VALUTAZIONE IN CONTESTI CRITICI RISPETTO ALLA SICUREZZA

F. Paternò & C. Santoro
Istituto CNUCE-C.N.R.- Pisa

Le applicazioni interattive critiche rispetto alla sicurezza hanno specifici requisiti che non possono essere adeguatamente catturati dalle tecniche di valutazione tradizionali. In questo articolo viene discusso come sia possibile eseguire una sistematica analisi basata sull'ispezione per migliorare gli aspetti di usabilità e di sicurezza di un'applicazione analizzando il prototipo di un sistema e il relativo modello dei task. L'obiettivo è di valutare cosa succede quando si verificano delle interazioni e dei comportamenti diversi da ciò che era stato previsto durante la progettazione del sistema. Viene inoltre discussa un'applicazione di questo metodo ad un reale case study.

INTRODUZIONE

L'area di ricerca della progettazione basata sui modelli e della valutazione di applicazioni interattive (Paternò, 1999) ha lo scopo di identificare dei modelli che siano in grado di supportare il progetto, lo sviluppo e la valutazione di applicazioni interattive. In particolare, i modelli dei task descrivono le attività che devono essere eseguite in maniera tale che gli obiettivi dell'utente siano raggiunti, dove con "obiettivo" dell'utente intendiamo una desiderata modifica dello stato di una applicazione.

Esistono diversi approcci che hanno lo scopo di specificare i task. In questo articolo noi considereremo modelli di task rappresentati usando la notazione ConcurTaskTrees (Paternò 1999). In ConcurTaskTrees le attività sono descritte su differenti livelli di astrazione in maniera gerarchica e rappresentate graficamente con un formato ad albero (vd. Figura 1 per un esempio), sfruttando un ricco insieme di operatori per descrivere le diverse relazioni temporali (concorrenza, interruzione, disabilitazione, iterazione, opzione e così via).

I modelli dei task possono anche essere utili nel supportare la progettazione e la valutazione di applicazione interattive critiche rispetto alla sicurezza. La caratteristica fondamentale di tali applicazioni è quella di controllare un'entità del mondo reale e di soddisfare un certo numero di requisiti, evitando allo stesso tempo che l'entità controllata raggiunga degli stati di hazard. Esistono molti esempi di sistemi critici rispetto alla sicurezza nella vita reale (il controllo del traffico aereo, il sistema ferroviario, i sistemi di controllo a livello industriale, ...). In questo campo sorgono delle specifiche problematiche: per esempio, le azioni dell'utente non possono essere annullate, quindi il problema degli errori degli utenti e di come appropriatamente progettare l'interfaccia utente assume un'importanza cruciale.

Lo scopo di questo articolo è di discutere come i modelli dei task possono essere usati in una valutazione dell'usabilità basata sull'ispezione per le applicazioni interattive critiche rispetto alla sicurezza. A questo proposito innanzitutto verrà introdotto il nostro approccio, quindi discuteremo un esercizio nell'applicarlo ad un caso di studio reale nell'area del controllo del traffico aereo e infine discuteremo la nostra esperienza in termini di risultati e di lezioni apprese.

MODELLO DEI TASK E VALUTAZIONE DELL'USABILITÀ'

Il lavoro di ricerca svolto finora nell'ambito della valutazione basata sui modelli si è focalizzato principalmente nel supportare la valutazione della performance (come ad esempio gli approcci di tipo GOMS) oppure nell'uso dei modelli per supportare l'analisi automatica delle interazioni dell'utente. (Ivory et al. 1999).

Sono stati proposti numerosi metodi di valutazione basati sull'ispezione, i più rilevanti sono la valutazione euristica (Nielsen 1993), e il cognitive walkthrough (Wharton et al. 1994). Se da una parte questi metodi possono essere applicati con uno sforzo limitato, essi però prestano poca attenzione a cosa succede se gli utenti commettono errori e alle conseguenze di questi errori.

Pochi lavori si sono occupati di questo tipo di problema. Reason (Reason 1990) ha introdotto una prima sistematica analisi riguardante l'errore umano, includendo la semplice distinzione tra *slip* e *mistakes.* Il filone della Human Reliability Analysis (Hollnagel 1993) ha tentato di quantificare gli effetti dell'errore umano sui rischi associati ad un certo sistema, Leveson (Leveson

1995) ha introdotto alcune linee-guida per progettare interazioni dell'utente che siano sicure, THEA (Fields et al. 1997) ha usato un approccio basato sugli scenari per analizzare i possibili problemi. In (Galliers et al. 1999) viene proposta un'analisi che supporta la ri-progettazione di una interfaccia utente per evitare il verificarsi di errori o ridurre il loro effetto, mentre in (Johnson 1999) vengono presentate una serie di tecniche per analizzare i report degli incidenti.

Il contributo del nostro metodo risiede nell'aiuto che esso fornisce ai progettisti al fine di analizzare sistematicamente cosa succede se si verificano deviazioni nell'esecuzione dei task rispetto a ciò che era stato originariamente previsto.

IL METODO

Nella nostra analisi noi consideriamo il modello dei task del sistema: come il progetto del sistema che deve essere valutato assume che i task debbano essere eseguiti. Lo scopo è quello di identificare le possibli deviazioni da tale progetto. Un insieme predefinito di classi di deviazioni è identificato dalle cosiddette "guidewords", che sono parole chiave o frasi che si riferiscono ad uno specifico tipo di comportamento anomalo del sistema. Interpretare le guidewords in relazione ad un task consente all'analista di generare sistematicamente modi in cui il task può potenzialmente deviare dal comprtamento aspettato, analizzando l'impatto che esso ha sul sistema corrente e generando suggerimenti su come migliorare il progetto corrente. Il metodo si compone di tre passi fondamentali:

- Sviluppo del modello dei task dell'applicazione considerata al fine di identificare come il progetto del sistema prevede che i task debbano essere eseguiti;
- Analisi delle deviazioni associate a task di base; i task di base sono le foglie nel modello dei task gerarchico, ossi i task che il progettista ha ritenuto opportuno considerare come singole unità;
- Analisi delle deviazioni associate a task ad alto livello: questi task consentono al progettista di identificare gruppi di task e conseguentemente di analizzare deviazioni che coinvolgono più di un task di base.

E' importante che l'analisi delle deviazioni sia eseguita da gruppi interdisciplinari dove tali deviazioni vengono considerate ed analizzate da diverse prospettive e punti di vista. L'analisi segue un approccio bottom-up (prima vengono considerati i task di base e successivamente i task ad alto livello), consentendo ai progettisti di focalizzarsi inizialmente sugli aspetti concreti dell'interfaccia e successivamente di ampliare l'analisi considerando degli aspetti più logici. Noi abbiamo analizzato le deviazioni associate con le seguenti guidewords:

- *None*, l'unità di analisi non è stata eseguita, oppure è stata eseguita ma senza produrre alcun risulatto. Viene decomposta in tre tipi differenti di deviazione: mancanza di input (*No Input*), task non eseguito (*No performance*), mancanza di risultati (*No output*).
- *Other than*, i task considerati sono stati eseguiti in maniera differente rispetto a come specificato nel modello dei task. Tre casi sono stati distinti: Less, More e Different e ognuno di essi può essere ulteriormente suddiviso nell'analisi dell'input, della performance del task e del risultato del task.
- *Ill-timed*, i task considerati sono stati eseguiti al momento sbagliato: noi distinguiamo quando l'esecuzione del task occorre in maniera anticipata o ritardata rispetto a ciò che era stato previsto.

Inoltre, per ogni task analizzato è possibile memorizzare in una tabella le seguenti informazioni:

- *Task*: il task correntemente analizzato;
- *Guideword*: il tipo di deviazione considerato;
- *Spiegazione*: come la deviazione è stata interpretata per questo task o gruppo di task;
- *Cause*: le potenziali cause (anche intese in termini di errori dal punto di vista cognitivo) che hanno generato la deviazione;
- *Conseguenze*: i possibili effetti che la deviazione produce sul sistema;
- *Protezioni*: ossia in che modo il design corrente fa fronte al verificarsi (o all'effetto) della deviazione;
- *Raccomandazioni*: suggerimenti per un progetto migliorato che riesca a gestire meglio la deviazione;

Abbiamo anche ritenuto utile classificare la spiegazione in termini di quale fase del ciclo di interazione (secondo il modello di Norman basato su intenzione, azione, esecuzione, percezione interpretazione e valutazione) potrebbe aver generato il problema.

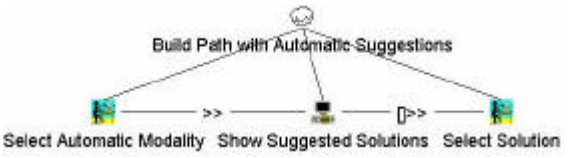

Figura 1. Un esempio di task ad alto livello.

Consideriamo ad esempio il task *Check deviation* (ossia l'utente controlla se un certo aereo sta seguendo il cammino che gli e' stato assegnato all'interno dell'aereoporto) e la classe di deviazione 'None'. Il caso 'No Input' (ossia la mancanza di informazione sullo stato corrente dell'aereo) può avere molteplici cause (ad esempio il controllore è distrratto, oppure c'è stato un fault di sistema) ma la stessa conseguenza: il controllore non ha una visione aggiornata del traffico aereo. Il caso 'No Performance' si ha per esempio quando i controllori non interpretano correttamente l'informazione, mentre nel caso 'No output' i controllori trovano una deviazione ma se ne dimenticano immediatamente in quanto interrotti da un'altra attività. Per i task ad alto livello l'interpretazione di ogni guideword deve essere appropriatamente adattata in funzione delle particolari relazioni temporali esistenti tra i sottotask a più basso livello. Consideriamo ad esempio il task *Build path with automatic suggestions* in Figura 1: a causa della relazione temporale di sequenzialità che esiste tra il primo sottotask e gli altri task fratelli, una mancanza di input per il primo task di base equivale ad una mancanza di input per l'intero task padre. Quindi, il caso 'None/No input' relativo al task padre viene riportato al corrispondente caso del primo figlio sinistro. Il caso 'None/No performance' significa che nessun task a basso livello (*Select Automatic Modality*, *Show suggested solutions* and *Select solution*) è stato eseguito. Il caso 'None/No output' si ha invece quando tutti i sottotask sono stati eseguiti ma nessun risultato è stato prodotto alla fine. Riferendoci al task *Build path with automatic suggestions* in Figura 1, questo caso può essere riportato al caso 'None/No output' del task *Select solution*. Per quanto riguarda le altre guidewords il ragionamento deve essere adattato di conseguenza. La guideword 'Less' applicata al task *Build path with automatic suggestions* sta a significare che uno dei sottotask associati non è stato eseguito.

Figure 2. L'interfaccia utente del prototipo valutato.

Il caso 'More' può scaturire da una esecuzione di un sottotask addizionale e non prevista (ad esempio un task è eseguito più di una volta) Il caso 'Different' si ha quando una differente relazione temporale è stata introdotta tra i suoi sottotask (ad esempio i sottotask sono eseguiti in maniera concorrente invece di esser eseguiti sequenzialmente), oppure vengono eseguiti dei sottotask errati. L'analisi delle guidewords 'Early' e 'Late' dovrebbe considerare il task padre nella sua globalità, analizzando l'impatto che hanno sul sistema tutte le situaizoni in cui una performance anticipata o ritardata del task padre potrebbe verificarsi.

ORGANIZZAZIONE DELLA VALUTAZIONE

Le sessioni di valutazione dovrebbero essere attentamente organizzate, coinvolgendo, oltre ai valutatori, anche alcuni sviluppatori di software e gli utenti finali. I partecipanti dovrebbero essere introdotti al metodo, e dovrebbero dedicare del tempo alla comprensione delle caratteristiche fondamentali del prototipo che (indipendentemente dal suo livello di raffinamento) dovrebbe essere disponibile durante tutta la durata dell'esercizio di valutazione. I valutatori dovrebbero guidare la discussione ponendo delle domande che scaturiscono sia dall'analisi del modello dei task che dalla lista delle deviazioni, spiegando anche quali sono gli elementi di presentazione dell'interfaccia utente che supportano il task logico considerato. Non è necessario considerare tutti i possibili casi nelle tabelle, solitamente la descrizione dei casi più importanti è sufficiente per avere un'analisi esauriente.

LA NOSTRA ESPERIENZA

Abbiamo applicato il metodo ad un prototipo per il controllo del traffico aereo in un aerodromo. Una delle funzionalità principali di tale prototipo è di supportare le comunicazioni data-link per gestire i movimenti degli aerei, dove il datalink è una tecnologia che permette uno scambio asincrono di dati codificati secondo una sintassi predefinita.

All'interno dell'aeroporto ci sono due controllori principali: il controllore di terra (o "ground"), che gestisce gli aerei tra le piste e i gates e il controllore di torre (o "tower") che gestisce i decolli e gli atterraggi degli aerei. I controllori scambiano messaggi con i piloti interagendo con interfacce utente grafiche (Figura 2) che mostrano in tempo reale il traffico all'interno dell'aereoporto e usando le cosiddette "enriched flight labels", che mostrano sullo schermo in maniera permanente solo le informazioni fondamentali ('standard' mode) mentre informazione addizionale viene presentata quando tali labels vengono selezionate ed allargate ('selected' mode).

Noi abbiamo svolto l'esercizio con un team di persone in cui sono stati coinvolti sia sviluppatori di software che utenti finali (controllori del traffico aereo in questo caso) che esperti in progettazione dell'interfaccia utente. Durante queste sessioni sono emersi una serie di aspetti interessanti. Un esempio è collegato all'attività del controllore di verificare se lo stato corrente (ad esempio la posizione) di un aereo si conforma a quello previsto. Se questa attività non viene eseguita, le protezioni esistenti nel sistema corrente si limitano solo al caso di una incursione nella pista (runway incursion), evidenziando una linea rossa che connette gli aerei coinvolti e che lampeggia sullo schermo del controllore. Tuttavia, un warning sonoro in questo caso sarebbe più adatto, specialmente quando è necessario comunicare in tempo informazione riguardante l'urgenza di una certa situazione (ad esempio calibrando i vari attributi sonori).

All'interno del prototipo era prevista inoltre una funzionalità che permetteva al controllore di poter effettuare uno zoom (dentro e fuori) sulla vista dell'aerodromo. Tuttavia, era possibile che un conflitto si verificasse nella parte dell'aerodromo correntemente fuori della vista del controllore (caso 'No Input' del task *Check Deviation*). Il fatto di avere un bottone che permettesse di ritornare alla vista di default non era ritenuto sufficiente per permettere al controllore di accorgersi in tempo di eventuali situazioni di percolo. Quindi il sugggerimento è stato di avere visualizzata in maniera permanenete la visione globale dell'aerodromo e inoltre la possiblità di attivare su richiesta una finestra separata per ottenere una visione dettagliata di una parte specifica dell'aerodromo.

Un'altra deviazione considerata era associata con il task *Send clearance*, caso 'No Performance'. Le cause possono essere o un fault di sistema o un pilota distratto. Le protezione contro simili deviazioni nel sistema corrente possono essere ottenute se il controllore si accorge della mancata reazione da parte del pilota dopo che il controllore aveva mandato una certa clearance. Tuttavia l'esercizio di valutazione ha evidenziato la necessità di introdurre in questo sistema altamente critico rispetto alla sicurezza un acknowledgement automatico del fatto che una certa clearance ha raggiunto il sistema del pilota.

Infine, un problema era associato con il task di fermare un aereo durante la sua procedura di decollo. Nel prototipo era sempre possibile fermare un aereo durante il decollo poichè il corrispondente comando (*Stop*) era sempre disponibile sull'interfaccia utente. Tuttavia, nelle situazioni reali il comando "Stop" non dovrebbe essere più abilitato ("ill-timed" performance) una volta che l'aereo ha iniziato la procedura di decollo poichè tentare di interrompere questa azione in quel momento è troppo tardi.

CONCLUSIONI

In questo articolo abbiamo discusso come modelli di task e guidewords possano supportare una sistematica valutazione basata sull'ispezione per applicazioni interattive critiche rispetto alla sicurezza, che può essere utile per analizzare come il progetto di un sistema supporta deviazioni inaspettate nell'esecuzione dei task.

La nostra esperienza ha mostrato l'efficacia del metodo nonostante quelli che abbiamo chiamato "social constraints" che spesso si verificano nelle società di sviluppo software (gli sviluppatori software tendono a difendere ogni decisione presa, gli utenti tendono a divagare, problemi connessi al fattore tempo, ecc).

RIFERIMENTI BIBLIOGRAFICI

Fields, R.E., Harrison, M.D. and Wright, P.C. (1997). THEA: Human Error Analysis for Requirements Definition. University of

York, Dept. of Computer Science, Technical Report YCS-97-294.

Galliers, J., Sutcliffe, A., Minocha, S.(1999), "An Impact Analysis Method for Safety-Critical User Interface Design", ACM Transactions on Computer-Human Interaction, Vol.6, N.4.

Hollnagel, R.(1993), Human Reliability Analysis - Context and Control. Academic Press.

Ivory, M., Hearst, M. (1999), Comparing performance and usability evaluation: New methods for automated usability assessment. 1999. Report available at http://www.cs.berkeley.edu/~ivory/ research/web/papers/pe-ue.pdf.

Johnson, C., and Botting, R. (1999), Reason's Model of Organisational Accidents in Formalising Accident Reports, Cognition, Technology and Work, 1(3), 107-118.

Leveson, N.G. (1995), Safeware: System Safety and Computers -A Guide to preventing accidents and losses caused by technology, Addison Wesley.

Nielsen, J.(1993), Usability Engineering, Boston: Academic Press.

Paternò, F. (1999), Model-based Design and Evaluation of Interactive Applications, Springer Verlag, ISBN 1-85233-155-0.

Reason, J. (1990), Human Error. Cambridge University Press.

Wharton, C., Rieman, J., Lewis, C., and Polson, P.(1994), The Cognitive Walkthrough: A Practitioner's Guide, in J. Nielsen and R.L. Mack (eds.), Usability Inspection Methods, John Wiley & Sons.

RINGRAZIAMENTI

Ringraziamo il supporto ricevuto dalla Commissione Europea e le utili discussioni con I nostri colleghi del progetto MEFISTO (http://giove.cnuce.cnr.it/mefisto.html).