

Unified Modeling Language

UML 2.0

- Sequence, Communication and Interaction

Overview diagrams -

Henry Muccini

Università degli Studi dell'Aquila

muccini@di.univaq.it



Engineering | IgTechnology | Imola
Informatica | Maggioli Informatica |
Micron Technology | Neta | Nous
Informatica | ObjectWay SED |
TechnoLabs | Taiprora

Master in Web Technology

IV Edizione 2007/08

Dipartimento di Informatica

Università degli Studi dell'Aquila

Copyright Notice

- » Il materiale riportato in queste slide puo' essere riutilizzato, parziale o totalmente, a patto che le fonti e gli autori vengano citati

Henry Muccini



Class Diagram: riassunto

- » Per riassumere i diagrammi mostrati la lezione precedente, far vedere:
 - > Elementi costituenti
 - > Semantica
 - > Sintassi di integrazione

- » Far vedere anche le relazioni tra i diagrammi



Sommario

» Interaction Diagram

> Sequence Diagram

- Generic form ed instance form
- Assi X e Y
- Focus of control
- Tipi di messaggi
- Creazione e distruzione di oggetti
- Branching e iteration
- Transition time
- Nuovi concetti in UML 2.0



Interaction Diagram

- » Mostra un'interazione tra un insieme di oggetti includendo i messaggi che vengono scambiati tra di loro
- » Utilizzati per esprimere l'aspetto dinamico di un sistema
- » Tipi
 - > Sequence diagram
 - > Communication diagram
 - (chiamati “Collaboration diagram” in UML 1.x)



Interaction diagram

- » Sequence Diagram e Collaboration Diagram sono **semanticamente equivalenti** dato che derivano dalle stesse informazioni nel meta-modello UML (in UML 1.x)
- » Da un diagramma è possibile passare all'altro (in UML 1.x)



Sequence Diagram

- » Enfatizza **l'ordine temporale** dei messaggi, cioè come i messaggi sono inviati e ricevuti tra gli oggetti durante il tempo
- » Particolarmente adatto per la modellazione di **sistemi real-time** e per la descrizione del **flusso di eventi degli use case**



Messaggi

- » Nella programmazione O.O. un'interazione tra oggetti è eseguita come un messaggio inviato da un oggetto ad un altro
- » Cosa si intende per “**invio di un messaggio da un oggetto A ad un oggetto B**”?
 - > A invoca un metodo di B
 - > B esegue l'operazione associata al metodo
 - > Il controllo ritorna ad A, con un eventuale valore di ritorno
- » Un messaggio puo' essere:
 - > Segnale
 - > Chiamata di procedura
 - > RPC (Remote Procedure Calls) in C++
 - > RMI (Remote Method Invocation) in Java



Messaggi

- » Sono presenti in **tutti** i diagrammi dinamici (sequence, collaboration, state e activity)
- » Rappresentano la **comunicazione** tra oggetti
- » Sono rappresentati tramite una linea con una freccia che va dall'oggetto che invia il messaggio (sender) all'oggetto che lo riceve (receiver)
- » **Differenti tipi di freccia indicano differenti tipi di messaggi**



Tipi messaggi

- » Linea continua con freccia chiusa piena
 - > Indica una chiamata di un'operazione o di un altro flusso di controllo annidato
 - > Se c'è un flusso di controllo annidato, il controllo passa al chiamante dopo che è terminata l'esecuzione del flusso annidato (**chiamata sincrona**)
- » Linea continua con Freccia aperta
 - > Indica una comunicazione **asincrona**, non c'è flusso di controllo annidato
 - > Sender invia lo stimolo e continua immediatamente con il prossimo passo nell'esecuzione
- » Linea tratteggiata con Freccia aperta
 - > Indica il **ritorno da una chiamata** di un'operazione
 - > Può essere omessa
- » In UML 2.0 sono stati introdotti altri tipi di messaggi, descritti nel seguito



Sequence Diagram

- » Asse X contiene gli oggetti che fanno parte del diagramma
- » **Oggetti possono essere creati o distrutti** all'interno di un diagramma
- » Creazione espressa utilizzando un messaggio di creazione
- » Distruzione indicata con una x



Transition Time

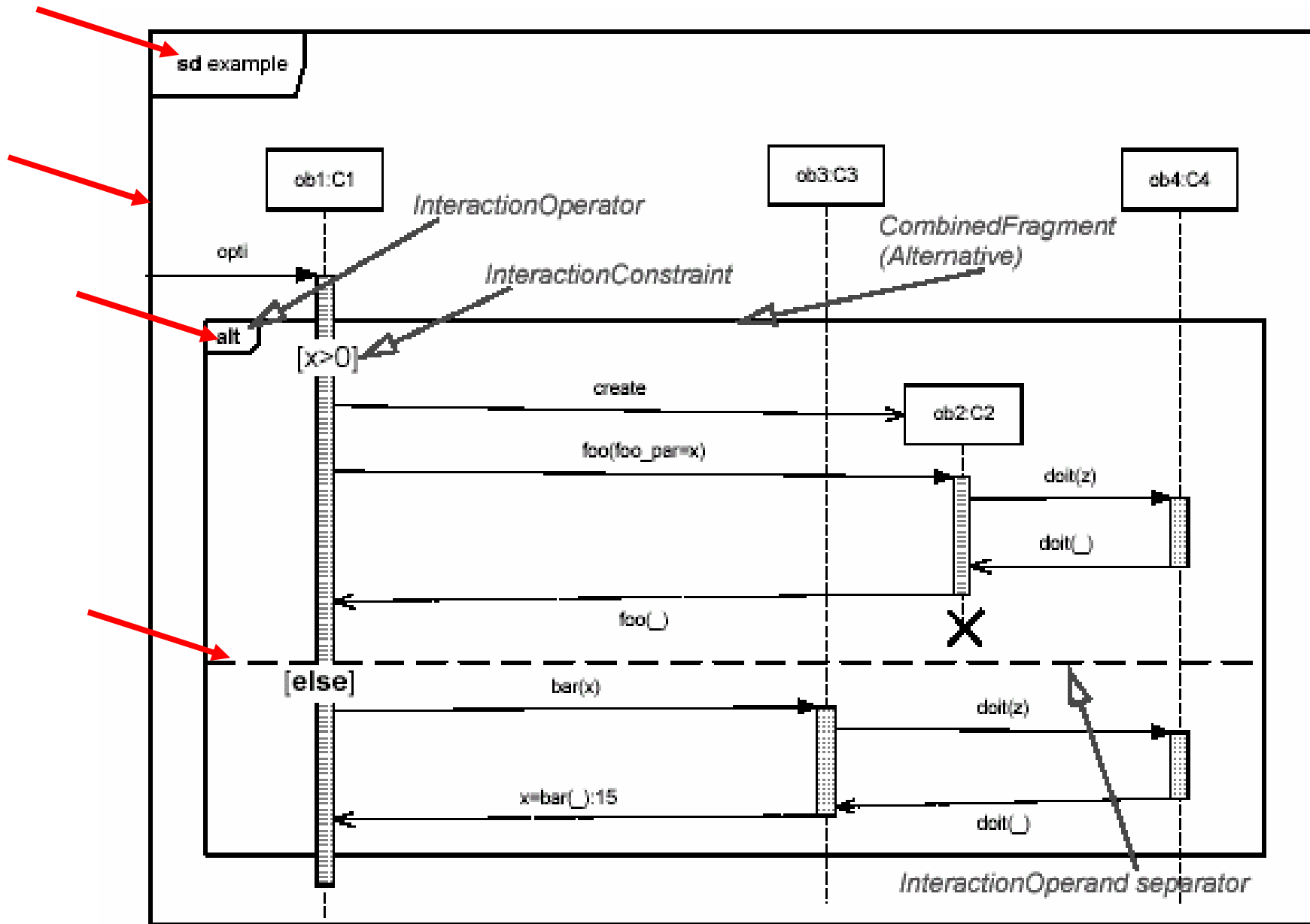
- » Viene utilizzato per esprimere **vincoli temporali** sul tempo di esecuzione dei messaggi
- » Espresso tramite **constraints**
- » In genere si trovano alla sinistra dei messaggi



Combined Fragment (CF)

- » Questo concetto permette di suddividere un Sequence Diagram in piu' "frammenti" e di **combinare** tali frammenti, seguendo diverse regole
- » Alcuni **operatori** (gli "InteractionOperator") definiscono come tali frammenti possono essere combinati insieme
- » Nel seguito, descriviamo tali operatori



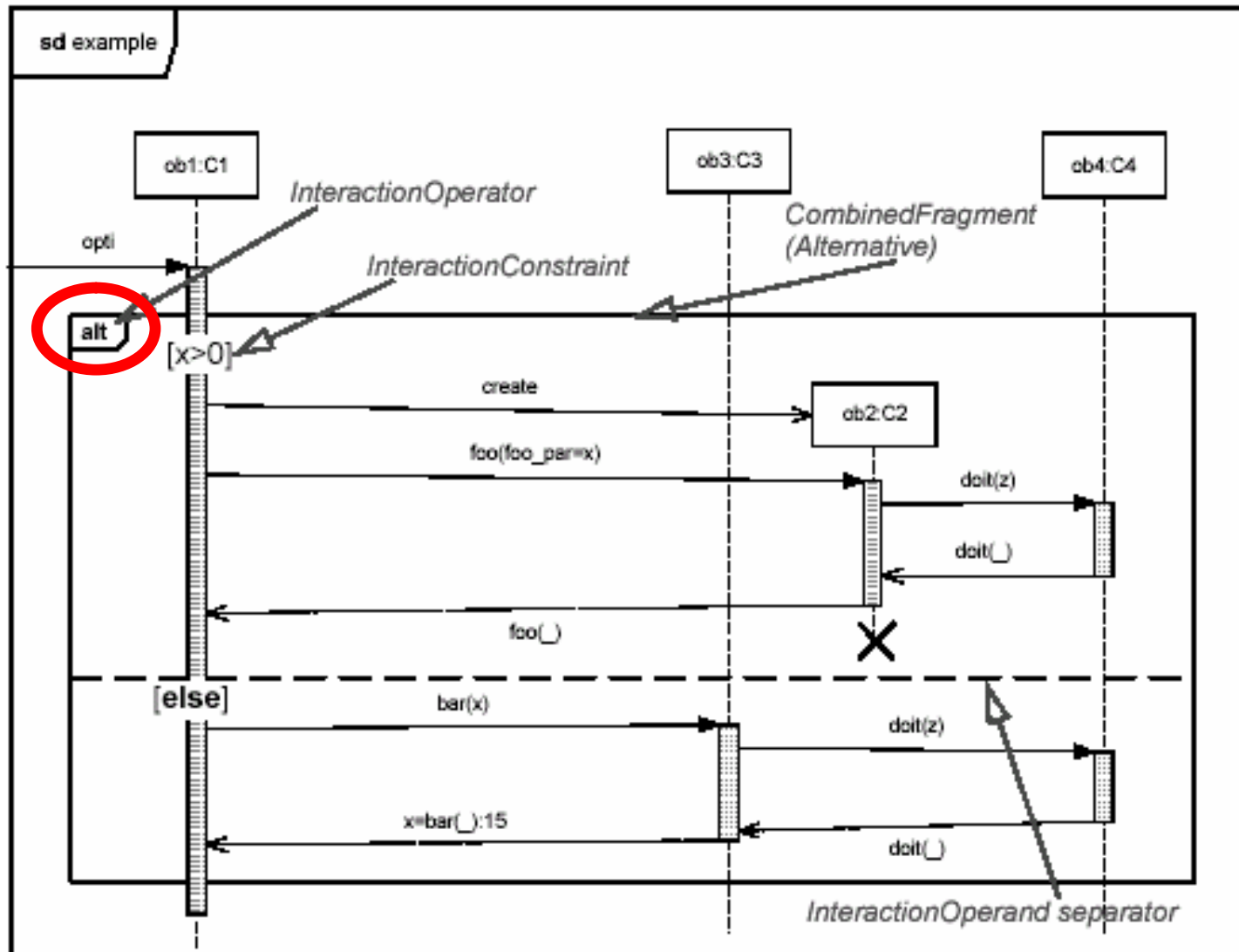


CF “Alternatives”

- » Operatore: **Alt**
- » Significato:
 - > rappresenta una scelta
 - > **Solo una** alternativa puo' essere selezionata
 - > Uso di “guardie”

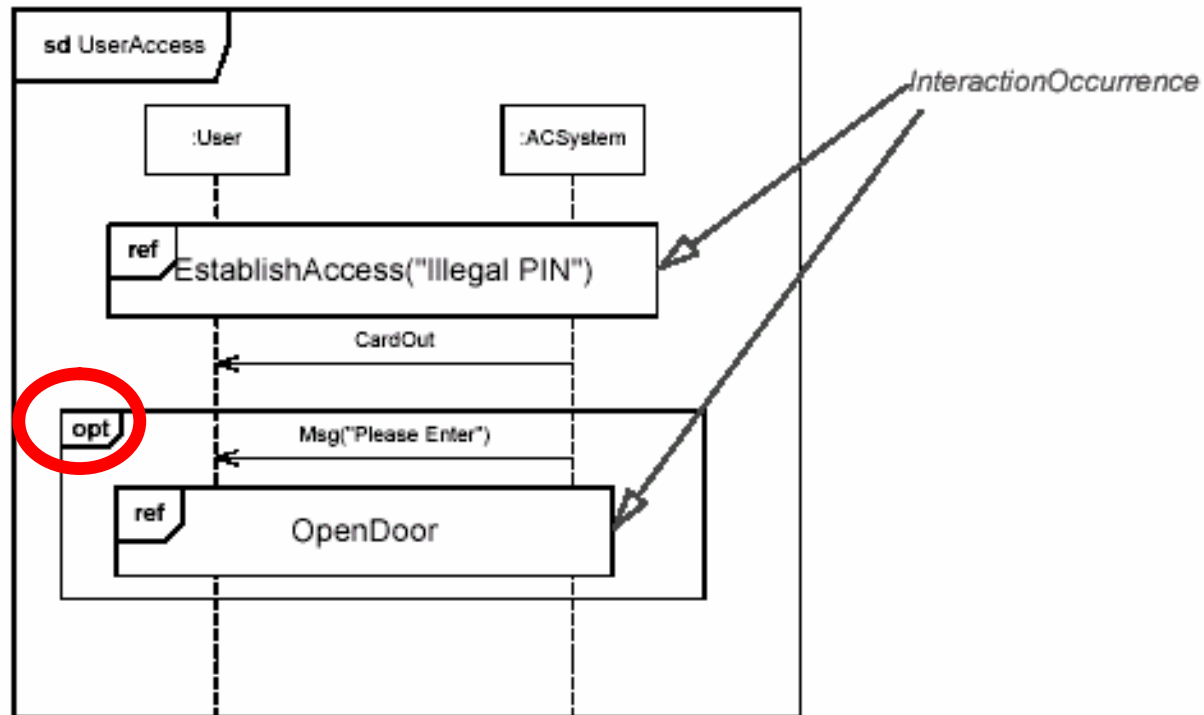


Alternatives



CF “Option”

- » Operatore: **Opt**
- » Significato:
 - > Questa frammento e' opzionale (nulla o tutto)

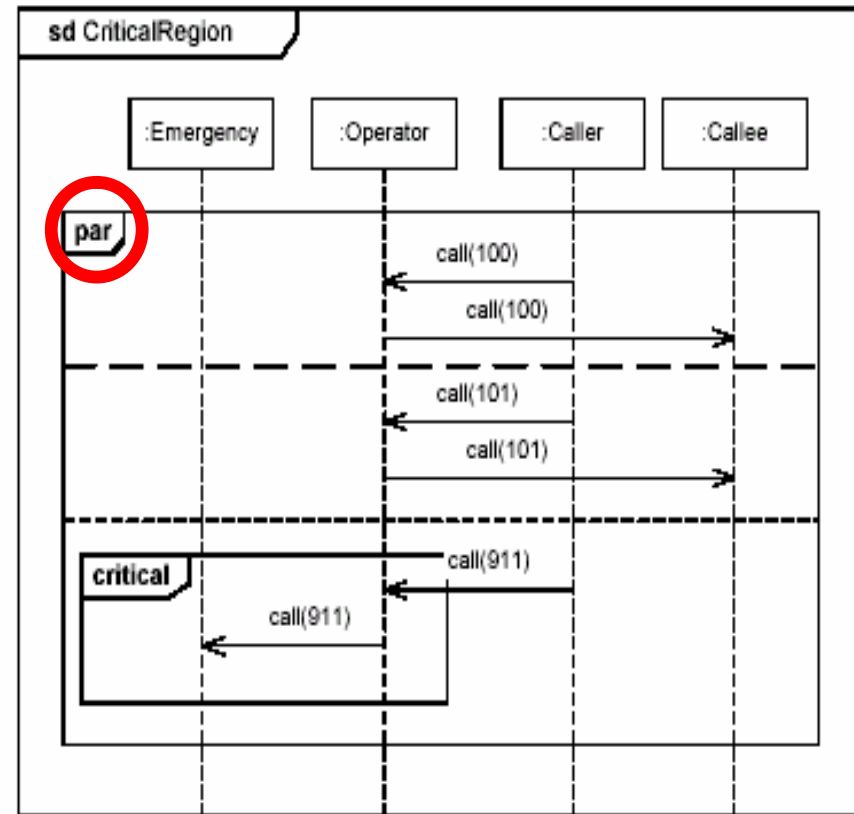


CF “Parallel”

» Operatore: **par**

» Significato:

- > I vari frammenti possono essere **interleaved**
- > l'ordine imposto da ciascun frammento deve però essere rispettato



CF “Weak Sequencing”

» Operatore: **seq**

» Significato:

> rappresenta una variazione al “par” dove:

- L’ordine di esecuzione degli eventi in ciascun frammento viene preservato
- eventi in **frammenti diversi e lifeline diverse**, possono occorrere in qualsiasi ordine
- eventi in **frammenti diversi e stessa lifeline** sono ordinati in modo tale da preservare l’ordine
- **Fare esempio**



CF “Negative”

- » Operatore: **neg**
- » Significato:
 - > rappresenta delle tracce da considerare **invalid**e
 - > **Esempio**

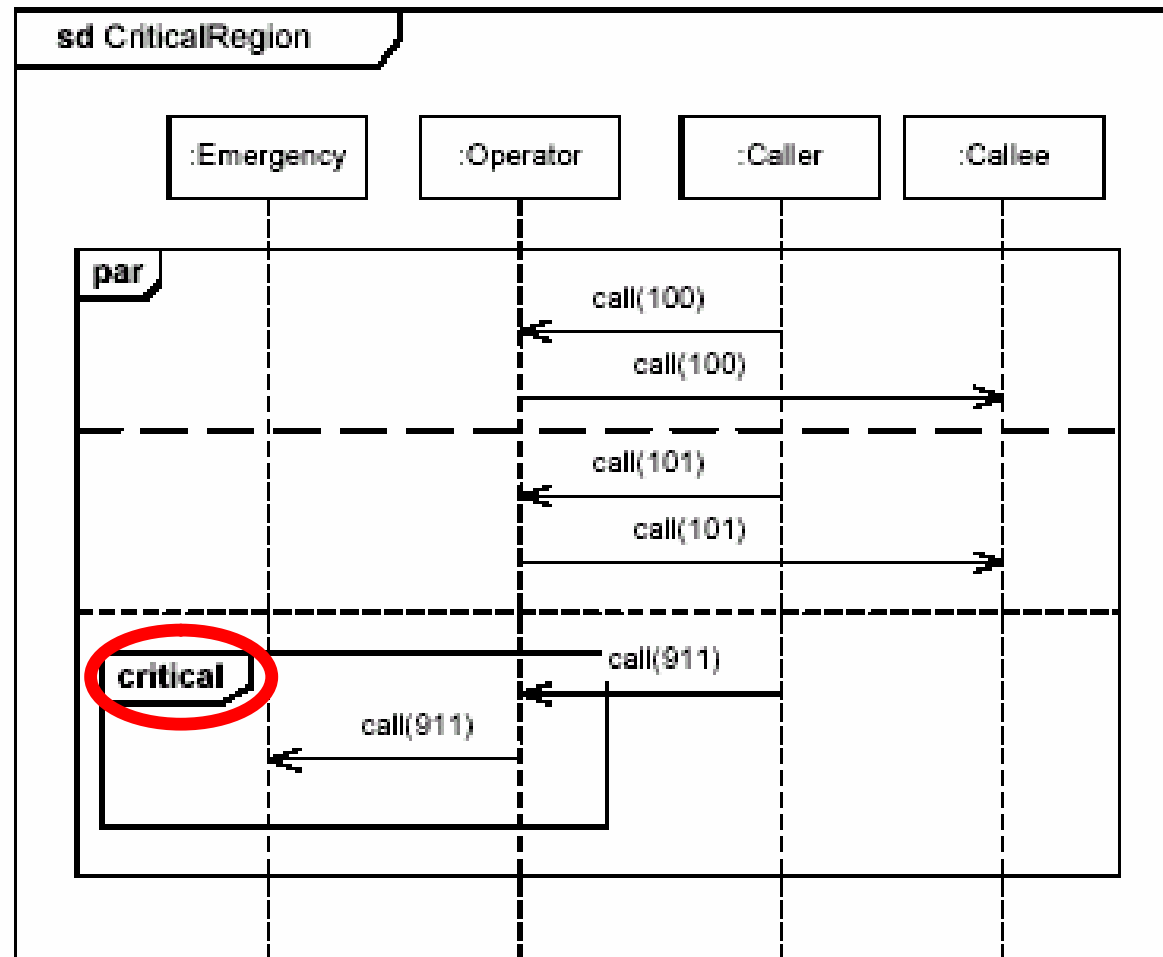


CF “Critical Region”

- » Operatore: **critical**
- » Significato:
 - > rappresenta una “regione critica”
 - > le tracce all’interno della regione critica **non possono essere interleaved**
 - > la regione e’ trattata **atomicamente**



Critical Region



CF “Ignore” and “Consider”

- » Operatore: **ignore**
- » Significato:
 - > se “m” e’ **ignored**, allora puo’ apparire ovunque
 - Se abbiamo [a1,a2,a3] e “m” ignored, le tracce [a1,m,a2,a3], [a1,a2,m,a3] sono entrambe valide
- » Operatore: **consider**
- » Significato:
 - > opposto di ignore



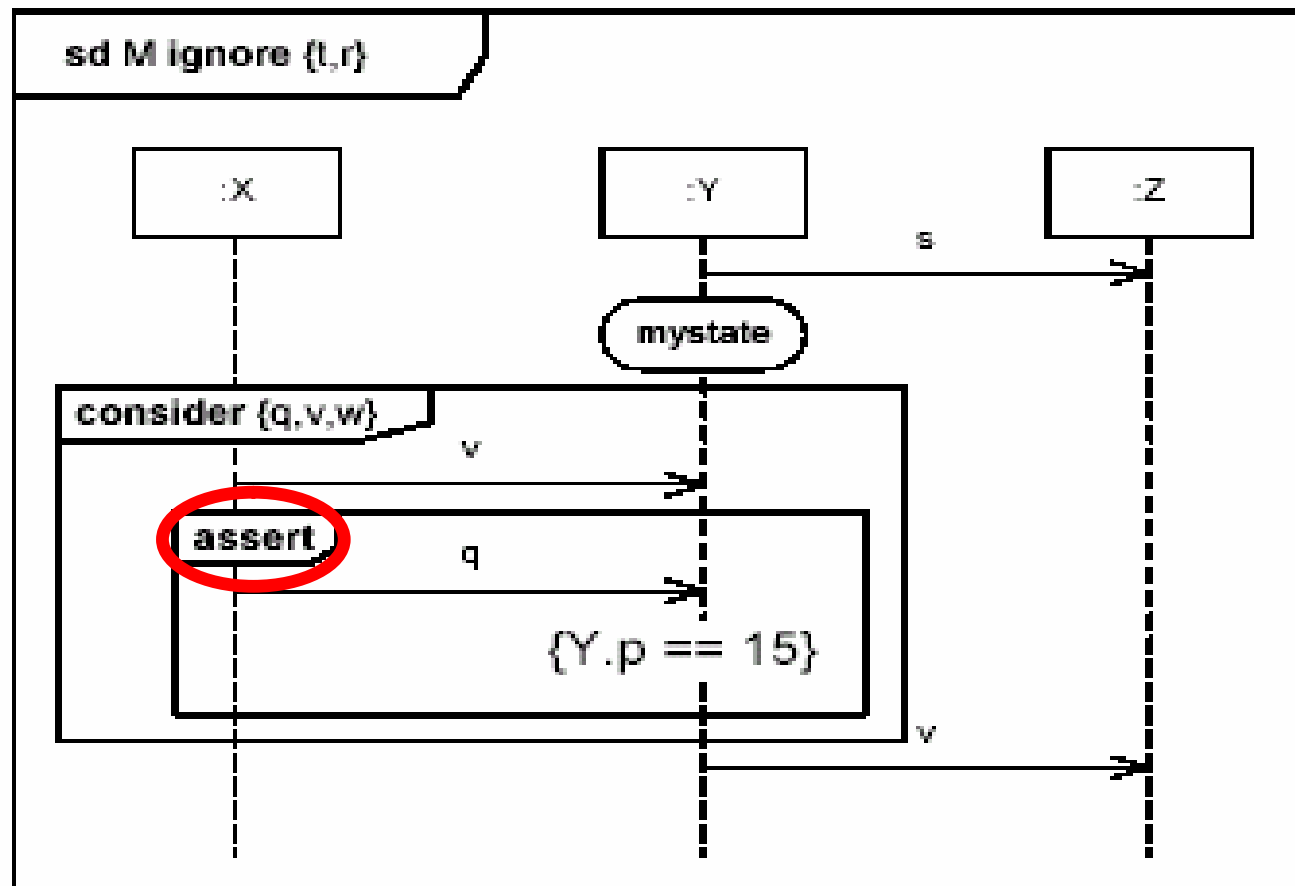
CF “Assertion”

- » Operatore: **assert**
- » Significato:
 - > rappresenta una asserzione
 - > spesso combinato con Ignore e Consider
 - > Nel sequence in slide successiva, asseriamo che “q” puo’ accadere solo dopo “v”



Consider e Assert

- “q” puo’ accadere solo dopo “v”
- Ogni messaggio diverso da {q,v,w} sara’ ignorato in fase di test
- L’occorrenza di “w” tra “v” e “q” rappresenterebbe un errore



CF “Loop”

- » Operatore: **loop**
- » Significato:
 - > il frammento deve essere ripetuto **piu' volte**
 - > una **guardia** o una **espressione booleana** esprimono il numero di ripetizioni

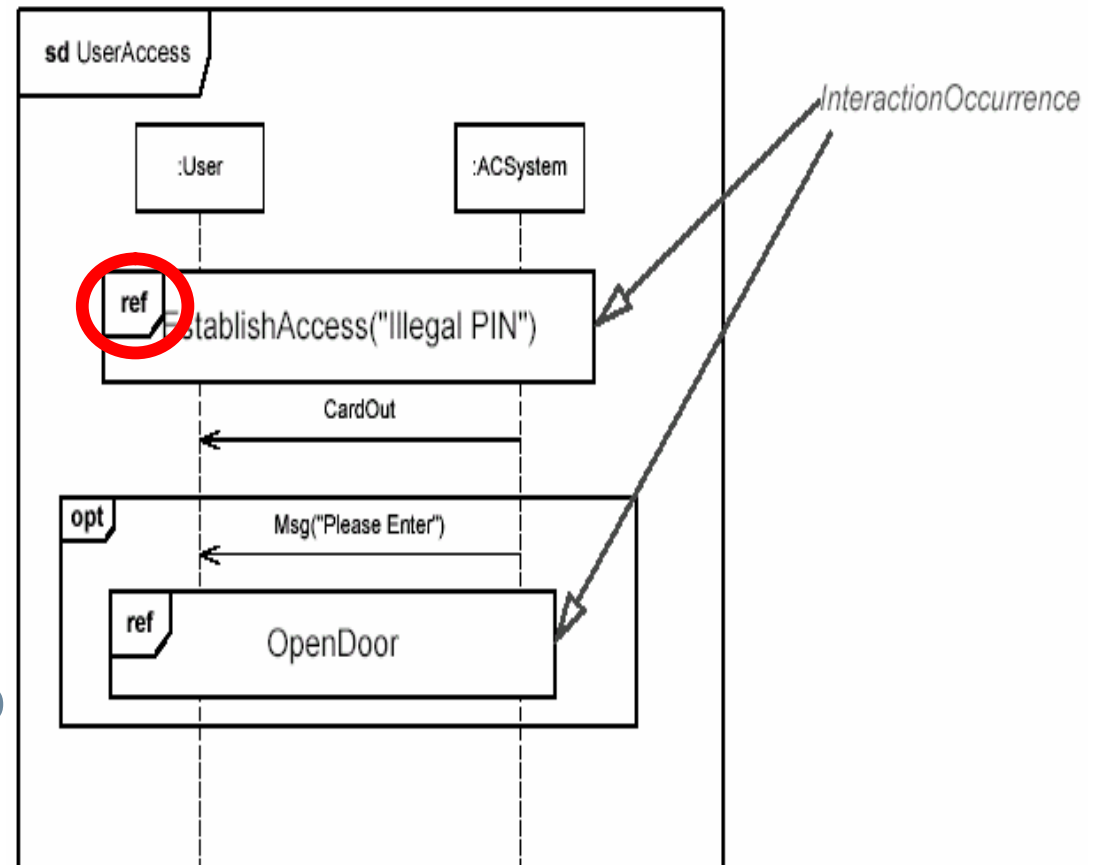


InteractionOccurrence

» Operatore: **ref**

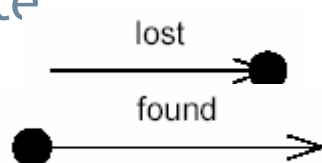
» Significato:

- > permette di richiamare, all'interno di un SD, altri SD
- > permette una maggior separation of concern, lasciando all'esterno del SD quella porzione utilizzata in piu' SD



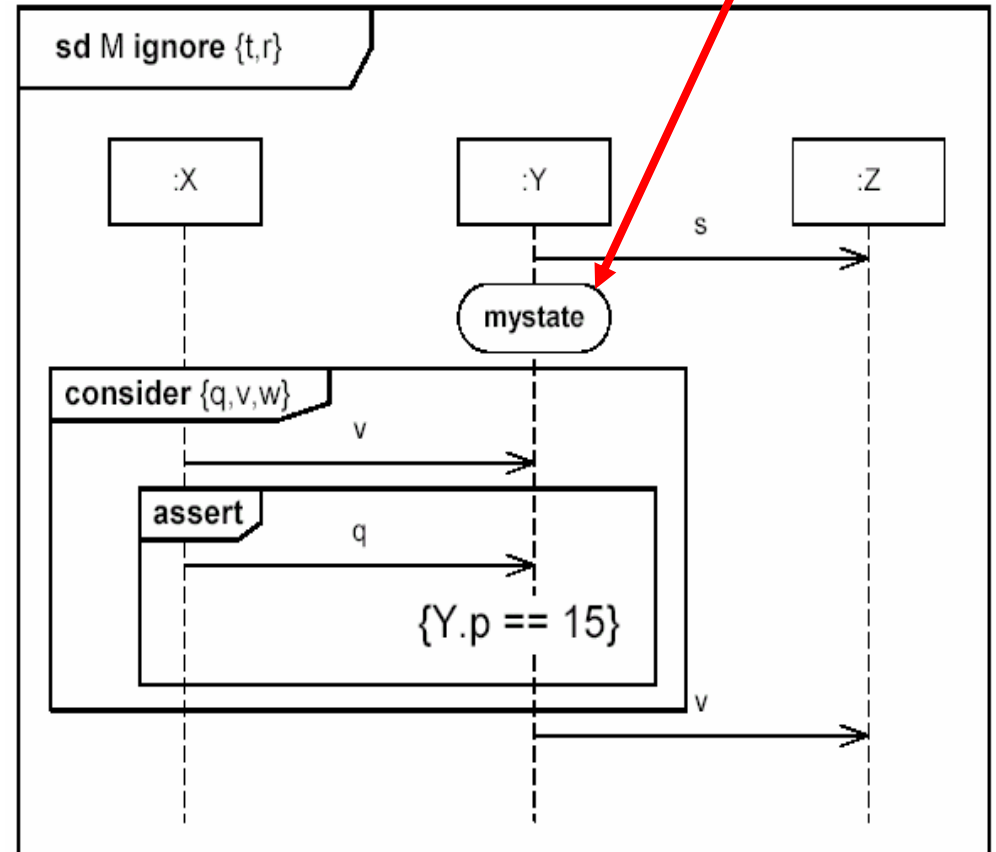
Messages

- » I messaggi rappresentano una particolare comunicazione tra Lifeline
- » Possono essere di diverso tipo:
 - > Complete
 - > Lost
 - > Found
 - > Unknown
- » Le metodologie di comunicazioni sono:
 - > Asincrona
 - > Sincrona
 - > Object creation

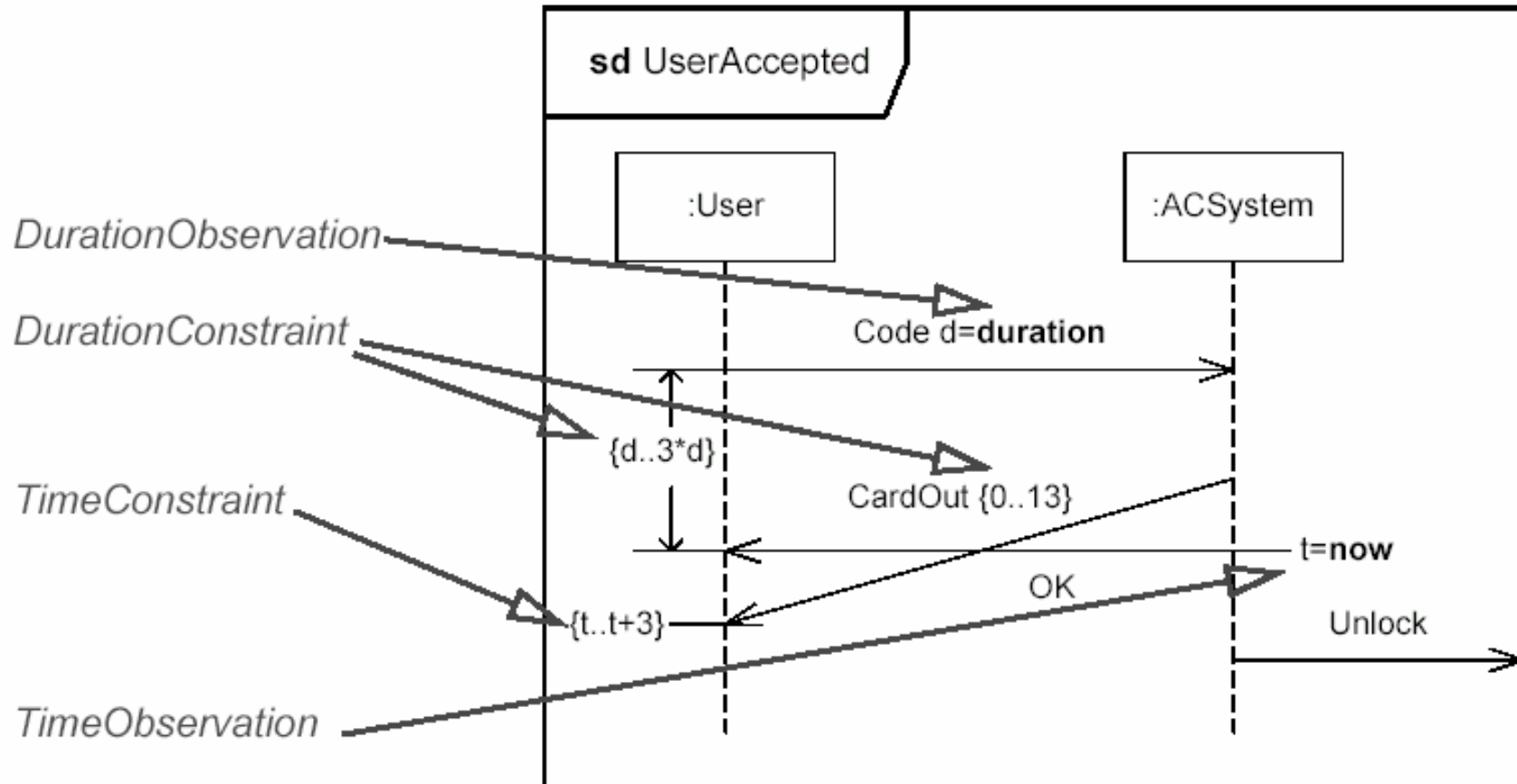


Stato

- » Ad un lifeline, puo' essere associato uno "stato"
- » Lo stato viene valutato:
 - > Se e' verificato, allora possiamo continuare ad eseguire il SD
 - > altrimenti, il SD non puo essere verificato.



SD con concetti di timing



Communication Diagram

- » Focalizza l'attenzione sia sulle *interazioni* che sulle *relazioni* tra gli oggetti che collaborano
- » Come un *sequence diagram* mostra le *interazioni*, ma il *sequence* si focalizza sul tempo (flusso temporale) mentre il **communication sullo spazio** (flusso di dati, quantità di messaggi scambiati tra le singole componenti)
- » **NOTA Importante:**
 - > Un communication diagram, in UML 2.0, NON e' piu' totalmente equivalente ad un sequence diagram.
 - I Combined Fragment ora presenti nei SD, per esempio, NON sono rappresentati nei communication diagram

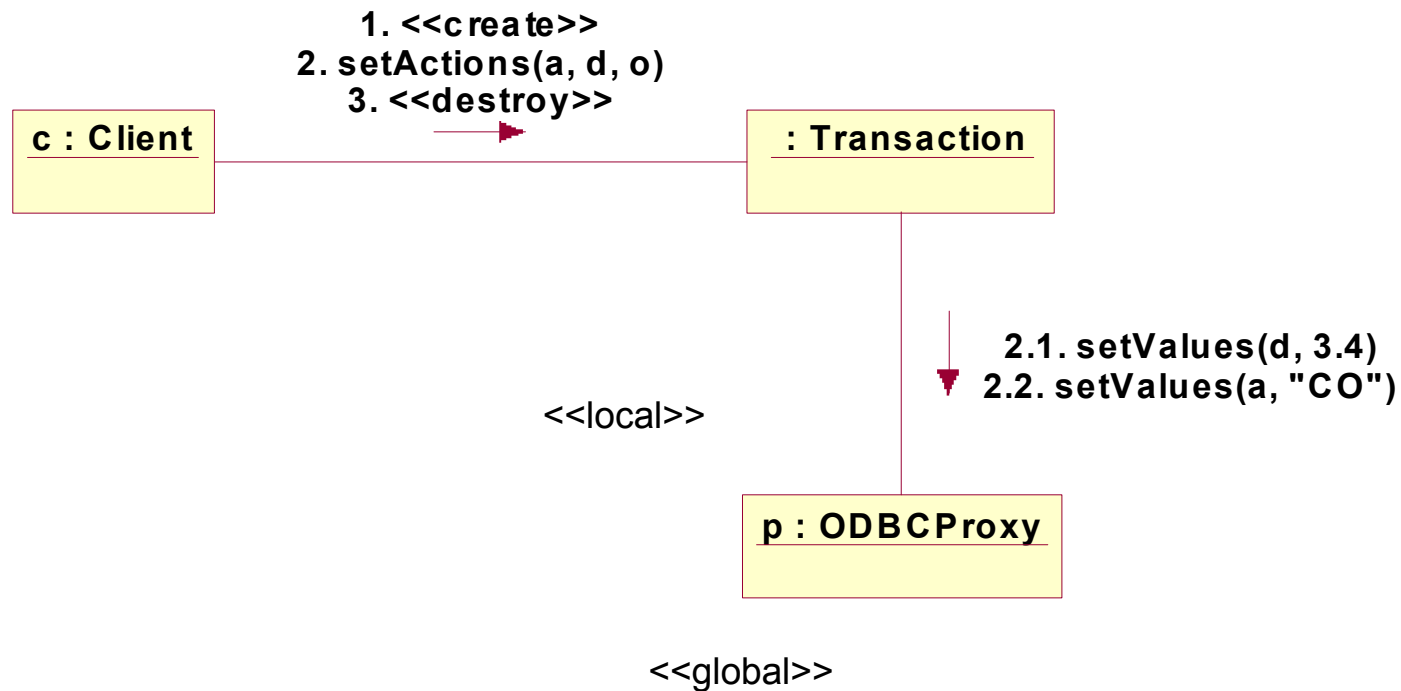


Communication Diagram

- » Graficamente rappresentato come un **grafo** dove i nodi sono gli **oggetti** e gli archi sono i **links**
- » **Oggetti**
 - > Disegnati con lo stesso simbolo bidimensionale che si usa per le classi, ma i loro nomi sono sottolineati
- » **Links**
 - > Sono istanze di un'associazione
 - > Disegnati tramite linee (*paths*)
 - > Sono adornati tramite **messaggi** che gli oggetti inviano e ricevono



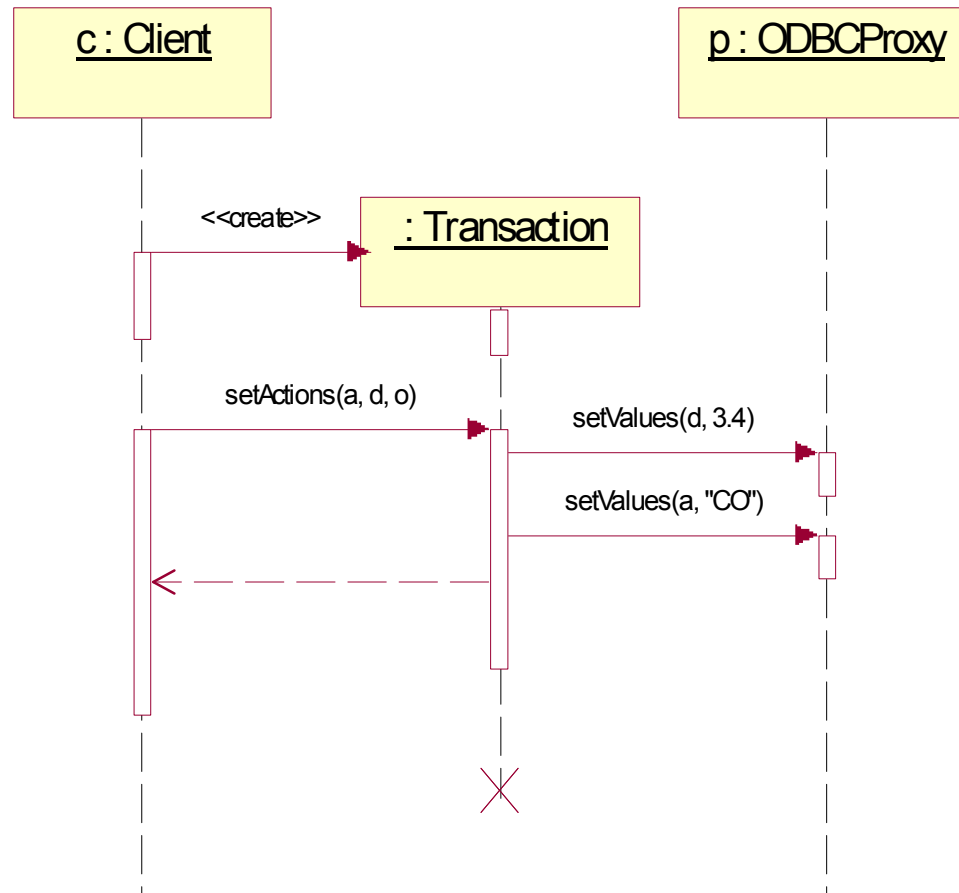
Communication Diagram



Trasformare tale Communication
in un Sequence



Sequence Diagram



Unified Modeling Language

- State Machine, Activity, Implementation and Deployment diagrams -

Henry Muccini
Università degli Studi dell'Aquila
muccini@di.univaq.it



Engineering | IgTechnology | Imola
Informatica | Maggioli Informatica |
Micron Technology | Neta | Nous
Informatica | ObjectWay SED |
TechnoLabs | Taiprora

Master in Web Technology

IV Edizione 2007/08

Dipartimento di Informatica

Università degli Studi dell'Aquila

Copyright Notice

- » Il materiale riportato in queste slide puo' essere riutilizzato, parziale o totalmente, a patto che le fonti e gli autori vengano citati

Henry Muccini



Sequence Diagram: riassunto

- » Per riassumere i diagrammi mostrati la lezione precedente, far vedere:
 - > Elementi costituenti
 - > Semantica
 - > Sintassi di integrazione

- » Far vedere anche le relazioni tra i diagrammi



Sommario

- » State machine diagram (**UML 2.0**)
 - > Stati
 - > Transizioni interne
 - > Stati composti
 - > Transizioni
 - > Synch states
 - > Protocol State Machine



State Machines

Pensiamo ad un sistema per la gestione delle telefonate...

Pensiamo a come funziona un telefonino...



- » Viene utilizzato per modellare **l'aspetto dinamico** di un sistema
- » In particolare
 - > Modella il comportamento di oggetti reattivi, cioè oggetti il cui comportamento è caratterizzato dalla risposta ad eventi generati dal di fuori del suo contesto
 - > Modella il comportamento di oggetti che possono trovarsi in vari stati durante il loro ciclo di vita
 - > Possono modellare anche il comportamento di use-case, attori, sottosistemi, operazioni o metodo



- » Derivano semanticamente e sintatticamente dagli state-chart di David Harel adattandoli al mondo Object-Oriented
- » Implementano anche alcuni aspetti delle macchine di Moore e Mealy

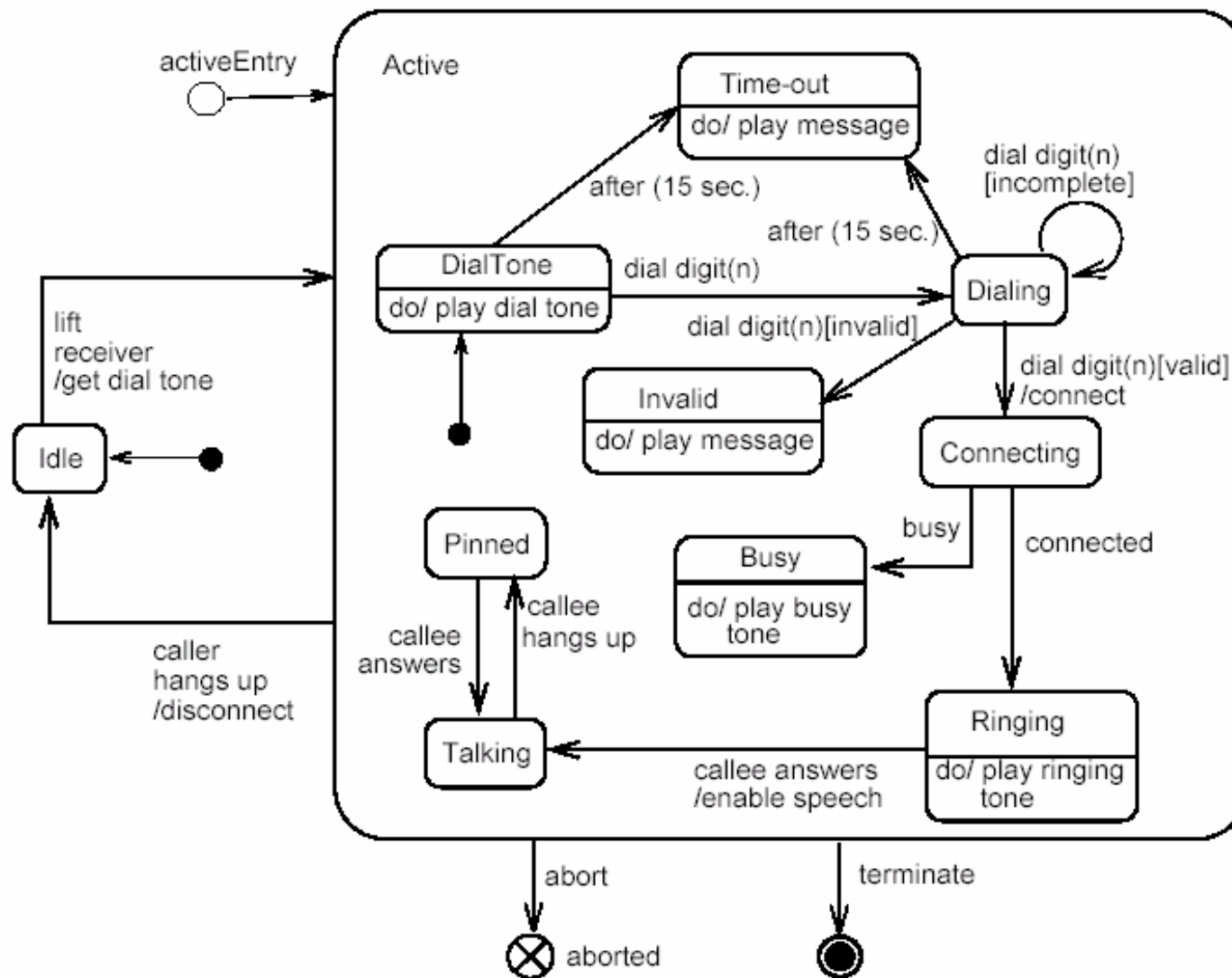


State machine diagram

- » Mostrano quindi una macchina a stati enfatizzando il **flusso di controllo da stato a stato**
- » **Macchina a stati** è un comportamento che specifica le **sequenze di stati** che ha un oggetto durante il suo ciclo di vita in risposta ad eventi. E' composto da
 - > Stato
 - > Attività/Azioni
 - > Eventi
 - > Transizioni
- » Graficamente è un grafo con vertici ed archi



State machine diagram



Stato

- » Rappresenta una **condizione** o una **situazione** (durante la vita di un oggetto) durante la quale delle condizioni vengono soddisfatte, delle azioni od eventi vengono eseguiti
- » Può essere **semplice** oppure **composto**
- » Esempio: **Caldaia** può trovarsi nei seguenti 4 stati
 - > Idle: in attesa del comando di inizio di riscaldamento della casa
 - > Activating: è attiva ma ancora non si raggiunge la temperatura
 - > Active: è attiva e ha raggiunto la temperatura
 - > ShuttingDown: non c'è più il gas e la temperatura sta scendendo

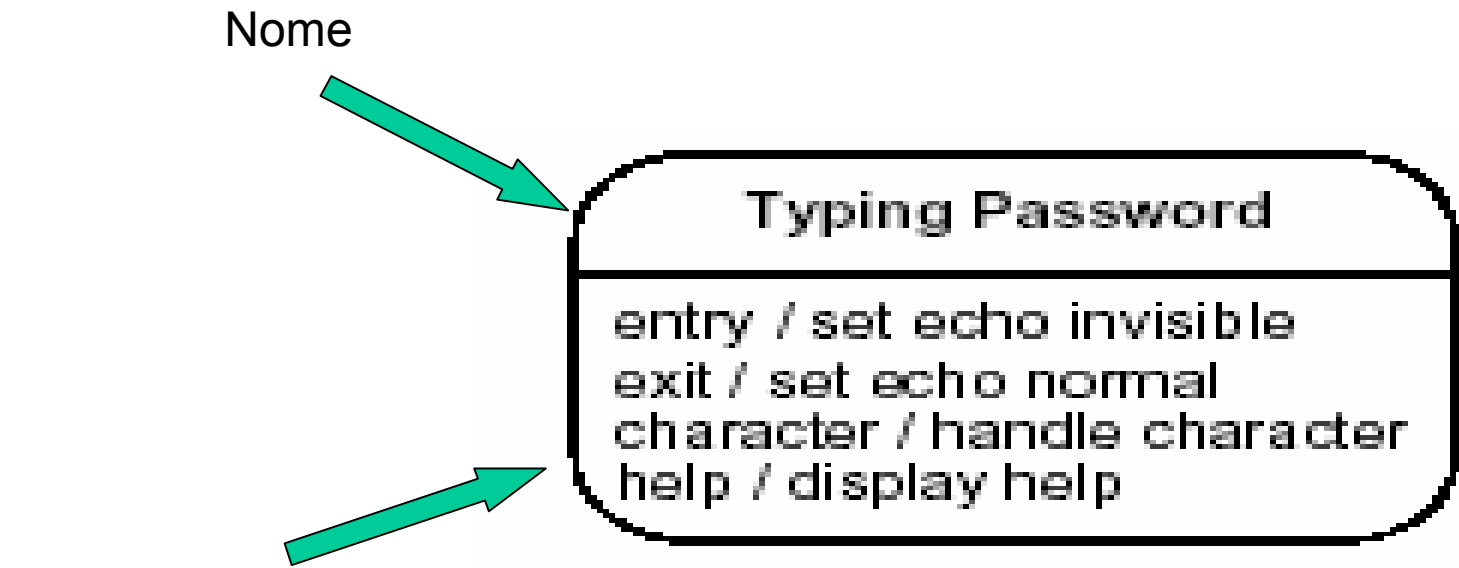


Stato

- » Rappresentato con un rettangolo con gli angoli smussati
- » E' composto da 2 *compartment*
 - > **Nome**
 - Una stringa testuale che distingue uno stato da un altro; se non ha nome è considerato anonimo
 - > **Transizioni interne**
 - Contiene una lista di **attività o azioni interne** che sono eseguite mentre l'elemento è all'interno dello stato. Quindi tali attività o azioni **non comportano il cambiamento di stato**
 - Sintassi: **[label] '/' action-expression**



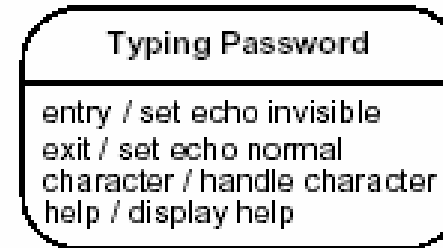
Stato



Transizioni interne



Transizioni interne



[label] '/' action-expression

» label

- > Identificano le circostanze sotto la quale l'azione specificata da action-expression viene invocata
- > Alcune label sono riservate (**entry, exit, do, include**)

» action-expression

- > **Attività:** Rappresentano operazioni interne eseguite mentre l'oggetto è in uno stato che possono richiedere tempo e sono interrompibili



Transizioni interne

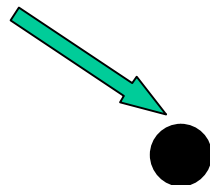
- » **entry/azione**: azione eseguita nel momento in cui si **entra** in uno stato
- » **exit/azione**: azione eseguita nel momento in cui si **esce** da uno stato
- » **do/attività**: identifica **un'attività in esecuzione** mentre l'oggetto è nello stato della transizione
- » **include/azione**: invocazione a una **submachine**. L'azione contiene il nome della submachine invocata



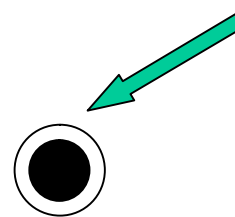
Stati iniziale e finale

- » Sono due tipi speciali di stati che identificano rispettivamente, il punto di partenza della macchina a stati, e la terminazione della macchina a stati

stato iniziale



stato finale

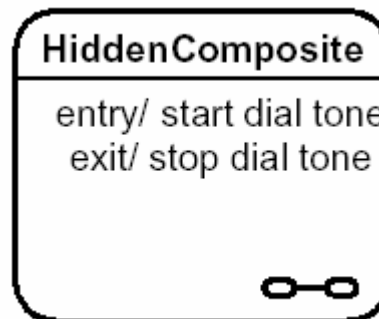
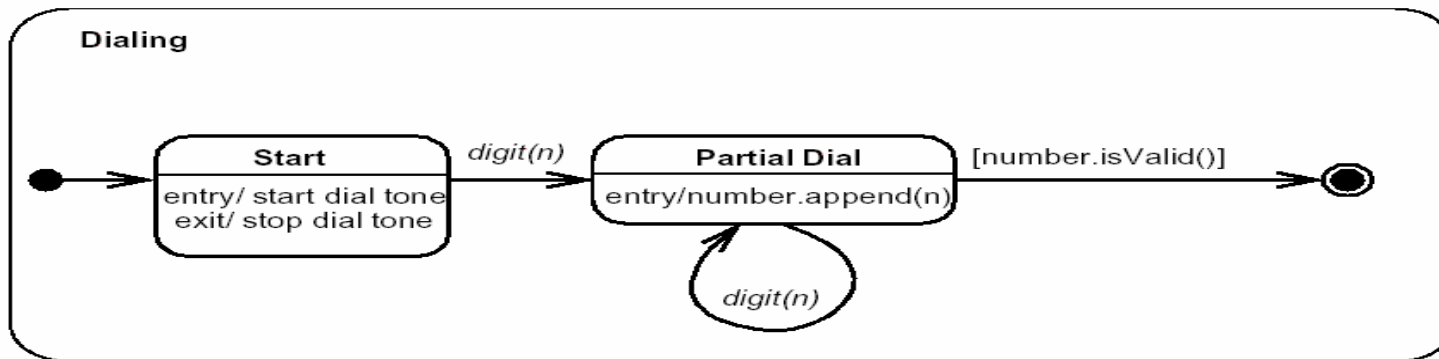


Stati composti

- » Uno stato che ha sotto-stati viene detto **stato composto**
- » Può essere **decomposto in**
 - > Due o più sottostati concorrenti (dette *regioni*)
 - > Sottostati sequenziali
- » Ogni sottostato può essere a sua volta uno stato composto



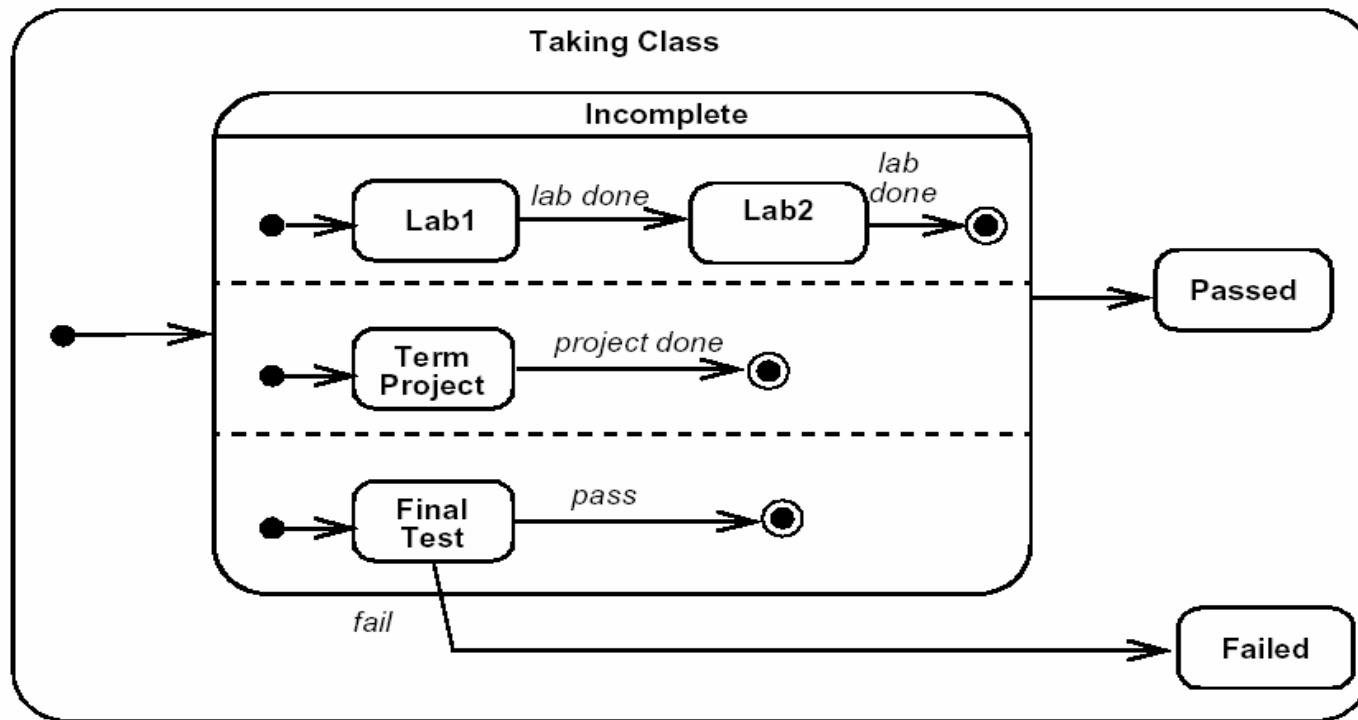
Sotto-Stati sequenziali



E' possibile nascondere il sotto-stato.

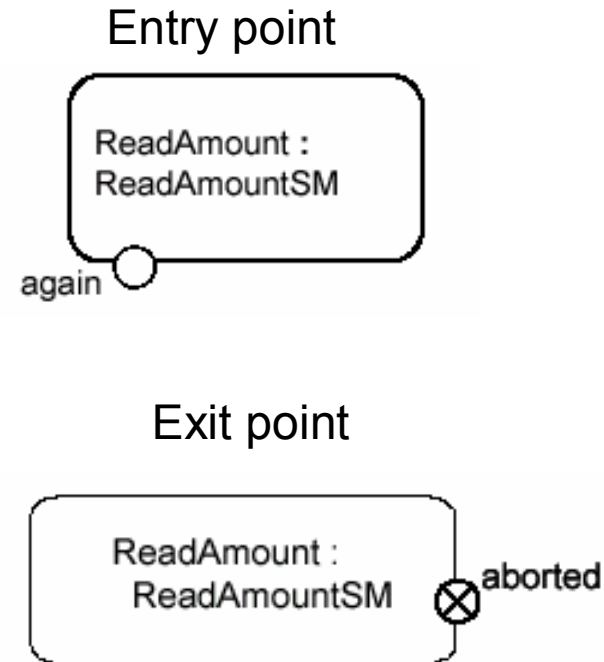


Sotto-Stati concorrenti

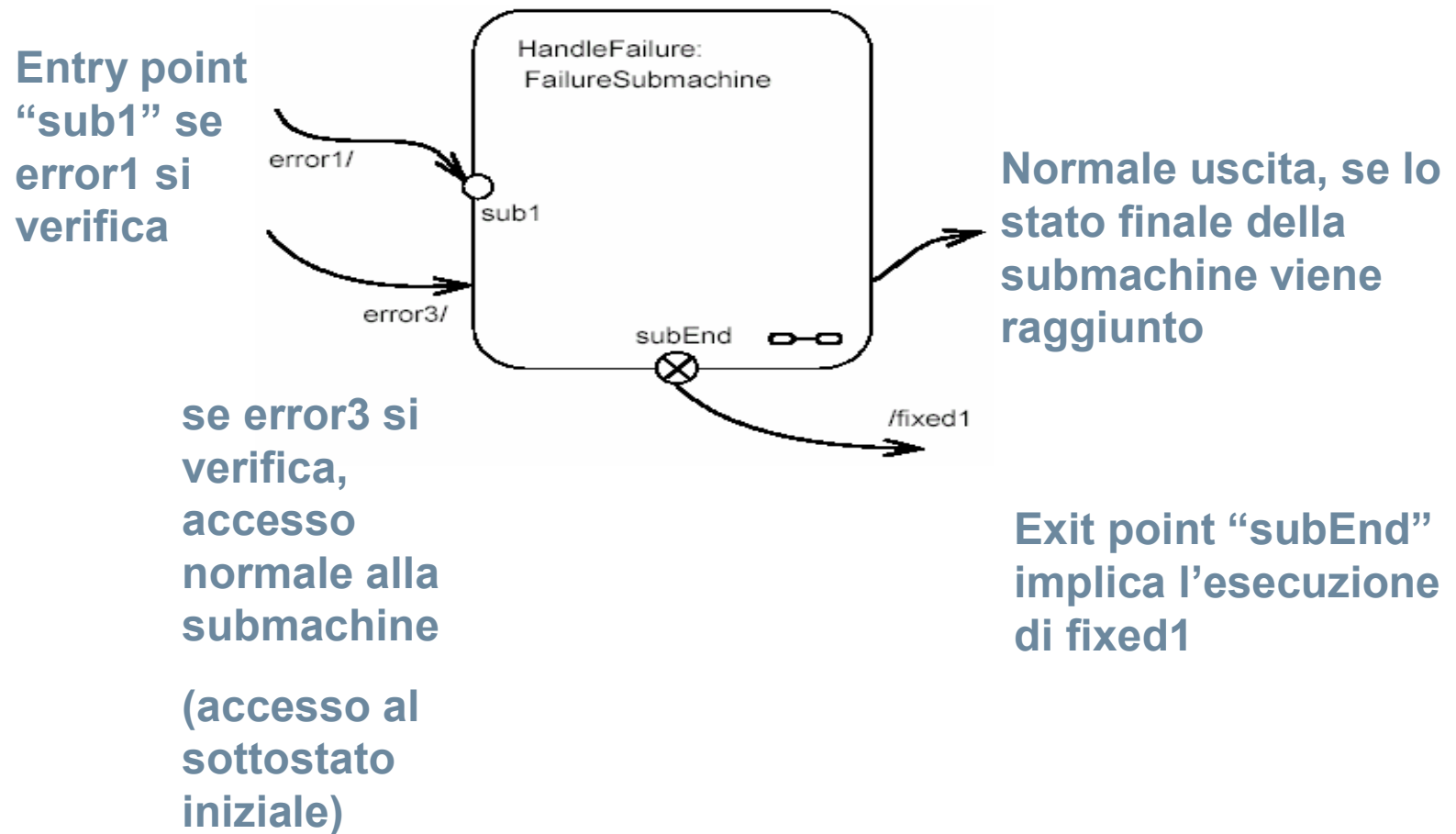


Submachine state

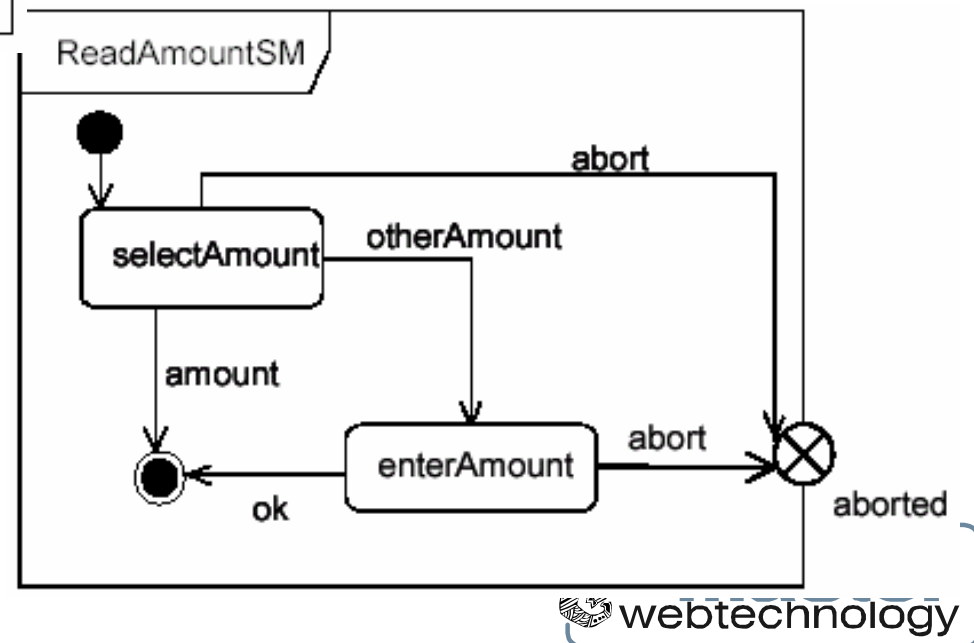
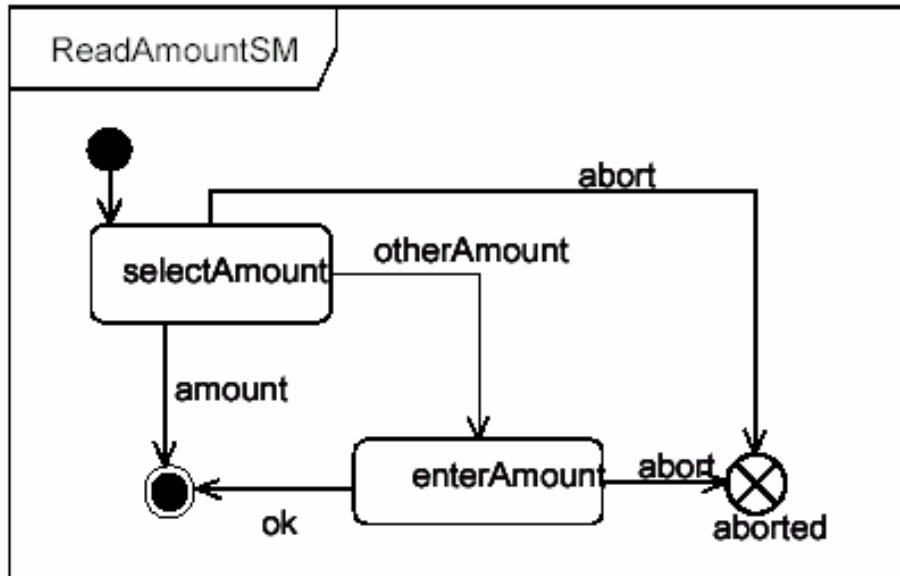
- » Sono semanticamente equivalenti a stati composti
- » La differenza e' che **si accede ed esce da un submachine state tramite "entry" ed "exit" point**
- » Una submachine composta puo' essere acceduta tramite entry points oppure tramite il suo stato iniziale (il default)

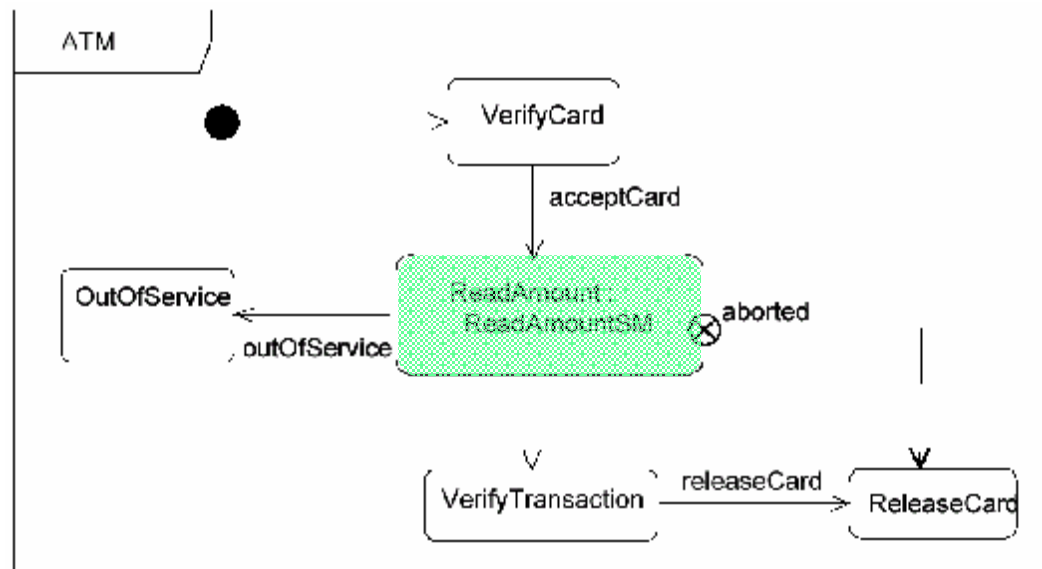
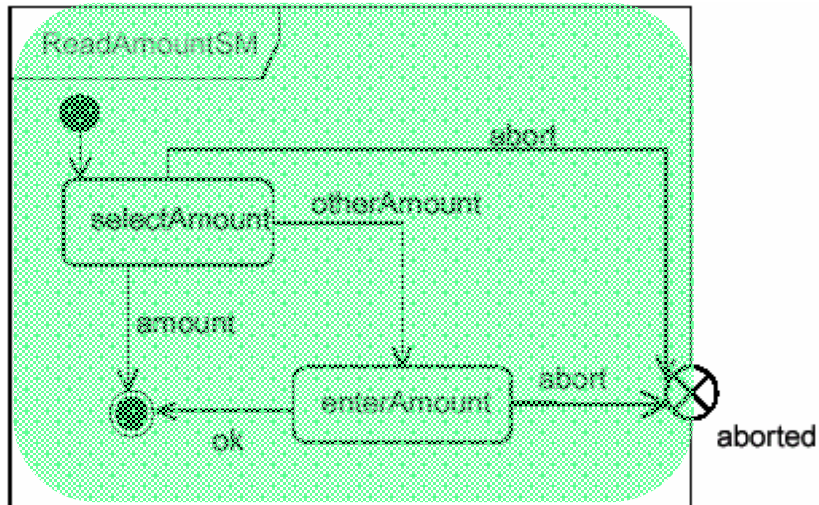


Submachine state



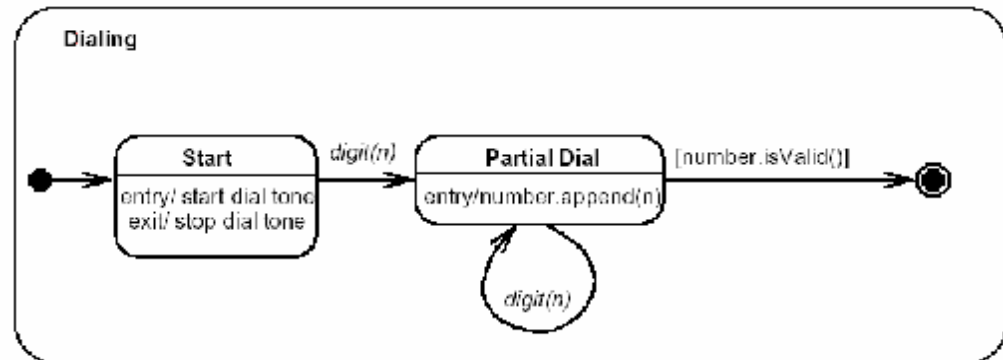
Submachine state



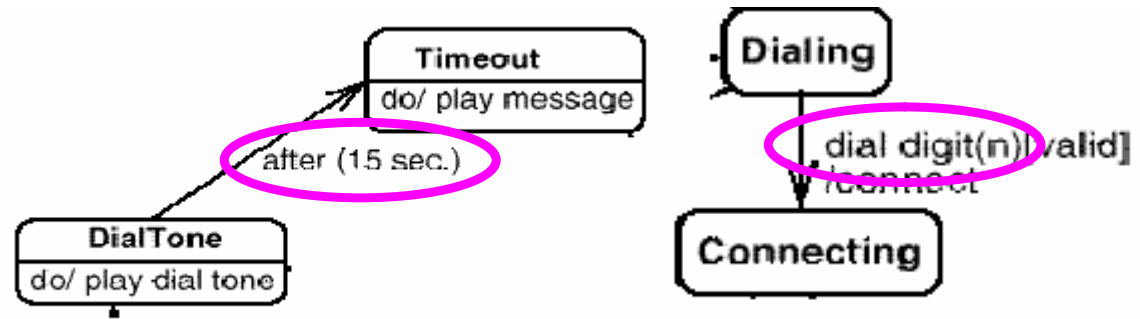


Transizioni

- » E' una relazione tra due stati la quale indica che **un oggetto nel primo stato eseguirà una serie di azioni ed entrerà nel secondo stato al verificarsi di un determinato evento** quando una condizione viene soddisfatta
- » Composta da 5 parti
 - > Stato sorgente
 - > Evento scatenante (trigger)
 - > Condizione di guardia
 - > Azione
 - > Stato destinatario
- » Rappresentata mediante una freccia diretta continua dallo stato sorgente a quello destinatario con associata un'etichetta
 - > *event-signature* '[' guard-condition ']' '/' *action-expression*
 - > *event-signature*= *event-name* '(' *comma-separated-parameter-list* ')'



Transizioni



event-signature '[' *guard-condition* ']' '/' *action-expression*

» Evento

> Verificarsi di un qualcosa degno di nota che può innescare una transizione di stato

> Sintassi

event-signature = *event-name* '(' *comma-separated-parameter-list* ')'

> Quattro tipi di eventi

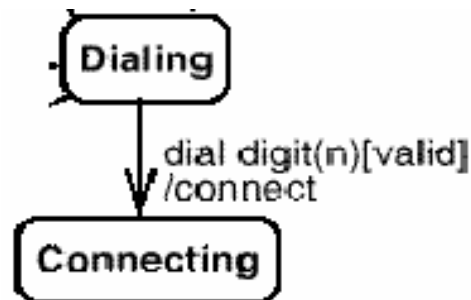
- **Change event:** L'evento si verifica quando il valore di una determinata condizione cambia da false a true
 - (esempio "when (date = Jan. 1, 2004)")
- **Time event:** Il passare un determinato periodo di tempo
 - (esempio "after (10 seconds)")
- **Segnale:** invio di un evento in modo asincrono
- **Calls:** ricezione di una chiamata di un'operazione (chiamata



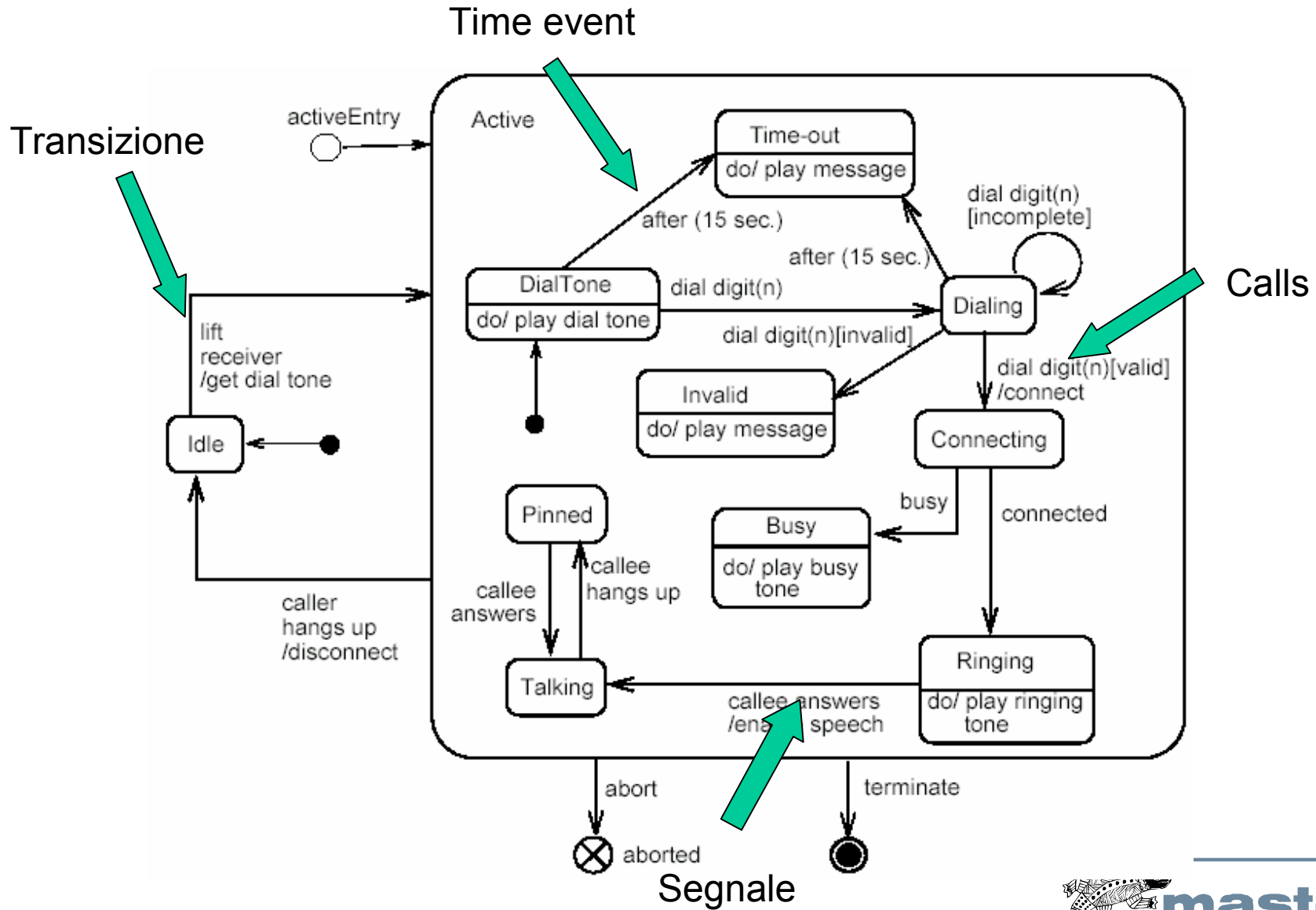
Transizioni

event-signature '[' *guard-condition* ']' '/' *action-expression*

- » Condizione di guardia (**[guard-condition]**)
 - > Espressione booleana che viene valutata alla ricezione dell'evento. Se è **true** la transizione viene attivata altrimenti non avviene la transizione di stato e l'evento viene perso
- » Azione (**action-expression**)
 - > Computazione eseguibile atomica che può agire sull'oggetto della macchina a stati oppure su altri oggetti **inviando segnali o eventi**



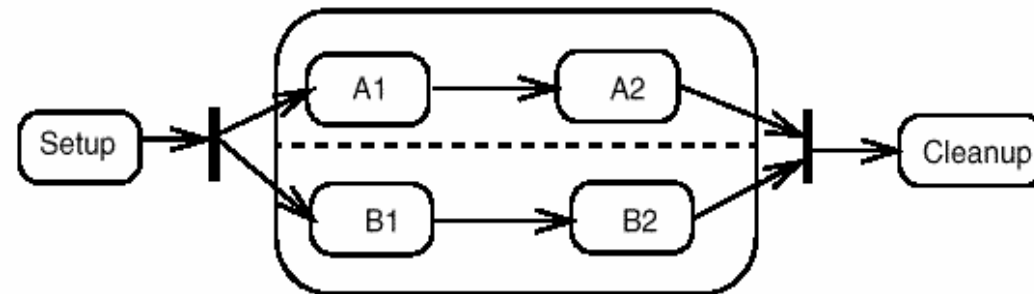
Transizioni



Transizioni da e verso stati concorrenti

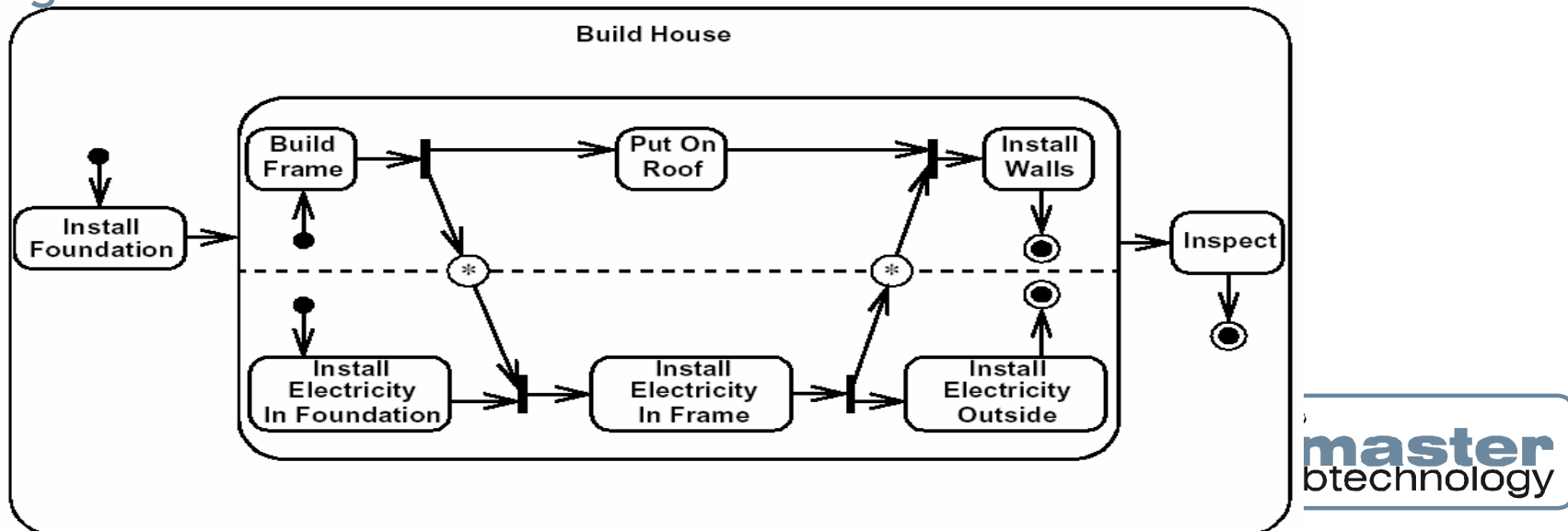
Una transizione concorrente può avere multipli stati sorgenti e target

- » Rappresenta una **sincronizzazione** e/o una divisione del controllo in thread concorrenti
- » Una transizione concorrente è “enabled” quando tutti gli stati sorgenti sono stati occupati
- » Viene rappresentata con una barra verticale, chiamata barra di sincronizzazione che rappresenta una **sincronizzazione, fork**

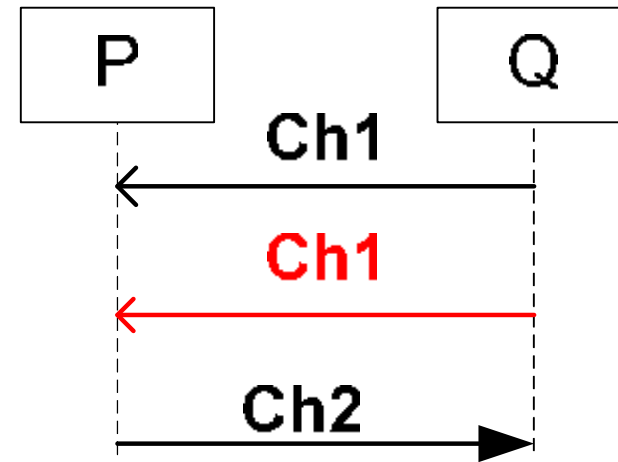
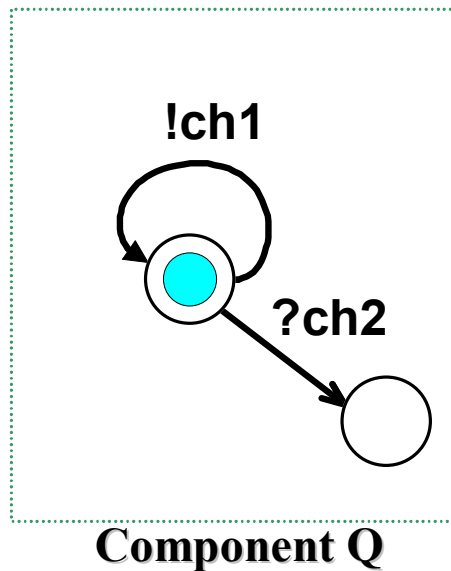
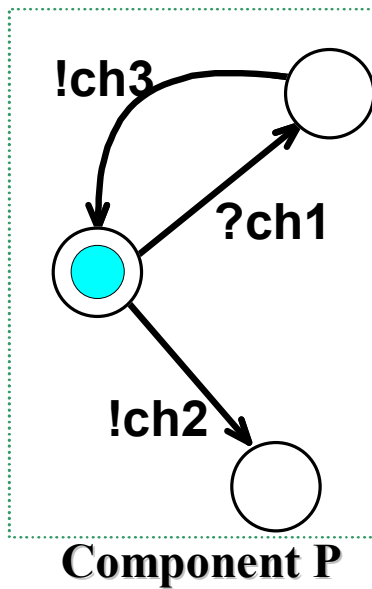


Synch States

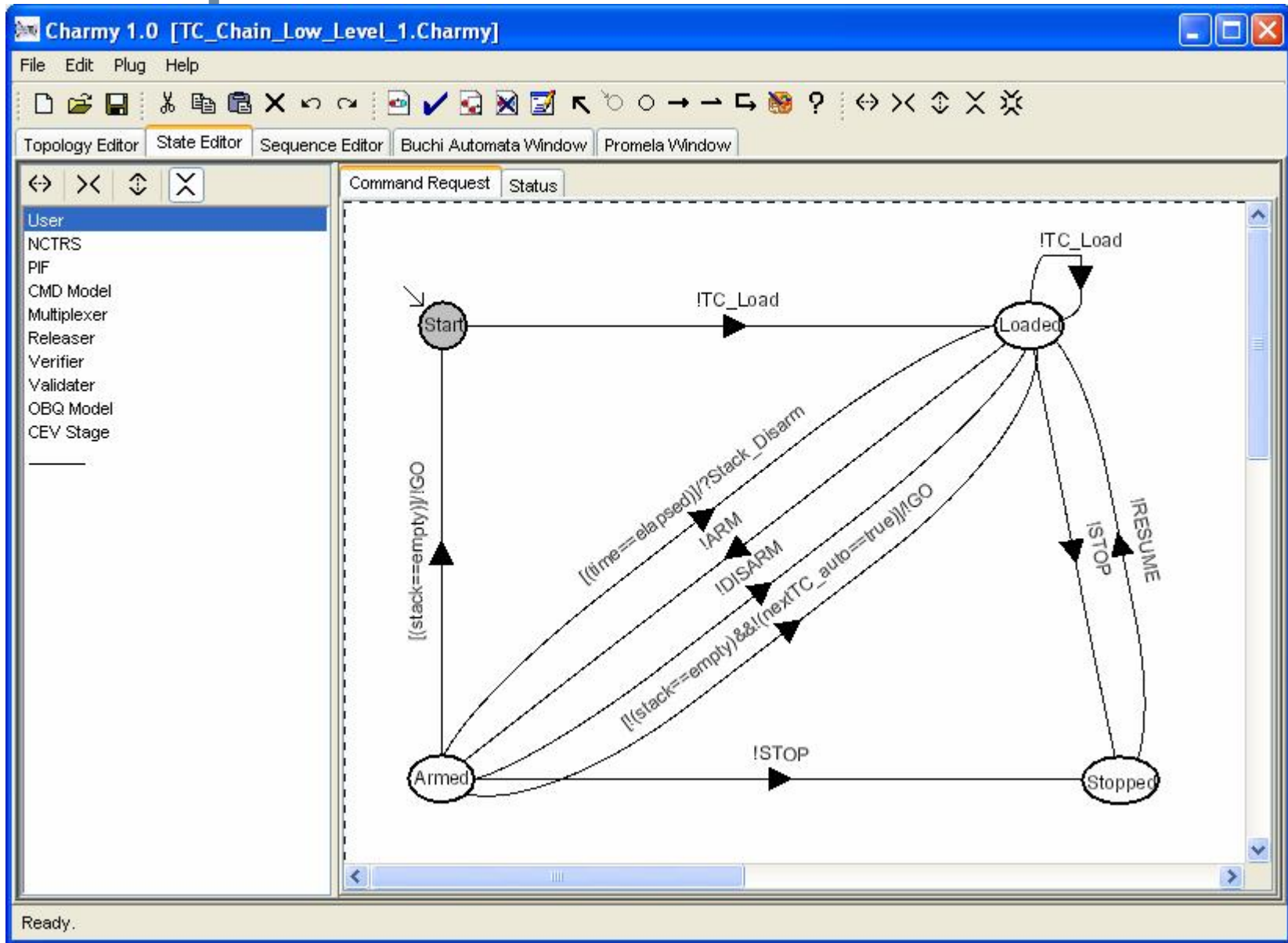
- » Un Synch State viene utilizzato per sincronizzare regioni concorrenti di una macchina a stati
- » Viene utilizzato in congiunzione con forks e join per assicurare che una regione esca da un particolare stato prima che un'altra regione entri in un particolare stato
- » Viene rappresentato con un cerchio al cui interno c'e' un '*'
- » Quando e' possibile vanno rappresentati sul confine tra regioni concorrenti



Relazioni tra State e Sequence D.



Esempio d'uso



Esercizio da fare in classe

- » Fare lo State Diagram di come funziona un telefonino
- » **Fare lo State Diagram della Chat**
- » Fare lo State Diagram della Bachecca:
 - > La bacheca parte da uno stato di attesa
 - > Può ricevere:
 - Richiesta di registrazione (A)
 - Richiesta di autenticazione (B)
 - Richiesta di lettura (C)
 - > (A) Se ottiene una richiesta di registrazione, verifica i dati inseriti e che l'utente non sia stato allontanato dalla bacheca. Se tutto ok, registra utente e torna in attesa.
 - > (B) Se ottiene una richiesta di autenticazione, verifica i dati utenti e consente/nega l'accesso in lettura/scrittura.
 - Se consente l'accesso, periodicamente monitorizza i messaggi. Permette ad un utente di scrivere e leggere.
 - > (C) Se ottiene una richiesta di lettura, fa' accedere l'utente alla bacheca.

