Gestual keyboard for programming

Amedeo Leo
Università degli studi di Salerno
amedeo.leo92@gmail.com

Luigi Lomasto
Università degli studi di Salerno
lomastoluigi@gmail.com

Simone Romano
Università degli studi di Salerno
s.romano1992@gmail.com

ABSTRACT
In questi ultimi anni, i dispositivi mobili sono notevolmente cresciuti in popolarità. È inevitabile che gli sviluppatori software vogliano scrivere codice su di essi.
In quest'articolo viene descritta una tastiera Android basata su gesture e ottimizzata per lo sviluppo di codice in linguaggio Java. In letteratura, tastiere per lo sviluppo sono gi state oggetto di studio. Non esiste, tuttavia, un approccio gestuale per velocizzare l'immissione di codice. I nostri risultati indicano che un programmatore può inserire codice Java con meno keystroke per character, mantenendo, al tempo stesso, una buona velocità di scrittura rispetto alle tastiere standard.

Author Keywords
Gestural input; Text entry; Touch screen; Gestural Keyboard; Android.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: *User Interfaces - input devices and strategies (e.g., mouse, touchscreen)*

INTRODUZIONE E LAVORI CORRELATI

Gli smartphone e i tablet hanno avuto un forte impatto sul mercato mondiale dei dispositivi intelligenti. L' International Data Corporation ha stimato che tra il 2013 e il 2017 le loro vendite aumenteranno rispettivamente del 71% e del 79% [2]. Sebbene i touch screen hanno grandi potenzialità, inserire del testo utilizzando la soft keyboard standard QWERTY (da adesso riferita come QWERTY), non è uno di esse.

A differenza delle tastiere fisiche le soft keyboard dei dispositivi mobili sono generalmente piccole e spesso richiedono di cambiare layer tra caratteri alfabetici e numeri o simboli. Diversi task per dispositivi mobili, tuttavia, sono effettuati in contesti molto specifici e richiedono un linguaggio strutturato a seconda del dominio.

Nel caso specifico si vuole proporre l'utilizzo di dispositivi touch screen per inserire codice di programmazione. Per tali dispositivi sono stati già proposti una serie di tool per effettuare operazioni di editing del testo [5] [6] [8] [7], mentre ne esistono pochi progettati per l'inserimento diretto di codice di programmazione. Un esempio è quello in [4] dove è presentata una estensione della soft keyboard (vedi Figura 1) progettata specificamente per programmatori Java. Il design della tastiera fornisce rapido accesso ai costrutti più utilizzati del linguaggio, raggruppati per specifiche funzioni ed utilizza un approccio syntax-directed per ridurre il numero degli errori.

In questo articolo è presentato un metodo alternativo per inserire codice Java su dispositivi touch screen che verrà confrontato con [4] e con la tastiera QWERTY di Android. Lo scopo è quello di migliorare la velocità e l'efficienza aiutando l'utente durante la stesura del programma.

Nella sezione seguente saranno discussi alcuni lavori correlati. Quindi sarà presentato il metodo di inserimento progettato per poi passare alla descrizione del design degli esperimenti e alla discussione dei risultati. In conclusione saranno descritti possibili sviluppi futuri.

BACKGROUND

In questa sezione sono mostrati lavori correlati alla progettazione di keyboard. Quindi saranno analizzate diverse metriche che verranno usate per valutare il metodo di inserimento progettato e verrà discusso l'approccio syntax directed sviluppato.

La maggior parte dei metodi di inserimento presenti allo stato dell'arte si focalizzano sulla scrittura di mail, word processing, messaggistica istantanea ed altri ambiti non specifici. In questo articolo l'attenzione sarà posta sulla progettazione di una soft keyboard in un ambito definito, ovvero la programmazione in linguaggio Java. Esistono alcuni lavori che forniscono linee guida per la realizzazione e valutazione di strumenti di immissione di codice per dispositivi touch screen.

1. TouchDevelop [7]: la maggior parte dei tool in commercio si concentrano sull'editing di codice esistente. TouchDevelop rappresenta uno dei pochi esempi di strumento di inserimento di codice di programmazione. Esso consiste di un linguaggio e di una IDE che facilitano l'utente nella creazione di programmi per PC. Nella IDE è integrata anche una soft keyboard pensata appositamente per il linguaggio da loro creato. L'interfaccia e la tastiera di TouchDevelop non sono state valutate per usabilità ed efficienza.

2. Syntax-directed editing [4]: gli editor syntax-directed sono stati introdotti nel 1981 per migliorare l'efficienza degli sviluppatori sfruttando la composizione gerarchica del linguaggio. Tali editor richiedono all'utente di inserire codice sintatticamente corretto prima di poter proseguire. In questo modo un programmatore può non essere efficiente se non ha ben memorizzato i comandi dell'editor. La tastiera proposta utilizza un approccio syntax-directed tale da non obbligare l'utente ad inserire codice sempre corretto sintatticamente [3].

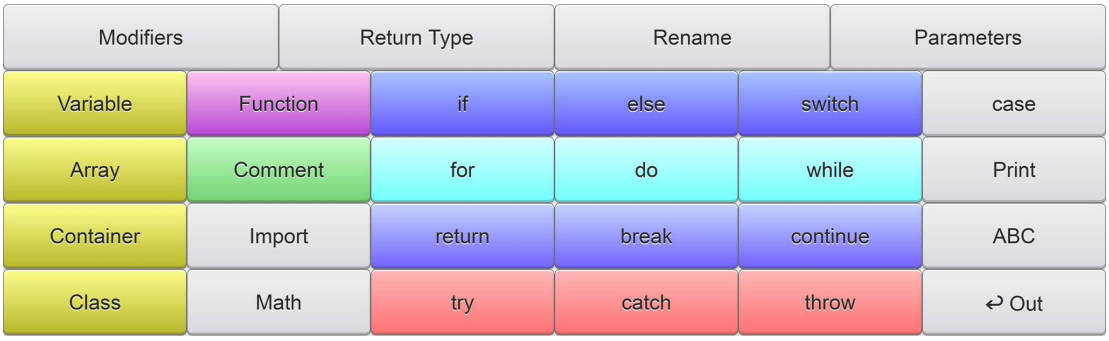

Figure 1. Syntax-directed keyboard extension for writing source code on touch screens

3. Metriche per valutazione di strumenti per immissione del testo [1]: per misurare l'inserimento di testo con una keyboard esistono tre metriche: accuratezza, efficienza e velocità. L'accuratezza è misurata in termini di errori. Per misurare gli errori sono usate MSD (minimum string distance error rate) e TER (total error rate). La prima è una misurazione del numero totale di errori nel testo man mano che lo si inserisce. TER rappresenta la stessa misura ma calcolata sul testo finale. I Keystrokes sono divisi in quattro classi: *Correct* (C), *Incorrect Fixed* (IF), *Fixes* (F) e *Incorrect and Not Fixed* (INF). Nel caso specifico, il TER è così calcolato:

$$TER = \frac{INF+IF}{C+INF+IF} \times 100\%$$

Keystrokes per character (KSPC) misura la media del numero di keystrokes richiesti per inserire un singolo carattere. Su una tastiera QWERTY standard il KSPC è circa uno ma è stato dimostrato che può raggiungere 1.21 in seguito alla correzione degli errori. La velocità di inserimento del testo è misurata in words per minute (WPM), dove la parola si ritiene lunga in media cinque caratteri.
Per valutare la tastiera proposta sono state utilizzate TER, KSPC e WPM per valutare efficienza, velocità e accuratezza dell'input.

GESTURAL KEYBOARD

Come per la tastiera sviluppata in [4], *Gestural Keyboard* (vedi Figura 7) è stata pensata per ridurre gli errori di immissione e quelli di sintassi con lo scopo di migliorare l'efficienza. Il design e le modalità di funzionamento consentono all'utente una rapidità di inserimento dei costrutti Java più utilizzati mantenendo un approccio syntax-directed.

La progettazione è stata determinata dall'analisi sulla frequenza delle parole e dei costrutti più comuni del linguaggio Java. È stato effettuato un attento studio sulla natura della grammatica del linguaggio.

I costrutti più utilizzati del linguaggio sono stati raggruppati in appositi menù (presenti al di sopra della tastiera QWERTY tradizionale) accessibili tramite intuitive gesture. In particolare ciascuna riga rappresenta le varie opzioni associate alla relativa gesture inserita dall'utente. La Figura 3 mostra un set di costrutti del linguaggio a cui si può accedere tramite la gesture mostrata. Alcuni dei bottoni dei menù mostrati in Figura 3, come ad esempio *try*, inseriscono

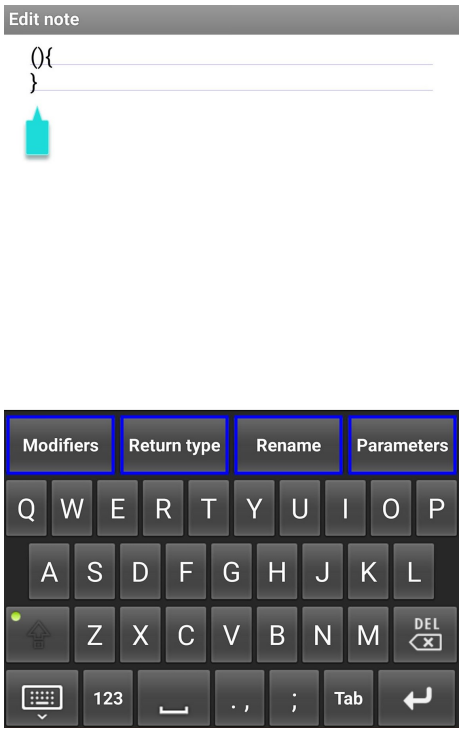

Figure 2. Gestural keyboard: tastiera per l'inserimento di codice in linguaggio Java.

direttamente blocchi di codice; altri consentono l'accesso ad ulteriori opzioni relative alla funzione scelta (es. *modifiers*).

Alcune delle gesture, come mostrato in figura 3, aprono un ulteriore menù di funzionalità. Ad esempio, la gesture *f* guida l'utente nell'inserimento di una nuova funzione. Inserita la gesture, l'utente vede inserito il seguente frammento di codice:

```
() {
}
```

L'utente ha quindi visibile il relativo menù (*modifiers*, *return type*, *rename*, *parameters*) ed il cursore posizionato prima delle parentesi tonde. A questo punto l'utente può inserire il nome della funzione, un modificatore (aiutandosi tramite l'apposito menù con *private*, *public*, *protected*, *static*, *final*, *public static*), il tipo di ritorno (*int*, *double*, *boolean*, *char*,

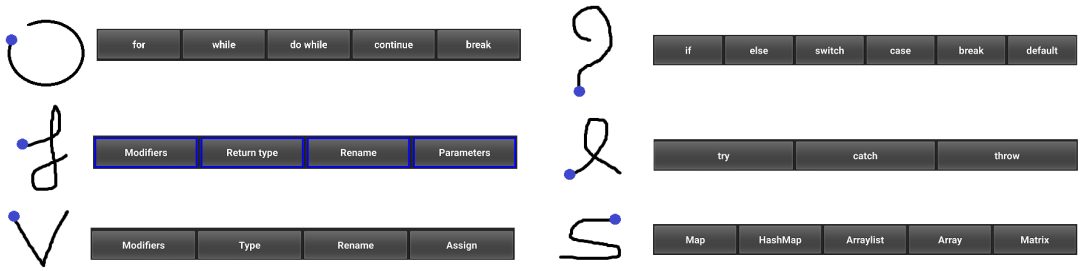

Figure 3. Gesture previste dalla tastiera Gestural Keyboard per accedere a specifici costrutti del linguaggio.

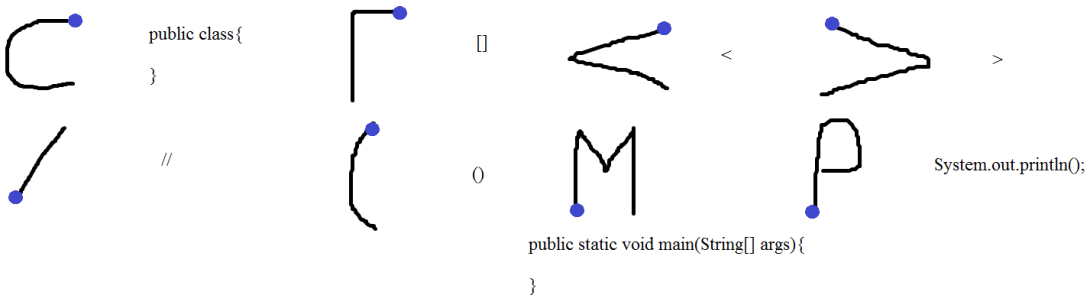

Figure 4. Gesture previste dalla tastiera Gestural Keyboard per inserire specifici costrutti del linguaggio.

Gesture	Descrizione
	Commenti
	Codice classe
	Simbolo maggiore
	Simbolo minore
	Codice metodo main
	Codice println
	Parentesi quadre
	Parentesi tonde
	Accesso menù cicli
	Codice funzione e accesso menù funzione
	Accesso menù operatori di controllo
	Accesso menù eccezioni
	Accesso menù strutture
	Accesso menù variabile

Table 1. Gesture previste dalla tastiera Gestural Keyboard.

String, *void*, *Custom*) ed eventuali parametri. Tutte le operazioni possono essere effettuate in qualsiasi ordine; il cursore si sposta, a seconda della scelta effettuata, nella posizione giusta.

Gestural Keyboard supporta una forma limitata di editing orientata alla sintassi. Uno degli svantaggi è quindi la mancanza di flessibilità: infatti ciascuno statement inserito deve preservare la correttezza sintattica. Ciò non è sempre apprezzato dai programmatori che potrebbero avere necessità di violare, in alcune circostanze, la sintassi usuale. Per eliminare questo problema è sempre presente la tastiera QWERTY tradizionale.

METODO

Si descrive in questa sezione la tipologia di esperimenti effettuati per comparare le performance della Gestural Keyboard realizzata con la tastiera descritta in [4] e con la tastiera QWERTY di Android. Si è scelto un task di copia dei due pezzi di codice in figura 5 e figura 6 piuttosto che la scrittura di nuovo codice per evitare la componente di difficoltà (anche minima) dovuta alla progettazione del software e per dare agli utenti la possibilità di focalizzarsi esclusivamente sull'utilizzo della tastiera.

Per effettuare gli esperimenti è stata progettata e realizzata la tastiera Gestural Keyboard in linguaggio Java. È stata quindi implementata la tastiera descritta in [3] ed un editor (sempre per dispositivi Android) per guidare gli utenti nell'esecuzione dei task e per ottenere le misurazioni di TER, KSPC, WPM come descritte in 3.

Partecipanti

Hanno partecipato agli esperimenti nove studenti (di cui otto uomini ed una donna) iscritti ai corsi di Laurea Triennale

```
public class Point{
    private double x;
    private double y;
    // constructor
    public Point(double px, double py) {
        x = px;
        y = py;
    }
    public void setX(double px) {
        x = px;
    }
    public void setY(double py)  {
        y = py;
    }
}
```

Figure 5. Primo codice immesso dai partecipanti. Il codice sottolineato deve essere inserito tramite la tastiera QWERTY tradizionale.

```
int[][] matrix = new int[10][5];
for (int i=0; i<10; i++){
    for(int j=1; j<5; j++){
        System.out.println(i + j);
        if(i==j){
            System.out.println("diagonal");
        }
    }
}
```

Figure 6. Secondo codice immesso dai partecipanti. Il codice sottolineato deve essere inserito tramite la tastiera QWERTY tradizionale.

e Magistrale in Informatica la cui età varia dai 20 ai 29 anni. Tutti i partecipanti hanno affermato di utilizzare quasi sempre dispositivi touch; tuttavia nessuno ha mai utilizzato tali dispositivi per la stesura di codice di programmazione. La maggior parte dei partecipanti (7 su 9) è destrorsa.

Apparatus

Il dispositivo utilizzato per gli esperimenti è un LG G3 D855 32GB con display da 5.5" e risoluzione dello schermo di 2560 x 1440 pixel.

Procedura

Prima di cominciare ogni partecipante compila un questionario informativo. Quindi segue una fase di training sulle due tastiere non familiari all'utente ([4] e Gestural Keyboard) della durata di cinque minuti ciascuna. Successivamente l'utente ha cinque minuti per esercitarsi e prendere confidenza con le tastiere prima di cominciare l'esperimento. I partecipanti sono stati incoraggiati a chiedere qualsiasi domanda prima di iniziare i task. Alla fine dell'esperimento è stato chiesto agli utenti di compilare un questionario NASA Task Load Index (NASA-TLX) per determinare il loro workload mentale con l'utilizzo delle tre tastiere ed altre domande sulla loro esperienza con le tastiere.

La partecipazione all'esperimento è richiesta come parte integrante del corso di *Progettazione di Sistemi Interattivi* del corso di Lurea Magistrale in Informatica.

Ogni partecipante ha effettuato tre esperimenti (uno per ogni tastiera). Ogni esperimento prevede due task che prevedono l'inserimento dei blocchi di codice 5 e 6. Ciascun task inizia quando l'utente digita il primo carattere e termina al click del bottone *End task* presente nell'editor realizzato. Per completare il task è necessario che l'utente non commetta più di cinque errori.

Design

L'esperimento prevede tre variabili dipendenti (KSPC, TER e WPM) ed una variabile indipendente con tre level (Gestural Keyboard, Syntax-directed keyboard e QWERTY). L'esperimento è un within-subject 9x3. L'ordine di utilizzo delle differenti tastiere è stato bilanciato con un Latin Square counterbalancing.

RISULTATI E DISCUSSIONI

L'ipotesi iniziale è che gli utenti potrebbero inserire codice in maniera più veloce, più efficace e commettendo meno errori con la tastiera Gestural Keyboard rispetto alla tastiera QWERTY e la tastiera descritta in [4]. Segue che la nostra ipotesi nulla è che non ci sono differenze significative tra le distribuzioni delle corrispondenti misure di performance delle tre differenti keyboard. Per tutte le misurazioni è stato usato il test ANOVA per validare i risultati.

TER

La media del total error rate è stata 15.43% (SD: 10.96%) per la tastiera Gestural Keyboard, 10.25% per la tastiera Syntax-Directed Keyboard Extension (SD: 7.9%) ed 8.04% (SD: 3.65%) con la QWERTY. L'analisi della varianza tuttavia non ha evidenziato significatività statistica ($p = 0.08$).

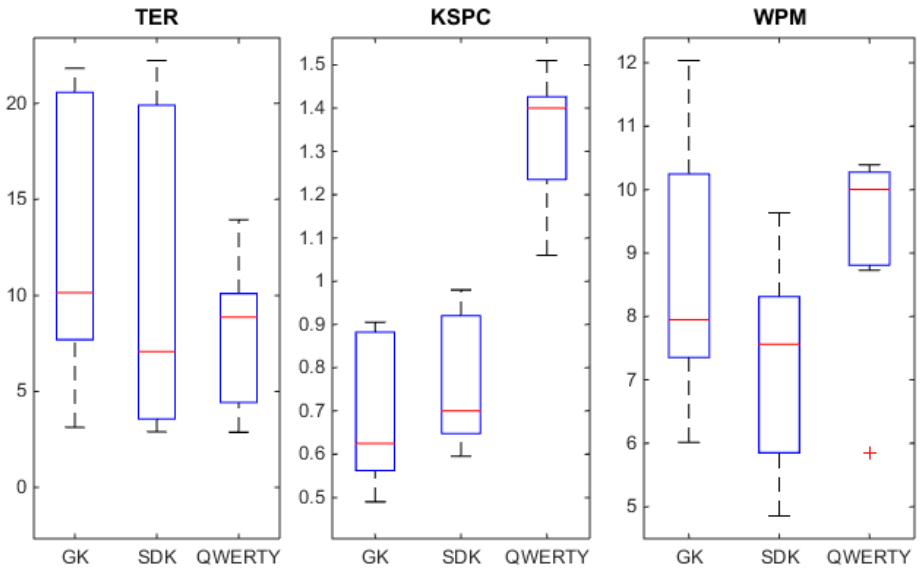

Figure 7. TER, KSPC e WPM per la tastiera Gestural Keyboard, Syntax-Directed Keyboard Extension e QWERTY.

KSPC

La media del KSPC è stata 0.75 (SD: 0.27) per la tastiera Gestural Keyboard, 0.78 per la tastiera Syntax-Directed Keyboard Extension (SD: 0.16) ed 1.33 (SD: 0.15) con la QWERTY. L'analisi della varianza ha mostrato significatività statistica ($p < 0.0001$). È stato effettuato un test post-hoc (Scheffé) che ha mostrato significatività statistica tra le tastiere Gestural Keyboard e QWERTY e tra la tastiere Syntax-Directed Keyboard Extension e QWERTY.

WPM

La media del WPM è stata 8.57 (SD: 2.24) per la tastiera Gestural Keyboard, 7.42 per la tastiera Syntax-Directed Keyboard Extension (SD: 1.52) ed 9.02 (SD: 1.52) con la QWERTY. L'analisi della varianza tuttavia non ha evidenziato significatività statistica ($p = 0.14$). Nonostante ciò il grafico in figura 7 mostra come i valori maggiori in WPM della tastiera Gestural Keyboard sono più alti dei valori maggiori della tastiera Syntax-Directed Keyboard Extension.

Feedback partecipanti

Al termine dell'esperimento, ai partecipanti è stato somministrato un questionario circa l'esperienza con le tre tastiere. È stato chiesto di valutare l'esperienza e l'uso delle tre tastiere nella stesura di programmi Java. La tastiera GesturalKeyboard ha ottenuto il 55.6% di risposte "Decisamente positiva" nell'esperienza e il 66.7% di risposte "Di buon aiuto" nella scrittura di codice, con un parere "Facile da usare" nel 66.7% dei casi. La tastiera Syntax-Directed Keyboard Extension ha ottenuto il 66.7% di risposte "Negativa" nell'esperienza, mentre la tastiera QWERTY un'opinione "Negativa" nell'utilizzo complessivo nel 44.4% dei casi. È stato anche somministrato un questionario NASA Task Load Index per determinare richiesta mentale, richiesta fisica, tempo, performance, frustrazione e sforzo delle tre tastiere. Le possibilità spaziano da 1 (fortemente in disaccordo) a 7 (pienamente daccordo). Le risposte alla domanda "Richiesta mentale" della GesturalKeyboard sono state nel 44.4% dei casi "Neutrale", così come le perfomance. La tastiera Syntax-Directed Keyboard Extension ha invece ottenuto il 44.4% delle risposte "Disaccordo" e il 66.7% delle risposte "Neutrale" nelle medesime domande. Tra i feedback dei partecipanti, 7 su 9 hanno preferito la tastiera GesturalKeyboard per la sua intuitività, semplicità e usabilità.

Discussioni

I risultati dello studio indicano che gli utenti hanno scritto codice in maniera più efficiente con la GesturalKeyboard, come riportato dal valore di KSPC. Una spiegazione plausibile è che le funzionalità desiderate dall'utente sono immediatamente disponibili in seguito all'inserimento della relativa gesture; invece per la tastiera Syntax-Directed Keyboard Extension è spesso necessario dover effettuare uno switch tra i costrutti del codice e la tastiera per l'immissione dei caratteri. Analogamente ciò avviene con la tastiera QWERTY poichè per poter inserire simboli o caratteri speciali sono necessari keystroke aggiuntivi per lo switch del layout. Il risultato ottenuto è in particolare dovuto alle funzionalità implementate: nell'immettere, ad esempio, il nome di una classe, nella tastiera GesturalKeyboard è solo necessario, dopo aver fatto la relativa gesture, inserire il nome della classe; lo stesso costrutto è utilizzabile tramite la tastiera Syntax-Directed Keyboard Extension con la key "Class". Per la qwerty, questa operazione necessita di un numero di keystroke pari o superiore al numero dei caratteri.

Per i motivi appena descritti l'analisi della varianza del parametro WPM non ha evidenziato significatività statistica tra le due tastiere, sebbene la GesturalKeyboard ha riscontrato valori più alti rispetto alla Syntax-Directed Keyboard Extension. La tastiera QWERTY invece è risultata la più perfor-

mante in termini di WPM essendo la tastiera maggiormente conosciuta.
I risultati ottenuti dal TER non hanno mostrato significatività statistica tra nessuna delle tastiere utilizzate. Come si può dedurre dal grafico nessuna delle tre tastiere sembra prevalere, in modo assoluto, sulle altre. Nonostante ciò la tastiera GesturalKeyboard è risultata la peggiore. Tali risultati possono essere attribuiti ad un'errata interpretazione delle gesture inserite. L'implementazione della tastiera ha infatti previsto l'associazione 1:1 tra gesture e riconoscitore della relativa funzionalità che ha portato all'immissione di costrutti e/o caratteri errati all'interno del testo. In alcuni casi, ad esempio, gli utenti hanno riscontrato difficoltà nell'inserire il costrutto classe, poichè veniva interpretato come l'immissione di una parentesi angolare.

CONCLUSIONI E SVILUPPI FUTURI

Il nostro studio indica che una nuova tastiera comprensiva di gesture ha diversi benefici nella stesura di codice Java; in particolare tali benefici riguardano il numero di keystroke per carattere. Inoltre è stata valutata dagli utenti una richiesta mentale minore delle altre. Il tempo percepito dagli utenti, per l'immissione di codice, è risultato inferiore.

La difficoltà incontrata dagli utenti, tutti novizi nella programmazione Java su dispositivi touch, è stata quella di dover apprendere l'utilizzo di due tastiere sconosciute all'interno dei 10 minuti concessi. Ciò è risultato negativo nei risultati ottenuti, sopratutto per quanto riguarda la tastiera GesturalKeyboard, poichè stato necessario apprendere o consultare le diverse gesture e le relative funzionalità.

Per gli sviluppi futuri, la maggior parte dei feedback degli utenti ha riguardato la possibilità di inserire una nuova key per automatizzare l'inserimento del "costruttore" di una classe. Un'ulteriore automatizzazione è stata richiesta nell'immissione del carattere ";" , poichè è spesso necessario dover andare a capo e continuare con l'immissione del testo. Per migliorare, infine, i risultati della variabile TER, si pu'o prevedere l'ottimizzazione del riconoscitore delle gesture inserendo un'associazione 1:N per ciascuna gesture.

REFERENCES

1. "Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and a New Unified Error Metric". Disponibile a: . **http://www.yorku.ca/mack/chi03.html**.
2. "Tablet Shipments Forecast to Top Total PC Shipments in the Fourth Quarter of 2013 and Annually by 2015, According to IDC [Online]". Disponibile a: . **http://www.businesswire.com/news/home/20130911006186/en/Tablet-Shipments-Forecast-Top-Total-PC-Shipments**.
3. "The cornell program synthesizer: a syntax-directed programming environment". Disponibile a: . **http://dl.acm.org/citation.cfm?id=358755**.
4. Almusaly, I., and Ronald, R. "A Syntax-Directed Keyboard Extension for Writing Source Code on Touch Screens". **http://www3.nd.edu/~rmetoyer/pubs/syntaxDirectedKeyboard.pdf**.
5. Cooper, S., Dann, W., and Pausch, R. "Alice: a 3-d tool for introductory programming concepts". *Journal of Computing Sciences in Colleges*.
6. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., and Silverman, B. "Scratch: programming for all".
7. Tillmann, N., Moskal, M., de Halleux, J., Fahndrich, M., and Burckhardt, S. "TouchDevelop: app development on mobile devices". *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. **http://dl.acm.org/citation.cfm?id=2393641**.
8. "Two Lives Left". Disponibile a: . **http://twolivesleft.com/Codea/**.