UNIVERSITÀ DEGLI STUDI DI PADOVA
Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Magistrale in Informatica

Analisi del progetto Open Source "django"

di Andrea Zilio

Corso di Tecnologie Open Source
Anno accademico 2009 - 2010

Chapter 1

Introduzione e storia

Django è un framework per lo sviluppo di applicazioni web nato nel 2003, sviluppato in Python e diventando Open Source qualche anno più tardi.

Gli obiettivi che questo framework intende assolvere, come dichiarati da uno dei suoi creatori, sono essenzialmente i seguenti:

- *Make web development stupidly fast*
- *Automate ripetitive tasks*
- *Follow best practices*
- *Support easy maintainability*

Django fa quindi parte della lunga lista di web framework che un web developer oppure il reparto IT di una azienda deve scegliere nel momento in cui si renda necessaria la creazione di un sito web: dal più semplice sito vetrina, ad una complessa web application.

1.1 Contesto e nascita

Tutto iniziò nel 2003, a *World Online*, la divisione Web di *The World Company* che pubblicava il quotidiano locale della piccola città di Lawrence in Kansas (circa 100 000 abitanti).

The Lawrence Journal, questo il nome del giornale, è stato storicamente un quotidiano sempre attento all'innovazione e fu il primo giornale ad avere sia un'edizione cartacea che un'edizione online, dove era l'edizione cartacea a derivare dai contenuti pubblicati il giorno prima da quella online e non il contrario.

Il reparto IT di questo giornale ha quindi sempre avuto un ruolo fondamentale ed aveva sviluppato inizialmente un Content Management System in PHP per gestire i contenuti delle edizioni del quotidiano.
La necessità di implementare nuove funzionalità per una più ricca e completa interazione con gli utenti, al fine di craeare una reale comunità online attorno

al quotidiano portarono alla creazione di un nuovo framework più comlpesso e modulare del precedente CMS e una certa insoddisfazione degli sviluppatori rispetto al linguaggio di scripting PHP (soprattutto riguardo alla manutenibilità del codice) portò questo nuovo framework ad essere sviluppato in Python.

Nel 2003 lo sviluppo di Django iniziò.

Nato nel mondo del giornalismo e dipendendo quindi le feature e le edizioni online del quotidiano interamente dallo sviluppo delle relative feature tramite Django, il principale scopo fu quello di adeguarsi ai tempi del news reporting e quindi di dotarsi di un framework che permettesse lo sviluppo più rapido possibile di nuove funzionalità.

Nell'anno 2005 un altro framework per lo sviluppo di applicazioni web (Ruby On Rails) iniziò ad ottenre crescente attenzione e successo e questo spinse gli sviluppatori di Django a uscire allo scoperto anche con il proprio framework sperando di ottenere una risposta simile dalla comunità. Django venne quindi presentato nel Marzo di quello stesso anno all'annuale conferenza di Python (PyCon) ottenendo la risposta sperata.

Pochi mesi più tardi il framework venne quindi rilasciato con licenza BSD, la versione 1.0 stabile seguì 3 anni dopo e da quella data i rilasci si susseguono ogni 9 mesi circa.

Il CMS usato dal quotidiano e basato appunto sul framework rilasciato con licenza open source continua invece ad essere sviluppato e ad essere commercializzato con licenza proprietaria.

Risulta probabilmente utile citare Jacob Kaplan-Moss, uno dei creatori di Django, per comprendere l'importanza di un framework che permetta di stare al passo con le deadline dei giornalisti.
Di seguito è riportata parte di un suo intervento ad una conferenza in cui spiega come alcune funzionalità di Django permettano di ridurre i tempi di sviluppo delle feature a poche ore. Nello specifico parla dell'importanza di avere un'interfaccia di amministrazione generata automaticamente e *production ready*: una delle principali feature di Django presente fin dalla prima versione.

> I can't tell you how much time it saves us: it's insane! We can simultaneously have our editors upload information on the site as we are writing the public controllers and views.
> It takes five minutes to knock out a model and then they can get to work.
>
> Let me better explain with an example.
>
> The Kansas State government released a week ago a list of salaries of state employees making over a hundred thousand dollars and we found out that the list was released at 10.30 PM on a Sunday night. Press time is 11.00 PM [so that didn't make it into the paper edition,] but it was not very long and the reporters said that we should have published something on the online edition.

At about 11.00 PM one of the developers started to work on a model for the data, had those models done about ten minutes later, reporters started entering the data through the administrative interface and by the time they had done entering it we already had ready a section in the site which enabled you to browse and query any salary of the employers, searching by who they work for, etc.
We launched that feature at midnight and we got close to a hundred thousand viewers the first day.

That's what I mean making web development stupidly quick.
These are the deadlines journalists work on, and we don't get an exception from that just because we work with bits instead of ink.

Chapter 2

Market, Licenza e Business Model

Esistono principalmente due tipologie di applicativi destinati allo sviluppo di siti web:

- *Web Content Management Systems*
- *Web Application Frameworks*

I *Web Content Management Systems*, o WCMS, sono applicativi utili a pubblicare velocemente siti web con caratteristiche diffuse e standardizzate cercando di limitare il più possibile la necessità di comprendere il funzionamento delle tecnologie alla base di Internet. Il loro scopo è di permettere di concentrarsi sui contenuti di un sito e solitamente, anche se permettono di implementare o estndere funzionalità (generalmente frontend) tramite moduli o plugin, difficilmente permettono di modificare la struttura del CMS stesso che va a riflettersi sulla struttura del sito.

I *Web Application Frameworks* seguono un approccio differente: il loro target sono sviluppatori che conoscono le tecnologie per lo sviluppo web e che, per avere un maggiore controllo, hanno intenzione di sviluppare le funzionalità del sito o applicazione web avvalendosi però di moduli, librerie o strumenti che possano velocizzarne il lavoro. Spesso i WCMS sono sviluppati tramite *Web Application Framework*.

La categoria dei *Web Application Framework* è decisamente vasta e comprende, ad esempio, sia soluzioni come CakePHP che come Apache Struts.
Questi framework hanno evidentemente un differente target di utenti: CakePHP è probabilmente più adatto a web agency o sviluppatori interessati a creare siti o applicazioni web di dimensioni medie o medio grandi; Apache Struts ha invece un target rivolto ad un segmento enterprise che deve rispondere a requisiti più complessi.
Forzando questa distinzione ci è quindi possibile dividere la categoria dei *Web Application Framework* in due sottocategorie: *Enterprise Web Application Frameworks* e *Not Enterprise Web Application Framework*.

Questa categorizzazione, benchè eccessiva e forzata, ci permette di identificare quello che può essere il target di utenza e quindi i concorrenti del framework che stiamo anlizzando.
Django è infatti senza dubbio inquadrabile come un *Not Enterprise Web Application Framework* che però sta cercando sempre più di rispondere anche ad alcuni requisiti generalmente lasciati agli *Enterprise Web Application Framework* senza abbandonare la propria relativa semplicità d'uso e configurazione.

2.1 Market

Dalla categorizzazione appena data segue dunque anche il target ed il potenziale mercato di questo framework.

I principali utenti di un framework come questo sono singoli web developer, web agency e aziende ineteressate a sviluppare servizi o applicazioni web anche di complessità media o medio-alta.
Ad esempio Django è attualmente utilizzato da molti giornali online medio-piccoli, da alcune sezioni di quotidiani più conosciuti come The Guardian o Washington Post e da un social network che conta oltre due milioni di utenti.

Il principale competitor di Django, essendo nato anche nello stesso anno, è Ruby On Rails che però richiede di imparare un linguaggio altrimenti scarsamente usato come Ruby. Nonostante questo Ruby On Rails è comunque più conosciuto di Django.

Uno svantaggio, ma come vedremo più avanti anche un vantaggio, può essere dovuto alla scelta di sviluppare Django in Python.
Il più comune linguaggio di programmazione server side per il web è infatti PHP e quindi, anche se Django risulta essere spesso superiore rispetto a molti altri framework scritti in PHP, resta difficile convincere professionisti che già hanno sviluppato la loro esperienza con un linguaggio di programmazione a cambiare rotta.

2.2 Business Model

Possiamo identificare tre attori che creando business tramite Django permettono il continuo sviluppo del framework.

2.2.1 I creatori del framework

Come abbiamo visto Django è nato a *World Online*, la divisione web di *The World Company*, che ha svilupapto questo framework per realizzare il Web Content Management System usato dalle edizioni online dei propri giornali.

Dei due prodotti (Django e il WCMS) solamente il framework è stato rilasciato con licenza Open Source nel 2005, mentre il WCMS è rimasto un prodotto proprietario che continua ad essere sviluppato e commercializzato sotto il nome

di Ellington CMS.

Ellington CMS è quindi un prodotto destinato a quotidiani medio-piccoli che ha riscosso un discreto successo; viene attualmente commercializzato da *Mediaphormedia LLC* (divisione software di *The World Company*) ad un costo di circa 15000$ a licenza (dato di tre anni fa).

Essendo Ellington CMS basato totalmente su Django, *Mediaphormedia* è certamente interessata a seguire lo sviluppo del framework per poter incidere sulla roadmap del progetto anche a proprio vantaggio.

2.2.2 I consulenti

Alcuni dei principali sviluppatori di Django riescono a sfruttare la competenza acquisita del framework come consulenti.

Alcuni esempi sono:

- Jacob Kaplan-Moss, ex lead developer di Ellington CMS, che,tramite la società *Revolution Systems*, fornisce supporto, consulenza e training ad aziende che utilizzano Django.
- James Tauber, creatore di Pinax, un CMS general purpose open source basato su Django, fornisce supporto ad aziende che sfruttano Django e/o Pinax per i propri servizi web
- Jeff Croft, ex web developer e designer a *World Online*, svolge training e workshop online destinati ad aziende.

2.2.3 Gli utilizzatori

Probabilmente la maggioranza dei *contributors* di Django sono semplici web developer, spesso free-lance, che, svilluppando i propri siti sfruttando questa tecnologia, sono ben disposti a contribuire allo sviluppo del framework e ad implementare le funzionalità che spesso ritengono necessarie.
In questo modo acquisiscono una profonda conoscenza del prodotto che permette loro di mantenersi aggiornati rispetto al proprio strumento di sviluppo e di poter incidere sulle scelte e sulle funzionalità future anche in base alle proprie necessità.

2.3 Licenza

Django è stato rilasciato nel 2005 con licenza BSD-new.

Grazie a questa licenza è possibile utilizzare e modificare il framework per qualsiasi scopo, a patto che in caso di redistribuzione del sorgente o di un formato binario vengano riportate le condizioni della licenza, venga mantenuta l'indicazione del copyright e che il nome del detentore del copyright non venga utilizzato per pubblicizzare o sostenere derivati.

E' chiaro che anche la scelta della licenza ha contribuito al successo di questo framework: una licenza permissiva come la BSD-new permette infatti anche l'utilizzo del framework in prodotti derivati proprietari. Il caso peggiore sarebbe stato quello di rilasciare il software con licenza AGPL che avrebbe obbligato la pubblicazione dell'intero codice sorgente a chiunque avesse realizzato un servizio web tramite Django.

La licenza BSD-new (o Modified BSD) è inoltre GPL compatibile in quanto la discussa clausola della versione originale della licenza, che imponeva di riportare in materiale pubblicitario e durante l'uso di software derivato l'indicazione del copyright holder originale ("This product includes software developed by ..."), è stata rimossa.

Chapter 3

Community

Procediamo ad analizzare la comunità di Django rispetto ad alcune delle caratteristiche identificate nella ricerca "*Open Source Community Building*" pubblicata dall'Università di Berna.

3.1 Comportamento e capacità tecniche della leadership

3.1.1 Il Team di sviluppo

Per parlare del team di sviluppo di Django introduciamo i principali componenti:

- Adrian Holovaty: Web developer con laurea in giornalismo. Lavorando per oltre due anni a World Online ha sviluppato Django fin dall'inizio.
- Jacob Kaplan-Moss: partner in *Revolution Systems* (azienda che fornisce supporto e training alle aziende che utilizzano Django). Ha contribuito anche lui a sviluppare il framework sin dai primi giorni nella divisione web di *The World Company*.
- Simon Willison: un rispettato web developer inglese che ha dato il via al progetto Django a *Wolrd Online*.
- Wilson Miner: Web Designer che si occupa dell'interfaccia grafica del sito web e dell'interfaccia amministrativa di Django
- James Bennett: Release Manager di Django, si occupa anche della documentazione. E' un web developer che lavora attualmente in *The World Company* dove continua a sviluppare Ellington CMS.

Questo è il team principale che segue lo sviluppo di Django, coloro che hanno i diritti di scrittura nel repository sono in totale 21.

La leadership vera e propria del progetto può essere identificata nelle due persone che si definiscono i *Benevolent Dictator For Life* del progetto: Adrian Holovaty e Jacob Kaplan-Moss.
Entrambi sono persone con molta esperienza e con ottime competenze tecniche

nell'ambito del web development.
Hanno più volte dimostrato un ottima capacità decisionale e di rappresentare una leadership sufficientemente carismatica da mantenere vivo l'interesse della comunità rispetto ai valori del progetto.

Il tempo che dedicano al progetto è destinato principalmente ai contributor: fanno la loro comparsa molto di rado nella mailing list destinata agli utenti del framework, ma seguono molto più da vicino gli sviluppatori di Django intervenendo spesso e volentieri nelle discussioni.
Questo non deve però far pensare che gli utenti siano lasciati soli, ci sono infatti molti altri contributor che seguono e aiutano i nuovi utenti dedicando il loro tempo ad aiutare e rispondere alle domande dei nuovi arrivati.

Più volte si è reso necessario spiegare alcune scelte fatte nel tempo (spesso rispetto alla retrocompatibilità o alla scelta di un linguaggio di templating limitato) e i due "*benevoli dittatori a vita*" si sono sempre mostrati pazienti e disponibili a comunicare i motivi di tali scelte.

Sia Jacob Kaplan-Moss che Adrian Holovaty mantengono una visione d'insieme su tutto il progetto (senza ritagliarsi solamente compiti precisi o entro una certa area) e una delle problematiche che loro stessi evidenziano è la lentezza della comunità nel rispondere alle segnalazioni degli utenti.
Spesso i bug segnalati completi di patch ricevono in effetti poca visibilità e rimangono spesso in attesa anche per una decina di mesi prima di essere inclusi nel repository. Questo è in genere dovuto alla scarsa o non sufficiente qualità dell'implementazione della soluzione o alla mancanza di test di unità o documentazione. In seguito a queste mancanze però spesso la segnalazione rimane in stallo per molti mesi senza provvedere al necessario adeguamento in modo sufficientemente rapido.

3.2 Precondizioni del progetto

Uno dei principali motivi del successo di Django è dovuto all'ottima qualità del codice.

Nel momento del rilascio pubblico di Django nel 2005 questo era già un framework in uno stato avanzato dello sviluppo essendo stato utilizzato nei due anni precedenti da *The World Company*.
Di conseguenza la qualità, la struttura di base e gli obiettivi del framework erano stati quindi già decisi, la roadmap era chiara e quindi i contributor potevano iniziare a lavorare su una base solida e ben strutturata per aggiungere le funzionalità mancanti che già erano state immaginate e previste.

La scelta di Python come linguaggio di programmazione poteva sembrare azzardata, ma in realtà è stata la risposta ad un target di utenti che stava crescendo sempre più, ovvero gli sviluppatori PHP delusi.
Gli stessi leader del progetto dichiarano di essere sopresi da quante persone abbiano abbandonato PHP per passare a Django e che abbiano fatto questo scelta nonostante la necessità di prendere confidenza con un nuovo linguaggio di programmazione.

Del resto cambiare linguaggio di programmazione era esattamente la strada che molti sviluppatori PHP volevano intraprendere e la scelta di Django ha dato loro la possibilità di seguirla.

Nel 2005 inoltre non erano nati o non erano ancora molto diffusi framework PHP paragonabili a Django e quindi molte persone hanno preferito abbracciare una nuova tecnologia come Python (scegliendo appunto Django) oppure preferendo Ruby On Rails.

Django e Ruby On Rails hanno avuto un altro vantaggio: sono stati sviluppati proprio nel periodo in cui le applicazioni web iniziavano ad aumentare esponenzialmente la propria complessità e di conseguenza nella comunità dei web developer cresceva la necessità di utilizzare dei framework per velocizzarne lo sviluppo.
Entrambi i framework sono stati una risposta a questa necessità, e, nonostante Ruby On Rails avesse ottenuto un vantaggio iniziale rispetto al concorrente, l'attenzione verso Django è cresciuta costantemente.

3.3 Gestione della comunità

Django è un framewok estremamente modulare ed in questo modo permette anche a chi non partecipa direttamente al suo sviluppo di partecipare comunque alla sua comunità realizzando plugin o applicazioni di cui tutti possano beneficiare.
A questo scopo sono disponibili molti siti web distinti che raccolgono i vari contributi. Uno fra tutti è *djangosnippets.com*, ma la maggior parte dei plugin o applicazioni sono disponibili nei siti di sviluppo collaborativo come Google Code o Git Hub.

Una nota negativa riguarda i canali di comunicazione che sono abbastanza scarsi (sono disponibili solamente le mailing list degli utenti e degli sviluppatori) e, sebbene il progetto abbia un'ottima documentazione, sarebbe probabilmente utile sfruttare in modo più propizio e organizzato la wiki disponibile sul sito che attualmente conta pochi contenuti e non dispone di una struttura adeguata.

Una nota positiva riguarda invece l'attenzione posta rispetto alla retrocompatibilità delle *API*. La documentazione dice esplicitamente che ogni *API* non interna sarà supportata in ogni Minor Release.
Questo estratto del codice sorgente del framework fa capire come l'attenzione rispetto a questo tema sia effettivamente alta:

```
# We need backwards compatibility with code which spells it this way:
#   my_view = cache_page(my_view, 123)
# and this way:
#   my_view = cache_page(123)(my_view)
# and this:
#   my_view = cache_page(my_view, 123, key_prefix="foo")
# and this:
#   my_view = cache_page(123, key_prefix="foo")(my_view)
# and possibly this way (?):
#   my_view = cache_page(123, my_view)
```

3.3.1 Django Software Foundation

Nel Marzo 2008, 3 anni dopo il rilascio pubblico di Django, è nata la *Django Software Foundation*, una fondazione no profit con i seguenti scopi dichiarati:

- Support development of Django by sponsoring sprints, meetups, gatherings and community events.
- Promote the use of Django among the world-wide Web development community.
- Protect the intellectual property and the framework's long-term viability.
- Advance the state of the art in Web development.

La fondazione accetta donazioni da privati e aziende ed il *board of directors* è composto da:

- Adrian Holovaty
- Jacob Kaplan-Moss
- Dan Cox

I primi due nomi dovrebbero esserci familiari: sono i due creatori del framework che però partecipano come rappresentanti delle due società di cui fanno parte: EveryBlock (servizio online acquistato ad MSNBC basato su Django) e Revolution Systems (azienda che fornisce supporto e training commerciale alle aziende che adottano Django).
Dan Cox è invece il rappresentante di *Mediaphormedia LLC*, l'azienda che sviluppa e commercializza Ellington CMS.

I fondi provenienti dalle donazioni libere e dall'interesse delle tre società, che già abbiamo analizzato nei capitoli precedenti, permettono alla fondazione di organizzare eventi come i Django Sprints (incontri fra sviluppatori per favorire lo sviluppo di Django) e, soprattutto, due annuali conferenze per la comunità che ruota attorno al framework: DjangoCon US (negli Stati Uniti) e DjangoCon EU (in Europa).

Le due conferenze erano state chieste a gran voce dalla comunità già prima della nascita della fondazione e fino dalle prime edizioni hanno portato gli sviluppatori del framework a conoscersi agevolando lo scambio di opinioni e l'ambiente ideale per favorire l'inclusione di nuove funzionalità nel progetto.

Per poter gestire e distribuire il codice del framework senza restrizioni e secondo la licenza scelta dalla comunità, ogni contributor deve firmare un documento analogo a quello utilizzato dalla *Apache Software Foundation* per i contributor dei propri progetti.
Si legge infatti sul sito della fondazione:

> Every contributor of non-trivial amounts of code (more than just a line or two) to Django will be required to sign such a document.
> If somebody is unable to sign the document, their contribution (whether it be code, or documentation or string translations) will need to be removed from Django.

Chapter 4

Processo di Sviluppo

4.1 Release process

Lo sviluppo di Django segue un rilascio di Minor Release ogni 9 mesi circa: dopo ogni release rilasciata il core development team esamina le proposte per la prossima release e fissa una deadline.

Ogni ciclo di relase è quindi diviso in tre periodi:

Feature Proposal
: In questa fase vengono analizzate le funzionalità proposte per la release successiva. Queste devono essere adeguatemnte descritte e almeno una prima fase di design deve essere iniziata. Successivamente ognuna di queste funzionalità viene categorizzata come "Must have", "Maybe" e "Not Going To Happen" (cioè da rimandare a release future).

Development
: Come il nome potrà far immaginare, la fase successiva è quella che, seguendo la roadmap definita precedentemente, sviluppa le varie feature. La fase termina con una versione alpha della release in sviluppo.

BugFixes
: In questo periodo nessuna funzionalità può essere aggiunta e ci si occupa solamente di correggere i bug identificati grazie all'alpha release.
Una versione beta viene rilasciata circa a metà di questa fase mentre la Release Canidate viene resa disponibile due settimane prima della scadenza prevista per la versione stabile.

4.2 Gestione dei ticket

La documentazione del progetto descrive in maniera molto chiara come contribuire allo sviluppo di Django, come segnalare bug, proporre funzionalità o cambiamenti e come ognuno di questi ticket verrà gestito dalla comunità.

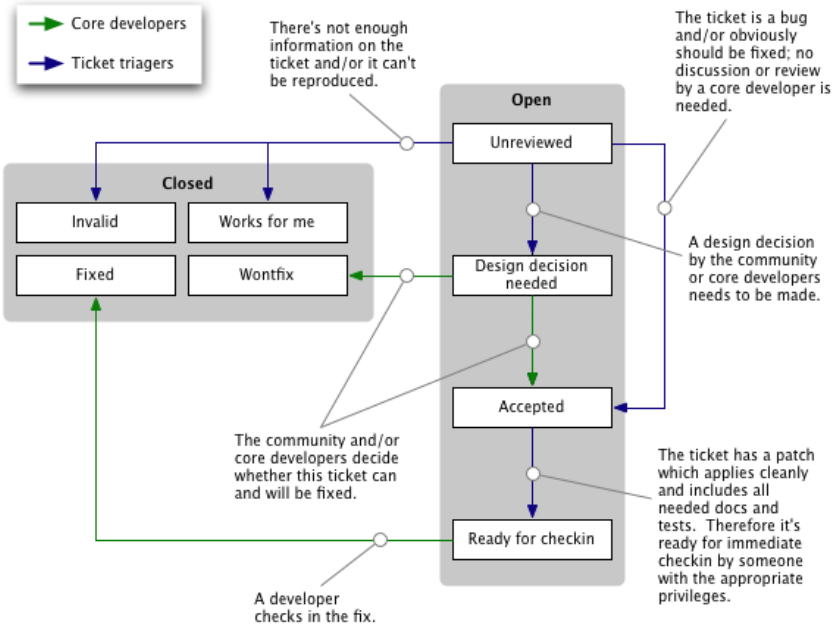
In seguito all'apertura di un ticket un developer o qualsiasi contributor può accettare tale ticket e inviare una patch che dovrà rispondere a certe caratter-

istiche per essere accettata ed inclusa nel codice.

Per poter quindi indentificare lo stato di un ticket vengono utilizzati dei campi speciali che, assieme, permettono di riconoscere la complessità, la necessità di intervento da parte di un core developer e alla qualità della patch di ogni ticket.
I campi speciali che vengono utilizzati sono:

- Triage Stage (che ha tre possibili valori: “Unreviewed”, “Design Decision Needed”, “Accepted”, “Ready For Checkin”)
- Has Patch
- Needs Documentation
- Needs Tests
- Patch Needs Improvement

Il seguente schema, disponibile sul sito del progetto, spiega nel dettaglio come ogni ticket venga valutato in base al suo stato e quali persone fra i Ticket Triagers (ovvero un insieme di cotributors fidati che possono effettuare delle semplici decisioni sui ticket) e i Core Developers interverranno in ogni fase:



Senza entrare nel dettaglio del diagramma, è importante notare come sia ben definito e limitato l’intervento dei core developer nelle decisioni dei singoli ticket.
Sono infatti i Ticket Triagers a segnalare un ticket con “Design Decision Needed” per richiedere il supporto di un core developer. Questo, in una comunità con centinaia di contributors e migliaia di ticket permette ai Core Developers di concentrarsi sulle cose importanti.

Risulta inoltre evidente come questa gestione dei ticket dia particolare importanza alla qualità del codice.
In seguito all'invio di una patch i Ticket Triagers si occupano di valutarne la qualità (impostando "Patch Needs Improvement") ed inoltre, prima che la patch possa essere inclusa nel repository, il ticket deve avere almeno uno unit test associato ("Needs Tests").
La stessa restrizione vale per la documentazione: se i Ticket Triagers ritengono che un determinato ticket abbia bisogno di estendere la documentazione (ad esempio per descrivere una nuova funzionalità, un nuovo parametro o un caso particolare), questa dovrà essere aggiornata prima di poter applicare le modifiche al repository.
In questo modo è sempre disponibile non solo la documentazione delle versioni stabili del framework, ma anche la documentazione completa della versione in via di sviluppo.

E' probabilmente dovuto anche a questa gestione dei task il fatto che Django sia riconosciuto per essere un framework di ottima qualità e con una documentazione superiore alla media.

Chapter 5

Osservazioni

Nonostante il discreto successo ottenuto da Django, una delle caratteristiche a mio avviso più interessanti non viene adeguatamente pubblicizzata nè indicata come punto di forza.
Con Django è possibile realizzare applicazioni web con un livello di modularità più elevato rispetto ad altri framework destinati allo stesso target di utenti.

Un sito web realizzato in Django si compone di più applicazioni che implementano le singole funzionalità di un sito.
Ad esempio, per realizzare un blog con le funzionalità di base (leggere gli articoli, commentarli e votarli) le applicazioni di cui il progetto si può comporre potrebbero essere:

- Comments, per gestire i commenti
- Ratings, per gestire le votazioni
- Articles, per gestire gli articoli

Sebbene questa possa sembrare la struttura di un blog qualunque, realizzandolo con molti framework concorrenti a Django sarà sempre necessario collegare e rendere quindi in parte dipendenti le applicazioni elencate a livello di database, di codice python (o php) o di configurazione.
Tramite gli strumenti forniti da Django è invece possibile realizzare le tre applicazioni in modo completamente indipendente, cioè con un grado di accoppiamento nullo.

Con framework come CakePHP e CodeIgniter (e probabilmente nemmeno con Ruby On Rails o Symfony) non è possibile raggiungere lo stesso grado di moduluarità con la stessa semplicità.

Benchè con Django ciò sia possibile (esiste infatti un'applicazione per i commenti sviluppata in questo modo dai core developers), le best practices da seguire e i pattern da implementare per permettere una modularità così spinta non sono documentati e l'unico modo per apprenderli consiste nello studiare il codice del framework o nel seguire le presentazioni di alcuni developers.
La mancanza di sufficiente documentazione su questo fronte è secondo me una lacuna che dovrà essere colmata.