Tecnologia, comunicazione e pratica nelle comunità di sviluppatori OpenSource: Plone e Php-ZeroNet.

1. Introduzione

Il lavoro si è occupato di analizzare il ruolo della tecnologia in due progetti di sviluppo di software OpenSource.
L'interesse per l'argomento nasce dall'osservazione di come Internet sia divenuto un ambiente affidabile e performante in cui aziende private, istituzioni statali ed in genere organizzazioni formali ed informali, archiviano, scambiano e producono informazioni. La produzione di un software implica, oltre ad gestione delle informazioni, anche una precisa strategia operativa, una puntuale pianificazione ed un impegno costante da parte dei partecipanti al progetto di sviluppo. Tutto questo per il felice raggiungimento dell'obiettivo comune che è, in sostanza, la produzione di un artefatto.
La relativa facilità di comunicazione consentita da Internet (attraverso l'uso di tecnologie come l'email o IRC) e un conseguente abbattimento delle barriere spaziali (Giddens, 1993), ha permesso, nella prima metà degli anni 90, la nascita di Linux, un sistema operativo per computer, sviluppato interamente attraverso la collaborazione spontanea di programmatori tenuti in comunicazione e coordinati esclusivamente da tecnologie informatiche. Oggetto della mia ricerca sono due progetti che si propongono di sviluppare un'applicazione di Content Management (un software che permette, anche a persone che non conoscono alcun linguaggio di programmazione, di pubblicare, gestire e organizzare i contenuti di un sito web), in un contesto in cui la comunicazione è fortemente mediata dalla tecnologia (Internet, appunto).

Nel capitolo 2 presenterò i modelli teorici di riferimento, individuati principalmente nei lavori di Mantovani (1996) e Wenger (1998). Nel terzo capitolo cercherò di utilizzare i concetti chiave dei modelli teorici per introdurre l'ambiente in cui prendono vita e sono sviluppati i due progetti presi in esame. Nel quarto capitolo illustrerò l'approccio metodologico e gli strumenti utilizzati per raccogliere i dati utili ai fini della ricerca. La sezione successiva (Capitolo 5) espone dettagliatamente i due casi di studio. Chiudono questo lavoro una sezione dedicata alle osservazioni finali (capitolo 5) e i riferimenti bibliografici.

2. Riferimenti teorici

Il framework teorico proposto da Mantovani (Mantovani, 1996) sottolinea la necessità di prestare attenzione al contesto sociale ed ai processi di conflitto/negoziazione per comprende e meglio progettare ambienti in cui la tecnologia funge da mediatore nella comunicazione tra gli individui.
Sono tre i livelli in cui si può scomporre una situazione di CMC (Computer Mediated Communication): il contesto sociale, le situazioni, gli artefatti.
Vedremo ora di introdurre brevemente i tre concetti.

- Livello 1 – Costruzione del contesto: il contesto sociale è il risultato dell'interazione tra i modelli culturali e le norme sociali (s*tructure*) e l'azione degli individui (*action*). Si sostiene, in pratica, che la *structure* sia preesistente a qualsiasi forma di interazione tra gli individui. L'emergere di modelli di attività (*patterns of activities*) da parte degli individui è regolata dalle norme sociali e dai modelli culturali definiti nella struttura. In realtà la relazione tra struttura ed azione e estremamente dinamica: la struttura definisce il lecito e l'illecito, il desiderabile e il non desiderabile, ma è continuamente ridefinita dall'azione degli attori che agiscono al suo interno.

L'azione, appunto, è il secondo importante elemento del primo livello del modello. Muove dalla valutazione della situazione corrente ed è il primo passo per modificarla a proprio favore.
La relazione dinamica tra struttura e azione produce la storia del contesto (*history*). La situazione in cui si muovono gli attori è intrinsecamente instabile. Si possono individuare due processi: nel primo, l'attore organizza i propri progetti e attribuisce significati agli oggetti in base ai modelli

disponibili nell'ordine culturale preesistente; nel secondo, l'attore deve riconsiderare in maniera creativa i propri modelli culturali, in quanto, sotto la costante pressione dell'azione non è possibile attribuirvi, in momenti diversi, gli stessi significati.

- Livello 2 – Interpretazione della situazione. I tre elementi presenti nel secondo livello sono le opportunità (*opportunities*), gli interessi (*interests*), gli obiettivi (*goals*).

Gli attori non hanno un unico interesse che intendono soddisfare; esistono una molteplicità di interessi in competizione tra loro per raggiungere l'attenzione e la priorità nel sistema d'azione dell'attore. Gli interessi che hanno la priorità in un dato momento rendono salienti alcuni elementi del contesto, che, a loro volta, rendono più forti quei dati interessi.

Le opportunità muovono quindi dagli interessi: le situazioni quotidiane contengono molteplici opportunità, che rimangono in un limbo fino al momento in cui non sono portate nel mondo reale da un particolare interesse. Le situazioni, come il contesto sociale, sono costruite dai significati ad esse attribuiti dagli attori.

Gli obiettivi sono il prodotto del cambiamento che segue la relazione tra opportunità ed interessi.

La relazione tra interessi e opportunità può essere riassunta in due fasi: le opportunità influenzano la priorità degli interessi; gli interessi riconoscono le opportunità che possono essere colte all'interno della situazione.

- Livello 3 - Interazione locale con l'ambiente. Gli elementi costitutivi di questo livello sono l'artefatto (tool), l'utente (user) e i compiti (tasks).

Gli artefatti (tools o artefatti cognitivi) sono oggetti creati per un determinato proposito. Non sono oggetti neutri, ma contengono conoscenza, in quanto, ad esempio, la loro foggia, le funzionalità che rendono disponibili sono strettamente legate al contesto per cui sono stati progettati. I sistemi informatici sono artefatti che contengono due classi di obiettivi e progetti. La prima classe si riferisce a quegli obiettivi che sono assegnati loro dai progettisti e dai tecnici che li hanno pensati e realizzati. La seconda classe di obiettivi e progetti deriva dall'utilizzo da parte degli utenti, che tendono a superare i confini d'uso pensati dai progettisti per adattarli alle necessità contingenti per la realizzaione dei loro progetti. Questo concetto introduce il terzo elemento costitutivo di questo livello: i task, i compiti che dobbiamo portare a termine per realizzare il progetto. Nelle situazioni quotidiane gli attori ragionano più in termini di progetti che in termini di compiti che devono essere eseguiti. Gli obiettivi dell'attore favoriscono l'emergere di *progetti* (alto livello di astrazione e complessità), da cui derivano *piani* appropriati (medio livello di astrazione e complessità), da cui derivano una serie di *compiti* (basso livello di astrazione e complessità) che devono essere eseguiti perché i piani siano portati a termine, i progetti realizzati, gli obiettivi raggiunti. L'eseguire un task utilizzando un sistema informatico non significa solamente eseguire un compito, ma comporta una certa impredicibilità dovuta alle proprietà dell'interazione tra attori e situazione. L'uso dell'artefatto da parte dell'attore in una determinata situazione implica una modifica del progetto iniziale incluso dal progettista nell'artefatto: l'uso produce innovazione. Si parla di task-artifact cycle: l'uso di uno strumento da parte di un utente è il primo passo per lo sviluppo di un nuovo strumento.

Le comunità di pratica (Wenger, 1998) offrono uno strumento interpretativo utile per analizzare la produzione e la trasmissione della conoscenza. Una comunità di pratica è un sistema sociale di apprendimento. Fare parte di una comunità vuol dire condividerne le conoscenze collettive. Ogni qual volta entriamo a far parte di una nuova comunità (ad esempio un nuovo ambiente lavorativo) apprendiamo nuove regole e nuove conoscenze: l'apprendimento è una relazione dinamica tra il nosto bagaglio di esperienze e le competenze consolidate del nuovo ambiente in cui ci stiamo inserendo. Le competenze sono socialmente definite. Ad esempio, nel mondo Open Source nessuno può dichiarare di essere un hacker (lo sviluppatore di Free software): è la comunità che riconosce tale titolo ad un individuo, che riconosce una competenza. Con competenza non si intende solo la capacità di fare bene qualcosa (un programmatore competente non è solo un programmatore preparato). Nel contesto delle comunità di pratica si definisce competente chi comprende il luogo in

cui opera, perchè ne conosce le regole, la storia, le modalità di interazione tra i membri, ne condivide gli stili di comportamento, le tradizioni e le risorse e si impegna per raggiungere l'obiettivo comune. Inoltre l'appartenenza ad una comunità ha implicazioni notevoli su come percepiamo noi stessi ed il resto del mondo. La nostra identità è plasmata dall'appartenere a comunità: abbiamo la capicità di identificarci fortemente con alcuni gruppi e di non identificarci per niente con altri. I confini fra ciò a cui riteniamo di appartenere e ciò a cui riteniamo di essere estranei sono estremamente variabili ed in costante mutamento. Si parla in realtà di multiappartenenza per sottolineare come noi ci muoviamo continuamente da una comunità all'altra: lo sviluppatore che la sera partecipa ad un progetto Free Software su Internet non cessa di essere il libero professionista che lavora durante il giorno. La multiappartenenza è un concetto fondamentale per le comunità di pratica perchè è la modalità attraverso cui avviene l'immissione di nuove esperienze personali nei sistemi sociali di apprendimento.

3. Il contesto

Il modello teorico precedentemente illustrato (Mantovani, 1996) assegna un ruolo fondamentale al contesto che fa da cornice ed influenza l'azione degli individui. Nel caso specifico della mia ricerca contesto è multi dimensionale: una prima dimensione è identificabile nel movimento Free Software, sviluppatosi a partire dai primi anni 80, che definisce sia le norme e le modalità di comportamento sia i valori condivisi a cui fanno riferimento gli individui che intendono partecipare alla comunità Free Software; la seconda dimensione è costituita dal luogo online in cui si svolge l'azione, Internet in generale, e il sito Sourceforge.net (http://www.sourceforge.net) in particolare. Una terza dimensione è costituita dalla storia personale degli individui, che ha un ruolo fondamentale nel nel definire il modo in cui gli attori percepiscono e attribuiscono significati alla situazione e all'ambiente in cui sono inseriti.
Nei prossimi due paragrafi presenterò le caratteristiche principali del movimento Free Software e la struttura e i servizi offerti da SourceForge.net.

3.1. Free Software

È bene chairire i significati di Free Software e Open Source. I due termini indicano due movimenti vicini, che lavorano spesso assieme, ma che hanno delle caratteristiche diverse. Sul sito http://www.fsf.org (il sito della Free Software Foundation) si trova la seguente definizione: "*La differenza fondamentale tra i due movimenti sta nei loro valori, nel loro modo di guardare il mondo. Per il movimento Open Source, il fatto che il software debba essere Open Source o meno è un problema pratico, non un problema etico. Come si è espresso qualcuno, "l'Open Source è una metodologia di sviluppo; il Software Libero è un movimento di carattere sociale." Per il movimento Open Source, il software non libero è una soluzione non ottimale. Per il movimento del Software Libero, il software non libero è un problema sociale e il software libero è la soluzione. (...) Il movimento del Software Libero e quello Open Source sono come due partiti politici all'interno della comunità del Software Libero.*" (http://www.fsf.org/philosophy/free-software-for-freedom.it.html). È vero, comunque, che nell'uso comune i due termini sono ormai considerati sinonimi.
Dalla definizione sopra riportata emerge esplicitamente il carattere sociale del movimento Free Software. Le caratteristiche fondamentali che deve avere un software per essere ritenuto Free chiariscono ulteriormente questo concetto, introducendo quelli che sono due dei principali valori condivisi dai partecipanti al movimento Free Software: la libertà per chiunque di utilizzare il software e la libertà per chiunque di modificarlo e di adattarlo alle proprie necessità. Di seguito

riporto la definizione di Free Software, reperibile all'indirizzo http://www.fsf.org/philosophy/free-sw.it.html: *"L'espressione "software libero" si riferisce alla libertà dell'utente di eseguire, copiare, distribuire, studiare, cambiare e migliorare il software. Più precisamente, esso si riferisce a quattro tipi di libertà per gli utenti del software:*

1. *Libertà di eseguire il programma, per qualsiasi scopo (libertà 0).*
2. *Libertà di studiare come funziona il programma e adattarlo alle proprie necessità (libertà 1). L'accesso al codice sorgente ne è un prerequisito.*
3. *Libertà di ridistribuire copie in modo da aiutare il prossimo (libertà 2).*
4. *Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio (libertà 3). L'accesso al codice sorgente ne è un prerequisito.*

Un programma è software libero se l'utente ha tutte queste libertà. In particolare, se è libero di ridistribuire copie, con o senza modifiche, gratis o addebitando delle spese di distribuzione a chiunque ed ovunque. Essere liberi di fare queste cose significa (tra l'altro) che non bisogna chiedere o pagare nessun permesso.[...]".

La libertà di accesso al software e la possibilità di modificarlo costituisce, senza dubbio, un efficiente motore per l'innovazione (attraverso il virtuoso task-artifact cycle). Eric S. Raymond ne "La cattedrale e il bazaar" (1998), divenuto uno dei testi di riferimento per la comunità Free Software, scrive: "*Ogni buon lavoro software inizia dalla frenesia personale di uno sviluppatore. (...) I bravi programmatori sanno cosa scrivere. I migliori sanno cosa riscrivere (e riusare)*" (Raymond, 1998). Il concetto di riutilizzo creativo del software scritto da altri, per adattarlo alle proprie necessità, è un elemento guida di tutta la produzione Free. Il modello di sviluppo scaturito dalle esperienze dei progetti Free Software (definito *bazaar* da Raymond), poggia fortemente su questo concetto: progetti abbandonati vengono poi ripresi da altri sviluppatori, riciclati e modificati per adattarli alla necessità di un utente o di un gruppo di utenti. Ecco un secondo importante elemento del modello di sviluppo *bazaar,* l'utente: la maggior parte dei progetti non sopravvivono a lungo se non vi sono utenti interessati ad essi, che danno il proprio contributo usando il software, individuandone i difetti, proponendo nuove funzionalità. Una delle caratteristiche del modello di sviluppo *bazaar* è la velocità e la frequenza con cui vengono rilasciate nuove release del software. Cito di nuovo Raymond: "*Altro punto di forza della tradizione Unix, portato felicemente agli estremi da Linux, è che molti utenti sono essi stessi degli hacker. Ed essendo i sorgenti disponibili a tutti, possono diventare degli hacker molto efficaci. Qualcosa di tremendamente utile per ridurre il tempo necessario al debugging. Con un po' d'incoraggiamento, ogni utente è in grado di diagnosticare problemi, suggerire soluzioni, aiutare a migliorare il codice in maniera impensabile per una persona sola.*" (Raymond, 1998). Rilasciare frequentemento release instabili del software vuol dire risolvere velocemente i difetti attraverso un apporto continuo da parte degli utenti.

La comunità ha quindi un ruolo cruciale nel successo dei progetti Free Software, sia perchè fornisce le risorse umane che si impegnano nello sviluppo, sia perchè è la fonte dei bisogni da cui scaturiscono i progetti.

È ovvio che in una situazione di questo tipo è indispensabile la visibilità dei progetti. Su internet sono pubblicati milioni di siti web; l'alto numero di siti rende molto difficile, per l'utente, l'individuazione dei contenuti di interesse. In sostanza, se gli utenti non sanno che un progetto è attivo, non possono utilizzare il software in questione, e, allo stesso modo, potenziali sviluppatori interessati a dare il proprio contributo non possono partecipare: il modello bazaar non è quindi applicabile.

La soluzione a questo problema è stata la creazione di un archivio centralizzato per i progetti Open Source.

3.2. Sourceforge.net

Sourceforge.net (http://www.sourceforge.net) è un sito web che offre agli sviluppatori, gratutitamente, strumenti utili allo sviluppo di software. Inoltre, è ormai divenuto punto di riferimento per quegli utilizzatori di Free Software che sono alla ricerca del prodotto più adatto alle loro esigenze. Un efficiente motore di ricerca interno al sito opera su più di 74.000 progetti.
Nella mission di Sourceforge.net si legge: "*SourceForge.net is the world's largest Open Source software development web site, providing free hosting to tens of thousands of projects. The mission of SourceForge.net is to enrich the Open Source community by providing a centralized place for Open Source developers to control and manage Open Source software development.*" (http://sourceforge.net/docman/display_doc.php?docid=6025&group_id=1).
Gli utilizzatori di software possono collegarsi e scaricare il software desiderato gratuitamente, senza la necessità di effettuare nessuna iscrizione.
Per utilizzare i servizi riservati agli sviluppatori è, invece, necessario registrarsi. Durante la registrazione viene fornito un nome utente ed una password da utilizzare per l'accesso al sito. Una volta in possesso di tali informazioni è possibile partecipare attivamente ai progetti ospitati o avviare un proprio progetto. Gli sviluppatori registrati sono più di 750.000.
Sourceforge.net mette a disposizione dei propri utenti strumenti utili sia alla parte tecnica dello sviluppo che alle attività di project management e di promozione del progetto.
Come detto precedentemente, il sito è un punto di riferimento per la comunità. Inserire il proprio progetto in Sourceforge.net vuole dire proporlo alla comunità: sarà poi quest'ultima che, in base all'interesse, deciderà di partecipare attivamente al progetto.
Ogni progetto ospitato ha a disposizione un vero e proprio sito dedicato: è possibile pubblicare i file realizzati, la documentazione, la lista dei task da svolgere, l'elenco dei bug (i difetti del software) e delle patch (le correzioni) da implementare. Ogni progetto ha a disposizione un suo forum su cui la comunità può esprimere valutazioni e richieste, ricevere informazioni e supporto sull'utilizzo del programma in questione.
Uno strumento molto importante messo a disposizione da Sourceforge.net è CVS. CVS sta per "Concurrent Version System" ed è un programma che consente la gestione dei molteplici file che costituiscono un software. CVS è un sistema di controllo della versione: tiene traccia delle modifiche fatte ad un file nel corso del tempo e permette il recupero di una versione precedente all'ultima prodotta. Un sitstema quale CVS è estremamente importante nello sviluppo di qualsiasi software e lo è ancora di più in un contesto nel quale le persone coinvolte non condividono lo stesso ufficio durante le stesse ore della giornata.
Un'ultima nota su Sourceforge.net: da tutte le aree del sito è possibile accedere alla sezione "Project Help Wanted". In questa sezione i responsabili dei progetti lanciano richieste di aiuto alla comunità, per trovare persone con capacità e conoscenze specifiche da integrare nei loro gruppi di lavoro.
È subito possibile individuare una stretta connessione tra Sourceforge.net e il modello di sviluppo *bazaar*. Sourceforge.net offre supporto alla comunità, attraverso strumenti per coordinare il lavoro a distanza di persone che risiedono in diverse parti del mondo, hanno lingue diverse, lavorano ai progetti ad ore del giorno completamente diverse. Si configura inoltre come un ambiente online in cui avviene la condivisione e lo scambio di esperienze, in cui avvengono incontri e si avviano collaborazioni. Non è raro, ad esempio, che uno sviluppatore collabori contemporaneamente a più progetti, mettendo in atto quei processi di confine tra un gruppo di sviluppo ed un altro (Wenger, 1998) che concorrono a produrre conoscenza.

4. Metodologia e strumenti

I due progetti seguiti nel corso della ricerca sono stati affrontati in modo diverso. Php-ZeroNet (http://sourceforge.net/projects/phpnetzero/) ha visto una mia partecipazione diretta come

sviluppatore per un periodo di circa cinque mesi. Esporrò la mia esperienza con Php-ZeroNet nel prossimo capitolo.
Per il secondo progetto (Plone: http://sourceforge.net/projects/plone) ho preparato un'intervista strutturata che ho somministrato agli sviluppatori coinvolti (l'elenco dei 77 partecipanti, con tanto di indirizzo email, è disponibile su Sourceforge.net). L'intervista, in lingua inglese, è costituita da 21 domande aperte divise in tre sezioni principali: la prima sezione (intitolata "Who you are") richiede allo sviluppatore di fornire le seguenti informazioni:

1. Age;
2. Gender
3. Country
4. Profession
5. Education level
6. Computer knowledge
7. Other interests.

L'obiettivo di questa prima sezione dell'intervista è quello raccogliere informazioni socioculturali relative ai partecipanti al progetto per poter far luce sulla storia personale degli attori, elemento che concorre a definire il contesto in cui ha luogo la situazione di CMC.
La seconda sezione, intitolata "Plone & you", presenta le seguenti domande:

1. Since when have you been working on Plone?
2. Did you need help to start working on Plone?
3. For what reason did you decide to join the Plone project?
4. Are you joining (or did you join) other projects?
5. What type of activities do you do in Plone?
6. How much time do you dedicate to the project per week?
7. In which hours of the day do you work on Plone?
8. Which tools (CVS repository, documents, etc.) do you more often use?

L'obiettivo di questa sezione è quello tracciare un quadro del rapporto tra lo sviluppatore e il progetto. Le domande indagano sull'accesso dello sviluppatore al progetto (da quanto tempo partecipa e se ha avuto bisogno d'aiuto per iniziare a lavorare), sul motivo per cui ha scelto di partecipare al progetto, su quali attività svolge all'interno del gruppo, sull'utilizzo del tempo e sull'utilizzo dei tool informatici di sviluppo.
Le informazioni relative al campo motivazionale e all'uso del tempo sono particolarmente significative, se teniamo conto che, in generale, la maggior parte delle persone coinvolte lavorano gratuitamente ai progetti Free Software e che, durante la giornata, svolgono un'altra professione. Inoltre la domanda 5 vuole verificare l'esistenza di quei fenomeni di multiappartenenza molto importanti per lo scambio e la produzione della conoscenza.
La terza sezione dell'intervista, intitolata "Communication tools" pone le seguenti domande:

1. Are you in direct contact with other participants? If yes, with whom?
2. With what frequency do you communicate with other Plone patecipants?
3. What are the tools you use to communicate (email, chat, forum...)?
4. Did you already use these tools before you joined Plone?
5. Can you put these tools in order from the more effective to the less effective?
6. Do you use these tools in other contexts?

Gli obiettivi di questa sezione sono, in primo luogo, quello di verificare se esiste comunicazione diretta tra i partecipanti al progetto, e, in caso affermativo, con quale frequenza avviene. La questione della comunicazione diretta può sembrare banale (sembra ovvio che per lavorare allo stesso progetto ci si debba parlare). In realtà esistono forme alternative di comunicazione, come ad esempio, attraverso l'uso di documenti stesi per supportare il lavoro dello sviluppatore, modalità che ho avuto modo di sperimentare durante la mia partecipazione a Php-ZeroNet. Altro obiettivo del gruppo di domande è quello di illustrare quali sono gli artefatti tecnologici utilizzati per comunicare e verificare se tali artefatti sono utilizzati esclusivamente nel contesto del progetto o se vengono utilizzati anche in situazioni esterne alla collaborazione a Plone.
Ho contattato via email uno dei responsabili di Plone, chiedendo il permesso di sottoporre l'intervista ai partecipanti al progetto. Una volta ottenuto il suo benestare ho contattato, sempre utilizzando la posta elettronica, gli sviluppatori, chiedendo ad ognuno di loro se era disponibile a rispondere ad un breve questionario. Dei 77 sviluppatori contattati 44 si sono dichiarati disponibili.
Dopo aver definito la struttura e le domande dell'intervista si è posto il problema di come somministrarla agli sviluppatori, data l'impossibilità di contattarli di persona o telefonicamente. Le possibilità che ho vagliato sono state inizialmente due. La prima, cioè l'invio dell'intervista via email, è stata scartata per la scarsa “autorevolezza” del mezzo (arrivano ormai decine di mail al giorno, molte delle quali sono cestinate automaticamente dai programmi definiti “anti-spam”). L'altro mezzo vagliato, ma anch'esso scartato, è stato l'utilizzo di un tool di comunicazione sincrono (come IRC o ICQ). L'evidente vantaggio di un approccio di questo tipo è quello di somministrare le domande in tempo reale, avvicinandosi così alla situazione face to face, e avendo quindi la possibilità di approfondire gli argomenti più interessanti che possono emergere nel corso dell'intervista. Purtroppo la somministrazione delle domande via mezzo sincrono richiede, ovviamente, di concordare un momento in cui intervistatore e intervistato si incontrino in rete; questo risulta essere un impegno eccessivo per lo sviluppatore di Free Software, che solitamente è molto impegnato, dividendosi tra lavoro e parecipazione ai progetti della comunità.
La soluzione adottata è stata la realizzazione di un software che, accessibile via internet utilzzando un normale browser, propone le domande all'intervistato, permette di scrivere le risposte in aree apposite, e archivia le informazioni in un database opportunamente strutturato. Il primo vantaggio, dal punto di vista dell'intervistatore, è quello di avere a disposizione le risposte in un formato oridinato e facilmente consultabile. Inoltre l'applicazione è costantemente disponibile e l'intervistato può rispondere in qualsiasi momento della giornata.