Tradotto da:
M.F. Costabile "Usability in the Software Life Cycle", in Handbook of Software Engineering and Knowledge Engineering, S.K. Chang Ed., World Scientific, Vol. 1, 2001, 179-192.

USABILITA' NEL CICLO DI VITA DEL SOFTWARE

MARIA FRANCESCA COSTABILE

Dipartimento di informatica. Università di Bari
Via Orabona 4, 70125 Bari. Italy
costabile@di.uniba.it

L'obiettivo di questo capitolo è chiarire il ruolo dei processi e dei metodi di usabilità all'interno del ciclo di vita del software. Dopo aver discusso il concetto di usabilità come fattore di qualità e dopo aver fornito varie definizioni, incluse quelle degli standard ISO, viene descritta la metodologia di progetto centrata sull'utente, per progettare sistemi usabili. Vengono quindi fornite indicazioni su come tener conto dell'usabilità nel ciclo di vita del software.

1. Introduzione

"**People are required to conform to technology. It is time to reverse this trend, time to make technology conform to people**" dice Donald A.Norman nell'articolo "Designing the future", pubblicato in Scientific American nel Settembre del 1995. Questo è particolarmente vero per la tecnologia dei calcolatori, che oggi fornisce a tutti la possibilità di utilizzare un computer e di interagire con sistemi software e esplorare risorse informative. Per molto tempo abbiamo progettato sistemi software che richiedono molto sforzo da parte degli utenti. Molti utenti di computer provano frustrazione e ansia, e questo fenomeno è anche in crescita a causa dell'enorme quantità di dati che è disponibile nei sistemi informativi su rete, come ad esempio, il Web. Di conseguenza gli attuali sistemi software devono fornire interfacce utente migliori che supportino questa intensa interazione con gli utenti.

Per un qualunque software, la componente più importante, dal punto di vista degli utenti, è l'interfaccia utente, poiché è quella che essi vedono e con la quale lavorano uando utilizzano un prodotto software. Abbiamo bisogno di creare interfacce utente di successo, in cui la qualità è valutata soprattutto dal punto di vista degli utenti. Il progetto di un interfaccia utente risulta buono, quando i progettisti comprendono bene non solo la tecnologia ma anche le persone. I progettisti devono comprendere gli utenti che utilizzeranno i loro prodotti, le loro caratteristiche, le loro capacità fisiche, i loro obiettivi, i compiti che devono svolgere e le situazioni nelle quali essi lavoreranno.

Sfortunatamente pochissima attenzione è stata dedicata all'interfaccia utente dagli sviluppatori software, con il risultato inevitabile che la maggior parte dei sistemi software è difficile da usare. L'interfaccia utente è stata considerata un dettaglio, da aggiungere al termine dello sviluppo del software. Questo non è più possibile con i sistemi interattivi, per i quali è stimato che circa il 50% del codice è dedicato all'interfaccia utente [1].

Dalla fine degli anni '70, nell'ambito dell'ingegneria del software sono stati introdotti i fattori di qualità come parte di misure della qualità del software. In [2], McCall dice che i fattori di qualità "rappresentano attributi o caratteristiche del software che un utente o un cliente del prodotto software associa alla qualità del prodotto stesso. I primi studi sui fattori di qualità furono compiuti da Boehm [3] e da McCall e al.[4]. E' interessante notare che già a quel tempo uno di questi fattori era l'usabilità, definita come lo "sforzo richiesto per imparare, operare, preparare l'input, e interpretare l'output di un programma" [2.4].

Il fatto che l'usabilità fosse già considerata un fattore di qualità, non significa che gli sviluppatori le abbiano realmente dedicato molta attenzione. Tradizionalmente essi sono abituati a giudicare il loro lavoro sulla base di criteri, che possono avere poco a che fare con le necessità e le difficoltà degli utenti finali. Sono più sensibili ad altri fattori, come l'efficienza del codice o la flessibilità dei programmi, che sono certamente degli obiettivi importanti per gli sviluppatori, ma sono poco rilevanti per la creazione di sistemi accessibili e di supporto per particolari tipi di utenti. Il bisogno di migliorare l'interfaccia utente sta ora emergendo, e i testi più recenti sull' ingegneria del software (e.g. [5]) dedicano maggiore attenzione alla progettazione dell'interfaccia utente. Gli ingegneri del software cominciano ad apprezzare l'importanza di interfacce utente di alta qualità, ma bisogna fare di più. In particolare occorre inserire le nozioni sviluppate nella disciplina Human-Computer Interaction (HCI) nella formazione dei professionisti del software. HCI è ora una disciplina consolidata, e c'è una notevole quantità di conoscenze riguardanti la progettazione di interfacce utente, che deve essere presa in considerazione per guidare i progettisti software nelle loro decisioni, in modo da evitare di ripetere progetti errati. Principalmente, occorre integrare le tecniche e i metodi di progettazione delle interfacce utente nelle metodologie standard di sviluppo del software e adottare procedure di valutazione delle interfacce e di controllo della qualità, analoghe a quelle già approvate per

valutare altri aspetti della progettazione software. Gli scopi di questo capitolo sono quelli di chiarire il ruolo dei metodi e dei processi di usabilità all’interno del ciclo di vita del software. L’organizzazione del capitolo è la seguente. La sezione 2 discute il concetto di usabilità e fornisce varie definizioni, incluse quelle degli standard ISO. La sezione 3 presenta le principali caratteristiche della metodologia di progetto centrata sull’utente il cui principale obiettivo è progettare sistemi software usabili. La sezione 4 fornisce indicazioni su come il ciclo di vita del software può essere modificato per includere l’usabilità, mentre la sezione 5 discute differenti metodi di valutazione dell’usabilità e la loro applicabilità alle varie fasi del ciclo di vita del software. Infine, la sezione 6 conclude il capitolo.

2. Definizione di usabilità

Se finora era stata dedicata scarsa attenzione all’usabilità da parte degli ingegneri del software, è ora ampiamente riconosciuto che l’usabilità è un fattore importante della qualità delle applicazioni interattive. Sono state proposte diverse definizioni di usabilità. Noi riportiamo qui quelle che consideriamo le più significative. La prima è quella di J. Nielsen che fornisce una descrizione dettagliata degli attributi dell’usabilità [8]. La seconda definizione è fornita dallo standard ISO/IEC 9126 e riflette le prospettive dell’ingegneria del software sulla qualità dei prodotti software [9]. Come sarà mostrato in seguito, da un certo punto di vista questa definizione è simile a quella data dallo standard ISO 9241[10], che è proprio lo standard delle comunità di HCI e dell’ergonomia.

Nielsen propone un modello nel quale l’usabilità è presentata come uno degli aspetti caratterizzanti l’accettabilità del sistema da parte degli utenti finali, cioè se il sistema è sufficientemente buono da soddisfare le necessità e le richieste degli utenti. Nel modello di Nielsen, l’usabilità non è una proprietà mono-dimensionale del sistema. E’ caratterizzata da cinque attributi: facilità d’apprendimento, cioè la facilità di apprendere le funzionalità e il comportamento del sistema; facilità d’uso, cioè il livello di produttività raggiungibile, dopo l’apprendimento del sistema; facilità di memorizzazione, cioè la facilità di ricordare le funzionalità del sistema, in modo che l’utente casuale può ritornare al sistema dopo un periodo di inattività, senza aver bisogno di capire nuovamente come utilizzarlo; basso livello di errori, cioè la capacità del sistema di aiutare gli utenti a non commettere errori durante l’uso, e nel caso si verificassero , dare la possibilità all’utente di risolvere facilmente; soddisfazione dell’utente che valuta quanto l’utente gradisce il sistema. Quest’ultimo attributo non deve essere sottovalutato in quanto un sistema gradevole da usare aumenta la produttività dell’utente.

Questi attributi possono essere verificati in modo oggettivo e empirico attraverso metodi di valutazione. Definire il concetto astratto di usabilità in termini di componenti più precise e misurabili è un passo molto importante verso la definizione di ingegneria dell’usabilità come disciplina, dove l’usabilità non è solo trattata, ma valutata e migliorata sistematicamente[8].

Abbiamo già detto nell’introduzione che l’usabilità è stata finora trascurata dagli ingegneri del software. Tuttavia è stata sempre menzionata nella definizione originale dello standard per la qualità del software ISO/IEC 9126. In una formulazione recente lo standard ISO/IEC 9126-1 (Information-Technology Software Product Quality) enfatizza l’importanza di progettare per la qualità, focalizzando sulle caratteristiche del sistema che possono aiutare a creare prodotti che siano efficaci, efficienti e soddisfacenti per gli utenti. La qualità di un prodotto software è data dalla sua capacità interna ed esterna di supportare il raggiungimento degli obiettivi degli utenti e delle loro organizzazioni, così da migliorare la produttività e il benessere degli utenti. Lo standard ISO/IEC 9126 descrive un modello di qualità del prodotto software, che include qualità interne, esterne e qualità d’uso. L’usabilità è definita come una di sei caratteristiche della qualità del software. Specificamente, è “la capacità di un prodotto software di essere compreso, appreso, usato e capace di attrarre l’utente, quando è usato in condizioni specificate”. L’usabilità è ulteriormente suddivisa in cinque sotto-caratteristiche: comprensibilità (understandability), la capacità intrinseca del prodotto software di mostrare agli utenti la sua adattabilità ai vari compiti che devono essere svolti nel contesto d’uso; apprendibilità (learnability), la capacità intrinseca del prodotto software di aiutare gli utenti ad apprendere facilmente le sue funzionalità; operabilità (operability), la capacità intrinseca del prodotto software di rendere possibile agli utenti l’esecuzione e il controllo della sue funzionalità; attratività (attractiveness), la capacità intrinseca del prodotto software di essere gradevole agli utenti; conformità (compliance), la capacità del prodotto software di aderire a standard, convenzioni e linee guida dell’usabilità.

Tale standard inoltre introduce il concetto di qualità d’uso, come caratteristica dell’interazione tra utente e prodotto software, che è misurabile solo nel contesto di un compito reale e osservabile, anche prendendo in considerazione differenti attributi rilevanti, come l’usabilità. La qualità in uso è definita in termini di fattori che

rappresentano il punto di vista dell'utente sulla qualità del software, quali efficacia, produttività, sicurezza e soddisfazione. Questi fattori sono collegati a quelli definiti in un altro standard, l'ISO 9241 (Ergonomic Requirements for Office Work with Visual Display Terminals). Il concetto di qualità d'uso, come descritto nell' ISO/IEC 9126-1, è simile a quello di usabilità dato nella parte 11 dell' ISO 9241 (Guida all'Usabilità), dove l'usabilità è definita come "the extent to which a product can be used by specified users achieve specified goals with *effectiveness*, *efficiency* and *satisfaction* in a specified context of use ". Per effectiveness (efficacia) s'intende accuratezza e completezza con cui utenti specifici raggiungono determinati obiettivi in specifici ambienti. Efficiency (efficienza) si riferisce alle risorse spese in relazione all'accuratezza e alla completezza degli obiettivi raggiunti (è simile al fattore di produttività che caratterizza la qualità d'uso nell'ISO/IEC 9621-I). Satisfaction (soddisfazione) è definita come comfort e accettabilità del sistema da parte degli utenti. L'usabilità è quindi intesa come un obiettivo di alto livello del progetto di un sistema. Possiamo concludere che entrambi i concetti di qualità d'uso e di usabilità, come definito nell'ISO 9241, includono gli aspetti più significativi generalmente associati all'usabilità dalle comunità HCI.

Lo standard ISO 9241 contiene linee guida per il progetto dell' interfaccia-utente, e fornisce requisiti e consigli che possono essere utilizzati durante il progetto e la valutazione di interfacce-utente[10]. Lo standard è strutturato in 17 parti. Le prime nove parti riguardano questioni legate all'hardware (requisiti per display visuali, colori, dispositivi di input) gli altri sono dedicati a questioni legate al software come ad esempio il progetto di diversi stili di dialogo tra utente e computer (principi di dialogo, stile di dialogo a menù, presentazione di informazioni, linee guida per l'utente, stili di dialogo a comandi, manipolazione diretta, form-filling).

L'usabilità è strettamente dipendente dalle particolari circostanze in cui il prodotto è usato, ad esempio i tipi di utente, i compiti che eseguono, e il contesto fisico e sociale nel quale operano. Un'attenzione particolare deve essere dedicata all'usabilità dei prodotti software utilizzati in condizioni di stress dell'utente, ad esempio sistemi di sicurezza come quelli per il controllo di traffico aereo.

La disciplina HCI sta dedicando molta importanza alla definizione di metodi che garantiscono l'usabilità dei sistemi interattivi. Attualmente anche l'industria rivolge molta attenzione all'usabilità, in quanto riconosce l'importanza dell'adottare metodi di usabilità durante lo sviluppo del prodotto per verificare l'usabilità di nuovi prodotti prima di immetterli sul mercato[11]. Alcuni studi hanno infatti dimostrato come l'uso di tali metodi fornisce un risparmio con un alto rapporto costo-beneficio[12,13].

3. Progetto centrato sull'utente vs progetto centrato sul sistema

Come discusso in [14], una delle ragioni perché molti prodotti tecnologici, inclusi sistemi informatici, varie apparecchiature elettroniche e utensili di uso quotidiano, sono così difficili da usare è che durante lo sviluppo, viene data importanza al sistema piuttosto che alle persone che saranno gli utenti finali. Probabilmente gli sviluppatori contano sul fatto che gli esseri umani sono flessibili e adattabili, essi possono meglio adattarsi alla macchina mentre il contrario è difficile. I bisogni degli esseri umani sono stati trascurati nel passato anche perché gli ingegneri sviluppavano prodotti per utenti finali che erano molto simili a loro. Con la diffusione dei calcolatori, la popolazione degli utenti è cambiata drasticamente e continua a cambiare ogni giorno. Uno dei requisiti più importanti della società della tecnologia dell'informazione è il progetto per "accesso universale", che significa che i sistemi informatici devono essere accessibili da ogni tipo di utente. Ciò che è stato fatto nel passato non funziona per gli utenti e per la tecnologia attuali. I progettisti devono permettere agli utenti di concentrarsi sui loro compiti e non sulle modalità per adempiere i compiti. Abbiamo bisogno di metodi e tecniche per aiutare i progettisti a cambiare il modo con cui progettano i prodotti, metodi che tengono presente le capacità e le necessità degli utenti.

Uno degli approcci in questa direzione è la metodologia di progetto centrato sull'utente, che si è già dimostrata un fattore chiave per lo sviluppo di interfacce di successo[14-16]. Il progetto centrato sull'utente implica un coinvolgimento degli utenti finali sin dall'inizio della pianificazione del progetto, e identificare i requisiti degli utenti diventa una fase cruciale. Il coinvolgimento degli utenti consente di prevenire errori gravi quando si progettano sistemi innovativi. In effetti, costringe i progettisti a pensare in termini di utilità e usabilità del sistema che devono sviluppare. L'approccio centrato sull'utente è utile perché migliora le funzionalità del sistema, consente di risparmiare nella manutenzione del sistema e determina una più alta soddisfazione dell'utente. Coinvolgere gli utenti nelle fasi iniziali del progetto permette di identificare come nucleo del sistema ciò che è effettivamente necessario. Specifiche di requisiti povere o inadeguate possono determinare difficoltà d'interazione e problemi di usabilità.

Anche se le valutazioni nelle fasi finali del ciclo di vita sono utili per accertare l’usabilità dei sistemi finali, non è realistico aspettarsi che questi risultati possano portare ad un riprogetto completo.

I principi base del progetto centrato sull’utente sono: 1) analizzare gli utenti e i compiti; 2) progettare e implementare il sistema iterativamente mediante prototipi di complessità crescente; 3) valutare il progetto e i prototipi con gli utenti. L’approccio centrato sull’utente richiede comprensione della scelta di realtà: chi userà il sistema, dove, come, e per fare cosa. Quindi, il sistema è sviluppato iterando un ciclo progetto-implementazione-valutazione. In questo modo è possibile evitare errori gravi e risparmiare il tempo di una nuova implementazione, poiché il primo progetto è basato su conoscenza empirica del comportamento dell’utente, dei suoi bisogni, e delle sue aspettative. Raccogliere informazioni dagli utenti non è un compito semplice, anche discutere con gli utenti non è facile, poiché gli utenti spesso trascurano aspetti che considerano erroneamente poco importanti.

Molte tecniche possono essere applicate per raccogliere le informazioni sugli utenti, tra queste osservazione diretta, interviste e questionari [13,15,16]. Osservazione diretta vuol dire osservare gli utenti mentre svolgono i loro compiti nell’ambiente di lavoro. Questo è un metodo attendibile e preciso per raccogliere dati sugli utenti, soprattutto per identificare le classi degli utenti e i loro relativi compiti. Inoltre permette di identificare fattori critici, come la pressione sociale, che possono avere un forte effetto sul comportamento dell’utente quando il sistema verrà utilizzato sul campo. Sfortunatamente, l’osservazione diretta è molto costosa perché richiede che lo sperimentatore osservi ogni utente singolarmente. Per questa ragione, è utile quando un numero ridotto di osservazioni è sufficiente per generalizzare le previsioni sul comportamento o quando le ipotesi devono essere verificate piuttosto che generate.

Le interviste raccolgono esperienze, opinioni, e motivazioni sul comportamento. Sono essenziali per individuare una conoscenza procedurale e anche problemi con strumenti usati correntemente. Le interviste costano meno delle osservazioni dirette, perché possono essere più brevi e più facili da codificare. Comunque, esse per essere efficaci, richiedono sperimentatori esperti. In contrapposizione, questionari possono essere distribuiti e raccolti da personale inesperto, consentendo così di raccogliere una notevole quantità di dati a basso costo. Essi permettono analisi statistiche e generalizzazioni più solide di quanto si possa ottenere con le interviste. I questionari permettono una panoramica sulla situazione attuale ma anche risposte specifiche.

Quale combinazione di questi metodi sia conveniente applicare dipende sia dai requisiti che dal budget. Dall’elaborazione del risultato della fase di analisi, i progettisti definiscono una prima versione del sistema. A questo punto, le tecniche di progetto (ad esempio task-centered[17] or scenario-based[16]) forniscono soluzioni soddisfacenti. L’obiettivo è esplorare diverse alternative di progetto prima di basarsi su una singola proposta da sviluppare in seguito. In questo modo i progettisti possono proporre diverse soluzioni e differenti strategie di interazione. E’ consigliato utilizzare schizzi su carta e prototipi. Gli schizzi su carta sono i più economici: parti dell’interfaccia del sistema sono disegnate su carta e l’interazione con l’utente è simulata da uno sperimentatore. Nonostante la sua apparenza futile, questa tecnica permette di raccogliere dati attendibili che possono essere utilizzati per revisioni parallele. L’uso di prototipi permette di valutare alcune funzionalità in profondità (vertical prototyping) o l’intera interfaccia (horizontal prototyping). Quindi, una o più soluzioni possono essere valutate con o senza gli utenti. Questo passo, chiamato valutazione formativa, mira a verificare alcune scelte e a raccogliere suggerimenti per revisionare il progetto.

Alla fine del ciclo di progetto, vengono effettuate valutazioni sommative che valutano il sistema finale con gli utenti reali che eseguono compiti veri nei loro ambienti di lavoro. Inoltre, una valutazione sommativa dovrebbe essere considerata come un’ultima conferma della correttezza delle ipotesi stabilite durante il processo di progettazione.

4. Il ciclo di vita del software modificato

Il processo di produzione del software è un modo di chiamare il processo seguito per la realizzazione, distribuzione e manutenzione del prodotto software. Riconoscendo che il software, come ogni altro prodotto industriale, ha un ciclo di vita che si estende dalla formazione di concetto iniziale di un sistema software fino al suo ritiro dal mercato, questo processo è generalmente chiamato ciclo di vita del software. Sono stati proposti diversi modelli del ciclo di vita del software, con lo scopo di controllare il ciclo di vita e realizzare così prodotti software di alta qualità in modo affidabile, prevedibile ed efficiente. Il nostro intento non è discutere i vari modelli ma sollevare alcuni problemi che riguardano l’usabilità dei sistemi interattivi e sono rilevanti, all’interno delle attività del ciclo di vita del software. Per tale scopo, prendiamo in considerazione il *waterfall model*, in cui ogni attività, in modo naturale, conduce a quella successiva: i requisiti sono raccolti all’inizio, e sono poi elaborati e convertiti in un

progetto generale che guida le successive fasi del progetto dettagliato dei moduli, i quali sono poi codificati, integrati e valutati. Quindi il prodotto è completato, valutato e manutenuto per il resto della sua vita. Le attività del classico modello a cascata sono i blocchi senza ombreggiatura in Fig. 1.

In questo ciclo, che è centrato sul sistema, l’usabilità non è considerata. Inoltre, ci sono alcuni ulteriori inconvenienti. Per esempio, il sistema è valutato solo alla fine del ciclo, quando sfortunatamente è troppo tardi per effettuare modifiche radicali al progetto per risolvere le possibili discrepanze con i requisiti. Un altro problema è che questi requisiti sono raccolti insieme ai clienti, che spesso sono diversi dalle persone che utilizzeranno il sistema. Chiamiamo clienti le persone che negoziano con i progettisti le caratteristiche del sistema, mentre utenti, o utenti finali, quelle persone che utilizzeranno il sistema. Ad esempio, possono essere clienti i managers di una compagnia che richiedono e acquistano il nuovo sistema, mentre gli utenti possono essere gli impiegati della compagnia, che lavoreranno con il nuovo sistema. Inoltre, i requisisti sono generalmente limitati a quelli funzionali, cioè ai servizi che il sistema deve fornire nel dominio dell’applicazione, e non prendono in considerazione le caratteristiche del sistema che riguardano più direttamente il modo in cui questi servizi devono essere forniti, come la facilità di apprendimento, la facilità d’uso, la sicurezza, etc.

Una conseguenza diretta della natura restrittiva della specifica dei requisiti è che, solitamente, la valutazione del sistema non solo è eseguita tardi nel ciclo di sviluppo, ma è anche limitata ad alcuni dei suoi aspetti funzionali, trascurando in tal modo l’usabilità del sistema. E’ da sperare che, le regole sulla salute e sulla sicurezza ora disponibili e gli standards ISO menzionati nella precedente sezione renderanno necessario certificare il sistema finale anche riguardo all’usabilità.

Al fine di creare sistemi interattivi usabili, è dunque necessario integrare il ciclo di vita standard per considerare esplicitamente l’usabilità. I rettangoli ombreggiati in Fig. 1 indicano alcune attività che devono essere svolte allo scopo di passare dal tipico progetto centrato sul sistema del classico modello a cascata a quello centrato sull’utente che conduce a progettare sistemi più usabili. Il rettangolo ombreggiato denotato con 1 indica che è obbligatorio integrare la fase di specifica dei requisiti con un’accurata analisi degli utenti, cioè delle persone che utilizzeranno il sistema, dei lavori che svolgono e dell’ambiente nel quale lavorano.

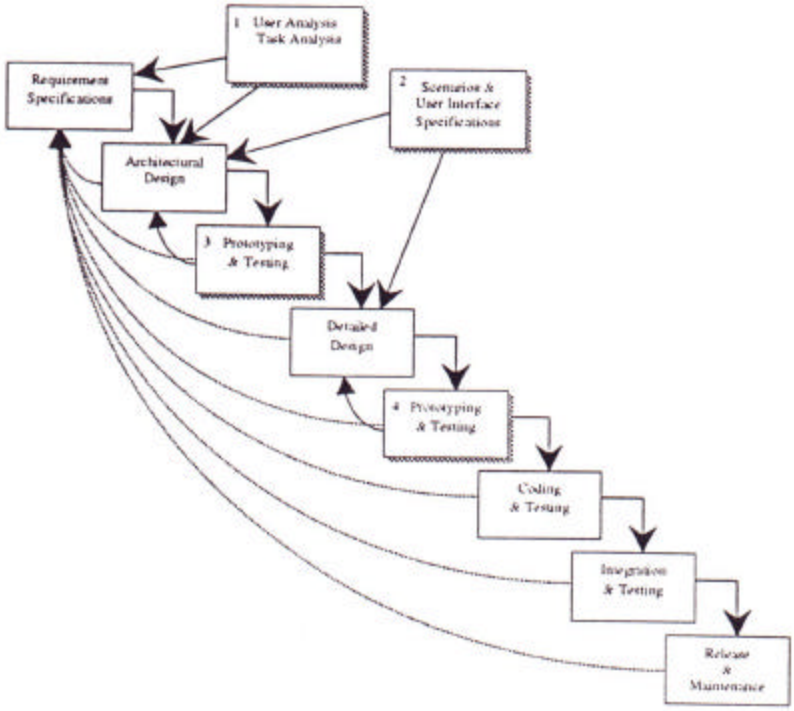

Fig. 1. The revised waterfall life cycle. White boxes refer to the classical model, while gray boxes are added to include usability.

Il miglior modo per raccogliere queste informazioni è visitare gli utenti sul loro posto di lavoro, osservare il modo in cui effettuano i loro compiti, e intervistarli [13]. L’analisi dell’utente può influenzare l’intero progetto del sistema. Inoltre, le classiche fasi del progetto architetturale e del progetto dettagliato dovrebbero esplicitamente includere il progetto dell’interfaccia utente, che non è più posticipato alla fine dello sviluppo del sistema. Infatti l’interfaccia utente è la parte più rilevante del sistema dal punto di vista degli utenti, e il suo progetto dovrebbe essere discusso dai progettisti direttamente con gli utenti sin dall’inizio del ciclo di vita. Come indicato nel rettangolo ombreggiato 2, gli scenari d’uso [16], che sono una sequenza di passi che descrivono un’interazione tra un utente e il sistema, possono aiutare a concepire il progetto di un’ interfaccia usabile. Come discusso nella sezione precedente, la metodologia di progetto centrata sull’utente accentua l’iterazione del ciclo progettazione-implementazione-valutazione. Abbiamo inserito, dunque, i rettangoli ombreggiati 3 e 4 nel modello a cascata allo scopo di mettere in risalto che entrambi i progetti, quello architetturale e quello dettagliato dei moduli software, devono essere realizzati attraverso lo sviluppo dei prototipi che sono valutati con gli utenti per controllare se verificano i requisiti attesi. Se non è così, viene sviluppato un nuovo prototipo e quindi valutato nuovamente, e questo processo iterativo termina quando i requisiti specificati sono verificati: solo a questo punto possiamo procedere con lo sviluppo di un prototipo più avanzato, che include nuove funzionalità. Inoltre, l’attuale approccio iterativo implica che da ogni fase del ciclo di vita del software è possibile ritornare ad una qualsiasi delle fasi precedenti, in particolare tornare indietro e aggiornare le specifiche dei requisiti, come indicato dalle frecce tratteggiate in Fig. 1. I prototipi su carta possono essere usati come primi prototipi , poiché sono facili da sviluppare e consentono di risparmiare tempo dando però la possibilità di controllare le idee

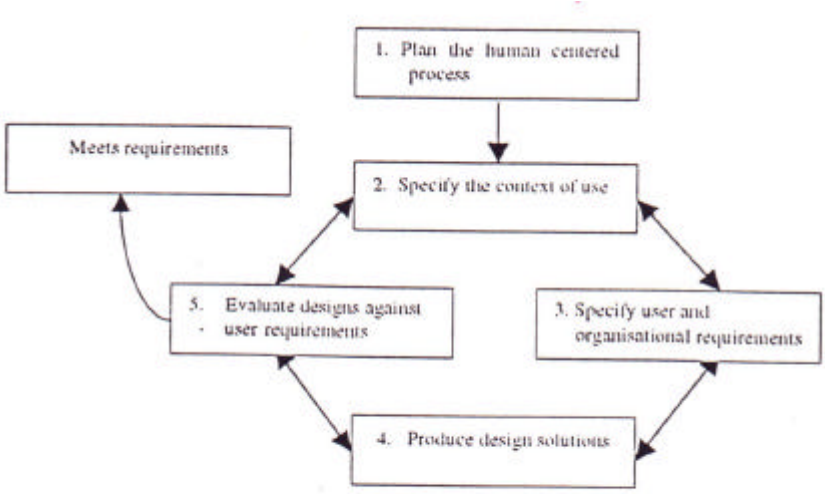

Fig. 2. The ISO 13407 Human-centered design process for interactive systems.

importanti con gli utenti. I prototipi sviluppati con linguaggi come Visual Basic™ sono la fase successiva ai prototipi su carta, e sono usati da molti progettisti.

I principi chiave della metodologia di progetto centrato sull’utente, cioè focalizzare sugli utenti, sul compito che essi eseguono e sul contesto nel quale lavorano, e lo sviluppo iterativo attraverso prototipi di complessità crescente che sono valutati con gli utenti, sono stati inglobati nello standard ISO 13407 (Human-centered design process for interactive systems) che è mostrato in Fig. 2 [19]. Le soluzioni di progetto menzionate nel blocco 4 di Fig. 2 sono implementate attraverso prototipi che sono valutati, e se non verificano i requisiti specificati, il processo è iterato attraverso una revisione delle specifiche e la proposta di un nuovo prototipo. Il processo iterativo termina quando il progetto verifica i requisiti.

Dalla discussione precedente, segue che la valutazione rappresenta la fase centrale nel ciclo di sviluppo. Per questa ragione, nell’ambito della comunità dell’ HCI, Hartson e Hix hanno sviluppato il modello del ciclo di vita a stella mostrato in Fig. 3 [20]. Mentre il modello a cascata propone un approccio top-down (analitico), il modello a stella riconosce che questo approccio ha bisogno di essere completato da un approccio bottom-up (sintetico), e può

Tradotto da:
M.F. Costabile “Usa vledge
Engineering, S.K. C

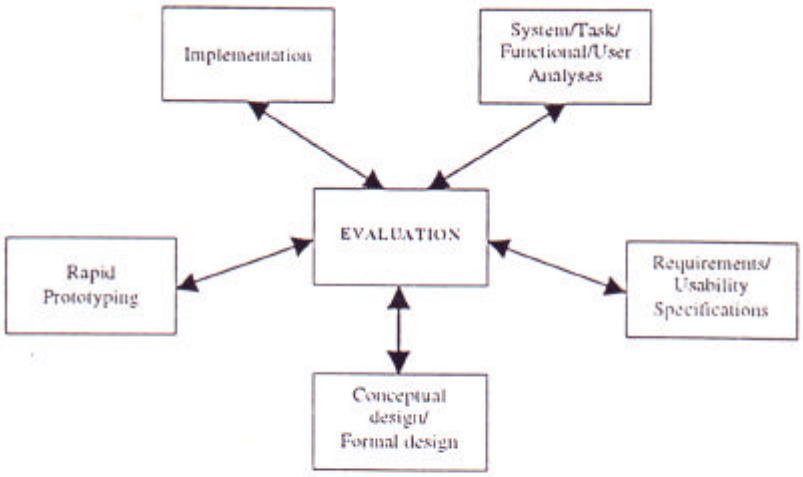

Fig. 3. The star life cycle model [20].

cominciare da qualsiasi punto della stella e proseguire da qualsiasi altro stadio (come mostrato dalle doppie frecce). In questo modo, i requisiti, il progetto e il prodotto evolvono gradualmente, divenendo passo dopo passo ben definiti.

5. Valutazione di usabilità nel ciclo di vita del software

Allo scopo di progettare sistemi usabili, abbiamo visto che nel ciclo di vita del software la valutazione dell’usabilità gioca un ruolo fondamentale. La ricerca in HCI ha fornito diversi principi e linee-guida che possono aiutare i progettisti nel prendere decisioni. Comunque, applicare linee-guida di progettazione disponibili è un buon inizio, ma non ci sono “ricette” la cui applicazione garantisce il risultato finale, e quindi non ci sono sostituti per la valutazione del sistema. Possono essere usati metodi diversi per la valutazione dei sistemi nelle diverse fasi del loro sviluppo: i più adottati sono i metodi che coinvolgono gli utenti e i metodi d’ispezione.

I metodi che coinvolgono gli utenti si basano fondamentalmente su user testing, cioè vengono valutate le proprietà dell’usabilità osservando come il sistema, o un prototipo del sistema, è utilizzato da utenti reali, o loro rappresentanti che svolgono compiti veri [15, 16, 21]. I metodi d’ispezione di usabilità coinvolgono solo i valutatori esperti, che ispezionano l’interfaccia utente con lo scopo di trovare possibili problemi di usabilità, fornendo giudizi basati sulla loro conoscenza, e facendo raccomandazioni per risolvere i problemi e migliorare l’usabilità dell’applicazione[22]. La valutazione con gli utenti fornisce una valutazione credibile, perché valuta l’usabilità attraverso campioni di utenti reali. Comunque, essa ha un numero di inconvenienti, come la difficoltà di selezionare adeguatamente un campione valido della comunità utente, ed istruirlo appropriatamente per usare il sistema.

Rispetto alla valutazione con gli utenti, i metodi di ispezione di usabilità sono più soggettivi, avendo una forte dipendenza dagli ispettori ‘skills’. Tra i metodi di ispezione, possiamo includere la valutazione euristica, cognitive walkthrough, l’ispezione formale di usabilità, le ispezioni di linee guida.[22]. La valutazione euristica è il metodo più informale; coinvolge l’esperto di usabilità che analizza gli elementi di dialogo dell’interfaccia utente per valutare se sono conformi ai principi di usabilità, usualmente detti euristiche, da qui il nome del metodo. Nel cognitive walkthrough, l’esperto usa alcune procedure dettagliate per simulare i processi di risoluzione di problemi degli utenti durante il dialogo utente-calcolatore, con l’obbiettivo di vedere se le funzionalità fornite dal sistema sono efficienti per gli utenti e li conducono verso azioni corrette. L’ispezione dell’usabilità formale è stata progettata per aiutare gli ingegneri a esaminare il prodotto e a trovare i difetti di usabilità. E’ molto simile ai metodi di ispezione del codice, con i quali gli sviluppatori del software hanno familiarità ed è eseguita dagli ingegneri che progettano il prodotto e da un gruppo di pari (team of peer) che cercano i difetti. Infine, nelle ispezioni di linee guida, l’esperto ispeziona l’interfaccia per valutare se è conforme alla lista di linee guida dell’usabilità che ha disponibile. Il metodo può essere considerato come un incrocio tra la valutazione euristica e l’ispezione di standard, la seconda è un altro tipo di ispezione per valutare la compatibilità dell’interfaccia rispetto ad alcuni standard per interfacce. Una descrizione dettagliata di queste e altri metodi di ispezione è in [22].

Il vantaggio principale dei metodi di ispezione è comunque il risparmio di costo: essi non coinvolgono utenti né richiedono attrezzatura speciale o disponibilità di laboratori[8,22]. Inoltre, gli esperti possono scoprire un’ ampia

gamma di problemi e possibili errori in un sistema complesso in un intervallo di tempo limitato. Per queste ragioni, i metodi di ispezione sono stati ampiamente utilizzati negli ultimi anni, specialmente negli ambienti industriali[23], dato che l'industria è molto interessata a metodi efficaci che possono fornire buoni risultati pur essendo poco costosi.

I metodi di ispezione hanno lo scopo di trovare i problemi di usabilità in una interfaccia-utente, e fare raccomandazioni per risolvere questi problemi. Quindi, possono essere applicati a vari passi dello sviluppo del software, e sono certamente usati per valutare prototipi del sistema, anche prototipi cartacei, in modo che i difetti possono essere corretti subito. Quando l'implementazione del sistema è disponibile, la valutazione con gli utenti è spesso raccomandata. Essa include i metodi sperimentali, i metodi basati sull'osservazione, e le tecniche di survey. Tra i metodi sperimentali, gli esperimenti controllati sono molto validi, essi forniscono una prova empirica per sostenere le specifiche ipotesi. Permettono una valutazione comparativa, molto utile quando sono disponibili prototipi alternativi o più versioni di uno stesso sistema. Un esperimento consiste dei passi seguenti: formulazione delle ipotesi da testare, definizione delle condizioni dell'esperimento che differiscono solo nel valore di alcune variabili controllate, esecuzione dell'esperimento, analisi dei dati raccolti.

Al fine di verificare l'usabilità di un singolo prototipo, possiamo osservare gli utenti che lavorano con esso. Una tecnica valida è il Thinking aloud, nel quale agli utenti è richiesto di pensare a voce alta quando usano il sistema o il prototipo. In questo modo, i valutatori possono scoprire le errate interpretazioni degli utenti e gli elementi del sistema che li causano. Entrambi i metodi, quello sperimentale e quello basato sull'osservazione, sono usati per raccogliere informazioni sulla "performance" degli utenti e del sistema: essi non forniscono informazioni circa la soddisfazione degli utenti, la quale è una misura soggettiva che può essere ottenuta con le tecniche di survey, come le interviste e i questionari i quali sono stati menzionati nella sezione 3 poiché usati anche per raccogliere i requisiti degli utenti. Per maggiori dettagli il lettore è rimandato a [15,16].

La valutazione euristica gioca un ruolo importante, dato l'interesse dell'industria per metodi economici ma efficaci. Essa prescrive di richiedere l'analisi del sistema da parte di un piccolo numero di esperti i quali valutano l'interfaccia, utilizzando una lista di principi di usabilità riconosciuti, le euristiche. Alcune ricerche hanno mostrato che la valutazione euristica è una tecnica di valutazione dell'usabilità molto efficiente, con un alto rapporto costo-benefici, e per questo è riferita come uno dei così detti metodi di usabilità economici. In principio un solo valutatore può effettuare la valutazione euristica. Comunque in un'analisi di sei studi, è stato stimato che un solo valutatore è in grado di trovare solo il 35% del numero totale dei problemi di usabilità esistenti [8,22]. Differenti valutatori tendono a trovare problemi diversi. Dunque, se più esperti sono coinvolti nella valutazione, è possibile trovare più problemi. Il modello matematico definito in [12] mostra che risultati ragionevoli possono essere ottenuti avendo solo cinque valutatori.

La valutazione euristica, e in genere i metodi di ispezione proposti finora non sono sistematici, e mancano di una struttura metodologica. In alcuni articoli recenti, è stato proposto l'User Action Framework (UAF) come un tentativo di organizzare meglio l'ispezione di usabilità, le linee-guida per il progetto, la classificazione e la raccolta di problemi di usabilità [25]. UAF fornisce una base di conoscenza, in cui differenti problemi di usabilità sono organizzati secondo il modello del ciclo di interazione – un'estensione della teoria delle azioni di Norman [26]. Seguendo tale modello i problemi sono organizzati tenendo conto di come l'interazione dell'utente è influenzata dal progetto del sistema nei vari punti dove gli utenti devono compiere azioni cognitive o fisiche. La base di conoscenza è un contributo notevole della metodologia UAF, che cerca di fornire una soluzione al bisogno di metodi di ispezione più specifici per vari tipi di applicazioni, e di rapporti che riportino i risultati dell'ispezione in modo più leggibile e adatto al confronto.

Classificare e organizzare i problemi in base di conoscenza, è anche un modo per tenere traccia di un gran numero di problemi trovati in differenti studi di usabilità, capitalizzando in questo modo sulle esperienze passate. Riutilizzare le precedenti esperienze di valutazione, e renderle disponibili alle persone meno esperte è anche uno degli obiettivi di una nuova tecnica di ispezione descritta in [27.28], che introduce l'uso di patterns di valutazione per guidare l'attività all'ispettore. I patterns descrivono precisamente quali oggetti cercare e quali azioni eseguire allo scopo di analizzare tali oggetti. Tali patterns di valutazione sono chiamati Abstract Tasks per le seguenti ragioni: Task perché ognuno descrive un compito che il valutatore deve eseguire durante l'ispezione allo scopo di scoprire i problemi di usabilità; Abstract, perché descrive le operazioni per portare a termine i compiti, senza alcun riferimento ad una specifica applicazione, fornendo invece una descrizione che è indipendente da qualsiasi applicazione. La loro formulazione riflette l'esperienza di valutatori qualificati. Gli Abstract Tasks permettono, così, anche ai valutatori meno esperti di ottenere buoni risultati. La maggiore efficacia ed efficienza dell'ispezione basata sull'uso del

Tradotto da:
M.F. Costabile “Usability in the Software Life Cycle”, in Handbook of Software Engineering and Knowledge Engineering, S.K. Chang Ed., World Scientific, Vol. 1, 2001, 179-192.

Abstract Tasks, rispetto alla tradizionale valutazione euristica, è stata verificata attraverso uno studio comparativo descritto in [29].

6. Conclusioni

Nel capitolo abbiamo enfatizzato l’importanza di una buona interfaccia utente per i sistemi interattivi odierni. La qualità dell’interfaccia deve essere valutata specialmente dal punto di vista degli utenti, prendendo in considerazione gli utenti che realmente useranno il sistema. L’usabilità, una qualità che è stata trascurata dalla comunità di sviluppatori software per molto tempo, diventa un fattore cruciale per giudicare la qualità dell’interfaccia utente. E’ stata discussa la metodologia di progetto centrata sull’utente per progettare sistemi usabili, e sono state fornite indicazioni su come includere l’usabilità nel ciclo di vita del software.

References

1. B. Myers and M.B. Rosson, Survey on User Interface Programming, *Proc. CHI’92, ACM Press*, 195-202.
2. J.A. McCall, Quality factors. In *Encyclopedia of Software Engineering* (John J. Marciniak, Ed), pp. 958-969, John Wiley , New York, 1994.
3. B. Boehm, *Characteristics of Software Quality*, North Holland Publishing Co., New York, 1978.
4. J. McCall, P. Richards, and G. Walters, *Factors in Software Quality*, 3 vol. NTIS AD-A049-015, 015, 055, Nov. 1977.
5. R.S. Pressman, *Software Engineering: A Practitioner’ s Approach*. Fourth Edition, McGraw-Hill Companies, New York, 1997.
6. D.J. Mayhew, *Principles and Guidelines in Software User Interface Design*, Prentice Hall, Englewood Cliffs, 1992.
7. D.J. Mayhew, *The Usability Engineering Lifecycle: a Practitioner’s Handbook for User Interface Design*, Morgan Kaufmann Publishers, California, 1999.
8. J. Nielsen. *Usability Engineering*, Academic Press, Cambridge, MA, 1993.
9. ISO/IEC (International Organization for Standardization and International Electrotechnical Commision), ISO/IEC 9126-1: Information Technology – Software Product Quality, 1998.
10. ISO (International Organization for Standardization), ISO 9241: Ergonomics Requirements for Office Work with Visual Display Terminal (VDT) - Parts 1-17, 1997.
11. K.H. Madsen, Special Issue on “The Diversity of Usability”, *Communication of ACM*, **Vol. 42, No. 5**, 1999.
12. J. Nielsen and T.K. Landauer, A Mathematical Model of The Finding of Usability Problems, *Proc. ACM INTERCHI’93 – International Conference on Human Factors in Computing Systems, ACM Press*, Amsterdam, NL, 296-213, 1993.
13. J. Hackos and J.C. Redish, *User and Task Analysis for Interface Design*, John Wiley & Sons, 1998.
14. J. Rubin, *Handbook of Usability Testing*, John Wiley & Sons, 1994.
15. Dix, J. Finlay, G. Abowd and R. Beale, *Human Computer Interaction*, Prentice Hall, 1998.
16. J. Preece, Y. Rogers, H. Sharp, D. Beyon, S. Holland and T. Carey, *Human-Computer Interaction*, Addison Wesley, 1994.
17. F. Paterno’, Task Models for Interactive Software Systems, In this volume.
18. C. Ghezzi, M. Jazayeri and D. Mandrioli, *Fundamentals of Software Engineering*, Prentice Hall International, 1991.
19. ISO (International Organization for Standardization), ISO 13407: Human-Centered Design Process for Interactive Systems, 1998.
20. H.R. Hartson and D. Hix, *Developing User Interfaces*, John Wiley, New York, 1993.
21. J. Whiteside, J. Bennet and K. Holtzblatt, Usability Enginnering Our Experience and Evolution, In *Handbook of Human-Computer Interaction*, (ed. M. Helander), Elsevier Science, 791-817, 1988.
22. J. Nielsen and R.L. Mack, *Usability Inspection Methods*, John Wiley & Sons, New York, 1994.
23. J. Nielsen, Guerrilla HCI: Using Discount Usability Engineering to Penetrate Intimidation Barrier, In *Cost-Justifying, Usability* (R.G. Bias, D.J. Mayhew, Eds) Academic Press, Also available at the URL *http://www.useit.com/papers/guerilla_hci.html*, 1994.
24. R. Jeffries, H.W. Desurvire, Usability Testing vs. Heuristic Evaluation: Was There a Context? *ACM SIGCHI Bull.*, **Vol. 24 N. 4**, 39-41, 1992.
25. H.R. Hartson, T.S. Andre, R.C. Willges and L.Van Rens, The User Action Framework: A Theory-based Foundation For Inspection and Classification of Usability Problems, *Proc. HCII ’99, Munich, Germany*, August 1999, vol. I, 1058-1062.
26. D.A. Norman, *The Psychology of Everyday Things*, Basic Books, 1988.
27. M. Matera, SUE: A Systematic Methodology for Evaluating Hypermedia Usability, Ph.D. Thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 2000.
28. M.F. Costabile and M. Matera, Proposing Guidelines for Usability Inspection, *Proc. TFWWG’2000, Biarritz, France*, October 7-8, 2000, Spinger Verlag, 283-292.
29. De Angeli, M. Matera, M.F. Costabile, F. Garzotto, and P. Paolini, Evaluating the SUE Inspection Technique, *Proc. AVI’2000 – International Conference on Advanced Visual Interfaces, Palermo, Italy, May 24-27, 2000, ACM Press*, 143-150.