Report per il corso di Tecnologie open-source

MeeGo

Analisi di MeeGo , versione 0.6

22 giugno 2010

Sommario

Analisi del progetto open-source MeeGo .

Stato del documento

Formale ad uso interno.

Lista di distribuzione

1. Dott. Ing. Luigi Bellio

Autori

1. Matteo Brunati

Registro modifiche

Versione	Data	Ragione	Autore
0.6	21-06-2010	Ultimato documento	Matteo Brunati
0.5	19-06-2010	Cambiato ordine e titolo delle sezioni (preso quello suggerito dal professore)	Matteo Brunati
0.4	18-06-2010	Scritta sezione 5. Modificata sezione 6.3	Matteo Brunati
0.3	17-06-2010	Conclusa stesura tutta sezione 7. Scritta sezione 6	Matteo Brunati
0.2	07-06-2010	Proseguita stesura sezione 1. Scritta sezione 2, 7 e 7.1	Matteo Brunati
0.1	07-06-2010	Iniziata stesura del documento	Matteo Brunati

Indice

Elenco delle figure

1 Introduzione

Questo documento intende essere un'analisi da un punto di vista "industriale" del progetto open source MeeGo [1].
Data la recente nascita del progetto, e quindi la sua continua e rapidissima crescita in questo momento iniziale, le informazioni utilizzate per stilare questo documento sono per lo più delocalizzate nel web. Si è perciò deciso di prendere come punto di riferimento per le informazioni riguardanti MeeGo il relativo sito web[1] e la community ufficiale [2] che sta nascendo attorno ad esso: articoli dai blog[3] degli addetti ai lavori di Nokia ed Intel ; presentazioni di MeeGo ad eventi del settore, come [23] e [26]; la wiki [4] di MeeGo e le sue mailing-list, ecc.

1.1 Cos'é MeeGo ?

MeeGo è un progetto open source basato su Linux, nato dalla fusione di altri due progetti open source basati su Linux: Maemo[5] capeggiato da Nokia , e Moblin[6] capeggiato da Intel [1].
MeeGo non è solo un sistema operativo (SO), lo si potrebbe definire quasi un framework open source per mettere assieme tutte quelle compementi software necessarie a costruire un sistema operativo com'è inteso oggi, rivolto ad un'ampia utenza grazie alla sua compatibilità con le più diverse piattaforme. Tuttavia gli ideatori di MeeGo lo definiscono uno stack di software open source user-friendly.
Infatti uno degli scopi di MeeGo è quello di costruire un sistema operativo che sia di fatto piattaforma comune per l'utilizzo di netbook, palmari, tablet PC, smartphone, TV connesse ad Internet ed altri sistemi multimediali e d'informazione.

1.2 Cosa fa MeeGo ?

Come appena preannunciato, lo scopo di MeeGo è quello di offrire un **client OSS stack** completo, ovvero una piattaforma software open source che vada a ricoprire tutti gli aspetti d'interazione tra utente finale ed hardware, sun un'ampia gamma di dispositivi.
Con ciò chi lavora a MeeGo intende fornire diverse cose [23]:

- un SO completo di kernel, interfaccia grafica e strumenti di gestione del sistema;
- delle applicazioni ed un'esperienza utente d'impatto positivo per l'utenza;
- un set di API standard per tutte le architetture supportate;
- flessibilità nel supportare software ed add-on proprietari o commerciali.

Da notare come l'ultimo punto sia molto importante se si vuole rendere questa nuova piattaforma appetibile per lo sviluppo di software commerciali, come è successo per Android[7] e l'iPhoneOS[8] (ora diventato iOS).

[1] http://meego.com/
[2] http://meego.com/community
[3] http://meego.com/aggregator
[4] http://wiki.meego.com/Main_Page
[5] http://maemo.org/
[6] http://moblin.org/
[7] http://www.android.com/index.html
[8] http://www.apple.com/it/iphone/softwareupdate/

1.3 Architettura software

Gran parte di MeeGo si appoggia su softwre già esistente, e che per questo viene detto di upstream. L'intera architettura può essere divisa in tre livelli:

Livello del SO base (OS base) Composto principalmente dal kernel Linux, dai servizi più importanti del sistema operativo, chiamati infatti *core services*, e da **HAS**[9].

Livello middleware del SO (Middleware) Fornisce le API per programmare applicazioni native e web indipendentemente dall'hardware su cui girerà l'applicazione, o dal suo modello d'utilizzo.
Probabilmente questa è una delle caratteristiche più importanti di MeeGo , da un punto di vista dei programmatori, per combattere la concorrenza sul mercato, dato che ognuno potrà costruire le proprie applicazioni preoccupandosi solo di cosa devono fare, e non di come lo devono fare.

Livello dell'interfaccia grafica (UX) A questo livello è demandato il compito di fornire l'interazione utente-macchina. Interazione che dovrà avvenire in modo differente tra palmari, smartphone, netbook o TV.

La figura 1 entra nel dettaglio dell'architettura software di MeeGo . Ci sono due considerazioni che si possono trarre da questa immagine.
La prima riguarda il fatto che lo schema fa riferimento alla caratteristiche dell'architettura della versione 1.0 di MeeGo . Come nota esplicita infatti si vede che l'UX per palmari, smartphone e tablet sarà rilasciata con la versione 1.1. Questo dimostra come MeeGo sia veramente una piattaforma ed un sistema operativo modulare, che lo rende quindi facilmente adattabile ad un'ampia gamma del mercato consumer.
La seconda nota si collega esattamente a questo ragionamento. Infatti si può notare come sia stata una scelta strategica importante quella di dividere le diverse interfaccie grafiche a seconda dell'architettura target. Questo permette di essere più agili rispetto alla concorrenza nell'adattamento del proprio prodotto a settori consumer diversi.

[9]Hardware Adaptation Software (HAS). É la parte più vicina all'hardware, e serve per adattare MeeGo alle specifiche architetture hardware da supportare

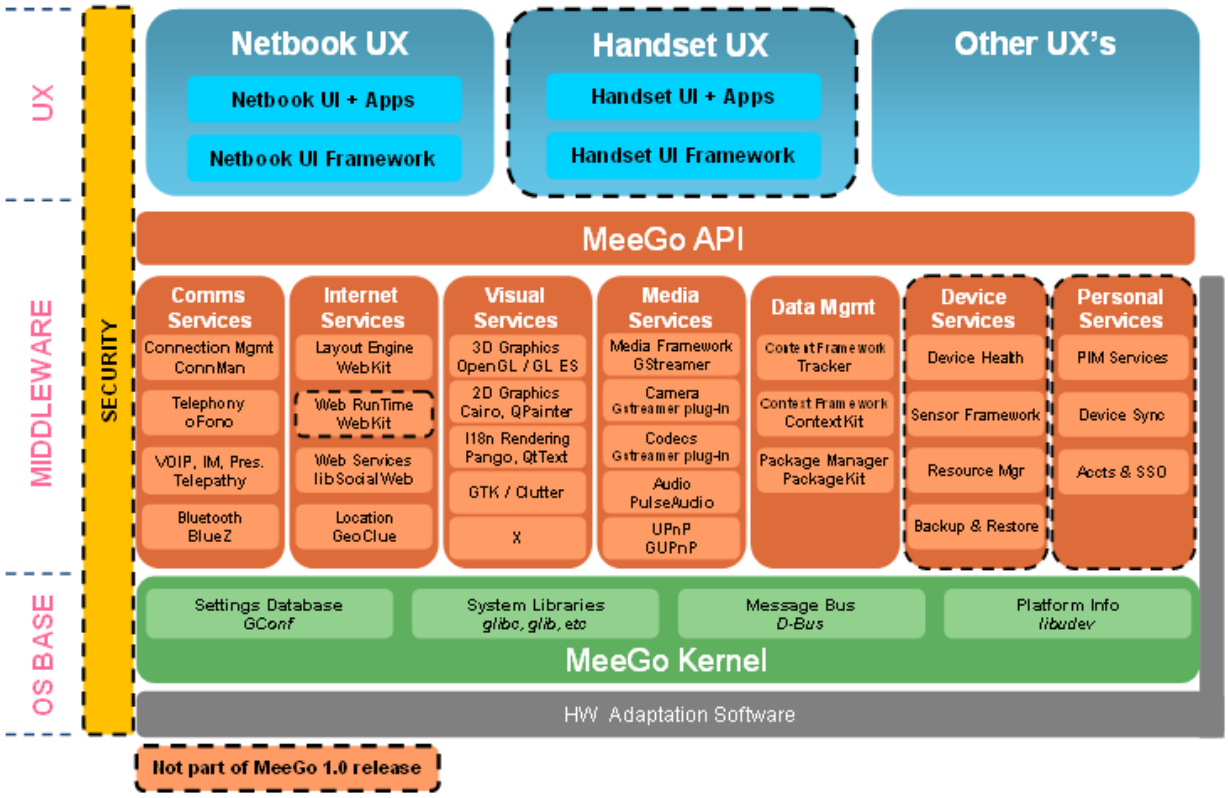

Figura 1: Architettura software di MeeGo

2 Vision

Ci sono diverse interviste e video che parlano di questo argomento [2], [3], [4]. Ciò che sembra chiaro è che MeeGo è stato pensato per "dominare il mondo", o meglio, cercare di unificare sotto un'unico cappello tutti quei dispositivi elettronici che stanno sempre di più diventando di uso comune. Si parla di telefoni cellulari, smartphone, tablet PC, netbook, sistemi IVI[10], TV connesse ad internet e videotelefoni.
Anssi Vanjoki, EVP Markets[11] di Nokia , con questa frase descrive precisamente ed in modo sintetico il ruolo che MeeGo (e gli altri sistemi concorrenti) stanno cercando di realizzare: "leeving in the media" [2]. D'altro canto anche Intel è d'accordo con questa visione, come Paul Otellini, presidente e CEO[12] di Intel , descrive in [4]: "Our vision for seamlessly communicating between computing devices from the home, auto, office or your pocket is taking a big step forward today with the introduction of MeeGo.".
Anssi Vanjoki la prima volta che Nokia annunciava il suo tablet nel 2004, disse che ci sarebbero voluto 5 iterazioni per raggiungere una piattaforma hardware e software abbastanza potente e comune da poter sostituire i nostri PC con qualcosa che ci possiamo portare sempre dietro, come gli smartphone [3]. Come ricorda nell'intervista, con MeeGo siamo ormai giunti alla 4 iterazione, e risulta quindi essere il banco di prova per una futura generazione di comunicazione ed intrattenimento nel mondo digitale.

[10] IVI, In-Vehicle Infotainment
[11] EVP Markets: Executive vice president del settore commerciale (*Markets*)
[12] CEO: Chief executive officer

3 Mercato

Il mercato in cui MeeGo si accinge ad entrare è parecchio fervido di novità e concorrenza in quest'ultimo periodo. Infatti quello degli smartphone e dei netbook è un mercato nato abbastanza di recente, e che ha avuto una crescita considerevole nel 2009. Per quanto riguarda i tablet PC invece si tratta di un mercato nuovissimo, dove che sta uscendo allo scoperto ora, sta anche gettando le basi di questo mercato e le future aspettative dei potenziali utenti.

3.1 Breve analisi di mercato

Per quanto riguarda gli smartphone, Nokia risulta ancora essere leader del mercato grazie a Symbian, ma l'impennata nella vendita di iPhone l'anno scorso, e smartphone targati Android quest'anno consiglia che la casa finlandese esca al più presto con qualcosa di valido sul mercato. Per il momento il competitor più grosso sembra rimanere Android, che grazie al sottostante kernel Linux sta facendo man bassa di smartphone, ma anche di tablet PC e netbook [13].
Altri studi poi danno come in maggior crescita il mercato dei tablet PC a scapito dei netbook [27]. Questo porta ad un'ulteriore domanda: ma quale sarà il vero utilizzo di questi dispositivi? Domanda alla qualce non c'è ancora una risposta definitiva, dato che il mercato asi sta definindo ora.
Per questo MeeGo dev'essere una piattaforma il più flessibile e potente possibile, per poter fronteggiare i propri rivali sul più ampio numero possibile di dispositivi, ed avere la prontezza di adattarsi alle nuove necessità del mercato consumer in modo molto veloce.

3.2 Concorrenza

Come Anssi Vanjoki dice in [2], ci sono tre tipi di concorrenti che Nokia (e MeeGo di conseguenza) può affrontare in questo periodo storico: gli "infrastructural competitors", come Google; chi cerca di sfondare su un'unica nicchia di mercato ma in modo verticale, come Apple o RIM; i diretti concorrenti di Nokia : chi costruisce dispositivi portatili con il proprio marchio, come HTC, Motorola, Samsung, ecc. In un modo o nell'altro, tutti questi sono i concorrenti di MeeGo , dal momento che sono loro ad appoggiare maggiormente lo sviluppo di piattaforme come Android, WebOS, iOS, ecc.
Anssi Vanjoki prosegue quindi nella stessa intervista dicendo che Nokia vuole cercare di passare di livello, partendo da una "mobile company" che è ora ed arrivando ad una "mobile solution platform company". Tutto questo anche anche grazie a MeeGo , che avrà quindi il ruolo di gettare le basi per un nuovo tipo di piattaforma: una piattaforma comune per il più ampio possibile numero di dispositivi di intrattenimento e lavoro multimediale.

3.3 Alleanze

MeeGo è un progetto nato da poco, ma che ha già avuto il supporto pubblico di diverse compagnie, come Asus, Novell, Acer, e moltissime altre [8], [31]. Il supporto viene dato non solo come contribuzione di codice al progetto, come nel caso di Linpus [12], ma anche di supporto ad hardware specifico o di vendita di dispositivi con preinstallato MeeGo , come annunciato da BMW Group per l'update del progetto GENIVI[13].

[13] http://www.genivi.org/

4 Storia

4.1 Maemo & Moblin

Maemo è una piattaforma open source sviluppata e mantenuta da una comunità opensource sotto la guida di Nokia . Pensato per portare Linux e tutte le applicazioni che ci girano su Internet tablet come il Nokia N810. Nel 2009 è stato lanciato sul mercato il Nokia N900, che è andato oltre all'Internet tablet, diventando lo smartphone di casa Nokia di fascia più alta. Maemo è nato nel 2005 per l'Internet tablet Nokia 770, ed arrivato nel 2009 alla versione 5 rilasciata per l'N900. Guardando sotto il cruscotto di Maemo, si può trovare un sistema operataivo basato su Debian, compilato per architettura ARM grazie a $[14]$. L'interfaccia grafica si appoggia al Framework UI Hildon: X Window System con window manager Matchbox, API per le applicazioni GTK+ con le estensioni dette Hildon.
Sull'altro piatto della bilancia c'è **Moblin**, che sta per "mobile Linux", un progetto open source per lo sviluppo di un sistema operativo, portato avanti ultimamente dalla Linux Foundation e finanziato principalmente da Intel . Pensato per device mobili più potenti dei telefoni, come i netbook o i sistemi IVI[11], è nato nel 2007 ed è passato sotto la custodia della Linux Foundation solo nell'Aprile del 2009. Il sistema operativo sottostante non si rifà a nessun'altra distribuzione, ma come Maemo, anche per la grafica ci si appoggia a GTK+.

4.2 La fusione: Maemo + Moblin = MeeGo

Il primo annuncio del merge di Maemo e Moblin è stato dato il 15 febbraio 2010. In quell'occasione sono stati pubblicati due filmati prodotti da Nokia e Intel [4] [5], oltre al primo articolo del blog ufficiale di MeeGo [33].
MeeGo è un progetto alquanto particolare, perché nato dalla fusione di due progetti open source di Nokia ed Intel già avviati, ed accudito dalla Linux Foundation. Come i suoi "genitori" MeeGo continua a rimanere open source, con una comunità che consta già circa 7500 partecipanti (da metà Febbraio 2010!).

4.3 Perché è nato MeeGo ?

Non si può dire con certezza quali siano state le vere motivazioni di Nokia ed Intel per unificarsi e dare alla luce MeeGo .
Le motivazioni apportate nei video ufficiali per il lancio del prodotto [4] [5] sono le più diverse. Si va dall'unificazione dell'API in dotazione agli sviluppatori di software per piattaforme diverse, all'unificiazione della piattaforma per l'utilizzo di queste piattaforme hardware.
Dal punto di vista del mercato la cosa verrà analizzata meglio in 3, ma è evidente che la scelta di unificare due piattaforme intese per l'utilizzo di device così differenti (tablet e smartphone per maemo, netbook per moblin), porta un notevole vantaggio in casa Nokia ed Intel . Infatti, com'è successo per Android e Symbian a suo tempo, la quantità di dispositivi supportati da una piattaforma unica andrebbe ad aumentare vertiginosamente. Questo permette inoltre lo sviluppo di applicazioni multipiattaforma, aumentando l'appetibilità dell'adozione di MeeGo per coprire il più ampio bacino consumer possibile.

[14] http://www.scratchbox.org/

5 Licenza

Questo argomento è probabilmente uno dei più delicati. Per questo si fa riferimento diretto alla pagina ufficiale del progetto, dove vengono esplicitate tutte politiche riguardo le licenze [21].

Dal momento che MeeGo è nato dal merge di altri due progetti, cerca di mantenere le politiche di licenza ereditate da questi. Questo implica che la politica di licenza adottata debba essere compatibile con tutte le licenze dei progetti che include, ovvero quei progetti deifiniti di upstream[15].
Poiché MeeGo si può dividere in due sottosistemi, si possono individuare due politiche di licenza principali: quella abbracciata da tutto ciò che fa partle del sistema operativo (operating system software), e quella utilizzata dal sottosistema relativo all'intereazione utente-macchina (user experience subsystem software).
C'è un'altro fattore che rende ancor più interessante l'ecosistema delle licenze che si possono e potranno trovare in MeeGo . Infatti uno degli scopi di MeeGo è essere una piattaforma utilizzabile per scopi commerciali. Quesot implica la possibilità di adottare estensioni proprietarie a tutti i livelli dell'architettura.

5.1 Licenza dell'*operating system software*

La politica di licenza per questa parte del progetto è guidata dall'essere il più allineata possibile con il resto delle licenze della comunità open source attorno a Linux.
Quindi le licenze delle componenti del sistema operativo seguiranno quelle dei progetti originali. Per esempio per il kernel Linux sarà sotto la licenza GNU GPL v2.
Per quanto riguarda invece le licenze delle API ed altri framework per lo sviluppo di applicazioni, dovrà essere possibile linkare componenti o plug-in proprietari. Sono quindi accettate tutte le licenze OSI-compatbili, ma è consigliato l'uso di licenze GNU (L)GPL vresioni 2.x.

5.2 Licenza della *user experience subsystem software*

Per quanto riguarda il sottosistema d'interazione utente-macchina, ci possono essere diverse licenze possibili. In prima istanza si auspica l'utilizzo di licenze permissive OSI compatibili, come le licenze BSD-style[16]. In caso ciò non fosse possibile, si consiglia l'utilizzo di licenze copyleft.
Ci sono tuttavia due eccezioni citate a questa politica. La prima riguarda il codice che estende le MeeGo Operating System API, che deve utilizzare la stessa licenza delle API estese, preferibilmente la GNU LGPL version 2.x. Questo per cercare di non segmentare troppo API di MeeGo .
La seconda eccezione riguarda quei software maturi ed indipendenti, che sono di fatto diventati standard o che aiutano a crescere gli standard. Per questo tipo di situazione si riporta come esempio il browser web Firefox.

5.3 MeeGo trademark

MeeGo è un trademark della Linux Foundation . Non è ancora stato registrato, infatti quando viene citato nelle pagine del sito, si può talvolta trovare la scritta "MeeGo ™".

[15] *upstream* significa letteralmente "corrente superiore". In questo caso con questo termine si identificano tutti quei software o progetti non direttamente sviluppati sotto il marchio di MeeGo , ma integrati o utilizzati da quest'ultimo. Alcuni esempi possono essere il kernel **Linux**, le librerie **Qt** o il framework video **X Windows System**

[16] Le licenze BSD-style non obbligano a rendere open source le modifiche al codice

5.4 Copyright delle contribuzioni

Un'altro apsetto importante riguardo le licenze è il trattamento delle contribuzioni da parte della comiunità open source. Al contrario di quanto fa l'Apache license v2, la politica di MeeGo è quella di non richiedere ne accettare l'assegnamento del copyright sul codice portato dai contributori esterni.
Le motivazioni apportate per queste decisioni sono due:

1. non vi è ulteriore burocrazia per la gestione del processo di contribuzione;
2. per enfatizzare il fatto che sono le stesse persone che contribuiscono al progetto che portano gli oneri e gli onori del proprio codice.

Aggiungerei inoltre che le persone sono più agevolate, e quindi invogliate, nell'apportare il proprio codice, software le proprie patch al progetto.

6 Business Model

Il business model adottato da MeeGo non si può far risalire esattamente ad uno specifico di quelli visti in classe e targati come "FLOSS success stories". Si può dire che il progetto è così complesso ed ampio che aggrega in se diversi modelli di business.
In particolare, dei modelli di business per i progetti FLOSS visti in classe ed adottati almeno in parte da MeeGo , si possono identificare i seguenti:

Distribuzione del software prodotto con marchio MeeGo É quanto fa la Linux Foundation , rilasciando open source e liberamente scaricabili tutte le versioni di MeeGo : da quella per gli sviluppatori, a quella per gli end-user.

Donazioni al progetto MeeGo Essendo il progetto nato da Nokia ed Intel , ma condotto dalla Linux Foundation (associazione non-profit), si può dire che parte dei fondi che la Linux Foundation riceve vengano devoluti allo sviluppo di MeeGo .

Costruire e vendere configurazioni hardware e software Poiché sia Nokia che Intel sono sia produttori di software, ma soprattutto di hardware, non c'è da meravigliarsi se questi due colossi cominceranno a vendere prodotti con MeeGo installato. Se per Nokia ci sono solo rumor del prossimo smartphone finlandese con sistema operativo MeeGo (il Nokia N920 o N9), e comunque ha annunciato la compatibilità del Nokia N900 con MeeGo , per quanto riguarda Intel i tablet PC con il nuovo processore Morestown sono già stati presentati al Computex 2010 a Taipei [25], rivelando tra le altre cose anche il concept per l'interfaccia tablet di MeeGo .

Rilascio di software proprietario basato su MeeGo Delle alleanze di MeeGo si è discusso nella sezione 3.3. Questo fa pensare al fatto che in molti potranno utilizzare MeeGo per rilasciare delle versione con il proprio marchio. Ricordiamo per esempio come Novell abbia annunciato il suo appoggio a MeeGo , pensando già ad una sua distribuzione basata su di esso [28], [32].

Software rilascio sotto doppia licenza Come spiegato nella sezione 5, le due principali famiglie di MeeGo sono quelle basate sulla (L)GPL v2.x e quelle BSD-like. É proprio grazie a quest'ultima famiglia di licenze che s'intende rendere MeeGo adatto all'utilizzo e distribuzion con software proprietario.

7 Processo di sviluppo

Come per altri altri progetti open source, anche per MeeGo nulla è dato al caso. Gli stessi processi di sviluppo sono pensati per garantire la maggior produttività nel minore tempo possibile, pur mantenendo un certo livello di qualità in ciò che si produce. Come si può vedere dalla figura 2, anche solo la gestione della roadmap e della richiesta dei requisiti ha un preciso processo che le guida.

Il processo più importante nel rilascio di una versione di MeeGo è la creazione e gestione di una release. Cosa può contenere una release in questo caso? Per esempio per la versione 1.0 sono state rilasciate due immagini del sistema operativo. Una rivolta ai netbook con processore Intel Atom, dotata sia della parte *core* di MeeGo che dell'interfaccia grafica e di tutto ciò che serve per l'intereazione tra utente finale e macchina. L'altra immagine invece è rivolta ai programmatori, e comprende solo la parte core di MeeGo . Per le prossime release sono state previste la pubblicazione di altre interfacce grafiche e relative API.
Vediamo quindi di analizzare più in dettaglio come avviene la creazione di una release [19].

7.1 Creazione di una release

Ogni release ha una durata di 6 mesi. Questo significa che ogni 6 mesi viene pubblicata una nuova versione di MeeGo che può contenere consistenti modifiche a tutti i livelli dell'architettura. Una release è vista un po' come la gestione di un singolo progetto: c'è un inizio, a cui segue un'analisi, una progettazione ed una realizzazione. Alla fine vi è il periodo di mantenimento del progetto ed eventualmente la sua morte.
Questi differenti processi sono stati suddivisi in 4 fasi. Le prime due sono descritte assieme.

Creazione di una road map e sua pianificazione Durante questo periodo si fa un'analisi dei requisiti che potrebbero essere inseriti nella nuova release, e di quelli che invece dovranno essere inseriti, come risoluzioni di bug o compatibilità. Questa fase viene iniziata 18 mesi prima della data di release prestabilita. Tutta la gestione di questa fase è raffigurata nell'immagine 2.

Fase di sviluppo Della durata di 7 mesi, questa fase è a sua volta suddivisa in altre sottofasi.

1. 2 mesi: apertura del Trunk[17] che avviene il mese prima del rilascio della release precedente. In questa fase si possono apportare cambiamenti intrusivi nella piattaforma, che possono causare un'interruzione del ciclo di build e delle compatibilità.
2. 2 mesi: integrazione delle nuove feature. Il Trunk diventa base stabile per lo sviluppo.
3. 2 mesi: bug fixing e stabilizzazione della release. Vengono integrate anche eventuali feature minori, e che non compromettono la tabella di marcia. Questa fase termina con la creazione del branch[18] della release da pubblicare.
4. 1 mese: risolti i bug e le incompatibilità più grosse. Si cerca di rilasciare i prodotti delle diverse categorie (palmari, smartphone, tablet, sistemi IVI, ecc.). Questo mese termina con la pubblicazione della release, dopo la quale ci saranno solo aggiornamenti.

[17] *trunk* letteralmente significa "tronco". Qui si usa per riferire un particolare ramo del repository, che per uso comune si riferisce alla più recente versione in sviluppo.
[18] *branch* si traduce letteralmente con "ramo". Come per il trunk, indica un particolare ramo del repository, questa volta per uso comune rappresenta la versione del software con il nome del branch

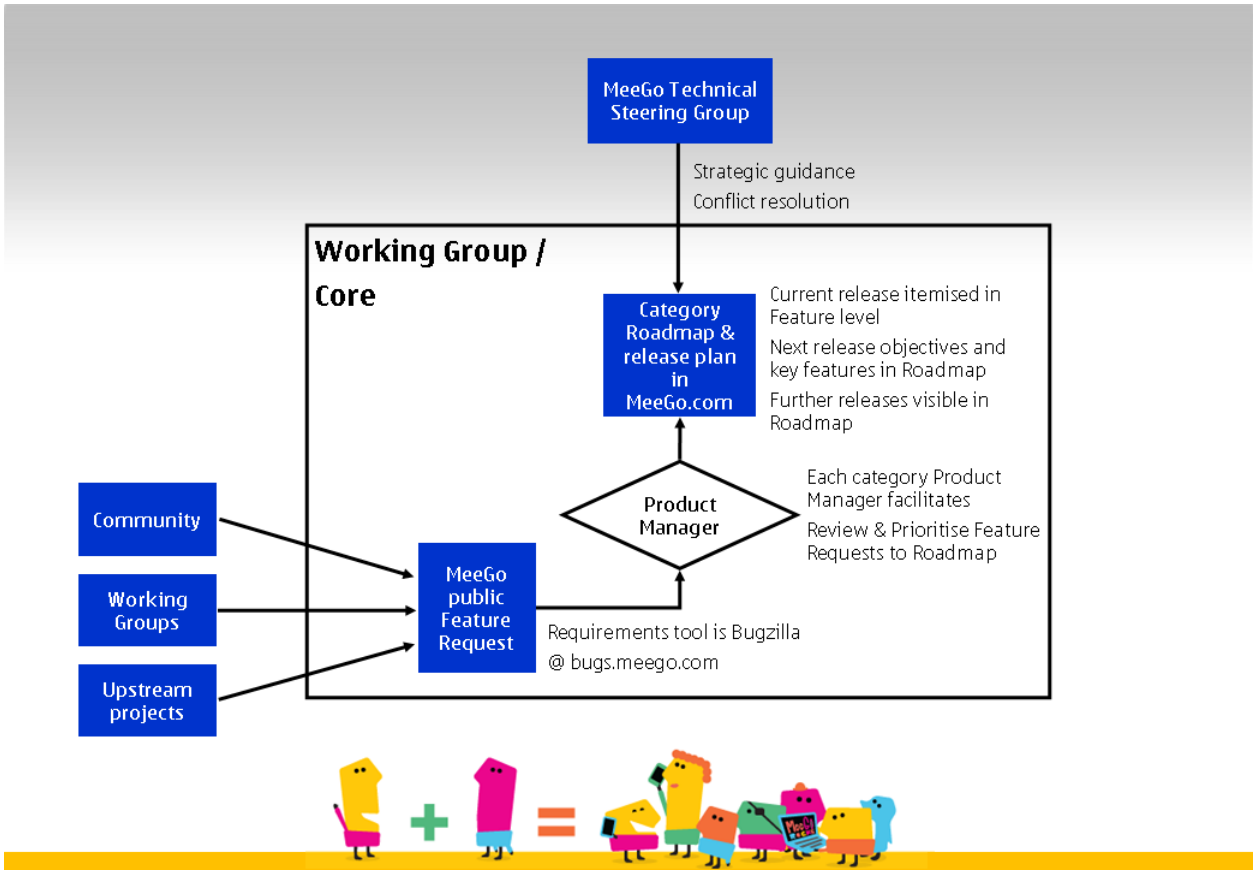

Figura 2: Road mapping e gestione dei requisiti

Fase di manutenzione In questa fase si pensa principalmdirettaente al mantenimento della qualità e della compatibilità del prodotto, dando assistenza ai produttori di hardare o software che vogliono adottare MeeGo per un servizio all'avanguardia da un punto di vista tecnologico.

Queste quattro fasi si possono vedere rappresentate graficamente nella figura 3.

7.2 Modello di sviluppo iterativo

Vediamo ora di analizzare un po' meglio come avviene la fase di sviluppo, che dovrebbe durare 7 mesi.
Il modello di sviluppo adottato è di tipo iterativo, con cicli settimanali. Questo permette di sviluppare abbastanza in fretta le feature via via programmate, ma soprattutto di accorgersi in fretta se c'è qualche errore, o se qualcosa non supera i test di qualità (quality assurance, QA [24]). Durante ogni iterazione vi sono nigthly build, ed alla fine della settimana viene rilasciata la versione del Trunk. La build risultante di ogni iterazione serve per tracciare lo stato di sviluppo di MeeGo .
Il Trunk viene aggiornato giornalmente, e contiene quindi l'ultimo codice funzionante. Due volte nell'arco di una iterazione viene anche ripopolato il così detto repository di preview, dove dal Trunk viene spostato il codice che deve essere revisionato per la QA. La prima generazione di questa "snapshot" viene fatta il venerdì. Durante il weekend ed i primi giorni della settimana

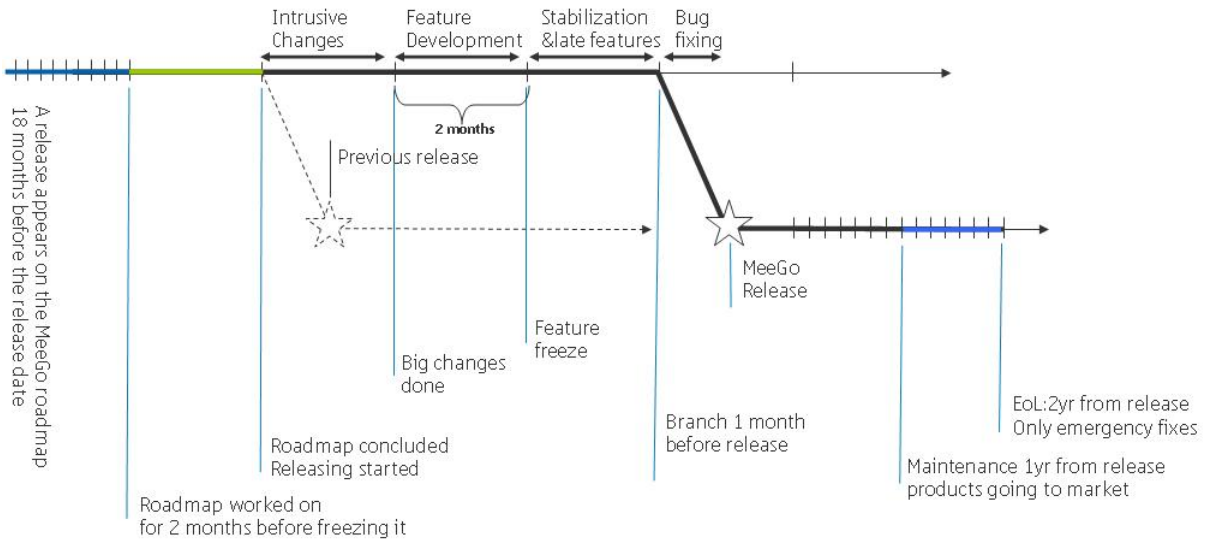

Figura 3: Grafico temporale di una release di MeeGo

sono fatti i test di QA, ed il mercoledì mattina si crea un'altra snapshot del repository per la versione finale dell'iterazione. Questa procedura può sembrare lineare, ma l'inserimento di codice nel Trunk è tutt'altro che banale. Si veda la sezione 7.3 per una visione più dettagliata di questo processo. Chi necessita di provare la compatibilità del proprio software con la versione "bleeding edge" di MeeGo , può fare riferimento alla snapshot del venerdì.
Come accennato prima, la versione del venerdì serve per dare una visione generale della release settimanale, e per questo è chiamata *preview*. Ogni sua modifica deve essere vagliata dai *Release Engineers*, che possono accettare, rifiutare o posporre l'inserimento di una feature o una modifica nella release. L'immagine 4 da un'idea del lavoro fatto durante un'iterazione. Sebbene nella mailing-list per gli sviluppatori [17] si parli di *sprint*[19], il programmatore che ha usato questo termine si riferiva probabilmente ad una iterazione del modello incrementale, dato che una settimana è abbastanza breve come periodo per un incremento.
Questo mi fa presupporre quindi all'utilizzo di un modello che di fatto è un misto tra un modello incrementale ed una metodologia agile, come scrum [20], vista anche la suddivisione gerarchica dei compiti all'interno del progetto, e la gestione del lavoro divisa in working-group (WG).

7.3 Come si porta il codice nel Trunk o in una release?

La regola principe dell'importazione di nuovo codice nel Trunk o in una release, è che non debba corrompere le immagini di riferimento di MeeGo . In questo modo si può continuare a lavorare mentre si aggiungono nuove feature.
Detto questo, vediamo le tre fasi principali che attraversa un pezzo di codice prima di entrare nel Trunk di MeeGo , come mostrato nella figura 5.

[19] Lo *sprint* è l'unità di tempo base utilizzata dai gruppo di lavoro nelle metodologie agili. Può variare da qualche giorno a massimo qualche settimana

[20] *scrum* letteralmente significa mischia. É una metodologia agile che si basa su sprint di breve durata e su gruppetti di lavoro di poche persone, dove tutti sanno fare tutto

Figura 4: Programma settimanale durante la fase di sviluppo di MeeGo .

Staging Gli sviluppatori producono codice e lo pachettizzano tramite l'OBS[21]. Se il mantainer del package accetta le modifiche in base ai risultati dei test automatici di QA, si passa alla fase successiva.

Testing In questa fase si fa QA mirata ed eventualmente manuale di validazione sul pacchetto che è stato proposto dalla fase precedente. Questa volta è il mantainer della distribuzione che decide se il nuovo codice può passare alla fase successiva.

Trunk Questa è l'ultima fase, in cui il build notturno di tutto il codice prodotto genera i report dell'integrazione di tutte le parti del sistema e dei test di QA. Se tutto va bene viene proposta la preview della release (quella prodotta il venerdì), che deve passare al vaglio del release manager. Se anche il release manager da il suo consenso viene prodotta la release settimanale (quella del mercoledì).

La procedura non è tutt'altro che banale ed immediata, ma assicura un alto livello di qualità nel codice prodotto.

7.4 Come creare un prodotto partendo da MeeGo ?

Come detto nella sezione 6, uno degli obbietti di MeeGo è quello di essere una piattaforma aperta per lo sviluppo di (o utilizzabile con) soluzioni proprietarie. Per questo serve uno modello per la creazione di release o di un prodotto basato su MeeGo piuttosto versatile ed elastico.
Chi ha pensato alla parte di building della piattaforma, ha pensato perciò a qualcosa che potesse

[21] OBS sta per OpenSuse Build System. É il sistema di build progettato per OpenSuse, progetto open source ed adottato anche per MeeGo . Oltre a compilare il codice di diversi linguaggi di programmazione per diverse piattaforme, può lanciare test sul prodotto e pacchettizzarlo in base alla piattaforma specificata (DEM, RPM, ecc.)

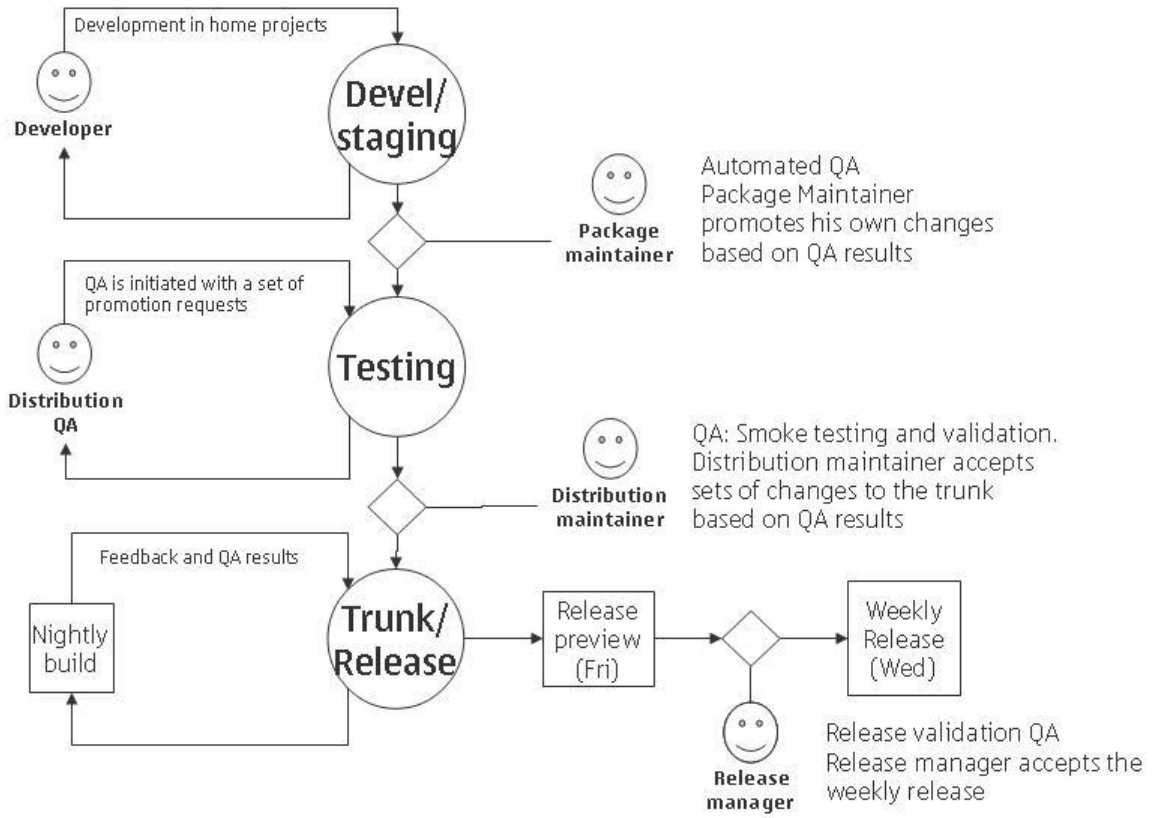

Figura 5: Flusso di integrazione in MeeGo di un pezzo di codice.

essere utilizzato sia dalla comunity di MeeGo , che da HW vendor che sviluppano le proprie soluzioni su MeeGo . Come mostrato in figura 6, queste due attività posso andare in parallelo. Così facendo le release ufficiali di MeeGo sfruttano il codice proprietario reso open source da chi lo produce, e viceversa, gli HW vendor posso integrare il proprio codice proprietario nelle release di MeeGo prodotte dal sistema di build ufficiale di MeeGo . Per assicurare gli HW vendor della compatibilità di MeeGo e delle sue potenzialità derivate dall'essere un progetto open source, è stata anche creata una pagina in cui si spiega un po' più nel dettaglio (cosa che non viene fatta ora) come adattare MeeGo ad hardware specifico [11].

7.5 Versionamento delle Release

Ogni versione (o release) di MeeGo è accompagnata da un numero, in modo che si possa capire al volo con che tipo di versione di MeeGo si sta lavorando.
Non viene esplicitato in questa sede il significato di ogni codice, dal momento che non lo si ritiene indispensabile per questa analisi di MeeGo . Si rimanda quindi alla pagina delle wiki di MeeGo [19] dov'è è spiegato in dettaglio questo argomento.
Per la nostra analisi, basti sapere che X.Y indicano delle major release, per esempio la 1.0 o la 1.1 sono due major release, che quindi hanno importanti differenze una dall'altra (non necessariamente di incompatibilità).

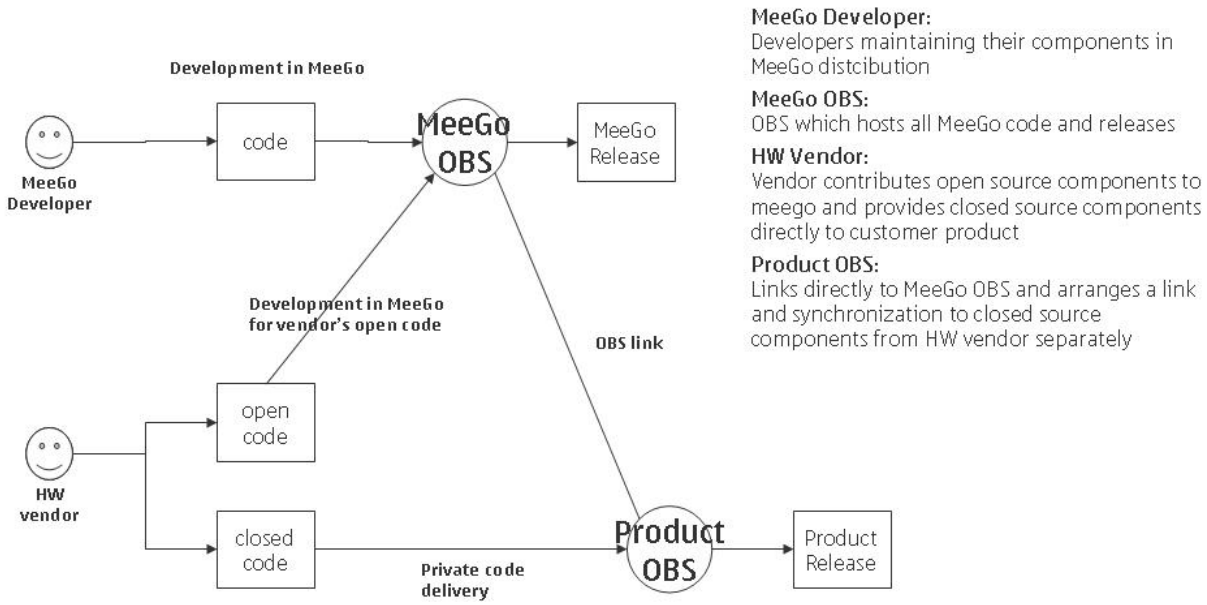

Figura 6: Modello di collaborazione tra la comuntià di MeeGo e gli HW vendor.

7.6 Come si puo' collaborare facendo parte della comunity?

Sebbene MeeGo stia crescendo abbastanza in fretta, come si puo' vedere dalle analisi pubblicate in [22], la risposta a questa domanda non ci e' ancora data a sapere con precisione [16]. Anche la pagina dove viene spiegato il processo di sviluppo delle applicazioni per la compunita' open source non va poi cosi' nel dettaglio [18].
Qualche intuizione sul processo di contribuzione potrebbe venire dal post che Robin Burchell ha fatto sul suo blog, riguardo la prima richiesta di inserimento nel trunk di un contributo esterno [9]. Qualche altro indizio si puo' trovare qua e la' nella wiki, ma ancora niente di ufficiale. Cio' che comunque sembra essere chiaro e' che le contribuzioni al codice di MeeGo , e quelle riguardanti lo sviluppo di proprie applicazioni per la piattaforma, devono percorrere due persorsi differenti ([18], sezione 7.3).
Sembra comunque che la cosa potrebbe essere risolta entro breve per la comunita' di sviluppatori di applicazioni per MeeGo . Infatti oltre all'iniziativa del Garage (v. sezione 11.3), Nokia sta lavorando per fornire una piattaforma web basata su Druopal, per la pubblicazione delle proprie applicazioni per MeeGo [14], sulla falsa riga di quanto è stato fatto per Android, gli sviluppatori di Apple, ecc.

8 Comunità

In quanto progetto open source, MeeGo ha bisogno di una guida tecnica e spirituale piuttosto forte, che possa garantirne la crescita mantenendo la parte di collaborazione ed apertura verso la comuntià.
Dal momento che il progetto è nato dall'unione di due progetti di casa Nokia ed Intel , e poiche' viene mantenuto sotto la custodia della Linux Foundation ed e' stato aperto alla comunita open source, probabilmente viene naturale chiedersi quali siano gli schemi secondo cui e' stato deciso di gestire tutte queste persone, dal momento che già a fine Maggio 2010 si sono superati le 7000 iscrizioni alla community open source (ovviamente non tutti sono programmatori ...).

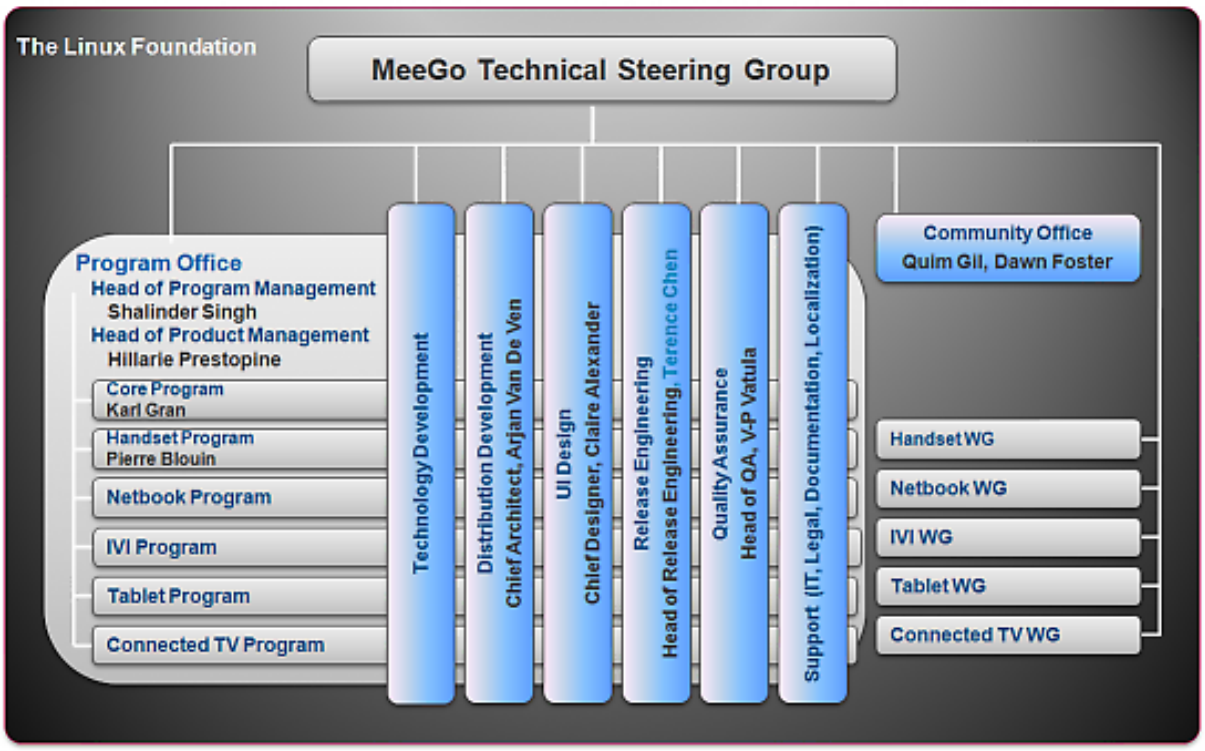

Figura 7: Organizzazione delle persone per lo sviluppo di MeeGo , all'interno di Nokia , Intel e la Linux Foundation .

8.1 Il Team principale di MeeGo : Nokia + Intel

Come si può vedere dalla figura 7, l'intero progetto fa capo a due persone, Imad Sousou di Intel e Valtteri Halla di Nokia , che costituiscono il **Technical Steering Group** (TSG).
Sotto il TSG sono state create altre strutture di persone con ruoli e responsabilità diverse. Dall'immgaine 7 si può capire come ci siano due tipi di responsaibilità principali, che presonalmente definirei *locale* e *globale*.
Le strutture con responsabilità locale sono così definite perché fanno riferimento ad un solo aspetto del progetto, come l'**Handset Working Group** (WG) che è guidato dall'**Handset Program**, o il **Netbook WG** guidato dal **Netbook Program**. Questi tipi di strutture sono visualizzate nell'immagine orizzontalmente.

Dall'altra parte ci sono le strutture a responsabilità globale, perché ciò che viene deciso da loro si ripercuote in tutto il progetto e per tutti i tipi di mercato, palmari, IVI, netbook, ecc. Un esempio di questi gruppi di persone sono: il **Distribution Development**, che gestisce l'evoluzione della distribuzione (intesa come SO) MeeGo ; **Quality Assurance** (QA), che definisce le regole per la creazione dei test per la verifica della qualità dei codice [24]; il **Community Office** (CO), che è l'ufficio preposto alla definizione delle strategie per gli strumenti ed i processi di sviluppo per MeeGo [15].

8.2 La comunità open source

Per diventare membri della comunita' open source di MeeGo non c'è nessun processo di ammissione, nè contratti da firmare o qualsiasi tipo di costo. Individui, ditte o organizzazioni sono accolte nella comunita', ed i tipi di contribuzioni possono essere delle piu' diverse: dallo sviluppo di nuove feature alla risoluzione dei bug, dalla produzione di documentazione all'evangelizzazione di MeeGo . Insomma, tutto cio' di cui necessita il progetto per crescere e mantenersi.
Importante notare come i principi di assegnazione delle responsabilita' siano basati sulla meritocrazia, ovvero si guarda la quantita' e la qualita' del codice che viene prodotto. Questo garantisce che solo chi è veramente motivato e disponibile nei confronti del progetto riesce ad eccellere nella comunita' ed eventualmente guadagnarsi qualche posto di rilievo nei WG. Così è più facile garantire una certa qualita' nel prodotto risultante.

8.2.1 Mentoring system

Proprio qualche giorno fa è stata pubblicata su due mailing list (*community* e *dev*) la proposta di creare un nuovo sistema di mentoring [7], che dovrebbe servire a diversi scopi.
Per prima cosa ad si avvicinerebbero maggiormente a MeeGo gli sviluppatori più esperti. In secondo luogo si creerebbe una forte comunità di sviluppatori tramite il meccanismo studente-insegnante, spiegando ai "novelli" le buone tecniche di programmazione, i trucchetti del mestiere e fidelizzando così le persone ad un modo di lavorare che li tratterrebbe più facilmente atttorno al progetto.

9 Strumenti per la gestione dell'informazione

In un progetto open source una delle cose più importanti è la comunicazione tra le persone, un po' come in una famiglia. Se poi pensiamo che questa famiglia può essere di diverse migliai di persone, si capisce come sia necessario munirsi non solo di strumenti per comunicare, ma anche per il tracciamento e la gestione di queste comunicazioni.
A quesot indirizzo si possono trovare alcuni riferimento di quando verrà detto di seguito: http://wiki.meego.com/Maemo_and_Moblin_community_assets.

9.1 IRC

Come strumento per lo scambio immediato di informazioni, o per la richiesta di aiuto, sono stati messi a disposizione alcuni canali IRC. I canali messi a disposizione sono tre:

- un canale sempre aperto: #meego su irc.freenode.net;
- un canale per le discussioni programmate tra i membri di MeeGo : #meego-meeting su irc.freenode.net;
- dal momento che questo canale viene moderato, vi è un'altro canale per le domande da fare in una discussione programmata: #meego-meeting-questions su irc.freenode.net.

9.1.1 Tracciamento delle riunioni

Per tenere traccia delle riunioni su IRC viene utilizzato **MeetBot** [22]. Nelle discussioni viene utilizzata una sintassi specifica per definire le cose di cui meetbot deve tenere traccia. Un esempio si traccia di una conversazione e della conversazione originale si possono trovare qui:

- http://trac.tspre.org/meetbot/meego-meeting/2010/meego-meeting.2010-06-09-19.03.html
- http://trac.tspre.org/meetbot/meego-meeting/2010/meego-meeting.2010-06-09-19.03.log.html

9.2 Wiki

Per quanto riguarda la wiki è stato deciso di appoggiarsi a **MediaWiki** [23]. Permette un alto livello di personalizzazione ed è piuttosto potente e performante, essendo il progetto che sta alla base di Wikipedia. Tuttavia è mio avviso mal organizzata, e nella maggior parte dei casi è difficile trovare ciò che si sta cercando (forse perché il sistema di ricerca non funziona?...), nonostante ci siano delle linee guida per la contribuazione alla wiki.

9.3 Sito web

Per quanto riguarda la gestione del sito web vetrina di MeeGo [1], è stata lasciata in mano alla Linux Foundation . Il sito è il punto di partenza per tutto ciò che interessa MeeGo : dal downoload delle varie release all'accesso alla community o alla wiki, ecc.
Sebbene non vi siano dichiarazioni esplicite della piattaforma utilizzata per questo sito, tutto fa pensare all'utilizzo di un CMS. Guardando meglio nel codice della prima pagina (come delle altre)

[22] http://wiki.debian.org/MeetBot
[23] http://www.mediawiki.org/wiki/MediaWiki

si può trovare questo pezzetto di codice: `jQuery.extend(Drupal.settings, { basePath:`, che fa pensare all'utilizzo di **Drupal** come CMS per il sito. La scelta non sarebbe poi così azzardata, dato che il numero di plugin rendono questa piattaforma piuttosto configurabile ed adatta per la gestione di una community.

9.4 Help

Altra parte importante del progetto è il manuale utente: http://help.meego.com/. Per il momento la pagina iniziale riporta la guida per la versione netbook di MeeGo , e ciò è più che plausibile dato che c'è solo questa versione per ora. sarà interessante vedere come si svilupperà questa guida, e se rimmarrà utilizzabile com'è ora.

9.5 Mailing-list

Quando ci si iscrive alla comunity di MeeGo [24] viene chiesto se ci si vuole iscrivere anche alle mailing-list pubbliche del progetto. Per il momento ci sono sei mailing-list: *MeeGo Announce*, *MeeGo Dev*, *MeeGo Community*, *MeeGo iL10n* (per la localizzazione), *MeeGo Security*, *MeeGo SDK*. Dal momento che ritengo i nomi di queste mailing-list autoesplicativi, non spiego per ciascuna di esse il loro tema.

9.6 Blogs

Questa è una risorsa utilizzabile in una sola direzione. Infatti sul sito di MeeGo vengono pubblicati anche i post dei blog di alcune persone che lavorano su MeeGo (o nelle vicinanze: Maemo, Moblin, KDE, ecc.). Ho detto che è una risorsa ad una sola via d'utilizzo perché i blog da cui vengono presi questi post sono scelti dall'alto.

9.7 Forum

Se le mailing-list sono intese più per gli sviluppatori, si è pensato al forum per coloro che non lo sono. Si possono trovare discussioni di tutti i tipi, ma soprattuto è un buon posto dove si possono trovare risoluzioni a problemi comuni, senza dover andare a cercare nelle mailing-list o nel sistema di bug tracking.

[24] http://meego.com/user/register

10 Strumenti di sviluppo

Lo svilupop per MeeGo dev'essere fatto su una macchina Linux. Questo perché sebbene le librerie Qt siano multipiattaforma, l'ambiente di sviluppo non lo è. Vediamo meglio perché.

10.1 SDK

L'SDK di MeeGo è composta principalmente da due parti [10]:

- un ambiente chroot che esegue le applicazioni per MeeGo tramite **Xephir**;
- uno script, chiamato **meego-sdk-chroot**, che esegue il simulatore e **QtCreator** all'interno di un ambiente chroot.

Inoltre per eseguire Xephyr, con l'attuale configurazione dells SDK che si può scaricare, è necessaria una macchina con processore Intel ed una distribuzione Linux "recente". Un requisito non da poco per un progetto open source.
C'è anche da notare come Xephyr serve per eseguire MeeGo con l'UX per netbook, dal momento che si dovrà aspettare la versione 1.1 di MeeGo per avere altri layer UX disponibili.

10.2 Building system

Il building system è basato su **OBS**[25] [29] di Novell. Il sistema è stato adottato dalla Linux Foudnation per gestire e creare le immagini ed i paccheti di Moblin dal 2008. Essendo risultato uno strumento molto potente e flessibile, è stato scelto anche per la gestione dello sviluppo, delle release e dei paccheti di MeeGo .
Per le necessità di MeeGo , è stato deciso di utilizzare una propria istanza di OBS[26], e non quella messa a disposizione da Novell. OBS viene qui utilizzato per gestire [6]:

il processo di build: dei pacchetti, della distribuzione, e di quant'altro possa essere recuperato da un repository, compilato e pacchettizato (.rpm, .deb);

i pacchetti: un pacchetto corrisponde ad un tarball e delle specifiche per come creare questo archivio;

i progetti: un progetto è una directory virtuale che contiene i pacchetti;

i repository: quando un pacchetto è creato, viene pubblicato su un prepository, da cui può essere scaricato ed installato.

[25] OBS: Open Build Service
[26] https://build.meego.com/

11 Strumenti di gestione dello sviluppo

Per la gestione dello sviluppo del progetto si utilizzano diversi strumenti , che vanno dalla gestione del codice alla gestione dei progetti o dei bug.

11.1 Repository

Per quanto riguarda il SVC[27] è stato scelto **git**. Non è difficle pensare al motivo di questa scelta, dato che il progetto è sotto la custodia della Linux Foundation . Inoltre risulta essere in linea con altri progetti che vengono usati da MeeGo .
Per gestire i repository dei progetti direttamente seguiti da MeeGo ci si appoggia a **Gitorious**[28], che oltre a fornire i repositori per i progetti, fornisce un sistema di notifica, di wiki e di gestione delle richieste di contribuzione ai progetti. Quest'ultima feature risulta essere molto utile in un progetto open source con molti contributori.
Per quanto riguarda invece la navigazione dei repository delle release di MeeGo e degli strumenti di sviluppo, si utilizza un'interfaccia web: `repo.meego.com`.

11.2 Issue tracking system

Come issue tracking system si utilizza **bugzilla**[29]. Il sistema di bug tracking di MeeGo si può trovare qui: `http://bugs.meego.com/`.
Questo ambiente non viene utilizzato solo per il bug tracking, bensì anche per proporre nuove feature o idee per tutto ciò che concerne MeeGo . Infatti come riportato in [30] il processo di proposta dei requisiti è stata resa pubblica.

11.3 Il "Garage" di MeeGo

Il **Garage** di MeeGo dovrebbe essere quel posto dove tutte le applicazioni installabili su MeeGo sono raccolte. Quindi in questo Garage dovrebbero confluire anche tutte le applicazioni sviluppate dalla comunity che sta nascendo.
Come si può vedere in [20], questa parte del progetto è ancora in sviluppo. Tuttavia nella versione 1.0 per netbook c'è già un programma chiamato "Garage" che funge da qualcosa di simile ad un packet manager, dato che si può gestire l'installazione e la disinstallazione di alcune applicazioni precompilate per MeeGo , come AbiWord o Spredsheet.
Ma non e' tutto. Infatti se questo progetto manterra' le funzionalità promesse [20], vi sara' la fusione delle funzionalita' del Garage di Maemo e di Moblin. Questo comportera' la possibilita' di installare non solo le piu' famose ed utilizzate applicazioni, ma anche quelle sviluppate dalla comunità open source.
Inoltre c'e' l'idea di integrare anche l'**OVI store**[30] ed **AppUp**[31], com e gia' successo rispettivamente per Maemo e Moblin. In questo modo probabilmente si riuscirebbe a rimontare la concorrenza schiaccheciante che per strumenti come l'**Android Market** [32] hanno.

[27] SVC: Source Version Control
[28] `meego.gitorious.org`
[29] `http://www.bugzilla.org/`
[30] Centro di gestione delle applicazioni sviluppate da Nokia e la sua comunity, `https://store.ovi.com/`
[31] Centro di gestione delle applicazioni sviluppate da Intel e la sua comunity, `http://www.intel.com/appup/index.htm`
[32] `http://www.android.com/market/`

Riferimenti bibliografici

[1] About meego. http://meego.com/about. Meego about page.

[2] Anssi vanjoki - looking to the future. http://www.youtube.com/watch?v=qHWsi_pnuQ4. Anssi Vanjoki speaks about his vision of future devices and platform.

[3] Anssi vanjoki of nokia (evp markets) on symbian, maemo and meego. http://www.youtube.com/watch?v=C-TqnG7tD94. Anssi Vanjoki speaks about the role Symbian, Maemo and MeeGo will have in the future.

[4] The big merge: A message from intel and nokia. http://www.youtube.com/v/tMLtdnqnhtc&hl. MeeGo overview from Intel and Nokia.

[5] The big merge: A message from the meego technical steering group. http://www.youtube.com/v/tMLtdnqnhtc&hl. MeeGo overview from the Technical Steering Group.

[6] Build infrastructure. http://wiki.meego.com/Build_Infrastructure. Build Infrastructure wiki page.

[7] Developer engagement. http://wiki.meego.com/DeveloperEngagement. Developer Engagement wiki page.

[8] Expressed support for meego. http://www.linuxfoundation.org/node/6144. Linux Foundation support for MeeGo page.

[9] Facebrick - gsoc. http://blog.rburchell.com/2010/06/facebrick-gsoc.html. First request of an external contribution merge.

[10] Getting started with the meego sdk for linux. http://wiki.meego.com/Getting_started_with_the_MeeGo_SDK_for_Linux. Getting started with the MeeGo SDK for Linux wiki page.

[11] Hardware enabling process. http://meego.com/developers/hardware-enabling-process. Hardware Enabling Process page - How to adapt MeeGo to specific hardware.

[12] Linpus adds better power management and social networking support through linpus lite for meego™. http://www.linpus.com/news/2010/06/04.html. Linpus to showcase one of the 1st MeeGo™-based products at Computex 2010 in Taipei.

[13] Linux to garner 33 percent smartphone share by 2015, study says. http://www.linuxfordevices.com/c/a/News/ABI-Research-Mobile-Linux-in-Smartphones/. Linux to garner 33 percent smartphone share by 2015, study says; linuxfordevices.com.

[14] Meego application developer site proposal. http://wiki.meego.com/Application_developer_site/Description. MeeGo Application developer site proposal wiki page.

[15] Meego community office. http://wiki.meego.com/Community_Office. MeeGo Community Office wiki page.

[16] Meego contribution guidelines. http://meego.com/about/contribution-guidelines. MeeGo Contribution Guidelines page.

[17] [meego-dev] issue of enabling accelerometer in sensor framework. http://lists.meego.com/pipermail/meego-dev/2010-June/003104.html. Message in the MeeGo-dev mailing list.

[18] Meego developer story. http://meego.com/developers/meego-developer-story-0. MeeGo Developer Story page.

[19] Meego development process. http://wiki.meego.com/MeeGo_Release_Creation. MeeGo Development Process wiki page.

[20] Meego garage. http://meego.com/garage. MeeGo Garage page.

[21] Meego licensing policy. http://meego.com/about/licensing-policy. Meego licensing policy page.

[22] Meego metrics. http://wiki.meego.com/Metrics. MeeGo Metrics wiki page.

[23] Meego presentation at linuxtag 2010. http://meego.com/community/events/presentations/meego-linux-mobile-devices. MeeGo Presentation at LinuxTag 2010.

[24] Meego quality assurance guidelines. http://wiki.meego.com/Quality. MeeGo Quality Assurance Guidelines page.

[25] Meego tablet on intel moorestown - close-up demo. http://www.youtube.com/watch?v=iVIKYF7MOzU. MeeGo Tablet on Intel Moorestown - Close-up Demo at Computex 2010 in Taipei. See also http://www.youtube.com/watch?v=L_J5FMXF99I.

[26] Meego technical overview. http://meego.com/community/events/presentations/meego-technical-overview. MeeGo Presentation.

[27] Netbooks still hot, but tablets starting to cut in, says study. http://www.linuxfordevices.com/c/a/News/DisplaySearch-2q-2010-notebook-study/. Netbooks still hot, but tablets starting to cut in, says study; linuxfordevices.com.

[28] Novell announces support for meego. http://www.novell.com/news/press/novell-announces-support-for-meego. Novell Announces Support for MeeGo page.

[29] Obs. http://wiki.meego.com/OBS. OBS wiki page.

[30] Opening meego requirements. http://meego.com/community/blogs/samipienimaki/2010/opening-meego-requirements. Opening MeeGo requirements blogs page.

[31] Public support for meego. http://meego.com/about/public-support-meego. Public support for MeeGo page.

[32] Suse meego. http://www.youtube.com/watch?v=4r5OOf3Kvls. SUSE MeeGo concept video demonstration.

[33] Welcome to meego. http://meego.com/community/blogs/imad/2010/welcome-meego. First official MeeGo announce.