Dipartimento di Informatica
Università degli Studi di Bari

DE_VISU

Programma di ricerca (cofinanziato dal MURST, esercizio 2000)

Specifica, Progetto e Sviluppo di Sistemi Interattivi Visuali

Linee Guida per il Progetto e la Valutazione di Sistemi Interattivi Visuali Usabili

RAPPORTO TECNICO: N.02 ANNO 2002

BA-R05 **30 Maggio 2002**

Sommario

In questo rapporto vengono fornite delle linee guida sia per la progettazione di sistemi interattivi visuali usabili sia per la loro valutazione. L'attenzione è rivolta alle interfacce di tali sistemi, poiché è l'interfaccia la componente significativa per gli utenti finali. Si analizzano quindi interfacce di tipo WIMP. La valutazione viene effettuata mediante una tecnica specifica di ispezione di usabilità, proposta nell'ambito della metodologia SUE, che fa uso di linee guida, dette Abstract Task (AT), per supportare il valutatore durante l'attività di ispezione.

Titolo ricerca unità	L'usabilità nel progetto di sistemi interattivi
Codice	BA-R05
Data	30 Maggio 2002
Tipo di prodotto	Rapporto tecnico
Unità responsabile	BA
Unità coinvolte	BA - SA
Autori	Maria Francesca Costabile, Carmelo Ardito, Rosa Lanzilotti, Grazia Minardi, Antonio Piccinno, Carlo Dell'Aquila.
Autore da contattare	Maria Francesca Costabile Dipartimento di Informatica Università degli Studi di Bari Via Orabona 4, 70125 Bari, Italia costabile@di.uniba.it

1. Introduzione

In questo rapporto vengono fornite delle linee guida sia per la progettazione di sistemi interattivi visuali usabili sia per la loro valutazione. L'attenzione è rivolta alle interfacce di tali sistemi, poiché è l'interfaccia la componente significativa per gli utenti finali. Si analizzano quindi interfacce di tipo WIMP. La valutazione viene effettuata mediante una tecnica specifica di ispezione di usabilità, proposta nell'ambito della metodologia SUE, che fa uso di linee guida, dette Abstract Task (AT), per supportare il valutatore durante l'attività di ispezione.

Il presente rapporto è così organizzato: nella sezione 2 è riportata la definizione dell'usabilità secondo Nielsen e i principi di usabilità generali che il progettista deve considerare per creare un sistema usabile. Nella sezione 3 si descrive la metodologia di progetto proposta dal PCL, proponendo delle linee guida per la progettazione di sistemi interattivi. Nella sezione 4 è descritta la metodologia di valutazione dell'usabilità SUE. Tale metodologia è applicata alla valutazione di interfacce visuali di tipo WIMP. A questo scopo si introduce il modello OO-WIMP per modellare gli elementi di presentazione (widget) di un'interfaccia WIMP. Tale modello aiuta a identificare gli elementi da considerare nell'ispezione dell'interfaccia visuale. Nella sezione 7, sono riportati degli esempi di valutazione di applicazioni commerciali.

Infine, è presente un'appendice dove sono riportati i 29 AT che supportano il valutatore nell'ispezione e un glossario dei termini usati nell'ambito del PCL.

2. Usabilità dei sistemi interattivi

Diverse definizioni di usabilità sono state proposte. Nielsen presenta l'usabilità come uno degli aspetti che caratterizza una caratteristica globale di un sistema che è l'accettabilità da parte degli utenti finali, riflettendo se il sistema è abbastanza buono da soddisfare i bisogni e le richieste degli utenti [Nie93]. Oltre agli aspetti usualmente considerati dagli ingegneri del software legati a costo, affidabilità, compatibilità con i sistemi attuali, ecc, l'aspetto che più interessa l'utente finale è l'utilizzabilità, che è distinta in: Utilità ed Usabilità, dove l'utilità indica se le funzionalità del sistema in principio possono fare quello che è necessario, e l'usabilità è relativa a come gli utenti possono usare le funzionalità (Figura 1).

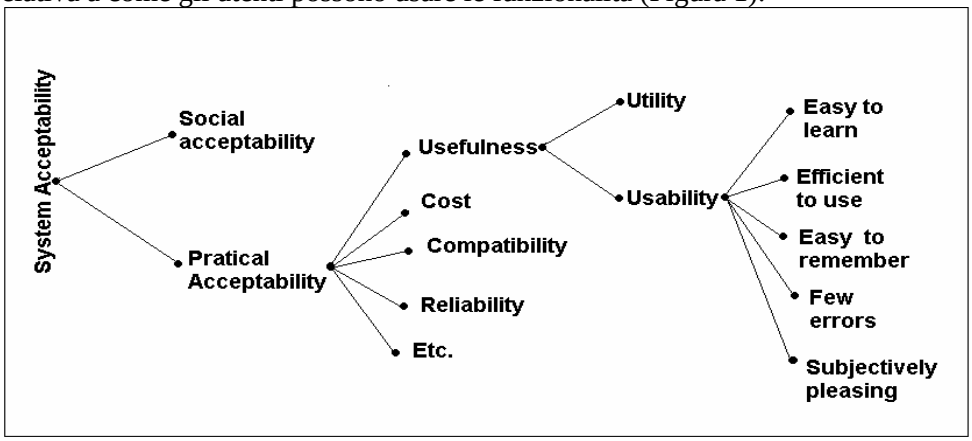

Figura 1 Usabilità come attributo dell'accettabilità di un sistema

Nella definizione di Nielsen, l'usabilità non è una proprietà uni-dimensionale di un sistema, ma essa ha diverse componenti. Può essere decomposta in cinque dimensioni:

- *Apprendibilità*. Facilità nell'apprendere il comportamento del sistema.
- *Efficienza d'uso*. Il livello di performance raggiungibile dall'utente.
- *Memorizzabilità*. Facilità nel ricordare le funzionalità del sistema.
- *Pochi errori*. Il sistema deve possedere un basso tasso di errori.
- *Soddisfazione*. Il sistema deve essere piacevole da usare [Nie93].

Nielsen fornisce dieci "Principi d'usabilità" per creare una buona interfaccia utente. Sono qui descritti:

1. Dialogo semplice e naturale

- il dialogo non deve contenere informazioni irrilevanti o usate raramente;
- l'informazione deve apparire in ordine naturale logico;

2. Utilizzare il linguaggio degli utenti

- il dialogo deve essere espresso chiaramente in parole, frasi e concetti familiari all'utente;

3. Minimizzare il carico di memoria dell'utente

- le istruzioni per l'uso del sistema devono essere sempre visibili o facilmente recuperabili quando necessario;

4. Coerenza

- parole, situazioni, azioni devono avere sempre lo stesso significato;

5. Feedback

- il sistema deve mantenere informato l'utente relativamente a quello che sta facendo, attraverso feedback appropriati, efficaci ed efficienti;

6. Uscite evidenziate chiaramente

- gli utenti spesso scelgono delle funzioni per errore e quindi hanno bisogno di lasciare facilmente uno stato non voluto;

7. Shortcut

- velocizzare l'interazione per utenti esperti con l'uso di acceleratori;

8. Messaggi di errore significativi

- esprimerli in linguaggio semplice, indicando precisamente il problema e suggerendo la soluzione in modo costruttivo;

9. Prevenire errori

- un progetto accurato deve prevenire il verificarsi di errori;

10. Aiuto e documentazione

- l'informazione deve essere di facile ricerca, focalizzata sul task dell'utente, coincisa e deve specificare i passi concreti da eseguire.

Oltre a questi principi riportati da Nielsen, riteniamo che per i sistemi ipermediali e per quelli a realtà virtuale, che più frequentemente dobbiamo valutare, è opportuno specificare i seguenti principi:

11. Visibilità

- I controlli devono essere visibili, con una buona e immediata corrispondenza con i loro effetti [Nor88, Pre94];

12. Prevedibilità

- Il progetto dei vari widgets deve suggerire la loro funzionalità (è simile al principio di "affordance" di Normann) [Nor88, Pre94];

13. Controllo da parte degli utenti

- Disponibilità dei meccanismi che supportano l'utente nell'interazione (tale principio è simile al criterio di "completezza" in [BaR02]);

14. Flessibilità

- Adattabilità alle esigenze dell'utente.

3. Approccio PCL al Progetto di Sistemi Visuali Usabili

L'approccio del Pictorial Computing Laboratory (PCL) al progetto di sistemi interattivi visuali usabili per l'utente finale è motivato da tre principi: a) l'utente deve sempre capire le conseguenze dell'attività del sistema rispetto all'esecuzione del suo compito; b) l'utente deve sempre avere il controllo della computazione interattiva, evitando di perdersi nello spazio virtuale; c) il sistema deve evitare e/o bloccare gli errori degli utenti mantenendosi affidabile e cioè in un insieme prevedibile di stati. Il progetto del sistema visuale è effettuato attraverso vari passi, partendo dall'analisi delle notazioni che gli utenti finali hanno sviluppato nei loro ambienti di lavoro, e producendo ad ogni passo un nuovo linguaggio visuale che deve essere verificato ma anche valutato dal punto di vista dell'usabilità. A tale scopo, la metodologia di progetto è integrata con la metodologia di valutazione dell'usabilità SUE [Mat02].

L'approccio del PCL ha generato una metodologia di progetto che mira a soddisfare i tre principi descritti precedentemente sulla base delle seguenti assunzioni: a) *adeguatezza della comunicazione utente-calcolatore* [Cha96] e *comunicabilità del sistema* [Pra00] sono prerequisiti per l'accettazione del sistema da

parte dell'utente, quindi sono dimensioni importanti per tutta l'usabilità del sistema; b) *le notazioni degli utenti* sono il modo in cui gli utenti esprimono la loro cultura, di conseguenza lo sviluppo di sistemi usabili deve capitalizzare su queste notazioni [Lat86]; c) la *formalizzazione* è un prerequisito per verificare validità, completezza e coerenza dei risultati ottenuti progressivamente nelle fasi di sviluppo del progetto [Dix98]; d) esistono due *semantiche,* una relativa all'utente e l'altra implementata nel sistema: la validazione del sistema è un'attività che mira a valutare la corrispondenza tra queste due semantiche; e) un uso coordinato delle attività di *validazione* e *verifica* è necessario per assicurare l'usabilità e l'affidabilità di un sistema interattivo visuale. L'attività di validazione del sistema è basata principalmente sulla valutazione dell'usabilità, stressando le dimensioni dell'adeguatezza della comunicazione e della comunicabilità. L'attività di verifica è basata su una teoria formale per la specifica ed il progetto di sistemi interattivi visuali, derivata dalla teoria delle sentenze visuali, sviluppata negli ultimi anni [Bot99].

Nell'approccio del PCL, l'interazione tra utente e calcolatore è un processo in cui l'utente ed il calcolatore comunicano materializzando e interpretando una sequenza di messaggi a istanti di tempo successivi, l'uomo usando i suoi criteri cognitivi, il calcolatore usando i criteri in esso programmati [Bot99]. Un sistema interattivo visuale può essere considerato come un generatore di messaggi dal progettista all'utente del sistema [Cha96].

In ogni interazione sono sempre definite implicitamente due semantiche; una interna al sistema, in cui ogni messaggio è associato al significato computazionale, come definito dal progettista e implementato nel sistema, ed una propria dell'utente che sta eseguendo il compito, e che dipende dal suo ruolo nell'esecuzione del compito, così come dalla sua cultura, esperienza, ecc. Come osservato in [Cha96], perché il compito dell'utente possa essere eseguito con successo, occorre che un significato simile sia associato dall'utente e dal sistema ad ogni messaggio, e quindi che si raggiunga una comunicazione adeguata. Nel caso di interfacce 2D a manipolazione diretta, che è lo stile di interazione che considereremo in questo documento, i messaggi scambiati sono le immagini rappresentate sullo schermo del calcolatore, composte da testo, grafici, icone, ecc.

Gli esseri umani interpretano tali immagini riconoscendo in esse delle *strutture caratteristiche* (dette anche *strutture* oppure ***cs*** dalle iniziali delle parole inglesi characteristic structures), cioè insiemi di pixel dell'immagine che sono riconosciute come unità percettive o funzionali. Riconoscere una struttura significa associare ad essa un significato. In generale, le persone esprimono il significato attribuito alla struttura tramite una descrizione verbale. L'identificazione della ***cs*** è influenzata dalla similarità (o dissimilarità) con entità grafiche e costrutti tradizionalmente adottati nella comunità dell'utente.

Anche il sistema associa entità grafiche con costrutti computazionali. E' esattamente questa associazione che rende il computer capace di interpretare le azioni dell'utente (come il click su un bottone) rispetto all'immagine i sul video, eventualmente attivando attività computazionali i cui risultati sono materializzati sullo schermo tramite creazione, cancellazione, oppure modificazione delle strutture caratteristiche. In altre parole, il sistema cattura ogni azione dell'utente e la interpreta relativamente all'immagine *i* attualmente sullo schermo, usando una descrizione *d* di essa. *d* è definita come un insieme di simboli attribuiti, ognuno che descrive il significato di una *cs* nell'immagine *i*.

La relazione tra le *cs* nell'immagine e i simboli attribuiti nella rappresentazione computazionale che descrivono il significato delle *cs* è specificata da due funzioni. Sia CSi l'insieme di *cs* nell'immagine *i*. Una funzione di interpretazione int: CSi → d associa le *cs* con i simboli attribuiti. Una funzione di materializzazione mat: d → CSi associa i simboli attribuiti con le *cs*. Una *vs* è definita come una tripla <i,d,<int,mat>>. Un linguaggio visuale (IVL) è un insieme di *vs*. Data una *vs* = <i,d,<int,mat>>, un pattern caratteristico ***cp*** è una tripla cp = <s,u,<intcs, matcs>> dove sÎ CSi, uÎ d, s=matcs(u) e u=intcs(s). Un *cp* specifica lo stato del processo computazionale che genera e mantiene attiva un'entità virtuale. Essendo un'entità virtuale rappresentata graficamente nella bitmap da *s*, *u* è il simbolo attribuito che completa la descrizione del processo di generazione, per esempio contiene il codice eseguibile del programma e i valori correnti delle sue variabili; la coppia <intcs, matcs> specifica il collegamento tra la bitmap e le componenti di *u*. Una *vs* è caratterizzata dall'insieme dei suoi *cp*.

Nell'appendice del presente rapporto, è riportato un glossario dei principali termini usati nella teoria delle "Visual Sentence" [Bot99] per il progetto di sistemi visuali.

3.1 Linee Guida

In questa sottosezione si discute dell'importanza delle nozioni di *scaffold* e *frame* di un'interfaccia visuale e si propongono delle linee guida per supportare il progettista nella creazione di interfacce usabili.

Nell'approccio del PCL, si parte dall'analisi delle notazioni utilizzate dagli utenti nel dominio applicativo per progettare il linguaggio visuale di interazione. Tali notazioni sono opportunamente formalizzate e costituiscono il Task Visual Language (TVL) [BaR03]. Il TVL e' poi arricchito dallo scaffold e dal frame, per costituire il linguaggio finale di interazione IVL. Infatti, diamo le seguenti definizioni:

1) *scaffold* è un insieme di *cs* non strettamente collegate allo specifico dominio applicativo, per supportare le azioni dell'utente e la navigazione (per esempio un'icona per attivare la stampa, frecce di navigazione, eccetera);
2) *frame* è un insieme di *cs* che rimangono invariate in immagini successive sullo schermo, fornendo così un contesto, dei punti di riferimento che evitano il disorientamento dell'utente. Ad esempio, la barra dei menù di Micorsoft Word e' parte del frame dell'interfaccia di tale applicazione, e rimane invariata al variare dei documenti su cui si lavora di volta in volta.

Di conseguenza, l'usabilità del progetto del linguaggio visuale di interazione IVL è fortemente influenzata dal progetto di scaffold e frame. Le linee guida che suggeriamo sono un supporto per un progetto accurato.

L'interazione visuale avviene attraverso lo schermo. Sullo spazio dello schermo non viene imposto un ordinamento lineare in maniera naturale, mentre possono essere definite varie relazioni spaziali tra coppie di *cs*. Infatti, due *cs* possono essere collegate, disgiunte, una a sinistra dell'altra, una a destra dell'altra, sopra, sotto, eccetera. Tutte queste relazioni sono potenzialmente significative, così che devono strettamente vincolate e chiaramente suggerite all'utente. Un primo importante requisito è che le relazioni spaziali siano usate coerentemente nei vari insiemi di *cp*.

Linea Guida 1*: Se una certa relazione del dominio è rappresentata da una relazione spaziale tra cs, tale relazione del dominio deve essere rappresentata dalla stessa relazione spaziale per tutti gli insiemi di cp nella stessa vs.*

Una buona norma per evitare inferenze arbitrarie è adottare disposizioni suggestive, ad esempio disponendo un insieme di *cs* intorno ad un altro insieme, soltanto per rappresentare relazioni effettivamente valide e non adottare tali disposizioni se non si intende comunicare alcun significato.

In interazioni di tipo "punta e clicca", assumiamo che ogni azione dell'utente implica un riferimento alla *cs* a cui il cursore sta puntando. Gli shortcut da tastiera possono essere considerati come la condensazione di una sequenza di azioni che fanno partire un'attività del computer, come se fosse stata selezionata una voce del menu dopo averla puntata. Da ciò segue la:

Linea Guida 2*: Ogni azione dell'utente deve essere supportata da una specifica cs.*

Durante l'interazione, le trasformazioni di vs sullo schermo sono particolarmente significative. Siccome assumiamo che le trasformazioni delle *vs* avvengano sotto il diretto controllo dell'utente, dobbiamo progettare dei feedback appropriati. In particolare, distinguiamo feedback sintattico e semantico. Poiché il feedback sintattico non implica calcoli complessi, può essere considerato istantaneo; invece, il feedback semantico può avvantaggiarsi dall'identificazione di stati significativi nel calcolo della reazione.

Linea Guida 3*: Se nell'esecuzione di un'attività non esistono stati intermedi significativi, le cs devono avere solo due aspetti possibili.*

Per esempio il passaggio di un'icona dallo stato abilitato a quello disabilitato deve apparire istantaneo all'utente. Se gli stati abilitato e disabilitato sono identificati da due colori opposti ("in reverse"), non ci devono essere stati intermedi in cui l'icona appare in un colore intermedio. Infatti, mostrare una rappresentazione intermedia potrebbe indurre l'utente a dedurre l'esistenza di uno stato collegato ad ogni differente aspetto di una *cs*.

Linea Guida 4*: Ogni aspetto differente assunto da un insieme di cs che rappresenta risultati di attività deve riflettere uno stato coerente dell' applicazione, e l'interruzione dell'attività da parte dell'utente deve lasciare l'applicazione in uno stato coerente.*

Se non è possibile garantire ciò, è meglio fornire una *cs* arbitraria e temporale che indica che la computazione è in progresso. In questo caso l'interruzione dell'attività dovrebbe ripristinare l'applicazione allo stato che aveva prima dell'inizio dell'attività.

La regola generale di mantenere costante ciò che non necessita di essere cambiato (espressa dal principio di *least astonishment* di Timbleby [Thi90]) si applica non solo all'aspetto di *cs* individuali ma anche a relazioni spaziali tra di esse, poiche' cercare una *cs* in una diversa posizione è particolarmente costoso per l'utente. Quindi, la modifica della posizione delle *cs* dovrebbe avvenire solo sotto il controllo diretto dell'utente o riflettere una modifica reale delle proprietà relative dei simboli attribuiti corrispondenti.

Linea Guida 5*: Dato un insieme di simboli attribuiti in una vs, se una data relazione spaziale tra le cs ad essi associate viene usata per rappresentare una certa relazione tra le cs, la stessa relazione spaziale deve essere coerentemente usata in tutti i cp in tutte le sentenze visuali dello stesso linguaggio IVL.*

L'interazione è basata su trasformazioni incrementali della *vs* attiva sullo schermo. Tali trasformazioni sono generate dai comandi dell'utente e dalle computazioni del sistema. La presenza e lo stato delle *cs*

nell’immagine corrente deve consentire di identificare quali sono i risultati dell’interazione precedente e quali interazioni sono ora possibili. In particolare, una sorgente importante di informazione per l’utente è sia il riconoscimento delle trasformazioni che le *cs* hanno subito sia la persistenza di *cs* specifiche durante l’interazione. In particolare, consideriamo il frame come un insieme di *cs* che restano costanti o mantengono supporto (si veda glossario in Appendice) costante in tutte le *vs* generate durante l’interazione. Quindi, una *cs* in un frame può subire un insieme limitato di trasformazioni che preservano il suo supporto. Il tempo di vita (lifetime) di un frame è il tempo di vita del supporto delle *cs* che lo formano.

Data una sequenza di *vs* generate durante un’interazione, possono essere identificati diversi frame. In particolare, l’esistenza di un frame F1 che è presente nell’intera sequenza è importante per permettere agli utenti di identificare l’ambiente con cui stanno interagendo. Un frame che ha una vita meno lunga può comunicare all’utente una specifica modalità, come l'impostazione di alcune proprietà.

Un frame può essere formato dalle *cs* appartenenti all'ATVL (che altro non e’ che il TVL aumentato grazie alle possibilità fornite da un calcolatore, si veda [BaR03]), così come allo scaffold. La definizione di frame pone due restrizioni sui suoi elementi, una geometrica: le *cs* mantengono lo stesso supporto; e una semantica: le *cs* sono definite strutture *caratteristiche*, proprio perche’ sono associate ad un significato.

Le restrizioni semantiche sono importanti poiché, durante l’interazione, una *vs* è soggetta a due interpretazioni: da parte dell’utente e da parte del sistema. Se l’utente e il sistema non identificano lo stesso frame in un insieme di *vs*, possono sorgere equivoci. Gli utenti possono anche essere indotti ad adottare un certo comportamento in base all’esistenza di un certo frame. A causa della doppia natura del frame, geometrica e semantica, bisogna fare in modo che l’utente non attribuisca significati semantici a arrangiamenti geometrici di pixel che il progettista ha disposto accidentalmente (per un esempio di un ambiguo utilizzo del frame, si veda l’esempio di Cave of Magic riportato in [Bot99].

Linea Guida 6*: L’esistenza di un frame deve essere suggerita all’utente dalla sua persistenza durante una trasformazione.*

Linea Guida 7*: All’utente non dovrebbe essere imposta alcuna azione irrilevante per il task.*

La linea guida 7 e’ in accordo col concetto di *viscosità* descritto in [Gre96], che risulta un requisito importante nell'interazione con un sistema visuale: il sistema deve chiedere all'utente solo azioni significative ai fini del task, facendosi carico di eseguire tutte quelle azioni che possono essere effettuate in modo automatico.

Linea Guida 8*: Le trasformazioni delle cs da una vs ad un’altra devono essere evidenti, in modo che l’utente sia sempre informato dello stato del processo.*

Il progetto dei frame è un task fondamentale nella costruzione di un sistema visuale interattivo. In particolare, i frame possono essere usati per identificare i differenti modi in cui l'interazione può avvenire, per facilitare la percezione di una continuità della trasformazione, fornendo così un background percettivo in cui le trasformazioni possono avvenire. A tal fine, è importante che le trasformazioni più veloci avvengano in delle posizioni che risaltino più facilmente all’utente, ad esempio al centro dello schermo.

Linea Guida 9*: le cs che si trasformano più lentamente devono trovarsi nella parte periferiche del display.*

In altre parole, gli elementi del frame sono quelli che cambiano più lentamente durante l'interazione proprio per essere di riferimento all'utente e fornirgli un supporto per orientarsi. La barra dei menù e i menù iconici di base nelle applicazioni di Microsoft Office sono parte del frame di tali applicazioni ed e' noto che la loro forma e la loro disposizione facilitano l'orientamento dell'utente.

4. Valutazione di usabilità: la metodologia SUE

SUE (Systematic Usability Evaluation) è una metodologia di valutazione dell'usabilità, sviluppata in cooperazione dai ricercatori dell'Università di Bari e del Politecnico di Milano. Combina metodi di ispezione con metodi empirici in modo sistematico, sfruttando le caratteristiche migliori di entrambi e riducendo gli svantaggi. L'assunzione base di SUE è che un'approfondita valutazione di un'applicazione interattiva si ottiene attraverso più processi di valutazione, ognuno focalizzato su una specifica dimensione dell'analisi [Gar99]. Oltre ai principi generali di usabilità, ogni processo di valutazione deve anche considerare attributi di usabilità più specializzati, in grado di catturare le caratteristiche di un'applicazione che rientrano nella dimensione dell'analisi scelta. SUE prescrive che ogni processo di valutazione inizi dall'ispezione dell'applicazione. Gli ispettori usano modelli per descrivere l'applicazione e identificare gli oggetti rilevanti della valutazione. Poi, per analizzare ogni oggetto, i valutatori usano gli Abstract Task, pattern di valutazione che descrivono al valutatore l'attività che deve eseguire e gli aspetti dell'applicazione su cui focalizzare l'analisi. Dopo l'ispezione, se necessario, può anche essere effettuato un test con utenti reali per analizzare in modo più dettagliato eventuali aspetti problematici dell'applicazione. Il progetto del test utente è guidato quindi dai risultati dell'ispezione, essendo così più focalizzato ed efficace [Mat99].

SUE, anche se inizialmente sviluppata per valutare ipermedia, fornisce strutture generali di valutazione per i sistemi interattivi che devono essere specializzate per ogni categoria specifica. L'idea di SUE è che la valutazione di usabilità dovrebbe analizzare vari aspetti di un prodotto, partendo da differenti punti di vista. Alcuni di questi aspetti si riferiscono a caratteristiche generali di presentazione comuni a tutti i sistemi interattivi, altri sono più specifici di un particolare tipo di prodotto, o di un particolare dominio d'uso [Gar99].

Il numero di punti di vista possibili è vario, ma è possibile raggrupparli secondo tre differenti livelli [Mat99]: livello generale (o livello 1), in cui la valutazione si concentra sulle caratteristiche generali, comuni a tutti i sistemi interattivi, come il layout della schermata, l'efficacia delle icone, i menu, ecc; livello categoria (o livello 2), in cui la valutazione si concentra sulle caratteristiche che sono peculiari per una determinata categoria, per esempio gli aspetti della navigazione, le strutture di accesso e i controlli sui media attivi nel caso degli ipermedia; livello dominio dell'applicazione (o livello 3), in cui la valutazione si focalizza sulle caratteristiche specifiche del dominio applicativo e sui requisiti del singolo prodotto esaminato. Per esempio, per un ipermedia didattico, la valutazione al livello 3 considererà l'adeguatezza del contenuto e lo stile della comunicazione rispetto agli utenti e all'obiettivo educativo prefisso, oppure rispetto al modello insegnamento/apprendimento.

Il livello di classificazione proposto non è realmente cruciale per SUE, nel senso che punti di vista differenti potrebbero essere organizzati differentemente, senza influenzare le caratteristiche principali della

metodologia. La classificazione riflette soltanto un rapporto generale di specializzazione in relazione alla natura dell'applicazione, partendo da un punto di vista più generale (livello 1) per spostarsi verso punti di vista più specializzati (livello 2, e poi 3). Ciò che è importante in SUE è che ogni punto di vista richiede una valutazione ad hoc. Certamente, per assicurare un processo di valutazione esaustivo di una determinata applicazione, bisogna valutarne differenti prospettive.

Per ogni livello di valutazione, SUE prescrive l'esecuzione di due fasi: una fase preparatoria, eseguita una sola volta, con lo scopo di creare una struttura concettuale che sarà usata per eseguire le valutazioni di applicazioni di un certo tipo, e una fase di esecuzione, eseguita ogni volta che una specifica applicazione deve essere valutata.

Fase preparatoria

In questa fase devono essere prese un certo numero di decisioni e devono essere eseguite tre importanti attività:

- La scelta di un modello, per descrivere l'applicazione ad un determinato livello di valutazione.
- La definizione di un insieme di criteri da verificare durante la valutazione.
- La definizione di un insieme appropriato di Abstract Task, da applicare durante la fase di ispezione.

Il *modello* ha lo scopo di identificare e descrivere, in modo non ambiguo, i componenti di un'applicazione interattiva che costituiranno gli oggetti della valutazione al livello di valutazione scelto. In SUE, il termine "modello" è usato in senso largo e indica un insieme di concetti, strutture di rappresentazione, primitive e termini, che possono essere usati per costruire una descrizione di un'applicazione. SUE non prescrive che debba essere usato uno specifico modello. Ogni laboratorio di valutazione o gruppo di valutatori può scegliere il proprio modello favorito. SUE richiede, comunque, che la descrizione di un'applicazione debba essere relativa al punto di vista della valutazione. Per esempio, nelle valutazioni di usabilità condotte su diverse applicazioni ipermediali, è stato usato HDM (Hypermedia Design Model) [Gar94]per specificare al livello 2 proprietà strutturali e di navigazione, e caratteristiche dei media attivi. Potrebbero essere usati anche altri modelli, come quelli presentati in [Bie95].

I *criteri* o *principi* di usabilità sono gli attributi in cui l'usabilità può essere scomposta, così da essere meglio analizzata. Tali principi generali di usabilità devono essere scomposti in attributi più particolari, che riflettano le caratteristiche peculiari dello specifico livello di valutazione considerato.

Gli *Abstract Task* sono modelli (pattern) di valutazione che forniscono una descrizione dettagliata delle attività che devono essere eseguite dai valutatori durante l'ispezione. Sono formulati precisamente per mezzo di un pattern language, che usa il "vocabolario" del modello adottato per la descrizione dell'applicazione, per fornire un formato coerente di formulazione. Per una determinata prospettiva di valutazione, una lista di Abstract Task rappresenta una guida sistematica su come esplorare le caratteristiche dell'applicazione rilevanti per quella prospettiva, permettendo la standardizzazione dell'attività di ispezione. Molti valutatori sono eccellenti nell'analizzare solo certe caratteristiche di un'applicazione interattiva; spesso, però, essi trascurano altre caratteristiche strettamente dipendenti dalla categoria specifica di un'applicazione e/o dal dominio applicativo. Sfruttare un insieme di Abstract Task pronti per l'uso permette soprattutto a valutatori

con poca esperienza di ottenere buoni risultati. Diverse esperienze sul campo dimostrano che valutatori differenti, se usano la stessa lista degli Abstract Task, ottengono più facilmente conclusioni veloci e comparabili [Gar99].

Fase di esecuzione

La fase di esecuzione è eseguita ogni volta che un'applicazione specifica deve essere valutata ad uno specifico livello. Generalmente è costituita da un'ispezione eseguita da uno o più valutatori. Se qualcuno dei problemi di usabilità riscontrati durante l'ispezione crea ambiguità o necessita di ulteriori analisi, il valutatore può pianificare un test empirico con lo scopo di verificare il reale impatto di quei problemi sugli utenti finali.

In SUE, il progetto e l'esecuzione di un test utente segue gli approcci tradizionali, ma è più a basso costo ed efficace, in quanto guidato sistematicamente dai risultati dell'ispezione precedentemente eseguita. Più specificamente, ciò che è originale in SUE è che i task che l'utente esegue durante il test (detti concreti, perché relativi ad una specifica applicazione, in contrapposizione ai task astratti usati come livello guida dei valutatori durante l'ispezione) sono definiti dai valutatori dopo la fase di ispezione e sono strettamente correlati agli Abstract Task usati durante l'ispezione, così come ai problemi scoperti con essa.

Il contributo più significativo di SUE consiste principalmente nel modo in cui l'ispezione è eseguita. Durante l'ispezione i valutatori analizzano l'applicazione e specificano un modello (quando non è già disponibile) per descrivere l'applicazione da una prospettiva specifica. Il tipo di modello è scelto nella fase preparatoria. Il modello aiuta nell'individuare i differenti componenti dell'applicazione che saranno oggetto della valutazione. Poi, tenendo presente i criteri di usabilità definiti durante la fase preparatoria, i valutatori applicano gli Abstract Task e producono un report in cui vengono descritti i problemi scoperti. Con la guida fornita dagli Abstract Task, anche i report prodotti dai vari ispettori risultano più omogenei nella struttura e quindi più facilmente comparabili.

5. Valutazione di Interfacce WIMP

5.1 Modello OO-WIMP

Spesso si ha la necessità di modellare un'applicazione interattiva, cioè di costruire un modello che tramite un insieme di concetti o primitive, più o meno formali, e un insieme di meccanismi per collegare le primitive, fornisca una descrizione dell'applicazione che catturi la sua essenza, trascurando i dettagli inutili. L'uso di questa notazione formale o semi formale può essere motivato dalla necessità di disporre di uno strumento rigoroso per specificare talune caratteristiche dell'applicazione. A seconda del tipo di caratteristiche che si vogliono descrivere si possono usare alcuni modelli anziché altri. Per esempio, per descrivere i tratti peculiari degli ipermedia potremmo utilizzare l'Amsterdam Hypermedia Model [Har93] oppure il modello HDM (Hypermedia Design Model) [Gar93]. Per generare, invece, una descrizione di un'applicazione interattiva ad un livello generico, cioè per formalizzare le caratteristiche di presentazione dei sistemi interattivi, sarebbe utile disporre di un modello che contenga gli strumenti per descrivere il modo con cui le

informazioni e le funzionalità dinamiche vengono presentate all'utente. In letteratura esistono linee guida proposte dalle maggiori software house, esse sono:

- Human Interface Guidelines da Apple [App92];
- Common User Access Guidelines (CUA) da IBM [Ibm93];
- OpenLook Graphical User Interface Style Guidelines da Sun [Sun90];
- Motif Style Guide da OSF [Ope93];
- The Windows Interface Guidelines da Microsoft [Mic92].

Un tentativo di formalizzare un modello è stato in [Ibm93]. Noi abbiamo definito un nuovo modello per la descrizione degli aspetti di presentazione di applicazioni interattive ad un livello generale: il modello Object Oriented WIMP (Windows, Icons, Menus, Pointers) che descriviamo in questo capitolo. WIMP è descritto in letteratura come uno stile di interazione, infatti non definisce meccanismi per collegare insieme le sue primitive. La novità introdotta dal nuovo modello è la costruzione di una gerarchia di classi per suddividere e categorizzare gli oggetti utilizzati dal WIMP; ogni classe è caratterizzata da un diverso tipo di interazione sui suoi oggetti.

Secondo questo modello, l'interfaccia utente grafica di un'applicazione interattiva è costituita da un insieme di finestre. Le finestre sono gli oggetti primari dell'applicazione e rappresentano l'unico mezzo di comunicazione tra utente e il sistema, cioè tutte le interazioni avvengono attraverso una finestra. Le finestre si definiscono anche oggetti contenitore poiché contengono un qualunque oggetto secondario. Tra le finestre sono definite relazione del tipo "padre-figlio", cioè ogni finestra "è figlia" della finestra che l'ha generata. La generazione di una "finestra figlio" può avvenire per esempio selezionando un elemento a cui è collegata una finestra di dialogo di un menu contenuto nella finestra padre. Tra le finestre e gli oggetti secondari, chiamati anche widgets, esiste invece una relazione del tipo "contiene - è contenuto". Cioè la finestra contiene gli oggetti secondari mentre viceversa gli oggetti secondari sono contenuti nella finestra. Le relazioni su citate sono entrambe bidirezionali e la loro rappresentazione grafica è visibile in Figura 2.

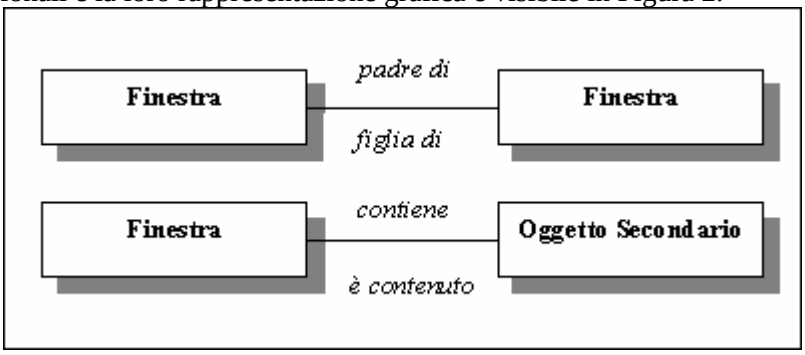

Figura 2 Relazioni tra classi in OO-WIMP

Scendendo in dettaglio, la classe Oggetto Secondario è una generalizzazione delle classi:

- Oggetto Reattivo
- Oggetto Passivo
- Oggetto Attivo

La classe Oggetto Reattivo raggruppa tutti gli oggetti che reagiscono a stimoli dell'utente. Questa classe è una generalizzazione delle classi:

- Bottone
- Slider
- Casella di testo
- Casella spin
- Icona
- Menu
- Barra di scorrimento

La classe Oggetto Passivo raggruppa tutti gli oggetti che non ammettono interazione con l'utente. Questa classe è una generalizzazione delle classi:

- Cornice
- Sfondo

La classe Oggetto Attivo, infine, raggruppa gli oggetti il cui stato varia nel tempo, anche questi oggetti non ammettono interazione con l'utente. Questa classe è una generalizzazione delle classi:

- Barra di progresso
- Tooltip
- Puntatore

La su descritta struttura è visibile in Figura 3.

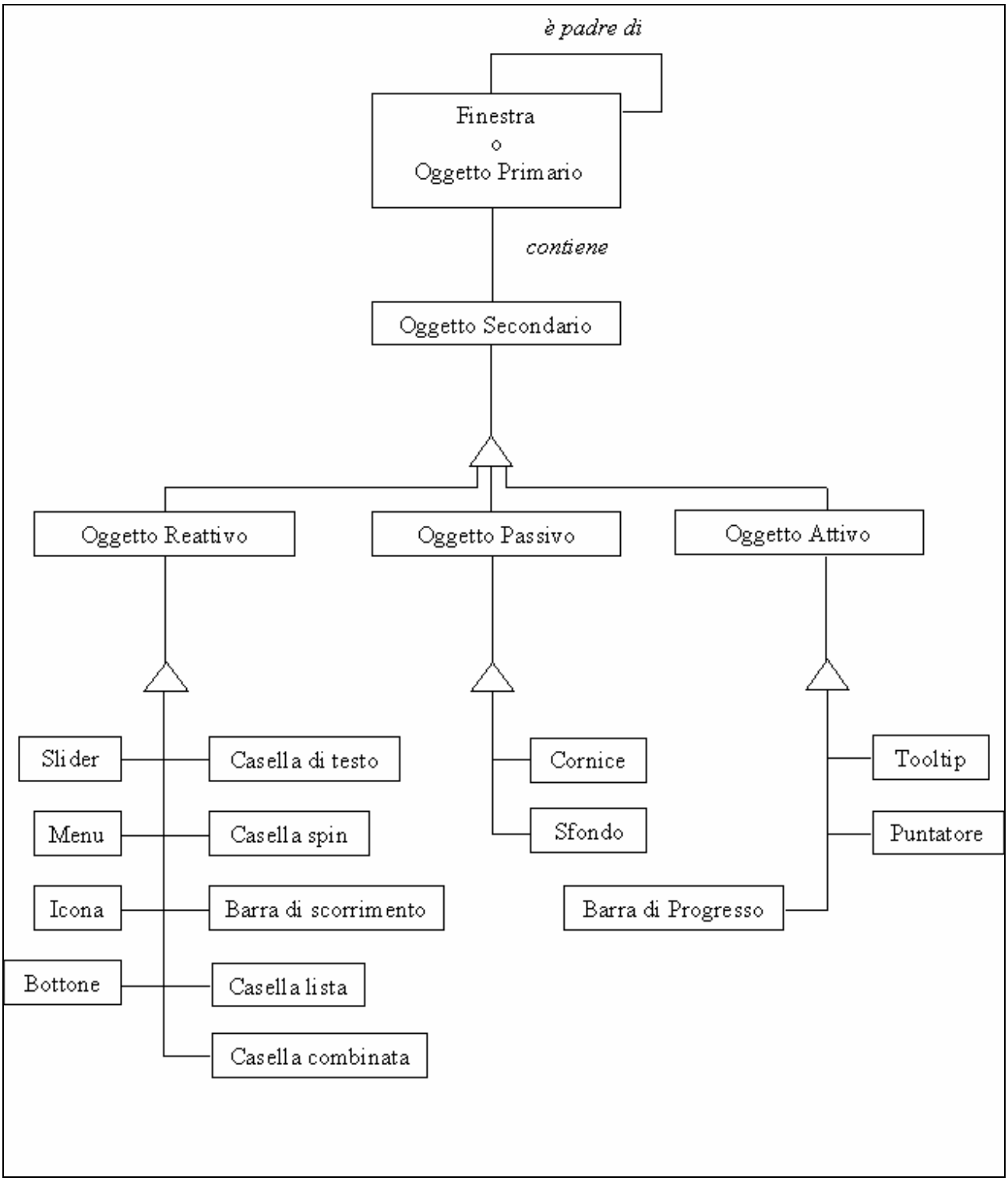

Figura 3 Struttura delle classi nel modello OO-WIMP con notazione OMT [Rum95]

6. Fase preparatoria di SUE per Interfacce WIMP

Il modello scelto per descrivere un'applicazione dal punto di vista della presentazione e dello stile di interazione è l'OO-WIMP. Perciò sarà questo modello che ci fornirà la terminologia per la definizione degli attributi di usabilità e conseguentemente dei task astratti. In questo modello l'oggetto principale è la finestra.

Il modello ci sembra il più opportuno, visto il grande successo che in questi ultimi anni hanno avuto le interfacce grafiche (Graphical User Interfaces) conosciute anche con il nome di “sistemi a finestre”.

La scelta dei criteri di usabilità, la seconda attività che compone la fase preparatoria, sarà oggetto di discussione della sezione 6.1. È un’attività molto importante che consiste nello scindere l’usabilità, che è un concetto molto astratto e generale, in una serie di attributi strettamente dipendenti dal particolare livello di valutazione.

Infine, nella sezione 6.2, saranno definiti i “task astratti” o AT (Abstract Task) individuati per questo livello, che costituiscono lo strumento, attraverso il quale il valutatore esperto individua i potenziali punti critici dell’applicazione. Questi task sono divisi, per oggetti di interesse, ovviamente relativi al modello OO-WIMP, per una migliore possibilità di individuazione. Ogni task è individuato univocamente da un codice.

6.1 Scelta degli Attributi di Usabilità

Per la valutazione sistematica delle applicazioni interattive a livello generale consideriamo prima due principi generali e universalmente conosciuti: l’efficienza e l’apprendibilità [Dix93, Nie93]. Ci concentriamo su questi due principi, perché sono quelli che possono più efficacemente essere valutati da una tecnica di ispezione, quale quella utilizzata da SUE. Per questo motivo, principi quali la soddisfazione dell’utente non sono stati presi in considerazione. Successivamente vediamo che cosa in particolare questi principi significhino quando valutiamo un’applicazione interattiva a livello generale. In questo modo si decompone l’efficienza e l’apprendibilità in sotto principi o criteri. A loro volta i criteri devono essere ulteriormente decomposti, o meglio specializzati, in sotto criteri che finalmente rappresentano gli attributi di usabilità usati per la valutazione. Questa gerarchia di principi, criteri e attributi di usabilità è mostrata nella Tabella 1.

6.1.1 Efficienza

L’efficienza riguarda quelle caratteristiche dell’applicazione che supportano l’utente mentre raggiunge, con successo e con un alto livello di produttività, il suo obbiettivo. Definisce cioè il livello di performance raggiungibile dopo che l'utente ha imparato il sistema. Per il livello generale questo principio è stato specializzato in quattro principali criteri: la disponibilità di controllo da parte dell’utente, la flessibilità, l’osservabilità dello stato e l’efficienza, a sua volta la flessibilità è stata specializzata in due attributi.

Disponibilità di controllo da parte dell’utente

Questo criterio si riferisce alla possibilità offerta all’utente di manipolare e quindi controllare con efficacia tutti gli oggetti del modello OO-WIMP. La disponibilità di controllo è soddisfatta, per esempio, se esistono funzionalità efficienti sulle finestre che permettano le operazioni elementari discusse (apertura, chiusura, scrolling, ecc.). Oppure se gli oggetti secondari contenuti nelle finestre, in particolare le caselle di testo, permettono il massimo controllo durante l’inserimento dell’input.

Osservabilità dello stato. Questo attributo riguarda la possibilità per l’utente di conoscere, in ogni istante, lo stato del sistema. Lo strumento utilizzato dall’applicazione per estrinsecare questa osservabilità sono le indicazioni visive. La maggior parte delle applicazioni usa una particolare barra, quella di stato, per

evidenziare tutti gli indicatori dei modi. Un esempio in cui questo attributo è soddisfatto è quando un bottone di comando modifica la sua shape quando è selezionato; questa indicazione visiva indica all'utente che lo stato del sistema si è modificato in virtù dell'avvenuta interazione. Altro esempio di indicatore di modo è quello presente nei word-processor: durante la digitazione dei caratteri ci si può trovare in "modo inserimento" o "modo sovrascrittura". Questo attributo prescrive anche che, per esempio, un oggetto reattivo sia visivamente distinguibile da uno passivo.

Principi Generali	Criteri	Attributi
Efficienza	Disponibilità di controllo da parte dell'utente	Osservabilità dello stato
	Flessibilità	Personalizzazione
		Molteplicità di meccanismi di interazione
Apprendibilità	Coerenza	Coerenza strutturale
		Coerenza di posizione
		Coerenza di comportamento
	Prevedibilità (Affordance)	Regolarità visiva
		Conformità alle notazioni dell'utente
	Visibilità	Localizzabilità
		Riconoscibilità
		Visibilità delle azioni

Tabella 1 Gerarchia degli attributi di usabilità per il livello 1 di SUE

Flessibilità

La flessibilità quale sotto principio dell'efficienza è stata decomposta in due attributi: la personalizzazione e la molteplicità di meccanismi di interazione.

Personalizzazione. Esprime il fatto che l'applicazione fornisce all'utente particolari funzionalità, che gli permettono di adattare i parametri dell'interfaccia alle sue caratteristiche. Se questo attributo è verificato l'interfaccia presenta delle caratteristiche di flessibilità e non di rigidezza. Un esempio è quando l'utente viene messo in grado, tramite particolari funzionalità dell'applicazione, di modificare la struttura di un menu adattandolo alle sue caratteristiche.

Molteplicità di meccanismi di interazione. Questo attributo è soddisfatto quando l'applicazione, ed in particolare la sua interfaccia, fornisce all'utente una pluralità di modi di interazione. Più questi modi sono numerosi più è alta la probabilità che l'utente si trovi a sua agio con uno di essi. Se ciò accade sicuramente l'applicazione verrà utilizzata in modo più efficiente.

6.1.2 Apprendibilità

L’apprendibilità riguarda “le caratteristiche di un’applicazione interattiva che permettono all’utente novizio di usarla la prima volta, e che gli permettono di ottenere il massimo livello di performance” [Dix93]. In altre parole la facilità di apprendere il comportamento dell’applicazione. Questo principio è stato decomposto in tre criteri generali: coerenza, prevedibilità e visibilità, i quali a loro volta sono stati specializzati per il livello di presentazione di SUE.

Coerenza

La coerenza misura la regolarità dell'applicazione, e molti autori la considerano uno dei più importanti criteri di usabilità; implica che elementi concettualmente simili vengano trattati in modo simile, mentre elementi concettualmente diversi debbano essere trattati in modo diverso. La coerenza influenza la prevedibilità, ed è particolarmente rilevante per gli utenti che utilizzano molto l'applicazione. Molto è stato scritto sulla necessità di progettare applicazioni interattive coerenti. Da un lato molti hanno scritto circa la difficoltà di capire che cosa è la coerenza e come applicarla efficientemente [Gru89]. La coerenza può avere molte differenti interpretazioni e molti differenti attributi. La coerenza di un criterio può essere in conflitto con la coerenza di un altro. Nondimeno, la coerenza è uno dei più significativi fattori che influenza l'usabilità delle applicazioni interattive.

Coerenza strutturale. Prescrive che tutti gli oggetti appartenenti ad una classe di OO-WIMP abbiano una struttura simile. Per struttura intendiamo tutto ciò che caratterizza gli oggetti di una classe. Significa, per esempio, che tutti i bottoni di comando devono avere forma, dimensione e colore simile in un’applicazione. Oppure che, le icone testuali che descrivono un bottone siano sempre definite adottando una terminologia uniforme. Stesso discorso nei titoli dei menu e negli item degli stessi.

Coerenza di posizione. Prescrive che gli oggetti che sono contenuti nelle finestre siano localizzati sempre nella medesima posizione. Per esempio, ogni bottone deve trovarsi sempre nel medesimo punto nella finestra in tutta l'applicazione. Questo tipo di coerenza può anche essere interpretata affermando che tutti gli oggetti secondari, contenuti in una finestra, con una qualche relazione, devono apparire in gruppo e devono occupare sempre una stessa area. Si parla in questo caso di coerenza poiché, l’esistenza di una qualche relazione tra gli oggetti, gli rende “simili” e perciò devono occupare posizioni vicine tra loro (posizioni coerenti).

Coerenza di comportamento. Questo tipo di coerenza prescrive che gli oggetti simili si comportino in modo simile; per comportamento simile intendiamo, per esempio, che l’interazione durante la selezione o il puntamento dei bottoni, in un’applicazione, avvenga sempre nel medesimo modo.

Prevedibilità

La prevedibilità esprime l'abilità dell'utente nell'anticipare il risultato di una futura interazione basandosi sulla conoscenza dell’interazione passata. La prevedibilità è rafforzata dalla visibilità. Mentre quest’ultima richiede che l'utente possa capire i comportamenti dell’applicazione già al primo uso, la prevedibilità

richiede che l'utente riesca a prevedere anche comportamenti complessi dopo una quantità di tempo significativa trascorsa con l'applicazione, e possa usarla efficientemente.

Regolarità visiva. Prescrive che la disposizione degli oggetti secondari all'interno delle finestre avvenga in modo regolare, cioè facendo in modo che la collocazione di tutti gli oggetti dia equilibrio e simmetria visiva alla finestra. Questa regolarità conferisce all'applicazione un aspetto non caotico e organizzato, con un conseguente impatto positivo sull'apprendibilità dell'utente.

Conformità alle notazioni dell'utente. Questo attributo prescrive che l'interfaccia delle applicazioni sia quanto più possibile conforme alle notazioni a cui l'utente è abituato. Se questo attributo è violato l'utente avverte una sensazione di disagio durante l'interazione con l'interfaccia, questo pregiudicherà la facilità di apprendimento delle funzionalità del sistema, ed in particolare la prevedibilità.

Visibilità

La visibilità misura l'abilità dell'utente nel capire il significato e lo scopo di ciò che gli viene presentato per la prima volta. Paragonato ai criteri di usabilità di Nielsen [Nie93], la visibilità è direttamente collegata all'apprendibilità, alla memorizzabilità, all'efficienza e alla soddisfazione. È un criterio particolarmente importante per utenti inesperti e per utenti che utilizzano l'applicazione per un breve periodo, perché gli permette di avere successo, e trarre soddisfazione nel loro primo tentativo. La visibilità è particolarmente importante per questi utenti anche perché essi sono condizionati fortemente dalla prima impressione che hanno del sistema; questa riguarda, ovviamente, le caratteristiche di presentazione.

Localizzabilità. Definisce la facilità di individuare sul lay-out delle finestre gli oggetti secondari, per esempio l'utente intuisce l'esistenza nelle finestre degli oggetti con cui può interagire. Questo processo di individuazione può essere portato a termine dall'utente con tanta maggiore efficacia quanto più, per esempio, l'applicazione soddisfa il feed-back per gli oggetti puntati.

Riconoscibilità. Definisce la facilità di riconoscere un oggetto come appartenente ad una determinata classe di oggetti del modello OO-WIMP. Per esempio questo criterio prescrive che gli oggetti abbiano un aspetto di presentazione che permetta all'utente di distinguere un bottone da uno slider. Un altro esempio in cui questo attributo è soddisfatto è quando agli oggetti sono associati degli help in linea che aiutano l'utente a riconoscere le funzionalità dello stesso. Riconoscibilità significa anche che l'utente può immediatamente e con facilità capire, grazie alla presenza di aiuti descrittivi, il significato di un item in un menu mentre lo punta con il mouse, oppure l'oggetto associato ad un'icona grazie all'etichetta descrittiva.

Visibilità delle azioni. Prescrive che le azioni disponibili in una finestra siano visibili e che non ci siano oggetti non visibili con i quali si può interagire. Un esempio potrebbe essere quello in cui una barra di scorrimento supporta l'asta di scorrimento, cioè permette all'utente di utilizzare la funzionalità propria di questo strumento, ma questa asta non è visibile sullo schermo; oppure questo attributo è violato quando un click in una determinata posizione della finestra modifica lo stato del sistema senza che in quel punto sia distinguibile visivamente un oggetto reattivo.

6.2 Task Astratti

I task astratti, o AT (Abstract Task), descrivono le attività operazionali che, durante l'ispezione, i valutatori devono eseguire per individuare eventuali difetti di usabilità. Si usa il termine “astratti”, poiché i task sono formulati indipendentemente dalla particolare applicazione, essi si riferiscono a categorie di oggetti più che ad uno specifico oggetto di una specifica applicazione. Un task astratto è descritto dai seguenti elementi:

- il *Codice di identificazione* e il *Titolo*;
- l'*Oggetto di Interesse* su cui il task si concentra;
- l'*Intento*: una breve frase che spiega il fine, lo scopo preciso del task astratto;
- la *Descrizione delle attività* che il valutatore deve eseguire;
- la lista dei *Criteri di usabilità*, che l'oggetto di interesse deve soddisfare;
- opzionalmente, un *Commento*.

La terminologia usata per la definizione dell'Oggetto di Interesse e per la Descrizione delle Attività è basata sul modello OO-WIMP.

6.2.1 Esecuzione per Induzione

Per quel che riguarda l'esecuzione di una valutazione “esaustiva” e “sicura”, dei task astratti, ed in particolare di quelli che verificano la coerenza, questi devono essere eseguiti su tutte le istanze nella categoria di oggetti di interesse dei task. Alcune volte può non essere facile portare a termine questo compito, specialmente per grandi applicazioni. In conseguenza di ciò, i task astratti sono eseguiti per induzione: durante la sessione di valutazione, essi sono applicati solo ad un campione limitato di oggetti, e i risultati sono generalizzati. Scegliere questo campione di oggetti è un compito molto difficile. Infatti, c'è il rischio di considerare degli oggetti nelle applicazioni che non hanno problemi, lasciando fuori altri oggetti più interessanti dal punto di vista della valutazione. I valutatori possono eseguire la valutazione senza scegliere a priori un tale campione; possono, infatti, iniziare l'esecuzione dei task astratti su alcuni oggetti a caso. Quindi, continueranno a selezionare oggetti ed a eseguire i task astratti su di loro, con l'intento di scoprire ulteriori problemi. Se non troveranno delle incoerenze o irregolarità, interromperanno la ricerca dopo aver valutato un numero sufficiente di oggetti che non presentano problemi. L'individuazione di questo numero è anche cruciale: esso è molto soggettivo e dipende dalle dimensione dell'applicazione.

7. Esempi di applicazione di Abstract task

In questa sezione vengono forniti tre esempi di applicazione di AT per la valutazione di due CD-Rom commerciali: Le Louvre, una guida ipermediale del famoso museo di Parigi, e Interactive Europe, una guida turistica ipermediale per itinerari e località europee. Gli AT utilizzati sono riportati in appendice.

1) Problema individuato per mezzo dell'AT OS3: “Coerenza della posizione degli oggetti secondari sul lay-out”.

   Oggetto di Interesse: oggetto secondario.

   Intento: verificare la coerenza della posizione degli oggetti secondari sul lay-out della finestra.

   Descrizione delle attività: per ogni finestra nell’applicazione:

1. Individuare tutti gli oggetti secondari che essa contiene;
2. Valutare la coerenza della loro posizione tra le finestre; cioè verificare che ogni oggetto sia localizzato sempre nella medesima posizione nella finestra in tutta l'applicazione.

Criteri di usabilità: coerenza (coerenza di posizione).

Risultato: l'applicazione di questo task ha messo in evidenza una incoerenza di posizione tra due finestre contenenti entrambe un menu iconico. L'icona in questione è quella di accesso al menu principale. Nella finestra con titolo "Luoghi della natura" questa icona appare nell'angolo superiore destro, mentre nel menu iconico contenuto nella finestra "Le montagne", l'icona è posizionata nell'angolo superiore sinistro. Ci troviamo, quindi, di fronte al caso in cui due oggetti uguali occupano, in diverse finestre, posizioni diverse. Nella Figura 4 sono visualizzate le finestre in esame; la freccia rossa in entrambe indica l'icona in posizione incoerente.

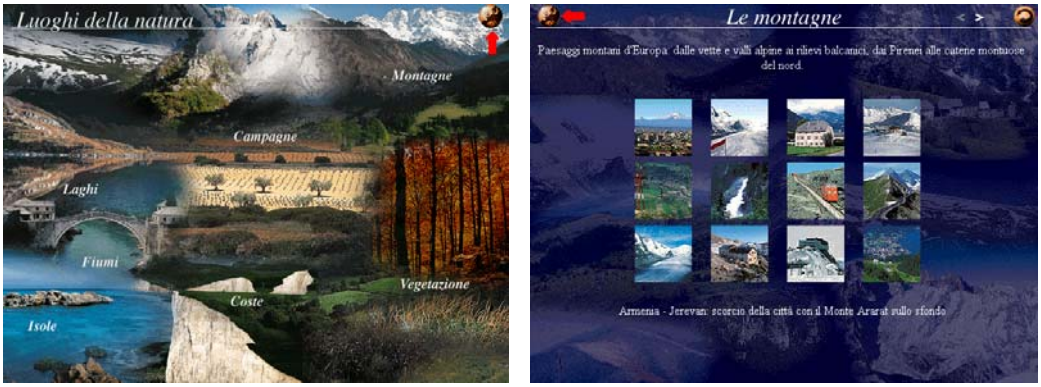

Figura 4 Incoerenza di posizione in "Europa Guida Interattiva"

2) Problema individuato per mezzo dell'AT OS-OR-ICO2: "Facilità di riconoscimento delle icone".

Oggetto di Interesse: icona.

Intento: valutare l'efficacia, definita in termini di facilità, per l'utente, di riconoscere ciò a cui un'icona si riferisce.

Descrizione delle attività: per ogni icona presente nell'applicazione; valutare la facilità nel riconoscere l'oggetto o l'azione ad essa associata.

Criteri di usabilità: visibilità (riconoscibilità).

Risultato: questo task ci ha permesso di individuare delle icone nell'applicazione che non soddisfano il criterio di usabilità che è la riconoscibilità. Le icone in esame sono quelle posizionate in basso a sinistra nelle finestre che contengono le informazioni sui quadri e in quelle sui palazzi del museo. Questi oggetti reattivi presentano una carenza di riconoscibilità dovuta alla mancanza di un'etichetta testuale di accompagnamento alla componente grafica dell'icona stessa. Queste oggetti presentano, al tempo stesso, un comportamento anomalo in quanto solo dopo essere state selezionate mostrano l'etichetta di descrizione. Questo è esattamente opposto al comportamento ottimale che prevede che all'utente sia data la possibilità di anticipare l'oggetto di arrivo associato all'icona, in modo da ottimizzare la ricerca delle informazioni.

3) Problema individuato per mezzo dell'AT OS-OR-ICO1: "Coerenza nello schema di progetto delle icone".

Oggetto di Interesse: icona.

Intento: verificare la coerenza nello schema di progetto delle icone, cioè verificare che tutte le icone nell'applicazione siano progettate in base a un singolo schema di progetto.

Descrizione delle attività: per ogni icona presente nell'applicazione; verifichiamo la presenza di un unico "modo" con cui essa rappresenta e descrive un oggetto o un'azione.

Criteri di usabilità: coerenza (coerenza strutturale), visibilità (riconoscibilità).

Risultato: Il task in esame ha evidenziato una incoerenza nello schema di progetto delle icone. Tutte le icone nell'applicazione sono formate da una "picture" e da una parte descrittiva. Tutte tranne l'icona la cui selezione porta l'utente nella finestra contenente il menu principale. In questa icona non è presente alcuna parte testuale che indichi all'utente inesperto la funzionalità legata a quell'oggetto. Questa diversità di struttura dell'oggetto reattivo in esame è rappresentata in Figura 5. In questa figura la prima icona a sinistra rappresenta l'oggetto incoerente rispetto agli altri tre che seguono. Per l'eliminazione di questa incoerenza bastava utilizzare come parte descrittiva la voce "Menu".

Figura 5 Icone con struttura incoerente in "Europa Guida Interattiva"

Appendice

Task Astratti sulle Finestre

OP-FIN1

Titolo: “Coerenza nel progetto del lay-out delle finestre”.
Oggetto di Interesse: finestra.
Intento: verificare la coerenza nel lay-out delle finestre all’interno dell’applicazione.
Criteri di usabilità: coerenza (coerenza strutturale).
Descrizione delle attività: per ogni finestra nell'applicazione; verificare la coerenza, nella struttura, delle finestre tra loro in relazione; cioè verificare che le finestre correlate abbiano sempre tipi standard di informazioni localizzate nella medesima posizione; per esempio il titolo della finestra, la barra di stato, i campi per l'inserimento dell'input, bottoni e icone devono apparire, coerentemente, nello stesso punto della finestra nell'applicazione. Non solo, queste informazioni devono anche essere formattate coerentemente.

Commento: Come suggerisce uno studio [Tei83], questo tipo di incoerenze possono incrementare il tempo di ricerca delle informazioni mentre si naviga nell'applicazione. La coerenza, invece, faciliterà l'apprendimento e renderà il sistema facile da usare per l'utente. È una buona idea guidare la fase di progetto del lay-out delle finestre in un sistema da una o più “finestre template”, le quali esprimono le strutture base da utilizzare poi nel progetto della finestra in particolare. La coerenza dentro un'applicazione è sempre importante, e spesso è importante anche raggiungere la coerenza tra applicazioni che ci si aspetta siano usate dagli utenti.

OP-FIN2

Titolo: “Proprietà stand-alone di una finestra”.
Oggetto di Interesse: finestra.
Intento: verificare che ogni finestra nell’applicazione sia “stand-alone”, contenga cioè tutta un'idea, un task o un contesto.
Criteri di usabilità: disponibilità di controllo da parte dell’utente.
Descrizione delle attività: per ogni finestra presente nell'applicazione; valutare la proprietà stand-alone; cioè controllare se tra le finestre esistono riferimenti incrociati che costringano l’utente a ricordare informazioni contenute in altre finestre.

Commento: questa proprietà è particolarmente importante poiché l'utente non dovrebbe mai preoccuparsi di cercare riferimenti incrociati tra le finestre, e non dovrebbe mai ricordare dati che sono contenuti in altre finestre. Per esempio se un utente deve riempire una form per la stampa di un file, il progettista deve fare in

modo che il file venga selezionato da un indice all'interno della stessa finestra e non costringere l'utente a “ricordare” il nome. In questo modo è il sistema che “ricorda” anziché l'utente.

OP-FIN3

Titolo: “Utilizzo di spazio bianco”.
Oggetto di Interesse: form.
Intento: verificare il corretto utilizzo degli spazi bianchi nelle form.
Criteri di usabilità: visibilità (localizzabilità).
Descrizione delle attività: per ogni form nell’applicazione; verificare il rapporto tra lo spazio vuoto e lo spazio occupato da oggetti.

Commento: il termine spazio bianco è usato come termine generale per indicare aree vuote in una finestra, cioè, spazio non occupato da oggetti testuale o grafici. Lo spazio bianco può essere bianco, giallo o di un qualunque altro colore, dipendentemente dal colore di sfondo della finestra. Gli spazi bianchi, in una finestra, sono un ottimo strumento per creare bilanciamento e simmetria. In una form la presenza di un numero eccessivo di elementi, può creare un senso di affollamento e di complessità che confonde l’utente, non permettendogli di individuare rapidamente e facilmente un controllo specifico, pregiudicando in questo modo l’efficienza nell’utilizzo della form. Da questo punto di vista l’utilizzo di aree vuote rappresenta un ottimo strumento per evidenziare e contraddistinguere in modo chiaro i vari elementi, permettendo nel contempo velocità nella scansione della form.
Studi in questo campo hanno individuato una misura oggettiva delle aree vuote di una form: la densità. Galitz in [Gal94] afferma che la densità ideale è compresa tra 25 e 30. In altre parole, il 70-75 % di una finestra deve essere vuota. La densità viene calcolata contando il numero totale di caratteri su una form e dividendo per il numero totale di caratteri visualizzati. Comunque questo non è un discorso assoluto. Infatti gli utenti esperti di particolari applicazioni, avendo famigliarità con la form preferiscono la più alta quantità di informazione visualizzabile poiché questo significa azzeramento del tempo speso in operazioni di ricerca delle informazioni non visualizzate. Per questi utenti la densità ideale varia tra 70 e 80 [Gal94].

OP-FIN4

Titolo: “Complessità visiva di una form”.
Oggetto di Interesse: form.
Intento: verificare se il numero e il tipo di oggetti localizzati in una form è adatto rispetto al suo uso.
Criteri di usabilità: efficienza.
Descrizione delle attività: per ogni form nell’applicazione:

1. Verificare la complessità visiva in termini di numero e tipo di oggetti contenuti.
2. Verificare se l’insieme di oggetti è appropriato al particolare utilizzo della form; controllare cioè che non esistano problemi di leggibilità o riempibilità della stessa, nonché ridondanza.

Commento: una comune ma errata assunzione e che un esistente modulo su carta è convertibile direttamente in una buona form su schermo. Se questo avvenisse si potrebbero riscontrare problemi di leggibilità o riempibilità, quest'ultima intesa come facilità di inserimento dell'informazione nella form; nonché ridondanza. Ciò si ripercuoterebbero negativamente sull'efficienza e quindi sull'usabilità dell'applicazione. Invece il modulo cartaceo deve essere opportunamente analizzato puntando a limitare la complessità della form su schermo.

OP-FIN5

Titolo: "Disponibilità di controllo per le operazioni elementari delle finestre".
Oggetto di Interesse: finestra.
Intento: verificare se nell'applicazione esistono delle funzionalità che permettano all'utente il massimo controllo sugli oggetti più importanti del modello OO-WIMP.
Criteri di usabilità: disponibilità di controllo da parte dell'utente.
Descrizione delle attività: per ogni finestra nell'applicazione; verificare la disponibilità di meccanismi che permettano di eseguire con efficienza su questi oggetti operazioni quali: il ridimensionamento, la chiusura, lo scrolling, ecc.

Commento: delle volte accade che l'utente abbia la necessità di interagire contemporaneamente con più finestre; a causa della frequenza con cui questo accade è molto importante che questo task venga eseguito con successo.

Task Astratti sugli Oggetti Secondari

OS1

Titolo: "Regolarità visiva nella disposizione degli oggetti secondari sul lay-out".
Oggetto di Interesse: oggetto secondario.
Intento: verificare l'ordine visivo nella disposizione degli oggetti sul lay-out di una finestra.
Criteri di usabilità: prevedibilità (regolarità visiva).
Descrizione delle attività:

1. Individuare tutti gli oggetti secondari contenuti nella finestra;
2. Valutare l'ordine visivo delle loro posizioni; controllare cioè la regolarità nella disposizione degli oggetti sul lay-out della finestra.

Commento: la regolarità nella disposizione crea un senso di armonia nell'applicazione. La mancanza di regolarità causa invece confusione e conferisce all'applicazione un aspetto caotico e disorganizzato, con una conseguente influenza negativa sull'utente ed in particolare sulla sua soddisfazione e di conseguenza sulla facilità di apprendimento.

OS2

Titolo: “Appropriatezza della posizione degli oggetti secondari sul lay-out”.
Oggetto di Interesse: oggetto secondario.
Intento: verificare se la posizione dell’oggetto è ottimale in base all'importanza dello stesso.
Criteri di usabilità: efficienza.
Descrizione delle attività: per ogni finestra:

1. Individuare tutti gli oggetti che essa contiene;
2. Valutare l'adeguatezza della loro posizione; cioè verificare che le posizioni può prominenti siano occupate dagli elementi più importanti o utilizzati con maggiore frequenza, in modo che siano facilmente identificabili, e assegnare invece le posizioni meno prominenti agli elementi meno importanti.

Commento: in genere gli elementi contenuti in una finestra non hanno tutti la stessa importanza. Nella maggior parte delle lingue la lettura viene eseguita da sinistra a destra e dall'alto verso il basso della pagina. In modo analogo, l'attenzione dell'utente sullo schermo del computer è attirata innanzitutto dalla parte in alto a sinistra dello schermo. In questa posizione è pertanto consigliabile includere l'elemento più importante. Se le informazioni contenute in una finestra sono, ad esempio, relative ad un cliente, il campo del nome deve essere posizionato in un punto in cui possa essere individuato immediatamente. Bottoni quali “OK” o “Avanti” devono invece essere posizionati nella parte in basso a destra dello schermo, in quanto sono in genere utilizzati solo dopo il completamento delle operazioni nella finestra.

OS3

Titolo: “Coerenza della posizione degli oggetti secondari sul lay-out”.
Oggetto di Interesse: oggetto secondario.
Intento: verificare la coerenza della posizione degli oggetti secondari sul lay-out della finestra.
Criteri di usabilità: coerenza (coerenza di posizione).
Descrizione delle attività: per ogni finestra nell’applicazione:

1. Individuare tutti gli oggetti secondari che essa contiene;
2. Valutare la coerenza della loro posizione tra le finestre; cioè verificare che ogni oggetto sia localizzato sempre nella medesima posizione nella finestra in tutta l’applicazione.

Commento: questo task serve per scoprire incoerenze riguardanti il posizionamento non omogeneo degli oggetti sul lay-out; l'incoerenza risiede nel fatto che oggetti uguali occupano posizioni diverse. Ogni oggetto dovrebbe essere localizzato sempre nella medesima posizione sullo schermo in tutta l'applicazione, se ciò non avviene possono nascere problemi di usabilità, in quanto l'utente è disorientato dalla mobilità sul lay-out dell'oggetto con cui deve interagire. Questo provoca, primo una perdita di tempo nel cercare l’oggetto, per esempio un bottone, nella più favorevole delle ipotesi, secondo può portare alla scelta involontaria di un oggetto, per esempio la pressione di un bottone diverso da quello che si vorrebbe, solo perché è stato messo

al posto di un altro. Ovviamente questo tipo di incoerenza influenza anche la prevedibilità. L'utente, in altre parole, non è in grado di prevedere la posizione di un oggetto, associato a una determinata operazione, sul lay-out di differenti finestre in diversi contesti.

OS4

Titolo: “Raggruppamento logico degli oggetti secondari”.
Oggetto di Interesse: oggetto secondario.
Intento: verificare l'esistenza del raggruppamento logico degli oggetti secondari in base a funzione o correlazione.
Criteri di usabilità: visibilità (localizzabilità), coerenza (coerenza di posizione).
Descrizione delle attività: per ogni finestra nell’applicazione:

1. Individuare tutti gli oggetti secondari con una qualunque relazione logica;
2. Verificare che esistano delle indicazioni visive che mettono in evidenza questa relazione;

Commento: quando in un'applicazione esiste la necessità che su uno stesso schermo appaiano molti oggetti, sorge il problema di scegliere la posizione migliore per questi oggetti rispettando nello stesso tempo i criteri di usabilità. Appare molto utile, in questi casi raggruppare in posizioni vicine oggetti che hanno una qualche relazione semantica, per esempio un gruppo di bottoni che gestiscono la navigazione tra i nodi di un'ipermedia. I bottoni per operazioni di spostamento, per esempio, avendo funzioni correlate, dovrebbero essere raggruppati in modo visivo anziché inseriti in punti diversi nella finestra. Lo stesso principio vale anche per le informazioni. I campi di nomi e indirizzi, ad esempio, dovrebbero essere in genere raggruppati insieme, in quanto sono strettamente correlati. In molti casi è possibile utilizzare cornici per evidenziare ed enfatizzare ulteriormente la relazione tra gli oggetti. Questo tipo di organizzazione aiuta l'utente a localizzare con facilità la posizione degli oggetti nell'applicazione; sicuramente aumenta la velocità di ricerca di un determinato oggetto, ed è tanto più efficace quanto più alto è il numero di oggetti sullo schermo. Questa rapidità ha un riverbero positivo sull'efficienza raggiungibile con l'applicazione, poiché se più velocemente s'individuano icone e bottoni sicuramente più velocemente si porterà a termine il task. In ultima analisi questo tipo di progetto del lay-out è sicuramente più coerente rispetto al posizionamento casuale, o quantomeno non per gruppi, poiché fa corrispondere ad oggetti con funzionalità simili posizioni vicine tra loro.

OS5

Titolo: “Presentazione degli oggetti reattivi ma momentaneamente passivi”.
Oggetto di Interesse: oggetto secondario.
Intento: verificare che esista una distinzione visiva tra un oggetto reattivo ed uno passivo.
Criteri di usabilità: osservabilità dello stato.
Descrizione delle attività: verificare che gli oggetti reattivi ma in un certo contesto passivi, abbiano caratteristiche di presentazione diverse rispetto a quelli reattivi.

Commento: questo task serve per scoprire la carenza di distinguibilità visiva tra un oggetto reattivo ed uno passivo. Questa distinzione a livello di presentazione di un oggetto appare ovvia in quanto, se due oggetti hanno diverse proprietà di interazione allora devono apparire con diverse forme. Se ciò è supportato nell'applicazione l'efficienza è favorita; in quanto si evitano le interazioni non corrette.

Task Astratti sugli Oggetti Reattivi

OS-OR1

Titolo: "Esistenza del feed-back durante il puntamento degli oggetti reattivi".
Oggetto di Interesse: oggetto reattivo.
Intento: verificare la facilità di individuazione delle oggetti reattivi nell'applicazione.
Criteri di usabilità: visibilità (localizzabilità e visibilità delle azioni).
Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; valutare l'esistenza del feed-back durante il puntamento.

Commento: è auspicabile che l'applicazione, al fine di rendere percettibili gli oggetti reattivi dell'interfaccia contenga dei meccanismi che li mettano in evidenza durante il puntamento, con un cambiamento di aspetto. In altre parole l'oggetto reattivo deve essere "sensibile" al puntatore. Per esempio, nel caso di puntamento di un bottone, il cambiamento di aspetto potrebbe consistere nel fare in modo che il bottone s'illumini o cambi colore. I feed-back rappresentano informazione "spedita indietro" all'utente durante l'interazione. Questa informazione segnala di prestare attenzione in quanto si può interagire con la regione sotto il puntatore. Sono il mezzo con cui chi utilizza l'applicazione percepisce il grado di controllo sulla stessa; se in un'applicazione i "ritorni visivi" dovessero essere scarsi, verrebbe pregiudicato questo controllo. La loro assenza si ripercuote sulla visibilità. Essi, infatti, rappresentano uno stimolo all'utente per intuire la presenza di oggetti con determinate caratteristiche interattive. I feed-back portano benefici a qualunque tipo di utente, sia esso esperto sia inesperto, in particolare, i vantaggi maggiori sono per l'utente inesperto che da questi trae gran sicurezza, durante l'uso dell'applicazione, per portare a termine il suo lavoro. Se il feed-back sugli oggetti reattivi è ottimale, l'applicazione ha molto sviluppata la visibilità degli oggetti reattivi. Questo si ripercuote positivamente sulla facilità, per l'utente, di localizzare gli oggetti con cui dovrà interagire.

OS-OR2

Titolo: "Efficacia del feed-back durante il puntamento degli oggetti reattivi".
Oggetto di Interesse: oggetto reattivo.
Intento: valutare l'efficacia del feed-back durante il puntamento degli oggetti reattivi.
Criteri di usabilità: visibilità (localizzabilità e visibilità delle azioni).

Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; valutare l'efficacia del feed-back durante il puntamento.

Commento: questo task è una continuazione del task OS-OR1; non solo è necessaria l'esistenza di feed-back durante il puntamento, ma è altrettanto importante che questa sia efficace.

OS-OR3

Titolo: "Coerenza del feed-back durante il puntamento degli oggetti reattivi".
Oggetto di Interesse: oggetto reattivo.
Intento: verificare la coerenza dei cambiamenti di forma sugli oggetti simili dell'interfaccia durante il puntamento.
Criteri di usabilità: coerenza (coerenza di comportamento).
Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; valutare la coerenza del feed-back durante il puntamento.

Commento: lo scopo di questo task e quello di scoprire le incoerenze e, di conseguenza, i comportamenti non prevedibili nel feed-back durante il puntamento degli oggetti reattivi. Per esempio può accadere che in un'applicazione per alcuni bottoni ci sia il ritorno visivo durante il puntamento e per altri no, oppure che il cambiamento di aspetto, interessi alcune volte bottone comprensivo di label altre volte solo il bottone.

OS-OR4

Titolo: "Esistenza del feed-back durante la selezione degli oggetti reattivi".
Oggetto di Interesse: oggetto reattivo.
Intento: valutare l'esistenza del feed-back durante la selezione degli oggetti reattivi.
Criteri di usabilità: osservabilità dello stato.
Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; valutare l'esistenza del feed-back durante la selezione.

Commento: questo task ha l'intento di scoprire se gli oggetti reattivi modificano la loro shape quando sono selezionati dall'utente. Questa indicazione visiva è importante, poiché avvisa l'utente che lo stato del sistema si è modificato in virtù dell'interazione verificatasi.

OS-OR5

Titolo: "Efficacia del feed-back durante la selezione degli oggetti reattivi".
Oggetto di Interesse: oggetto reattivo.
Intento: valutare l'efficacia del feed-back durante la selezione degli oggetti reattivi.
Criteri di usabilità: osservabilità dello stato.

Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; valutare l'efficacia del feed-back durante la selezione.

Commento: se questo task viene eseguito con successo si rafforza la visibilità dello stato del sistema, cioè il sistema è più efficiente da usare rispetto ad una situazione in cui il feed-back non è efficace.

OS-OR6

Titolo: "Coerenza del feed-back durante la selezione degli oggetti reattivi".
Oggetto di Interesse: oggetto reattivo.
Intento: verificare la coerenza dei cambiamenti di forma sugli oggetti simili dell'interfaccia durante la selezione.
Criteri di usabilità: coerenza (coerenza di comportamento).
Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; valutare la coerenza del feed-back durante la selezione.

Commento: lo scopo di questo task è di verificare la coerenza del feed-back durante la selezione degli oggetti interattivi. Costituisce una continuazione dei task OS-OR4 e OS-OR5.

OS-OR7

Titolo: "Descrizione on-line di un oggetto reattivo durante il suo puntamento".
Oggetto di Interesse: oggetto reattivo.
Intento: verificare la facilità, per l'utente, di riconoscere le funzionalità associate ad un oggetto.
Criteri di usabilità: visibilità (riconoscibilità).
Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; verificare la presenza di descrizioni on-line, o più brevemente "micro-help", durante il puntamento dell'oggetto.

Commento: se questo task viene eseguito senza problemi, l'applicazione si presta ad essere utilizzata con successo e soddisfazione anche dagli utenti occasionali ed inesperti che trovano negli help in linea un valido aiuto durante il raggiungimento del loro goal.

OS-OR8

Titolo: "Localizzabilità degli oggetti reattivi sul lay-out delle finestre".
Oggetto di Interesse: oggetto reattivo.
Intento: verificare la facilità di individuazione degli oggetti reattivi.
Criteri di usabilità: visibilità (localizzabilità e visibilità delle azioni).
Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; verificare la facilità di individuarlo senza utilizzare gli eventuali feed-back durante il puntamento dello stesso.

Commento: molte volte nelle applicazioni, lo sfondo di una finestra rende l'individuazione degli oggetti contenuti in essa alquanto difficile. Questa task vuole essere una risposta a questo tipo di problemi.

OS-OR9

Titolo: "Coerenza strutturale degli oggetti reattivi".
Oggetto di Interesse: oggetto reattivo.
Intento: verificare la coerenza nella struttura degli oggetti reattivi.
Criteri di usabilità: coerenza (coerenza strutturale).
Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; verificare la coerenza strutturale; cioè controllare che oggetti secondari uguali presentino forme e colori uguali.

Commento: un esempio di incoerenza strutturale è quando, ad esempio, in un'applicazione un gruppo di bottone associati a funzionalità simili presenta caratteristiche di struttura (forma, colore) diverse.

OS-OR10

Titolo: "Coerenza nel comportamento degli oggetti reattivi".
Oggetto di Interesse: oggetto reattivo.
Intento: verificare se ogni oggetto reattivo che viene riusato in più finestre dell'applicazione si comporti sempre nello stesso modo.
Criteri di usabilità: coerenza (coerenza di comportamento).
Descrizione delle attività: per ogni oggetto reattivo presente nell'applicazione; verificare la coerenza di comportamento.

Commento: molto spesso accade che gli oggetti con cui l'utente può interagire siano riusati in più finestre. Se ciò accade bisogna che il progettista dell'applicazione associ all'oggetto un comportamento sempre identico, questo per non violare un criterio importantissimo ai fine dell'usabilità che è la coerenza.

Task Astratti sulle Icone

OS-OR-ICO1

Titolo: "Coerenza nello schema di progetto delle icone".
Oggetto di Interesse: icona.
Intento: verificare la coerenza nello schema di progetto delle icone, cioè verificare che tutte le icone nell'applicazione siano progettate in base a un singolo schema di progetto.
Criteri di usabilità: coerenza (coerenza strutturale), visibilità (riconoscibilità).
Descrizione delle attività: per ogni icona presente nell'applicazione; verifichiamo la presenza di un unico "modo" con cui essa rappresenta e descrive un oggetto o un'azione.

Commento: ci sono differenti schemi di progetto che possono essere usati da base per realizzare le icone per azioni e oggetti [May92]. Ne possiamo indicare alcuni: progettare l'icona rappresentando il prima e il dopo l'esecuzione del comando, oppure rappresentando lo strumento del mondo reale che ci permette di compiere l'azione, o ancora rappresentando l'azione stessa che il comando esegue. Ciò che viene valutata, in questo task, è la coerenza nello scegliere uno schema di progetto unico per tutte le icone nell'applicazione. Questo porta dei vantaggi a livello di usabilità, in quanto l'utente si abitua più facilmente ad apprendere l'unico schema di associazione tra icona e suo referente, in particolare la visibilità viene ad essere soddisfatta in quanto in una qualunque interazione si può applicare lo stesso criterio di riconoscimento di un'interazione passata.

OS-OR-ICO2

Titolo: “Facilità di riconoscimento delle icone”.
Oggetto di Interesse: icona.
Intento: valutare l'efficacia, definita in termini di facilità, per l'utente, di riconoscere ciò a cui un'icona si riferisce.
Criteri di usabilità: visibilità (riconoscibilità).
Descrizione delle attività: per ogni icona presente nell'applicazione; valutare la facilità nel riconoscere l'oggetto o l'azione ad essa associata.

Commento: il riconoscimento, per l'utente, di ciò a cui un'icona si riferisce può essere difficile, tanto più quando le icone rappresentano azioni invece di oggetti. Studi sulla facilità di riconoscimento delle icone, rispetto alla scelta di un particolare schema di progetto [Rog89], hanno evidenziato che le migliori performance si ottengono scegliendo uno schema di progetto che renda il mapping tra icona e oggetto o azione che descrive o cui si riferisce, il più diretto possibile; in altre parole che abbia la più piccola “articulatory distance” [Hut86].
Nolan nei suoi studi [Nol89], suggerisce che gli utenti preferiscono e riconoscono con più facilità, le icone concrete e famigliari rispetto a quelle astratte e non familiari. Si può aumentare, in tutti quei casi in cui ci sia necessità, la facilità di riconoscimento, associando alle icone delle etichette. Questa servirà all'utente per riconoscere più facilmente che cosa un'icona significa. [May92]. Ovviamente più facile è il riconoscimento delle icone per l'utente, più l'applicazione sarà apprendibile.

Task Astratti sui Menu

OS-OR-MEN1

Titolo: “Descrizione on-line di un item durante il suo puntamento”.
Oggetto di Interesse: menu.

Intento: valutare l'efficacia definita in termini di facilità per l'utente di riconoscere ciò cui una voce di menù si riferisce.
Criteri di usabilità: visibilità (riconoscibilità).
Descrizione delle attività: per tutti gli item di ogni menu presente nell'applicazione; verificare la presenza di descrizioni on-line, o più brevemente “micro-help”, per la scelta puntata.

Commento: durante la navigazione dell'utente in un menu, alla ricerca di un item che gli permetta di raggiungere il suo obiettivo: portare a termine un task, egli deve contare, per comprendere il significato della scelta, solo ed esclusivamente sull'etichetta o Titolo dell'item. Da qui la necessità che queste piccole parole che descrivono l'item siano effettivamente significative e non generiche. Delle volte quest'obiettivo è difficile da realizzare. L'aggiunta di altra informazione a supporto della sola etichetta dell'item non può che giovare all'applicazione in termini di usabilità. Quest'informazione aggiuntiva a disposizione mentre si naviga e in particolare mentre si puntano le diverse opzioni in un menu, va sotto il nome di “micro-help” [May92]. Essa, in genere appare o su un'apposita barra posta in fondo allo schermo, oppure in una qualunque area dello schermo dedicata a quest'uso. Vediamo più in dettaglio i criteri di usabilità che in particolare sono soddisfatti con questo tipo di approccio. La presenza di questi descrittori fa aumentare la ricchezza di contenuti dell'applicazione per quanto riguarda l'aspetto di presentazione. Non solo, anche la visibilità è soddisfatta poiché il microhelp incrementa l'abilità dell'utente nel comprendere il significato e lo scopo di ciò che gli viene presentato per la prima volta. Si favorisce anche l'apprendibilità, in quanto chi utilizza l'applicazione è facilitato nell'apprendere il comportamento del sistema. L'utilizzo di questi “aiuti” avvantaggia, in modo particolare, gli utenti casuali che non utilizzano l'applicazione di frequente, d'altro canto non intralcia gli utenti esperti.

OS-OR-MEN2

Titolo: “Presentazione degli item inattivi in un menu “
Oggetto di Interesse: menu.
Intento: valutare l'efficacia definita in termini di facilità per l'utente di riconoscere quali sono le voci di menù selezionabili relativamente allo stato attuale del sistema.
Criteri di usabilità: osservabilità dello stato.
Descrizione delle attività: per ogni menu presente nell'applicazione; verificare che tutti gli l'item inattivi, abbiano caratteristiche di presentazione diverse rispetto a quelli attivi.

Commento: questo task si occupa dell'aspetto di presentazione degli item temporaneamente inattivi in un menu. Un'applicazione può essere vista come un sistema, e come tale in essa si hanno dei passaggi da uno stato ad un altro. Può succedere che in un certo stato, o contesto, alcuni item in un menu non siano selezionabili dall'utente, poiché porterebbero il sistema in una condizione di errore. Allora la domanda che ci si pone è: come si devono trattare questi item dal punto di vista della presentazione. Abbiamo due possibilità: eliminare temporaneamente la scelta non selezionabile dal menu o disattivarla, per esempio “ingrigendola”, e

non permettere che venga selezionata. Delle due in termini di usabilità è sicuramente migliore la seconda. Vediamo perché. La cancellazione modifica arbitrariamente la posizione degli item all'interno della struttura del menu, questo fa si che l'utente non possa più prevedere la posizione di un item attivo. Ciò si ripercuote negativamente anche sulla facilità di apprendere e di navigare efficacemente in un menu. In altre parole il mantenimento della struttura permette all'utente di crearsi un modello di navigazione che lo aiuta nella ricerca dell'item. La disattivazione, in definitiva, facilita l'apprendimento poiché l'utente si abitua a vedere sempre lo stesso ordine nelle scelte, anche quando alcune non sono attive.

OS-OR-MEN3

Titolo: “Adattabilità della struttura dei menu”.
Oggetto di Interesse: menu.
Intento: verificare esistono delle funzionalità efficienti per adattare la struttura dei menu alle caratteristiche dell’utente.
Criteri di usabilità: flessibilità (personalizzazione).
Descrizione delle attività: verificare se l’applicazione definisce dei meccanismi per modificare la struttura dei menu, per renderla più adeguata alle caratteristiche dell’utente che lo usa.

Commento: molto spesso non tutte le funzionalità di un’applicazione sono sfruttate uniformemente, questo per vari motivi; primo l’utente può essere inesperto e non conoscere determinate funzionalità associate agli item dei menu; secondo l’utente si può trovare nelle condizioni di non avere necessità di utilizzare tutti gli item dei menu, durante il raggiungimento dei suoi goal. Oppure, ancora, una funzionalità particolarmente importante non è associata a nessun item dei menu presenti. Per risolvere questo tipo di situazioni l’applicazione dovrebbe permettere di adattare la struttura dei menu, cioè dare la possibilità di modificare gli item dei menu, a quello che sono le particolari esigenze dell’utente.

Task Astratti sulla Casella di Testo

OS-OR-CT1

Titolo: “Casella di testo versus casella lista o casella combinata”.
Oggetto di Interesse: casella di testo.
Intento: verificare il corretto utilizzo delle caselle di testo.
Criteri di usabilità: disponibilità di controllo da parte dell’utente.
Descrizione delle attività: per ogni finestra nell’applicazione:

1. Individuare tutte le caselle di testo;
2. Valutare la possibilità di sostituirle con caselle lista;

Commento: l’uso di una casella lista al posto di una semplice casella di testo riduce gli errori dell’utente dovuti al fatto che egli deve ricordare tutte le possibili scelte ammissibili per quell’oggetto. Per esempio se c’è la necessità in una form di avere l’acquisizione di un colore e i colori accettati sono in numero limitato si può utilizzate quest’oggetto in cui gli item della lista sono proprio i colori. Questo favorisce tutti gli utenti rendendo più efficiente l’interazione con la casella, in particolare quelli inesperti e occasionali che incontrano maggiori difficoltà a ricordare i valori ammissibili. La condizione che deve essere verificata per poter utilizzare una casella lista è che i possibili input siano in numero finito. L’efficienza non consiste solo in un miglior utilizzo dell’oggetto dovuto alla riduzione di errori di inserimento ma anche per l’aumento della velocità con cui l’input viene inserito nella casella, poiché si evita l’utilizzo della tastiera lasciando esclusivamente il mouse come dispositivo di input. Basterà semplicemente un click sulla casella in particolare sul bottone che la fa passare dallo stato “chiuso” a quello “aperto”, e successivamente, sempre con il mouse, selezionare la scelta proprio come in un menu a cascata.

OS-OR-CT2

Titolo: “Casella di testo versus casella spin”.
Oggetto di Interesse: casella di testo.
Intento: verificare il corretto utilizzo delle caselle di testo.
Criteri di usabilità: disponibilità di controllo da parte dell’utente.
Descrizione delle attività: per ogni finestra nell’applicazione:

1. Individuare tutte le caselle di testo;
2. Valutare la possibilità di sostituirle con caselle spin;

Commento: l’uso di una casella spin al posto di una casella di testo tradizionale aumenta l’efficienza del controllo durante l’inserimento dell’input in una finestra. Ciò è dovuto al fatto che l’uso esclusivo del mouse e non di tastiera e mouse riduce i gesti da compiere per l’inserimento dell’input, aumentando in questo modo la rapidità di esecuzione del compito.

Task Astratti sul Puntatore

OS-OA-PNT1

Titolo: “Appropriatezza della forma del puntatore”.
Oggetto di Interesse: puntatore.
Intento: verificare se la forma del puntatore è appropriata rispetto all'oggetto puntato.
Criteri di usabilità: visibilità (localizzabilità e visibilità delle azioni), efficienza.
Descrizione delle attività:

1. Verificare l'esistenza di cambiamenti di forma del puntatore, durante l’interazione con l'applicazione.

2. Verificare l'efficienza delle forme in funzione dell'oggetto puntato; cioè verificare che la forma del puntatore si modifichi in modo opportuno rispetto all'oggetto puntato, per riflettere, per esempio, l'assenza o presenza di interazione con l'oggetto.

Commento: Nella maggior parte delle applicazioni la forma del puntatore si modifica, durante il dialogo, per dare, in questo modo, un feed-back all'utente su che cosa il sistema sta facendo, o su che cosa l'utente si aspetta che possa essere fatto in un certo momento. La forma del puntatore rappresenta un'informazione per l'utente rispetto alle azioni a sua disposizione sull'oggetto puntato. Ora, poiché in una applicazione l'utente sposta il puntatore su oggetti diversi in termini di interazione, sarebbe auspicabile che la forma del puntatore cambiasse per riflettere l'assenza o la presenza di interazione, non solo ma anche il tipo. Finita la verifica dei cambiamenti di forma, bisogna passare a verificare l'efficienza, rispetto all'oggetto puntato, di questa forma. Per esempio una "mano puntante" andrà bene quando il puntatore si trova su un bottone, su uno slider andrà meglio, invece, una "mano prensile". Al bottone è associata infatti una tipo di interazione definita di "pressione" mentre il tipo di interazione permessa con una slider è lo "scorrimento"

OS-OA-PNT2

Titolo: "Coerenza della forma del puntatore".
Oggetto di Interesse: puntatore.
Intento: verificare che nell'applicazione esista un comportamento uniforme dei "ritorni visivi" del puntatore rispetto agli oggetti puntati.
Criteri di usabilità: coerenza (coerenza strutturale).
Descrizione delle attività:
1. Individuare tutti i cambiamenti di forma del puntatore;
2. Verificare la coerenza di questi cambiamenti; cioè controllare che tutti gli oggetti appartenenti ad una determinata categoria presentino un uguale feed-back del puntatore quando sono puntati.

Commento: molto spesso accade che, in un'applicazione, una determinata classe di oggetti non goda di un uguale feed-back del puntatore quando questi oggetti sono puntati. Scoprire incoerenze di questo tipo è il compito di questo task.

Glossario della Visual Sentence Theory

Shortened	Entity	defined as	in
τ	Transparent Symbol	an element of the alphabet which is inferior to any lowest element in any possible ordering on the alphabet.	[1]
i	(digital) image (on an alphabet V)	a bi-dimensional string specified by the function i: {1, 2, ..., r_{max}} X {1,2, ..., c_{max} } → V ∪{τ}, where r_{max} and c_{max} are two integers indicating the size of the image and τ is the transparent symbol	[2]
	pixel	a triple: (r, c, v), where r and c are two integers which specify the position of the pixel and v is an element of the alphabet V ∪ {τ} which specifies the value. If v ∈V, the pixel is said visible	[3]
	pel (of an image i)	a restriction of i to one of its elements	[4]
cs	characteristic structure	set of image pixels which form functional or perceptual units for the user	[3]
u	attributed symbol (on alphabet T)	an element of T X D_{a1} X ... X D_{am} where D_{a1}, ... , D_{am} are sets of values	[2]
u	Computational construct	specifies the state of a program being executed, i.e. it contains the executable code of the program and the current values of its variables	[11]
d	description of an image	a string without repetitions on an alphabet of attributed symbols.	[2]
cp	characteristic pattern	<*cs*, u, <intcs, matcs>>, i.e. it is the triple formed by a *cs*, an attributed symbol, and the relations among them	[5]
		cp specifies the state of a computational process that generates and maintain active a virtual entity. It is specified as <*cs*, u, <intcs, matcs>>.	[11]
int	interpretation function	associates css in an image with attributed symbols in a description	[5]
intcs	interpretation function	maps a *cs* into an attributed symbol u	[10]
mat	materialization function	associates attributed symbols in a description with css in an image	[5]
matcs	materialization function	maps an attributed symbol u into a *cs*	[10]
CPT	set of types ti of cps	CPT={t1,…tn} of *cp* types	[11]
cp_{ti}	represenatative *cp* of type ti	an instance of ti from which any other instance of a *cp* of the same type can be derived	[11]
f(δ, cp_{ti})	equivalence function	a function f such that *cp*=f(δ, cp_{ti}), where δ is a set of suitable parameters e cp_{ti} is the representatve *cp* of type ti	[11]
CP		the set of all representative cp_{ti}s corresponding to cp types in CPT.	[11]
vs	visual sentence	a triple <i, d, <int, mat>> where i is an image, d is a description, int is an interpretation function and mat is a materialization function	[5]
		a triple <i, u, <int, mat>> which describes the current state of the system	[10]
VL	visual language	a set of vss	[5]
	(user) gesture	elementary user actions on a specific tool	[7]
	(user) action	sequences of gestures achieving a sub-goal in the task	[7]
a	user activity	pair a=<operation, cp>, where operation is the sequence of events perceived by the machine and generated by the user acting on an input device, and cp is the active cp	[11]
OP	set of operations		[11]
A	set of user activities	A={a_i = <op_k, cp_j> | op_k∈O and cp_j∈CP}	[11]
tr	transformation of visual sentences	tr:<a,<vs1⇒vs2>> a visual sentence vs_1 is transformed into a visual sentence vs_2 as the consequence of some human activity a.	[12]

r	rewriting rule	r=<ant, cond, cons>, where ant (antecedent) is a set of representatives cp_{ti}, cond is a condition on ant, and cons (consequent) is a second set of cp_{ti}	[11]
tr	transformation rule	***tr***:<a_i, r>, where r=<ant, cond, cons> ∈ P, a_i ∈A is the user activity <op, cp> with op∈O and cp∈ant	[12]
P		Set of rewriting rule	[12]
TR	set of transformation rules		[12]
	support (of a *cs*)	the set of coordinates of the pixels belonging to the *cs*	[6]
	frame	the set of **cs**s which remain constant or maintain a constant support in all the **vs**s generated during the interaction.	[6]
		the set of **cp**s which remain constant (in a transformation) (???)	[12]
	active *vs*	the *vs* the user is currently interacting with	[6]
TVL	Task Visual Language	the whole set of documents produced to accomplish a task.	[7]
ATVL	Augmented TVL	version of TVL appearing on the screen during the interaction in which the symbol associated with a *cs* may also contain the executable specification of some activity relative to the entity the *cs* represents	[8]
IVL	Interaction Visual Language	The set of all the active vss which can be generated using a given Visual Interactive System.	[7]
		The set of (possibly infinite) sequences of vss generated by the user starting from vs0	[11]
		IVL = <vs0, TR>, from which every sequence of admissible transformations can be computed	[12]
	computational activity	the system reaction due to a user action	[7]
	legal *vs*	*vs* which a) belongs to the intended VL or b) can be transformed into a legal *vs* without destroying the effect of any previous action.	[5]
	legal action	an action which users can perform in order to generate a legal *vs*	[5]
	scaffold	set of css which are added to the css of ATVL and which denote the activities which can be performed to manage the active ***vs***,	[9]
sel	selection function on image i	sel: {1, 2, ..., r_{max}} X {1,2, ..., c_{max} } →{0, 1}	[1]
s	image structure of image i	a copy of i in which selected pixels maintain the original values and the others are mapped to τ	[1]
ker(s)	kernel of a structure s	the set of pixels (r, c, v) forwhich sel_s(r, c) = 1	[1]
shd(s)	shadow of a structure s	the set of positions of visible pixels in s.	[1]
vis (s)	visibility set of a structure s	the set of visible pixels in s	[1]

Bibliografia relativa al Glossario IVL

[1] P. Bottoni, M.F. Costabile, S. Levialdi, P. Mussio, “On Order in Visual Interaction”, Journal of Visual Languages and Computing (to appear).

[2] P. Bottoni, M.F. Costabile, S. Levialdi, P. Mussio, “Specification of Visual Languages as Means for Interaction”, in Visual Language Theory, K. Marriott, B. Meyer (eds.), Springer-Verlag, New York, 1998, pp. 353-375.

[3] P. Bottoni, M.F. Costabile, S. Levialdi, P. Mussio, “Formalising Visual Languages”, Proc. IEEE Symposium of Visual Languages ’95, pp. 334-341.

[4] S.K. Chang, P. Mussio, “Customized Visual Language Design”, Proc. 8° Intl. Conf. on Software Eng. and Knowledge Eng. - SEKE'96, 1996, 553-562.

[5] P. Bottoni, M.F. Costabile, P. Mussio, “Specification and Dialog Control of Visual Interaction through Visual Rewriting Systems”, ACM Transaction on Programming Languages and Systems, 21(6), 1077-1136, 1999.

[6] P. Bottoni, S.K. Chang, M.F. Costabile, S. Levialdi, P. Mussio, “On the Specification of Dynamic Visual Languages”, Proc. IEEE Symposium of Visual Languages ’98, pp 14-21.

[7] A. Bianchi, P. Bottoni, P. Mussio, “Issues in Design and Implementation of Multimedia Software Systems”, Proc. IEEE Intl. Conf. On Multimedia Computing and Systems 1999, (to appear).

[8] N. Bianchi, P.Bottoni, P. mussio, G. Rezzonico, M.G. Strepparava, “Particiaptory interface design:from naive models to systems”, in Design of computing system, vol 1: Cognitive considerations, Salvendy, Smith, Koubek (eds), Elsevier-Amsterdam, pp. 573-576.

[9] P. Bottoni, S.K. Chang, M.F. Costabile, S. Levialdi, P. Mussio, Dimensions of Visual Interaction Design, Proc. IEEE Symposium of Visual Languages ’99, (to appear).

[10] S. Arondi, P. Baroni, D. Fogli, P. Mussio, Supporting co-evolution of users and systems by the recognition of Interaciton Patterns. Proceedings of the International Conference on Advanced Visual Interfaces (AVI 2002), Trento (I), May 2002, 177-189.

[11] P. Carrara, D. Fogli, G. Fresta, P. Mussio, Co-operative design of Visual Interactive Systems, submitted to the Workshop on Visual Computing, 2002.

[12] P. Carrara, D. Fogli, G. Fresta, P. Mussio, Toward overcoming culture, skill and situation hurdles in human-computer interaction. International Journal on Universal Access in the Information Society, DOI 10.1007/s10209-002-0028-4, 2002.

Bibliografia

[App92]	APPLE COMPUTER, “Human Interface Guidelines”. Addison-Wesley, 1992.
[BaR02]	Costabile, M.F., Pittarello, F., Ardito, C.. Usabilità di Sistemi a Realtà Virtuale. Rapporto DEVISU, BA-R02. 2001.
[BaR03]	Costabile, M.F.. Progettazione di Sistemi Visuali Usabili. Rapporto DEVISU, BA-R03. 2001.
[BaR04]	Costabile, M.F., Ardito, C., Lanzilotti, R., Minardi, G., Pittarello, F.. Ispezione di Usabilità e Valutazione della Componente Sonora all’interno del Sito Web: “Espressionismo Tedesco a Palazzo Grassi”. Rapporto DEVISU, BA-R04. 2002.
[Bia91]	A. Bianchi, M. D'Enza, M. Matera, A. Betta, "Designing Usable Visual Languages: the Case of Immune System Studies". Proc. IEEE Symposium Visual Languages '99, pp.254-261, 1999.
[Bie95]	Bieber, M. and Isakowitz, T.. “Special Issue on Designing Hypermedia Applications”. Communication of ACM, 38 (8). 1995.
[Bot99]	P. Bottoni, M.F. Costabile, P. Mussio, "Specification and Dialogue Control of Visual Interaction through Visual Rewriting Systems". ACM TOPLAS, Vol. 21, N. 6, pp. 1077-1136, 1999..
[Cha96]	S.K. Chang, P. Mussio, "Customized Visual Language Design". Proc. of Eighth Int'lConference on Software Engineering and Knowledge Engineering, SEKE’96, pp. 553-562, 1996.
[Dix93]	Dix, A., Finlay, J., Abowd, G., and Beale, R.. “Human-Computer Interaction”. Prentice Hall, 1993.
[Dix98]	A. Dix, J. Finlay, G. Abowd, R. Beale, Human Computer Interaction, Prentice Hall, London, 1998.
[Gal94]	Galitz, W. O.. “It’s Time to Clean Your Windows: Designing GUIs That Work”. New York, John Wiley & Sons, 1994.
[Gar93]	Garzotto, F., Paolini, P., Schwabe, D.. “HDM - A Model Based Approch to Hypermedia Application Design”. In ACM Trans. Inf. Syst., 11 (1), January 1993.
[Gar94]	Garzotto, F., Paolini, P. and Schwabe, D.. “HDM – A Model Based Approach to Hypermedia Application Design”. ACM Trans. Inf. Syst., 11 (1). 1993.
[Gar99]	Garzotto, F., Matera, M., Paolini, P.. “Abstract Task: a Tool for the Inspection of Web Sites and Off-line Hypermedia”. In Proc. ACM HT, pp. 157-163. 1999.
[Gre97]	Green, T.R.G, Petre, M.. Usability Analysis of Visual Programming Environments. Journal of Visual Language and Computing 7(2), pp. 131-174, 1996.
[Gru89]	Grudin, J.. “The Case against User Interface Consistency”. Communications of ACM Vol.32, n. 10, pp. 1117-1164, 1989.

[Har93]	Hardman, L., Bulterman, D. C. A., Van Rossum, G.. “The Amsterdam Hypermedia Model: extending hypertext to support real multimedia”. Hypermedia Journal 5 (1), July 1993, pp. 47-69.
[Hut86]	Hutchins, E. L., Hollan, J. D., Norman, D. A.. “Direct Manipulation Interfaces”. User Centered System Design , NORMAN D. A. e DRAPER S. W., eds, Hillsdale, N. J.: Lawrence Erlbaum Associated, 1986, pp. 87-124.
[Ibm93]	IBM. “Common User Access Guidelines”. Que Publishing, 1993.
[Lat86]	B. Latour, "Visualization and Cognition: Thinking with Eyes and Hands”. In: Knowledge and Society: Studies in the Sociology of Culture Past and Present, Vol. 6, pp.1-40, 1986.
[Mat99]	Matera, M.. “Abstract Task For The Usability Inspection Of Hypermedia Applications”. Tesi di Dottorato di ricerca in Informatica. 1999.
[Mat02]	Matera, M., Costabile, M. F., Garzotto, F., Paolini, P.. “SUE Inspection: An Effective Method for Systematic Usability Evaluation of Hypermedia”. IEEE Transactions on Systems, Man and Cybvernetics- Part A, vol. 32, no. 1, Jenuary 2002.
[May92]	Mayhew, D. J.. “Principles and Guidelines in Software User Interface Design”. Prentice Hall, Englewood Cliffs, 1992.
[Mic92]	MICROSOFT CORPORATION. “The Windows Interface Guidelines for Software Design”. Microsoft Press, Redmond, 1992.
[Nie93]	Nielsen, J.. “Usability Engineering”. Academic Press, New York. 1993.
[Nol89]	Nolan, P. R.. “Designing Screen Icons: Ranking and Matching Studles”. Prooceedings of the Human Factors Society 33rd Annual Meeting, 1989, pp. 380-384.
[Nor88]	Normann, D.. The Psychology of Everyday Things, 1988.
[Ope93]	OPEN SOFTWARE FOUNDATION. “OSF Motif Style Guide”. Release 1.2, Englewood Cliffs, Nj, Prentice Hall, 1993.
[Paa88]	Paap, K. R., Roske-Hofstrand, R. J.. “The Optimal Number of Menu Options per Panel”, Human Factors, 28, no. 4 (August 1986), pp. 377-386 and “Design of Menus”, Handbook of Human Computer Interaction, Martin Helander, ed., (Amsterdam: North-Holland, 1988), pp. 205-235.
[Pit97]	Pittarello, F.. Palazzo Grassi web site. (http://www.palazzograssi.it). 1997.
[Pit00]	Pittarello, F.. “Desktop three-d interfaces for Internet users: efficiency and usability issues”. Tesi di Dottorato di ricerca in Informatica. 2000.
[Pra00]	R. Prates, C. De Souza, and S. Barboza, “A Method for Evaluating the Communicability of User Interfaces”. Interactions, Vol.7, No.1, pp.31-38, 2000.
[Pre94]	Prece, J.. Human Computer Interaction, Addison Wesley, 1994.
[Rog89]	Rogers, Y.. “Icons at the Interface: Their Usefulness”, Interacting with Computers: The Interdisciplinary Journal of Human-Computer Interaction, 1, n. 1, April 1989, pp. 105-117.

[Rum95]	Rumbaugh, J., Blama, M.. “Object-Oriented Modeling and Design”. Prentice Hall International, 1995.
[Shn92]	Shneiderman, B.. “Designing the User Interface: Strategies for Effective Human Computer Interaction”, Reading Ma: Addison-Wesley, 1992.
[Sun90]	SUN MICROSYSTEMS. “Open Look Graphical User Interface Application Style Guidelines”. Addison-Wesley, 1990.
[Tei83]	Teitelbaum, R. C., Granda, R. E.. “The Effects of Positional Constancy on Searching Menus for Instructions”, CHI Proceedings, ACM, pp. 150-153, 1993.
[Thi90]	Thimbleby, H.. User Interface design. ACM Press, 1990.