Alessandro Soro (ed.), *Human Computer Interaction*, 11-15


# Interfacce Utenti Multi-Dispositivi

**Fabio Paternò** – Laboratorio Interfacce Utenti dell'Istituto di Scienza e Tecnologie dell'Informazione del Consiglio Nazionale delle Ricerche
fabio.paterno@isti.cnr.it

**Abstract**. Questo articolo mira a fornire una discussione su come ottenere interfacce utenti che si adattano a diversi dispositivi preservandone l'usabilità. A questo scopo, esso analizza e discute lo stato dell'arte in termini di approcci, criteri di progettazione e strumenti automatici; e mostra esempi di applicazione di tali concetti. L'obiettivo è di consentire di comprendere lo spazio delle possibili soluzioni nell'ambito dell'adattamento al dispositivo sia in fase di progettazione che di esecuzione al fine di applicarle meglio e stimolare a pensare a nuove soluzioni.

**Keywords**: HCI, Ambienti Multi-Dispositivi, Adattamento, Ubiquitous Computing.

## Introduzione

Le motivazioni per affrontare le problematiche relative alle interfacce utenti multi-dispositivi sono sotto gli occhi di tutti. La nostra realtà quotidiana è caratterizzata dalla presenza di un numero sempre crescente di dispositivi informatici interattivi. Nel mercato di massa vengono proposti continuamente nuovi dispositivi con capacità di calcolo e risorse di interazione sempre maggiore. Il risultato è una offerta tecnologia che spazia da orologi con schermi interattivi a display delle dimensioni di una parete. Quindi diventa fondamentale proporre delle soluzioni che consentano agli utenti di sfruttare questa abbondanza tecnologica. Questo implica la capacità delle interface utenti di adattarsi al variare delle risorse di interazione.

Il punto di vista di chi scrive è quello di un informatico anomalo, ovvero una persona che ha una formazione informatica ma che è

stato uno dei primi in Italia, ormai una ventina di anni fa, a capire l'importanza della disciplina chiamata human-computer interaction (HCI) (Paternò, 2004). Questo implica non vedere più l'informatica in modo tradizionale, ovvero come una disciplina orientata essenzialmente a trovare soluzioni per fare i calcoli più velocemente, ma vederla come una disciplina che mira a fornire ad un numero sempre più ampio di utenti la possibilità di interagire e comunicare informazioni in modo usabile ed accessibile. Il successo del Web è un chiaro esempio di questa visione, che ora può sembrare anche un po' banale ma che per molti anni non è stata facilmente recepita dal mondo accademico italiano.

Quindi questo capitolo cercherà di fornire una discussione delle problematiche relative ad ambienti caratterizzati dalla presenza di vari tipi di dispositivi, in particolare sistemi desktop e mobile. Questa discussione è basata sulle esperienze maturate in vari progetti, in particolare progetti Europei che hanno coinvolto vari gruppi sia in ambito di ricerca che di aziende. Gruppi che spesso hanno avuto una caratterizzazione multi-disciplinare coinvolgendo non solo informatici ma anche esperti di psicologia cognitiva, designers, esperti dei domini applicative considerati, ecc. Il mio lavoro si è svolto nel Laboratorio Interfacce Utenti dell'Istituto di Scienza e Tecnologie dell'Informazione del Consiglio Nazionale delle Ricerche, che si focalizza nello sviluppo di soluzioni software e tecnologiche per interfacce utenti che siano accessibili nei più disparati contesti di uso.

# 1. Concetti di Base

Per poter meglio seguire la discussione è utile richiamare alcuni concetti di base. Il primo è quello relativo all'adattamento e la prima distinzione da fare tra le tecniche dell'adattamento è quella tra adattabilità ed adattività. Una soluzione è adattabile quando ha la capacità di modificare aspetti su richiesta esplicita dell'utente in accordo a opzioni predefinite. Un tipico esempio è un'applicazione dove vi sono alcuni profili di accesso predefiniti (esperto, intermedio, iniziale), all'inizio della sessione l'utente ne sceglie uno e poi navigherà e riceverà informazione in modo conseguente alla scelta

iniziale. Invece, una soluzione è adattiva quando ha la capacità di modificare aspetti dinamicamente senza richiesta esplicita dell'utente. Quindi questo implica che nel sistema vi sono delle regole che a seconda di quello che viene rilevato determinano come modificare le modalità di interazione.

Gli aspetti che si possono adattare nelle interfacce utenti sono di tre tipologie:

- le presentazioni (nella scelta di modalità, layout, attributi grafici, …)
- il comportamento dinamico (nella scelta del modello di navigazione, l'abilitazione e disabilitazione dinamica delle tecniche di interazione, …)
- il contenuto dell'informazione che viene presentata.

Noi quindi vedremo come l'adattamento si applica per gestire ambienti che sono disponibili a causa delle tendenze principali tecnologiche caratterizzate dalla presenza di vari tipi di dispositivi interattivi (desktop, cellulari, PDAs, TV digitale, dispositivi vocali, …). Questo ha anche portato ad una evoluzione nei linguaggi usati per specificare e programmare il comportamento interattivo, con una attenzione sempre maggiore a linguaggi capaci di descrivere i principali aspetti da considerare astraendo dalla miriade di dettagli implementativi associati ai vari possibili dispositivi e relativi linguaggi di programmazione.

Quello che caratterizza gli ambienti multi-dispositivi è la variabilità del contesto di uso. Come indicato nella Figura 1 il contesto di uso è caratterizzato da tre dimensioni principali:

**utente**, che è caratterizzato dalle sue preferenze, background, obiettivi, ecc.

**dispositivo**, che è caratterizzato dalle sue risorse di interazione (ampiezza schermo, supporto vocale, modalità di interazione, ecc.)

**ambiente**, che a sua volta può essere distinto in quello fisico (luce, rumore, temperatura, posizione, …) e sociale, che indica chi sono le persone che sono vicino e soprattutto che tipo di relazione abbiamo con esse perché questo può avere un impatto

sulla scelta dell'informazione che vogliamo condividere con loro.

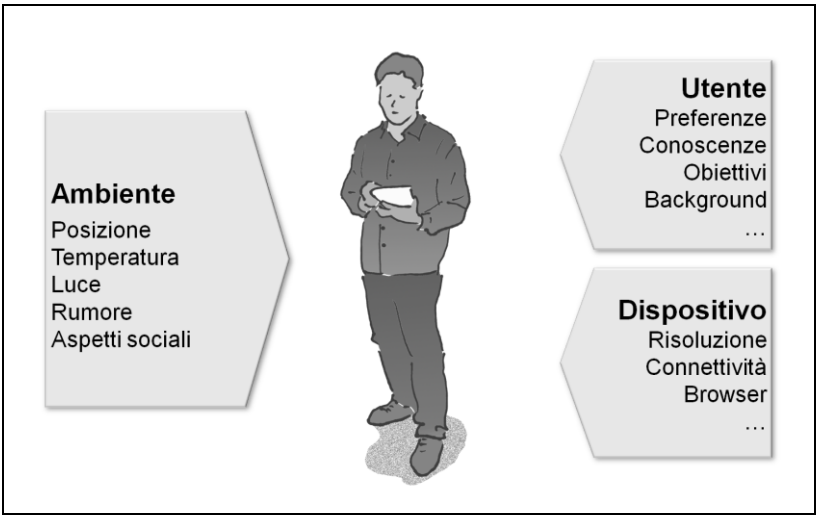


Figura 1. Il Contesto di Uso

## 2. Gli Ambienti Multi-Dispositivi

Negli ultimi anni vari approcci sono stati adottati e proposti per la progettazione e sviluppo di interfacce utenti in ambienti multi-dispositivi. Una prima classificazione di questi approcci è:

**Soluzioni Manuali**, che implicano essenzialmente che una versione per ciascuna tipologia di dispostivi che si vuole supportare viene sviluppata. È chiaramente una soluzione costosa in termini di risorse umane e temporali richieste, in particolare con il crescere delle tipologie di dispositivi che si vuole supportare.

**Transcoders**, in questo caso viene effettuate una traduzione automatica da un linguaggio per un tipo di dispositivo ad un altro (uno dei primi esempi erano i transcoders da HTML a WML). I criteri che solitamente adottano sono sintattici e quindi le soluzioni che ne derivano hanno spesso una limitata usabilità, in

quanto cercano di forzare una progettazione per un tipo di dispositivo ad un altro che ha caratteristiche diverse.

**Style sheets**, essi consentono di cambiare il modo di presentare vari tipi di informazione in base al tipo di dispositivo; rappresentano quindi un utile supporto anche se sono in realtà una soluzione parziale in quanto non consentono di modificare la struttura dell'applicazione interattiva (cosa che talvolta è utile per meglio supportare l'utente tramite un dispositivo di tipologia diversa).

**Information Visualization** (Spence, 2007), questa è un'area che ha studiato varie tecniche utili per rappresentare grandi quantità di dati in modo da poter accedere facilmente alle informazioni associate. Tali tecniche possono essere applicate utilmente quando si accede tramite dispositivi con schermi piccoli. Il problema è che spesso richiedono notevoli risorse di calcolo che piccoli dispostivi possono non avere.

**Approcci basati su modelli**, dove l'idea di fondo è di evidenziare nei modelli le scelte principali di progettazione e poi avere delle trasformazioni che le adattano ai dispositivi correnti. In questo caso un aspetto delicato è trovare un giusto equilibrio tra il livello di astrazione e la possibilità di controllare a pieno i risultati dell'adattamento da parte del progettista.

Un aspetto che caratterizza gli ambienti multi-dispositivi è la variabilità delle dimensioni degli schermi. I Personal Computer (PC) solitamente variano tra 800x600 e 1800x1440 pixel, i PDAs solitamente variano tra 240x240 e 480x640 pixel, i cellulari solitamente variano tra 128x128 e 240x240 pixel. L'IPhone, che è a cavallo tra queste due ultime categorie, ha attualmente una risoluzione di 320x480 pixel. In genere, le dimensioni dello schermo variano più tra dispositivi mobili che tra i sistemi desktop. É bene ricordare che la legge di Moore cambia continuamente questi numeri! Essa infatti dice sostanzialmente che il numero di componenti in un chip raddoppia ogni 18 mesi, quindi le capacità di memoria o di elaborazione dei sistemi informativi evolvono di conseguenza.

Quando si parla di dispositivi mobili in realtà si parla di una categoria molto variegata al suo interno. Infatti, se consideriamo i cel-

lulari possiamo facilmente notare che variano in termini di softkeys (i tasti fisici che forniscono). Un altro aspetto che può variare è la modalità di base di interazione. Alcuni consentono di poter selezionare liberamente qualsiasi punto dello schermo senza restrizioni (come accade con il mouse per i sistemi desktop), mentre altri danno la possibilità di fare solo 5 attività di base: spostare il cursore a destra o sinistra, in alto o in basso, sempre sequenzialmente, senza poter saltare liberamente da un punto all'altro), oppure di selezionare l'elemento corrente nell'interfaccia utente (vedi Figura 2), quindi mancano di un dispositivo di puntamento.


Figura 2. Esempio di cellulare con interazione a 5 vie

L'interazione con i dispositivi cellulari deve tener conto delle loro caratteristiche particolari. Il display è piccolo e la sua ampiezza può variare molto. L'input testuale è lento. Talvolta non c'è dispositivo di puntamento. Le softkeys sono usate per attivare i comandi ma il loro numero e scopo varia a secondo del dispositivo. Spesso l'utente ha da pagare per accedere ai dati. Buoni cellulari supportano anche l'accesso tramite WLAN diminuendo il tempo di scaricamento durante la navigazione, e migliorano continuamente in termini di caratteristiche e possibilità.

Al fine di ottenere soluzioni usabili nell'interazione mobile c'è da tenere presente una serie di fattori. É importante minimizzare

l'input testuale, e sfruttare gli elementi del dispositivo, come gli accesskeys. É utile mantenere un certo livello di coerenza tra le versioni di un'applicazione per piattaforme diverse, in modo che l'uso di una versione per un dispositivo diverso non richieda sforzi eccessivi di apprendimento. É importante prevenire gli errori dell'utente in quanto un accesso sbagliato può richiedere tempo (e quindi costi) per poter tornare ad una situazione corretta. Per questo lo scopo degli elementi dell'interfaccia deve essere chiaro e bisogna evitare di sovraccaricare l'interfaccia con molti elementi. É anche utile limitare il bisogno di scrolling. In generale, c'è da tener presente che l'accesso tramite dispositivo mobile è breve. Gli utenti non vogliono usare dispostivi con limitate risorse per lunghe sessioni interattive in quanto sarebbe alquanto scomodo. Quindi, l'accesso tramite dispositivo mobile è solitamente orientato a piccole quantità di informazioni che servono al momento.

Virpi Roto (Roto, 2006), ricercatrice di Nokia Research, nella sua tesi di dottorato evidenzia bene come l'usabilità nei dispositivi mobili si differenzia da quella nei sistemi desktop perché i sistemi desktop sono molto standardizzati in termini di hardware e software. Viceversa nei dispostivi mobili c'è una notevole variabilità in termini di ampiezza di schermi, tasti disponibili (sia in termini di numero che posizione e forma) e poi come questi tasti possono essere sfruttati dai microbrowser Web che consentono la navigazione nelle applicazioni. Questi microbrowser Web sono diversi dai browser a cui siamo abituati nei sistemi desktop e variano sensibilmente anche tra di loro (ad esempio nel modo in cui associano i tasti del telefono a dei comandi).

In ambito W3C (W3C, 2008) c'è stato recentemente uno sforzo per fornire guidelines per ottenere interfacce utenti usabili per dispositivi mobili. Esse considerano vari punti:

**Comportamento generale**, ovvero sfruttare le capacità del dispositivo per fornire una migliore user experience (quindi fornire una migliore user experience su dispositivi più capaci);

**Navigazione e Links**; Tenere brevi gli URIs delle home dei siti, fornire minimo supporto alla navigazione in cima alla pagina;

**Layout e Contenuto**; Cercare di limitare lo scrolling ad una sola direzione, assicurarsi che il contenuto che è centrale alla pagina preceda quello meno importante;

**Definizione pagina**; Fornire un titolo della pagina breve ma descrittivo, non usare frames, non usare tavole annidate e tavole per il layout;

**User Input**; tenere al minimo il numero di keystrokes, evitare input testuale libero quando possibile, fornire valori pre-selezionati di default quando possibile.

# 3. Comprendere le Interfacce Utenti Multi-Dispositivi

Per poter effettuare una progettazione efficace in ambienti multi-dispositivi è importante aver chiaro lo spazio delle possibili scelte. In particolare. Va considerato lo spazio che mette in relazione i compiti (task in inglese) che gli utenti intendono svolgere e le piattaforme interattive considerate. Infatti una regola principale dell'usabilità e di focalizzarsi sull'utente e le attività che intende svolgere. Qui si intende per piattaforme gruppi di dispositivi che hanno risorse di interazione simili (desktop, PDA, cellulari, dispositivi vocali, …)

I casi possibili sono essenzialmente quattro:

**Stesso task nello stesso modo in piattaforme diverse**, ad esempio un login richiede una interazione simile per qualsiasi piattaforma si consideri;

**Stesso task su piattaforme diverse ma in modo diverso**, ovvero l'attività da svolgere è la stessa ma vi sono tecniche di interazione diverse che le supportano. Tali tecniche hanno quindi la stessa semantica ma richiedono risorse di interazione diverse che quindi sono più adatte per certe piattaforme che per altre;

**Dipendenze tra task eseguiti su piattaforme diverse**, questo implica che il fatto di aver eseguito un'attività attraverso un certo dispositivo abiliti o disabiliti la possibilità di svolgere un'altra attività attraverso un altro dispositivo;

**Task significativi solo in un tipo di piattaforma**, in questo caso ci sono delle attività che sono significative se si utilizza un certo dispositivo ma non con un altro e quindi se si cambia dispositivo non ha più senso supportarle.

Vediamo alcuni esempi concreti per meglio capire questa classificazione di casi. Sicuramente a nessuno viene in mente di organizzare un viaggio aereo tramite un cellulare perché questa attività richiede di accedere ai siti di diverse compagnie aeree, confrontare i risultati, modificare le richieste per vedere se si trovano soluzioni più vantaggiose. Queste sono attività che si fanno bene seduti con uno schermo ampio a disposizione. Viceversa, se si è in auto e si vuole sapere se il volo che si vuole prendere è in orario o in ritardo l'unica possibilità è l'uso di un cellulare. Analogamente, in una applicazione cinematografica, la decisione di quale film vedere può richiedere l'accesso a critiche dei film disponibili e relativi trailer che si fa meglio con un sistema desktop mentre verificare se ci sono ancora posti disponibili al cinema all'ultimo momento mentre si è al bar con gli amici si fa decisamente meglio con il cellulare.

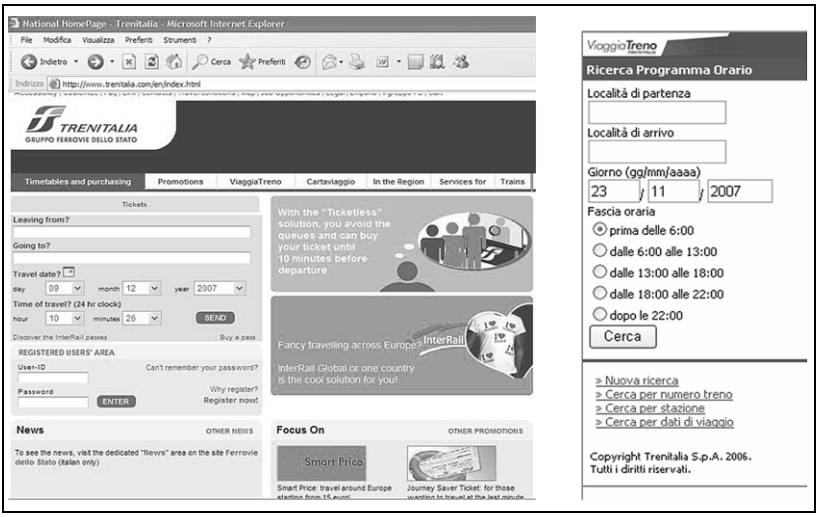


Figura 3. Esempio Stesso Task, Interfacce Diverse.

Se torniamo alla classificazione precedente, vediamo come in Figura 3 si fornisce un esempio del caso stesso task ma supportato in maniera differente. Qui vediamo una versione precedente del sito delle ferrovie italiane. Vediamo, ad esempio, come la selezione del giorno e dell'ora nella versione desktop avviene tramite dei pull-down menu, alcuni dei quali, quando selezionati devono mostrare una lista di possibili valori notevolmente lunga. Siccome sul dispositivo mobile lo spazio a disposizione è scarso queste due attività sono supportate in modo differente: la scelta del giorno viene specificata non selezionando da una lista di possibili valori ma immettendo direttamente i valori di interesse, mentre per le ore sono state predefinite alcune fasce in numero molto limitato da cui è possibile scegliere quella di interesse.


Figura 4. Altro Esempio di Stesso Task, Interfacce Diverse.

Un altro esempio dello stesso caso è presentato in Figura 4. In questo esempio consideriamo due versioni differenti del sito del quotidiano La Repubblica. Anche qui possiamo notare che anche se entrambi consentono l'accesso agli stessi articoli le interfacce utenti si differenziano consistentemente. Nella versione desktop si fa maggior uso di immagini, anche più grandi, l'informazione è strutturata su più colonne, con servizi aggiuntivi (come le news) e maggiori pubblicità. Mentre il disegno della versione mobile è più sobrio e lineare per facilitare l'accesso da dispositivi con capacità più limitate.

La Figura 5 ci mostra un esempio del caso in cui entrambe le piattaforme supportano lo stesso task principale ma con un diverso livello di decomposizione. Ovvero in un caso vi è il supporto di un numero di task secondari che non c'è nell'altro caso. Nell'esempio il task principale è di effettuare una prenotazione alberghiera. Vediamo come nel caso del dispositivo mobile si forniscono solo le informazioni essenziali (nome, data di arrivo e partenza, contatto telefonico) mentre nel caso di accesso desktop si rende possibile specificare una serie di preferenze (come il tipo di stanza) e di dati personali (come il numero del documento di identità).

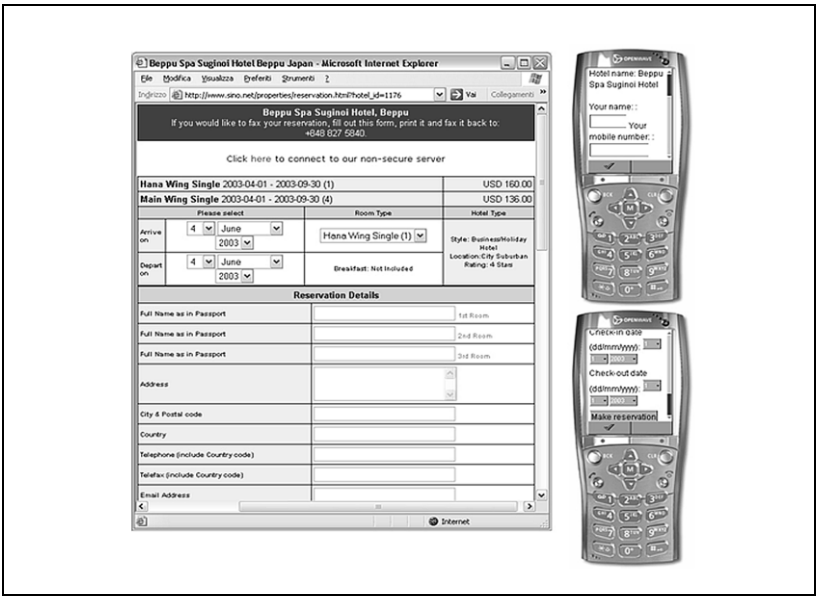


Figura 5. Stesso task principale ma con diversa decomposizione.

L'ultimo esempio di questa sezione riguarda il caso in cui vi sono dipendenze tra task eseguiti su diverse piattaforme. In particolare, l'esempio mostra il caso in cui l'utente tramite l'interfaccia desktop accede ad un servizio di prenotazione di un volo aereo, che quando è andato a buon fine abilita automaticamente la possibilità di accedere tramite il cellulare ad informazioni in tempo reale relative al

volo selezionato. Quindi un'attività svolta tramite il desktop abilità successivamente un'attività tramite il cellulare (Figura 6).

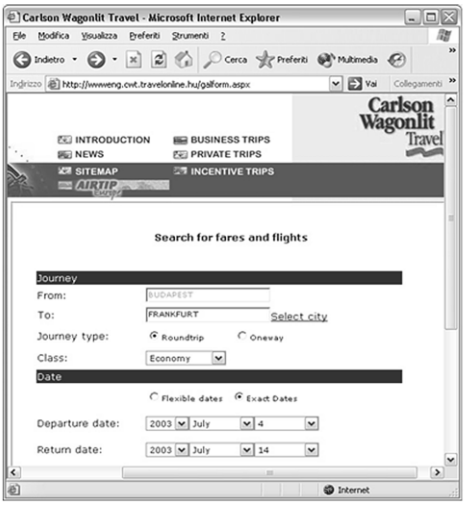


Figura 6. Esempio di dipendenza tra attività attraverso diverse piattaforme.

Per quanto riguarda esempi di attività che hanno senso con una piattaforma ma non con altre un esempio interessante è la partita di calcio. Vi fu un tempo in cui c'era chi era sicuro che questa poteva essere la killer application che avrebbe fatto esplodere il mercato dei cellulari. Questo non è accaduto finora. Non è un problema tecnologico, infatti è possibile vedere partite di calcio su alcuni tipi di cellulari ma ben pochi si sognano di farlo. Il motivo è abbastanza semplice: le persone guardano le partite di calcio per rilassarsi e queste durano almeno 90 minuti. Stare 90 minuti avendo attaccato alla faccia un cellulare (anche di ultima generazione) è qualcosa di alquanto frustrante, considerando che i giocatori che vengono mostrati sono piccolini ed il pallone quasi invisibile. Cosa ben diversa di quando si sta seduti sul sofà con di fronte uno schermo di 40-50 pollici (e magari nel frattempo sorseggiando qualche bibita). Altra cosa sono le radiocronache che consentono in situazioni particolari

(ad esempio mentre si guida) di essere informati o servizi in tempo reale che aggiornano sullo stato del risultato. Questi sono servizi che forniscono informazioni limitate e quindi possono essere veicolati tramite canali con maggiori limitazioni. Conseguentemente, è chiaro che anche la TV fruita tramite cellulare richiede un ripensamento dei contenuti proposti che non possono essere quelli della TV classica.

# 4. La Progettazione di Interfacce Utenti Multi-Dispositivi

La progettazione di interfacce utenti multi-dispositivi può seguire quattro strategie di fondo:

**Sviluppo specifico per ogni piattaforma**, viene sviluppata una versione diversa per piattaforma, questo consente di avere pieno controllo su di esse ma è chiaramente costoso in termini di tempo e lavoro;

**Sviluppo di una versione con differenti sottoversioni**, quindi si crea una versione con la possibilità di indicare piccole modifiche per piattaforme diverse;

**Sviluppo di una versione generale**, una unica versione generica che poi verrà specializzata per le varie piattaforme da qualche supporto a run-time;

**Adattamento automatico**, si crea la versione per una piattaforma e poi c'è un supporto automatico che la adatta alle altre piattaforme.

Un esempio di sviluppo specifico per ogni piattaforma è il Sito Web di Amazon. Come mostrato in Figura 7 vi sono due versioni diverse a seconda se si accede tramite sistema desktop o mobile[1] e, come si può ben vedere la differenza è notevole in quanto la versione mobile è alquanto essenziale, supporta il task principale (ricerca di informazioni) e non fornisce molti dettagli su contenuti ed anticipazioni.

[1] http://www.amazon.com/anywhere


Figura 7. Esempio di versioni diverse a seconda della piattaforma.

Un esempio diverso è proposto nel tool Damask (vedi Figura 8) sviluppato nella tesi di dottorato all'Università di Berkeley di James Lin (Lin 2008). L'idea in questo caso è di avere un ambiente di editing che supporta la conversione di grafici a mano in corrispondenti specifiche della interfaccia utente. L'ambiente consente anche di dire se una parte della specifica è valida per tutte le piattaforme o solo per una specifica (vengono considerate desktop, mobile, e voce). Inoltre, sempre per facilitare l'editing, è possibile sfruttare una libreria di pattern predefiniti che catturano un po' di best practice nella progettazione di interface utenti.

Per quanto riguarda l'approccio basato sullo sviluppo di una versione generale esso può essere ottenuto inserendo nella specifica indicazioni degli autori su come il contenuto si deve presentare o tramite descrizioni basate su modelli che astraggono dalle caratteristiche specifiche delle varie piattaforme.

Per quanto riguarda l'adattamento automatico vi sono varie strategie al suo interno:

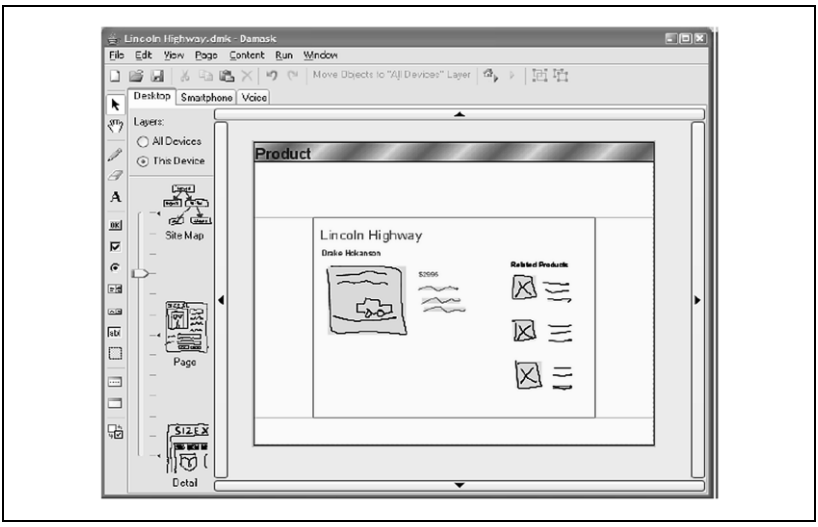


Figura 8. Interfaccia Utente dell'Ambiente Damask.

**Scaling**, un semplice cambiamento di scala che spesso lascia risultati poco usabili;

**Transducing**, converte elementi ed immagini in altri formati, e comprime e converte immagini a seconda delle caratteristiche del dispositivo, come AvantGo[2];

**Transforming**, va oltre nel modificare maggiormente contenuti e struttura.

Esempi di transforming sono:

**Single column**, (per esempio Opera SSR) elimina lo scrolling in una dimensione, aumenta lo scrolling nell'altra;

**Fisheye** è un fisheye Web browser che mostra un focus in una scala leggibile e comprime le zone esterne, un esempio è Fishnet (Baudish, 2004);

**Overview + detail** divide una pagina Web in sezioni multiple e fornisce un overview with links a queste sezioni. La pagina di

[2] http://www.avantgo.com

overview può essere o una immagine thumbnail, o un riassunto della pagina Web.

Nei transforming possiamo citare anche le overviews techniques, come:

**Smartview** (Milic-Frayling & Sommerer, 2002) - Una vista thumbnail in zoom-out, riempie lo schermo orizzontalmente. Partiziona la pagina in regioni logiche; quando una è selezionata il suo contenuto è mostrato in dettaglio;

**Gateway** (Mackay, 2004) – La vista dettagliata usa una tecnica focus-plus-context, allargando la regione selezionata;

**Summary Thumbnail** (Lam, Baudish, 2005)- Usa la vista thumbnail ma i testi sono più brevi e più grandi assicurando una buona leggibilità (le font sono ingrandite ed i caratteri sono presentati finché c'è spazio).

# 5. Progettazione Interfacce Utenti basata su Modelli

I modelli sono astrazioni della realtà. Mirano ad evidenziare gli aspetti principali di interesse senza perdersi in tanti dettagli. I modelli possono essere utili anche quando si progettano o valutano applicazioni interattive.

Nel mondo HCI l'uso di modelli di vario genere è stato attivo fin dai primi anni 80. Possiamo individuare tre generazioni di approcci. La prima generazione mirava a creare dei modi dichiarativi per specificare interface utenti grafiche. Ad esempio in questa generazione troviamo il lavoro del gruppo di Jim Foley al Georgia Tech con UIDE (Foley, 1994) in cui si usavano pre e post condizioni associate con i vari oggetti di interazione. Un altro esempio era Humanoid (Szekely, 1993) che mirava ad esprimere esplicitamente le scelte di progettazione delle varie parti dell'interfaccia utente. Nella seconda generazione di approcci si è passati ad usare modelli di task per supportare la progettazione e lo sviluppo di interface utenti, in quanto questi modelli erano visti un po' come un punto di incontro tra progettisti, sviluppatori ed utenti finali. Esempi di strumenti svi-

luppati in questo ambito erano Adept (Wilson, 1993) e Mobi-D (Puerta, 1999). Negli ultimi anni abbiamo assistito ad un rinnovato interesse in questi approcci in quanto sono visti come uno strumento utile per gestire la complessità derivate dal proliferare di dispositivi interattivi con i loro linguaggi di implementazione. Esempi di approcci che cadono in questa generazione sono UIML (Abrams, 1999), TERESA (Mori, 2004)].

In generale, un sistema interattivo può essere considerato a vari livelli di astrazione. Un possibile modo è quello di considerare i compiti da eseguire per raggiungere gli obiettivi dell'utente e gli oggetti logici che vanno manipolati per il loro svolgimento. Questa è una visione logica del sistema che può essere discussa tra le varie persone coinvolte nella progettazione (utente finale, committente, progettista di interfacce, sviluppatori software). Si può avere un'altro punto di vista, che è sempre logica ma è più focalizzata sull'interfaccia, ovvero considerare le presentazioni e le interazioni che ne fanno parte e come muoversi da una presentazione all'altra. Le interazioni sono identificate in base alla loro semantica (i risultati che consentono di ottenere). Per esempio, si può dire che in un certo punto si ha bisogno di una selezione, ma senza specificare il tipo di modalità richiesta per realizzarla (che potrebbe essere ad esempio selezione grafica, vocale, o tramite un gesto). Vi è, poi, una possibile descrizione più concreta dove si specificano le modalità e le tecniche di interazione che si vogliono usare. Per esempio, si può dire che in un sistema desktop grafico la selezione avviene tramite una lista con una barra di scorrimento. Infine, si ha l'implementazione, che può essere in HTML, Java ecc.. Quando si progetta, il livello di astrazione del punto di partenza può cambiare a seconda dei casi (vedi Figura 9). Certe volte si identificano i compiti da supportare e quelli sono il punto di partenza per ottenere, tramite raffinamenti successivi, l'implementazione. In altri casi, si parte da una certa implementazione che esiste e si creano le descrizioni logiche corrispondenti, ad esempio per cercare di capire se effettivamente quella è la migliore per supportare le attività dell'utente.

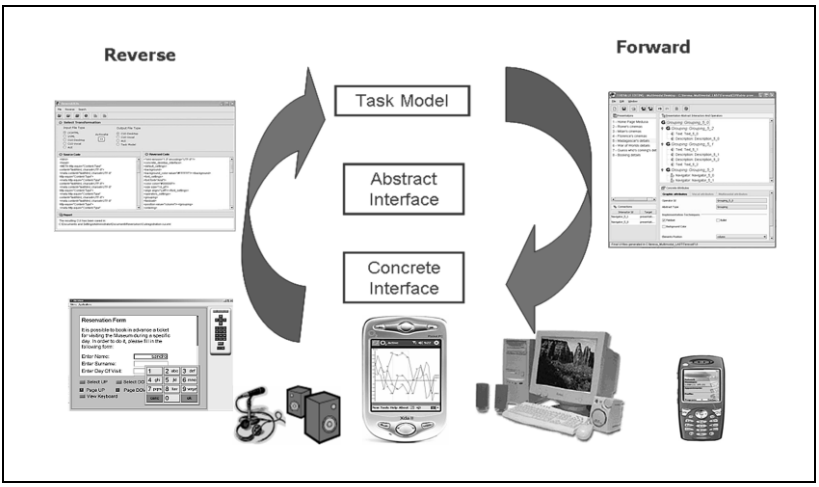


Figura 9. Possibili Trasformazioni tra Livelli di Astrazione.

L'uso di livelli multipli di astrazione ha vari vantaggi: consente di focalizzare sulle principali scelte di progettazione; collegano informazione semantica agli elementi implementativi; e, con il supporto di appropriate trasformazioni consentono di ottenere interoperabilità tra vari linguaggi implementativi.

Per quanto riguarda i modelli di task una notazione largamente usata in università ed aziende è ConcurTaskTrees (Paternò, 1999). Come mostra la Figura 10 le sue caratteristiche principali sono una organizzazione gerarchica delle attività descritte, dove quelle generali sono decomposte in attività più dettagliate; un ricco insieme di operatori temporali che permettono di specificare comportamenti flessibili dove le attività possono andare in sequenza, concorrentemente, interrompersi, ecc. Vi è anche la possibilità con icone diverse di indicare come i compiti devono essere allocati: all'utente, al sistema od ad una loro interazione.

Per ciascun task è possibile specificare inoltre una serie di attributi, che includono anche le piattaforme per cui quel task è significativo.

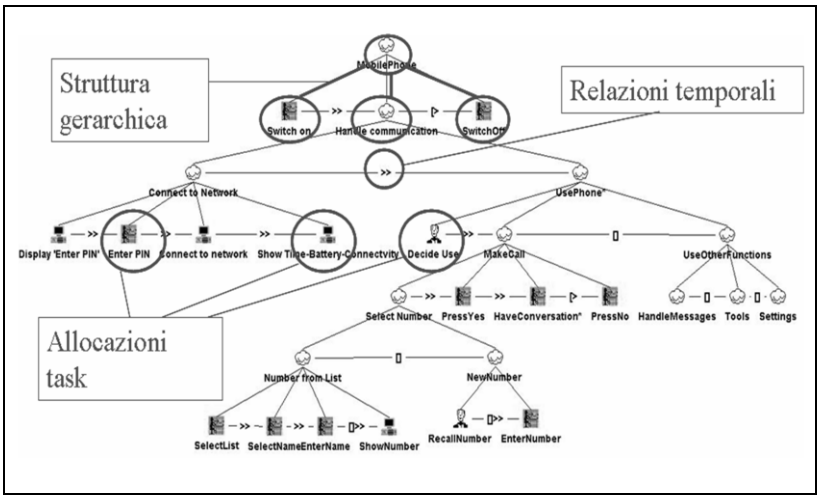


Figura 10. Esempio di specifica in ConcurTaskTrees

Il tool che supporta l'editing e l'analisi di questi modelli di task è il CTTE (ConcurTaskTrees Environment[3]) (Mori, 2002),. Il tool, oltre vari strumenti di editing, consente anche una simulazione interattiva del modello, per cui il progettista può selezionare un task ed il tool mostra quelli successivamente abilitati a seguito del suo svolgimento.

L'informazione contenuta nei modelli di task può anche essere utile per la progettazione di interface concrete che sono coerenti con le loro indicazioni. Le relazioni temporali tra i task possono guidare la strutturazione dei dialoghi dell'interfaccia utente, mentre la struttura dei modelli può fornire utili indicazioni su come strutturare le interface utenti corrispondenti. Ad esempio, se vi sono dei task che sono parte di una stessa attività più generale allora significa che sono strettamente collegati logicamente e gli elementi dell'interfaccia corrispondente dovrebbero essere raggruppati in qualche modo per esprimere tale relazione logica. Inoltre la tipologia del task è utile per identificare le tecniche di interazione più idonee per supportare la sua semantica.

[3] Disponibile all'indirizzo http://giove.isti.cnr.it/ctte.html

Per quanto riguarda gli altri livelli di astrazione (interfaccia astratta e concreta) diversi linguaggi XML basati su modelli per interfacce utenti sono stati proposti: XIML[4] (Puerta, 2002), sviluppato da RedWhale, UIML[5] (Abrams, 1999), sviluppato da Harmonia, TERESA-XML[6] (Mori, 2004), sviluppato al Laboratorio di Interfacce Utenti dell'ISTI-CNR, USIXML[7] (Limbourg and Vanderdonckt, 2004), sviluppato alla Louvain University, ed XForms[8], sviluppato dal W3C.

XForms applica concetti di progettazione model-based sviluppati in ambito di ricerca. Esso separa presentazione da contenuto (i tag per i controlli nella form sono separati dai tipi di dati e valori ritornati alla applicazione). I controlli XForms che possono stare nelle form sono device-independent (select, trigger, output, secret, …). Esso riduce anche il bisogno di script attraverso verifiche effettuate lato client sui dati in formato. In pratica in XForms sono presenti sia il livello astratto (attraverso il vocabolario dei controlli e dei costrutti) che quello concreto (attraverso gli attributi di presentazione dei tipi di dati). Ad esempio XForms consente di specificare un elemento di selezione singola tramite il controllo select1 e poi tramite l'attributo appearance si possono dare vari valori (full, compact, minimal) che determinano diverse implementazioni (rispettivamente radio button, list box, drop down list) a seconda delle caratteristiche del dispositivo corrente.

Un approccio diverso è seguito in TERESA XML dove vi è una chiara distinzione tra livello astratto e livello concreto. Più precisamente in TERESA XML vi è un linguaggio astratto e un linguaggio concreto per ciascuna piattaforma. I linguaggi concreti hanno la stessa struttura del linguaggio astratto ma aggiungo raffinamenti per ogni oggetto di interazione (interattore) o operatore di composizione che indicano aspetti specifici per la piattaforma considerata. La Figura 11 mostra una rappresentazione grafica di un esempio di interfaccia utente astratta in TERESA XML. L'interfaccia è vista come un insieme di presentazioni, in ciascuna presentazione vi sono

[4] http://www.ximl.org/
[5] http://www.uiml.org/
[6] http://giove.isti.cnr.it/teresa.html
[7] http://www.usixml.org/
[8] http://www.w3.org/MarkUp/Forms/

interattori ed operatori di composizione che indicano come organizzare gli interattori. Le connessioni indicano come ci si muove da una presentazione ad un'altra. In sostanza gli operatori di composizione hanno lo scopo di strutturare l'interfaccia utente per indicare elementi che sono raggruppati logicamente (e che quindi bisogna presentare in modo che questo raggruppamento logico sia facilmente percepibile dall'utente).

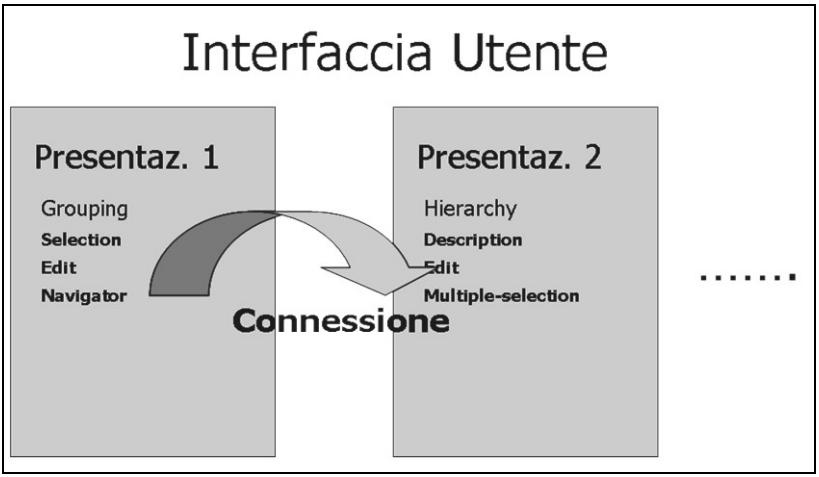


Figura 11. La Struttura di un'Interfaccia Astratta

Quando ci si muove dal livello astratto a quello concreto si danno ulteriori dettagli che dipendono dalla piattaforma considerata. Per esempio a livello astratto possiamo dire che c'è un interattore di tipo navigator (che consente di muoversi da una presentazione ad un'altra), a livello concreto (nel caso di piattaforma grafica) si specificherà anche se si tratta di un link grafico o testuale od un button con associato un link (che sono tre tecniche diverse per supportare lo stesso concetto). A livello implementativo si specificherà poi in ulteriore dettagli a seconda del linguaggio implementativo scelto (XHTML; Java, Windows Forms, …).

L'approccio risultante assume che i progettisti conoscano le potenziali piattaforme (non i dispositivi) dalle prime fasi del processo di progettazione. Il risultato principale è che il metodo consente agli

sviluppatori di evitare una marea di dettagli implementativi (la trasformazione da descrizione concreta a implementazione è automatica). Inoltre è facile aggiungere supporto per nuovi linguaggi implementativi. La Figura 12 mostra un authoring tool che supporta l'editing di specifiche e la generazione di interface con TERESA XML. Sulla sinistra appare la lista di presentazioni correntemente editate, la presentazione selezionata corrisponde alla descrizione astratta che è nella parte a destra in alto, l'elemento correntemente selezionato nella descrizione astratta corrisponde a quello che si può editare a livello concreto nella parte a destra in basso. Nella parte a sinistra in basso vi è la lista di connessioni correntemente disponibili.

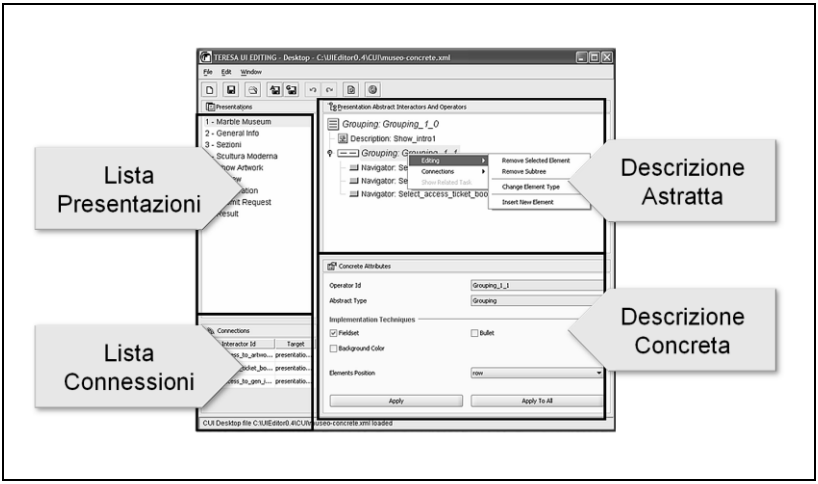


Figura 12. L'interfaccia Utente dell'Ambiente TERESA

# 6. Adattamento Automatico al Dispositivo a Run-Time

In questa sezione discutiamo tecniche di adattamento automatico al dispositivo a run-time, che significa mentre l'utente sta accedendo all'applicazione e non in fase di progettazione.

Vi sono tre fasi principali che vanno attraversate in questo caso:

- identificazione del dispositivo
- identificazione delle risorse del dispositivo
- adattamento.

In ambienti Web la tecnica che viene solitamente usata per identificare il dispositivo è il rilevamento dello User Agent nel Protocollo HTTP. Questo consente di sapere il tipo di dispositivo corrente, il browser, il sistema operativo ed altre informazioni correlate.

Per quanto riguarda le tecniche per l'identificazione delle risorse del dispositivo, possiamo citarne tre:

**CC/PP** (W3C), il Composite Capability/Preference Profiles (CC/PP) è una specifica per definire capacità e preferenze (anche dette 'delivery context') dello user agent. É basato su RDF e mira a fornire solide fondamenta per lo UAPROF.

**UAPROF** (OMA), descrive le capacità di un dispositivo mobile, comprese risoluzione e capacità multimedia. I dispositivi mobili mandano un header (generalmente "x-wap-profile") dentro a http request con lo URL al suo UAProf. La produzione dello UAProf per un dispositivo è volontaria da parte dei costruttori. É un'applicazione dello CC/PP.

**WURFL**, è un file XML di configurazione che può essere memorizzato localmente e contiene informazioni riguardo alle capacità e caratteristiche per una ampia gamma di dispositivi[9].

Lo UAProf ha varie componenti: Hardware Platform, Software Platforms, Network Characteristics, Browser UA, WAP Characteristics, Push Characteristics. Un estratto di quello associato al Nokia Communicator 9500[10], è:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf= "http://www.w3.org/..."
xmlns:prf=http://www.openmobilealliance.org/...
xmlns:mms=http://www.wapforum.org/...
xmlns:pss5="http://www.3gpp.org/...">
<rdf:Description rdf:ID="Profile">
……
```

[9] WURFL è un progetto open source http://wurfl.sourceforge.net/.
[10] Disponibile a http://nds1.nds.nokia.com/uaprof/N9500r100.xml

```
<prf:component>
<rdf:Description rdf:ID="HardwarePlatform">
……
<prf:PixelAspectRatio>1x1</prf:PixelAspectRatio>
<prf:PointingResolution>Pixel</prf:PointingResolution>
<prf:ScreenSize>640x200</prf:ScreenSize>
<prf:ScreenSizeChar>29x5</prf:ScreenSizeChar>
<prf:StandardFontProportional>Yes</prf:StandardFontPropo
rtional>
<prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
<prf:TextInputCapable>Yes</prf:TextInputCapable>
……
```

Da un punto di vista architetturale vi sono tre tipi di soluzioni per supportare l'adattamento automatico (vedi Figura 13). Esse si differenziano essenzialmente su dove l'adattamento viene eseguito.

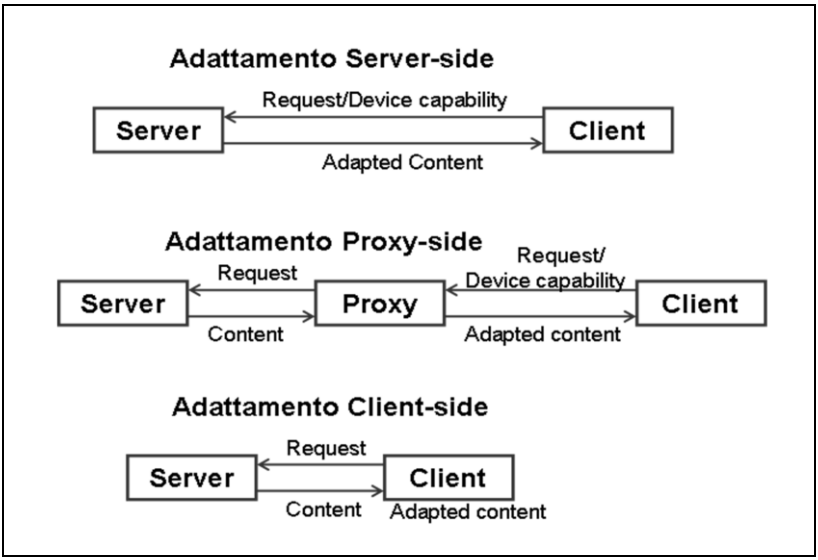


Figura 13. Possibili Soluzioni Architetturali per l'Adattamento

Nel primo caso avviene al lato del server dell'applicazione. Il dispositivo client manda la richiesta di accesso ed un'indicazione delle capacità del dispositivo ed il server applicativo manda il contenuto adattato conseguentemente. Il limite di questa soluzione è che richiede che le funzionalità che eseguono l'adattamento siano dupli-

cate in tutti i server applicativi. Questo viene evitato nella seconda soluzione in cui l'adattamento viene svolto in un proxy server. Questo riceve la richiesta di accesso e l'indicazione delle capacità del dispositivo client, poi gira la richiesta al server applicativo che fornisce i contenuti che vengono adattati dal proxy server prima di essere mandata al richiedente. La terza possibilità è quella in cui l'adattamento avviene direttamente nel dispositivo client. La più semplice forma di client-side adaptation è rappresentata dall'uso di CSS. Il browser Opera è capace di fornire il narrow layout. Il problema è che molti dispositivi mobili hanno limitate capacità di calcolo, memoria ed ampiezza di banda. Tuttavia, le capacità dei dispositivi mobili sono in crescita continua

La maggior parte dei cellulari tradizionali supportano due viste: originale e narrow. L'originale mostra la pagina come era per il desktop. Il vantaggio è che è più familiare ed è più facile trovare il contenuto, lo svantaggio è che se il testo va oltre i limiti dello schermo diventa difficile da leggere e se ci sono spazi bianchi tra le righe è facile sentirsi persi. La Figura 14 mostra la differenza tra la porzione dell'applicazione che viene mostrata da un sistema desktop e da un sistema mobile.

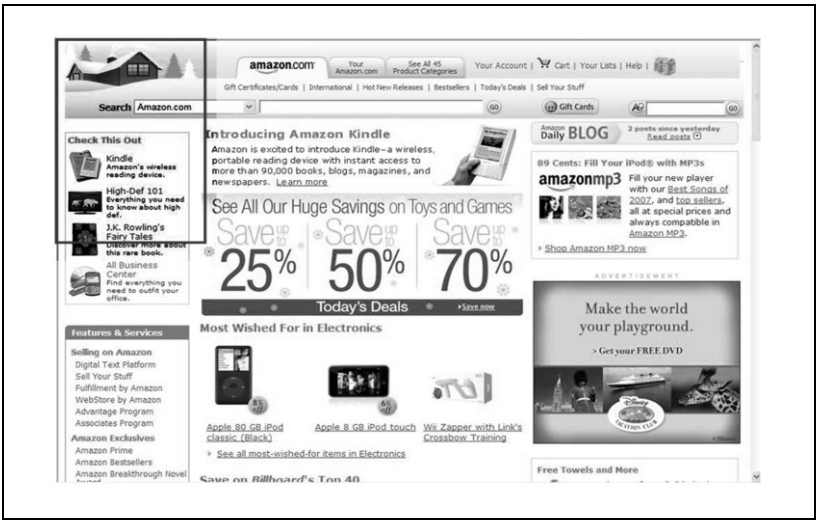


Figura 14. Differenze di vista tra Dispositivo Desktop e Mobile

Nella versione narrow l'ordine del contenuto segue quello del file markup a partire dall'alto (vedi Figura 15). Il testo è impacchettato e le immagini sono scalate all'ampiezza dello schermo. Il testo è sempre visibile ed il contenuto è compattato senza spazi bianchi. Le limitazioni della soluzione narrow sono: contenuto che deve rimanere largo come cartine e tabelle diventa impossibile da leggere; è difficile capire dove si trova il contenuto che l'utente cerca perché il risultato della trasformazione è imprevedibile; certe volte non si capisce che una selezione ha cambiato la pagina perché le pagine condividono la stessa parte iniziale; la trasformazione ha effetti imprevedibili sugli script nella pagina originale. Inoltre, questo tipo di soluzione richiede molto scrolling verticale da parte dell'utente per vedere il contenuto.

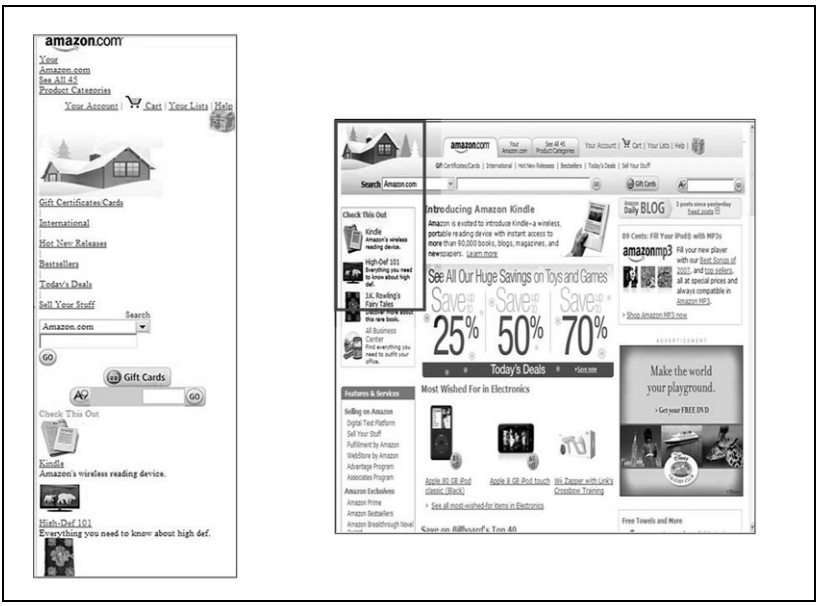


Figura 15. La soluzione narrow per Dispositivi Mobili

Un browser Nokia mira a superare queste limitazioni (Roto et al., 2006). Esso ha l'obiettivo di fare calzare la maggior parte del contenuto allo schermo, eliminare il bisogno di scrolling orizzontale

per leggere il testo, e fornire abbastanza informazione di contesto per dare un'idea della struttura della pagina e comunicare la locazione corrente nella pagina. Fare tutto questo senza distruggere l'originale layout della pagina e senza introdurre interazioni modali. La soluzione si chiama MiniMap ed è ottenuta tramite modifiche alla formattazione delle CSS ed al processo di visualizzazione del browser. Gli elementi non testuali diventano più piccoli e l'ampiezza del testo non deve oltrepassare quella dello schermo. E' una soluzione simile a quelle dell' information visualization tramite una tecnica di overview + detail. Una soluzione tipo fisheye è stata stimata in questo caso eccessiva in tempo di elaborazione per dispositivi mobili. La Figura 16 mostra l'accesso al sito del quotidiano La Repubblica, sulla sinistra si vede una porzione della pagina corrente, con un apposito tasto è possibile attivare la vista sulla destra che mostra tramite un rettangolo rosso come la porzione corrente è posizionata nella pagina complessiva.

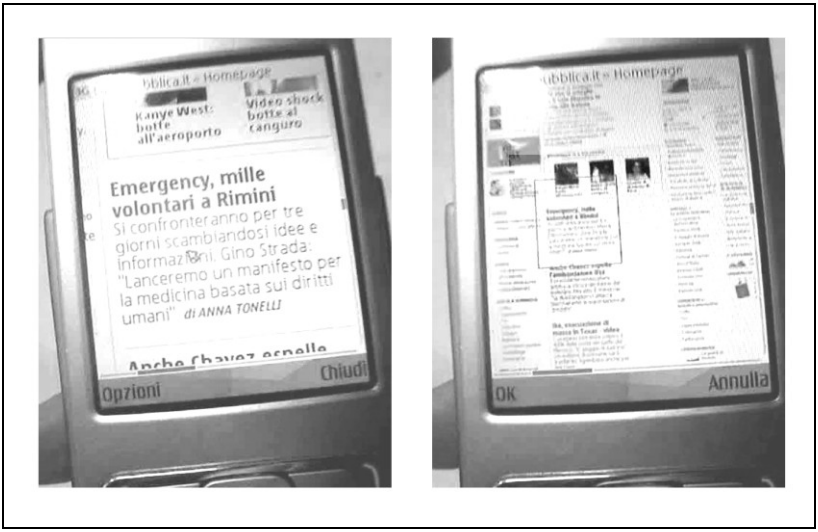


Figura 16. Navigazione con il browser Nokia.

Una soluzione di adattamento tramite server proxy è stata proposta anche dal motore di ricerca Google per presentare i risultati delle

interrogazioni e navigare in essi(Kamvar, 2006). A seconda dello user agent nella richiesta http si veniva ridiretti o a www.google.com/xhtml o a www.google.com/pda. XHTML search è usato per cellulari convenzionali con tastiere a 12 tasti. XHTML PDA veniva usato per dispositivi che hanno tastiere QWERTY o input con penne. Rispetto a quest'ultima, la versione XHTML ha le seguenti caratteristiche:

- ha radio button invece di tab per accedere le varie sezioni
- non ci sono pubblicità
- i pezzi di testo per ogni link sono più piccoli
- i risultati non contengono link cached o a pagine simili e non indicano l'ampiezza della pagina
- l'utente può accedere solo alla pagina precedente o successiva di risultati.

La Figura 17 mostra un esempio dove sulla sinistra c'è il risultato per una interrogazione sul desktop mentre a destra c'è sia la parte di interrogazione che di risultato per la stessa richiesta.

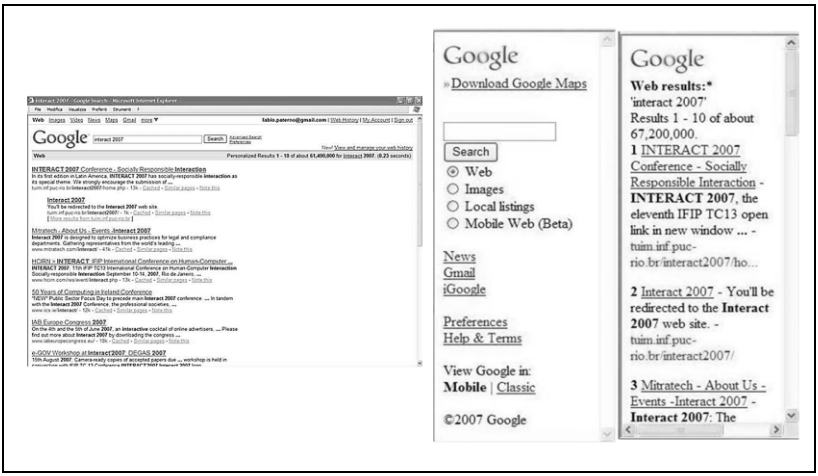


Figura 17. Differenze di risultati di google a seconda del dispositivo

La figura successiva mostra poi come si naviga nel risultato da una parte sul desktop, e dall'altra tramite dispositivo mobile, e quindi si può notare come anche il contenuto selezionato tramite il risultato dell'interrogazione viene adattato.

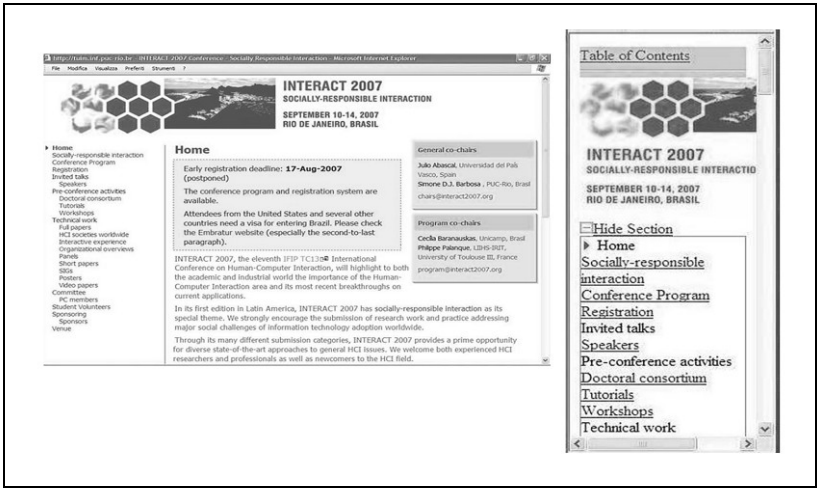


Figura 18. Differenze di navigazione tramite google a seconda del dispositivo

Un altro strumento che supporta adattamento desktop-mobile è Skweezer[11]. La politica di adattamento di default supporta scrolling piuttosto che splitting. Mira a ridurre l'ampiezza delle pagine. Esso include nelle pagine trasformate un CCS fisso orientato per i dispositivi mobili (per esempio: ridefinizione delle fontsize per H1, H2, H3 etc.)

Una tecnica che supporta adattamento sfruttando le descrizioni logiche introdotte precedentemente è Semantic Transformer (Paternò, 2008). Questa tecnica mira a preservare la semantica dell'interfaccia utente quando si adatta da un dispositivo desktop ad uno mobile e dall'altra cerca di adattarlo alle diverse risorse di interazione. L'adattamento avviene spezzando una pagina desktop in più pagine se non è sostenibile per le capacità del dispositivo mobi-

[11] http://www.skweezer.net/

le corrente e cambiando gli elementi di interazione se ve ne sono che hanno gli stessi effetti ma richiedendo meno spazi (ad esempio una scelta tra varie opzioni che nel desktop è supportato da un radio-button può diventare supportata da un pull-down menu che occupa meno spazio). L'algoritmo che determina la divisione di una pagina in pagine multiple analizza la descrizione logica di una pagina e valuta quanto spazio necessita per esser presentata effettivamente, se questo spazio non è sostenibile per lo schermo del dispositivo corrente allora si attiva la procedura di divisione. L'algoritmo va a cercare i gruppi di elementi composti tramite un operatore nella specifica logica, e che quindi sono correlati logicamente e devono restare conseguentemente nella stessa pagina mobile. In particolare va a cercare la composizione che richiede maggior spazio, la toglie dalla pagina originaria e la associa ad una nuova pagina per la versione mobile. Quindi crea i link per poter accedere a questa nuova pagina per la versione mobile e per andare da questa alle altre pagine. Il procedimento continua ricorsivamente a spezzare la pagina desktop in altre pagine mobili fino a che la pagina originaria non è sostenibile per il dispositivo mobile.

# 7. Interfacce Utenti nell'Ubiquitous Computing

L'ubiquitous computing è caratterizzato da molte persone e molti dispositivi comunicanti dinamicamente. Diventa quindi importante trovare soluzioni per fornire supporto continuo all'utente mobile. Le possibilità di accesso remoto furono introdotte dal X Window System che consente di cambiare dinamicamente lo schermo in cui appare un'interfaccia utente. Nell'ambito di ambienti multi-dispositivi un importante contributo fu dato da Rekimoto con la tecnica Pick-and-Drop (Rekimoto, 1997; Rekimoto, 1998). L'idea era dare la possibilità di scambiare facilmente dati tra dispositivi diversi (vedi figura 19) tramite semplici gesti: una selezione dell'elemento nel dispositivo sorgente che poi veniva rilasciato selezionando un punto nel dispositivo target.

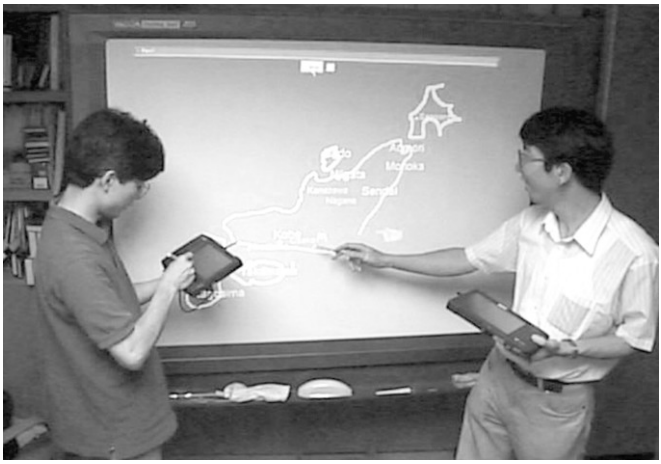

Figura 19. Interfaccia Utente di Pick-and-Drop

Un altro contributo interessante è PUC (Nichols, 2002): esso genera dinamicamente interfacce utenti capaci di controllare dispositivi domestici a partire dalla loro descrizione logica. Un livello aggiuntivo (UNIFORM) (Nichols, 2006) fu aggiunto per generare interfacce utenti consistenti con le precedenti esperienze dell'utente. Gli autori di PUC presentarono uno studio (Nichols, 2007) che indicava che PUC è in grado di generare interfacce utenti superiori a quelle disponibili ad approssimativamente lo stesso costo. Tuttavia PUC non fornisce la possibilità di supportare lo svolgimento continuo di task muovendosi attraverso diversi dispositivi.

In generale, l'accesso flessibile in ambienti Multi-Dispositivi può avvenire in varie modalità:

- muovere oggetti attraverso diversi dispositivi di interazione con pick-and-drop
- interfacce utenti distribuite: la logica applicativa riceve input da diversi dispositivi
- interfacce utenti migratorie: consentono il cambio di dispositivo, con migrazione dell'interfaccia con mantenimento del suo stato.

La motivazione per questo tipo di accesso parte dalla constatazione che la nostra vita sta diventando un esperienza multi-dispositivi, nel senso che le persone sono sempre più circondate da vari dispositivi di interazione. Vi è quindi un bisogno di facilitare l'accesso continuo ai servizi interattivi attraverso diversi dispositivi. Una delle maggiori fonti di frustrazione è che dobbiamo ricominciare la sessione ad ogni cambio di dispositivo. Le interfacce migratorie possono trasferirsi attraverso diversi dispositivi (da dispositivi sorgenti a target) in modo da consentire all'utente di continuare le attività che stanno svolgendo. Domini applicativi che possono beneficiare da questo tipo di interfacce sono shopping, aste on line, giochi, prenotazioni, …

I primi di studi in questa direzione furono di Bharat e Cardelli (Bharat & Cardelli, 1995) che prevedevano la migrazione di intere applicazioni, cosa problematica per dispositive con capacità limitata. Kozuch e Satyanarayanan (Kozuch & Satyanarayanan, 2002) proposero una soluzione per la migrazione basata sull'incapsulamento di tutto lo stato di esecuzione di una macchina virtuale (migrazione di un'applicazione tra desktop e laptop). Chung e Dewan (Chung & Dewan, 1996) invece hanno proposto che quando la migrazione è attivata l'ambiente inizia una nuova copia dell'applicazione nel sistema target e le applica la sequenza degli input di utente salvata. Non c'è supporto per l'adattamento in questa soluzione.

La migrazione può essere totale o parziale. Nel primo caso tutta la interfaccia migra da un dispositivo ad un altro mentre nel secondo solo una parte. Un esempio di migrazione parziale è nella Figura 20 in cui dopo la migrazione i controlli restano nel dispositivo mobile mentre il contenuto si sposta nel dispositivo a schermo largo. La migrazione può essere attivata dall'utente o dal sistema (ad esempio perché ha rilevato che la batteria del dispositivo mobile si sta esaurendo). La scelta del dispositivo al quale migrare può essere fatta analogamente dall'utente o dal sistema.

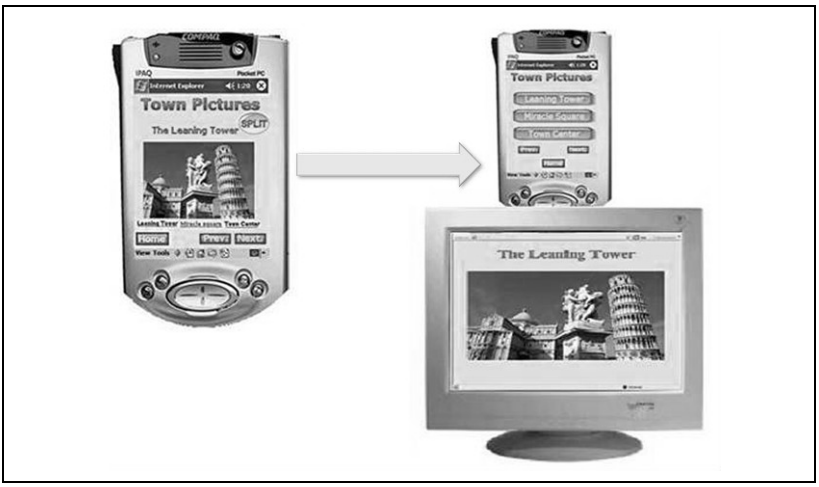


Figura 20. Esempio di interfaccia utente migratoria parziale

Per quanto riguarda l'usabilità, nelle interfacce migratorie sono importanti la continuità e la facilità con cui l'utente continua il proprio task attraverso diversi dispositivi. Fattori che possono influenzare sono il tempo, in particolare il tempo richiesto dalla migrazione per attivare la nuova versione dell'interfaccia nel dispositivo target, ed il processo di adattamento, ovvero l'adattamento dell'interfaccia utente al nuovo dispositivo deve consente all'utente di capire facilmente come continuare il task corrente. É quindi importante anche la prevedibilità del risultato della migrazione per l'utente finale, ovvero consentire all'utente di capire facilmente come continuare i suoi compiti, predire quale è il dispositivo a cui migrare, quale parte dell'interfaccia migra, su quale dispositivo verrà presentato il risultato di un'interazione dopo la migrazione.

# Conclusioni

La nostra vita quotidiana è caratterizzata dalla disponibilità di vari tipi di dispositivi di interazione con capacità diverse. Questo pone la necessità di cambiare il modo in cui le interfacce utenti vengono specificate, progettate e supportate durante le sessioni interattive.

Questo capitolo fornisce una descrizione e discussione delle problematiche relative e delle tendenze correnti. Il forte impulso tecnologico e di mercato, in particolare nell'area dei dispostivi mobili, pone continuamente problematiche e possibilità nuove che stimolano la necessità di nuove soluzioni. Nel capitolo abbiamo visto come l'utilizzo di linguaggi logici, basati su XML, fornisce uno strumento utile per gestire questa complessità, ed ambienti innovativi, come quelli in grado di supportare interfacce utenti migratorie.

## Bibliografia

Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S., Shuster, J. UIML: An Appliance-Independent XML User Interface Lan-guage, Proceedings of the 8th WWW conference, 1999. Available at http://www.harmonia.com/resources/papers/www8_0599/index.htm

Bharat K. A. and Cardelli L.[1995]. Migratory Applications. In proceedings of User Inteface Soft-ware and Technology (UIST '95). Pitts-burgh PA USA. November 15-17. pp. 133-142.

Baudisch, P., Lee, B., Hanna L. [2004] Fishnet, a fisheye Web browser with search term popouts: a comparative evaluation with overview and linear view. AVI 2004: pp. 133--140.

Chung G., Dewan P. [1996]. A mechanism for Supporting Client Migration in a Shared Window System, Proceedings UIST'96, pp.11-20, ACM Press.

Foley, J., Sukaviriya, N., [1994]. History, results, and bibliography of the user interface design environment (UIDE), an early model-based system for user interface design and development, in: Paterno, F. (Ed.), Interactive Systems: Design, Specification, Verification. Springer, Berlin, pp. 3–14.

Kamvar M., Baluja S.. A Large Scale Study of Wireless Search Behavior: Google Mobile Search. Proceedings CHI 2006, ACM Press.

Kozuch M., Satyanarayanan M., Internet Suspend/Resume, Proceedings of the Fourth IEEE Workshop on Mobile Computing Sys-tems and Applications (WMCSA'02) IEEE Press, 2002.

Lam H., Baudisch P. [2005] Summary thumbnails: readable overviews for small screen web browsers. Proceedings CHI 2005, Portland, pp. 681-690, ACM Press.

Limbourg, Q., Vanderdonckt, J., UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence, in Matera, M., Comai, S. (Eds.), Engineering Advanced Web Applications, Rinton Press, Paramus, 2004.

Lin J., Landay J.,: [2008] Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. Proceedings CHI 2008: 1313-1322.

Mori, G., Paterno`, F., Santoro, C., [2002]. CTTE: support for developing and analysing task models for interactive system design. IEEE Transactions in Software Engineering 28 (8), 797–813. IEEE Press.

Mori, G., Paterno`, F., Santoro, C., [2004]. Design and development of multi-device user interfaces through multiple logical descriptions. IEEE Transactions on Software Engineering 30 (8), 507–520.

MacKay, B., Watters, C. R. Duffy, J. [2004] Web Page Transformation When Switching Devices. In Proceedings of Sixth International Conference on Human Computer Interaction with Mobile Devices and Services (Mobile HCI'04) (Glasgow, September 2004), LNCS 3160. Springer-Verlag, 228-239.

Milic-Frayling N., Sommerer R.. [2002] Smartview: Enhanced document viewer for mobile devices. Technical Report, Microsoft Research, Cambridge, UK, November 2002.

Nichols, J. Myers B. A., Higgins M., Hughes J., Harris T. K., Rosenfeld R., Pignol M.. Generating remote control interfaces for complex appliances. Proceedings ACM UIST'02. October 27 – 30. Paris, France. Vol.4, pp.161-170.

Nichols, J., Myers, B.A., and Rothrock, B. UNIFORM: Automatically Generating Consistent Remote Control User Interfaces, in CHI'2006, pp.611-620, ACM Press..

Nichols, J., Chau D., Myers, B.A., Demonstrating the Viability of Automatically Generated User Interfaces, in CHI'2007, ACM Press.

Paternò, F. [1999]. Model-Based Design and Evaluation of Interactive Applications. Springer, Berlin. ISBN 1-85233-155-0.

Paternò F. [2004]. Interazione Uomo-Computer: Un Introduzione, Mondo Digitale, N.4, Dicembre 2004.

Paternò, F. [2005]. Interacting with Computers 17 291–315 315

Paternò, F., Santoro, C., Scorcia, A. [2008] Automatically Adapting Web Sites for Mobile Access through Logical Descriptions and Dynamic Analysis of Interaction Resources. AVI 2008, Naples, May 2008,ACM Press, pp. 260-267

Puerta, A.R., Eisenstein, J., [1999]. Towards a General Computational Framework for Model-Based Interface Development Systems, IUI99: International Conference on Intelligent User Interfaces. ACM Press, New York, pp. 171–178.

Puerta A., Eisenstein J., "XIML: A Common Representation for Interaction Data", Proceedings IUI2002: Sixth International Confer-ence on Intelligent User Interfaces, ACM, Gennaio 2002. Available at http://www.ximl.org/documents/XIMLBasicPaperES.pdf

Rekimoto J.. "A Multiple Device Approach for Supporting Whiteboard-based Interactions", CHI'98, 1998.

Rekimoto J., "Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments", Proceedings of UIST'97, pp. 31-39, 1997.

Roto, V., Popescu, A., Koivisto, A., Vartiainen E.: [2006] Minimap: a Web page visualization method for mobile phones. CHI 2006: 35-44.

Roto V., [2006] doctoral dissertation in Helsinki University of Technology (TKK), Department of Computer Science and Engineering, in 2006, Web Browsing on Mobile Phones – Characteristics of User Experience.

Spence R., [2007] Information Visualization (2nd Edition), Prentice-Hall (Pearson), 2007, ISBN: 0132065509.

Szekely, P., Luo, P., Neches, R., [1993]. Beyond Interface Builders: Model-based Interface Tools, Proceedings INTERCHI'93. ACM Press.

W3C, [2008] Mobile Web Best Practices 1.0 http://www.w3.org/TR/mobile-bp/

Wilson, S., Johnson, P., Kelly, C., Cunningham, J., Markopoulos, P., [1993]. Beyond Hacking: a Model-based Approach to User Interface Design, Proceedings HCI'93. Cambridge University Press, Cambridge, pp. 40–48.

XForms, [2004]. The Next Generation of Web Forms, http://www.w3.org/MarkUp/Forms/.

**Fabio Paternò**, è Dirigente di Ricerca e responsabile del Laboratorio su Interfacce Utenti dell'ISTI-CNR. È stato uno dei pionieri del settore Human-Computer Interaction in Italia nel settore informatico. È stato anche eletto dalla comunità nazionale italiana operante nel settore interazione uomo-macchina Presidente dell'associazione ACM SIGCHI Italy per quattro anni (2000-2004), oltre ad essere designato come rappresentante dell'Italia nel Technical Committee dell'IFIP N.13 (Interazione Uomo-Macchina) dal 1996. Fa parte del gruppo del W3C su Ubiquitous Web Applications e, sempre in ambito W3C, è uno dei fondatori del gruppo su Model-based User Interface Design. Ha pubblicato oltre centosettanta articoli in riviste, libri e conferenze internazionali con processo di revisione con esperti internazionali (una lista è disponibile a http://giove.isti.cnr.it/~fabio/biblio.html). Ha partecipato ai comitati di programma delle principali conferenze al mondo del settore principale di appartenenza, l'interazione uomo-macchina. Ad esempio è stato paper chair della conferenza ACM CHI 2000, la più grande, importante, e selettiva conferenza al mondo del settore Human-Computer Interaction, e conference co-chair di IFIP INTERACT 2005 (che è considerata la seconda al mondo per importanza e qualità nel settore) che si è tenuta a Roma. A questo va aggiunta la responsabilità di importanti progetti internazionali: è stato coordinatore globale di sei progetti Europei (MEFISTO, EUD-Net, GUITARE, CAMELEON ed attualmente OPEN), in questi progetti ha coordinato l'attività scientifica di squadre di ricercatori e sviluppatori provenienti da accademia e industria di diverse nazionalità.. Ha tenuto corsi in varie università in Italia ed all'estero e per varie conferenze ed aziende. Attualmente insegna Progettazione di Interfacce all'Università di Pisa.