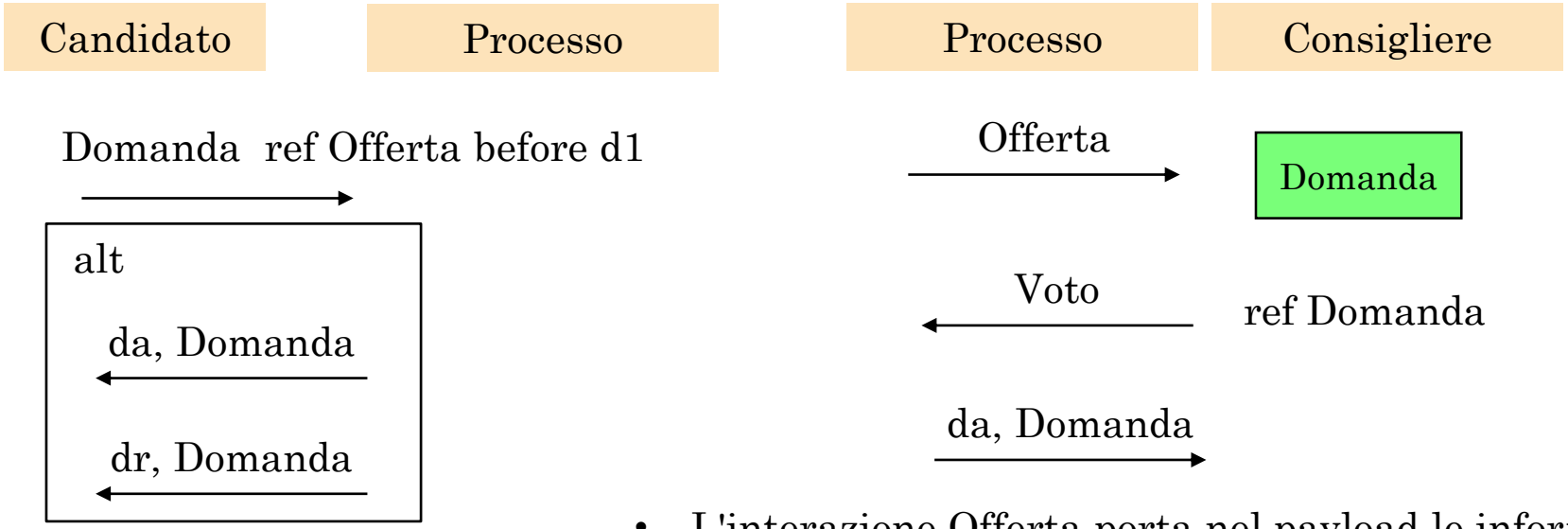
Il processo B2B Gestione Offerte tratta offerte di lavoro proposte da gestori (ruolo di staff).

Nel sistema informativo sono registrati gestori, candidati e consiglieri.

Un'offerta è proposta da un gestore che la collega ad alcuni consiglieri; l'offerta ha una scadenza d1.

I candidati prima di d1 possono presentare una domanda. Al tempo d1, se non ci sono almeno 3 domande, il processo cancella l'offerta e respinge le domande ai mittenti; altrimenti l'offerta con le domande (interazioni con payload strutturato) è inviata ai consiglieri. I consiglieri esprimono un voto per la domanda che ritengono migliore.

Ricevuti tutti i voti, il processo accetta la domanda che ha ottenuto più voti. I consiglieri sono informati della domanda accettata e i candidati dell'esito della loro domanda (accettata o respinta).



da = domanda accettata
dr = domanda respinta

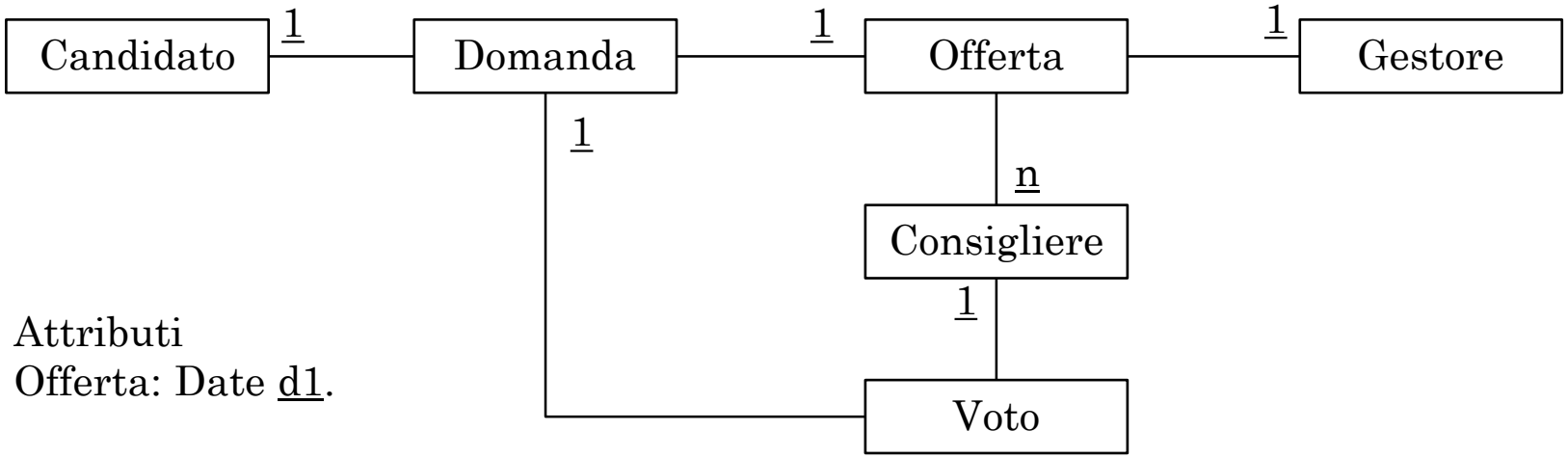
- L'interazione Offerta porta nel payload le informazioni relative all'offerta e alle domande presentate.
- Il voto è correlato sia all'offerta sia alla domanda scelta.
- L'interazione *da* si riferisce a una delle domande trasmesse con l'offerta.

Offerta: Date d1.

Se il candidato non invia la domanda entro il termine Offerta.d1, la collaborazione termina.

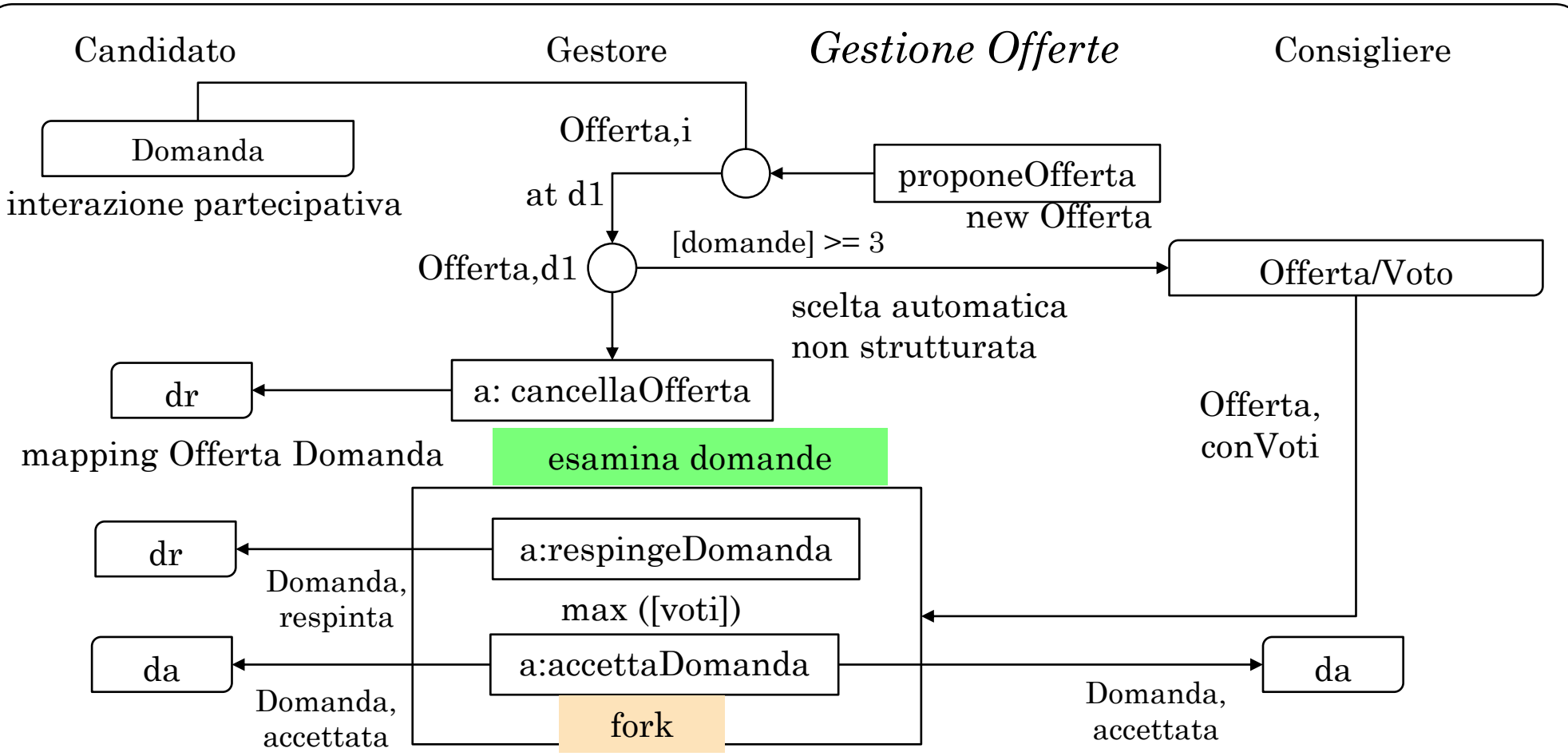
Albero
Offerta – Domanda
Voto ref Domanda

Modello informativo del processo

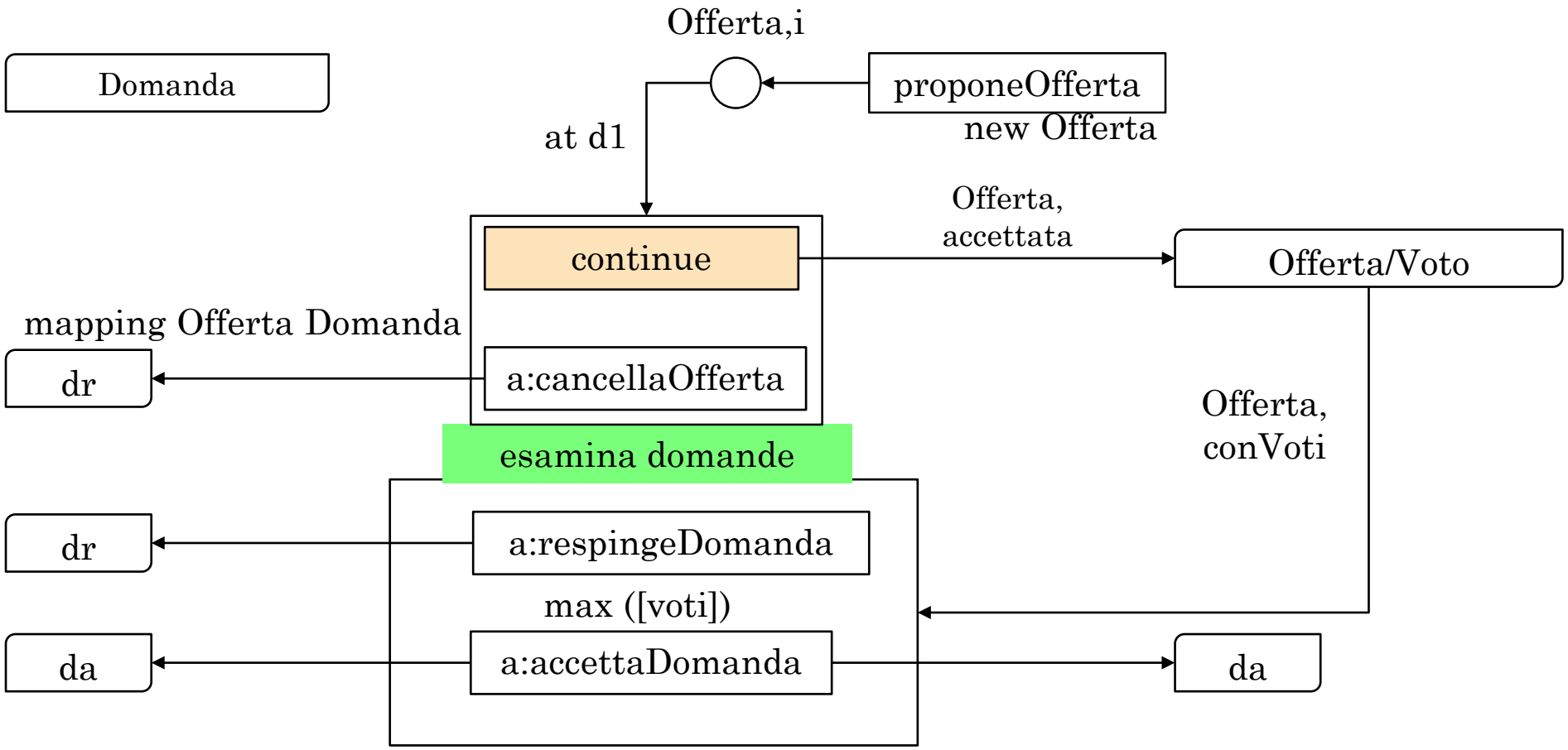


Attributi
Offerta: Date d1.

La relazione Voto – Offerta è derivata:
Voto – Domanda – Offerta.
La relazione diretta Voto – Offerta è ridondante.



La precondizione max([voti]) sceglie la domanda con il numero di voti maggiore. Tutte le altre domande sono respinte.



Variante con scelta automatica strutturata (evita la diramazione):
continue: [domande] >= 3.
Il task **continue** è un task fittizio, che rappresenta una condizione di routing.

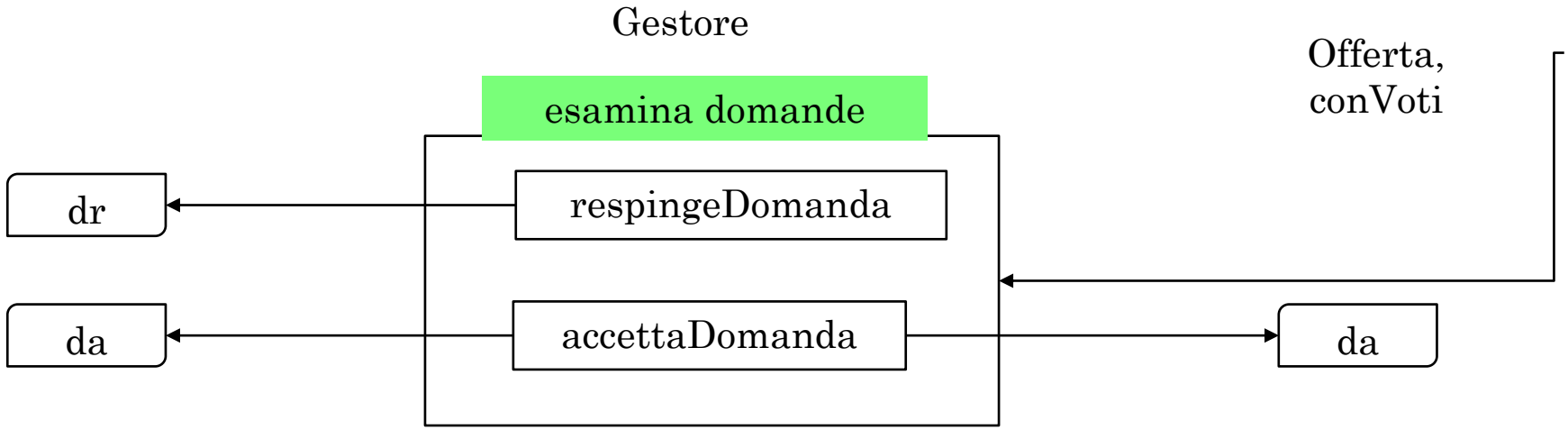
Variante

Voto ← ref Domanda

Attributo: int val (4..8). Si tratta di un attributo del voto.

Voto

Attributo: int val (4..8).



Il gestore accetta la domanda che ha la somma dei voti maggiore.
accettaDomanda: pre: domande.max (voti.val.sum). Il risultato è la domanda.

Processo con loop

Task d'interazione con restrizione

GestioneRichieste 2

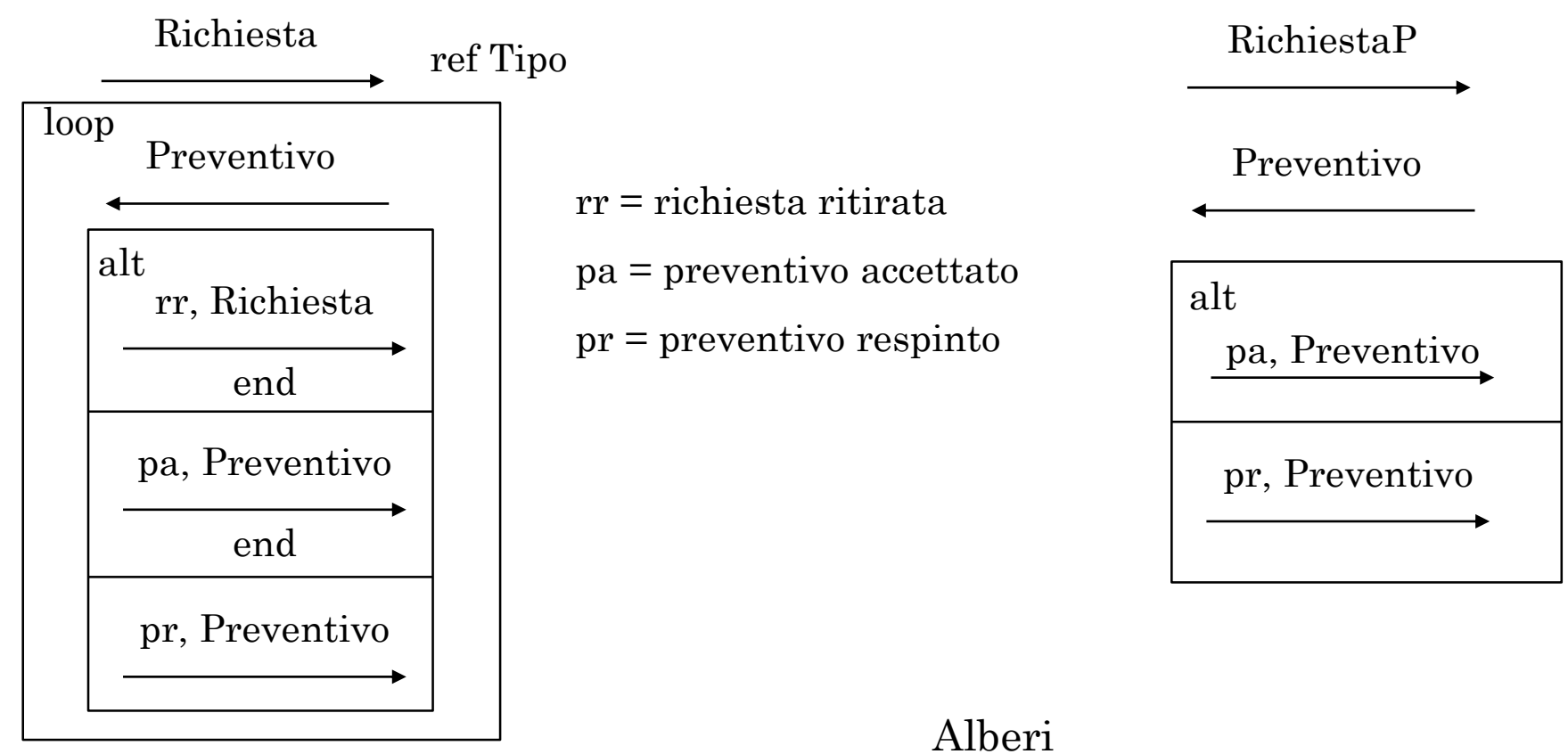
Il processo B2B GestioneRichieste opera in un'agenzia che tratta le richieste d'acquisto (Richiesta) provenienti dai clienti. Nel sistema informativo sono registrati i clienti, i fornitori, i tipi (di prodotto) e le relazioni tra i fornitori e i tipi. Ogni cliente è associato ad un accountMgr (ruolo di staff). Ogni richiesta è associata ad un tipo.

Le richieste provenienti dai clienti sono collocate nello stato pendente. Per ciascuna di esse l'accountMgr corrispondente genera una richiesta di preventivo (RichiestaP) diretta ad un fornitore che tratta il tipo associato alla richiesta (1). Il fornitore risponde con un preventivo che il processo invia al cliente. Il cliente può accettare o respingere il preventivo, oppure può ritirare la richiesta. Se respinge il preventivo, il processo informa il fornitore (che il preventivo è respinto), inoltre ricolloca la richiesta associata al preventivo (tramite la richiestaP) nello stato pendente dove è trattata come descritto in precedenza. Se il cliente ritira la richiesta, il processo informa il fornitore che il preventivo è respinto. Se il cliente accetta il preventivo, il processo informa il fornitore che il preventivo è accettato.

Note: I fornitori ai quali sono indirizzate le richieste di preventivo relative alla stessa richiesta d'acquisto devono essere distinti (1). Una richiesta può essere associata a n preventivi (tramite le richiesteP). I preventivi hanno uno stato (accettato o respinto).

(1) Si esprima il vincolo con un invariante.

Collaborazioni

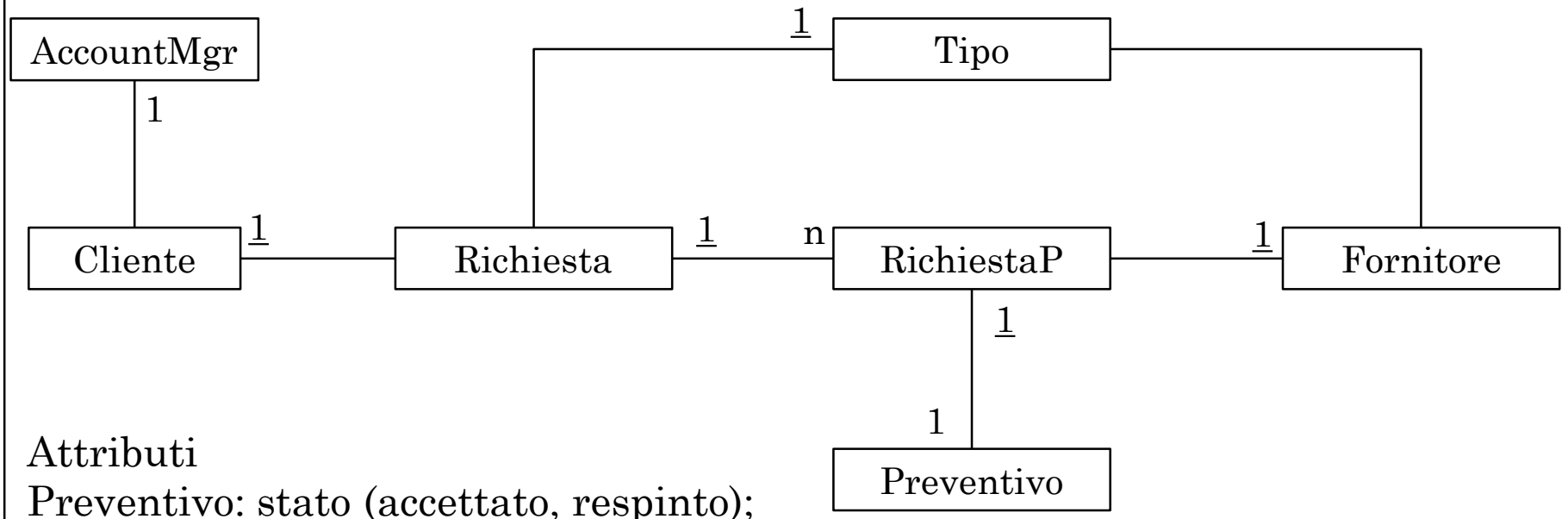


Alberi

Richiesta - Tipo
Preventivo

RichiestaP
Preventivo

Modello informativo



Attributi
Preventivo: stato (accettato, respinto);

Invarianti:
richiestaP.fornitore in richiestaP.richiesta.tipo.fornitori
richiesta.richiesteP.fornitore distinct.

Nota: Richiesta – Preventivo è una relazione derivata

Nota
Con l'invariante si valida l'associazione richiestaP.fornitore

Note

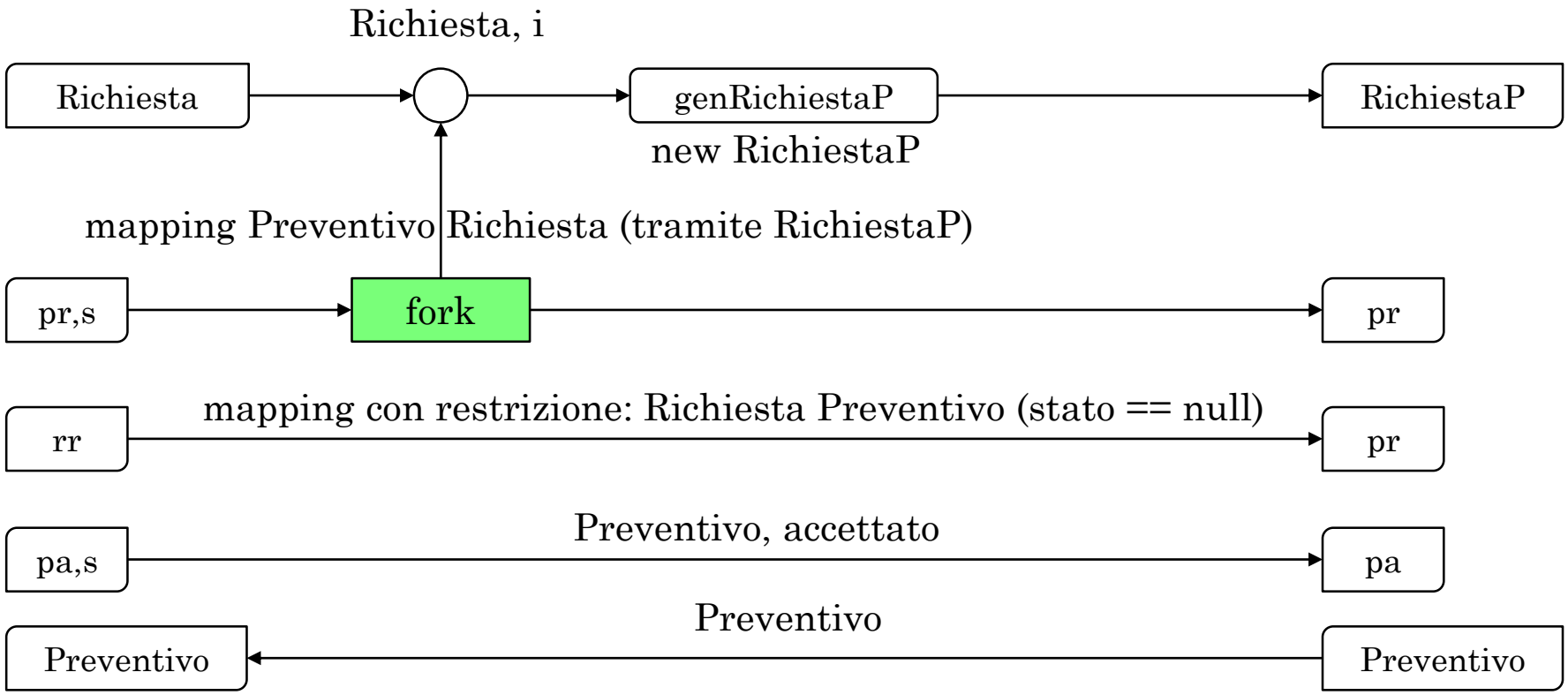
Ricevuto un preventivo, il cliente risponde in 3 modi: l'accetta, lo respinge o ritira la richiesta. Nei primi due casi lo stato del preventivo è accettato o respinto.

Se il preventivo è respinto, la richiesta associata al preventivo torna nello stato pendente e il fornitore è informato (serve un fork).

Se il preventivo è accettato, il fornitore è informato e la richiesta non torna nello stato pendente.

Se la richiesta è respinta, il fornitore dell'ultimo preventivo, che è quello con lo stato null, è informato e la richiesta non torna nello stato pendente.

Una richiesta può tornare più volte nel posto Richiesta,i; quindi il processo ha un *loop*.



* mapping con restrizione

rr = richiesta ritirata
pa = preventivo accettato
pr = preventivo respinto

Variante

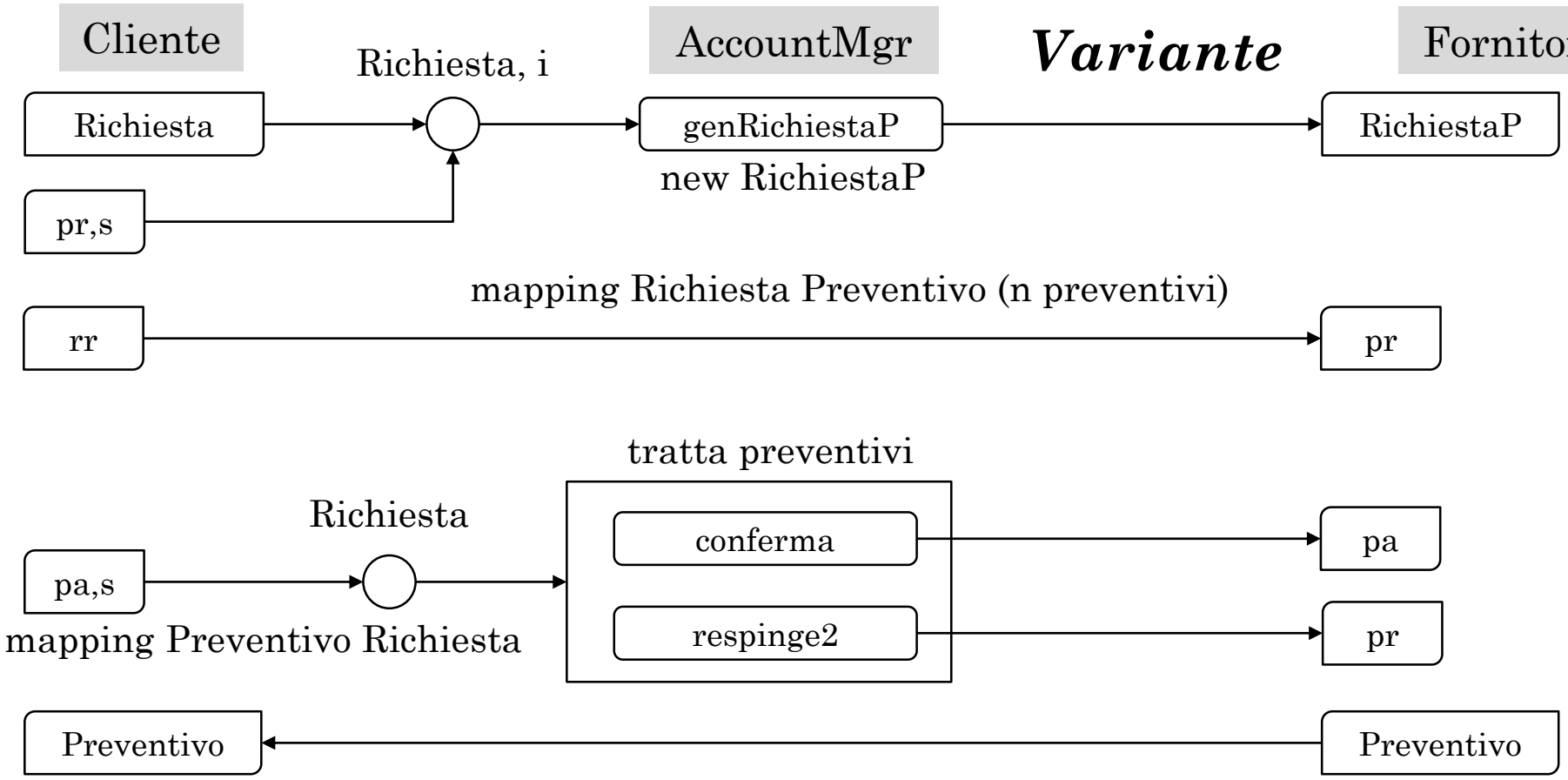
Ricevuto un preventivo, il cliente risponde in 3 modi: l'accetta, lo respinge o ritira la richiesta. Nei primi due casi lo stato del preventivo è accettato o respinto.

Se il preventivo è respinto, la richiesta associata al preventivo torna nello stato pendente.

Se la richiesta è respinta, l'accountMgr respinge tutti i preventivi e i fornitori sono informati.

Se il preventivo è accettato, l'accountMgr conferma il preventivo e respinge gli altri: il processo informa i fornitori dell'esito dei loro preventivi.

Nota: si colleghi al preventivo accettato un posto di mapping di tipo Richiesta per poi usare una scelta composta.



conferma: pre: stato == accettato.
respinge2: stato == respinto.

rr = richiesta ritirata
pa = preventivo accettato
pr = preventivo respinto

SOA Service Oriented Architecture

Gli attori del sistema interagiscono mediante servizi. I servizi sono funzionalità accessibili mediante protocolli di rete.

Un servizio è attivato mediante una richiesta ed è concluso da una risposta. Nei casi semplici, la richiesta e la risposta sono sincrone; in generale sono asincrone. Richiesta e risposta sono trasmesse mediante messaggi (ad es. HTTP).

SoaML (SOA Modeling Language) offre un approccio alla modellazione di sistemi SOA mediante l'uso dei diagrammi UML.

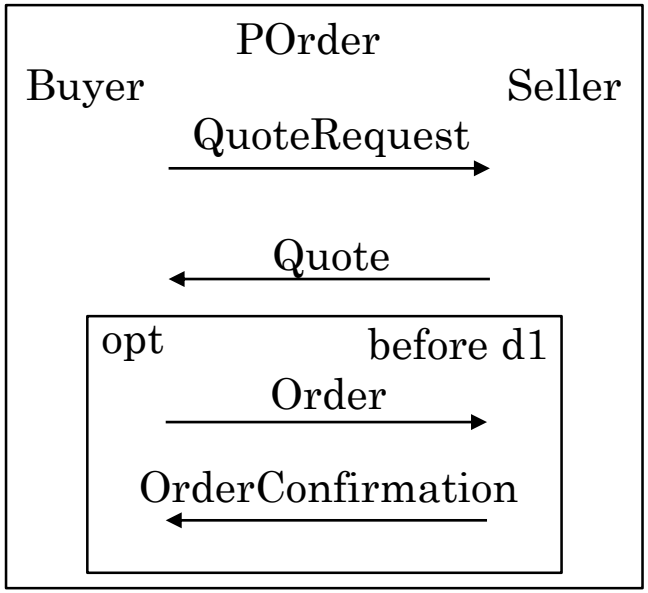
Alcuni esempi sono tratti dalla documentazione OMG (https://www.omg.org/spec/SoaML/About-SoaML/)

con alcune varianti.

Collaborazione

Una collaborazione definisce un'interazione complessa che è modellata con un sequence diagram (sequenza di richieste e risposte). La collaborazione ha un simbolo particolare.

L'esempio tratta una collaborazione d'acquisto tra un buyer e un seller.



QuoteRequest: Date d1.

La collaborazione PlaceOrder è collegata al sequence diagram POrder.

Sono classi attive

Domain Driven Design (DDD)

Eric Evans, Domain Driven Design, Addison Wesley, 2003.

Obiettivo: gestire la complessità quando un domain model è grande e interessa vari team di sviluppatori.

Concetti principali:

- bounded context
- ubiquitous language
- capability

Bounded context

Porzione di un domain model costituita da classi strettamente correlate e con pochi collegamenti con altri bounded context.

Le classi si suddividono in varie categorie:

Entity: elementi che hanno un'identità specifica ed evolvono nel tempo (stateful). Es. persona (codice fiscale), studente (matricola).

Value Object: elementi che non hanno un'identità specifica e sono generalmente immutabili (stateless). Es. data, indirizzo.

AggregateRoot: la classe che gestisce la consistenza degli elementi di un bounded context (come una classe di facciata). Per avere l'accesso dall'esterno ad un elemento occorre chiederlo all'aggregateRoot.

Repository: definisce le operazioni riguardanti la persistenza e le interrogazioni.

Ubiquitous language

In questo contesto significa un *linguaggio comune* a esperti di dominio e sviluppatori.

La base è un domain model che attraverso considerazioni e discussioni tra gli stakeholder deve essere approfondito e raffinato finché non risulta completo e comprensibile a tutti.

Capability

La capacità di operare in un certo ambito, ad es. in un'area di business (gestione ordini, servizio clienti, marketing …).

Un bounded context è assegnato ad un team che è cross-funzionale (cioè include persone con le capability necessarie).

Microservices

In short, the microservice architectural style is an approach to developing a single application as a *suite of small services*, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around *business capabilities* and are *independently deployable* by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in *different programming languages* and use *different data storage technologies*.

https://martinfowler.com/articles/microservices.html

Microservices

I programmi monolitici sono difficili da mantenere, es. banking domain. La modifica di una parte comporta l'aggiornamento e il rilascio di tutto il sistema.

Un'architettura a microservizi ha queste caratteristiche:

si compone di piccoli servizi (ciascuno eseguito nel proprio processo) che interagiscono con protocolli leggeri

un microservizio può essere aggiornato indipendentemente dagli altri ed è scalabile in modo autonomo

i microservizi possono usare linguaggi di programmazione e database diversi; il sistema è quindi poliglotta (polyglot)

un microservizio è gestito da un unico team; Amazon's motto “you build, you run it”. Il team è cross-funzionale; riunisce quindi competenze diverse (user experience, logica lato server, database, project management, …)

Rapporti tra Domain Driven Design e microservizi

Il DDD suddivide il domain model in vari bounded context.

Solitamente un bounded context corrisponde ad un microservizio.

I microservice operano su database separati. I bounded context offrono quindi la base di tale separazione.

Tuttavia transazioni e interrogazioni (query) pongono problemi alla decomposizione.

Transazioni

In un programma monolitico che opera su un singolo database gli aggiornamenti dei dati sono eseguiti in modo transazionale.

Se il database è distribuito si possono usare transazioni distribuite che sono però difficilmente applicabili ai microservizi.

La soluzione consiste nel coordinamento dei microservizi con eventuali operazioni di compensazione (cioè di ripristino dello stato precedente).

Un programma monolitico può essere più performante ma è meno flessibile circa la scalabilità e l'evoluzione.

Evoluzione da monoliti a microservizi

Integrazione

Il monolito è il core del sistema e nuove funzionalità spesso temporanee sono aggiunte mediante microservizi che chiamano le API del monolito.

Esempi di questo approccio provengono da banche e giornali.

Sostituzione

Un monolito esistente, quando dà problemi, può essere suddiviso in componenti che diventano microservizi. Le comunicazioni interne vanno in parte trasformate in interazioni e contratti tra servizi.

Nuova applicazione basata su microservizi

Si parte da zero (from scratch).

Esempio di progetto con microservizi

Il progetto riguarda una piattaforma che consente agli utenti di mettere in vendita e acquistare libri.

Per iscriversi occorre inserire nome, username (univoco), password, indirizzo e email. Le informazioni di un libro includono titolo, autori, editore e prezzo; il sistema aggiunge l'id del libro.

Siccome possono esserci in vendita varie copie dello stesso libro, il sistema distingue tra libro (Book) e informazioni (BookInfo) che valgono per tutte le copie. BookInfo contiene titolo, autori ed editore di riferimento; Book contiene il prezzo, l'editore e lo stato (disponibile, prenotato, venduto …).

Esempio di progetto con microservizi

Per mettere in vendita un libro, l'utente (venditore) lo inserisce nella piattaforma. Per acquistare dei libri, un utente (compratore) genera un ordine e aggiunge gli id dei libri che sceglie tramite ricerche (per titolo e/o autori).

Con il comando **placeOrder** un utente può porre in esecuzione l'ordine. Il sistema controlla che tutti i libri siano disponibili e li pone nello stato prenotato. Poi chiede il pagamento al compratore. Il *pagamento* avviene mediante carta di credito ed è effettuato tramite un servizio di pagamento esterno. Infine il sistema genera un *ordine di spedizione* per un corriere e glielo invia; il corriere manda i dettagli della spedizione.

Il comando placeOrder fallisce se non sono disponibili tutti i libri o il pagamento non va a buon fine. In tal caso le azioni intraprese vanno cancellate (*compensazione*).

Note

L'inserimento di un libro in vendita richiede vari passi: la ricerca del bookInfo corrispondente, l'inserimento del bookInfo se non esiste, l'inserimento del libro con l'associazione al bookInfo.

I corrieri sono inseriti dall'*amministratore* della piattaforma (utente predefinito).

Un utente può modificare l'indirizzo.

Dato un ordine, con il servizio *getBookList* l'utente può visualizzare un elenco di libri disponibili tra i quali può sceglierne uno da aggiungere all'ordine.

Per ogni entità inserita il sistema genera un id univoco.

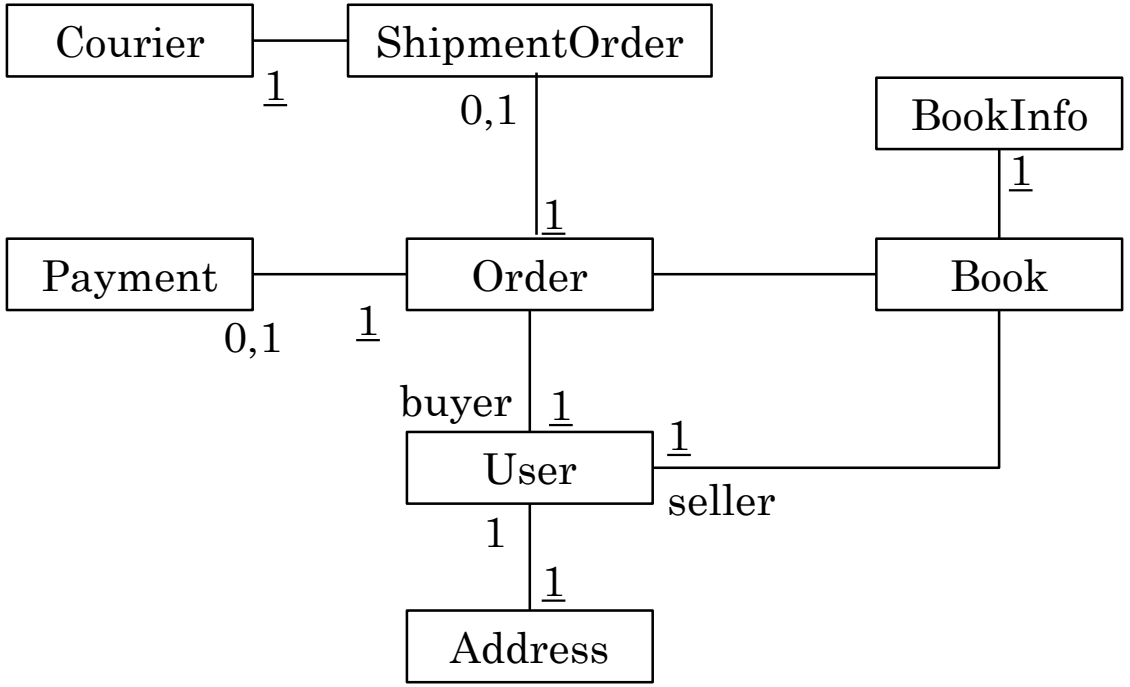
Estensioni

L'inserimento di un nuovo utente può richiedere alcuni passi: ricezione domanda con copia della carta d'identità, verifica dei dati, inserimento dell'utente, invio della conferma.

Un corriere può inviare informazioni di tracciamento di una spedizione.

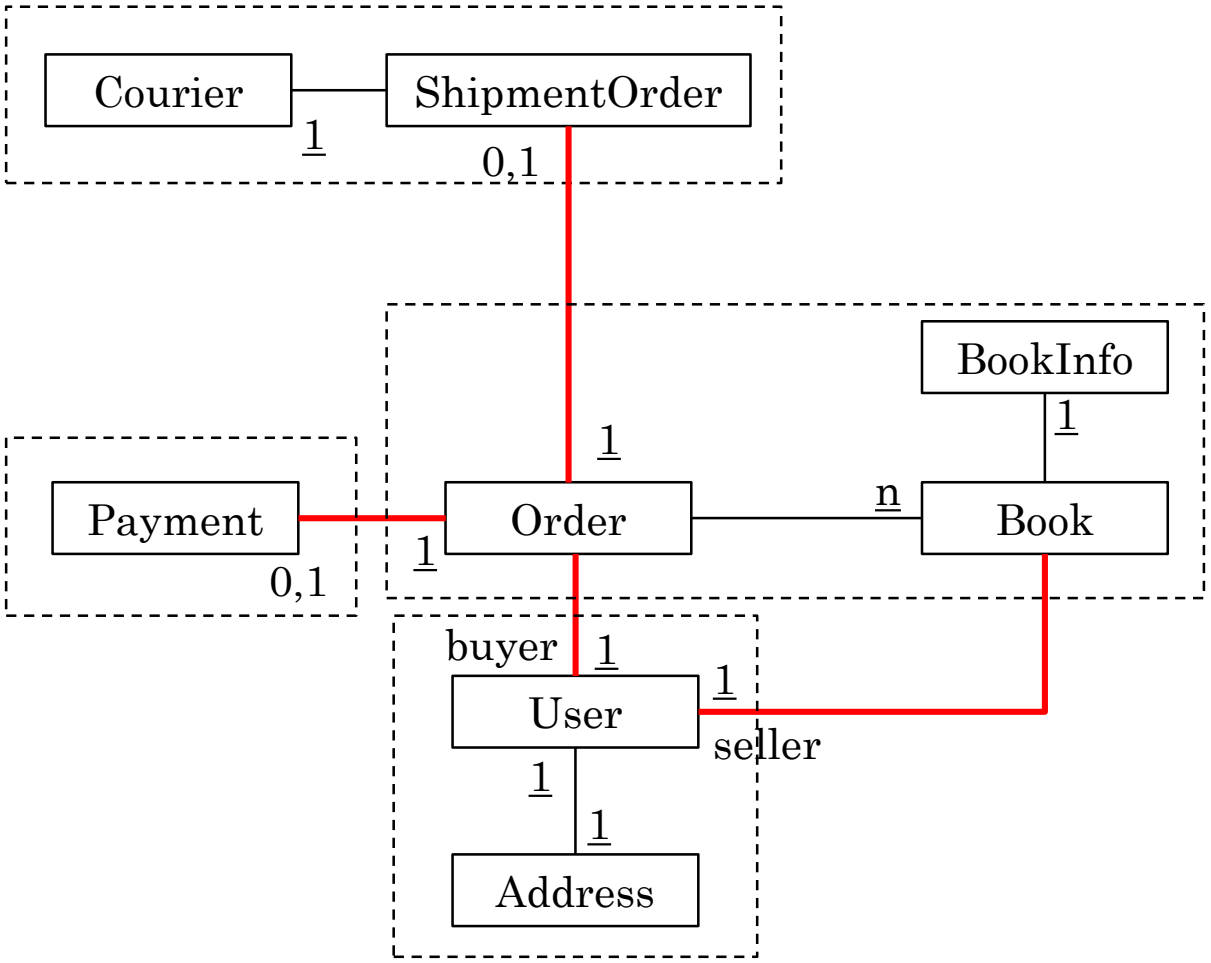
Dal pagamento di un compratore derivano 3 pagamenti: per il venditore, per il corriere e per la piattaforma.

Domain model



Attributi
User: String userName (univoco), String name, String passw, String email.
Address: String town …
BookInfo: String title, String authors, String publisher.
Book: String publisher, Double price, state (available, reserved, …).
Order: Date date, state (placed, accepted, rejected, …)

Bounded context



In rosso i legami informativi tra microservizi

Quali criteri?
Gestione utenti
Gestione ordini
Gestione spedizioni
Gestione pagamenti

Le capacità richieste sono diverse, le evoluzioni sono disaccoppiate.