Capitolo 1

Analisi dell'utenza potenziale e dei compiti svolti

1.1 Introduzione
1.2 Un caso di Studio
1.3 Progettare l'interazione: un processo iterativo
1.4 Analisi dell'utenza
1.5 Analisi dei compiti (task analysis)
1.6 Analisi degli oggetti (object analysis)
1.7 Realizzazione di Prototipi

1.1. Introduzione

L'utilizzo di tecnologie da parte di fasce crescenti di popolazione può essere visto come un fenomeno positivo, come un segnale di progresso, oppure come il risultato di una spinta non necessaria verso modelli di vita imposti dalla società industriale. Indipendentemente dalla visione che ognuno di noi può avere di questo fenomeno, quello che è certo è che una tecnologia difficile da usare, non 'amichevole', rende la vita più complessa. Gli esempi di interfacce mal progettate, negli oggetti di uso comune così come nel software, sono innumerevoli. Esiste anche per questo un sito web (vedi Scheda 1.1), che segnala casi di errori clamorosi, ma ognuno di noi può esercitarsi ad osservare alcuni degli strumenti che usa comunemente, per scoprirne le difficoltà d'uso ed esaminare le ragioni. Un esempio tipico: il numero di persone che sanno utilizzare un registratore video in modo 'raffinato' (e cioè per eseguire tutti i compiti previsti) è probabilmente minore del numero di apparecchiature di questo tipo oggi installate. D'altra parte, è anche noto a tutti che i manuali d'uso sono quasi sempre mal concepiti, troppo lunghi e quindi parzialmente utilizzati.

Il risultato di questa situazione è che la larga maggioranza degli utenti conosce soltanto un sottoinsieme delle funzioni che ciascuna delle apparecchiature di cui dispone gli permetterebbe di realizzare: malgrado questo, impiega a volte un tempo eccessivo per realizzarle e facendo un numero elevato di errori.

Qual è la causa di questa situazione e come evitare questi problemi? E' questo il quesito a cui un Corso di Interazione[1] cerca di rispondere meglio che può e a cui riesce, in genere, a dare risposte soltanto parziali. Va detto infatti che, come più in generale per la progettazione del software, non esistono regole certe da seguire, che garantiscano la realizzazione di interfacce 'ben fatte' (o, come si dice comunemente, 'usabili'). Esistono linee-guida che facilitano il raggiungimento di un buon risultato e metodi che aiutano ad ottenerlo anche se a prezzo di un certo investimento di risorse. Ma soprattutto, ci vuole inventiva, creatività, spregiudicatezza e ragionamento. L'inventiva e la creatività aiutano a immaginare soluzioni nuove, che

[1] Diremo, d'ora in poi, HCI, per 'Human-Computer Interaction'

tengano conto dell'evoluzione (oggi velocissima) della tecnologia e si adeguino al futuro invece che al presente. La spregiudicatezza aiuta a liberarsi dai modelli dei software commerciali più frequenti, vedendone con chiarezza limiti ed errori (vedi Scheda 1.1). Il ragionamento aiuta a riflettere su come l'utenza prevista utilizzerà l'oggetto che si sta realizzando, quali modelli mentali attiverà e quali regole implicite applicherà. E' quest'ultimo il punto centrale del processo di progettazione di una qualsiasi forma d'interazione con la tecnologia, ed è quindi da qui che partiremo.

Scheda 1.1: Interface Hall of Shame

Questo sito web è una raccolta, continuamente aggiornata, di errori comuni nel progetto delle interfacce. Provare con http://www.iarchitect.com/shame.htm
Ma l'indirizzo cambia spesso, per cui conviene fare una ricerca su 'Interface Hall of Shame', per raggiungere quello più recente. Esempi di errori analizzati:

scelta di oggetti grafici
progettazione di elementi visivi dell'applicazione
terminologia utilizzata
'tabbed dialogs' (dialoghi per etichette)
globalizzazione dell'interazione
interruzioni per domande stupide o informazioni inutili
uso improprio dei colori
messaggi di errore ambigui, scortesi o scorretti
metafore improprie

... e molti altri.
Guardatelo e provate a trovare nuovi esempi: il forum è aperto!

Esercizio 1.1
Analizza alcuni degli errori descritti nel sito 'Interface Hall of Shame' e proponi una tua soluzione.

1.2. Un Caso di Studio

Oltre a discutere numerosi esempi specifici, faremo riferimento, in queste dispense, a un Caso di Studio, descrivendone i passi principali per lo sviluppo dell'interfaccia e discutendo alcuni dei problemi incontrati.

Il Caso di Studio considera il progetto dell'interazione con un *Sistema Informativo per il Medico di Base* (SIMB) che è orientato a medici di medicina generale con diverse caratteristiche (giovani, anziani, convenzionati o non con il Servizio Sanitario Nazionale, con forme diverse di organizzazione dello studio) e si pone gli obiettivi seguenti:

- ottimizzare il compito di cura dei singoli pazienti;
- aggiornare il medico sui progressi scientifici nei metodi di diagnosi e cura;
- garantire il coordinamento dell'attività del medico con quella svolta, sui suoi pazienti, da altre strutture (come ambulatori specialistici, ospedali, ecc);

- permettere al medico di conoscere le caratteristiche complessive della sua utenza e di auto-valutare l'efficacia dell'attività svolta.

Il sistema si basa sull'istituzione di un *database di cartelle cliniche* in cui è possibile fare ricerche di singoli casi, visualizzare e aggiornare l'intera cartella o alcune sue parti, generare e stampare documenti ed elaborazioni statistiche. Questo database principale è organizzato su un pc, che il medico utilizza per le visite nello studio. Il sistema informativo principale è integrato da un sistema-satellite organizzato su un palmare, che viene utilizzato per la gestione dei dati principali raccolti nelle visite a domicilio. In queste Dispense non svilupperemo tutto il Progetto, ma faremo soltanto degli esempi su come applicare a questo Caso di Studio i metodi che descriveremo.

1.3. Progettare l'interazione: un processo iterativo

Abbiamo detto che il progetto dell'interazione non è una disciplina scientifica, anche se richiede metodo e razionalità. La prima conseguenza di questa affermazione è che la progettazione e la realizzazione di questa parte del software si svolge, come tutti gli esperti suggeriscono, attraverso un *processo iterativo* in cui analisi, progettazione e verifica si succedono fino a produrre un risultato 'accettabile' (secondo deteminati standard definiti all'inizio del progetto).

La figura 1.1 riporta i passi essenziali di questo processo. In questa figura si notano alcune tappe principali (analisi dell'utenza e dei compiti svolti) che descriveremo in questo Capitolo, e altre (produzione e test di uno o più prototipi dell'interfaccia) che descriveremo nei Capitoli seguenti. Si nota poi come, a seconda dei problemi evidenziati nella fase di valutazione o test, il processo possa essere iterato ripartendo da fasi diverse (Definizione del dialogo, Piano del Progetto o Produzione di un Prototipo).

A cosa servono le metodologie HCI, se il processo di progettazione va avanti, in ogni caso, per raffinamenti successivi? Servono, ovviamente, a ridurre il numero di iterazioni necessarie, ottenendo un prodotto finale migliore. Un secondo quesito di fondo: qual è la relazione fra l'interfaccia e il software principale (che d'ora in poi chiameremo *applicazione*)? Per anni, queste due componenti del software sono state strettamente intrecciate: i programmi che gestivano l'interazione con l'utente venivano realizzati mediante linee di codice inserite all'interno del codice dell'applicazione. La separazione fra le due componenti si è resa necessaria con l'emergere delle interfacce grafiche, in seguito all'aumento della complessità del software richiesto da questa forma d'interazione.

Questa separazione comporta due vantaggi collaterali: da un lato, la possibilità di realizzare l'*adattamento dell'interazione all'utente* collegando la stessa applicazione a interfacce diverse, progettate sulla base di una analisi delle caratteristiche di chi dovrà utilizzarla: esempio tipico, il collegamento di una applicazione a interfacce in lingue diverse. Vedremo come questo adattamento può essere progettato e realizzato. Dall'altro, la possibilità di *introdurre cambiamenti* in una delle due componenti senza dover necessariamente modificare anche l'altra (ad esempio, migliorare =l'interfaccia lasciando l'applicazione invariata).

La conseguenza di questa separazione è stato lo sviluppo di *metodi* propri dell'interazione: un'interfaccia si progetta, in modo iterativo, partendo da una analisi accurata dell'utenza attuale, potenziale o futura, dei

compiti che questi utenti svolgono, degli oggetti implicati nello svolgimento di questi compiti e delle relazioni esistenti fra utenti, compiti e oggetti. La revisione delle ipotesi di progetto viene sempre realizzata a stretto contatto con un campione dell' utenza.

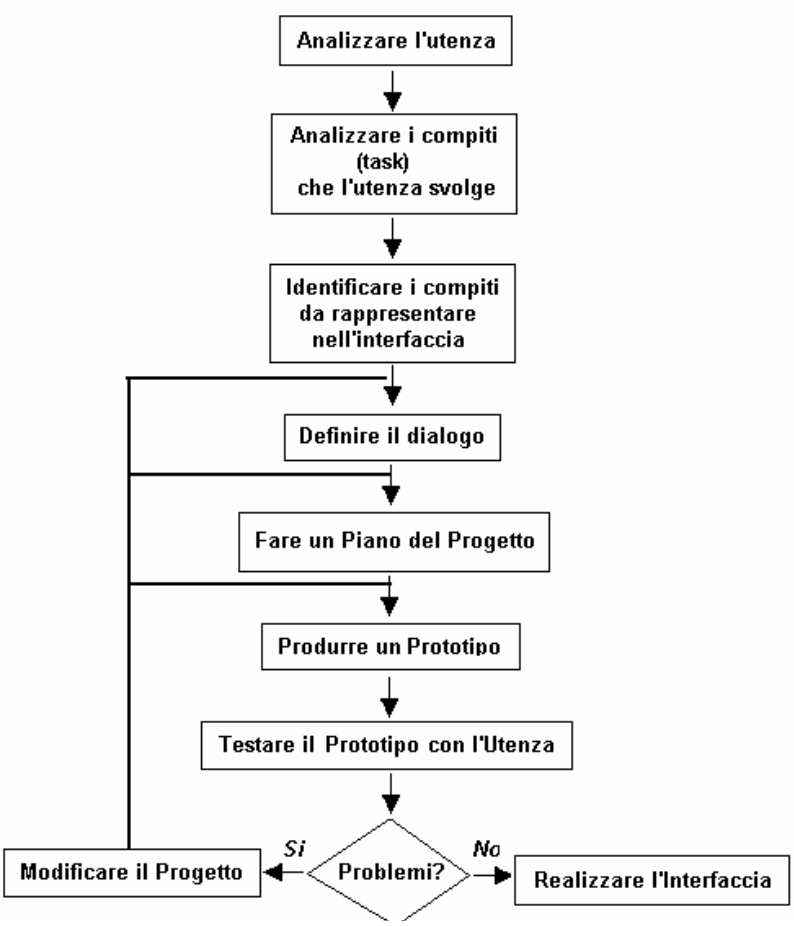

Figura 1.1: Passi principali del processo iterativo di progettazione di una interfaccia

1.4. Analisi dell'utenza

E' evidente che ogni progetto di sistema nasce con obiettivi precisi e, in particolare, con uno sguardo ben orientato sull'utenza potenziale a cui il sistema si rivolge, e si propone di razionalizzare o rendere più piacevole il suo lavoro.

Ogni nuovo sistema si sovrappone ad una situazione precedente, in alcuni casi già automatizzata: ad esempio, un sistema informativo per il medico di base può essere installato in uno studio medico che aveva già un sistema precedente o in uno studio senza esperienze pregresse di automazione. In entrambi i casi, potrà introdurre nuove funzioni o semplicemente modi nuovi di compiere funzioni esistenti e dovrà cercare di rendere questo passaggio il più possibile indolore, o magari piacevole: questo compito è affidato essenzialmente all'interfaccia. Questa non deve essere realizzata semplicemente *immaginando come gli utenti dovrebbero lavorare* ma piuttosto partendo da una analisi di *come lavorano.*

Questa analisi produce una descrizione di quali categorie di utenti presumibilmente utilizzeranno il sistema, quali sono le loro caratteristiche, le loro forme organizzative, i tipi e i livelli di competenza, le caratteristiche del lavoro svolto (individuale o di gruppo). Può essere realizzata con una combinazione di diversi metodi:

- *Studio della documentazione esistente,*
- *Osservazione* del modo in cui gli utenti lavorano,
- *Interviste* agli utenti,
- *Questionari* somministrati a un campione significativo di utenti,
- *Invito agli utenti a 'pensare a voce alta',* registrando le loro riflessioni.

I metodi nella lista hanno complessità e costi crescenti e producono, di conseguenza, risultati sempre più dettagliati. La combinazione più opportuna dipende dalla particolare applicazione considerata.

I risultati di questa analisi possono essere formalizzati in schede e sono il punto di partenza per descrivere uno *scenario dei cambiamenti che, in questo stile di lavoro, verranno prodotti dal nuovo sistema:* l'analisi della plausibilità di questi scenari è il primo passo di una valutazione dell'usabilità del sistema progettato.

Definizione 1.1
Chiameremo *scenario* una descrizione informale delle attività che un utente o un gruppo di utenti svolgeranno con l'uso del sistema da progettare.

La Scheda 1.2 riporta uno scenario relativo all'uso di uno 'Smartphone' (cellulare intelligente) che risolve problemi comuni di persone in viaggio. Questo esempio descrive i *compiti* che il sistema si propone, i *contesti* in cui questi compiti vengono svolti e i *risultati* che essi producono.

Scheda 1.2: Uno Scenario d'uso di uno Smartphone ('cellulare intelligente')
(da Preece, Rogers and Sharp, 2002, p.225)

"A businesswoman is traveling from San Francisco to Paris on a business trip. On her way to the airport, she narrowly misses a traffic delay. She avoids the traffic jam because her Smartphone beeps, then sends her a text message warning her of the traffic accident on her normal route from her office to the airport.
Upon arrival at the airport, the location-sensitive Smartphone notifies the airline that she will be checking in shortly, and an airline employee immediately finds her and takes her baggage. Her on-screen display shows that her flight is on time and provides a map to her gate. On her way to the gate, she downloads tourist information such as maps and events occurring in Paris during her stay.
Once she finds her seat in the plane, she begins to review all the information she has downloaded. She notices that an opera is playing in Paris that she has been wanting to see, and she books her ticket. Her Smartphone can make the booking using her credit card number, whixh it has stored in its memory. This means that she does not need to reenter the credit card numbetr each time she uses wCommerce (i.e. wireless commerce) facilities. The security written into the software of the Smartphone protects her against fraud.
The Smatphone stores the opera booking along with several emails that she writes on the plane. As soon as she steps off the plane, the Smartphone makes the calls and automatically sends the emails.
As she leaves the airport, a map appears on the Smartphone's display, guiding her to her hotel.

Esempio sul Caso di Studio 1.1

Nell'analisi dell'utenza per il sistema informativo per il medico di base, sono individuabili i passi seguenti:

- Definire un 'campione' significativo degli studi medici a cui il sistema è indirizzato.
- Analizzare chi lavora nello studio, quali compiti svolge e quali di questi compiti potrebbero essere affidati al sistema, il modo in cui ciascuno degli utenti lavora, il loro livello di esperienza nell'uso di mezzi di calcolo.
- Classificare, se necessario, gli studi medici in diverse 'tipologie di utenza'.
- Sintetizzare i risultati dell'analisi in relazione alle tipologie di utenti individuate

Supponiamo ora che l'analisi abbia evidenziato situazioni diverse. Fra queste, esaminiamo in particolare le seguenti:

Ipotesi 1: studio medico convenzionato, con segretaria

- una *segretaria* e' responsabile dell'inserimento dei dati, in parte prima della visita, in parte alla fine; e' anche responsabile della stampa dei documenti (prescrizioni di farmaci, richiesta di esami e visite specialistiche, ecc);
- il *medico* accede alle informazioni memorizzate: quelle relative al paziente e quelle relative ai farmaci nel prontuario; utilizza queste informazioni per prendere le sue decisioni e riceve dal sistema un aiuto nel suo processo decisionale sulla diagnosi e la terapia. Effettua personalmente analisi periodiche dell'attività svolta.

Ipotesi 2: studio medico convenzionato, senza segretaria:

il *medico* accede alle informazioni sul paziente che sono state memorizzate in visite precedenti; le aggiorna durante o dopo la visita; stampa i documenti da consegnare al paziente (prescrizioni di farmaci, richiesta di esami e visite specialistiche, ecc); utilizza queste informazioni per prendere le sue decisioni e riceve dal sistema un aiuto al suo processo decisionale.

Un ulteriore fattore significativo per caratterizzare l'organizzazione del lavoro è dato (come vedremo meglio in seguito) dall'organizzazione delle visite: per appuntamento o per ordine di arrivo.

Esercizio 1.2

Consideriamo il seguente scenario d'uso nel nostro sistema per il medico di base:

"Il Medico arriva nello studio ed esamina la lista (preparata dalla sua segretaria) dei pazienti prenotati per quel giorno. Quando un paziente entra, ricerca la sua cartella e ne visualizza il contenuto (patologie croniche e recenti, terapie in atto, esiti di queste terapie). Quindi, chiede al paziente quali sono i suoi problemi attuali e aggiorna la cartella con i nuovi dati. Infine, lo visita, definisce la terapia e aggiorna la cartella".

Sulla base della tua esperienza personale, ti sembra che questo scenario sia plausibile? Credi che lo sia in alcuni casi ed in altri no?

Le risposte a questo esercizio saranno probabilmente diverse per ognuno di voi, e dipenderanno dal *tipo di medico* con cui ognuno di voi si è trovato ad interagire. Per progettare l'interazione, bisogna prima di tutto conoscere bene gli utenti a cui il sistema si rivolge ed esaminare se esistano *tipologie diverse* di utenza, con bisogni e organizzazioni del lavoro diversi, che richiedono soluzioni diversificate. In particolare, gli utenti potranno essere distinti in base al loro rapporto con il sistema:

- gli utenti *diretti* interagiscono direttamente con esso, mentre quelli
- *indiretti* ricevono output generati da altri utenti;

o in base alla *frequenza* con cui lo utilizzano:

- gli utenti *primari* lo utilizzano frequentemente, mentre quelli
- *secondari* lo utilizzano soltanto in modo occasionale*;*

o in base al loro livello di esperienza nell'uso della tecnologia, o di alcune funzioni in particolare.

E' evidente che la classificazione di un insieme di utenti sulla base delle prime due tipologie dipende fortemente dalla loro organizzazione del lavoro: un utente potrà interagire direttamente o indirettamente, frequentemente o raramente con il sistema, a seconda di come è organizzato il lavoro della struttura in cui è inserito.

Esempio sul Caso di Studio 1.2

Sempre nel caso del SIMB:

nell' Ipotesi 1:

- la *segretaria* e' un utente diretto e frequente, per tutti i task relativi all'inserimento e all'aggiornamento dei dati e per le funzioni di stampa dei documenti; si può ipotizzare che divenga (dopo una fase di addestramento) esperta in questi task;
- il *medico* è un utente diretto e frequente per i task di ricerca e visualizzazione dei dati e di aiuto alla decisione; si può ipotizzare che diventi esperto in questi task anche se inizialmente non lo è; può essere anche un utente occasionale per i task generalmente svolti dalla segretaria;
- il *paziente* e le strutture sanitarie esterne sono utenti indiretti per i task di stampa dei documenti.

nell' Ipotesi 2:

- il *medico* è un utente diretto e frequente per i task relativi alla ricerca e alla visualizzazione dei dati, all'inserimento ed aggiornamento di nuovi dati e alla stampa dei documenti e per i task di aiuto alla decisione; si può ipotizzare che diventi esperto in questi task;
- il *paziente* e le strutture sanitarie esterne sono utenti indiretti per i task di stampa dei documenti.

In entrambe le ipotesi, il titolare dello studio e la sua segretaria sono utenti primari del sistema, mentre chi sostituisce il titolare nel periodo di vacanza è un utente secondario.

Esempio sul Caso di Studio 1.3

Riprendiamo l'esempio descritto nell'Esercizio 1.2:

"il Medico arriva nello studio ed esamina la lista (preparata dalla sua segretaria) dei pazienti prenotati per quel giorno. Quando un paziente entra, ricerca la sua cartella e ne visualizza il contenuto (patologie croniche e recenti, terapie in atto, esiti di queste terapie). Quindi, chiede al paziente quali sono i suoi problemi attuali e aggiorna la cartella con i nuovi dati. Infine, lo visita e definisce la terapia."

Questo scenario descrive una delle possibili ipotesi organizzative dello studio (visite su prenotazione, con compiti limitati per la segretaria) e quindi uno dei modi in cui il task di 'Gestione dell'archivio' può essere effettuato, nell'ipotesi 1. Uno scenario alternativo, per la stessa ipotesi organizzativa, è il seguente:

"il Medico arriva nello studio ed esamina la lista dei pazienti prenotati per quel giorno. La segretaria ha selezionato, in un archivio temporaneo, le cartelle dei pazienti prenotati, organizzandole in ordine di prenotazione. Prima che ogni paziente venga introdotto, il medico visualizza ed esamina il contenuto della 'parte principale' della sua cartella. Quindi, chiede alla Segretaria di introdurre il paziente e lo visita, aggiornando la cartella e inviando un messaggio alla Segretaria, la quale provvede a stampare prescrizioni ed altri eventuali documenti."

E' evidente che è possibile immaginare numerosi altri scenari, sia per l'ipotesi 1 che per l'ipotesi 2. Ad esempio, se le visite sono organizzate per ordine di arrivo, entrambi gli scenari dovranno essere rivisti radicalmente. E' dall'analisi del modo in cui campioni di medici lavorano che sarà possibile isolare quelli più comuni, secondo i quali organizzare l'interazione. E' anche chiaro come separare l'interfaccia dall'applicazione consenta, anche in questo caso, di adattare le modalità d'interazione alle diverse situazioni organizzative in cui il sistema verrà installato.

Esercizio 1.3

Prova a immaginare qualche scenario per l'ipotesi organizzativa 2 (studio senza segretaria), in uno studio in cui le visite sono organizzate per ordine di arrivo.

Quando l'analisi dell'utenza individua diverse tipologie di utenti, con diversi scenari d'uso del sistema, il progetto dell'interazione non potrà essere unico, ma dovrà adattarsi alle tipologie individuate: quello dell'*adattamento dell'interazione agli utenti* è uno dei temi che approfondiremo in queste dispense, trattandolo in particolare nel Capitolo 3.

1.5. Analisi dei compiti (Task Analysis)

Definizione 1.2

Chiameremo *task* un insieme di procedure ed azioni che un generico utente compie su un oggetto, per raggiungere un obiettivo inerente al lavoro svolto.

La scheda 1.2 comprende la descrizione (anche se approssimativa) di alcuni task svolti dall'utente esaminato:

fare viaggi di lavoro
trasferirsi in macchina all'aeroporto
evitare ingorghi
fare il check-in del volo
consegnare i bagagli
controllare l'orario del volo
cercare il gate
scaricare dati sugli eventi in corso nel luogo di destinazione
visualizzare queste informazioni
prenotare biglietti per uno spettacolo
scrivere email
spedire un insieme di email precedentemente scritti
localizzare l'albergo

Come si può notare, si tratta di task con livelli di complessità diversi; alcuni di essi sono più specifici e fanno parte di task più generali elencati in precedenza. Ad esempio, fare viaggi di lavoro richiede, innanzitutto, di recarsi (in macchina o con altri mezzi) in aeroporto; se il percorso è fatto in macchina, questo richiede di evitare ingorghi, eccetera.

Esempio sul Caso di Studio 1.4

In ogni situazione, il SIMB deve permettere ai suoi utenti di:

- cercare nel database la cartella di un paziente;
- visualizzare i dati relativi ai 'Problemi' o alle sezioni 'Fisiologia', 'Fattori di rischio', 'Patologie' o 'Terapie' nella cartella;
- aggiornare i dati in una o più di queste Sezioni;
- compilare e stampare una prescrizione di farmaci;
- ricevere avvisi e suggerimenti su possibili effetti collaterali, controindicazioni, costi di un farmaco o sulle interazioni fra diversi farmaci prescritti;
- ricevere suggerimenti sulla diagnosi più plausibile in un determinato caso;
- fare analisi statistiche dei dati.

Se lo studio è organizzato per appuntamenti, il sistema deve anche consentire di creare liste di pazienti prenotati e database temporanei delle loro cartelle. Saranno definiti, per alcune categorie di utenti (le segretarie) vincoli sull'accesso ad alcuni task: ad esempio, l'aggiornamento di alcuni dati 'critici'.

In generale il processo di analisi dei task compiuti dagli utenti può essere effettuato per livelli successivi di approfondimento (cioè in modo top-down). Questa analisi si riflette in una *descrizione gerarchica dei task*.

Definizione 1.3

Una *gerarchia di task* è una struttura ad albero la cui radice rappresenta il task generale che l'applicazione si propone di compiere. Dato un nodo n_i, i suoi nodi-figlio n_{i1}, n_{i2}, …, n_{ik}, … rappresentano i task più semplici in cui il task associato ad n_i si decompone. I nodi-foglia (quelli che non hanno figli) corrispondono a 'task elementari', che non possono essere ulteriormente decomposti.

Vedremo, nei prossimi Capitoli, quali sono i vantaggi di questa descrizione gerarchica: descrizione *formale* dell'interazione per livelli successivi di approfondimento (ad esempio, con le Reti di Petri) e rappresentazione della gerarchia di task negli 'oggetti grafici' dell'interfaccia.
In particolare, se in una interfaccia gli oggetti grafici rispettano (nel loro aspetto e nella loro organizzazione) la logica che l'utente segue abitualmente per svolgere il suo lavoro, logica che a sua volta è rappresentata nella gerarchia di task, capire come l'applicazione si usa sarà più semplice per gli utenti.

Nella larga maggioranza dei casi, non tutti i compiti che gli utenti svolgono verranno automatizzati: la descrizione dei task nella gerarchia può essere *completa* o limitata ai task che l'applicazione automatizza. Nel primo caso, verranno indicati con una label opportuna i task da automatizzare. Un'altra label indicherà i cosiddetti 'task comuni', che possono essere eseguiti in fasi diverse dell'applicazione: esempi tipici, il salvataggio dei dati o la stampa. Evidenziare questi task (non duplicando la descrizione della loro decomposizione nella gerarchia) aiuta a rispettare criteri di *consistenza* nell'interfaccia: garantisce cioè che task eguali vengano eseguiti, in punti diversi dell'applicazione, in modo eguale. Vedremo meglio questi concetti nel prossimo Capitolo, quando parleremo di *usabilità.*

Esempio sul Caso di Studio 1.5

La figura 1.2 mostra una parte della gerarchia dei task per il SIMB. Nella figura, sono decomposti soltanto alcuni subtask; 'input di cognome e nome', della 'data di nascita' e del 'numero di cartella' sono task elementari. I task più frequenti sono indicati con un asterisco. Il task di 'Gestione del DB su palmare' è responsabile dell'integrazione fra sistema informativo principale (gestito nello studio) e secondario (gestito nelle visite a domicilio).

Esercizio 1.4

Costruisci la gerarchia dei task che un generico telefono cellulare permette di effettuare, indicando i task più frequenti e quelli comuni. Analizza ora l'interfaccia del tuo cellulare. Noterai, probabilmente, che la gerarchia che avevi definito non è ben riflessa nell'albero di menu/sottomenu adottata dall'interfaccia del cellulare che usi. Commenta le differenze e proponi una riorganizzazione di questa interfaccia.

Il concetto di decomposizione di un task n_i non va inteso come elencazione di elementi *tutti* necessari per eseguire n_i: non bisogna cioè assumere che i nodi-figliodi n_i siano tutti in relazione di 'and' fra loro. In realtà, la gerarchia può includere relazioni diverse: *decomposizioni alternative,* quando il task associato a n_i può

essere eseguito in modi diversi (in questo caso, fra i nodi-figlio di n_i esistono relazioni di 'or') oppure situazioni di 'and' in cui l'ordine di esecuzione dei subtask può essere scelto dall'utente con margini più o meno ampi di libertà.

La semplice rappresentazione gerarchica dei task non permette di rappresentare i diversi tipi di relazione che esistono fra di essi. Altri formalismi possono essere applicati per integrare l'informazione contenuta nella gerarchia di task.

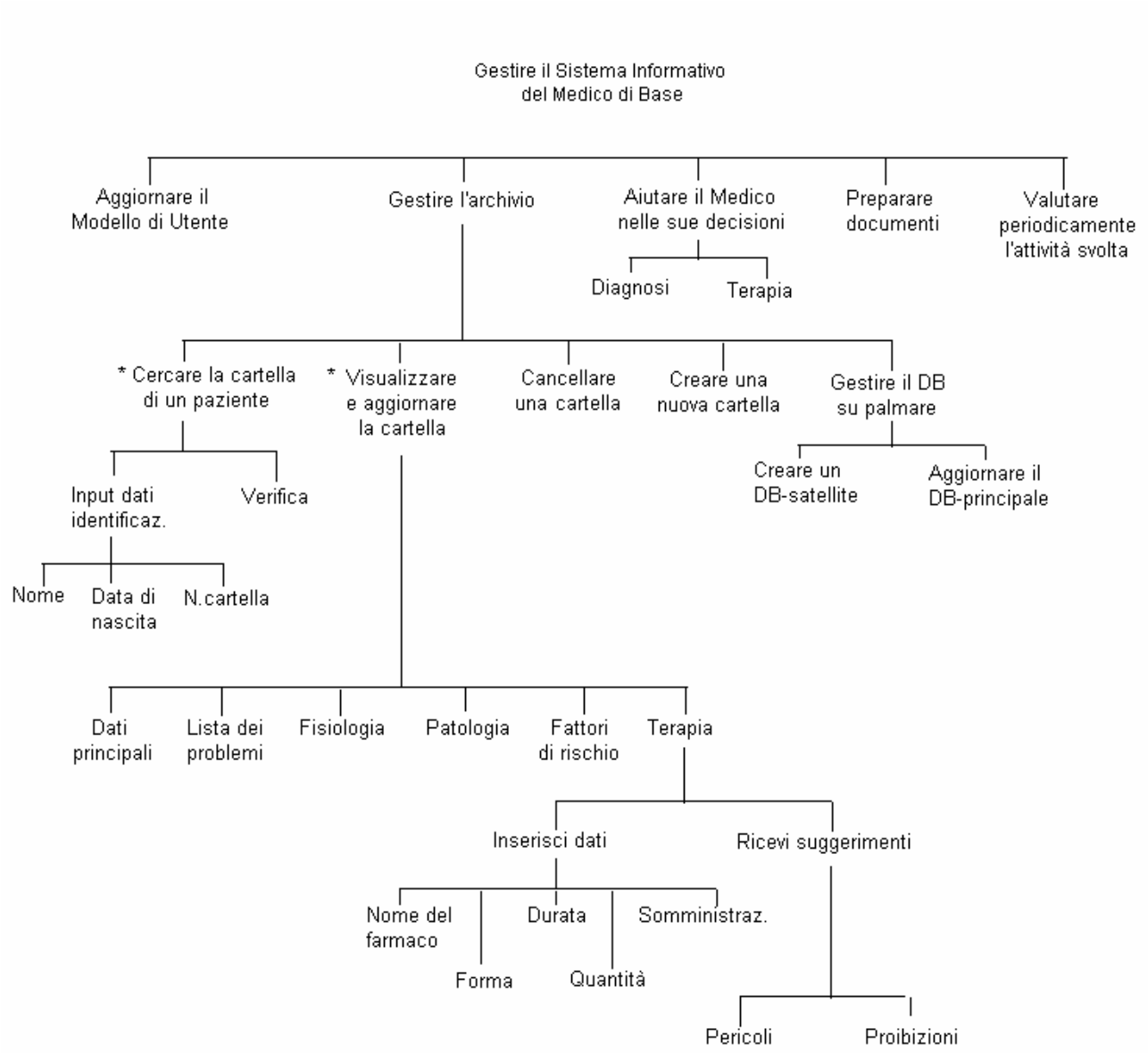

Figura 1.2: Una parte della gerarchia dei task per il SIMB

Definizione 1.4: 'User Action Notation' (UAN) : Palanque e Bastide, 1999

Task complessi possono essere descritti per combinazione delle seguenti relazioni elementari:

sequenza: eseguire il task A prima del task B	A B
iterazione: eseguire il task A f volte ((A)* se f non definito)	$(A)^f$
scelta: eseguire il task A oppure il task B	A|B
indipendenza d'ordine: eseguire prima il task A e poi il task B, o viceversa	A&B
concorrenza: eseguire il task A concorrentemente al task B	A||B
interrompibilità: il task A può essere interrotto dal task B	A→B
interleavability: ognuno dei task A e B può interrompere l'altro	A⇔B

Esempio 1.1

L'inserimento di dati in un form senza vincoli di precedenza è un caso tipico di indipendenza d'ordine. Se il form contiene tre elementi a, b, c, e denotiamo con A il task elementare d'inserimento dei dati relativi ad a (e analogamente per B e C), il task complesso di compilazione del form può essere rappresentato da: A & B & C. Se esiste un vincolo di precedenza nell'inserimento dei dati, per cui il task A deve precedere il task B, avremo la relazione: ((A B) & C). Questo avviene, ad esempio, quando il dato a introduce un vincolo sul valore del dato b. Se l'intera operazione può essere ripetuta più volte, il task sarà denotato da: (A & B & C)* mentre, se ognuno dei task elementari può essere iterato, lo denoteremo con (A)*&(B)*&(C)*. Infine, la scelta ripetuta (che consiste nell'eseguire il task A o il task B più volte) può essere rappresentata come: (A|B)*.

Esempio sul Caso di Studio 1.6

L'Input dei dati di identificazione' può essere rappresentato come una combinazione di scelta e indipendenza d'ordine:
(('Input di nome' & 'Input di data di nascita') | 'Input di Numero di Cartella').
La 'visualizzazione della terapia' può essere rappresentata come una concorrenza:
('Inserisci dati' || 'Ricevi suggerimenti').
Il task di 'Creare nuona cartella' può essere iterato… ecc.

Esercizio 1.5

Prova a descrivere con una UAN la relazione fra i task elementari nella procedura di 'Salva con nome' in Word.

1.6. Analisi degli oggetti (Object Analysis)

Nella fase di Object Analysis, vengono descritti gli oggetti (concreti o astratti) che ogni categoria di utenti manipola nell'eseguire ogni task.[2] Anche fra gli oggetti esistono, in genere, relazioni (di 'part-of', 'is-a' o altre). Queste possono essere descritte con metodi comuni nella progettazione del software, come i Diagrammi Entità-Relazioni[3]. Fra Utenti, Task e Oggetti esistono poi relazioni evidenziabili nella fase di task analysis: ogni utente esegue un insieme di task con frequenza diversa; ogni task si applica ad uno o più oggetti; ogni oggetto può contenere, a sua volta, diversi oggetti o essere in relazione di 'is-a' con altri. Queste relazioni possono essere descritte arricchendo ulteriormente la gerarchia dei task.

Esempio 1.2

Oggetti coinvolti in Word: il documento, l'archivio, il paragrafo, la porzione di testo, il carattere, la tabella, l'immagine (molti dei quali legati fra loro da una relazione di part-of); ma anche la stampante, la tastiera, i tasti-funzione, ecc

Oggetti coinvolti in Paint: disegni, forme geometriche, linee, colori, …

Frequenza nell'esecuzione dei task: sono frequenti i task di formattazione del testo (scelta dei caratteri e giustificazione dei paragrafi; più rari, i task di conteggio di parole, sunto automatico, definizione di macro.

Esercizio 1.6

Prova a elencare gli oggetti, astratti e concreti, coinvolti nell'applicazione descritta nella Scheda 1.2 (Smartphone).

Esempio sul Caso di Studio 1.7

Nel SIMB:

- *Esempi di oggetti 'astratti':* modello dell'utente, problema, fisiologia, fattori di rischio, patologia, terapia,
- *Esempi di oggetti 'concreti':* paziente, cartella clinica, farmaco, lettera di dimissione ospedaliera
- *Relazioni fra oggetti:* dati di identificazione e lista dei problemi sono *parte della* 'componente principale' della cartella clinica; la prescrizione di un farmaco *è parte della* componente 'terapia'; la lettera di dimissione ospedaliera *è parte della storia* del paziente; il modulo di prescrizione del SSN e la lettera di dimissione ospedaliera *sono esempi di* documenti; una terapia *si applica ad* un problema

  ……
- *Relazioni fra Utenti, Task e Oggetti:* queste sono facilmente derivabili dalle descrizioni delle attività negli Scenari. Ad esempio: nell'ipotesi 1, la segretaria trova la cartella clinica di un paziente e la aggiorna; il medico visualizza il contenuto di una o più sezioni della cartella, prescrive uno o più esami specialistici, stabilisce una diagnosi e prescrive uno o più farmaci (eventualmente con l'aiuto del sistema).

[2] Attenzione: alcuni di questi oggetti verranno rappresentati come oggetti grafici nell'interfaccia, ma non è vero il contrario: non tutti gli oggetti grafici dell'interfaccia rappresentano oggetti manipolati dagli utenti nel corso delle loro attività: vedremo bene questa differenza nel Capitolo 5.

[3] Che diamo per noti, da uno dei Corsi precedenti

1.7. Realizzazione di prototipi

Un prototipo può avere livelli diversi di 'fedeltà'. Può andare da uno *storyboard* su carta a un pezzo più o meno complesso di software realizzato con un GUI, cioè un tool per lo sviluppo di interfacce come Visual Basic.

Definizione 1.5: Storyboard

Uno storyboard è una sequenza di disegni che illustrano, in modo più o meno schematico, come l'interfaccia del sistema evolverà durante l'esecuzione di un task. Se utilizzato in combinazione con uno scenario, indica in modo approssimato, ma comprensibile per gli utenti che collaborano al progetto dell'interazione, come le diverse categorie di utenti interagiranno con il sistema stesso.

Esercizio 1.7

Prova a disegnare lo storyboard dell'applicazione descritta nella scheda 1.2

Nella progettazione dell'interfaccia e ancor più nella realizzazione di prototipi, occorre stabilire, prima di tutto, un *paradigma* d'interazione, e cioè l'ambito nel quale il sistema verrà utilizzato: il paradigma d'interazione influenza non soltanto la dimensione del display, le modalità di input/output, la memoria e la capacità di calcolo, ma anche la modalità con cui l'interazione fra utente e sistema avverrà.

Esempio 1.3

Riprendiamo l'esempio dello smartphone descritto nella Scheda 1.2: è evidente che le funzioni descritte non potrebbero essere realizzate con una tecnologia diversa e che, d'altra parte, la scelta del paradigma (mobile computing) determina vincoli di progetto ben precisi relativamente alle dimensioni del display, all'input dei dati, alle condizioni esterne in cui il dispositivo verrà utilizzato eccetera.

La Scheda 1.3 descrive i principali paradigmi proposti negli anni più recenti, mentre la Scheda 1.4 ne descrive più in dettaglio alcuni, a partire dalle definizioni date nei Call di recenti Congressi scientifici sull'argomento. Per un approfondimento sul possibile ruolo del mobile computing in medicina, si veda, in particolare, la Scheda 1.6.

Esempio sul Caso di Studio 1.8

Nel caso del SIMB, la necessità di un accesso 'ubiquo' al sistema (sia nello studio medico che nelle visite a domicilio) richiede l'adozione di una tecnologia mista: un classico pc per l'accesso al sistema nello studio e un 'handheld computer' (ad esempio un palmare) per le visite a domicilio. Il palmare è visto come 'satellite' del sistema principale, che gestisce una parte delle funzioni e su un sottoinsieme del database: soltanto i pazienti da visitare a casa e, per ognuno di essi, soltanto le informazioni principali, ad esempio sulla terapia).

Esercizio 1.8

Rifletti su quale paradigma ti sembra più adatto per le seguenti applicazioni e perché:

1. Un sistema di gestione di dati bancari.
2. Un sistema per la prenotazione di esami universitari.
3. Un sistema per la descrizione di opere d'arte in un Museo.
4. Una guida ai percorsi stradali.
5. Un sistema di monitoraggio in sala parto.
6. Un sistema informativo per ordinazioni nei Ristoranti
7. Un sistema informativo per lo scambio di messaggi in un Pub

Se il sistema è realizzato con nuove tecnologie, lo sviluppo del prototipo comprende anche la progettazione e la realizzazione (più o meno 'grezza') dell'hardware (un cosiddetto *mockup*). Nelle loro diverse forme, i prototipi rappresentano uno strumento per discutere con gli utenti le modalità d'interazione con il tool da realizzare, per avere da loro un feedback e quindi per rivedere iterativamente il progetto.

Infine: la scelta dei nomi e dei simboli con i quali tradurre, nell'interfaccia, i concetti rilevati nella fase di analisi dell'utenza, dei task e degli oggetti inizia con la scelta di una cosiddetta *metafora* d'interazione (vedi Scheda 1.5).

Esempio sul Caso di Studio 1.9

Qualche esempio di metafora per il SIMB: chiamare il SIMB 'Emergency' o 'CroceRossa' ne esalterebbe gli aspetti d'intervento in situazioni difficili. Chiamarlo 'Ippocrate' evocherebbe l'idea di un sistema che si richiama ai principi originari della medicina. Chiamarlo 'ePocrates' (che, in inglese, ha una pronuncia simile) evidenzierebbe, invece, gli aspetti tecnologici del sistema. Mentre non sarebbe opportuno, ad esempio, chiamarlo 'Bovary' (come nel romanzo di Flaubert), perché evocherebbe un'idea di inefficienza.

Esercizio 1.9

Le diverse metafore possibili per il SIMB (proposte nel precedente esempio) comportano scelte diverse per i simboli da utilizzare in una interfaccia grafica. Prova a fare degli esempi e giustificali.

Esercizio 1.10

Dai un nome (attraverso la scelta di una metafora) agli esempi di sistema informativo elencati nell'esercizio 1.8

Scheda 1.3: Paradigmi d'interazione

Alla fine degli anni '80, nasce a Xerox PARK quella che sarà destinata a diventare una rivoluzione nel campo della tecnologia informatica e delle modalità d'interazione con questa tecnologia.

Mark Weiser inventa il termine di 'Ubiquitous Computing' per denotare *"a possible next generation computing environment in which each person is continually interacting with hundreds of nearby wirelessly interconnected computers"*. In questa sua idea di 'definire e costruire new computing artifacts for use in everyday life', Weisner immagina tre tipi fondamentali di oggetti, che chiama *boards, pads and tabs*. Il primo è una *'wall-sized interactive surface'* (un enorme schermo interattivo proiettato sul muro), il secondo un '*notepad*', visto non come un pc ma come qualcosa di analogo ad un foglio di carta che l'utente possa prendere ed usare in modo naturale, e infine un '*tiny computer*' simile ad un PostIt: ogni utente doveva, nella sua visione, poter usare diversi pad e diversi tab allo stesso tempo. A Xerox PARC furono costruiti diversi prototipi di questi tre tipi di oggetti.

Con gli sviluppi successivi di questa idea, il termine 'ubiquitous computing' si specificato in direzioni diverse: si parla, oggi, di *mobile o nomadic computing*, di *pervasive computing*, di *disappearing o invisible computers*, di *embedded interactive objects*: i mezzi di calcolo diventano mobili, si inseriscono ovunque (negli oggetti di uso comune), in modo così diffuso da dare l'impressione di sparire. L'interazione fra persone e tecnologia cambia di conseguenza: si parla, genericamente, di *natural interaction* o, più specificamente, di *intuitive, seamless interfaces*, di *tangible artefacts*, *haptic devices*, o *wearable computers* per immaginare mezzi di calcolo che si attivano toccandoli o indossandoli o addirittura senza accorgersene, perché dotati di 'autonomia' e di un sistema di sensori che captano ed interpretano presenza e intenzioni degli utenti.

Una ricerca su web, con i termini contenuti in questa scheda, permetterà di trovare quantità d'informazioni inimmaginabili fino a pochissimo anni fa e di percepire la velocità con cui questo settore si muove e lo sforzo d'immaginazione che la progettazione dell'interazione con questa nuova tecnologia richiede.

Scheda 1.4: Nuovi Paradigmi d'interazione (dai Call di alcuni Congressi recenti)

Ubiquitous Computing

The vision of Ubiquitous Computing bears the promise of permanent access to information and computing resources through user interfaces (UIs) accessible to every human in many different situations and contexts. Current research in intelligent and instrumented environments shows that these UIs present quite a number of challenges, such as
- new 'devices' such as tags or everywhere displays
- new UI paradigms, such as tangible, physical and hybrid UIs
- new UI metaphors for bridging the physical and virtual world
- larger and 3-dimensional space of interaction
- spatial and temporal mappings between real and virtual world
- dynamic set of devices (i.e. people moving in and out)
- dynamic adaptation among several dimensions: devices, users, services
- restrictions of technical resources in the environment
- restrictions of cognitive resources of users
- presentation planning for single users vs. groups

Another direct consequence of ubiquitous user interfaces is the fact that they are used by more than one user at a time, more specifically, this implies
- sharing of resources and control
- synchronous and asynchronous collocated collaboration
- conflict resolution such as control over parts of the environment
- supporting and monitoring social protocols for group interaction
- the need for UI metaphors for multiple simultaneous users
- public vs. private services and devices
- large UIs and specialized settings e.g. industrial control centers

Mobile computing

The ongoing migration of computing and information access from the desktop and telephone to mobile computing devices such as PDAs, tablet PCs, and next generation (G3) phones poses critical challenges for research in information access and, in particular, for Information Retrieval (IR). These devices offer limited screen size and no keyboard or mouse, making complex graphical interfaces cumbersome. In addition, these mobile devices will be used in situations involving different physical and social environments and tasks, and they will need to allow users to interact wherever he/she is and using whichever mode or combination of modes are most appropriate given the situation, their preferences and the task at hand. Furthermore, unlike traditional library or office settings, users of mobile IR devices will, typically, be subject to much higher levels of interruption and task switching - thus needing very different interface designs.

Tangible interfaces

Physical interfaces to digital information have been the subject of much research in the last decade. Many novel approaches have been introduced during this time, considered under names such as graspable interfaces, tangible interfaces, physical interfaces, and embodied interaction. By moving interaction from the virtuality of the screen into the physicality of the real world, the design space is significantly extended, enabling new and richer forms of interaction. However, at present we are likely only at the beginning of understanding the implications of establishing a direct physical link between digital information and human users.

Scheda 1.5 : Metafore d'interazione

S'intende, per metafora, un modello concettuale sviluppato in modo da essere *simile*, per qualche aspetto, ad un'entità fisica, pur avendo sue proprietà ed un suo comportamento specifici. Esempi classici: il desktop ('scrivania'), lo spreadsheet ('foglio di lavoro quadrettato') e il 'motore di ricerca' di dati su web. In quest'ultimo caso, la scelta dell'entità di riferimento nasce dall'idea di un programma che si propone di 'estrarre' informazione utile da un ampio insieme di file, sollevando l'utente da un compito gravoso per riflettere più a fondo sul problema che sta esaminando.

Per essere efficace, una metafora d'interazione deve far riferimento ad oggetti o attività di uso comune. Un'interfaccia deriva, spesso, dalla combinazione di diverse metafore. Ad esempio, una toolbar combina l'idea di tool ('strumenti') con l'idea di bar (striscia, barra); il 'portale' è una porta d'ingresso ad una raccolta di pagine o a una rete di siti su un particolare argomento; Internet è un insieme di 'isole' (i nodi) in cui si 'naviga' con un 'esploratore' (Explorer) fino a tornare a casa ('home') o un libro che si 'sfoglia' (browse) e in cui si mettono 'segnalibri' (bookmark). In Unix, i programmi si inviano dati attraverso 'tubi' (comandi di pipe).

La metafora è, inevitabilmente, poco chiara a chi non ha familiarità con il concetto. Ad esempio, 'Bluetooth' viene dal nome di un re vikingo che unificò i paesi scandinavi e quindi rappresenta l'idea di 'facilitare il dialogo fra persone'.

Scheda 1.6: Sistemi medici su palmari

2) Reference Materials

Due to the large amounts of information involved in medicine, using handheld computers as a quick reference has perhaps been the most popular application. Computers have been very good at storing large amounts of textual information. The natural extension of this is to provide electronic forms of the reference books that physicians often carry around with them.

Searchable references are available for both Windows CE and Palm, with a slight advantage to Palm due to it's popularity. The largest distributor of these references is HandMedical (http://www.handmedical.com/). Other companies include:

PDR for Palm
http://www.franklin.com/medical/pdr_palm.asp
ePocrates qRx for Palm
http://www.epocrates.com/
K2's Library for Palm and WinCE
http://www.skyscape.com/k2new/nindex.html
Handmedical's library for Palm and WinCE
http://www.handmedical.com/

3) Patient Tracking

Patient Tracking software is the most ambitious use of handhelds in medicine. Patient H&Ps, progress notes, and daily lab values essential information for physicians to generate and track. Naturally, migrating this information to a handheld computer is an obvious step.

This application was recognized early on, and Physix (http://www.physix.com/) created the first handheld tracking software (Pocketchart) for the Newton platform. PocketChart still exists and is available for WinCE, and a number of other solutions have also emerged.

Unfortunately, Patient Tracking software is one of the most difficult applications for handheld computers and requires a time commitment on the part of the physician to learn how to use the software. In more specific situations or specialities, it may be more realistic to utilize pregenerated pathways to input data, but even this would require personal customization.

Software:

Patient Tracker 3 for WinCE
http://www.handheldmed.com/code/ce/code/software/pt30announced.htm
Pocketchart for WinCE
http://www.physix.com/
Ward Watch for Palm
http://www.watch.aust.com/pilot/wardwatch/
Patient Keeper for Palm
http://www.patientkeeper.com/
Raphael for Palm
http://www.pdamed.com/