SOGGETTI

cliente

acmesky

compagnie aeree

prontogram

banca

noleggi

OPERAZIONI

\* il cliente specifica il periodo e il costo (+ volte)

\* acmesky interroga le compagnie (+ volte)

\* acmesky riceve risposta\_offerte last minute

\* acmesky offre le risposta\_offerte last minute ai clienti compatibili con prontogram

\* il cliente riceve il codice offerta su prontogram

\* il cliente inserisce il codice offerta (risposta\_offerte normali e l.m.) sul portale di acmesky

\* il cliente "paga" acmesky

\* acmesky inoltra il cliente sul fornitore di servizi bancari

\* il fornitore conferma il pagamento ad acmesky

\* acmesky offre al cliente il servizio di trasferimento

\* acmesky accetta richiesta di trasferimento (se il cliente dice si)

\* acmesky richiede trasferimento alla compagnia di noleggio con autista

# 

# Abstract

La seguente relazione intende descrivere le scelte progettuali e implementative che il nostro gruppo, costituito da S. Fabbri, M. Perlino e G. Spiller, ha intrapreso nella realizzazione del progetto relativo all’insegnamento di Ingegneria del Software orientato ai Servizi.

# Modellazione scenario

La prima attività che abbiamo compiuto è stata la lettura attenta della descrizione del dominio e del problema; individuati i partecipanti coinvolti e comprese le operazioni salienti in cui essi sono impegnati, abbiamo proceduto con la modellizazione dello scenario mediante una coreografia.

Per la definizione formale della coreografia ci siamo serviti di un *process calculi*, essendo ritenuto più accurato di WS-CDL e della formalizzazione per le coreografie di BPMN: mediante il suo formalismo, abbiamo potuto comprendere in maniera puntuale quanto accade nello scenario, nonchè raffinare la coreografia stessa per migliorarne le proprietà di connectedness. Queste ultime sappiamo essere proprietà sufficienti (e non necessarie) a garantire una successiva corretta proiezione.

Per definizione, una coreografia descrive il comportamento di un sistema distribuito da un punto di vista globale. A partire da una coreografia è possibile derivare le specifiche implementative dei singoli partecipanti mediante l’operazione detta *proiezione*: essa permette appunto di “proiettare” la prospettiva globale descritta dalla coreografia sui singoli punti di vista locali dei partecipanti. Una proiezione eseguita correttamente assicura che il sistema derivante dall’interazione dei singoli partecipanti così implementati aderisca, nel complesso, alla coreografia originaria.

## 

## Coreografia

La coreografia cui siamo giunti dopo numerosi confronti, riflessioni e iterazioni è presentata di seguito:

C ::= (invio\_interessi: cliente --> acmesky)\*

|

(( (richiesta\_risposta\_offerte: acmesky --> compagnia\_aerea\_a ;

risposta\_offerte: compagnia\_aerea\_a --> acmesky) |

... |

(richiesta\_risposta\_offerte: acmesky --> compagnia\_aerea\_z ;

risposta\_offerte: compagnia\_aerea\_z --> acmesky) ) ;

1 + (comunica\_risposta\_offerte: acmesky --> prontogram ;

inoltra\_risposta\_offerte: prontogram --> cliente\_1 |

... |

inoltra\_risposta\_offerte: prontogram --> cliente\_n ))

|

((last\_minute: compagnia\_aerea\_a --> acmesky ; 1 + (comunica\_risposta\_offerte\_LM: acmesky --> prontogram ;

inoltra\_risposta\_offerte\_LM: prontogram --> cliente\_1a | ... | inoltra\_risposta\_offerte\_LM: prontogram --> cliente\_na)) |

... |

(last\_minute: compagnia\_aerea\_z --> acmesky ; 1 +

(comunica\_risposta\_offerte\_LM: acmesky --> prontogram ;

inoltra\_risposta\_offerte\_LM: prontogram --> cliente\_1z | ... | inoltra\_risposta\_offerte\_LM: prontogram --> cliente\_nz ))

)

|

(invio\_codice\_offerta: cliente --> acmesky ;

1 + (inoltro\_servizi\_bancari: acmesky --> cliente ;

pagamento: cliente --> fornitore\_servizi\_bancari ;

1 + (conferma\_pagamento: fornitore\_servizi\_bancari --> acmesky ;

1 + (propongo\_trasferimento: acmesky --> cliente ;

1 + prenota\_trasferimento: acmesky --> compagnia\_vicina))) )

Per illustrare ogni punto di questa coreografia, la articoliamo in 4 blocchi per poi focalizzarci su ognuno di essi.

registrazione\_interessi\_utente ::= (invio\_interessi: cliente --> acmesky)\*

ricerca\_inoltro\_risposta\_offerte

C ::= registrazione\_interessi\_utente | ricerca\_inoltro\_risposta\_offerte | ricezione\_inoltro\_LM

Seguitamente, abbiamo discusso circa le *proprietà di connectedness* soddisfatte ovvero violate da questa coreografia.  
Abbiamo dovuto comprendere se lo scenario oggetto del progetto rientrasse nel caso sincrono ovvero in quello asincrono; a dirimere la questione sono stati, tra gli altri, i clienti che, secondo la nostra modellazione, possono indicare le proprie esigenze sul portale ACMESky più volte, ancor prima di ricevere le eventuali risposta\_offerte corrispondenti e le risposta\_offerte last-minute che possono, in qualunque momento, essere inviate dalle compagnie aeree alla volta di ACMESky.  
Pertanto, siamo giunti alla conclusione che questo scenario rientri nel *caso asincrono*.

In dettaglio, abbiamo valutato ogni operatore sequenziale, di scelta non deterministica, ciascuna iterazione, nonchè l’utilizzo multiplo di medesime operazioni: quindi, abbiamo rifinito la coreografia scritta in principio, migliorando le proprietà di connectedness, arrivando alla coreografia sopra presentata, che risulta essere connessa per le scelte condizionali e per sequenza secondo il pattern Sender.

## Proiezione

~~cliente~~ - ~~acmesky~~ - ~~compagnia\_aerea\_a~~ - ~~compagnia\_aerea\_z~~ - ~~cliente\_1~~ - ~~cliente\_n~~ - ~~cliente\_1a~~ - ~~cliente\_na~~ - ~~cliente\_1z~~ - ~~cliente\_nz~~ - ~~fornitore\_servizi\_bancari~~ - ~~compagnia\_vicina~~ - ~~prontogram~~

Cliente

**proj(C, cliente):** (@acmesky)\* |

(( (1;1) | ... | (1;1) ) ; 1 + (1 ; 1 | ... |1 )) | ((1 ; 1 + (1; 1 | ... | 1)) | ... |(1 ; 1 + (1 ; 1 | ... | 1 )) ) | (@acmesky ; 1 + (inoltro\_servizi\_bancari@acmesky ; @fornitore\_servizi\_bancari ; 1 + (1 ; 1 + (propongo\_trasferimento@acmesky ; 1 + 1))))

Cliente\_1

**proj(C, cliente\_1):** (1)\* |(( (1;1) | ... | (1;1));

1 + (1 ; inoltra\_risposta\_offerte@prontogram |... | 1))

| ((1 ; 1 + (1; 1 | ... | 1)) | ... |(1 ; 1 + (1 ; 1 | ... | 1 )) ) | (1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + 1))))

Cliente\_n

**proj(C, cliente\_n):** (1)\* |(( (1;1) | ... | (1;1));

1 + (1 ; 1 |... | inoltra\_risposta\_offerte@prontogram))

| ((1 ; 1 + (1; 1 | ... | 1)) | ... |(1 ; 1 + (1 ; 1 | ... | 1 )) ) | (1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + 1))))

Cliente\_1a

**proj(C, cliente\_1a):** (1)\* |(( (1;1) | ... | (1;1));

1 + (1 ; 1 |... | 1))

| ((1 ; 1 + (1; inoltra\_risposta\_offerte\_LM@prontogram | ... | 1)) | ... |(1 ; 1 + (1 ; 1 | ... | 1 )) ) | (1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + 1))))

Cliente\_1z

**proj(C, cliente\_1z):** (1)\* |(( (1;1) | ... | (1;1));

1 + (1 ; 1 |... | 1))

| ((1 ; 1 + (1; 1 | ... | 1)) | ... |(1 ; 1 + (1 ; inoltra\_risposta\_offerte\_LM@prontogram | ... | 1)) ) | (1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + 1))))

Cliente\_na

**proj(C, cliente\_na):** (1)\* |(( (1;1) | ... | (1;1));

1 + (1 ; 1 |... | 1))

| ((1 ; 1 + (1; 1 | ... | inoltra\_risposta\_offerte\_LM@prontogram )) | ... |(1 ; 1 + (1 ; 1 | ... | 1 )) ) | (1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + 1))))

Cliente\_nz

**proj(C, cliente\_nz):** (1)\* |(( (1;1) | ... | (1;1));

1 + (1 ; inoltra\_risposta\_offerte@prontogram |... | 1))

| ((1 ; 1 + (1; 1 | ... | 1)) | ... |(1 ; 1 + (1 ; 1 | ... | inoltra\_risposta\_offerte\_LM@prontogram )) ) | (1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + 1))))

Compagnia\_vicina

**proj(C, compagnia\_vicina):** (1)\* | (( (1;1) | ... | (1;1) ) ; 1 + (1 ; 1 | ... |1 )) | ((1 ; 1 + (1; 1 | ... | 1)) | ... |(1 ; 1 + (1 ; 1 | ... | 1 )) ) | (1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + prenota\_trasferimento@acmesky))))

Fornitori\_servizi\_bancari

**proj(C, fornitori\_serivizi\_bancari):** (1)\* | (( (1;1) | ... | (1;1) ) ; 1 + (1 ; 1 | ... |1 )) | ((1 ; 1 + (1; 1 | ... | 1)) | ... |(1 ; 1 + (1 ; 1 | ... | 1 )) ) | (1 ; 1 + (1 ; pagamento@cliente ; 1 + (@acmesky ; 1 + (1 ; 1 + 1))))

Compagnia\_aerea\_a

**proj(C, compagnie\_aerea\_a):**

(1)\* | (( (richiesta\_risposta\_offerte@acmesky ; @acmesky) | ... | (1;1) ) ; 1 + (1 ; 1 | ... |1 )) | ((@acmesky ; 1 + (1; 1 | ... | 1)) | ... |(1 ; 1 + (1 ; 1 | ... | 1 )) ) | (1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + 1))))

Compagnia\_aerea\_z

**proj(C, compagnie\_aerea\_z):** (1)\* | (( (1;1) | ... | (richiesta\_risposta\_offerte@acmesky ; @acmesky) ) ; 1 + (1 ; 1 | ... |1 )) | ((1 ; 1 + (1; 1 | ... | 1)) | ... |(@acmesky ; 1 + (1 ; 1 | ... | 1 )) ) | (1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + 1))))

ACMESky

**proj(C, acmesky):**

(invio\_interessi@cliente)\*

|

(

( (@compagnia\_aerea\_a ; risposta\_offerte@compagnia\_aerea\_a) | ... | (@compagnia\_aerea\_z ; risposta\_offerte@compagnia\_aerea\_z) ) ;

1 + (@prontogram ;

1 | ... | 1 )) | ((last\_minute@compagnia\_aerea\_a ; 1 + (@prontogram ;

1 | ... | 1)) | ... | (last\_minute@compagnia\_aerea\_z; 1 +

(@prontogram ; 1 | ... | 1 )))|

(invio\_codice\_offerta@cliente@cliente ; 1 + (@cliente ;

1 ;

1 + (conferma\_pagamento@fornitore\_servizi\_bancari ;

1 + (@cliente ;

1 + @compagnia\_vicina))) )

Prontogram

**proj(C, prontogram):** (1)\* |(( (1;1) | ... | (1;1) ) ; 1 + (comunica\_risposta\_offerte@acmesky ; @cliente\_1 |

... | @cliente\_n))|

((1 ; 1 + (comunica\_risposta\_offerte\_LM@acmesky ; @cliente\_1a |

... | @cliente\_na)) | ... | (1 ; 1 +

(comunica\_risposta\_offerte\_LM@acmesky ;

@cliente\_1z| ... | @cliente\_nz )))|

(1 ; 1 + (1 ; 1 ; 1 + (1 ; 1 + (1 ; 1 + 1))))

Codice univoco per cliente e non per offerta