Modello degli Attori
(Attori)

Gli ***attori*** sono agenti computazionali che mappano ogni messaggio che ricevono su una 3pla composta da:

- Un insieme di messaggi da inviare agli altri attori
- Un nuovo comportamento (usato per processare il messaggio successivo)
- Un insieme di nuovi attori

Nota:

- il comportamento di un attore può dipendere dalla storia
- non c'è un ordine con cui un attore esegue delle azioni (le sue azioni sono funzione del suo comportamento corrente e del messaggio che si sta processando)
- La creazione di nuovi attori è parte integrante del modello.

Un attore identificato da: un ***indirizzo mail*** e un ***comportamento***.

Modello degli Attori
(sistema ad attori)

Gli attori sono l'unità computazionale del modello.

La computazione è il risultato dei messaggi che gli attori si scambiano all'interno del sistema.

Le comunicazioni sono contenute in ***task***.

Un sistema ad attori evolve includendo nuovi task e nuovi attori che sono il risultato dell'elaborazione di task presenti nel sistema.

I task già elaborati e gli attori non più utilizzati sono eliminati dal sistema (garbage collection).

Modello degli Attori
(Task)

I task sono triple composte da:

- **Un’etichetta che identifica il task**
- **Un riferimento al *__indirizzo di posta__* dell’attore destinatario (*__target__*) del messaggio**
- **Le informazioni che l’attore dovrà elaborare (una tupla di valori)**

Modello degli Attori
(Comportamento di un Attore)

Gli attori *__accettano__* un messaggio quando processano un task contenente tale messaggio.

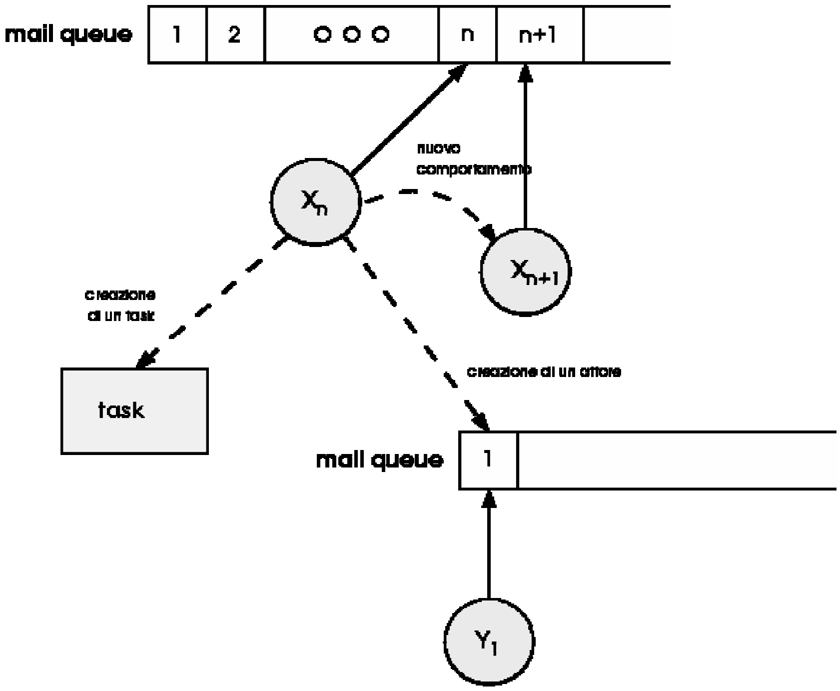
Un attore può processare solo quei task di cui è target.

Quando un attore accetta un messaggio può creare nuovi attori e task e rielabora (cambiandolo) il proprio comportamento.

L’ordine d’arrivo dei messaggi indirizzati ad uno specifico attore è lineare. Quindi deve essere presente un meccanismo (*__mail queue__*) di buffering e di arbitrazione dei messaggi in arrivo.

Quando un attore X_n accetta l’n−simo messaggio presente nella sua mail queue, creerà l’attore X_{n+1} che lo rimpiazzerà proseguendo nell’elaborazione del messaggio successivo presente nella coda di X_n.

Modello degli Attori
(esempio di transazione)



Modello degli Attori
(linguaggio di programmazione basato sugli attori)

Un linguaggio di programmazione pensato per supportare il modello computazionale degli attori deve permettere di:

- **Definire i comportamenti per gli attori.**
- **Creare nuovi attori, quindi fornire una primitiva (*new*).**
- **Creare nuovi task, fornire, quindi, comandi di invio.**
- **Dichiarare quali sono gli attori che possono interagire con l'esterno (definire dei *receptionist*).**
- **Dichiarare quali sono gli attori esterni al sistema con cui è possibile interagire.**

Gli ultimi due punti servono ad aumentare la modularità del sistema

Modello degli Attori

(definizione del comportamento di un attore)

Ogni comportamento deve prevedere un possibile comportamento che lo rimpiazzerà. Potrebbe dare origine ad una definizione infinita.

- **i comportamenti sono definiti per induzione (o ricorsione)**

ogni comportamento sarà parametrizzato sia sulle informazioni caratterizzati un attore (*acquaintance list*) sia sui messaggi che un attore può ricevere (*communication list*).

- **become <expressione>**

la primitiva become permette di cambiare il comportamento di un attore.

- **Comportamento per difetto**

Ogni attore deve avere un comportamento sostitutivo, quando non è definito per difetto l'attore viene rimpiazzato con un altro attore con lo stesso comportamento.

Modello degli Attori

(creare attori e task)

- **new <nome del comportamento> (<lista di aquaintance>)**

  un attore viene creato tramite un'invocazione di new, come valore di ritorno abbiamo l'indirizzo di mail del nuovo attore. <nome comportamento> identifica quale comportamento iniziale dovrà avere il nuovo attore. Tra parentesi sono inseriti i valori attuali per le aquaintance dell'attore.

- **send <messaggio> to <destinatario>**

  un task viene creato specificando il destinatario ed il contenuto del messaggio.

Modello degli Attori

(esempio: stack come lista linkata di attori)

Lo stack è una collezione di attori con comportamento uniforme.

```
def stack−node (content, link)
   [case operation of
      pop: (customer)
      push: (new−content)
   end case]
if operation = pop ∧ content ≠ NIL then
   become forwarder (link)
   send content to customer
fi
if operation = push then
   let P = new stack−node(content, link)
   {become stack−node (new−content, P)}
fi end def
```

Il top dello stack l'unico elemento che comunica con l'esterno ed creato con: let p = new *stack−node* (NIL, SINK)

Modello degli Attori

(esempio: fattoriale)

```
def exp rec−factorial() [n]
     become rec−factorial()
     if n=0 then reply [1]
     else reply [n*(call self[n−1])]
     fi
end def
```

Riferimenti Bibliografici

- Distributed Object−Based Programming Systems. Roger Chin and Samuel Chanson. In ACM Computer Surveys, Vol. 23(1), March 1991.
- Analysis of Inheritance Anomaly in Object−Oriented Concurrent Programming Language. Satoshi Matsuoka, and Akinori Yonezawa. In G. Agha, P. Wegner, and A. Yonezawa Editors Research Directions in OO Concurrent Programming. MIT Press. 1993.
- Gul Agha, Actors. MIT Press 1986. Capitolo 3.