UNIVERSITÀ DEGLI STUDI DI TRENTO
Scuola di Dottorato in Sociologia e Ricerca Sociale
Indirizzo *Information Systems and Organizations*
XXV Ciclo

La partecipazione degli utenti nella configurazione del Software Libero
Il caso di *Battle for Wesnoth*

PROGETTO DI RICERCA

Giacomo Poderi
giacomo.poderi@soc.unitn.it
Settembre 2010

Indice

Sommario

Questa ricerca si pone l'obiettivo di approfondire il tema della partecipazione dell'utente all'interno del Software Libero (anche "FOSS"): un paradigma di sviluppo software che si caratterizza per la sua natura di 'progettazione dal basso', distribuita, collaborativa e mediata da Internet. I numerosi studi esistenti sul fenomeno hanno evidenziato come l'apertura alla partecipazione e collaborazione da parte degli utenti sia, di fatto, una caratteristica chiave per questo paradigma. Tuttavia, questo tema è, paradossalmente, trattato solo in maniera parziale all'interno di questi studi. Infatti, l'interesse specifico si focalizza spesso solo su quegli utenti che partecipano ad attività di sviluppo e mantenimento del codice sorgente, tralasciando i numerosi altri modi in cui è possibile partecipare. Inoltre, gli utenti sono studiati nel loro ruolo di 'potenziali sviluppatori', ma quasi mai in qualità di utilizzatori.

A partire da una concettualizzazione dei progetti FOSS come sistemi socio-tecnici, e studiando come gli attori coinvolti in questi sistemi siano continuamente impegnati in attività di configurazione della tecnologia (i programmi sviluppati), questa ricerca mira a fornire un'analisi più approfondita e comprensiva della partecipazione degli utenti.

La metodologia utilizzata sarà quella dello studio di caso che prenderà in esamina il progetto di sviluppo del videogioco FOSS *Battle for Wesnoth*. Questo studio prevede una permanenza sul campo di circa dodici mesi, e si avvarrà di tecniche di ricerca empirica tipiche degli studi cyber-etnografici quali l'osservazione partecipante di gruppi on-line e le interviste mediate dal computer.

Parole chiave: Free and Open Source Software, utente, partecipazione, pratiche, sviluppo software, cyber-etnografia

1 Introduzione

L'interesse formulato in questa ricerca trae le sue origini dalla mia esperienza personale di utente di Software Liberi. Infatti, a partire dall'uso quotidiano di questo tipo di software, per compiere le attività più disparate e comuni, mi sono reso conto che l'idea di 'utente' implicata da questo paradigma è in contrasto con quella del tradizionale 'consumatore finale del prodotto'. Dall'immagine, stereotipata, di persona che utilizza un oggetto come fosse una *black-box* utile a portare a termine specifiche attività, ho realizzato di essere molto più interno a questa *black-box* di ciò che mi sarei potuto immaginare.

Da anni utilizzo un sistema operativo basato su Software Libero (detto anche, *Free and Open Source Software* "FOSS"[1]), ma allo stesso modo, passo del tempo sui forum di supporto on-line ad aiutare i nuovi utenti del sistema operativo. Ogni volta che riscontro un *bug* in una delle applicazioni che uso, lo segnalo nei giusti canali in modo che possa essere preso in considerazione dagli sviluppatori e corretto; per alcuni programmi, ho anche contribuito a migliorare la manualistica d'uso. I miei contributi, che nello sviluppo software tradizionale sarebbero responsabilità di figure professionali inquadrate in un contesto aziendale, sono possibili ed auspicabili proprio perché il FOSS integra profondamente la partecipazione dell'utente nelle attività di sviluppo, mantenimento e diffusione (von Hippel, 2001; Iivari, 2009b). Stimolato da questa situazione, ho iniziato ad interessarmi in maniera più approfondita al fenomeno con particolare interesse al tema dell'utente.

Un'indagine preliminare sulle attuali direzioni di ricerca sul FOSS e sui vari approcci allo sviluppo di sistemi software, ha evidenziato alcuni aspetti d'interesse per questo progetto di ricerca. Da decadi, il tema degli utenti nelle varie dimensioni del coinvolgimento, rappresentanza, partecipazione, *empowerment* e usabilità, è ampiamente trattato nei campi dell'informatica, dei sistemi informativi, e dell'ingegneria del software. Paradossalmente, questi temi sono trattati in maniera parziale all'interno degli studi sul FOSS, nonostante questi sottolineino spesso come l'apertura alla partecipazione degli utenti e la collaborazione tra i diversi attori siano degli elementi chiave (Koch, 2004; Weber, 2004). Infatti, tale partecipazione è studiata principalmente in relazione al contributo al codice sorgente – sviluppo del software, in senso stretto – ma raramente è inquadrata nel più ampio contesto di uno sviluppo distribuito in cui convivono molte attività, diversi modi di partecipare e molti ruoli. Inoltre, gli utenti sono visti nel loro ruolo di 'potenziali sviluppatori', quasi mai in qualità di 'utilizzatori del software'. Questi approcci hanno certamente contribuito a formare un'importante base di sapere su alcune dinamiche interne al FOSS, ma non sono sufficienti a fornire un quadro esauriente del fenomeno.

[1] Brevi cenni sulla questione terminologica sono forniti in Sez. 3.1.

[1] Una panoramica delle principali direzioni di ricerca si può trovare in Scacchi (2007); Muffatto (2006).

Al fine di aggiungere al dibattito accademico la voce di un gruppo di attori che è, a mio avviso, tanto fondamentale quanto trascurato, in questa ricerca mi pongo l'obiettivo di problematizzare la figura dell'utente e di esaminarne il ruolo in relazione al processo di sviluppo. Per raggiungere tale obiettivo ho deciso di utilizzare la metodologia dello studio di caso, con cui analizzerò il progetto di sviluppo del videogioco FOSS *Battle for Wesnoth*. Le tecniche cyber-etnografiche dell'osservazione partecipante di gruppi on-line e le interviste mediate dal computer saranno i principali strumenti per l'indagine empirica.

Nelle sezioni che seguono illustrerò meglio l'inquadramento teorico della ricerca, l'oggetto dell'indagine ed il suo disegno empirico.

2 Inquadramento teorico

2.1 Il nesso utenti–sviluppatori

Da circa tre decadi, la figura dell'utente quale 'persona che usufruisce di un bene prodotto da altri'[2] è sotto revisione critica in molte discipline che si occupano di progettazione, sviluppo, mantenimento e diffusione della tecnologia. In particolare, tutti gli approcci alla progettazione e sviluppo software riconoscono il valore che gli utenti rappresentano sin dalle primissime fasi produttive. (Cavaye, 1995; Damodaran, 1996)

La situazione informatica rispecchia un andamento generale più ampio che coinvolge tutto il panorama dello sviluppo e produzione tecnologico contemporaneo e che è andato diffondendosi sin da fine anni '70. In particolare, la riflessione critica al determinismo tecnologico portata dagli *Science and Technology Studies* (STS) ha contribuito largamente all'affermazione di una visione della tecnologia come un fenomeno molto più dipendente e legato alla società di ciò che era precedentemente immaginato. Le reti di relazioni, interessi e interazioni esistenti intorno ad una determinata tecnologia la influenzano e, allo stesso tempo, ne sono influenzati. Comprendere le implicazioni dello sviluppo e diffusione tecnologico significa studiare questa *seamless web* nel suo formarsi e modificarsi proprio in relazione alla tecnologia (Bijker et al., 1987; Bijker & Law, 1994).

A seguito di questo dibattito, il gruppo degli *utenti* inizia ad essere preso in considerazione ed aggiunto alla 'mappa dell'innovazione tecnologica' in qualità di gruppo rilevante: non più consumatori finali, passivi ed esterni alla produzione della tecnologia, ma dei veri e propri agenti che operano sia durante le fasi di sviluppo sia in quelle di utilizzo dell'artefatto tecnologico. Essi possono appropriarsi della tecnologia addomesticandola, imponendole usi inaspettati, reinventandola, ma anche rifiutandola a seconda di quanto questa tecnologia rispecchi le loro aspettative e necessità d'utilizzo

[2] Dizionario Treccani, voce "utente". http://www.treccani.it.

(Oudshoorn & Pinch, 2003). Diventa quindi prioritario riuscire ad anticipare questi *scenari d'uso*, e per fare ciò gli utenti – le loro aspettative – devono essere considerati (direttamente o indirettamente) già durante le fasi di progettazione e di sviluppo (Akrich, 1994).

Oggi, il tema dell'integrazione del punto di vista degli utenti è diventato un importante oggetto di ricerca in diversi approcci e discipline che s'interessano di sviluppo ed innovazione tecnologica. Ad esempio, approcci come il *Participatory design* (PD) e lo *User-centered design* (UCD), la *user-innovation research* e l'area della *Human-computer Interaction* (HCI), rispecchiano tutti l'interesse verso questo tema (Damodaran, 1996; Iivari, 2006; Pollock & Williams, 2008). Ognuno di questi campi ha il proprio interesse specifico, impostazioni tipiche delle indagini, e metodi adottati per la raccolta e analisi dei dati. Alcuni si concentrano sul coinvolgimento dell'utente nella fasi di progettazione e sviluppo; altri sulle relazioni tra l'interfaccia dello strumento e l'utente; altri ancora sui modi di appropriazione della tecnologia da parte degli utenti o sullo sviluppo successivo al lancio della tecnologia.

Tuttavia, questi campi di ricerca non forniscono un quadro coerente e compatibile che spieghi il legame esistente tra sviluppatori e utenti nelle varie fasi di produzione e diffusione della tecnologia. Come sostenuto in (Hyysalo, 2010, p. xxiii–xxv), questo sarebbe dovuto a due aspetti: (*i*) la maggior parte delle ricerche sono studi di casi che non riescono a fornire una panoramica generale sulla relazione sviluppo/utenza, mentre i pochi studi di respiro più ampio affrontano il tema ad un livello troppo generale senza riuscire ad essere informativi su questa relazione; (*ii*) la maggioranza di questi studi si concentrano o sullo sviluppo o sull'utilizzo della tecnologia. Nel primo caso non si segue la tecnologia in contesti di utilizzo reale, mentre nel secondo, tutto il processo di progettazione e sviluppo viene dato per scontato.

Con particolare riferimento a quest'ultima limitazione, il presente progetto di ricerca si avvicinerà allo sviluppo FOSS senza separare aprioristicamente lo sviluppo dall'uso e gli sviluppatori dagli utenti.

2.1.1 Oltre la dicotomia sviluppo/utilizzo: configurazioni della tecnologia

Nell'interessarsi alla relazione tra utenti e sviluppatori nel paradigma FOSS emerge subito una problematica molto meno pronunciata nello sviluppo software tradizionale: non esiste una demarcazione normata a priori tra gli sviluppatori e gli utenti. Non ci sono contratti aziendali ad inquadrare gli sviluppatori, né uffici vendite a puntare gli utenti. Come cercherò di evidenziare nella Sez. 3, le barriere d'ingresso alla partecipazione nello sviluppo FOSS sono quasi inesistenti[3]. Ogni utente è potenzialmente uno sviluppatore, e infatti molti sviluppatori sono utenti dello stesso software che contribuiscono a

[3]Nella maggioranza dei casi bastano un accesso ad Internet, comprendere la lingua, rispettare le norme del gruppo.

sviluppare. Da situazioni legate all'utilizzo del software molti utenti si ritrovano coinvolti o legati a contesti di sviluppo dello stesso.

Per questo motivo, si è deciso di vedere la tecnologia come facente parte di un insieme socio-tecnico i cui attori e attrici utilizzano e sviluppano contemporaneamente l'artefatto tecnologico (Bijker et al., 1987; Bijker, 1995). Il concetto di *configurazione* è utile qui per capire in che modo la tecnologia muta nel tempo.

Inizialmente utilizzato da Grint & Steve Woolgar (1997) per descrivere come dei progettisti di computer modellavano ed immaginavano il prodotto alla luce dei potenziali futuri utenti, l'idea del *configuring the users* è diventata molto utilizzata negli STS. Semplificando: durante la fase di produzione tecnologica, l'*utente* viene immaginato – configurato – dagli addetti alla progettazione e sviluppo, e la tecnologia viene prodotta in luce di quel futuro utente. Successivamente, la tecnologia entra in scenari d'uso reali e questo permette di capire se e come la 'configurazione' usata durante la produzione si conformi effettivamente alle aspettative dell'utente reale.

Il dibattito sollevatosi attorno a questa concettualizzazione del cambiamento tecnologico, ha arricchito la formulazione originale della *configurazione dell'utente*, che trattava troppo superficialmente alcuni aspetti. Infatti, raramente una tecnologia è rivolta ad un solo tipo di utente, più spesso il gruppo di utenti di riferimento è eterogeneo[4], quindi la configurazione *one size fits all* è problematica e diventa più appropriato parlare di configurazion*i* e dei processi di negoziazione tra di esse (Oudshoorn et al., 2004; Rohracher, 2005). Inoltre, come dimostrato in Oudshoorn & Pinch (2003) non esiste un modo uniforme in cui gli utenti reagiscono in relazione all'uso della tecnologia, diventa cruciale quindi trattare non superficialmente le loro reazioni. Infine, l'attività di configurazione è un'attività situata che si colloca in un preciso contesto storico-tecnologico-culturale. Così come i progettisti formano la propria visione sugli utenti e la tecnologia a partire da esperienze passate e dai contesti in cui lavorano, anche gli utenti possono formarsi le proprie aspettative su nuovi prodotti tecnologici già prima di iniziare ad utilizzarli: anche gli utenti configurano i produttori e l'artefatto tecnologico prima di entrarne in possesso ed utilizzarlo (Mackay et al., 2000).

Nel tentativo di studiare lo sviluppo FOSS nel modo più completo possibile, e quindi includendo anche il tema dell'utilizzo e degli utenti, questo verrà concettualizzato come un fenomeno di sviluppo socio-tecnologico, e si prenderà come inquadramento teorico sensibilizzante l'idea di configurazione della tecnologia come delineata in (Hyysalo, 2010, p.246–249). Composta cioè dai seguenti modi di configurazione:

Pre-configurare. Il tentativo di materializzare le proprie aspettative nella tecnologia, mentre questa è ancora in sviluppo o nel primissimo periodo di diffusione. Nel caso degli sviluppatori, pre-configurare si avvicina al significato originale di

[4] Adolescenti, giovani, anziani, uomini, donne, tecnicamente esperti o novizi...

"configurare" dato da Grint & Steve Woolgar (1997), per gli utenti si concretizza nell'aspettativa di poter applicare a quella tecnologia certe pratiche, usi, o convenzioni familiari.

De-configurare. Quando l'artefatto si sposta dalla produzione all'utilizzo, vengono messe alla prova sia la tecnologia sia le pratiche d'uso sia le convenzioni che ad esso si legano. Qui, le pre-configurazioni esistenti possono essere contestate, de-costruite: il rendersi conto che non corrispondono alle pratiche ed aspettative reali.

Ri-configurare. Strettamente legato alla de-configurazione, vi è anche un processo di riconfigurazione: il tentativo di ridare senso, riallineare e modificare le convenzioni, abitudini e pratiche d'uso in relazione all'artefatto (e all'insieme socio-tecnico a cui appartiene).

Co-configurare. Il processo di aggiornare o modificare la tecnologia attraverso una stretta collaborazione tra utenti (o loro 'rappresentanti') e sviluppatori, alla luce delle reazioni precedenti.

2.2 Sviluppo software e sviluppo FOSS

Sebbene in questa ricerca si voglia evitare un'analisi incentrata esclusivamente sullo *sviluppo* del software, in senso stretto, una panoramica sulle pratiche, i modelli, i princípi e la procedure che caratterizzano questa attività è necessaria per poter contestualizzare il FOSS.

Generalmente, con il termine "sviluppo software" ci si riferisce allo sviluppo di un prodotto software tramite l'uso di processi pianificati e strutturati. Oggigiorno, esistono numerosi approcci o modelli[5] per pianificare ed implementare questi processi, sebbene l'obiettivo resti sempre lo stesso: produrre un software che incontri e soddisfi le necessità di specifici committenti, quelle generali di potenziali utenti, o anche necessità personali. Ognuno di questi approcci ha i propri vantaggi in relazione alla tipologia di prodotto che deve essere creato, ma nessuno è privo di problematiche. Sviluppare software è un'attività intrinsecamente complessa: il suo prodotto è un oggetto non fisico che esiste in forma digitale; che deve costantemente interagire con i bisogni degli utenti e altri componenti (sia *hardware* sia *software*); che evolve e si rinnova a velocità molto sostenute; ed infine, che deve essere *user-friendly*[6] ed efficiente, per essere competitivo oggi (Quintas, 1994).

Uno tra i primi e più diffusi approcci allo sviluppo software è quello a cascata (*waterfall model*), dove sono previste delle fasi sequenziali, ognuna delle quali ha un

[5] Per una panoramica generale sull'ingegneria del software e sui maggiori modelli di sviluppo si veda (Sommerville, 1995).

[6] Il grado di facilità e soddisfazione con cui l'interazione uomo-strumento si compie.

obiettivo e delle procedure specifici. Si passa a una fase successiva nel momento in cui quella precedente è stata correttamente ultimata. Le fasi tipiche sono: *(i)* identificazione dei requisiti del prodotto finale; *(ii)* progettazione formale del sistema; *(iii)* scrittura del codice (o implementazione); *(iv)* integrazione e consolidamento delle varie componenti del sistema; *(v)* test e verifica del software; *(vi)* installazione ed uso; *(vii)* mantenimento e manutenzione del software (o suo abbandono) (Royce, 1970). Tuttavia, la rigidità dovuta alla sequenzialità delle fasi e la dipendenza dai requisiti che vengono definiti all'inizio del ciclo, pongono un serio limite al modello che oggi tende ad essere meno utilizzato in favore di approcci più flessibili. Infatti, tutto il processo si basa sull'assunto che i requisiti siano noti o facilmente identificabili, e che rimangano costanti. Questo non è quasi mai il caso.

Per tentare di superare questi problemi sono stati formalizzati numerosi altri approcci. I più noti sono: lo sviluppo iterativo (o agile), il modello evolutivo, quello a prototipi, il modello a spirale, l'*extreme programming* (XP), e il *rapid application development* (RAD). (Sommerville, 1995; Beck, 2000; Larman & Basili, 2003)

Di particolare interesse per questa ricerca sono il modello evolutivo e quello a prototipi. Il primo è un miglioramento del modello iterativo. Nel modello iterativo il prodotto viene scomposto in parti. Ognuna di queste viene sviluppata all'interno di un ciclo (uguale a quello del modello a cascata) e aggiunta a quella precedentemente ultimata. I requisiti in questo modello vengono definiti in maniera vaga per permettere più flessibilità. Il valore aggiunto del modello evolutivo su quello iterativo è quello di introdurre il concetto di sviluppo parallelo. Le varie componenti non vengono sviluppate in sequenza, ma parallelamente ed autonomamente. Questo è possibile grazie ad una progettazione *modulare* del software. Ogni modulo è, dal punto di vista dello sviluppo, indipendente dagli altri.

Con il modello a prototipi si cerca di risolvere il problema della definizione dei requisiti tramite la diffusione di versioni semplificate del software ad un gruppo selezionato di utenti, così da ricevere dei *feedback*, prima che il prodotto sia ultimato. Anche in questo approccio è cruciale l'idea dello sviluppo parallelo: vengono sviluppate contemporaneamente sia la versione prototipo, sia quella 'ufficiale'. I commenti e le impressioni ricavate dal prototipo vengono utilizzati per migliorare sia la successiva versione del prototipo sia il prodotto ufficiale. (Muffatto, 2006, Cap.4)

Il FOSS viene ancora considerato un modello di sviluppo a sé stante sebbene siano identificabili dei tratti comuni con alcuni di questi approcci, e in particolare degli ultimi due. Uno dei motivi principali per questa differenziazione si basa sulla diversa visione dei diritti di proprietà del software: un'idea di 'privato-collettivo' (Lee & Cole, 2003; von Hippel & von Krogh, 2003) che dà origine ad un nuovo insieme di relazioni tra sviluppatori, utenti, aziende software, ed altri attori. Tratto comune di queste relazioni è il loro carattere *emergente*, in contrasto alla natura strutturata delle tradizionali relazioni di produzione software. Lo sviluppo FOSS resta uno sviluppo dal basso. Ogni progetto

è un'entità a sé che si adatta solo in parte ai princípi tradizionali dell'ingegneria del software: alcuni progetti sono più vicini al modello iterativo, altri a quello evolutivo, altri ancora presentano caratteristiche tipiche di più modelli di sviluppo. Tuttavia, alcuni aspetti ricorrenti che stanno alla base dello sviluppo del FOSS possono essere identificati: (*i*) il software viene rilasciato frequentemente anche se imperfetto, in modo da stimolare feedback ed integrarli in maniera incrementale (Raymond, 1999); (*ii*) è costantemente in 'versione beta', nessuna versione è considerata come finale, tutte sono migliorabili (Weber, 2004, p.78), infatti l'interruzione dello sviluppo del software implica la dissoluzione della comunità che se ne prendeva cura (Kelty, 2008); (*iii*) è altamente modulare e fa affidamento sull'utilizzo di rami di sviluppo paralleli (Kogut & Metiu, 2001; González-Barahona et al., 2004).

2.3 La prospettiva delle pratiche

La difficoltà nel definire in maniera precisa e univoca le fasi di sviluppo FOSS, il concetto stesso di utente, ed il modo in cui questi si relazionano a vicenda, fa sorgere la necessità di avvicinarsi al fenomeno d'interesse nel modo meno prescrittivo possibile ed il più possibile aderente alla quotidianità di ciò che gli attori fanno. Per questo motivo, una prospettiva basata sulle pratiche risulta particolarmente adatta a questa ricerca.

Gli studi basati sulle pratiche vantano una lunga storia d'impiego in ambiti diversi ed hanno per questo assunto la forma di un 'carrozzone' in cui convivono diversi modi di essere intesi. È quindi necessario chiarire un punto che viene spesso trascurato: la *pratica* è sia una prospettiva, un punto d'osservazione su un fenomeno, sia un 'oggetto', un fenomeno d'interesse per la ricerca. In quest'ultima accezione il termine non va confuso con quello di *routine* o di semplice *attività* (Corradi et al., 2008; Wenger, 1999). Per evidenziare meglio la peculiarità della pratica, Rouse (2001) introduce la dimensione sociale nella definizione del concetto, sostenendo che le pratiche sono "ways of doing things together" o, secondo una definizione più forte:

> Within a workspace, a stabilized way of doing things becomes a practice when it is institutionalized and made normatively accountable both for its practitioners and for those who view it from outside. (Gherardi, 2010, p.505)

In questa ricerca, la pratica è usata come prospettiva, una lente che porta l'attenzione su ciò che le persone fanno nelle loro attività situate (Orlikowski, 2000). Ciò significa spostarsi dallo studio di quello che le persone dovrebbero fare, o che ci si aspetta esse facciano, a ciò che le persone effettivamente fanno (Pickering, 1992). Inoltre, significa portare al centro dell'indagine anche il contesto in cui queste azioni si compiono. Ne segue che la 'lente' delle pratiche mette a fuoco gli aspetti *relazionali* dei fenomeni

d'interesse (Østerlund & Carlile, 2005): ciò che le persone fanno *in relazione al* e *nel* contesto in cui lo fanno.

Studiare gli utenti e le attività in cui sono coinvolti attraverso la lente delle pratiche significa concentrarsi su: (*i*) le regolarità, gli schemi e le conoscenze più o meno condivise che caratterizzano e organizzano le attività: pratiche dall'esterno; (*ii*) l'emergere e la negoziazione dell'ordine delle azioni che vengono svolte, il modo in cui un'azione collettiva crea relazioni e legami tra le risorse disponibili e gli ostacoli presenti: pratiche dall'interno; (*iii*) gli effetti sociali che il fenomeno osservato ha in relazione all'essere praticato all'interno di una società: pratiche come pratiche sociali. (Gherardi, 2010)

Come sostenuto in (Sole & Edmondson, 2002; Orlikowski, 2000), questa prospettiva rappresenta un utile strumento per studiare sia l'uso situato di tecnologie informatiche sia il lavoro di gruppi distribuiti. Considerando che, nel FOSS, le attività di sviluppo e mantenimento del software sono portate avanti in maniera distribuita e mediata da specifiche tecnologie collaborative, gli elementi sopra menzionati risultano particolarmente adatti per questa ricerca.

3 Oggetto della ricerca

3.1 FOSS: cenni storici e fondamenti

Con il termine "Free and Open Source Software" si fa riferimento ad un fenomeno che inizia a svilupparsi nei primi anni '80, in opposizione alle pratiche di produzione e diffusione del software che si andavano diffondendo sempre più sin dagli anni '70. Propongo qui di seguito alcuni cenni storici per identificare le pietre miliari che caratterizzano, a livello macro, l'evoluzione di questo fenomeno.

La pubblicazione del *GNU Manifesto* ed il contemporaneo avvio del *GNU Project* nel 1983, la creazione della Free Software Foundation (FSF) nel 1985, ed il rilascio della prima versione della *GNU General Public Licence* (GNU GPL) nel 1989, sono gli avvenimenti principali che vengono oggi considerati come fondativi del *Free Software* (Software Libero), e che incarnano l'ambizioso progetto di creare un sistema operativo completamente libero. Il progetto, nato all'interno della cultura *hacker* dei laboratori accademici del Massachusetts Institute of Technology (MIT) e guidato principalmente da Richard Stallman, avrà notevole diffusione al di fuori dell'ambiente accademico grazie a Internet. (Williams, 2002)

Un contributo importante a questo progetto viene dato, a partire dal 1991 in poi, da *Linux*. Un kernel per sistemi operativi basato su UNIX, avviato e sviluppato come progetto personale da un informatico finlandese: Linus Torvalds. La combinazione di Linux con i componenti del progetto GNU[7] porta alla luce il sistema operativo libero

[7] Tra i quali mancava proprio un kernel.

iniziato nei primi anni '80. Si diffondono le "Distribuzioni GNU/Linux": assemblaggi di diversi componenti software che possono essere utilizzate come veri e propri sistemi operativi. (Torvalds & Diamond, 2001; Moody, 2002)

Licenze permissive, pubblico accesso, sviluppo distribuito e collaborativo sono i capisaldi che emergono in questi anni dal successo di GNU/Linux, e sono gli stessi che stanno alla base di altre due pietre miliari della storia del FOSS: *Apache* e *Mozilla Firefox*. Il primo è un web server nato da un progetto abbandonato[8] del *National Center for Supercomputing Applications* (NCSA) americano. Ripreso e riportato in vita da alcuni programmatori del progetto originale, l'Apache Web Server è oggi utilizzato su più del 50% dei server mondiali[9]. Il secondo è uno dei più grandi casi di conversione da software proprietario a libero, operata da una compagnia software di livello mondiale[10]. Il codice sorgente del *web browser* Netscape Navigator viene rilasciato sotto licenze libere per evolvere poi in *Firefox*: uno dei browser più utilizzati oggi. (DiBona et al., 1999)

A fine anni '90, un gruppo di attivisti e programmatori FOSS, tentano di portare questo modello di sviluppo più vicino alle logiche e alla terminologia *business oriented* con un'operazione di marketing. Tramite l'adozione del termine *Open Source*, la creazione della *Open Source Initiative* (OSI) e della *Open Source Definition*, si cerca di evidenziare le qualità tecniche del Free Software, mettendo in secondo piano quelle etico-filosofiche del termine "Free". Per questo motivo inizia a diffondersi la terminologia, ridondante, del "Free and Open Source Software"[11] ad indicare un insieme ben definito di pratiche, culture di sviluppo e di artefatti tecnologici e legali, indipendentemente dalla connotazione etica o pragmatica che si voglia dare al fenomeno. Questioni terminologiche a parte, resta ancora da chiarire che cosa si intenda per "Free and Open Source Software".

Un software è *libero*[12] – è FOSS – se e solo se garantisce i seguenti quattro diritti:

- Libertà di eseguire il programma per qualsiasi scopo
- Libertà di studiare il codice e modificarlo per le proprie esigenze
- Libertà di ridistribuire copie del software
- Libertà di ridistribuire le copie modificate

Lo strumento legale per garantire queste libertà è la licenza con cui il software viene distribuito[13] (De Paoli et al., 2008). Il requisito necessario (e implicazione dei termini

[8] Il nome del progetto era "NCSA HTTPd".

[9] Fonte Netcraft: http://news.netcraft.com/archives/web_server_survey.html

[10] *Netscape Communication Corporation*.

[11] In alcuni casi è utilizzato anche "Free/Libre/Open Source Software" (FLOSS).

[12] Si veda http://www.gnu.org/philosophy/free-sw.html per il manifesto contenente la definizione ufficiale, secondo la FSF.

[13] Oggi, ne esistono molte e di diverse ma la più conosciuta e diffusa resta ancora la GNU GPL. Per una lista delle licenze che garantiscono le quattro libertà si veda: http://www.gnu.org/licenses/license-list.html.

di licenza) per poter esercitare questi diritti è la disponibilità del codice sorgente del software.

Semplificando al limite, e lasciando da parte per un momento gli aspetti motivazionali, si potrebbe riassumere così la dinamica di base del FOSS: una persona inizia a scrivere un programma e lo diffonde, insieme ai termini di una licenza libera, in un posto pubblicamente accessibile. Tipicamente, raggiungibile via Internet. Altre persone notano questo programma e iniziano ad usarlo. Tramite l'uso, oltre ai punti forti dell'applicazione, si notano anche gli eventuali errori e limitazioni. Nel caso gli utenti abbiano la competenza tecnica per riparare gli errori del software (i *bug*) o per aggiungere le funzionalità mancanti, potrebbero farlo perché il codice sorgente è disponibile e la licenza lo permette. Nel caso non avessero le competenze informatiche adeguate, sarebbero comunque motivati a segnalare l'errore all'autore originale. Infatti, raramente lo sviluppatore è consapevole di tutti i bug presenti nel suo software e finché non ne viene a conoscenza non può tentare di porvi rimedio. Nel momento in cui il programma in questione non è più un piccolo progetto personale, ma qualche cosa di più ambizioso e complesso, allora aumentano i collaboratori, gli utenti, i feedback, i bug prodotti e quelli individuati. Nascono questioni di mantenimento, scalabilità, sostenibilità e innovazione del progetto.

Tuttavia, la possibilità di vedere, studiare e modificare il codice è solo una condizione necessaria affinché un progetto FOSS abbia successo. Questa non è sufficiente a convincere le persone a collaborare né a coordinarle in maniera efficace. Per questo motivo le dimensioni della partecipazione, collaborazione e coordinamento sono diventate tra le più studiate nel FOSS (Weber, 2004, Cap.3).

Partecipazione: Le motivazioni per cui sviluppatori e utenti dovrebbero devolvere volontariamente il proprio tempo in un progetto FOSS, sono diverse. Alcuni partecipano per ideologia: il FOSS incarna un modo di concepire il software eticamente o tecnicamente migliore di quello proprietario. Altri partecipano per soddisfare necessità personali come il creare un programma per risolvere un problema, correggere o segnalare errori che limitano l'utilizzo dei programmi. Altri ancora vedono, nella pratica dello sviluppo FOSS, una possibilità per migliorare il proprio valore sul mercato del lavoro. Infine, ci sono anche coloro che partecipano per divertimento, per esprimere la loro creatività, o per crearsi una reputazione all'interno della comunità. (Lerner & Tirole, 2002; Lakhani & Wolf, 2005; Shah, 2006; Ye & Kishida, 2003)

Collaborazione: All'interno di un progetto FOSS, la collaborazione è vitale. Pochi individui possono dedicarvisi a tempo pieno, quindi molti compiti vengono portati avanti da più persone. I progetti FOSS fanno uso di diversi strumenti tecnologici per facilitare la collaborazione. Gli strumenti di controllo di revisione (*revision control system*) permettono di collaborare a distanza sul software e di

tenere traccia della storia di tutti i cambiamenti ad esso fatto. Similmente, i *bug tracking system* sono sistemi con cui vengono tracciati tutti i bug individuati ed i progressi fatti per risolverli. Mailing-list, Internet forum e *Internet Relay Chat* (IRC) sono i principali strumenti di comunicazione che vengono utilizzati all'interno di un progetto FOSS. Infine, ci sono un serie di elementi normativi che vanno dai regolamenti ufficiali alle norme condivise che istituzionalizzano il modo appropriato di collaborare. (Crowston et al., 2006; D'Andrea et al., 2009)

Coordinamento: I due punti già esposti sono di aiuto ai meccanismi di coordinamento alla base del progetto, ma non bastano. In tutti i progetti FOSS vale un principio di auto-assegnazione dei compiti: chi partecipa è libero di scegliere il tipo e la misura del contributo che vuole dare (Crowston et al., 2007). Nell'economia del progetto è però importante che l'apporto dei contributi sia bilanciato. Un sistema di incentivi individuali ed un principio meritocratico favoriscono l'emergere di figure carismatiche e attendibili, che vanno ad occupare ruoli 'centrali' e ad occuparsi di compiti più sensibili. Insieme alla gerarchia di ruoli che si stabilisce all'interno di un progetto (e che evolve nel tempo), si diversificano anche le responsabilità, le competenze e il tipo di contributo dei partecipanti. (von Krogh et al., 2003; Crowston & Howison, 2005; van Wendel de Joode & de Bruijne, 2006)

3.2 FOSS: comunità dinamiche ed eterogenee

Comunemente si usa il termine "progetto FOSS" per indicare gli strumenti, gli obiettivi e la comunità di sviluppo di un determinato software. Ad esempio, con "progetto di Battle for Wesnoth" si fa riferimento a tutto l'apparato di sviluppo e mantenimento del videogioco *Battle for Wesnoth* (si veda la Sez.4.2 per maggiori dettagli).

Ogni progetto FOSS ha un proprio ciclo di vita che passa per diverse fasi: introduzione, crescita, maturità, declino o ripresa. Durante questo ciclo di vita, ciò che evolve non è solamente il software, ma anche la comunità che lo sviluppa, le loro pratiche collaborative e l'infrastruttura tecnologica utilizzata per contribuire. Considerando che i partecipanti sono dei volontari e che il flusso di persone che iniziano a contribuire e di quelle che smettono è sempre presente, la sostenibilità a lungo termine è un fattore importante che caratterizza il FOSS (Wynn, 2003; Lattemann & Stieglitz, 2005). Tramite un processo identificabile con la *partecipazione periferica legittima* (Lave & Wenger, 1991) le comunità FOSS riescono a garantirsi la crescita ed il rinnovamento dei membri partecipanti, così come il mantenimento e il rinnovamento delle pratiche collaborative. (Edwards, 2001; kun Huang & min Liu, 2005; Crowston et al., 2006)

Sebbene si parli spesso di sviluppo e di comunità FOSS in relazione allo sviluppo del software in senso stretto, i partecipanti a queste comunità, e le attività di cui si occupano, sono molto più eterogenee di quello che si potrebbe pensare. Scrivere e revisionare il codice sono solo alcune delle tante attività di cui si occupano i membri

delle comunità FOSS. Così come le competenze informatiche non sono le uniche ad essere messe a frutto. Il mantenimento dell'infrastruttura e degli strumenti con cui lavora la comunità, la scrittura della documentazione del software, la traduzione in altre lingue dell'interfaccia e della documentazione, il supporto ai nuovi utenti e, in molti casi, la creazione di contenuti accessori o multimediali, sono tutte attività delle quali si fa carico una comunità FOSS e che fanno parte integrante dello sviluppo di un progetto. (Ye & Kishida, 2003; Gläser, 2007; D'Andrea et al., 2009).

Come già menzionato in precedenza, la letteratura oggi esistente sul FOSS trascura queste attività 'periferiche' ed il loro ruolo all'interno del progetto, per concentrarsi sulle pratiche di sviluppo e mantenimento del codice. In questa ricerca intendo utilizzare un'accezione ampia del concetto di sviluppo del FOSS che includa anche queste attività e gli attori che vi prendono parte: principalmente gli utenti.

4 Disegno empirico della ricerca

4.1 Obiettivi specifici

L'obiettivo generale della ricerca può essere riassunto come l'interesse nell'approfondire il ruolo dell'utenza e dell'uso nello sviluppo di progetti FOSS. Una tematica largamente ignorata nei maggiori filoni di ricerca che si interessano al FOSS. Tali studi utilizzano principalmente degli approcci riduzionisti al fenomeno: i processi d'interesse sono quelli relativi all'attività dello sviluppo del codice, e solo gli attori in stretta relazione con questa attività vengono considerati come rilevanti[14]. In opposizione a questa tendenza, è stato evidenziato da alcuni studi quanto una maggiore attenzione sui temi relativi alle esigenze dell'utenza finale e all'usabilità del software sia di fatto auspicabile. Infatti, la teorica apertura al coinvolgimento di tutti si traduce spesso in una reale partecipazione di pochi, i quali contribuiscono mossi da motivazioni personali (Lakhani & Wolf, 2005) e che raramente mettono in cima alla lista delle priorità di sviluppo l'usabilità del software e la soddisfazione dell'utente finale (Benson et al., 2004; Morten Sieker Andreasen et al., 2006; Nichols & Twidale, 2006).

Partendo dai concetti esposti nella Sez. 2, si è deciso di inquadrare i progetti FOSS come degli insiemi socio-tecnici atti a produrre e mantenere degli artefatti – delle applicazioni software. Tali insiemi sono costantemente coinvolti in processi di (pre-,de-,ri-,co-)configurazione atti ad influenzare l'artefatto prodotto, alla luce delle esigenze dei vari attori coinvolti (e, potenzialmente, anche di quelli esclusi). In questo inquadramento, lo studio di *come gli utenti finali e l'uso del software si intreccino con lo sviluppo e gli sviluppatori* può essere scomposto nelle seguenti domande:

- Attraverso quali pratiche si realizza il processo di pre-configurazione del software? In che modo, cioè, gli sviluppatori raccolgono e usano informazioni riguardanti

[14] Per alcune eccezioni si vedano (Iivari, 2009a; Barcellini et al., 2008; Iivari, 2009b).

le necessità (o priorità) di sviluppo per le nuove versioni, e di contro, in che modo gli utenti si creano aspettative su queste nuove versioni?

- Quali sono i diversi modi in cui gli utenti reagiscono (de-/ri- configurano) alle caratteristiche del software che differiscono da quelle aspettate?
- Esistono delle pratiche di negoziazione (co-configurazione) tra utenti e sviluppatori riguardanti le future caratteristiche del software? Come si declinano queste pratiche?
- Quali attività (oltre allo sviluppo e mantenimento del codice) sono aperte alla partecipazione degli utenti? Come si realizza la partecipazione dei vari attori in queste attività? Ed infine, queste attività come si collegano alle esperienze d'uso del software e come al suo sviluppo?

Per rispondere alle domande qui proposte, la ricerca verrà effettuata attraverso lo studio di caso che prende come unità d'analisi il videogioco FOSS *Battle for Wesnoth*, e come unità di rilevazione principali gli sviluppatori e gli utenti di questo software.

4.2 Campo di ricerca

Partendo dagli argomenti già svolti e riflettendo sull'eterogeneità dei vari progetti FOSS oggi esistenti, ho deciso di orientarmi sullo studio di un unico caso FOSS, che presenti caratteristiche interessanti, piuttosto che tentare analisi comparative tra campioni più ampi.

Per scegliere il caso oggetto di studio ho proceduto nel seguente modo. A partire dalle due più grandi *FOSS forgery* oggi esistenti, ho selezionato 40 progetti FOSS per applicazioni Desktop[15] dalla lista di quelli attivi[16]. Con "FOSS forgery" ci si riferisce comunemente ad una piattaforma web per la gestione dello sviluppo di software di tipo collaborativo. La piattaforma fornisce tutti i principali strumenti necessari a sviluppare un software in maniera distribuita. Chi inizia a sviluppare un'applicazione FOSS, spesso si appoggia su questi servizi: così facendo, l'infrastruttura per lo sviluppo non deve essere gestita dagli sviluppatori. *SourceForge* e *Freshmeat*[17] sono le più diffuse, e quelle utilizzate per la scelta del caso.

Nell'arco di tre settimane, tramite una panoramica sullo stato di sviluppo del progetto, sulla 'vivacità' della comunità e delle varie modalità di partecipazione previste, sono giunto ad un campione di sette progetti. Ho condotto questa scrematura preliminare osservando i siti web ufficiali dei progetti, visitando i rispettivi on-line

[15] Quindi, progettati per un uso comune, piuttosto che *business oriented* o per la produzione professionale.

[16] In queste *forgery* è possibile filtrare l'elenco dei progetti attraverso vari attributi.

[17] Si vedano: http://sourceforge.net/ e http://freshmeat.net/.

forum e le sezioni *Contribute*[18]. Tra questi sette progetti quattro erano videogiochi, due degli editor audio/video e uno un piccolo programma gestionale. Dopo un'ulteriore esamina delle caratteristiche di questi progetti, i videogiochi sono risultati quelli potenzialmente più interessanti. Tra questi ho scelto *Battle for Wesnoth* (BfW): un gioco di strategia a turni ambientato in un mondo di fantasia (il regno di Wesnoth) e popolato da diverse razze (umani, elfi, orchi, nani).

Le caratteristiche che hanno orientato la 'scelta ragionata' (David Silverman, 2002, Cap.8) su *Battle for Wesnoth* possono essere così riassunte:

Progetto attivo Nato nel 2003, il progetto ha raggiunto in due anni la versione 1.0. Rilasciata nel 2005, questa versione ha dato fama e diffusione al gioco, che perdura fino ad oggi. In Maggio 2010 è stata raggiunta la versione attuale: 1.8.1. Mediamente, il progetto rilascia due nuove versioni stabili all'anno, senza contare le versioni di sviluppo. Queste informazioni sono indicatrici di un progetto FOSS attivo e che ha quindi una vivace e produttiva comunità (Crowston et al., 2003). Il *credit file* del progetto elenca più di 150 tra sviluppatori, traduttori e creatori di contenuti. Il forum ufficiale del progetto ha attualmente[19] 14125 utenti registrati.

Multi-piattaforma Il software è stato sviluppato per essere multi-piattaforma. Questo significa che la comunità può facilmente produrre e mantenere versioni del software che funzionino su tutti i maggiori sistemi operativi oggi esistenti. Oltre alle versioni per GNU/Linux[20], Windows, e Mac OS è disponibile anche una versione per Apple iPhone. Questo aspetto è importante perché mette una tecnologia FOSS alla portata di un'utenza solitamente meno vicina ai suoi ideali e al suo paradigma di sviluppo, allo stesso tempo porta BfW al di fuori del contesto d'utilizzo di un'applicazione Desktop. Per questa ricerca, l'eterogeneità degli ambiti d'uso a cui è esposto BfW lo rende un caso molto interessante.

Community-based Il progetto è interamente gestito a livello di comunità. A differenza di altre applicazioni FOSS di successo, BfW non presenta un modello ibrido, in cui attori *corporate* o compagnie software 'tradizionali' cercano di entrare nel ed interagire con lo sviluppo del progetto (Lin, 2006; Shah, 2006). Inoltre, tutto il software, inclusi i contenuti del gioco (grafiche, suoni, *storylines*...), sono rilasciati sotto un'unica licenza: la GNU GPL. Questo aspetto mi garantisce una uniformità di fondo rispetto alla produzione e distribuzione degli artefatti che non potrei avere nel caso venissero usate più licenze all'interno dello stesso progetto.

[18]Queste sezioni elencano i modi in cui è possibile contribuire al progetto. Tali informazioni possono essere raggruppate anche sotto un altro nome come *Join the community*, *Help us*...

[19]Cifra verificata in data: 25 maggio 2010. `http://forums.wesnoth.org/`.

[20]I sistemi operativi GNU/Linux sono il target principale per lo sviluppo di applicazioni Desktop FOSS.

Rilevanza dell'user-made-content Per come è progettato BfW, i contenuti creati dagli utenti assumono grande rilevanza. Le campagne di gioco (*campaigns*) con le quali ogni giocatore può cimentarsi, hanno una propria linea narrativa, dei personaggi, degli obiettivi e delle ambientazioni specifici. Per questo è fondamentale che le nuove versioni del gioco includano anche nuove campagne. Infatti, non ci sarebbe valore aggiunto nel ri-giocare vecchie storie, neanche se il software fosse più performante o con funzionalità aggiuntive. BfW permette di creare le proprie campagne e di condividerle con altri giocatori. Alcune di queste campagne sono liberamente scaricabili dall'interno del gioco, alla sezione *Add-ons*. Inoltre, quelle ritenute meritevoli dalla comunità, entrano a far parte delle campagne ufficiali di BfW. Questo aspetto è così importante per l'economia del progetto che è stato creato un progetto parallelo per permettere agli utenti di coordinarsi, sviluppare e mantenere i loro contenuti: il *Coordinated Wesnoth UMC Development project*[21] (detto anche "Wesnoth-UMC-Dev").

Utilizzo single- e multi-player BfW presenta due modalità di gioco: quella da giocatore singolo e quella da multi-giocatore. Nella prima, l'utente si cimenta con gli obiettivi forniti dalle campagne. Nella seconda, non ci sono obiettivi pre-definiti, ma più semplicemente, i giocatori si sfidano in partite *1-vs-1*, *2-vs-2* o *free for all* (FFA). Questo aspetto è molto interessante poiché le due modalità di gioco danno origine a dinamiche differenti. I limiti, le problematiche e le migliorie del gioco assumono significati diversi all'interno delle due modalità, e questo vale sia a livello di 'motore' del gioco (lo sviluppo del software) sia per quello 'contenutistico' (lo sviluppo di campagne, scenari e altri contenuti).

4.3 Metodologia

Le impostazioni teoriche e gli obiettivi fin qui enunciati indicano già abbastanza chiaramente la maggior adeguatezza di un approccio qualitativo rispetto ad uno quantitativo. La dimensione situata dei processi d'interesse richiede una raccolta ed un'analisi dei dati basata su una permanenza prolungata sul campo. Lo studio di caso (Walsham, 1995; Yin, 2003) di BfW verrà condotto attraverso tecniche *cyber-etnografiche* quali l'osservazione partecipante di gruppi on-line e le interviste mediate dal computer.

La scelta di utilizzare un approccio qualitativo si base sulle seguenti motivazioni. Ad oggi, le ricerche sul FOSS restano per la maggior parte quantitative. Le ricerche in profondità, sebbene in aumento, restano una minoranza (Stol & Babar, 2009). Per questo, delle interazioni tra individui e delle 'micro-dinamiche' conosciamo solo alcuni indicatori 'superficiali'. Non abbiamo dati empirici sulla complessità dei contesti, delle pratiche quotidiane, del bagaglio di conoscenze tacite e delle strategie d'azione nascoste che permeano queste comunità. Come riassunto chiaramente in (Fele, 2009)

[21] http://wesnoth-umc-dev.ai0867.net/.

queste conoscenze sono diventate sempre più interessanti per le scienze informatiche e dei sistemi informativi.

La particolarità dell'oggetto di ricerca rende comunque opportuna una riflessione sulle tecniche che verranno utilizzate. Infatti, sebbene i membri della comunità d'interesse collaborino ed interagiscano quotidianamente, essi lo fanno senza la necessità di doversi incontrare faccia a faccia, né di condividere lo stesso spazio fisico né lo stesso tempo. Le loro comunicazioni, le loro interazioni e le dinamiche oggetto di studio si concretizzano principalmente on-line, in uno spazio virtuale o, per meglio dire, in un *cyberspazio* (Escobar et al., 1994).

Primo e centrale problema dello studio di questo *cyberspazio* è proprio l'individuazione del campo e la definizione dei suoi confini. Basandomi su uno dei lavori pionieristici sull'etnografia di gruppi on-line (Hine, 2000) e su alcune riflessioni critiche ad esso parallele (Lyman & Wakeford, 1999; Green, 1999; Hakken, 1999; Teli et al., 2007), ho deciso di non definire a priori e rigidamente il campo di ricerca. In prima istanza individuerò gli *spokesperson* (portavoce) principali che mi indicheranno alcuni punti d'ingresso preferenziali e una prima idea dei confini del campo. Questi confini potrebbero comunque essere ridefiniti durante la ricerca nel caso ne sorgesse la necessità. Lo *spokesperson* è un'entità – non necessariamente umana – che è dotata di sufficiente autorità per parlare a nome degli individui che ci interessa osservare, e che può servire da prima approssimazione per la definizione del campo di ricerca (Latour, 2005). Nel mio caso, il sito ufficiale del progetto (`http://www.wesnoth.org/`) è usato quale indicatore più ampio per identificare la comunità di BfW. La raccolta dati si muoverà principalmente all'interno di questo sito web e nei canali ad esso collegati. Non escludo comunque che, sia all'interno sia all'esterno del sito possano essere scoperti nuovi portavoce che facciano emergere altre comunità specifiche. Ad esempio, un altro importante indicatore è il sito web di Wesnoth-UMC-Dev (`http://wesnoth-umc-dev.ai0867.net/`), che identifica e delinea i partecipanti allo sviluppo dei contenuti di BfW.

In linea con i fondamenti epistemologici della cyber-etnografia e con quanto appena esposto, non effettuerò scelte prescrittive riguardo la distinzione tra off-line/on-line per quel che riguarda l'osservazione sul campo. Ovviamente, considerata la natura distribuita della comunità d'interesse e il prevalente svolgimento delle interazioni attraverso la mediazione del computer, il principale dominio per la raccolta dati sarà quello on-line. Resta contemplata l'eventualità di poter condurre osservazione partecipante anche in contesti off-line, ad esempio, durante conferenze o workshop per progetti FOSS[22].

Prima di poter iniziare l'attività sul campo resta da definire le strategie di accesso ad esso. In generale, i canali di comunicazioni della comunità sono pubblicamente

[22]In passato alcuni rappresentanti della comunità BfW, prevalentemente sviluppatori, si sono incontrati al Free and Open Source Developers' European Meeting (FOSDEM) (`http://fosdem.org`), per socializzare, parlare di BfW e sviluppare nuove idee.

accessibili, così come sono accessibili gli archivi del materiale prodotto in questi anni, quindi un approccio basato su osservazione non intrusiva sarebbe possibile. Tuttavia, durante le fasi preliminari, mi sono reso conto che questo tipo di osservazione (Hilbert, 1980), non sarebbe adeguata a rilevare tutta la ricchezza delle interazioni tra membri della comunità. Per questo, ho deciso di utilizzare tecniche di osservazione partecipante (Waddington, 1994) per interagire con i membri della comunità e poter aver accesso ai significati che si producono durante le interazioni.

Come già menzionato, la partecipazione ai canali della comunità è generalmente aperta a tutti ed incoraggiata, indipendentemente dalle motivazioni personali (von Krogh et al., 2003; Lakhani & Wolf, 2005). Questo fa ritenere la negoziazione dell'accesso al campo relativamente meno problematica rispetto a quella di un'etnografia tradizionale (Cardano, 2003, Cap.4). Attualmente, si sono presi contatti con gli amministratori della principale mailing-list della comunità, mentre per gli altri canali (on-line forum, *chat room*) una 'negoziazione' in senso stretto non sembra necessaria, proprio per la natura aperta di questi spazi. Tuttavia, la questione di come suscitare l'interesse dei membri della comunità ad interagire con la figura del ricercatore, non può essere tralasciata. Per ovviare a questo problema, ho deciso di (ri)creare, attraverso l'implementazione di un semplice sito web, la mia identità di ricercatore (Turkle, 1997), e di illustrare gli obiettivi generali della ricerca attraverso queste pagine. Come già documentato in (Forte, 2004; Wakeford & Cohen, 2008), tale sito può rivelarsi utile in due modi: (*i*) facilitando la formazione di fiducia e chiarezza nella relazione tra ricercatore ed informatori, che avranno sempre un punto di riferimento al quale attingere; (*ii*) favorendo una co-costruzione della ricerca: la possibilità di commentare i contenuti pubblicati permetterà di migliorare ed integrare l'analisi con il punto di vista dei 'nativi'. Il sito sarà raggiungibile all'indirizzo `http://www.poderi.eu` ed inizierà a pubblicare contenuti relativi alla ricerca, contemporaneamente al primo periodo di lavoro sul campo.

Quando si utilizza l'osservazione partecipante per fenomeni che si realizzano on-line, è importante sottolineare che la natura di ciò che è osservabile e di ciò che non lo è differisce da quella dell'osservazione partecipante tradizionale. Non si osservano delle persone (nella loro fisicità), ma le interazioni comunicative che queste mantengono attraverso la mediazione della tecnologia. Di conseguenza, il principale tipo di dato raccolto sarà testuale: messaggi a mailing-list, conversazioni in *chat room*, e risposte a gruppi di discussione (Garcia et al., 2009).

Per quel che riguarda la permanenza sul campo, stimo un periodo di 12 mesi come adeguato a coprire tutte le fasi principali di un ciclo di rilascio del software. Questa possibilità mi sembra importante perché in fasi di sviluppo diverse si manifestano tendenze generali diverse. Ad esempio: in occasione del rilascio di nuove versioni l'arrivo di nuovi utenti è molto più consistente rispetto ad altri periodi.

Sebbene nel caso specifico di BfW, un ciclo di rilascio è più breve dei 12 mesi, una

permanenza sul campo di questa durata mi permetterebbe di avere una sovrapposizione rispetto al momento d'ingresso e di uscita dal campo, grazie alla ripetizione di un (nuovo) ciclo di rilascio. Soprattutto, questo arco temporale mi permette di essere flessibile rispetto ad eventuali ritardi che si potrebbero manifestare nel progetto e che non sono né prevedibili, né completamente inusuali (Michlmayr et al., 2007).

Per quel che riguarda il lavoro nei dodici mesi pianificati, va detto che una definizione dettagliata della tempistica risulta complicata e potenzialmente fuorviante proprio per l'utilizzo di tecniche etnografiche. Premesso ciò, è comunque possibile descrivere a grandi linee le fasi principali della ricerca e i loro obiettivi. In una prima fase di due mesi circa, cercherò di individuare le attività alle quali è possibile contribuire, i vari gruppi o sottogruppi ad esse collegati, gli attori principali, i potenziali soggetti da intervistare, le norme e le tecnologie che caratterizzano ognuna di queste. Contemporaneamente, familiarizzerò con il funzionamento e l'utilizzo di BfW. La mia partecipazione sul campo durante questa fase sarà minima. Utilizzerò la fase centrale della ricerca empirica, per intervistare gli attori principali precedentemente individuati, osservare e seguire le pratiche di cui si occupano, e partecipare nella vita attiva della comunità. Durante questa fase, i periodi di raccolta dati si alterneranno a momenti (relativamente brevi) dedicati alla loro analisi. Ogni momento di analisi sarà anche occasione per aggiornare i contenuti del sito web e per meglio impostare nuove direzioni specifiche di raccolta dati. Stimo la durata di questa fase di circa otto mesi. Nell'ultima fase, inizierò gradualmente a passare più tempo sull'interpretazione dei dati, alla loro pulitura ed eventuale integrazione. Questa fase non dovrebbe superare i tre mesi. In termini di date, questa è la suddivisione del periodo della ricerca empirica

Fase iniziale ottobre 2010 – dicembre 2010

Fase centrale dicembre 2010 – luglio 2011

Fase conclusiva luglio 2011 – ottobre 2011

Dopo aver completato l'ultima fase e durante l'anno successivo, procederò con la riorganizzazione del materiale raccolto ed analizzato, al confronto della letteratura con ciò che è emerso dal campo e alla stesura della tesi di dottorato.

Riferimenti bibliografici

Akrich, M. (1994). The de-scription of technological objects. In *Shaping Technology/Building Society. Studies in Sociotechnical change*, (pp. 205–224). Cambridge: MIT Press.

Barcellini, F., Detienne, F., & Burkhardta, J.-M. (2008). User and developer mediation in an open source software community: Boundary spanning through cross participation in online discussions. *International Journal of Human–Computer Studies*, *66*, 558–570.

Beck, K. (2000). *Extreme programming explained*. XP Series. Reading, MA: Addison-Wesley.

Benson, C., Muller-Prove, M., & Mzourek, J. (2004). Professional usability in open source projects: GNOME, OpenOffice. org, NetBeans. In *Extended abstracts of CHI'04*, (pp. 1083–1084). New York, NY: ACM.

Bijker, W. E. (1995). *Of bicycles, bakelites, and bulbs: Toward a theory of sociotechnical change*. Cambridge, MA: MIT Press.

Bijker, W. E., Huges, T. P., & Pinch, T. (Eds.) (1987). *The Social Construction of Technological Systems. New Directions in the Sociology and History of Technology*. Cambridge, MA: MIT Press.

Bijker, W. E., & Law, J. (Eds.) (1994). *Shaping Technology/Building Society. Studies in Sociotechnical change*. Cambridge, Massachusetts: MIT Press.

Cardano, M. (2003). *Tecniche di ricerca qualitativa. Percorsi di ricerca nelle scienze sociali*. Roma: Carocci.

Cavaye, A. L. M. (1995). User participation in system development revisited. *Information & Management*, *28*(5), 311–323.

Corradi, G., Gherardi, S., & Verzelloni, L. (2008). Ten good reasons for assuming a 'practice lens' in organization studies. In *International Conference on Organizational Learning, Knowledge and Capabilities*. Copenhagen: OLKC.

Crowston, K., Annabi, H., & Howison, J. (2003). Defining open source software project success. In *Proceedings of the 24th international conference on information systems (icis 2003)*, (p. 327–340).

Crowston, K., & Howison, J. (2005). The social structure of free and open source software development. *First Monday*, *10*(2).

Crowston, K., Li, Q., Wei, K., Eseryel, U. Y., & Howison, J. (2007). Self-organization of teams for free/libre open source software development. *Information and Software Technology*, *49*, 564–575.

Crowston, K., Wei, K., Li, Q., & Howison, J. (2006). Core and periphery in free/libre and open source software team communications. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS 06)*. IEEE Computer Society.

Damodaran, L. (1996). User involvement in the systems design process – a practical guide for users. *Behaviour & Information Technology*, *15*(6), 363–377.

D'Andrea, V., De Paoli, S., & Teli, M. (2009). The construction of peers and artifacts: The organizing role of Programming guidelines. In *Hawaii International Conference on System Sciences*, vol. 0, (pp. 1–9). Los Alamitos, CA, USA: IEEE Computer Society.

David Silverman, D. (2002). *Come Fare Ricerca Qualitativa*. Scienze politiche e sociali. Roma: Carocci.

De Paoli, S., Teli, M., & D'Andrea, V. (2008). Free and open source licenses in community life: Two empirical cases. *First Monday*, *13*(10).

DiBona, C., Ockman, S., & Stone, M. (Eds.) (1999). *Open sources : voices from the open source revolution*. Sebastopol, CA: O'Reilly Media.

Edwards, K. (2001). Epistemic communities, situated learning and open source software development. In *'Epistemic Cultures and the Practice of Interdisciplinarity' Workshop at NTNU*.

Escobar, A., Hess, D., Licha, I., Sibley, W., Strathern, M., & Sutz, J. (1994). Welcome to cyberia: Notes on the anthropology of cyberculture [and comments and reply]. *Current Anthropology*, *35*(3), 211–231.

Fele, G. (2009). Perché l'informatica si interessa all'etnografia? Appunti di una storia in corso. *Etnografia e Ricerca Qualitativa*, *1*, 49–76.

Forte, M. (2004). Co-construction and field creation: Website development as both an instrument and relationship in action research. In E. A. Buchanan (Ed.) *Readings in Virtual Research Ethics: Issues and Controversies*, (pp. 219–245). Idea Group Inc (IGI).

Garcia, A. C., Standlee, A. I., Bechkoff, J., & Cui, Y. (2009). Ethnographic approaches to the internet and Computer-Mediated communication. *Journal of Contemporary Ethnography*, *38*(1), 52–84.

Gherardi, S. (2010). Telemedicine: A practice-based approach to technology. *Human Relations*, *63*(4), 501–524.

Gläser, J. (2007). The social order of open source software production. In *Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives*. Idea Group Pub.

González-Barahona, J. M., López, L., & Robles, G. (2004). Community structure of modules in the apache project. In *In Proceedings of the 4th Workshop on Open Source Software Engineering*, (pp. 307–317). Springer.

Green, N. (1999). Disrupting the field: Virtual reality technologies and Multisited ethnographic methods. *American Behavioral Scientist*, *43*(3), 409–421.

Grint, K., & Steve Woolgar, S. (1997). *The Machine at Work: Technology, Work and Organization*. Malden, MA: Polity Press.

Hakken, D. (1999). *Cyborgs@cyberspace?: An Ethnographer Looks to the Future*. London: Routledge Chapman & Hall.

Hilbert, R. A. (1980). Covert participant observation: On its nature and practice. *Journal of Contemporary Ethnography*, *9*(1), 51–78.

Hine, C. (2000). *Virtual Ethnography*. London: SAGE Publications.

Hyysalo, S. (2010). *Health Technology Development and Use – From Practice-Bound Imagination to Evolving Impacts*. Routledge studies in technology, work and organizations. London: Routledge.

Iivari, N. (2006). 'Representing the user' in software development—a cultural analysis of usability work in the product development context. *Interacting with Computers*, *18*(4), 635–664.

Iivari, N. (2009a). "Constructing the users" in open source software development – an interpretive case study of user participation. *Information Technology & People*, *22*(2), 132–156.

Iivari, N. (2009b). Empowering the users? a critical textual analysis of the role of users in open source software development. *AI & Society*, *23*(4), 511–528.

Kelty, C. M. (2008). *Two bits: the cultural significance of free software*. Duke University Press.

Koch, S. (2004). *Free/open source software development*. London: IGI Global.

Kogut, B., & Metiu, A. (2001). Open-source software developmente and distributed innovation. *Oxford Review of Economic Policy*, *17*(2), 248–264.

kun Huang, S., & min Liu, K. (2005). Mining version histories to verify the learning process of legitimate peripheral participants. In *in Proceedings of International Workshop on Mining Software Repositories (MSR'05) (St*, (pp. 1–5). ACM Press.

Lakhani, K. R., & Wolf, R. G. (2005). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. In J. Feller, B. Fitzgerald, S. Hissam, & K. R. Lakhani (Eds.) *Perspectives on Free and Open Source Software*. MIT Press.

Larman, C., & Basili, V. R. (2003). Iterative and incremental development: A brief history. *Computer*, *36*(6), 47–56.

Latour, B. (2005). *Reassembling the social. An introduction to Actor-Network-Theory*. Oxford University Press.

Lattemann, C., & Stieglitz, S. (2005). Framework for governance in open source communities. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, (p. 192a).

Lave, J., & Wenger, E. (1991). *Situated Learning*. Cambridge University Press.

Lee, G. K., & Cole, R. E. (2003). From a firm-based to a community-based model of knowledge creation: The case of the linux kernel development. *Organization Sicence*, *14*(6), 633–649.

Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *Journal of Industrial Economics*, *50*(2), 197–234.

Lin, Y. (2006). Hybrid innovation: The dynamics of collaboration between the FLOSS community and corporations. *Knowledge, Technology, and Policy*, *18*(4), 86–100.

Lyman, P., & Wakeford, N. (1999). Introduction: Going into the (Virtual) field. *American Behavioral Scientist*, *43*(3), 359–376.

Mackay, H., Carne, C., Beynon-Davies, P., & Tudhope, D. (2000). Reconfiguring the user: Using rapid application development. *Social Studies of Science*, *30*(5), 737–757.

Michlmayr, M., Hunt, F., & Probert, D. (2007). Release management in free software projects: Practices and problems. In *Open Source Development, Adoption and Innovation*, (pp. 295–300).

Moody, G. (2002). *Rebel Code: Linux and the Open Source Revolution*. London: Penguin.

Morten Sieker Andreasen, H. N., Schrøder, S., & Stage, J. (2006). Usability in open source software development: opinions and practice. *Information technology and control*, *25*(3A), 303–312.

Muffatto, M. (2006). *Open source: a multidisciplinary approach*. London: Imperial College Press.

Nichols, D. M., & Twidale, M. B. (2006). Usability processes in open source projects. *Software Process: Improvement and Practice*, *11*(2), 149–162.

Orlikowski, W. J. (2000). Using technology and constituting structures: A practice lens for studying technology in organizations. *Organization Science*, *11*(4), 404–428.

Østerlund, C., & Carlile, P. (2005). Relations in practice: Sorting through practice theories on knowledge sharing in complex organizations. *The Information Society*, *21*(2), 91–107.

Oudshoorn, N., & Pinch, T. (Eds.) (2003). *How Users Matter – The Co-Construction of Users and Technology*. Inside Technology. Cambridge: MIT Press.

Oudshoorn, N., Rommes, E., & Stienstra, M. (2004). Configuring the user as everybody: Gender and design cultures in information and communication technologies. *Science, Technology & Human Values*, *29*, 30–63.

Pickering, A. (1992). *Science as practice and culture*. University of Chicago Press.

Pollock, N., & Williams, R. (2008). *Software and Organisations: The Biography of the Packaged Enterprise-Wide System, or, How SAP Conquered the World*. Routledge Studies in Technology, Work and Organizations. London: Routledge.

Quintas, P. (1994). Programmed innovation? Trajectories of change in software development. *Information Technology & People*, *7*(1), 25–47.

Raymond, E. S. (1999). *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media.

Rohracher, H. (Ed.) (2005). *User Involvement in Innovation Processes: strategies and Limitations From a Socio-technical Perspective*. Munich; Wien: Profil.

Rouse, J. (2001). Two concepts of practices. In T. Schatzki, K. Knorr-Cetina, & E. von Savigny (Eds.) *The Practice Turn in Contemporary Theory*, (pp. 189–198). London: Routledge.

Royce, W. W. (1970). Managing the development of large software systems. In *Proceedings of IEEE Wescon*, vol. 26, (p. 9).

Scacchi, W. (2007). Free/open source software development: recent research results and methods. *Advances in Computers: Architectural Issues*, *69*.

Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, *52*(7), 1000–1014.

Sole, D., & Edmondson, A. (2002). Situated knowledge and learning in dispersed teams. *British Journal of Management*, *13*(S2), S17–S34.

Sommerville, I. (1995). *Software Engineering*. International computer science series. Wokingham, England: Addison-Wesley, 5 ed.

Stol, K.-J., & Babar, M. A. (2009). Reporting empirical research in open source software: The state of practice. In *Open Source Ecosystems: Diverse Communities Interacting*, IFIP Advances in Information and Communication Technology. Springer.

Teli, M., Pisanu, F., & Hakken, D. (2007). The internet as a library-of-people: For a cyberethnography of online groups. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, *8*(3).

Torvalds, L., & Diamond, D. (2001). *Just for fun: the story of an accidental revolutionary*. HarperBusiness.

Turkle, S. (1997). *Life on the screen: Identity in the Age of the Internet*. New York, NY: Simon & Schuster.

van Wendel de Joode, R., & de Bruijne, M. (2006). The organization of open source communities: Towards a framework to analyze the relationship between openness and reliability. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS 06)*. IEEE Computer Society.

von Hippel, E. (2001). Innovation by user communities: Learning from open-source software. *MIT Sloan Management Review*, *42*(3), 82–86.

von Hippel, E., & von Krogh, G. (2003). Open source software and the private-collective innovation model: Issues for organization science. *Organization Science*, *14*(2), 209–223.

von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, *32*(7), 1217–1241.

Waddington, D. (1994). Participant observation. In C. Cassell, & G. Symon (Eds.) *Qualitative Methods in Organizational Research – A Practical Guide*. Sage Publications.

Wakeford, N., & Cohen, K. (2008). Fieldnotes in public: Using blogs for research. In *The SAGE handbook of online research methods*, chap. 17, (pp. 307–326). SAGE Publications.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, *4*(2), 74–81.

Weber, S. (2004). *The Success of Open Source*. Cambridge, MA: Harvard University Press.

Wenger, E. (1999). *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press, 1 ed.

Williams, S. (2002). *Free as in Freedom: Richard Stallman's Crusade for Free Software*. O'Reilly.

Wynn, D. E. (2003). Organizational structure of open source projects: A life cycle approach. In *Abstract for 7 th Annual Conference of the Southern Association for Information Systems, Georgia*.

Ye, Y., & Kishida, K. (2003). Toward an understanding of the motivation of open source software developers. In *Proceedings of the 25th International Conference on Software Engineering*, (p. 419). Los Alamitos, CA, USA: IEEE Computer Society.

Yin, R. K. (2003). *Case study research: design and methods*. Applied social research methods series. Thousand Oaks, CA: Sage, 3 ed.