



UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA  
FACOLTÀ DI SCIENZE MM.FF.NN. - VARESE  
CORSO DI LAUREA IN INFORMATICA

# **Agile Tracking**

*Relatore:*  
Prof. Matteo Vaccari  
*Correlatore:*  
Federico Gobbo

*Tesi di Laurea di:*  
DOMENICO MARINI  
*Matricola:* 608483

**Anno Accademico 2006-2007**

*Domenico Marini*

*Alla mia famiglia,  
che mi ha sostenuto fin dall'inizio  
e ha sempre creduto in me.*

<b>Introduzione</b> .....	4
<b>CAPITOLO I</b> .....	7
<b>Analisi generale del problema</b> .....	7
Metodologie di sviluppo software.....	8
Metodologie Agili.....	12
User Stories.....	13
La Tecnica del pomodoro.....	15
<b>CAPITOLO II</b> .....	17
<b>Retrospective Agili</b> .....	17
Le Retrospective.....	18
Un Modello da seguire.....	19
<i>Preparare l'ambiente</i> .....	19
<i>Raccolta dati</i> .....	19
<i>Generare intuizioni</i> .....	20
<i>Decidere cosa fare</i> .....	21
<i>Chiudere la retrospettiva</i> .....	21
Agile Tracking e le retrospettive.....	22
<b>CAPITOLO III</b> .....	26
<b>Analisi di mercato</b> .....	26
Acunote.....	27
Mingle.....	30
Easy Track.....	32
<b>CAPITOLO IV</b> .....	35
<b>Agile Tracking</b> .....	35
Introduzione.....	36
Funzionamento.....	36
Ajax.....	42
Gemme e Plugin.....	43
<i>Red Cloth</i> .....	43
<i>Acts as Attachment</i> .....	44
<i>Acts as Authenticated</i> .....	44
<i>Gruff</i> .....	44
<b>Conclusioni</b> .....	46
<b>Bibliografia</b> .....	47
<b>Sitografia</b> .....	49

# Introduzione

*La perfezione dell'uomo  
consiste proprio nello scoprire  
le proprie imperfezioni.*

*Sant' Agostino*

Durante lo sviluppo di un software, sia esso svolto da un team o da un singolo programmatore è di grande importanza sapere in ogni momento quale sia la situazione globale dell'intero progetto, soprattutto nel caso in cui questo sia di grandi dimensioni. Ad oggi esistono diverse tecniche e metodologie sia per quanto riguarda la creazione e la gestione di un progetto software che per il suo monitoraggio.

Tra le varie metodologie di sviluppo software, troviamo le metodologie agili, un insieme di tecniche di sviluppo, amministrazione e gestione sia di progetti che di team che hanno come fine comune quello di apportare un aumento della produttività e dell'efficienza di un gruppo di lavoro. Proprio questo aspetto ha fatto in modo che molte società informatiche abbiano deciso di inserire queste tecniche all'interno dei loro metodi per riuscire ad aumentare il rendimento.

Una di queste tecniche, *Extreme Programming*, sebbene nata solamente nel 1999, sta riscontrando un notevole successo, ha portato alla nascita di quelli che vengono definiti gli *xp-ug* ovvero gli *Extreme Programming User Group*, team di sviluppatori che adottano tecniche come la

programmazione a coppie (*pair programming*), la verifica continua del programma durante lo sviluppo per mezzo di programmi di *test* e il frequente *refactoring* del software, nonché tecniche di time boxing.

Ad esempio l'eXtreme Programming User Group di Varese, durante lo sviluppo di Examinando ha adottato un *wiki* per monitorare l'andamento del progetto. Sebbene il *wiki* sia un potentissimo strumento di scrittura collaborativa, in grado di agevolare la cooperazione tra gli individui che lo sfruttano, si è rivelato inadatto per le esigenze dell'xp-ug poiché non permetteva una gestione appropriata del progetto ed in particolare delle user stories. Questo svantaggio creava rallentamenti nello sviluppo e soprattutto rendeva sempre meno chiara la situazione di ogni storia e quindi dell'intero progetto, diminuendo di conseguenza la velocità di tutto il team. In tal senso sono state analizzate alcune delle soluzioni attualmente disponibili sul web come ad esempio *Assembla*. Tuttavia allo stato dell'arte non sono stati individuati strumenti adatti che permettessero loro una gestione semplice ed immediata del progetto. Proprio per questa reale esigenza è nato *Agile Tracking* che riesce ad offrire nello stesso tempo semplicità e professionalità mettendo a disposizione dell'utente utilizzatore del servizio una serie di strumenti validi per il monitoraggio di uno o più progetti e la gestione delle relative storie ad essi collegate, con la possibilità di assegnare ad ogni storia un'immagine (*Cap. IV*) che potrebbe essere la scansione digitale della storia stessa, precedentemente scritta nel consueto formato cartaceo. Inoltre fornisce una sorta di bacheca dove le storie vengono visualizzate come una collezione di anteprime suddivise per iterazione. Qui le storie possono essere marcate con dei colori per identificarne lo stato o semplicemente per attribuire un valore particolare

secondo le convenzioni del team.

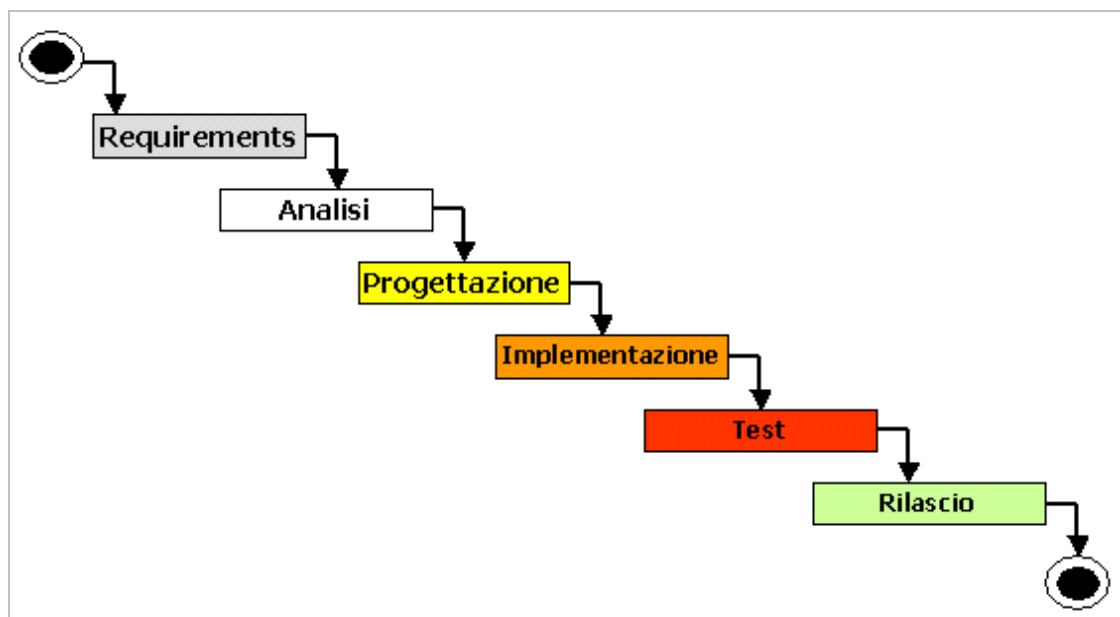
Agile Tracking, quindi, in linea con le ideologie e le tecniche agili, si pone come una valida applicazione che permette a chiunque di utilizzare tecniche *XP* per il monitoraggio di un progetto software in modo chiaro e molto intuitivo.

# **CAPITOLO I**

## **Analisi generale del problema**

## Metodologie di sviluppo software

Tra la fine degli anni sessanta e l'inizio della decade successiva, avvenne un notevole mutamento nella concezione nello sviluppo di software, che passò da quello artigianale, ovvero affidato alla libera creatività dei singoli individui, ad un approccio più industriale in cui la creazione di programmi e sistemi software viene considerata come un processo complesso che richiede pianificazione, controllo, e documentazione appropriata. Questo cambiamento fu inevitabile se si pensa a come, nel corso degli anni, sia aumentata la complessità dei sistemi e sia nato un vero e proprio mercato del software basato su nuovi e più stringenti requisiti di qualità, legati per esempio all'uso di sistemi informatici in contesti critici (centrali energetiche, sistemi aerospaziali, armamenti e così via). Iniziarono così a svilupparsi le prime metodologie di sviluppo software che secondo una suddivisione recente, potrebbero essere divise in tre grosse categorie, quelle definite pesanti, quelle iterative e quelle agili.



**Figura 1:** Metodologia di sviluppo a cascata [18]



Alla prima categoria appartiene quello che viene chiamato il *modello a cascata* (*waterfall model* o *waterfall lifecycle* in inglese), il primo modello di ciclo di vita del software. Esso rappresenta innanzitutto un importante mutamento di prospettiva nella pratica dello sviluppo del software, abbandonando definitivamente la programmazione per tentativi ed errori (*code and fix*) e riconoscendola finalmente come processo industriale vero e proprio.

In questo tipo di modello, il processo di sviluppo è diviso in fasi sequenziali, ogni fase produce un output che è usato come input nella fase successiva e, molto importante, ogni fase viene documentata. Si incontrano quindi nell'ordine dapprima uno studio di fattibilità seguito dall'analisi dei requisiti e a seguire, progettazione, collaudo, integrazione e manutenzione. Esso ha fissato due concetti fondamentali: il processo di sviluppo del software deve essere soggetto a disciplina e pianificazione; l'implementazione del prodotto deve essere rimandata fino a quando non sono perfettamente chiari gli obiettivi.

Questo modello ebbe un enorme successo negli anni settanta ma a partire dagli anni ottanta fu soggetto a profonde critiche e revisioni

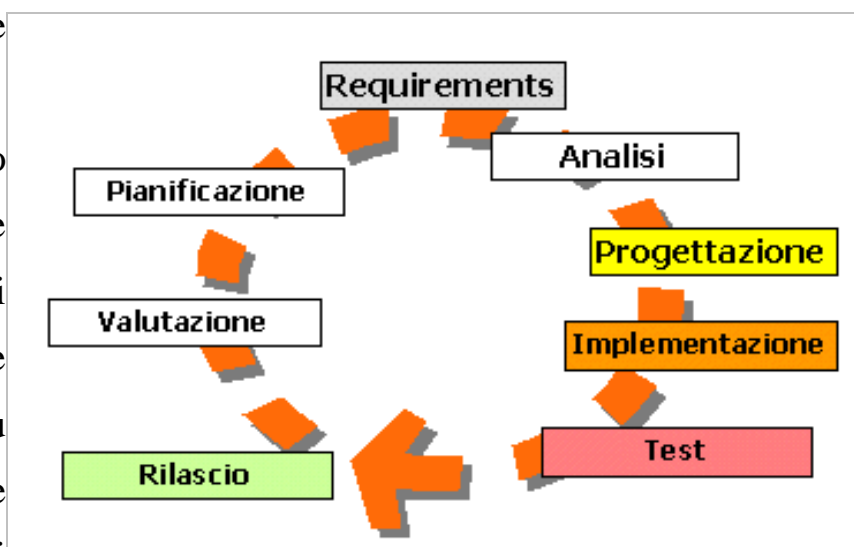


Figura 2: Metodologia di sviluppo iterativa [18]

soprattutto dovute all'evoluzione del software stesso e dei linguaggi di programmazione. Nonostante tutto, ancora oggi viene più spesso associato

alla programmazione procedurale e strutturata ed è spesso il primo modello di sviluppo software che si insegna agli studenti.

La metodologia iterativa propone un modello in cui sono presenti le stesse fasi del modello a cascata ma queste invece di giacere su una linea retta, si trovano su una circonferenza e l'ultimo passo innesca uno nuovo riciclo (dalla fase di test si torna a quella di pianificazione e così via).

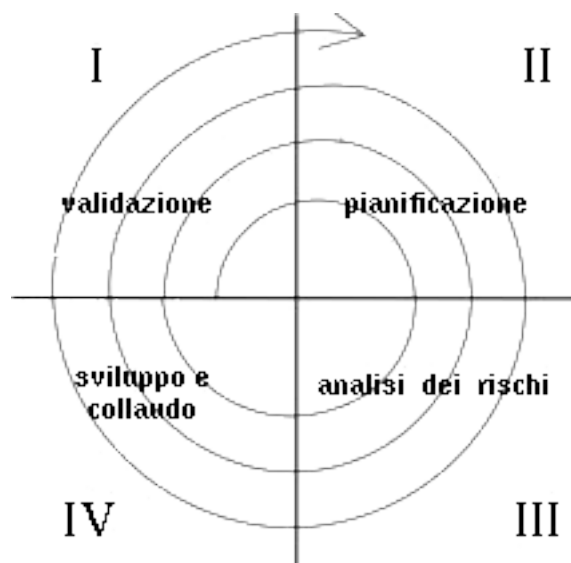


Figura 3: Modello a spirale[17]

Tra le metodologie iterative invece troviamo il modello a spirale il quale, consentendo di rappresentare i diversi cicli di vita di un software, può essere visto come un *metamodello*. Questo modello si basa sul concetto di rischio ovvero un insieme di circostanze avverse che posso pregiudicare il processo di sviluppo e la qualità del software. Esso si concentra

sull'identificazione e l'eliminazione di problemi ad alto rischio, tralasciando quelli banali. La caratteristica principale di questo modello è di essere ciclico e non lineare. Ogni ciclo di spirale si compone di quattro fasi, il raggio rappresenta il costo accumulato e la dimensione angolare il progresso nel processo.

La prima fase identifica gli obiettivi e le alternative, poi le alternative si valutano nella seconda fase in cui vengono evidenziate le potenziali aree di rischio. La terza fase consiste nello sviluppo e nella verifica del prodotto, infine la quarta fase consiste nella revisione dei risultati delle tre fasi precedenti.

Da ultima troviamo le metodologie agili. Questo particolare metodo di sviluppo software, a differenza dei precedenti, coinvolge quanto più possibile il committente, ottenendo in tal modo un'elevata reattività alle sue richieste.

La gran parte dei metodi agili tentano di ridurre il rischio di fallimento sviluppando il software in finestre di tempo limitate chiamate iterazioni che in genere durano qualche settimana. Ogni iterazione è progetto a sé stante e deve contenere tutto ciò che è necessario per rilasciare un piccolo incremento nelle funzionalità del software: pianificazione (*user stories*), design, implementazione, test. Anche se il risultato di ogni singola iterazione non ha sufficienti funzionalità da essere considerato completo, deve essere rilasciato e nel susseguirsi delle iterazioni, deve avvicinarsi sempre di più alle richieste del cliente. Alla fine di ogni iterazione il team dovrà rivalutare le priorità del progetto.

Pertanto, Agile Tracking si è sviluppato seguendo i principi dettati dalla metodologia agile.

## Metodologie Agili



**Figura 4:** *Manifesto agile* [5]

Nel Febbraio del 2001, un gruppo formato da diciassette progettisti software si riunirono a formare la Agile Alliance. Ciò che emerse da questo incontro fu *Il Manifesto Agile*, sottoscritto da tutti i partecipanti. Il loro intento era ed è tuttora quello di scoprire metodi migliori di sviluppare software facendolo e aiutando gli altri a farlo.

Dietro al *Manifesto Agile* troviamo numerosi principi che hanno ispirato i suoi creatori per la stesura dei valori che lo rappresentano.

L'interazione tra il cliente e il team, e all'interno del team stesso è indispensabile e la conversazione faccia-a-faccia è senza dubbio il più efficiente ed efficace metodo per comunicare in un team di sviluppo o a un

team di sviluppo. Questo serve ad avere una buona analisi dei requisiti e una proficua collaborazione tra i programmatori anche in un ambito di quasi totale assenza di documentazione.

E' bene inoltre che periodicamente il team rifletta su come diventare più efficace, quindi regoli e modifichi il suo comportamento di conseguenza. Una continua attenzione verso l'eccellenza tecnica ed un buon design fa aumentare molto la produttività e l'efficienza del team.

L'aspetto che ha massima priorità è quello di soddisfare il cliente attraverso frequenti e continui rilasci di software funzionante, in un paio di settimane o al più un mese, preferendo una scala di tempo più piccola. Grazie a questi frequenti rilasci è possibile stabilire una misura dei progressi del progetto proprio in base al software che viene rilasciato ad ogni iterazione piuttosto che da centinaia di righe di documentazione.

Agile Tracking è uno strumento che grazie alla sua semplicità riesce a condensare molti dei principi agili alla base di questo manifesto.

## **User Stories**

Nel primo principio del *Manifesto Agile* viene sottolineata la maggior importanza che le interazioni e le persone ricoprono rispetto ai processi e agli strumenti e viene messo in evidenza come le relazioni e la comunicazione tra gli attori di un progetto software siano la miglior risorsa dello stesso.

Extreme Programming, ad esempio, utilizza la scrittura delle storie d'uso per raggiungere tale fine Wake (2000).

Esse infatti sono un potente strumento che garantisce al cliente la possibilità di capire autonomamente le sue reali necessità ed esigenze

facilitando il programmatore nel soddisfare tutte le sue richieste.

Ogni storia viene scritta su un piccolo foglio ed è caratterizzata dal titolo, dalla descrizione, dalla difficoltà espressa dal programmatore, dal valore di business per il cliente e dai test di accettazione che vengono scritti sul retro.

E' importante scrivere delle storie comprensibili a entrambi in modo tale da mettere il cliente in condizione di sapere fin da subito quali saranno le funzionalità che verranno implementate nel corso delle iterazioni. Infatti spesso i clienti, essendo gli esperti di dominio, conoscono i termini del loro campo di competenza, ma non quelli propri dello sviluppo software. Gli sviluppatori, d'altro canto, possono descrivere bene un sistema in termini funzionali ma privo di significato per un esperto di dominio. Il mancato uso di un linguaggio comune (*ubiquitous language* Evans (2004)) rischia di confondere la visione di insieme del progetto, e della sua implementazione. Bisogna riuscire a mantenere questa filosofia anche nel momento in cui ci si trova di fronte ad un cliente che ha competenze nell'ambito dello sviluppo software. In questa particolare situazione infatti può accadere che il cliente provi a deviare la scrittura delle storie da informale a tecnica.

Oltre alla chiarezza di significato è necessario che ogni storia non dipenda da nessun'altra. Allorquando nascono dei legami inscindibili tra le storie si è costretti a rinunciare a parte della flessibilità per seguire un cammino di sviluppo dettato dalle dipendenze stesse. Una buona scrittura delle storie in alcuni casi riesce ad evitare questo tipo di problema. Affinché una storia sia efficace è necessario che esprima la più piccola ed indipendente funzionalità possibile, quindi riconoscere e suddividere le richieste del cliente in unità atomiche.

La difficoltà di una storia viene espressa dallo sviluppatore attraverso una stima, in termini di tempo o punti-difficoltà, mentre il cliente la valuterà in base al valore di mercato, attendendosi pertanto alla reale importanza che l'implementazione della storia stessa riveste nel suo progetto. Questo significa che se per il cliente una storia ha un alto valore di mercato, è per lui fondamentale averla subito perché grazie a quella riuscirebbe ad avere fin dal primo rilascio un rientro in termini economici.

Le storie permettono pertanto di stabilire con maggiore precisione e semplicità il corso delle future iterazioni, guidando il programmatore nello sviluppo e al contempo fornendo al cliente una visione accurata e flessibile sull'andamento del progetto.

## **La Tecnica del pomodoro**

*La Tecnica del Pomodoro nasce con l'obiettivo di utilizzare il tempo come un utile alleato per realizzare le proprie attività nel modo desiderato e consentire un continuo miglioramento nel proprio processo di lavoro o studio. Cirillo (2006) .*

È dimostrato che periodi di tempo tra i 20 e i 45 minuti sono in grado di massimizzare attenzione e attività mentale se seguiti da una breve pausa.

La Tecnica del Pomodoro deve favorire consapevolezza, concentrazione e lucidità prevedendo infatti la suddivisione delle ore lavorative in periodi di tempo di venticinque minuti (1 pomodoro) più cinque minuti di pausa.

La giornata viene così espressa in pomodori lavorativi, alternando ad ogni pomodoro una breve pausa. Questa tecnica permette di aumentare l'efficacia personale, consente di ottenere maggiore lucidità, consapevolezza, focalizzazione, facilita l'apprendimento e consente di

concentrare gli sforzi sulle attività da realizzare.

Tale tecnica prevede inoltre svariati accorgimenti per la pianificazione della giornata di lavoro, nonché per la gestione degli imprevisti.

Affiancare questa tecnica alle user stories rende il processo di sviluppo molto più logico e dettagliato allo stesso tempo. Tuttavia, le stime in pomodori, a differenza di quanto si possa pensare, non sono semplici da fare. Serve sicuramente avere abbastanza esperienza nell'utilizzo di questa tecnica per effettuare delle stime con buona precisione. Ciò nonostante, può capitare di ritrovarsi con un numero di pomodori lavorati maggiore di quello stimato, e ciò può dipendere da moltissimi aspetti, non per forza da una errata stima iniziale.

Infatti per quanto si riesca a stimare con accurata precisione, possono accadere degli imprevisti durante la fase di implementazione, imprevisti di cui non si era tenuto conto nella stima iniziale e che vanno ad influire sul tempo totale di lavoro. D'altro canto può anche succedere di sovrastimare una storia e di accorgersi in seguito di riuscire ad implementarla in meno pomodori di quelli stimati all'inizio.

Nello sviluppo di Agile Tracking lo strumento di misura e stima del tempo fornito dalla Tecnica del Pomodoro ha avuto una duplice valenza: è stata lo strumento di time boxing utilizzato per lo sviluppo vero e proprio e quello fornito per la stima delle storie.



# **CAPITOLO II**

## **Retrospective Agili**

## Le Retrospettive

*Supponi di far parte di un team di sviluppo software. Stai facendo un buon lavoro ma non un ottimo lavoro. Inizi a notare i primi attriti interpersonali tra gli altri membri del gruppo, e alcune persone che tu vorresti mantenere nel gruppo non si dimostrano all'altezza del loro curriculum vitae. Capisci che hai bisogno di alleviare le tensioni interpersonali prima che le cose vadano peggio.*

***E' necessario che tu introduca le retrospettive nel tuo team.***(Derby-Larsen, 2006)

Con il termine *retrospettiva* si intende una particolare riunione, che si svolge al completamento di un incremento nel progetto, nella quale si riunisce il team di sviluppo al fine di analizzare ed eventualmente adeguare i propri metodi e il lavoro svolto. Le retrospettive permettono all'intero team di comprendere al meglio la situazione globale del progetto, sia per quanto riguarda i progressi nello sviluppo, sia per individuare e quindi risolvere tutti i problemi che hanno creato rallentamenti, anche nel caso in cui questi siano determinati dalla mancata interazione tra le varie parti del team.

Grazie alle retrospettive è possibile migliorare la produttività del team introducendo test più frequentemente, individuando in tal modo gli errori in anticipo e quindi diminuendo drasticamente il lavoro di fine rilascio; si riesce pertanto ad avere un notevole miglioramento nelle potenzialità di un team e nella qualità del prodotto, avendo più tempo da dedicare alla rifattorizzazione e ad un'attività di prevenzione dei difetti del progetto stesso; aumenta la capacità di individuare quali peculiarità necessitano di essere svolte prima di altre permettendo così release in meno tempo.

## Un Modello da seguire

In Derby-Larsen (2006), viene presentato dall'autrice un modello per la preparazione e lo svolgimento di retrospettive agili che può essere schematizzato in cinque punti principali: preparare l'ambiente, raccolta dati, generare intuizioni, decidere cosa fare, chiudere la retrospettiva.

### ***Preparare l'ambiente***

*Serve ad aiutare i partecipanti ad accantonare tutte le loro preoccupazioni e a concentrarsi solamente sulla retrospettiva, mettendoli a loro agio per aver la massima libertà di esprimersi e di pensare.*

Tutto deve essere curato nei minimi dettagli, non bisogna lasciare niente al caso. E' necessario che anche l'ambiente di lavoro sia confortevole e privo di distrazioni, nonché fornito di tutta la strumentazione necessaria allo svolgimento della retrospettiva.

Uno degli obiettivi principali è aiutare il gruppo a pensare e ad imparare insieme, proprio per questo è importante che ognuno dei partecipanti esprima ciò che si aspetta di ottenere dalla retrospettiva. Inoltre è proprio in questa fase che entrano in gioco i valori e gli accordi presenti all'interno del team che, se adattati correttamente, potrebbero apportare ulteriori vantaggi.

### ***Raccolta dati***

*Serve a stimolare la memoria dei partecipanti su quanto è accaduto in un particolare incremento di lavoro, favorendo diversi punti di vista da diverse prospettive e analizzando chi ha fatto cosa e come.*

In questa fase i partecipanti esprimono i loro dubbi e i loro successi in

maniera autonoma seguendo l'ordine di pensiero ad essi più congeniale per permettere a tutti di capire quanto è accaduto in un'iterazione.

Una rappresentazione grafica dei dati rende più facile la comprensione dello schema e i collegamenti tra le varie parti: si possono usare grafici o post-it e attribuire una priorità ad ogni singolo task con colori diversi a seconda del significato e disposti lungo una linea del tempo che ci permette di dare una collocazione temporale alle varie considerazioni.

In questo modo, al primo impatto, si riesce ad avere un quadro generale della situazione del gruppo e del progetto.

### ***Generare intuizioni***

*Serve a far emerge il più grande numero di idee possibile, filtrandole attraverso una serie di criteri predefiniti.*

E' questo il momento in cui domandarsi i motivi che hanno determinato le situazioni che sono emerse dalla precedente fase e quindi pensare a cosa fare.

Si analizzano quindi i dati emersi e si cercano punti di forza e problemi, si cerca di guidare il team ad esaminare le condizioni, le interazioni e i modelli che hanno portato al successo. Spesso per motivare e rendere più sicuri di se i partecipanti, sarà necessario condurli velatamente verso una soluzione che dovrà in ultimo essere individuata da loro stessi.

Potrebbe sembrare facile arrivare subito ad una soluzione e le prime soluzioni potrebbero anche essere corrette, tuttavia non sempre lo sono. Queste intuizioni aiutano il team a comprendere come lavorare nel modo più efficace possibile.

### ***Decidere cosa fare***

*Serve a sviluppare un piano dettagliato per le sperimentazioni e per le proposte.*

Nel caso in cui vengano individuati dei problemi, sarà necessario prenderne in considerazione solo una piccola parte e decidere quali fra le soluzioni emerse siano le più idonee, scegliendone magari alcune più semplici nel caso in cui l'iterazione precedente sia stata molto faticosa. Per responsabilizzare l'intero team, si può decidere di assegnare ad ognuno dei membri un compito da risolvere per la prossima retrospettiva. Ogni singolo individuo del team deve accettare ed impegnarsi a rispettare il compito pianificato.

### ***Chiudere la retrospettiva***

*Serve a concludere la retrospettiva con la certezza di aver impiegato il tempo a disposizione nel modo migliore.*

Tutto il materiale prodotto durante la retrospettiva e tutti gli insegnamenti appartengono al team e ai membri del team. Perché le energie accumulate durante la retrospettiva non svaniscano con la fine di questa, bisogna terminare in modo deciso scegliendo come documentare la retrospettiva e impiegando alcuni minuti per riflettere sul lavoro fatto, in particolare su ciò che è andato bene e su ciò che invece bisogna fare diversamente nella prossima retrospettiva.

Seguire il metodo presentato sopra porta il team a capire i diversi punti di vista dei membri che lo compongono incrementando quindi la produttività di tutti. Inoltre aiuta anche a seguire un ordine naturale di pensiero e ad

avere una corretta visione delle tecniche e dei metodi del team stesso. Per questo è importante lasciare che la discussione prosegua senza seguire schemi predeterminati e far sì che ogni individuo, spontaneamente esprima le sue perplessità. In questo modo, oltre ad avere dei reali vantaggi nello sviluppo del software, si garantisce maggior compatezza all'interno del team di sviluppo, che è senza dubbio l'aspetto che maggiormente influisce sull'efficienza di tutto il gruppo.

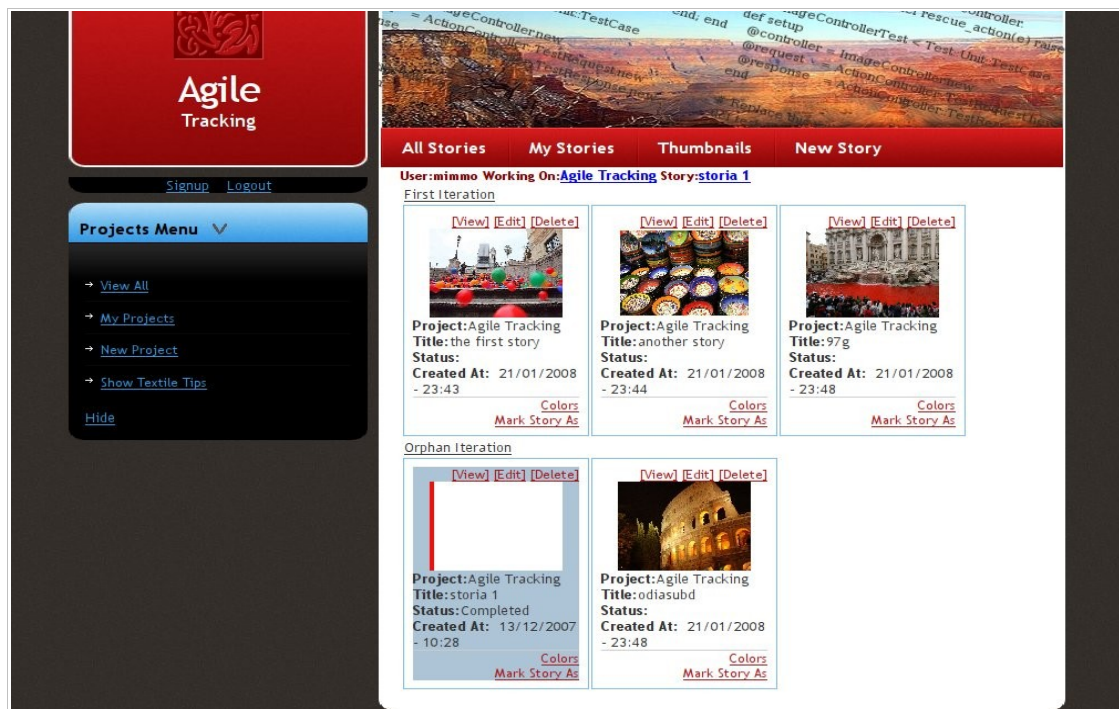
## **Agile Tracking e le retrospettive**

Agile Tracking è un'utile applicazione in grado di fornire al team una valutazione di insieme sull'andamento del progetto e pertanto può essere impiegato come linea guida per lo svolgimento delle retrospettive. A tal fine sono stati sviluppati e integrati nell'applicazione alcuni strumenti che agevolano il team in questo senso.

Nelle retrospettive è molto importante la fase di raccolta dei dati, ovvero quella fase in cui tutti i partecipanti sono invitati ad esprimere il loro punto di vista sui molteplici aspetti che caratterizzano un progetto, dalle interazioni tra gli sviluppatori, a quelle con il cliente e con il project manager, fino ad arrivare allo sviluppo delle storie e all'integrazione di queste. Per agevolare proprio quest'ultima parte è importante avere una immagine complessiva delle storie sviluppate e ancora più importante, per avere un dettaglio maggiore, sapere lo stato di ogni singola storia, altrimenti a fine iterazione, e durante la retrospettiva, si trascorrerebbe troppo tempo ad analizzare tutte le storie per trovare quelle incomplete o quelle bloccate. Per questo motivo in Agile Tracking, oltre alla più comune visualizzazione ad elenco è presente per le storie la visualizzazione rapida

delle anteprime (Figura 5).

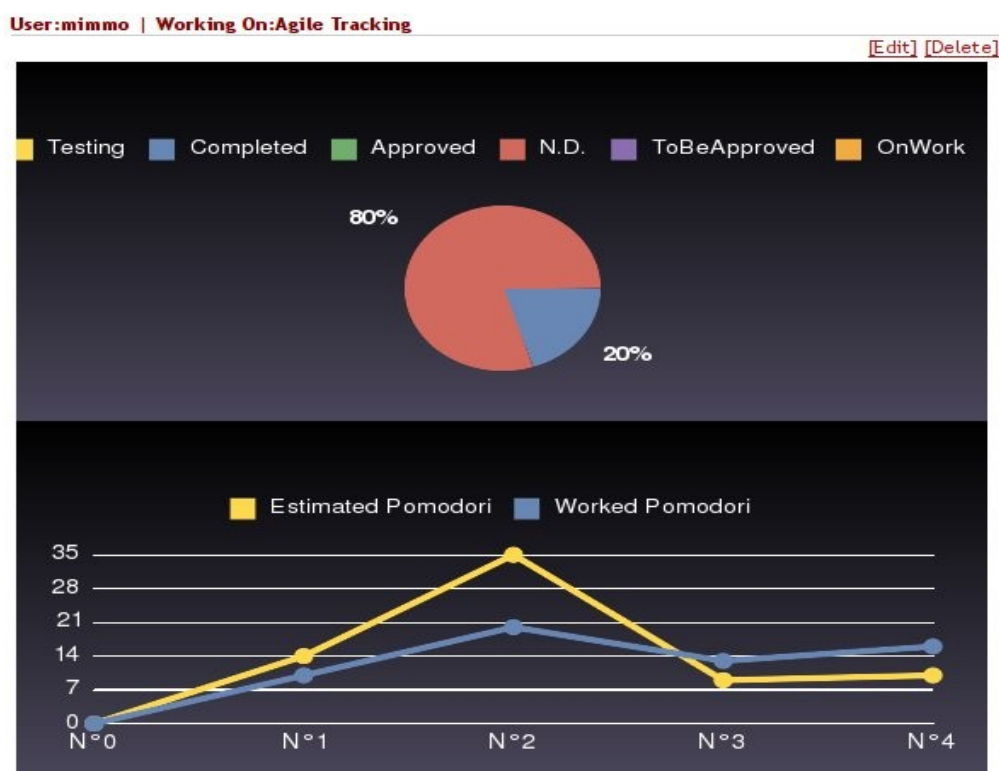
In questa particolare modalità si riesce immediatamente a determinare lo stato di tutte quante le storie, suddivise per iterazione, grazie alla possibilità di utilizzare un set di colori predefinito per contrassegnarle. La semantica dei colori è totalmente arbitraria e viene decisa dal team,



**Figura 5:** Agile Tracking - Thumbnails

conferendo piena autonomia alla gestione del progetto. Se infatti viene stabilito che il colore rosso sta a significare che la storia è bloccata, perché dipendente da un'altra o per altri motivi, quando ci si trova a fare la retrospettiva si potrebbe decidere di analizzare dapprima i problemi che hanno portato a questa situazione di stallo e in seguito, una volta risolte le problematiche più urgenti, proseguire con quelle di minor peso.

Per definire lo stato di una storia è inoltre possibile selezionare tra cinque diverse possibilità, *on work*, *in testing*, *completed*, *to be approved*, *approved*.



**Figura 6:** *Agile Tracking Project Graph*

Questo ulteriore parametro serve a fornire al team di sviluppo un grafico (*Figura 6*) nella pagina del progetto nel quale vengono espresse in percentuale le storie corrispondenti ad ogni stato, permettendo di avere sempre visibili i progressi nello sviluppo.

Inoltre sulla pagina del progetto è presente un altro grafico che indica per ogni storia la stima effettuata in pomodori e i corrispondenti pomodori lavorati (*Figura 6*).

Grazie alle stime si riesce ad individuare per quali storie si è impiegato più tempo di quello stimato e da qui iniziare nella retrospettiva a discutere i motivi che hanno causato tale rallentamento.

L'utilizzo di questi strumenti uniti al modello proposto in Derby-Larsen (2006) descritto sopra, porta notevoli miglioramenti nelle interazioni del



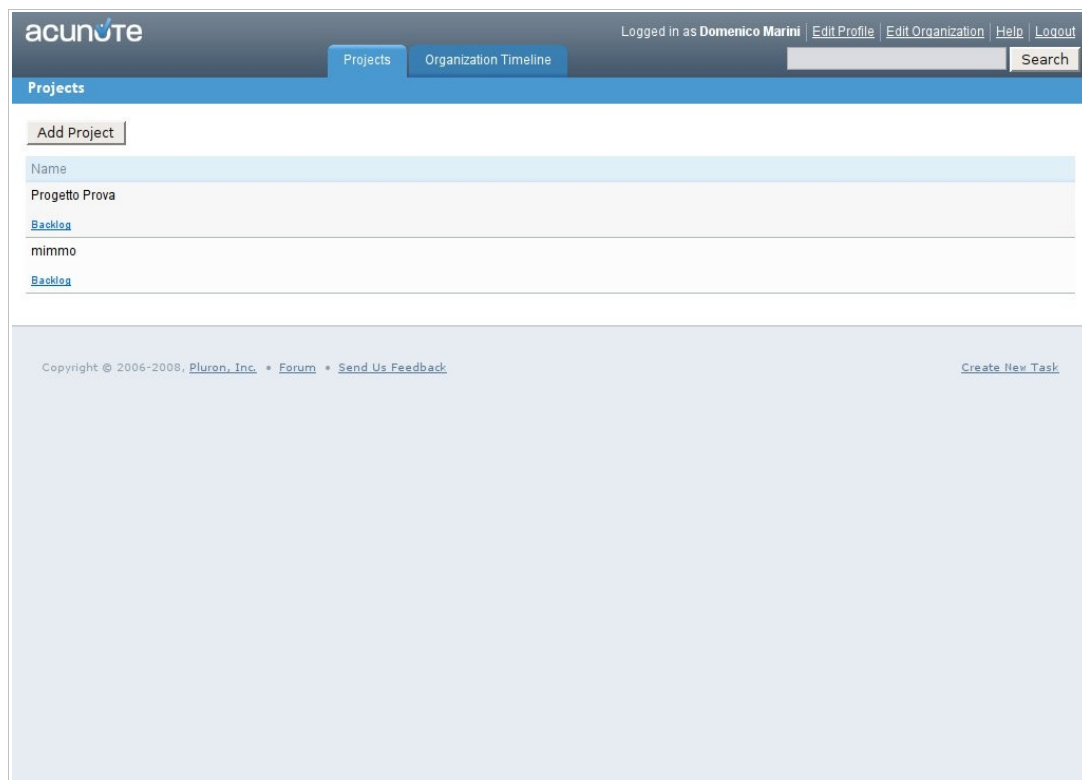
team, miglior efficienza nello sviluppo e soprattutto maggior efficacia nell'applicazione delle tecniche agili.

# **CAPITOLO III**

## **Analisi di mercato**

Dopo un'attenta analisi di mercato sono stati trovati diversi strumenti di tracking agile, di seguito ne verranno analizzati alcuni per presentare le loro caratteristiche principali, pregi e difetti.

## Acunote



**Figura 7:** Acunote [1]

Acunote, sviluppato dalla Pluron è uno strumento di amministrazione dei progetti software agili che adotta Scrum come tecnica di base per lo sviluppo.

Scrum si basa su tre semplici punti: *Sprint*, *Backlog* e *Scrum Meeting*. Complementare ad Extreme Programming prevede di dividere il progetto in blocchi rapidi di lavoro (Sprint) alla fine dei quali consegnare una versione al cliente, indica come definire i dettagli del lavoro da fare nell'immediato futuro (Backlog) per averne una definizione estesa,

organizza riunioni giornaliere del team di sviluppo (Scrum Meeting) per verificare cosa si è fatto e cosa si farà.

Seguendo questa idea, Acunote offre una serie di utilità per la creazione di un progetto e la sua conseguente amministrazione.

- **Timeline:**

Una visione cronologica ed unificata di tutte le attività del progetto

- **Burndown:**

Si possono tracciare i progressi grazie a grafici autogenerati in base ai dati inseriti negli sprint e nei backlog. Per fare questo utilizza una gemma di Ruby on Rails che si chiama Gruff.

- **Integrazione:**

E' possibile l'integrazione con altri strumenti come Subversion, Perforce, Bugzilla, Mantis, Trac, JIRA

- **Sprints**

Iterazioni da 1 a 4 settimane

- **Tempo**

Tracking del tempo che mostra quanto tempo è stato schedulato per un'attività e quanto lavoro manca per completare tale attività.

Dopo avere effettuato la registrazione e quindi l'accettazione del contratto proposto, si accede all'applicazione che risulta ad un primo impatto molto intuitiva e semplice. L'utente può quindi creare un nuovo progetto e aggiungere a questo i membri del team, il cui numero è diverso a seconda del tipo di licenza scelta nella fase iniziale. E' possibile specificare per un progetto, un repository *svn*, uno strumento di gestione dei bug ed

un'eventuale notifica tramite mail per ogni cambiamento apportato al progetto.

A questo punto, seguendo la politica Scrum si può passare alla creazione degli Sprint, specificando per ognuno di questi un nome, una data di inizio e una di fine. Per ognuno di questi sprint si possono inserire i task relativi, inserendo semplicemente una descrizione. In aggiunta si possono specificare il proprietario, lo stato del task e una stima del tempo espressa in ore che servirà per completare il grafico che descrive il tempo totale dei task inseriti.

Analogamente avviene la gestione del Backlog e la sua relativa creazione dei task. Grazie all'utilizzo delle etichette è possibile marcare i task con delle parole chiave, utili nelle ricerche. E' permessa la modifica di tutti questi dati anche in un secondo momento anche se non risulta molto intuitiva.

Inoltre è disponibile una sezione Timeline nella quale vengono registrate tutte le azioni eseguite secondo l'ordine cronologico, in questo modo si ha la possibilità di sapere sempre e in ogni momento chi sta lavorando al progetto, su quale sprint ed in particolare su quale task.

Sono possibili tre diverse versioni dell'applicazione, due prevedono il pagamento mensile di una quota di utilizzo, che risultano abbastanza accessibili anche per team emergenti e di piccole dimensioni, mentre la terza è una versione gratuita nella quale alcune funzionalità sono limitate come per esempio il numero di utenti, fino ad un massimo di cinque.

In definitiva Acunote risulta essere un valido strumento di gestione e

controllo dei progetti software ma risulta essere adatto solamente a team di sviluppo che agiscono rispettando le politiche e i metodi offerti da Scrum, non adeguato quindi per un team XP.

## Mingle



**Figura 8:** *Mingle* [7]

Sviluppato dalla Thoughtworks Studios, Mingle è un' applicazione per la collaborazione e la gestione di progetti software agili.

Un grande vantaggio che presenta Mingle rispetto ad altri software analizzati è senza dubbio la possibilità di potersi adattare tanto ad un team che sviluppa seguendo metodi e tecniche di Extreme Programming, quanto ad un team che invece utilizza Scrum.

Mingle offre una vasta gamma di funzionalità per la gestione e l'amministrazione dei progetti software:

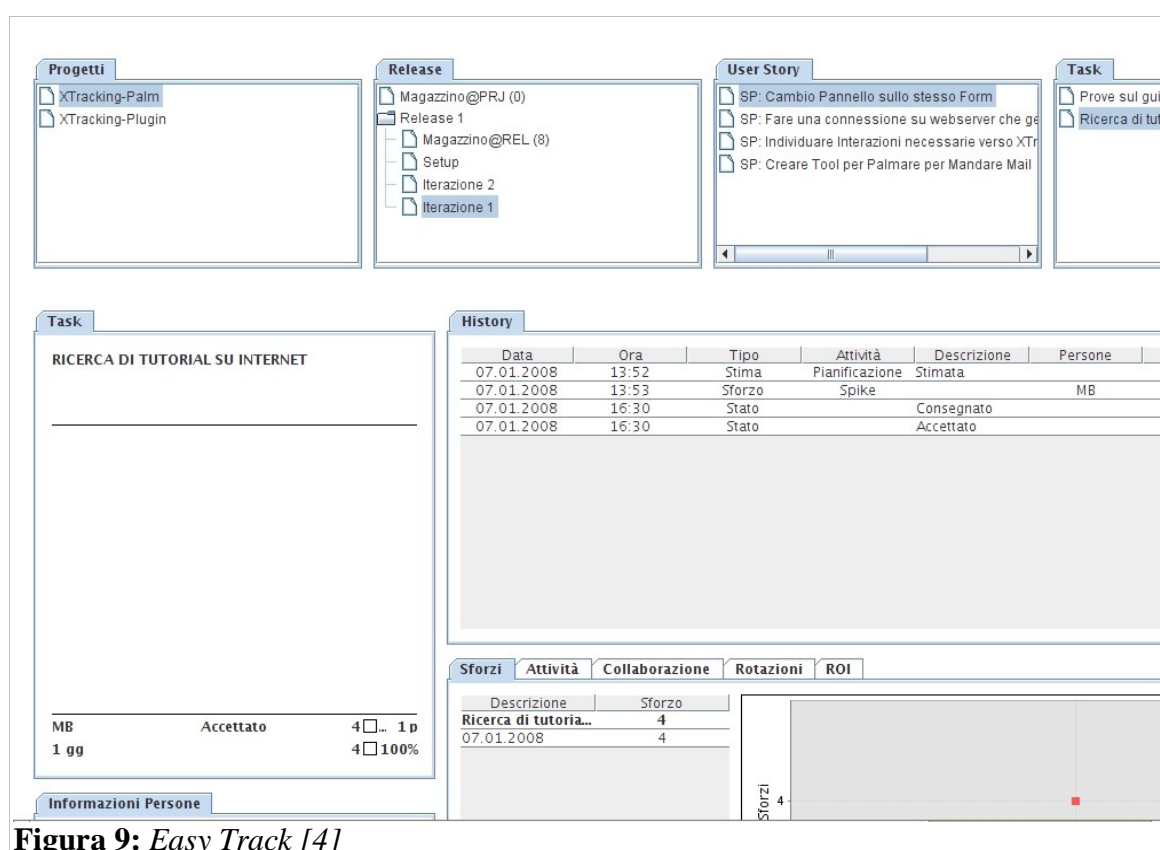
- **Flessibilità e Collaborazione ai Progetti**
  - Visione generale di storie, task, difetti, rischi e problemi
  - Un'insieme di filtri che permette di personalizzare le viste e gli ambienti di lavoro per ogni singolo membro del team.
  - Prprietà del progetto completamente personalizzabili: si può scegliere tra XP, Scrum e Agile Hybrid
  - Wiki integrato
  - Coda di lavoro
  - Campo di ricerca testo libero
- **Gestione dei Progetti**
  - Importazione ed esportazione su Excel
  - Utilizzo di tecniche XP, Scrum o Agile Hybrid
  - Tracciamento dell'andamento del progetto
  - Possibilità di creare un template e usarlo anche su altri progetti
  - Storico del progetto
  - Avvisi tramite mail o feed RSS
- **Tecnologie**
  - Integrazione con sistemi di gestione del codice sorgente(Source Code Management System)
  - interattività e divertimento grazie all'utilizzo di Ajax

L'intera applicazione è stata sviluppata in Ruby on Rails e può essere scaricata dal sito web ma, poiché sono previste alcune versioni a pagamento, non viene fornito il codice sorgente. Per questo motivo i controllers e le models vengono criptati e descriptati ogni volta che ne è richiesto l'utilizzo. Proprio questo aspetto rende Mingle poco performante

con dei tempi di attesa il più delle volte inaccettabili e inconcepibili se si pensa che il server viene eseguito in locale, a meno di impiegare dell'hardware molto costoso.

Esiste una versione di prova di trenta giorni che permette di avere fino ad un massimo di cinque persone nel team. Nel caso i membri del gruppo di lavoro siano di più, si può scegliere tra ben sei versioni differenti, da cinque a oltre cinquanta membri, aspetto che va a influire notevolmente sul prezzo finale del prodotto che infatti può non essere conveniente per team emergenti e formati da poche persone.

## Easy Track



**Figura 9:** Easy Track [4]

Nasce per ridurre lo sforzo richiesto dalla quotidiana compilazione di moduli cartacei ed elettronici destinati al tracking, per il controllo e la



visualizzazione dello stato di avanzamento dei progetti. Sviluppata da XPLabs, la prima azienda italiana e una tra le prime al mondo dedicata all'applicazione e diffusione dei Metodi Agili e fondata da Francesco Cirillo, è ancora in fase di sviluppo.

Easy Track permette un'ottima gestione di tutti gli aspetti concernenti un progetto XP:

- **Gestione dei progetti**
  - CRUD (Create, Read, Update, Delete)
  - Assegnazione Responsabile
  - Aggiunta di Persone al progetto
  - Aggiunta di una Release
  - Aggiunta di una Storia
  - Grafici relativi alle Linee di codice, Nr. Classi, Nr. Metodi e McCabe
  - Grafici statistici su distanza, velocità fino ad arrivare al ROI(Return of Investement)
- **Gestione delle rilasci**
- **Gestione delle User Stories**
- **Gestione dei Task**

Sicuramente il punto di forza di questa applicazione è la possibilità monitorare l'intero progetto molto dettagliatamente e utilizzando tecniche di analisi specifiche come la metrica di McCabe [17], metrica strutturale relativa al flusso di controllo di un programma che rappresenta la sua complessità logica cioè lo sforzo per realizzarlo e comprenderlo. Anche per quanto riguarda le stime in termini di tempo vengono adottati grafici molto ben dettagliati per definire velocità, distanza, sforzi, integrazioni,

*ROI(Return Of Investment).*

Tutti questi strumenti forniti sono sicuramente utili ma potrebbero risultare eccessivi e troppo tecnici per utenti il cui unico obbiettivo è avere un tracking semplice del proprio progetto e soprattutto per utenti che non conoscono fino in fondo tutte le metriche e le tecniche utilizzate per il calcolo dei grafici all'interno di Easy Track.

Easy Track è stato implementato in java ed è accessibile per ora in una versione demo tramite web. Purtroppo la sua complessità lo vede adatto solamente ad una fascia di utenti limitata.

# **CAPITOLO IV**

## **Agile Tracking**

## Introduzione

Agile Tracking è nato con lo scopo di facilitare il monitoraggio e l'organizzazione di progetti software. Su questa base si è deciso di svilupparlo come un'applicazione web, permettendo a chiunque sia dotato di un browser grafico e di una connessione ad internet di utilizzare il servizio senza l'obbligo di installare ulteriori librerie o programmi aggiuntivi. E' stato implementato con Ruby on Rails.

## Funzionamento

Agile Tracking prevede due modalità d'uso, una che non richiede la registrazione al servizio, l'altra nella quale è prevista la registrazione e l'autenticazione.

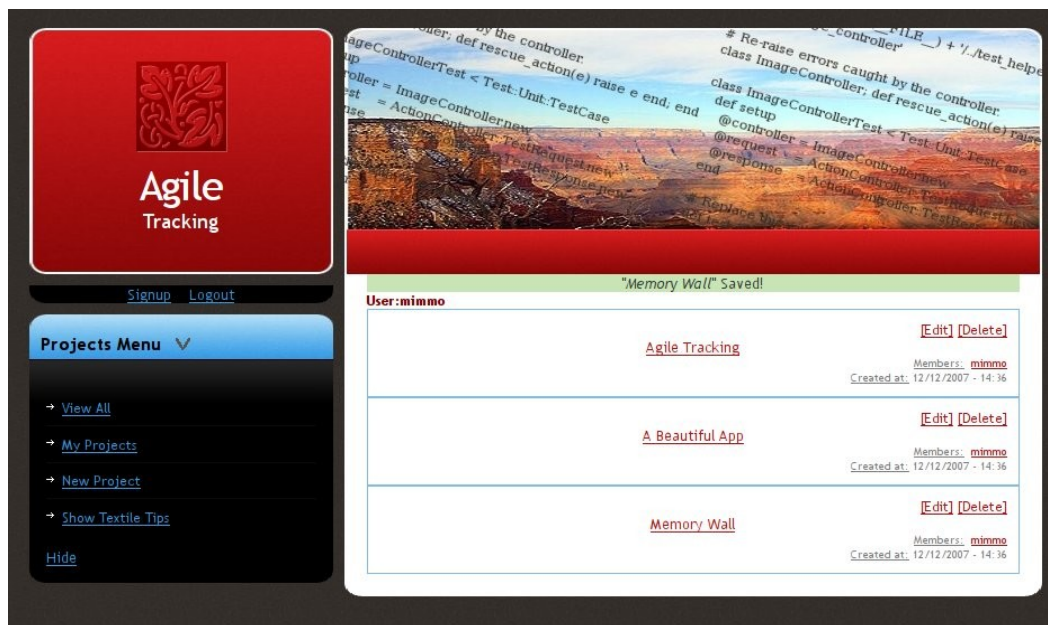
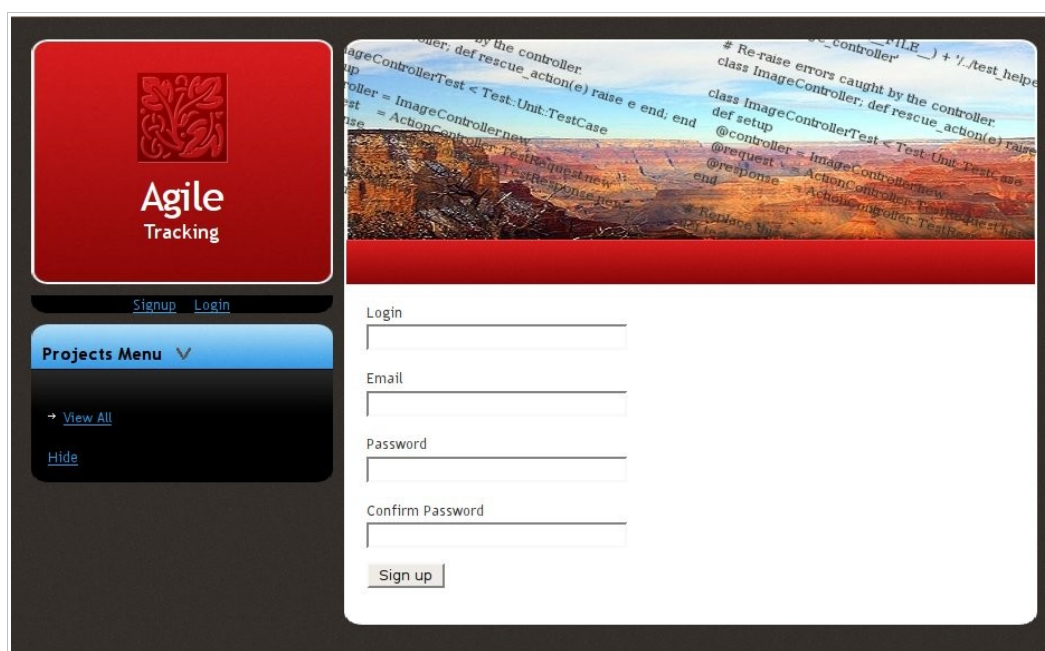


Figura 10: Agile Tracking - List All Projects

Nella prima modalità, l'utente può utilizzare il servizio usufruendo di una parte delle funzionalità offerte dall'applicazione. E' infatti possibile per lui

vedere l'elenco dei progetti inseriti (*Figura 10*) e, per ognuno di questi, visualizzarne il titolo, la descrizione, i membri che ne fanno parte. Inoltre si può decidere di contattare i componenti del team tramite e-mail.

Le funzionalità avanzate di Agile Tracking sono accessibili in seguito ad una autenticazione. Nella fase di registrazione (*Figura 11*) occorre specificare un nome per il login, la propria mail, ed una password con la sua relativa conferma.

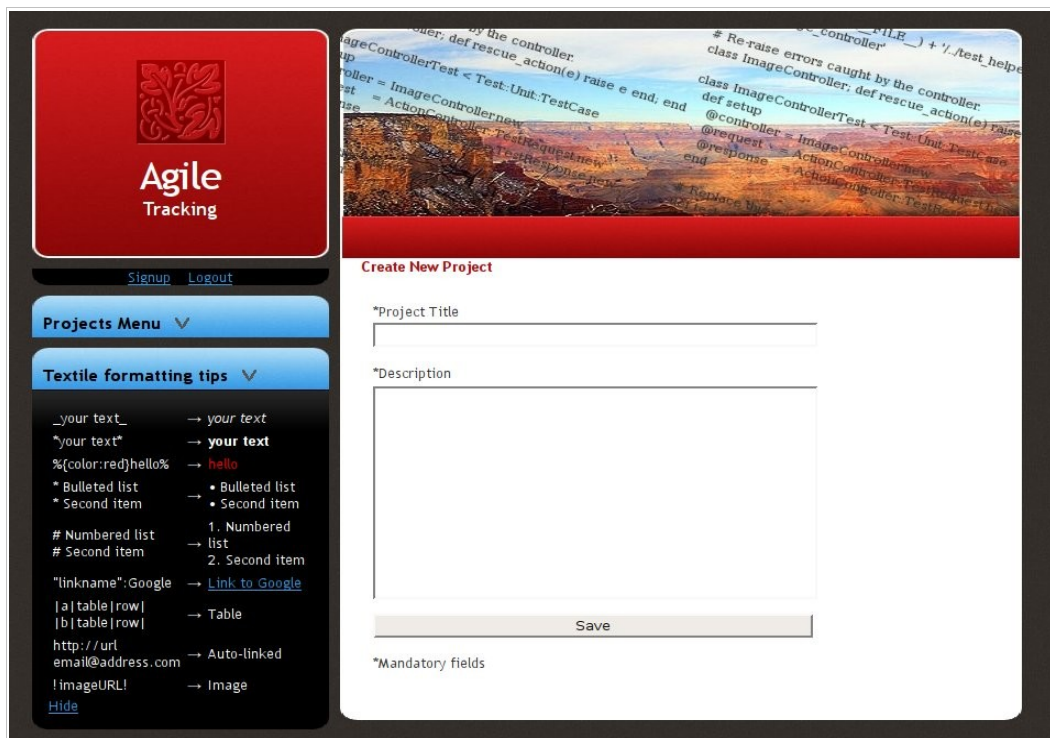


**Figura 11:** Agile Tracking - Signup

Al termine della registrazione, avviene l'autenticazione e si viene rediretti, attraverso una chiamata asincrona, alla lista dei progetti attualmente in lavorazione. A questo punto è possibile creare un nuovo progetto, selezionando l'apposito link presente sulla pagina o nel menù di navigazione laterale.

La creazione di un progetto prevede l'inserimento di un titolo e di una

descrizione: quest'ultima può essere inserita usando al suo interno i caratteri di formattazione di *Textile*, per avere effetti come il grassetto, il corsivo, elenchi puntati e numerati, link, immagini e molto altro (*Figura 12*). Per agevolare l'utente nella scrittura della descrizione è disponibile nel menù di navigazione laterale, un link che fa comparire, sempre utilizzando una chiamata asincrona, un pannello di aiuto con alcuni esempi di formattazione e relativi output.

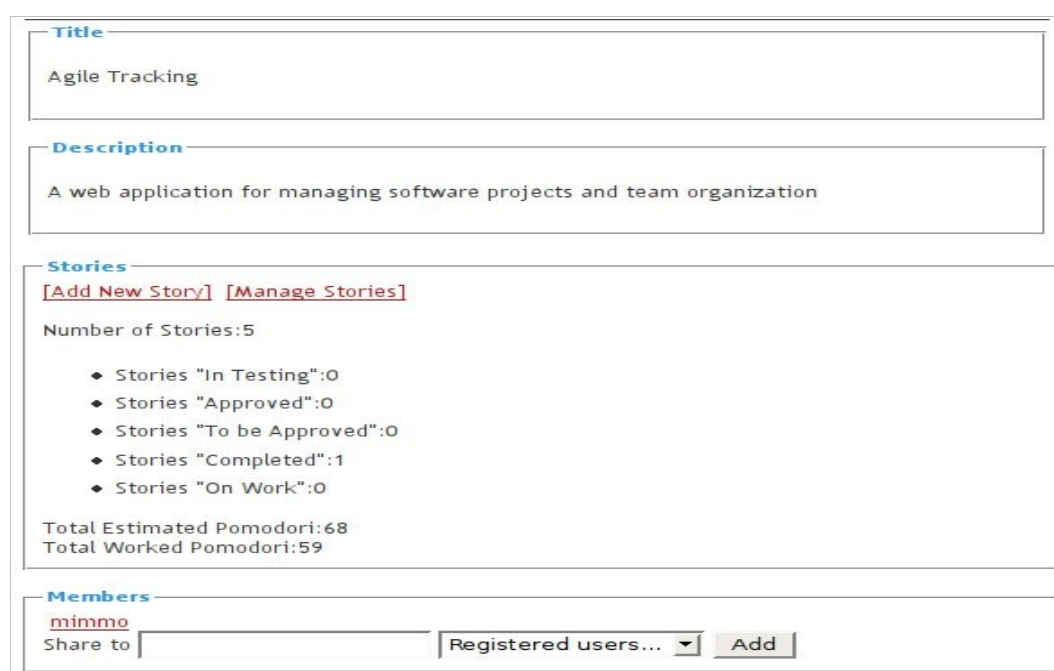


**Figura 12:** Agile Tracking - Add New Project

La creazione di un progetto termina con il salvataggio dei dati inseriti e la visualizzazione della lista dei progetti. Per avere un accesso rapido ai propri progetti è sufficiente selezionare il link posto sul menù di navigazione laterale. Verranno infatti mostrati tutti i progetti a cui un utente appartiene, siano essi creati da lui o da altri.

Se dalla lista dei progetti ne viene selezionato uno, si accede ad una pagina di report del progetto dove, nel caso l'utente disponga dei privilegi necessari, sono visualizzate informazioni aggiuntive sul progetto nonché link per la gestione del progetto e delle storie.

La prima cosa che si nota all'interno di questa pagina sono i due grafici che compaiono in cima (*Figura 6*). Questi rappresentano informazioni statistiche sullo stato del progetto in generale. Nel grafico a torta sono rappresentati in percentuale i vari stati delle storie del progetto, aiutando l'utente a comprendere meglio come il si stia evolvendo. L'altro grafico rappresenta, attraverso due linee, le stime dei pomodori per ogni storia, mostrando all'istante eventuali errori di stima ed aiutando così il team a



**Figura 13:** Agile Tracking - Project Report  
migliorarsi.

In fondo alla pagina di report del progetto è possibile amministrare gli utenti appartenenti al progetto, decidendo chi aggiungere o togliere dal

team.

La gestione del progetto permette, oltre alla modifica del titolo e della descrizione, anche l'intera gestione delle storie relative al progetto.

In questa pagina, nella sezione *Stories*, sono mostrate alcune informazioni relative alle storie del progetto, in particolare il numero totale, il numero di storie per ogni stato e anche il numero totale di pomodori stimati e lavorati. Da qui inoltre sono disponibili due collegamenti, uno alla creazione di una nuova storia e un altro all'elenco completo delle storie del progetto. Selezionando uno di questi viene ricaricata l'intera pagina con il nuovo layout che offre in più una barra di navigazione contenente il menù delle azioni per le storie.

*Illustrazione 1: Figura 14: Agile Tracking - New Story*

La creazione di una storia comporta l'inserimento di un titolo e di una descrizione negli appositi campi che sono obbligatori al fine di un salvataggio corretto. In aggiunta si possono specificare l'iterazione, lo



stato, una difficoltà, un valore di business e i valori che riguardano i pomodori stimati e lavorati.

Comunque è possibile modificare tutte le informazioni relative ad una storia in un secondo momento. Al termine della creazione avviene il salvataggio della storia del progetto e si viene rediretti alla lista delle storie.

Così come per i progetti, anche per le storie è disponibile una pagina di report nella quale vengono mostrate tutte le caratteristiche di una storia. Da qui è anche possibile aggiungere delle immagini alla storia, possibilità pensata avendo come riferimento i casi reali nei quali ci si trova a scrivere con il proprio cliente le storie nel classico formato cartaceo che potranno in seguito essere scansionate e assegnate alla storia stessa. In particolare per ogni storia è possibile assegnare due immagini: una rappresentante la storia vera e propria (*front image*), l'altra i relativi test di accettazione (*back image*).

**Agile Tracking**

Signup Logout

**Projects Menu** ▾

- View All
- My Projects
- New Project
- Show Textile Tips
- Hide

**All Stories My Stories Thumbnails New Story**

User:mimmo Working On:Agile Tracking Story:the first story [Edit] [Delete]

**Front Image**

Default Front Image

**Back Image**

Default Back Image

**Title**

the first story

**Description**

as person i want to do something for...

**Additional Info**

Iteration Number --> 1  
 Status -->  
 Difficulty --> 0  
 Business Value --> 0  
 Estimated Pomodori --> 0  
 Worked Pomodori --> 0  
 Creator --> mimmo

**Figura 15:** Agile Tracking - Story Report

Oltre alla classica visualizzazione ad elenco delle storie c'è la possibilità di aprire le storie come insieme di anteprime, una sorta di bacheca riepilogativa di tutte le storie. Ogni anteprima fornisce una descrizione minimale della storia, mostrando però i dati di maggior rilievo come ad esempio il titolo, lo stato e la *thumbnail* della front image di ogni storia e fornisce la possibilità di avere un rapido colpo d'occhio sullo stato di ogni storia soprattutto grazie alla possibilità di assegnare un colore ad ogni storia tramite un menù a comparsa posto in fondo ad ogni storia.

## Ajax

Ajax è una tecnica di sviluppo web per creare applicazioni web interattive. La tecnica Ajax utilizza una combinazione di HTML e CSS per il markup e lo stile, DOM (Document Object Model) manipolato attraverso un linguaggio come JavaScript per mostrare le informazioni ed interagirvi, l'oggetto XMLHttpRequest per l'interscambio asincrono dei dati tra il browser dell'utente e il web server e XML come formato di scambio dei dati, anche se di fatto qualunque formato può essere utilizzato, incluso testo semplice Gross (2006).

L'utilizzo di Ajax in un'applicazione web rende l'applicazione più reattiva agli occhi dell'utente.

Supportato nativamente da Ruby on Rails, in Agile Tracking Ajax è stato utilizzato principalmente nei collegamenti, in particolare nella maggior parte dei casi si è preferito al classico *link\_to* il corrispondente *link\_to\_remote*, evitando in tal modo il caricamento completo di una pagina e facendo aggiornare semplicemente una porzione della stessa. In alcuni casi tuttavia è stato necessario mantenere all'interno del codice

collegamenti del primo tipo poiché era necessario aggiornamento dell'intera pagina. Infatti nella pagina di report del progetto, nella sezione *stories* i link *Add New Story* e *Manage Stories* sono stati implementati nel formato *link\_to*. Questo perchè era necessario il caricamento del nuovo layout, che oltre al menù di navigazione laterale fa comparire in alto una barra con le azioni possibili per le storie.

## Gemme e Plugin

### *RedCloth*

Per la formattazione del testo si è deciso di utilizzare la gemma RedCloth. Si tratta di un implementazione per Ruby di Textile, un linguaggio di markup più semplice dell'HTML che ha come scopo quello di facilitare gli utenti nella scrittura di testi formattati. L'idea è quella di fornire un linguaggio semplificato che può essere tradotto facilmente in HTML standard. Esiste un'alternativa a Redcloth: la gemma BlueCloth, un implementazione per Ruby di Markdown, un linguaggio simile a Textile. Le due gemme attualmente sono equivalenti per funzionalità offerte e modalità di utilizzo, tuttavia, durante lo sviluppo di Agile Tracking sono stati riscontrati due punti importanti in favore di RedCloth: Textile è largamente impiegato in molti progetti diffusi in rete, come wiki e blog, pertanto il suo utilizzo è già noto a molti utenti, in più, a differenza della più giovane BlueCloth, il suo codice è molto stabile. Una dimostrazione è data dal fatto che i nuovi rilasci servono maggiormente per aggiungere nuove funzionalità e non per correggere errori.

### ***Acts as Attachment***

Per gestire l'upload delle immagini in Agile Tracking è stato usato il plugin Acts as Attachment. Questo serve per memorizzare i file nel database o nel filesystem, salvando anche il nome del file, il tipo di file, le dimensioni e nel caso delle immagini anche le dimensioni dell'immagine stessa. Per ogni immagine è stato aggiunto un nuovo record nel database contenente oltre che le informazioni fornite dal plugin anche un campo che identifica se l'immagine è quella frontale o meno.

Il plugin prevede anche una buona gestione delle miniature che però implica l'installazione, tra le librerie di sistema, di Rmagick, una libreria per la gestione delle immagini e della loro modifica.

### ***Acts as Authenticated***

La parte di autenticazione degli utenti, inclusa la loro registrazione, è stata realizzata installando il plugin Acts as Authenticated. L'installazione di questo plugin crea all'interno dell'applicazione la tabella relativa agli utenti, i test unitari e funzionali, il modello, il controller e le viste. Il funzionamento del plugin è molto semplice e può essere suddiviso in tre fasi, il recupero dei dati necessari alla registrazione, il salvataggio di questi nella base di dati e l'autenticazione dell'utente appena registrato. Con questo plugin è possibile anche inviare una mail di notifica dell'avvenuta registrazione, grazie ad un mailer fornito con il plugin stesso.

### ***Gruff***

La libreria grafica Gruff è un plugin che permette la creazione di grafici con Ruby. In Agile Tracking la creazione di un grafico viene effettuata nel

controller delle storie tramite due metodi che generano una url temporanea dell'immagine che viene generata.

## Conclusioni

E' stato realizzato uno strumento per la gestione dei progetti software in grado di offrire semplicità e professionalità nello stesso tempo. Esso si pone come valida applicazione per tutti i team di sviluppo che utilizzano tecniche di Extreme Programming. Grazie ad Agile Tracking è possibile avvicinarsi al mondo XP e usufruire di tutti i vantaggi che questa tecnica offre pur senza essere esperto del settore.

I grafici generati con *Gruff* non sono particolarmente esplicativi ma abbastanza efficaci ed esprimono in modo chiaro il messaggio. A seguito di una ricerca per trovare qualche libreria integrabile con Rails per la gestione di grafici, si è comunque scelto di utilizzare *Gruff* essendo la miglior soluzione trovata.

Allo stato attuale non è stato rilevato un metodo per la gestione degli upload di immagini che preveda l'utilizzo di chiamate asincrone. Si rimanda l'implementazione di questo per gli sviluppi futuri.

Inoltre si è pensato di aggiungere la possibilità di creare gruppi di utenti all'interno di un progetto in modo tale da poter invitare a collaborare al progetto solamente le persone che fanno parte del gruppo. Una sorta di network privata e interna all'applicazione.

## **Bibliografia**

Black David (2006) Ruby for Rails - Ruby techniques for Rails developers , Greenwich, Manning

Buzan Tony (1974) Use your Head , London, BBC Books

Dave Thomas, Fowler Chad, Hunt Andy, (2005) The Pragmatic Programmers Guide - Programming Ruby , Dallas-Raleigh, The Pragmatic Programmers

Dave Thomas, Hansson David, (2005) Agile Web Development With Rails , Dallas-Raleigh, The Pragmatic Programmers

Eric Evans (2004) Domain-Driven Design: Tackling Complexity in the Heart of Software , Addison Wesley

Esther Derby, Diana Larsen (2006) Agile Retrospectives , The Pragmatic Bookshelf

Fowler Chad (2006) Rails Recipes , Dallas-Raleigh, The Pragmatic Programmers

Fowler Martin (2001), "The Agile Manifesto: Where it Came From and Where it May Go", <http://martinfowler.com/articles/agileStory.html>

(ultimo aggiornamento significativo luglio 2006)

Francesco Cirillo (2006), *La Tecnica Del Pomodoro*

Gross Christian (2006) *Ajax Patterns and Best Practices* , Berkeley, Apress

Hibbs Curt, Tate Bruce, (2006), *Ruby on Rails: Up and Running* , Gravenstein Highway North Sebastopol, O Reilly

Krasner Glen, Pope Stephen (1988) "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80", *Journal of Object Oriented Programming (JOOP)*, Vol. 1, Denville

William C. Wake, (2000) *Extreme Programming Explored*

Yona McGregor, Tony Buzan, (1998) *A Guide to Animal Farm*

Zeldman Jeffrey (2003) *Progettare il Web del futuro: Standard e tecniche per il design* , Milano, Pearson Education Italia



## Sitografia

(Reperita il 21 gennaio 2008)

- [1] Acunote - <http://www.acunote.com>
- [2] Agile Alliance - <http://www.agilealliance.org>
- [3] DDD (Domain Driven Design) - <http://domaindrivendesign.org>
- [4] Easy Track <http://www.xplabs.it>
- [5] Extreme Programming - <http://www.extremeprogramming.org/>
- [6] Manifesto Agile - <http://agilemanifesto.org/>
- [7] Mingle - <http://studios.thoughtworks.com/mingle-project-intelligence>
- [8] O'Reilly open source web plattaform - <http://www.onlamp.com/>
- [9] Pluron - <http://www.pluron.com/corporate>
- [10] Ruby - <http://www.ruby-lang.org/>
- [11] Ruby API - <http://www.ruby-doc.org/core/>
- [12] Ruby on Rails API - <http://api.rubyonrails.org/>

- [13] Ruby on Rails - <http://www.rubyonrails.org/>
- [14] Ruby on Rails API - <http://api.rubyonrails.org/>
- [15] Textile markup language - <http://www.textism.com/tools/textile/>
- [16] Thoughtworks <http://www.thoughtworks.com>
- [17] Wikipedia Italia - [http://it.wikipedia.org/wiki/Pagina\\_principale](http://it.wikipedia.org/wiki/Pagina_principale)
- [18] Wikipedia Inghilterra - [http://en.wikipedia.org/wiki/Main\\_Page](http://en.wikipedia.org/wiki/Main_Page)
- [19] Mokabyte, <http://www.mokabyte.it/2004/03/metod-1.htm>