La specifica dei sistemi interattivi attraverso gli statechart

Progettazione dell'Interazione con l'Utente
Interazione Uomo-Macchina
a.a. 2010-2011

Docente: M. F. Costabile

Il controllo nel software tradizionale e nel software interattivo

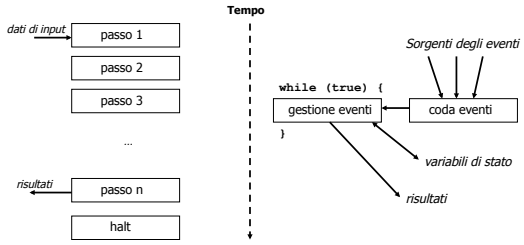
- ***Software tradizionale***: il controllo viene passato da una procedura all'altra
- I parametri del programma possono fare in modo che il flusso di controllo cambi da un'esecuzione all'altra
- Il percorso di esecuzione è definito dallo sviluppatore
- Il ***software interattivo è event-driven***: ogni oggetto (entità virtuale) del sistema interattivo risponde a degli eventi esterni
- Il flusso di controllo è determinato dall'utente

Event-Driven Programming

- Paradigma imperativo:
  - Programma controlla la sequenza dei passi a run-time
  - I dati di input hanno un ruolo passivo nel regolare come i passi sono portati avanti
  - Il programma termina quando i passi sono completati
- Paradigma *event-driven:*
  - Sequenza di controllo non predetta nel programma
  - I dati di input (eventi) determinano la sequenza di controllo
  - L'ordine degli eventi non è predicibile
  - Il programma non termina

[Tucker, Noonan, 2002 *Programming Languages*]

Imperative vs. Event-Driven



I gestori degli eventi

- Gli ***event handler*** sono procedure associate
  - agli oggetti specifici di interfaccia
  - a specifici eventi
- Il codice di un event handler viene eseguito in risposta ad uno specifico evento fornito allo specifico oggetto
- Es: l'utente clicca su un bottone e l'event handler *when_mouse_clicked* associato al bottone viene eseguito
- Ogni oggetto dell'applicazione riconoscerà e gestirà un certo numero di eventi (es. *on_mouse_over, on_mouse_up)*
- Il progettista decide quali event handlers contengono codice affinchè venga compiuta una certa computazione in risposta ad un certo evento

Controllare l'interazione

- L'ordine con cui gli eventi accadono non può essere previsto dallo sviluppatore
- L'esecuzione della applicazione può essere diversa ogni volta
- **Come evitare errori**?
- Occorre fare in modo che l'utente non possa generare eventi che causano un errore nell'applicazione
- Es. word processor: l'utente non può dare comandi di formattazione prima di aver aperto un documento
- *L'**utente** determina il flusso di controllo in un'applicazione fornendo eventi, ma lo **sviluppatore** dell'applicazione è responsabile nella determinazione di quali eventi l'utente può fornire in ogni istante e del loro trattamento*

Dal paradigma event-action al paradigma event-state-action

Obiettivo: usare 'stati' per definire il contesto nel quale occorrono gli eventi

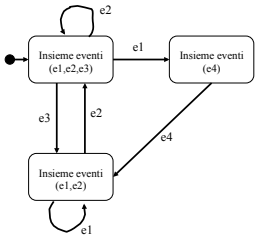
Nell'event-action:

- L'utente *fornisce* un evento ad un certo oggetto
- L'oggetto esegue alcune computazioni in risposta (che possibilmente visualizza progressivamente)
- Eseguite le azioni, l'oggetto si ferma ed attende la prossima azione

Ma l'evento da solo può determinare il comportamento dell'oggetto?

Il paradigma event-state-action

- Quando un utente interagisce con un'interfaccia, questa "si muove" da un insieme di possibili eventi a un altro
- Gli eventi che l'utente genera fanno sì che il software "si muova" da un insieme di possibili eventi a un altro
- In altre parole, l'interfaccia "si muove" da uno stato a un altro stato
- Lo ***stato*** definisce l'insieme dei possibili eventi che un utente può generare
- Gli stati definiscono il ***contesto*** in cui un evento può accadere



Non è un'idea nuova…

- L'approccio basato sugli stati non è un'idea nuova
- E' una via non intrapresa perché, anche per un'interfaccia piuttosto semplice, il numero degli stati e delle transizioni diventa grande
- Questo problema è dovuto alla notazione utilizzata: le macchine a stati finiti
- Le cose migliorano se si utilizzano gli statecharts (Harel, 1987)

Macchine a stati finiti

- Le macchine (automi) a stati finiti (o diagrammi di stato) consentono di descrivere il comportamento macchine sequenziali
- **I** = $\{i_1, i_2, \ldots, i_p\}$ insieme dei simboli di ingresso
- **U** = $\{u_1, u_2, \ldots, u_r\}$ insieme dei simboli di uscita
- **S** = $\{s_1, s_2, \ldots, s_n\}$ insieme degli stati
- **δ** funzione di transizione che specifica il nuovo stato in base allo stato corrente e all'ingresso
- **ω** funzione di uscita che specifica il valore di uscita in base allo stato corrente e all'ingresso
- Automa A = <I, U, S, δ, ω>
- La coppia δ-ω può essere descritta graficamente o con una tabella degli stati

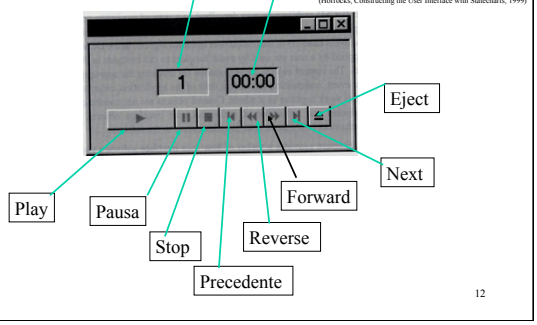
Macchine di Moore e di Mealy

- Le **macchine di Mealy** hanno, per ogni arco, un simbolo di entrata e uno di uscita
- Nelle **macchine di Moore** l'uscita è invece già codificata nel valore dello **stato** in cui si trova la macchina, ovvero la funzione di uscita dipende solo da *S* anziché da *I x S*
- E' possibile trasformare una macchina di Mealy in una di Moore e viceversa, solitamente quelle di Moore hanno più stati

Esempio: CD player

Due campi di testo traccia e tempo

(Horrocks, Constructing the User Interface with Statecharts, 1999)



Una prima specifica del sistema

difficile da interpretare e tradurre in programmi

1 There shall be eight buttons named: Play, Pause, Stop, Previous, Next, Forward, Reverse, and Eject.
2 There shall be two text fields for displaying data. They shall be named Time and Track.
3 When there is no disc in the CD player or when the CD player drawer is open, the system shall be in a state named <No CD Loaded>.
4 When the CD player drawer is closed and a CD is in the CD player, the system shall be in one of three possible states: <CD Stopped>, <CD Playing>, or <CD Paused>.
5 When in the <No CD Loaded> state, with the drawer open and a CD in the drawer, pressing the Eject button shall cause the drawer of the CD player to close and the system shall enter the <CD Stopped> state.
6 When in the <No CD Loaded> state, with the drawer open and no CD in the drawer, pressing the Eject button shall cause the drawer of the CD player to close and the system shall remain in the <No CD Loaded> state.
7 When in the <No CD Loaded> state, pressing the Play, Pause, Stop, Previous, Next, Forward and Reverse buttons shall have no effect.
8 The <CD Playing> state is entered from the <CD Stopped> state by a user clicking the Play button.
9 The <CD Stopped> state is entered from the <CD Playing> state by a user clicking the Stop button.
10 The <CD Paused> state is entered from the <CD Playing> state by a user clicking the Pause button.
11 The <CD Playing> state is entered from the <CD Paused> state by a user clicking the Pause button or the Play button.
12 When in the <CD Stopped> state, the <Time> field shall display 00:00 and the <Track> field shall display [1].
13 When in the <CD Stopped> state, the Pause and Stop buttons shall be disabled.
14 When in the <CD Playing> state, the Play button shall be disabled.
15 When in the <CD Stopped>, <CD Playing> or <CD Paused> states, clicking the Next button when the current track is not the last track on the CD, will cause the CD player to move to the next track, the <Time> field will display 00:00 and the <Track> field shall display the track number.
16 When in the <CD Stopped>, <CD Playing>, or <CD Paused> states, clicking the Next button when the current track is the last track on the CD, will cause the CD player to move to the first track, the <Time> field will display 00:00, the <Track> field shall display [1] and the <CD Stopped> state will be entered.
17 When in the <CD Stopped>, <CD Playing> or <CD Paused> states, clicking the Previous button when the current track is not the first track on the CD, will cause the CD player to move to the previous track, the <Time> field will display 00:00 and the <Track> field shall display the track number.
18 When in the <CD Stopped>, <CD Playing> or <CD Paused> states, clicking the Previous button when the current track is the first track on the CD, will cause the CD player to move to the start of the first track, the <Time> field will display 00:00 and the <Track> field shall display [1].
cont'd

Figure 5.2 A natural language specification of a CD player application.

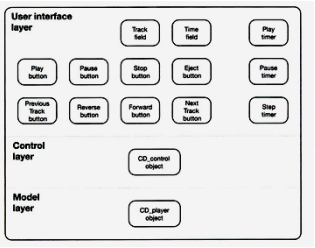
19 When in the <CD Stopped>, <CD Playing> or <CD Paused> states, clicking the Forward button down will cause the CD to stop playing and the CD to step forwards through the CD in one-second intervals, each step will take no more than 0.1 seconds. For each step, the <Time> field will display the current track time and the <Track> field will display the current track number. The application will stop stepping through the CD when the user stops holding down the Forward button with the mouse pointer, or the end of the CD is reached. If the end of the CD is reached, the <CD Stopped> state will be entered and the <Time> field will display 00:00 and the <Track> field shall display [1].
20 When in the <CD Stopped>, <CD Playing> or <CD Paused> states, clicking the Reverse button down will cause the CD to stop playing and the CD to step backwards through the CD in one-second intervals, each step will take no more than 0.1 seconds. For each step, the <Time> field will display the current track time and the <Track> field will display the current track number. The application will stop stepping through the CD when the user stops holding down the Reverse button with the mouse pointer, or the start of the CD is reached.
21 When in the <CD Playing> state, the <Time> field shall be updated every second with the elapsed playing time of the current track and the <Track> field shall display the current track number.
22 When in the <CD Paused> state, the values in the Time and Track fields will be displayed initially and then after one second they will be hidden. After a further second they will be displayed again. This displaying and hiding cycle will continue while the system is in the <CD Paused> state.
23 When in the <No CD Loaded>, <CD Stopped>, <CD Playing> or <CD Paused> states, the balloon help for the buttons shall be as follows: Stop button = Stop, Previous button = Previous Track, Next button = Next Track, Forwards button = Step Forward, Reverse button = Step Backwards.
24 When in the <No CD Loaded> state and the CD player door is open, the balloon help for the buttons shall be as follows: Play button = Play, Pause button = Pause, and Eject button = Close.
25 When in the <No CD Loaded> state and the CD player door is closed, the balloon help for the buttons shall be as follows: Play button = Play, Pause button = Pause, and Eject button = Eject.
26 When in the <CD Stopped> or <CD Playing> state, the balloon help for the buttons shall be as follows: Play button = Play, Pause button = Pause, and Eject button = Eject.
27 When in the <CD Paused> state, the balloon help for the buttons shall be as follows: Play button = Resume, Pause button = Resume, and Eject button = Eject.

Figure 5.2 (Cont'd) A natural language specification of a CD player application.

(Horrocks, Constructing the User Interface with Statecharts, 1999)

Verso una specifica più precisa

Esempio del CD



I problemi di specifica:

Lo strato di *modello* si specifica come programma

Lo strato di *controllo* con i metodi che stiamo vedendo

Lo strato di *interfaccia* con altri metodi

(Horrocks, Constructing the User Interface with Statecharts, 1999)

Lo strato di modello

Un oggetto che mette a disposizione dei metodi che sono chiamati dal controllo

CD_player public methods:

Open_drawer;
Causes the CD player drawer to open.

Close_drawer() return boolean;
Causes the CD player drawer to close. When the drawer has been closed, the function will return the value true.

Stop;
Causes the CD player to stop and the current track and time to be lost.

Play(Track_no,Time);
Starts playing the CD at the specified track number and time (in seconds) within that track.
If the track number and time are not valid for the CD then the CD will not start to play.
When the end of the CD is reached, the CD player will stop playing the CD.
If the CD player has been paused, calling the play procedure will cause the CD to resume playing at the point it was paused at.

Pause;
Causes the CD currently playing to be paused. The track number and time at which the CD is paused will be remembered.

cd_loaded () return boolean;
Returns True if a CD is in the machine, returns false otherwise.

current_track () return number;
Returns the current track number. Returns 0 if the CD is not playing or there is no CD in the machine.

current_time () return number;
Returns the current elapsed time of the current track. Returns 0 if the CD is not playing or there is no CD in the machine.

previous_track;
Causes the CD player to step backwards one track from the current track.
If the current track is the first track then calling this procedure will cause the CD player to move to the last track. The current time will be set to 0 when this procedure is called.
If the CD is playing when this function is called, then it will continue to play after the step has occurred.

next_track;
Causes the CD player to step forwards one track from the current track.
If the current track is the last track on the CD then calling this procedure will cause the CD player to move to the first track. The current time will be set to 0 when this procedure is called.
If the CD is playing when this function is called, then it will continue to play after the step has occurred.

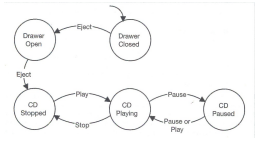
last_track () return boolean;
Returns True if the current track is the last track.

Forward_one_sec;
Causes the CD player to move forwards one second from the current track and time.
If the end of the CD is reached then the CD player will move to the start of the first track.
If the CD is playing when this function is called, then it will continue to play after the step has occurred.

Back_one_sec;
Causes the CD player to move backwards one second from the current track and time.
If the start of the first track is reached then calling this procedure will cause the CD player to move to the end of the last track.
If the CD is playing when this function is called, then it will continue to play after the step has occurred.

(Horrocks, Constructing the User Interface with Statecharts, 1999)

La specifica in linguaggio naturale tradotta in MSF



Macchina a stati finiti per il controllo del CD player, assumendo che l'utente metta il CD nel cassetto prima di richiuderlo

(Horrocks, Constructing the User Interface with Statecharts, 1999)

Problemi:

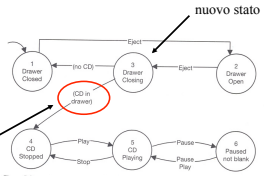
- descrivere le computazioni associate alle transizioni
- rendere espliciti condizioni sulle variabili di stato non esplicitate e non esplicitabili dal diagramma
- la specifica descrive parzialmente il comportamento

Raffinare la tecnica di specifica

Per esprimere condizioni sui valori delle variabili di stato

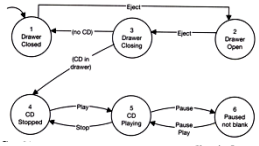
Macchina a stati finiti per il controllo del CD player, in cui viene controllato che il CD sia stato posto nel cassetto

Nota: il formalismo delle macchine a stati finiti è stato arricchito con le ***condizioni sulle transizioni*** (presenti negli statechart)



(Horrocks, Constructing the User Interface with Statecharts, 1999)

La specifica FSM arricchita



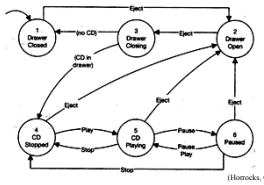
(Horrocks, Constructing the User Interface with Statecharts, 1999)

Questa specifica MSF
a- mette in luce alcuni problemi della specifica in linguaggio naturale: in molte situazioni non so cosa succede (se schiaccio stop, eject etc.)
b- non cattura tutta la specifica in linguaggio naturale

Un completamento (1)

La specifica in linguaggio naturale non dice :
- che succede se si schiaccia eject dopo aver caricato il disco?
- che succede se in pausa (6) schiaccio stop?

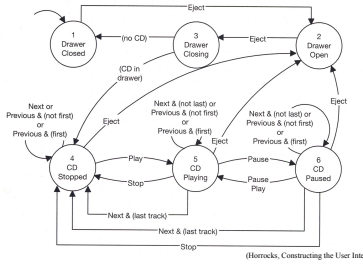
Aggiungo transizioni



(Horrocks, Constructing the User Interface with Statecharts, 1999)

Vengono gestiti questi eventi negli stati 4, 5, 6 (ma non so se queste erano le intenzioni di chi ha fatto la specifica)

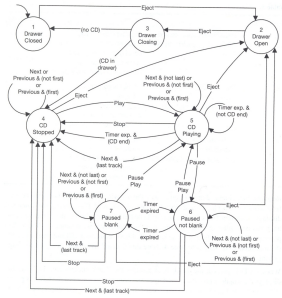
…un altro (2)



(Horrocks, Constructing the User Interface with Statecharts, 1999)

… con specifica del comportamento dei bottoni "Next Track" e "Previous Track", anche quando è posizionato sulla prima ed ultima traccia: si aggiungono 11 transizioni con ripetizioni di nomi di eventi

…e un altro ancora (3)

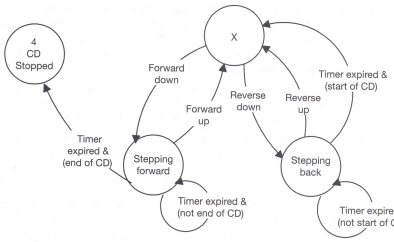


… con specifica del comportamento dei campi "Time Field" e "Track Field", il cui contenuto lampeggia nel caso in cui il CD venga messo in pausa

Ma il diagramma diviene complesso difficile da leggere e da verificare

(Horrocks, Constructing the User Interface with Statecharts, 1999)

…l' ultimo (4)



(Horrocks, Constructing the User Interface with Statecharts, 1999)

… con specifica del comportamento dei bottoni "Reverse" e "Forward": lo stato 'X' va sostituito con gli stati 4, 5, 6, e 7

Considerazioni

- Il formalismo delle macchine a stati finiti
  - deve essere esteso per poter specificare i sistemi interattivi
  - il numero degli stati e degli eventi (transizioni) aumenta rapidamente
  - molti stati ed eventi duplicati
  - macchine a stati finiti grandi e difficili da leggere, da gestire e da verificare
  - macchine a stati finiti non scalabili

Richieste

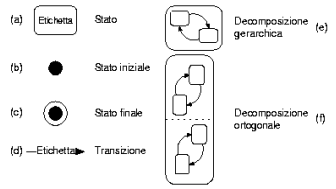
Ad una notazione
- Non ambigua
- Concisa
- Strutturata
- Con possibilità di astrazione (gerarchica)
- Modulare
- Verificabile (formalmente, con supporti programmati)
- Facilmente modificabile
- Espressiva
- Suggestiva (per il progettista)
- Traducibile in codice
- Usabile come strumento di comunicazione fra progettista e realizzatore

Statechart (Diagrammi di stato)

- Adottato da UML, è basato sulle statechart di David Harel (1987)
- Estendono gli state transition diagram
- Forniscono una notazione ricca ed espressiva che permette di specificare sistemi complessi in maniera concisa e a differenti livelli di astrazione
- Seguono un importante principio di economia: come risparmiare segni senza perdere in espressività e suggestività

25

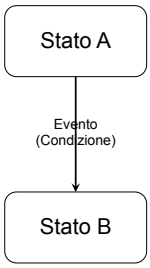
Lessico e regole di composizione



26

Statechart

- Lo statechart è una semplice rete di stati e di eventi. Siamo interessati a statechart che modellano il comportamento di un'interfaccia
- Il singolo stato viene rappresentato da un rettangolo descrivente il particolare stato in cui l'applicazione si trova
- Gli stati sono collegati fra loro da delle "frecce evento (condizione)" e rappresentano il verificarsi di determinate situazioni
- La descrizione associata alla freccia indica l'evento che si deve verificare sotto determinate condizioni perché si possa entrare nello stato indicato



27

Etichetta di una transizione e attività associata a uno stato

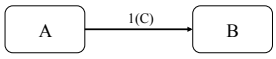
- L'etichetta di una transizione ha 3 elementi tutti opzionali:
  Evento [(Condizione)] / Azione

  dove:
  - *Evento*: input, stimolo che può produrre una transizione di stato
  - *Condizione*: espressione logica
  - *Azione*: computazione atomica (non interrompibile da eventi)
- Lo stato può avere associata una attività, questo si indica con la sintassi *do/attività*
- Un'attività impiega un tempo più lungo di un'azione e può essere sempre interrotta da un evento

28

Tabelle evento-azione

- Statechart: rete di stati ed eventi



- L'etichetta è un nome (espressivo) che specifica la condizione; le azioni associate sono spesso molte
- "Se accade l'evento 1 quando il sistema è nello stato A e la condizione C è vera, allora avviene una transizione nello stato B"
- L'azione spesso non si specifica sul diagramma: ce ne possono essere più di una e può essere lungo descriverle. Si scrive una tabella evento-azione

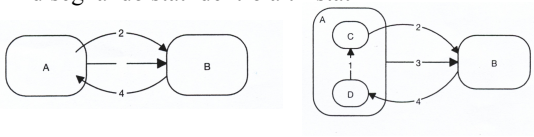
Stato corrente	*Evento e condizioni*	*Azioni*	*Stato futuro*
A	click su bottone 'crea' e stato del campo = 'X'	Inserisci dati P nel database Q	B

29

Stati con profondità (1/3)

Gli stati sono disposti in maniera gerarchica, ovvero con una certa profondità

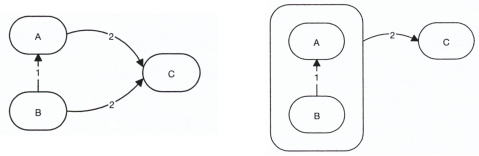
Si può rappresentare una gerarchia di stati disegnando stati dentro altri stati



30

Stati con profondità (2/3)

La profondità può essere usata per raggruppare più stati in modo da ridurre il numero di frecce rappresentanti gli eventi in un diagramma



Stati con profondità (3/3)

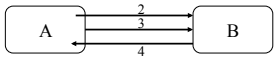
Utilizzare la profondità offre diversi vantaggi:

- la definizione dei dettagli degli stati del livello più alto può essere eseguita simultaneamente da persone diverse
- il sistema può essere visto a diversi livelli di astrazione, per cui si può studiare una porzione dettagliata del diagramma senza conoscere l'intero sistema in dettaglio
- il numero delle frecce che rappresentano le transizioni può essere notevolmente ridotto

32

Descrizione a diversi livelli di astrazione

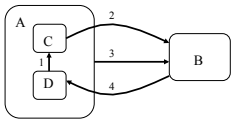
- Gli stati di uno statechart possono formare una struttura gerarchica: uno stato individua un sistema di altri stati – fino ad arrivare a stati atomici



- Gli stati A e B forniscono una descrizione del sistema ad alto livello di astrazione, esiste un livello di dettaglio inferiore che non è mostrato: l'evento 2 inizia dentro lo stato A e l'evento 4 finisce dentro A

33

Raffinare la descrizione di uno stato



Quando il sistema è nello stato A, significa che può essere o nello stato C o nello stato D

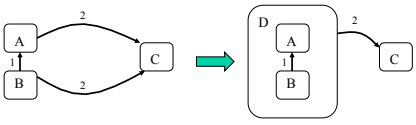
La transizione 2 inizia in C e termina in B (questo è lecito anche se B è a un livello di astrazione superiore)

Normalmente si specificano gli stati di più alto livello del sistema e successivamente si vanno a specificare i dettagli di livello più basso ➔ ciò aiuta a gestire la complessità del sistema

34

Sintetizzare (Clustering)

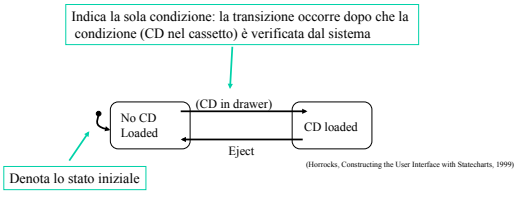
- La gerarchia può essere usata anche per ridurre il numero degli archi: economia di transizioni
- In questo caso si introduce uno stato di livello di astrazione più alto che raggruppa stati di caratteristiche dinamiche simili



Quindi, 2 scopi della struttura gerarchica: ***raffinamento*** (top-down) e ***clustering*** (bottom-up)

35

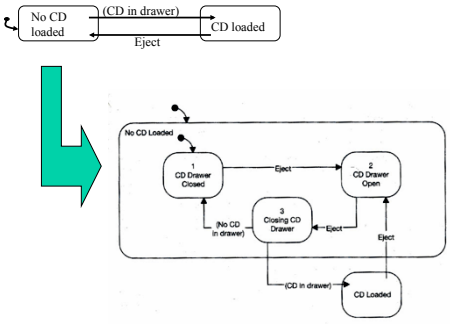
Specificare il comportamento del CD player



Al livello più astratto

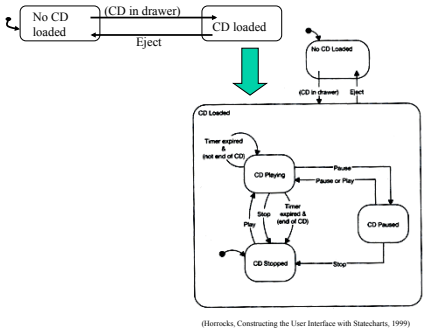
36

Scendiamo di livello nello stato "No CD Loaded"



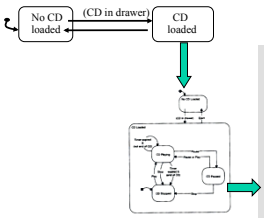
(Horrocks, Constructing the User Interface with Statecharts, 1999)

Scendiamo di livello nello stato "CD Loaded"

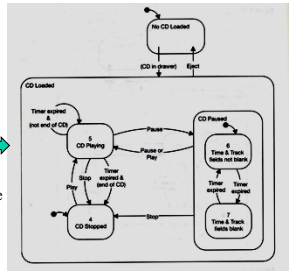


(Horrocks, Constructing the User Interface with Statecharts, 1999)

Scendiamo di livello nello stato "CD Loaded"



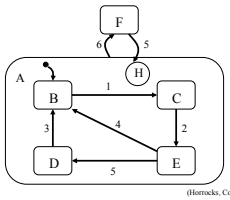
(Horrocks, Constructing the User Interface with Statecharts, 1999)



Stati descritti a *tre livelli* di astrazione
il contesto è chiaro
l'operazione locale dettagliata
gli stati 6 e 7 atomici

Gestire la persistenza

Il meccanismo di History (H)

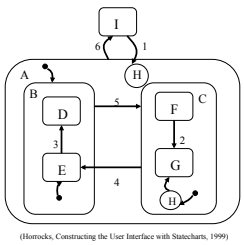


(Horrocks, Constructing the User Interface with Statecharts, 1999)

Fornisce un modo per entrare in un gruppo degli stati, sulla base della storia compiuta dal sistema in quel gruppo: lo stato in cui si entra è quello più recentemente visitato del gruppo

Nell'esempio: quando accade l'evento 5, si entra nello stato A, e il meccanismo di history permette di decidere in quale stato di A entrare (se è la prima volta entra nello stato B)

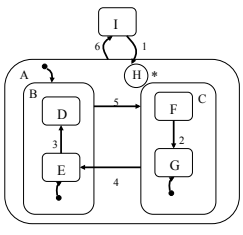
Meccanismo di History: altri esempi



(Horrocks, Constructing the User Interface with Statecharts, 1999)

- ✓Per applicare il meccanismo di history anche ai livelli inferiori occorre aggiungere il simbolo H anche in questi
- ✓Quando avviene l'evento 1 si entra in B o in C a seconda di qual è il più recentemente visitato
- ✓Se si entra in C si decide fra F e G a seconda di quello più recentemente visitato
- ✓La freccia per lo stato iniziale è in questo caso attaccata al simbolo H, attaccato a sua volta allo stato G: se C non è mai stato visitato, si entra in G di default

Meccanismo di History: altri esempi

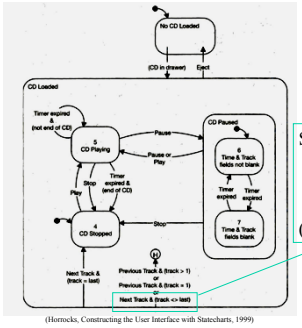


(Horrocks, Constructing the User Interface with Statecharts, 1999)

Il simbolo * vicino a un simbolo di history indica che il meccanismo di history va applicato a tutti i livelli

Se D è lo stato in A acceduto più di recente:
Se occorre 1 B è preferito a C e D ad E.
Se D non è atomico, al suo interno si riapplica il meccanismo di history

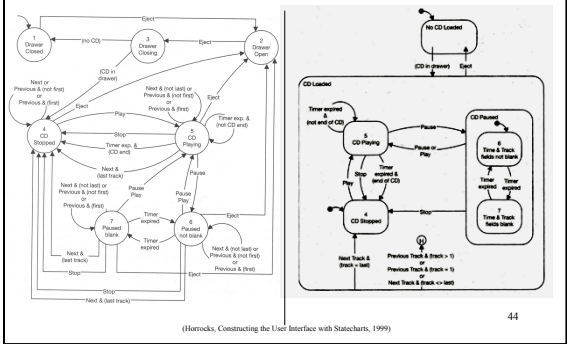
Esempio CD player: gestione scorrimento tracce col meccanismo di History



Se seleziono Next Track e non sono sull'ultima traccia, seleziono la prossima (azione) e torno nello stato in cui ero (storia)

(Horrocks, Constructing the User Interface with Statecharts, 1999)

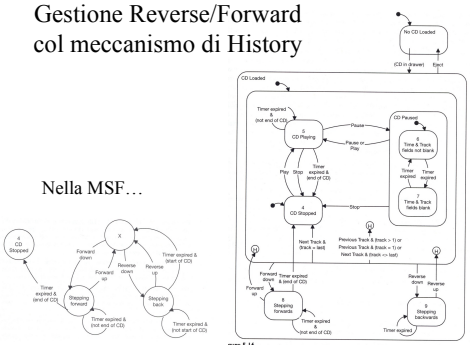
Un confronto



(Horrocks, Constructing the User Interface with Statecharts, 1999)

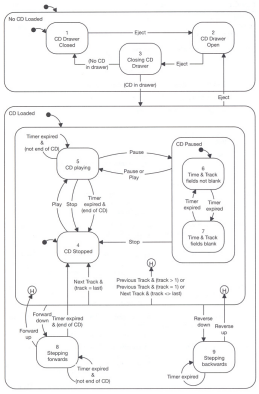
Gestione Reverse/Forward col meccanismo di History

Nella MSF…



(Horrocks, Constructing the User Interface with Statecharts, 1999)

Statechart completo



(Horrocks, Constructing the User Interface with Statecharts, 1999)

Gestione di processi concorrenti

- Gli statechart evitano l'esplosione combinatoria tipica delle MSF
- Consentono la specifica di interfacce complesse
- Processi concorrenti vengono descritti da un unico stato separato da linee tratteggiate in tante zone quanti sono i processi: in ogni zona è rappresentato lo statechart di un processo componente
- Entrare nello stato globale significa entrare in ogni processo componente ed attivare tutti i processi simultaneamente
- Nella realtà non c'è simultaneità- le transizioni in generale saranno eseguite una dopo l'altra ma questo non importa poiché i processi sono definiti come concorrenti

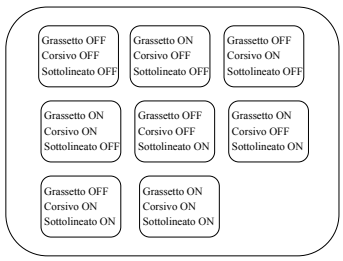
Concorrenza

3 Bottoni indipendenti

G *C* S

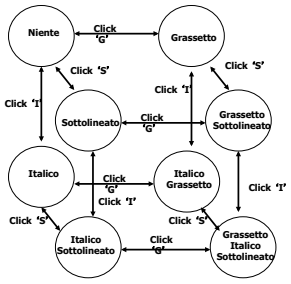
8 stati possibili

Quante transizioni?

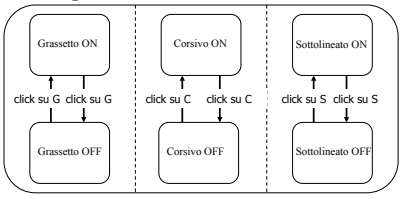


E se aggiungo altri bottoni?

Ricordate la STN…



Concorrenza fra oggetti indipendenti specificata tramite statechart



Oggetti indipendenti possono essere specificati da diagrammi indipendenti

Diagrammi indipendenti possono operare in modo concorrente

Oggetti non indipendenti

Aggiungiamo altri 4 bottoni 

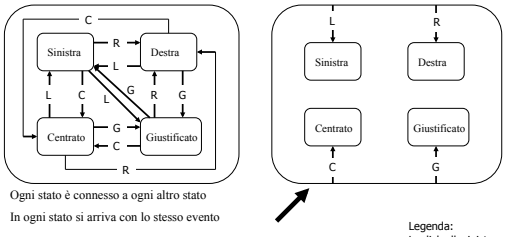
Allinea a sinistra, allinea a destra, centra, giustifica non operano in modo indipendente

Soltanto uno di loro può essere attivato in ogni istante (si comportano come un radio button)

L'attivazione di uno di loro fa disattivare tutti gli altri

Lo stato corrente è determinato dall'ultimo bottone che è stato cliccato

Diagramma per gli oggetti 



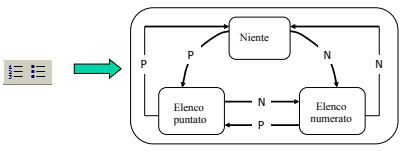
Ogni stato è connesso a ogni altro stato

In ogni stato si arriva con lo stesso evento

Diagramma equivalente: da qualsiasi stato si parta con un certo evento si arriva sempre nello stesso stato

Legenda:
L: click all. sinistra
R: click all. destra
C: click centrato
G: click giustificato

Aggiungiamo ancora 2 oggetti…



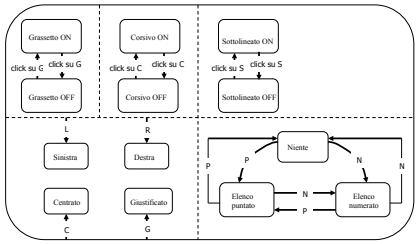
Anche questi sono dipendenti l'uno dall'altro

Non si comportano come radio button: esiste il caso in cui nessuno è attivato

Tutti insieme… 

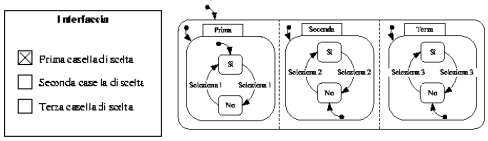
5 statechart indipendenti

3 1 1

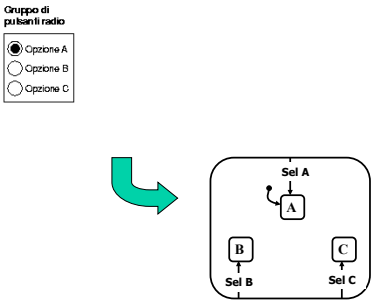


13 stati contro 96 (8 x 4 x 3) del diagramma a stati finiti

In generale: caselle di scelta



In generale: gruppo di bottoni radio



Euristiche di sviluppo

- Far coincidere le schermate principali di un'applicazione con gli stati di più alto livello
- Dettagliare il comportamento all'entrata e all'uscita di uno stato
- Identificare gli oggetti di una interfaccia che cambiano il loro stato a seconda dell'interazione dell'utente
- Identificare gli eventi che provocano tali cambiamenti
- Identificare gli elementi che rimangono costanti
- Evitare che parti concorrenti gestiscano gli stessi oggetti
- Isolare le parti indipendenti fra loro in modo da aumentare la manutenibilità
- Sincronizzare gli eventi simultanei

Test di progetto 1/2

- Controllare se tutte le transizioni sono state descritte o se ve ne sono alcune che mancano
- Per ogni stato bisogna considerare tutti gli oggetti e verificare se gli eventi associati sono stati modellati
- Se un evento in uno stato particolare non causa una transizione, allora bisogna giustificare se quell'evento è davvero impossibile
- Controllare se uno stato deve fare qualcosa per evitare che succeda un particolare evento
- Controllare se tutti gli stati siano in qualche modo raggiungibili

Test di progetto 2/2

- Controllare che non ci siano degli stati che non possano essere abbandonati in alcun modo per evitare cicli
- Controllare che non si introducano comportamenti non deterministici in uno statechart
- Controllare che non ci si basi sul verificarsi di alcuni eventi in un ordine particolare
- Le condizioni usate in uno statechart dovrebbero essere sempre associate agli stati cui le frecce evento portano
- Le azioni non dovrebbero contenere condizioni che provocherebbero l'esecuzione di diverse azioni a seconda del contesto

Conclusioni

- Gli Statechart rappresentano un utile formalismo per la specifica di interfacce visuali
- Sono relativamente facili da comprendere
- Costituiscono un efficace guida per la successiva fase di implementazione
- Per progetti di larga scale potrebbero diventare difficilmente gestibili
- Tale svantaggio è però mitigato dalle numerose possibilità di astrazione

Riferimenti

1. Ian Horrocks, *Constructing the User Interface with Statecharts*, Addison-Wesley 1999
2. David Harel, *On Visual Formalism*, ACM, 1988
3. David Harel e Chaim-Arie Kahana, *On Statecharts with Overlapping*, ACM, 1992

61