Agile sempre più Agile

*L'attenzione ai costi ed all'efficienza induce molte aziende a considerare l'approccio Agile. Non è una moda, ma un'esigenza ed anche un dato di fatto che Agile ha la sua valenza se adottato correttamente senza sottovalutazioni. Il PMI propone una nuova certificazione **Agile PMI-ACP**, segno che i tempi sono maturi.*

Prospettive dello sviluppo software

La principale aspirazione dell'industria dello sviluppo software è l'aumento della produttività, della profittabilità e forse anche della soddisfazione del cliente.

In questo scenario, la figura del project manager è in prima linea, anche se, in molte realtà non è chiaro il suo ruolo e ancor meno le sue competenze. E' costume attribuire prima il titolo di project manager e poi si cerca di dargli un contenuto.

Questa facile flessibilità sulla figura del project manager adatta a tutte le stagioni, la rende vulnerabile e con l'avvento dell'Agile project management le cose si complicano addirittura.

Lo sviluppo software negli ultimi venti anni ha prodotto tanti bocconi amari. Molto spesso ne hanno fatto le spese i project manager, ma in realtà le cause sono da ricercare più nell'incapacità di un certo management spregiudicato o privo di competenze. I bravi project manager sono stati costretti a piegarsi e assecondare scelte assurde. L'alternativa sarebbe stata la perdita del lavoro, cosa che puntualmente è arrivata dopo l'ennesimo insuccesso.

Spesso si determinano situazioni insostenibili. Ad esempio, un direttore generale per grazia ricevuta, dovendo proporre qualcosa, chiede alla Direzione IT di sviluppare un data warehouse che gli consenta di avere lo stato della produzione, delle vendite e del morale del personale sulla sua scrivania, anzi sul suo tablet, in ogni istante, come ha visto pubblicizzato casualmente in un aeroporto.

Finge di stupirsi del fatto che l'IT non ci ha ancora pensato e inizia a fare pressione, iniziando a chiedere: QUANTO COSTA? QUANDO SARA' PRONTO?

Anche se il progetto potrebbe essere molto interessante, è difficile relazionarsi con simili personaggi e con l'approccio tradizionale l'insuccesso è garantito.

Con quale approccio rispondere?

Vediamo i pro ed i contro dell'approccio waterfal e di un approccio Agile.

Con l'approccio tradizionale dovremmo avviare un progetto su più fasi, dove la soluzione andrebbe disegnata prima sulla carta e poi rilasciata tutta insieme o per singola area: produzione, vendite e personale. Un progetto del genere può costare do poche migliaia ad alcune centinaia di Euro in base alla complessità dei dati da integrare ed alla specificità dell'applicazione che si desidera realizzare. Teoricamente si potrebbe utilizzare la stima per analogia con un progetto simile. Ma quale progetto considerare simile, se in azienda non si è mai parlato di data warehouse? Il rischio che possa occorrere il doppio del tempo o la metà del budget preventivato è molto alto.

Approccio a cascata (waterfal)

L'approccio tradizionale prevede come minimo le fasi di: **raccolta requisiti, analisi dei requisiti, disegno della soluzione, codifica, test** e **rilascio dell'applicazione**, con almeno le seguenti attività:

- raccogliere tutti i requisiti e le funzionalità desiderate dalle singole direzioni;
- analizzare i requisiti per comprenderne la logica e la consistenza della soluzione;
- disegnare una soluzione e sottoporla all'accettazione dell'utente sulla carta;

- realizzare la codifica dell'applicazione;
- effettuare i collaudi con l'aiuto nuovamente dell'utente per l'accettazione;
- implementare la soluzione e avviarle l'esercizio e la manutenzione.

Quando l'utente viene invitato ad accettare il disegno dell'applicazione ne sa quanto prima, anzi meno, perché nel frattempo ha maturato altre idee. Nell'esempio, il nostro arguto direttore generale si è convinto di una nuova soluzione letta dai cartelloni pubblicitari del suo aeroporto, accantonando la sua prima richiesta all'IT.

L'utente accetta il disegno della soluzione proposta dall'IT, iniziando a chiedete modifiche e integrazioni con la sua nuova idea, scompaginando i piani dell'IT. Praticamente, l'IT sta lavorando ad un'applicazione già morta, se non asseconda l'utente nella sua nuova fantasia. L'IT ha davanti a sé due scelte:

1) resistere e realizzare l'applicazione come richiesto all'inizio;
2) rivedere l'analisi ed il disegno per recepire le nuove idee del direttore.

Entrambe le scelte portano ad un disastro annunciato. Nel primo caso, l'applicazione non sarà di gradimento dell'utente; nel secondo, i tempi ed i costi lieviteranno al punto da non giustificare più il progetto. Il project manager rischia di farne le spese, se non prende le opportune cautele.

Approccio Agile

L'approccio Agile modifica completamente lo scenario, condividendo la responsabilità del progetto con l'utente o un suo rappresentante che per l'IT è la stessa cosa.

Purtroppo, esistono ancora alcuni manager che considerano l'IT il reparto dei miracoli. Si fanno venire un'idea e l'affidano all'IT che dovrebbe sapere cosa e come fare o avere in tasca già una soluzione. Il suo compito è sviluppare soluzioni! Questa tipologia di manager, fortunatamente in estinzione, evita accuratamente il dialogo con l'IT, perché temono il confronto con i tecnici e perché alla prima osservazione non saprebbero cosa rispondere. Con simili soggetti, bisogna fare molta attenzione a non illuderli, perché alle prime difficoltà te le ritrovi contro, senza che abbiano minimamente compreso cosa stavi facendo per loro. In questi casi, anche l'approccio Agile rischia di fallire.

In ogni caso, l'approccio Agile prevede la costituzione di un team di lavoro di 8-10 persone al massimo, con ruoli ben definiti in grado di cooperare per realizzare l'idea del direttore generale. Il vantaggio di un approccio Agile è che il team non ha un capo delle persone, ma tutti sono responsabili dei processi:

- *Product Owner* - il portavoce dell'utente,
- *Scrum Architect* - il responsabile delle architetture disponibili,
- *Scrum Master* - il sostenitore dei processi Scrum,
- *Team di Sviluppatori* - coloro che realizzano l'applicazione.

Apparentemente in contrapposizione, in realtà queste figure determinano l'equilibrio del team, realizzando esattamente quanto sono in grado di realizzare.

Il compito più difficile è far capire al direttore generale che la responsabilità è condivisa, invitandolo ad esplicitare la sua idea in modo graduale, oppure farsi rappresentare da una persona di sua fiducia. Questo è il salvacondotto dell'IT.

Anziché farsi raccontare tutti i dettagli dei requisiti dall'utente ed estrometterlo dalla realizzazione, l'approccio Agile prevede la collaborazione continua tra sviluppatori e utente, per tutta la durata del progetto. Insieme si decide cosa realizzare per prima, per poi migliorarlo successivamente aggiungendo tutte le funzionalità ed i dettagli necessari.

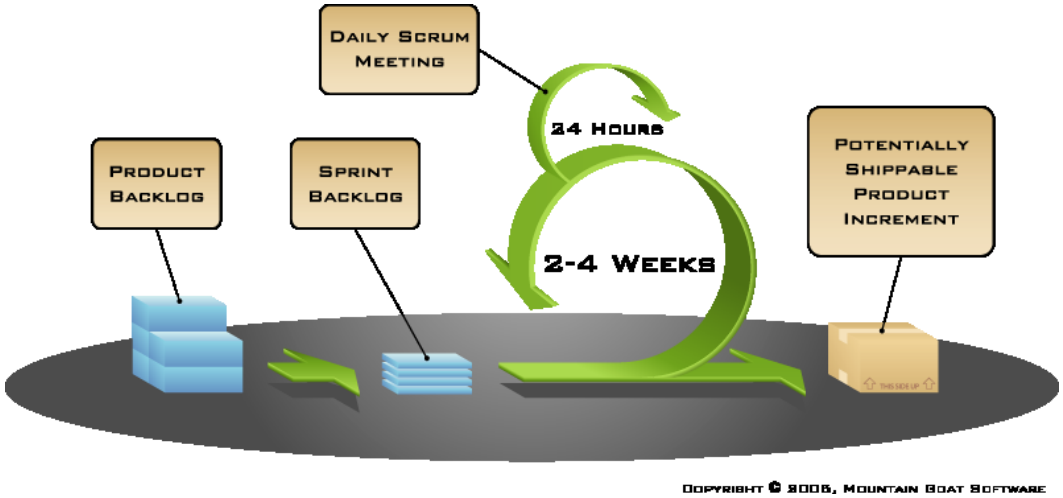
Gli utenti raccontano le loro esigenze al *Product Owner*, il quale negozia con gli altri responsabili la realizzazione in una serie di iterazioni dette *Sprint*.

L'utente alla conclusione di ogni *sprint* vede cosa è stato realizzato, fa le sue osservazioni e decide di avanzare le successive richieste di miglioramento.

Il team di sviluppatori prende in carico soltanto incrementi di soluzione (*stories*) che ha compreso e si impegna a realizzarli nel'arco del successivo *sprint* di 2 - 4 settimane.

Una volta avviata, la macchina Agile è perfetta.

- L'*utente* vede subito realizzata la sua idea e se ne fa venire un'altra.
- Il *team di sviluppatori* esegue solo lavoro che ha compreso.
- Il *Product Owner* media le richieste degli utenti con lo *Scrum Architect*.
- Lo *Scrum Master* verifica il corretto utilizzo di processi e tool Agile.



Troppo semplice, quando sono consolidati e si rispettano questi ruoli. Va notato che queste figure collaborano senza nessuna subalternità. Ognuno è responsabile di dare il proprio contributo in pieno auto governo.

Venendo da una mentalità gerarchico - funzionale questo concetto è difficile da far passare. Oltre ai Manager che preferiscono dare ordini, ci sono anche professionisti che non desiderano autogovernarsi. C'è chi preferisce eseguire anziché decidere. Queste persone vanno aiutate a comprendere che è tempo di cambiare approccio e assumersi la propria fettina di responsabilità. Questa è la parte più difficile dell'adozione di Agile.

Il team si incontra ogni giorno in una breve riunione detta "*Daily Scrum*" dove ogni partecipante risponde alle seguenti domande:

1. Cosa hai fatto ieri?
2. Cosa farai oggi?
3. Quali ostacoli vedi nel tuo lavoro?

Dopo 2-4 settimane il team deve rendere conto di ciò che si era impegnato a realizzare nello *sprint* e non può accampare scuse, avendo ogni giorno la possibilità di dare l'allarme senza aspettare l'ultimo giorno dello sprint.

In questo modo, le difficoltà emergono immediatamente, senza aspettare la fase di collaudo come avviene con l'approccio waterfal. Il rappresentante dell'utente è sempre al corrente di cosa si sta accadendo, quali difficoltà sono emerse e in che modo si è pensato di superarle, anche con il suo avallo.

Differenze tra Waterfal e Agile

Con l'approccio waterfal l'utente esprime i suoi desiderata con i requisiti, accetta l'analisi ed il disegno a scatola chiusa e si pone in attesa della fase di collaudo.

Il tempo è tiranno, per cui esiste il rischio concreto che nel frattempo l'esigenza sia cambiata e quindi anche se il team di progetto ha fatto un ottimo lavoro, l'utente può non essere molto soddisfatto della soluzione chiesta un anno prima.

Con l'approccio Agile sviluppatori e utente vivono insieme l'avventura di creare la soluzione che in ogni istante fa riferimento alle reali esigenze dell'utente. Non c'è stato nessun impegno su una soluzione determinata, pertanto insieme utente e sviluppatore maturano i dettagli da aggiungere per sofisticare la soluzione quanto basta.

Agile è una filosofia di vita, con coinvolgimento dinamico basato su processi condivisi, iterativi e incrementali che portano alla responsabilità collettiva.

Come si chiude un progetto

Qualsiasi progetto è limitato da costo e tempo e livello di qualità.

Con l'approccio waterfal tutto viene stabilito a priori: durata del progetto, costo del progetto, numero di funzioni (FP), etc. Gli impegni assunti vanno rispettati. Eventuali sforamenti vengono gestiti, più che con il buon senso, in base alle clausole contrattuali sottoscritte (change management, penali, incentivi, etc.). In sostanza vanno gestite le varianze, mentre tutto dovrebbe essere già previsto.

Con l'approccio Agile è esattamente il contrario. Non è noto all'inizio la quantità di lavoro da eseguire, né il numero di funzioni da realizzare. Non essendo definito l'ambito del progetto è impossibile stimare tempi e costi.

Come si esce da questa situazione?

Poiché il progetto non può avere una durata infinita, si fissa fin dall'inizio una data di chiusura del progetto oppure un tetto massimo di spesa o entrambi. Praticamente, si collabora nella realizzazione della migliore soluzione entro un certo tempo o entro una certa spesa. Finito il tempo o i fondi, il progetto deve essere considerato concluso.

Ciò è possibile perché con Agile ogni iterazione (*sprint*) aggiunge dei dettagli o delle funzionalità e ciò che è stato rilasciato è funzionante.

Se si chiude un progetto senza realizzare le ultime funzionalità, significa che quelle erano le meno importanti. L'utente non gli ha dato priorità.

L'importante è eseguire il numero di *sprint* promesso a inizio progetto.

Con Agile l'utente per prima manifesta le sue esigenze di base e la visione generale del futuro. Indica prima le funzionalità indispensabili e poi si dedica alle sofisticazioni o dettagli. Se qualche dettaglio resta fuori, non è un grande danno. L'utente è in grado di decidere cosa lasciare fuori, mentre con l'approccio waterfal non ha più questa discrezionalità in quanto a inizio progetto ha trasferito al team, tutti i dettagli, impegnandoli anche su aspetti marginali.

Paradosso delle misurazioni con i Frattali

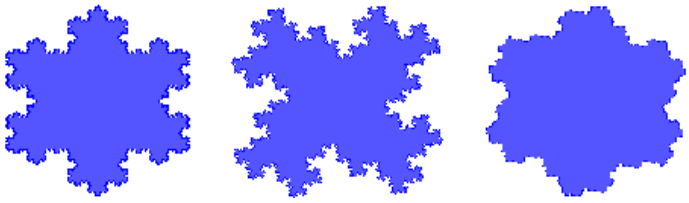
Gli aspetti marginali incidono notevolmente sui tempi ed il costo di un progetto.

Per stimare il costo, la durata e l'impegno necessari per realizzare un progetto abbiamo bisogno di stabilire delle unità di misura in termini di costo, durata e impegno.

Benoit Mandelbrot ha assimilato la misurazione del perimetro della Gran Bretagna al calcolo dei frattali, osservando che ogni pezzo di costa ha le sue piccole baie,

insenature, capi per cui il perimetro dipende dai dettagli o dall'unità di misura utilizzata. Questa osservazione porta ad un paradosso.

Ad esempio, il Portogallo dichiara che il suo confine con la Spagna è di 1214 KM, la Spagna dichiara che lo stesso confine è lungo solo 987 KM. Ciò dimostra che con una unità di misura più grande otteniamo un perimetro inferiore rispetto a quello ottenuto con una unità di misura più piccola.



Fonte **Coastlines and Borderlines**:*http://library.thinkquest.org/26242/full/ap/ap4.html*

La figura a lato, con la tecnica dei frattali, dimostra che il perimetro della Gran Bretagna è di:

- KM 2350 con unità di misura 200KM
- KM 2775 con unità di misura 100KM
- KM 3425 con unità di misura 50KM

Fonte: http://en.wikipedia.org/wiki/File:Britain-fractal-coastline-combined.jpg

Con lo sviluppo software siamo di fronte ad un paradosso analogo.

Più dettagli raccogliamo e più grande appare il lavoro da eseguire, più particolari dobbiamo analizzare, disegnare e quindi il progetto apparirà sempre più grande.

Con Agile è come se utilizzassimo una unità di misura più grande. Sviluppiamo l'essenziale per fornire una soluzione minimale, non ancora soddisfacente, ma valida come punto di partenza. Successivamente scendiamo nei dettagli, in modo selettivo, in funzione delle priorità dell'utente, definendo gradualmente l'intero ambito del progetto in funzione della capienza di schedulazione e budget.

L'approccio tradizionale avrebbe elencato a priori tutte le attività necessarie, anche quelle riguardanti dettagli secondari, considerandoli parimenti necessari per arrivare alla soluzione. Falso! I dettagli possono essere utili per arrivare ad una soluzione completa, ma non sono indispensabili per avere una soluzione funzionante.

Spesso alcuni dettagli sono frutto di assunzioni fatte direttamente dall'analista, per cui all'atto pratico possono risultare anche inutili. Con l'approccio Agile i dettagli vengono ignorati se non c'è più tempo o più budget.

Invece, con l'approccio tradizionale, una volta integrati nel disegno della soluzione, non è più possibile escludere i dettagli di qualsiasi provenienza. La loro eliminazione potrebbe richiedere uno sforzo paragonabile alla loro implementazione.

Responsabilità dei requisiti

Con l'approccio tradizionale l'utente, una volta consegnati i requisiti, viene escluso dallo sviluppo, mentre gli sviluppatori assumono la responsabilità di aver compreso i requisiti.

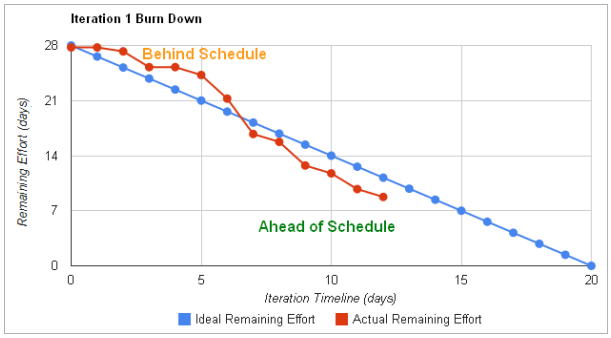
Se qualcosa non risulta chiara, il team di progetto torna ad intervistare l'utente oppure, come spesso accade, fa delle assunzioni. Spesso, gli sviluppatori assumono oltre ogni ragionevole misura, ingigantendo l'applicazione senza alcun vantaggio per nessuno.

Con l'approccio Agile, l'utente è coinvolto in tutte le fasi di sviluppo, per cui in ogni istante può indirizzare gli sviluppatori verso la soluzione ideale e non ipotetica. L'approccio Agile è al tempo stesso iterativo, incrementale e adattivo ed implica la simultaneità di specifiche, produzione e collaudo di piccoli rilasci continui.

L'utente, svolge un ruolo di primo piano, illustrando esempi di funzioni necessarie e indicando le giuste priorità. Vede i suoi esempi realizzati nell'arco di 2-4 settimane ed ha la possibilità di correggere il tiro se la soluzione non lo convince. La collaborazione tra sviluppatori e utente riduce le incomprensioni e gli errori, eliminando i ritardi.

Controllo della schedulazione

Verrebbe da dire che con Agile non esiste la schedulazione, ma non è così. Cosa è in esecuzione in un determinato momento è sempre noto al team ed anche a chi osserva il team, attraverso la tecnica detta "Information radiator", una bacheca sulla quale procede la scheda di ogni story. Intanto sono bene definiti il numero e la durata delle iterazioni. La schedulazione avviene all'interno delle singole iterazioni (sprint), tenendo sotto controllo il tempo rimanente alla conclusione dello *sprint*. Ogni giorno, si cerca di sapere se il tempo rimanente è sufficiente o meno. Questo metodo, detto **Burn Down** traccia l'avanzamento giornaliero del progetto valutando l'impegno necessario per completare l'iterazione e i giorni mancanti alla scadenza dell'iterazione. Se ci si discosta dalla linea ideale, bisogna intervenire e recuperare.



Inoltre, l'avanzamento viene tracciato in termini di funzionalità rilasciate nel tempo, una sorta di velocità di rilascio: funzionalità senza dettagli, poi dettagli aggiunti alle funzionalità, piccoli dettagli fino a soddisfare l'intero ambito del progetto se siamo fortunati. Se qualcosa resta fuori, vuol dire che non aveva molta importanza non a dire degli sviluppatori, ma dell'utente stesso.

Vito Madaio - esperto di project management di lunga data - Già Service Manager alla Camera dei deputati per Cap Gemini; Direttore Sistemi Informativi del Gruppo Skandia e Architetto di Sistemi di IBM Italia. Attualmente, da circa 9 anni Responsabile di **TenStep Italia** e **REP** del PMI, si occupa principalmente di Certificazioni PMP, CAPM e Agile PMI-ACPsm, oltre a diffondere l'adozione della Metodologia di Project Management TenStep. **Vito Madaio, PMP** **TenStep Italia**