Dallo sviluppo waterfall a quello iterativo, una transizione impegnativa per i Responsabili di progetto

Philippe Kruchten

White paper del software Rational

TP 173, 5/00

Rational®
the software development company

Indice

Introduzione

RUP (Rational Unified Process) predilige un approccio iterativo o a spirale al ciclo di vita di sviluppo del software, poiché si è rivelato spesso superiore all'approccio waterfall sotto molti aspetti. Ma non si deve pensare che i molti vantaggi di un ciclo di vita iterativo non abbiamo un prezzo. Lo sviluppo iterativo non è una bacchetta magica che risolve tutti gli eventuali problemi e difficoltà di uno sviluppo software. I progetti non sono più semplici da impostare o pianificare solo perché iterativi, il responsabile di progetto avrà un compito più difficile, specialmente durante il primo progetto iterativo e in particolare nelle prime iterazioni di quel progetto, quando i rischi e la possibilità di errore sono maggiori. In questo documento, saranno descritte alcune delle difficoltà dello sviluppo iterativo dalla prospettiva del responsabile di progetto e alcune delle comuni "insidie" o trabocchetti in cui si sono imbattuti molti responsabili di progetto, come riscontrato in base all'esperienza di Rational e grazie a racconti di colleghi.

Sviluppo iterativo

I processi di sviluppo software classici seguono il ciclo a cascata, come viene illustrato nella seguente figura. Nell'approccio mostrato in Figura 1, lo sviluppo procede in modo lineare dalle analisi di requisiti attraverso progettazione, verifica di codice e di unità, verifica di sottosistemi e di sistemi, con feedback limitati sui risultati della fase precedente.

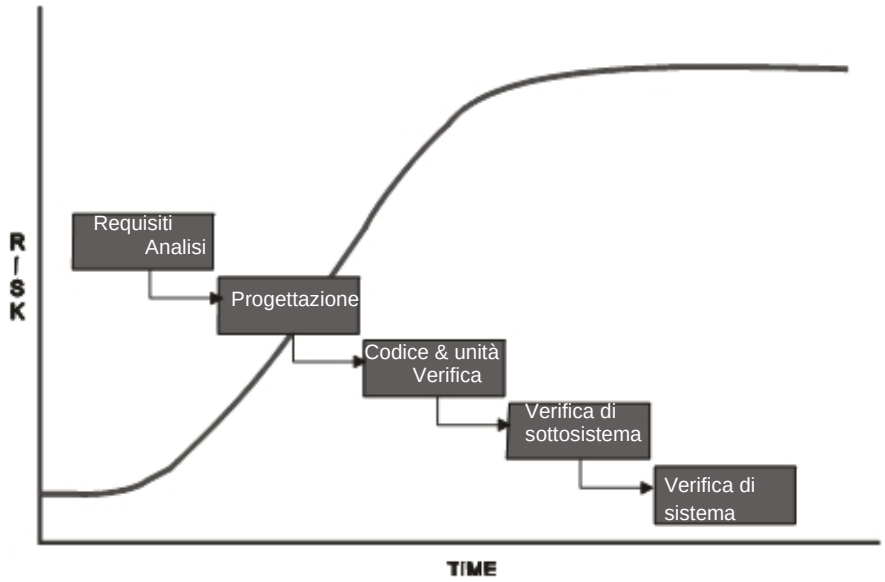

Figura 1: Il processo di sviluppo waterfall

Il problema principale di questo approccio è che protrae i rischi nelle fasi successive, quando risolvere gli errori commessi in precedenza comporta dei costi. È probabile che in una progettazione iniziale si producano degli errori rispetto ai requisiti principali, la successiva scoperta di questi difetti di progettazione tende a generare costi imprevisti e/o la cancellazione del progetto. L'approccio waterfall tende a mascherare i rischi reali del progetto finché è troppo tardi per cercare di arginarli.

Un'alternativa a questo tipo di approccio è il processo iterativo e incrementale, come si vede in Figura 2.

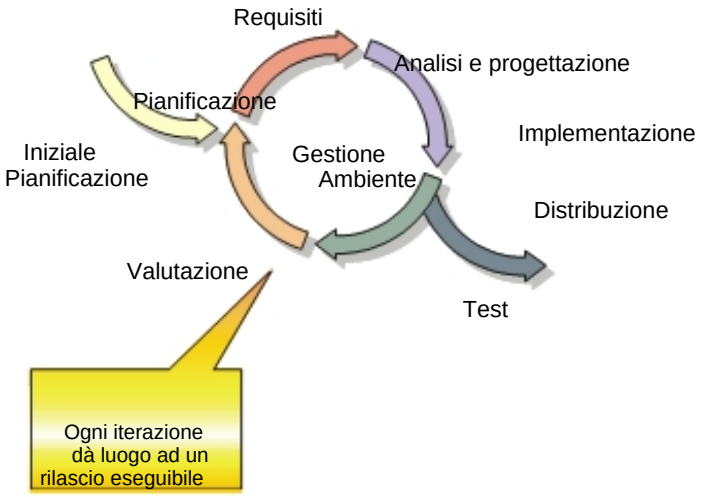

Figura 2: Un approccio iterativo allo sviluppo

In quest'approccio, costruito sul modello a spirale di Barry Boehm (si veda "Ulteriori letture"), l'identificazione dei rischi di un progetto è imposta all'inizio del ciclo di vita, dove è possibile attaccarli e affrontarli in maniera efficace e tempestiva. Quest'approccio è soggetto a continue scoperte, invenzioni e implementazioni e ogni iterazione obbliga il team di sviluppo a chiudere gli artefatti del progetto in modo prevedibile e ripetitivo.

Lato positivo: i benefici dello sviluppo iterativo

Rispetto al processo waterfall tradizionale, il processo iterativo presenta molti vantaggi:

1. Le gravi imprecisioni sono rese evidenti in anticipo nel ciclo di vita, dove è possibile affrontarli.
2. L'approccio incoraggia e consente feedback dell'utente, così da dedurre i veri requisiti del sistema.
3. Il team di sviluppo è costretto a concentrarsi su questi problemi fondamentali per il progetto e i membri del team sono protetti da questi problemi che potrebbero distrarre dai veri rischi del progetto.
4. Le verifiche continue e iterative consentono una valutazione oggettiva dello stato del progetto.
5. Le incongruenze tra requisiti, progettazioni e implementazioni sono individuate in tempo.
6. Il carico di lavoro del team, specialmente del team di verifica, è distribuito uniformemente nel ciclo di vita.
7. Tale approccio consente al team di potenziare le lezioni apprese e, di conseguenza, di migliorare il processo.
8. Gli stakeholder del progetto potranno avere prove concrete dello stato del progetto nel ciclo di vita.

Mitigazione dei rischi

Un processo iterativo consente di mitigare i rischi in anticipo poiché un'integrazione è in genere l'unico momento in cui i rischi vengono scoperti e risolti. Nell'eseguire le prime iterazioni si attraversano tutti i componenti del processo, mettendo in pratica molti aspetti del progetto compreso i tool, software già pronti e skill delle persone. I rischi percepiti si riveleranno innocui e ne verranno scoperti di nuovi.

Se per qualche ragione un progetto fallisce, estinguerlo velocemente per evitare sprechi di denaro, impegno e tempo, senza nascondere la testa nella sabbia a lungo, affrontare piuttosto i rischi. Tra gli altri rischi, come la costruzione di un prodotto sbagliato, esistono due categorie di rischi che un processo di sviluppo iterativo aiuta a mitigare in anticipo:

- Rischi di integrazione

- Rischi strutturali

Un approccio iterativo determina una struttura più solida poiché gli errori vengono corretti in varie iterazioni. Le imprecisioni sono individuate nelle prime iterazioni mentre il prodotto procede oltre l'inizio. I colli di bottiglia nelle prestazioni vengono individuati in un momento in cui è ancora possibile risolverli, piuttosto che scoprirli alla vigilia della consegna.

L'integrazione è solo un “big bang” alla fine del ciclo di vita, gli elementi invece vengono inseriti progressivamente. In realtà l'approccio iterativo qui consigliato implica un'integrazione quasi continua. Quello che solitamente era un lungo periodo di incertezza, che prendeva fino al 40% dell'impegno totale alla fine di un progetto, è ora suddiviso in una serie di integrazioni (da sei a nove) più piccole che iniziano con molti meno elementi da integrare.

Modifiche appropriate

È possibile prevedere varie categorie di modifiche:

- Modifiche nei requisiti

  L'approccio iterativo consente di prendere in considerazione i requisiti di modifiche, che in effetti cambieranno sistematicamente. Tali modifiche, insieme all'"instabilità dei requisiti" costituiscono da sempre la fonte principale dei problemi di un progetto, determinando ritardi nella consegna, mancate pianificazioni, insoddisfazione del cliente e frustrazione degli sviluppatori. Presentando agli utenti (o rappresentati di utenti) una prima versione del prodotto, è possibile assicurare una migliore fedeltà del prodotto al compito.

- Modifiche tattiche

  Un processo iterativo fornisce alla gestione un modo di attuare modifiche tattiche nel prodotto. Ad esempio, per competere con i prodotti esistenti si può decidere di rilasciare in anticipo un prodotto con funzioni ridotte al fine di contrastare una mossa da parte di un concorrente, oppure adottare un altro fornitore per una data tecnologia. È inoltre possibile riorganizzare i contenuti di un'iterazione per alleviare un problema di integrazione che deve essere risolto da un fornitore.

- Modifiche tecnologiche

  In misura minore, un approccio iterativo permette di apportare modifiche tecnologiche. È possibile farlo durante la fase di elaborazione, ma è preferibile evitare questo tipo di modifiche durante la costruzione e la transizione in quanto rischiose.

Imparare strada facendo

Un vantaggio del processo iterativo è che gli sviluppatori possono imparare durante il percorso e le varie competenze e specializzazioni vengono impiegate più pienamente durante l'intero ciclo di vita. Ad esempio, i verificatori iniziano prima i test, gli scrittori tecnici scrivono prima e così via, mentre in uno sviluppo non iterativo le stesse persone dovrebbero attendere per iniziare il lavoro, facendo piani su piani. Le esigenze di formazione, o di un ulteriore aiuto (magari esterno), vengono individuate presto durante le revisioni della valutazione.

Il progetto stesso può essere migliorato e perfezionato durante lo sviluppo. La valutazione al termine di un'iterazione guarda allo stato del progetto da una prospettiva del prodotto/pianificazione e analizza ciò che dovrebbe essere modificato nell'organizzazione e nei processi di modo che funzionino meglio nell'iterazione successiva.

Maggiori opportunità per il riutilizzo

Un processo iterativo facilita il riutilizzo degli elementi del progetto, essendo più facile identificare parti comuni quando vengono parzialmente progettate o implementate piuttosto che doverle identificare in comune in anticipo. L'identificazione e lo sviluppo di parti riutilizzabili è difficoltoso. Le revisioni di progettazione nelle iterazioni iniziali consentono agli architetti di identificare un riutilizzo potenziale non previsto e di sviluppare e maturare un codice comune per le iterazioni successive. Durante le iterazioni e la fase Elaborazione vengono trovate soluzioni comuni a problemi comuni e sono identificati i pattern e i meccanismi strutturali applicati al sistema.

Migliore qualità generale

Il prodotto di un processo iterativo sarà di migliore qualità generale rispetto ai prodotti derivanti da un processo sequenziale convenzionale. Il sistema sarà stato verificato più volte, migliorando la qualità del test. I requisiti saranno stati perfezionati e saranno più vicini alle esigenzse reali dell'utente; al momento della consegna il sistema sarà già stato a lungo in esecuzione.

Lato negativo: ostacoli imprevisti e insidie comuni

Lo sviluppo iterativo non significa necessariamente meno lavoro e una pianificazione più breve. Il vantaggio principale consiste nell'apportare maggior prevedibilità al risultato e alla pianificazione, darà vita a prodotti di maggiore qualità e soddisferà le reali esigenze degli utenti, poiché si avrà avuto il tempo di sviluppare i requisiti, la progettazione e l'implementazione.

Lo sviluppo iterativo implica in realtà una maggiore pianificazione e probabilmente comporterà un maggiore carico per il responsabile del progetto: ci sarà la necessità di sviluppare un piano generale con conseguenti piani dettagliati per ogni iterazione. Implica inoltre una continua negoziazione di compromessi tra il problema, la soluzione e il piano. Avverrà inoltre in anticipo una pianificazione strutturale più approfondita. Gli artefatti (piani, documenti, modelli e codici) dovranno essere modificati, riesaminati e approvati ripetutamente ad ogni revisione. Le modifiche tattiche o di ambito costringeranno a continue ripianificazioni, anche la struttura del team dunque dovrà essere leggermente modificata ad ogni iterazione.

Insidia: pianificazione troppo dettagliata fino alla fine

È praticamente inutile costruire un piano dettagliato end-to-end, se non come esercizio di valutazione dell'involucro della pianificazione e delle risorse. Tale piano sarà divenuto obsoleto prima di raggiungere la fine della prima iterazione. Prima di avere un'architettura
in posizione ed una presa stretta dei requisiti, che ricorrono nel punto cardine LCA, non è possibile costruire un piano realistico.

Dunque, integrare precisione nella pianificazione si commisura alla conoscenza dell'attività, avendo pianificato l'artefatto o l'iterazione. I piani a breve termine sono più dettagliati e rifiniti, quelli a lungo termine sono gestiti in formato dettagliato.

Contrastare la pressione che potrebbe essere causata da una gestione male informata o sconosciuta, nel tentativo di sviluppare un “piano generale esauriente". Istruire i responsabili e spiegare loro la nozione di piano iterativo e l'inutilità di prevedere dettagli per il futuro. Un'utile analogia: si pensi ad un'automobile che viaggia da New York a Los Angeles. Nonostante il conducente abbia programmato l'intero tragitto saranno necessarie le istruzioni per uscire dalla città e immettersi sulla strada. Programmare i dettagli precisi del tragitto attraverso il Kansas, nonché dell'arrivo in California, è inutile, in quanto lungo la strada si potrebbero trovare lavori in corso e si dovrà scegliere una strada alternativa.

Riconoscimento in anticipo della rilavorazione

In un approccio waterfall si va incontro ad una lunga rilavorazione, come noiosa e imprevista conseguenza dei molti errori rilevati nelle fasi finali di test e integrazione. Peggio, si potrebbe scoprire che la maggior parte "danni" derivano da errori nella progettazione, che si è cercato di mitigare durante l'implementazione costruendo soluzioni che hanno arrecato ulteriori danni.

In un approccio iterativo si conosce in anticipo l'esigenza di rilavorazione, che all'inizio sarà maggiore per via dei problemi nei precedenti prototipi strutturali da eliminare. Inoltre, al fine di costruire prototipi eseguibili, dovranno essere creati stub e strutture da sostituire in seguito con implementazioni più mature e robuste. In un buon progetto iterativo, la percentuale di materiale di scarto e di rilavorazione diminuisce rapidamente, le modifiche saranno meno frequenti mentre l'architettura si stabilizza e i problemi vengono risolti.

Insidia: il progetto non è convergente

Lo sviluppo iterativo non implica scarti e rilavorazione di qualunque cosa ad ogni iterazione, questi anzi devono diminuire da un'iterazione all'altra specialmente dopo che l'architettura è confrontata con la linea di base nel punto cardine LCA. Gli sviluppatori tendono a sfruttare lo sviluppo iterativo per attuare perfezionamenti: introdurre una tecnica migliore, eseguire rilavorazioni, ecc. Il responsabile di progetto deve vigilare perché non avvengano rilavorazioni di elementi non danneggiati e in buone condizioni. Inoltre, mentre il team cresce di numero e alcuni vengono spostati, arrivano nuove persone, le quali tendono ad avere idee proprie
su come si debba lavorare. Analogamente i clienti (o i rappresentanti nel progetto: marketing,
gestione del prodotto) potrebbero approfittare della libertà offerta dallo sviluppo iterativo per eseguire modifiche e/o per modificare o aggiungere requisiti senza limiti. Tale effetto è definito “instabilità dei requisiti”. Il responsabile di progetto deve essere anche in questo caso risoluto nell'effettuare compromessi e nel negoziare le priorità. Nel punto cardine LCA, i requisiti sono definiti e, a meno che la pianificazione e il budget siano rinegoziati, ogni modifica ha un costo finito.

Ricordare che "La perfezione è nemica del bene" (originale in Francese: “Le mieux est l’ennemi du bien”).

Insidia: iniziare e decidere strada facendo

Lo sviluppo iterativo non significa uno sviluppo costantemente approssimativo. Non si dovrebbe iniziare la progettazione e la scrittura di codice per tenere il team impegnato o con la speranza di imminenti obiettivi. È necessario definire obiettivi netti, metterli per iscritto e ottenere l'approvazione di tutti, poi perfezionarli, estenderli e di nuovo ottenere l'approvazione di tutti. Il lato positivo è che nello sviluppo iterativo è necessario che tutti i requisiti siano decisi prima dell'inizio della progettazione, della scrittura del codice dell'integrazione, della verifica e della convalida.

Insidia: essere vittime del proprio successo

Un rischio significativo può presentarsi verso la fine del progetto, nel momento in cui le richieste del cliente si fanno pressanti. Com'è noto a questo punto, l'utente passa dall'idea che nulla gli sarà mai consegnato alla pretesa che gli venga realizzato anche l'impossibile. La buona notizia è che le sue aspettative nei confronti del progetto sono migliorate: se lunedì l'utente si sarebbe accontentato di un prodotto semplice, il martedì pretende invece che tutte le proprie richieste vengano integralmente soddisfatte. E questa è invece la cattiva notizia: all'improvviso tutto ciò che poteva essere inserito nella seconda versione beta deve necessariamente fare parte del primo rilascio. Improvvisamente questo diventa il problema principale, il responsabile di progetto cessa di preoccuparsi di produrre una funzionalità minima accettabile e si interessa di una situazione in cui ogni requisito ultimo è ora "essenziale" per la prima produzione. È quasi come se, tutti gli elementi in sospeso venissero elevati ad uno stato di priorità "A". La verità è che si ha sempre lo stesso numero di cose da fare nella stessa quantità di tempo. Sebbene la percezione esterna sia cambiata, l'assegnazione di priorità resta ancora molto importante.

Se, in questo momento cruciale, il responsabile di progetto perde il controllo e comincia a cedere a tutte le richieste mette di nuovo a rischio il progetto, a questo punto che deve continuare ad essere deciso e non soccombere alle richieste. Anche negoziare qualcosa di nuovo per qualcosa che è stata eliminata aumenta il rischio. Senza controllo si può raccogliere una sconfitta invece di un successo.

Dare priorità al software

In un approccio waterfall non viene data molta enfasi alle "specifiche" (ad esempio, la descrizione di spazio-problema) e non non vengono completate, rifinite e terminate. Nel processo iterativo, il software da sviluppare ha la priorità. L'architettura software (ad esempio, la descrizione spazio-soluzione) guida le decisioni del primo ciclo di vita. I clienti non acquistano specifiche, al centro dell'attenzione c'è il prodotto software, che si evolve insieme alle specifiche. Quest'attenzione sul software ha un certo effetto sui vari team: i tester, ad esempio, potrebbero essere abituati e ricevere delle specifiche stabili e complete con molte notizie per avviare la verifica, laddove in uno sviluppo iterativo devono iniziare a lavorare subito mentre le specifiche e i requisiti sono ancora in evoluzione.

Insidia: troppa attenzione agli artefatti di gestione

Alcuni responsabili dicono “Io sono un responsabile di progetto, per cui devo concentrarmi per ottenere i migliori artefatti di gestione che sono la chiave di tutto". Non è esattamente così, sebbene una buona gestione sia fondamentale, il responsabile di progetto deve far si che il prodotto finale sia il massimo che si poteva avere. La gestione del progetto non deve essere una giustificazione nel caso in cui il progetto debba fallire, nonostante una buona direzione. Analogamente, ci si potrebbe concentrare sullo sviluppo di migliori specifiche perché in passato si è verificata un'esperienza di cattiva gestione, ma sarebbe inutile se il prodotto dovesse risultare instabile, lento e fragile.

Affrontare per prima i problemi difficili

In un approccio waterfall, molti dei problemi difficili, delle cose rischiose e ignorate vengono indirizzati alla soluzione nel processo di pianificazione durante la temuta attività d'integrazione del sistema. Ciò rende la prima metà del progetto una fase relativamente tranquilla, durante la quale i problemi sono affrontati su carta, senza il coinvolgimento di troppi stakeholders (tester, ecc.), piattaforme hardware, utenti effettivi o ambiente reale. Poi improvvisamente il progetto entra nella difficile fase di integrazione e tutto si rovina. Nello sviluppo iterativo, la pianificazione è basata sui rischi e sugli elementi sconosciuti, dunque tutto dovrebbe andare bene sin dalla partenza. Alcune questioni difficili, tecniche e di basso livello devono essere affrontate immediatamente, piuttosto che lasciate in sospeso per una soluzione successiva. In breve, in uno sviluppo iterativo non è possibile mentire (a se stessi e agli altri) a lungo. Un progetto software destinato a fallire dovrebbe terminare prima in un approccio iterativo.

Si potrebbe fare un'analogia con un corso universitario in cui il professore passa la prima parte del semestre a fornire concetti relativamente di base, dando l'impressione che sia un corso semplice che permette agli studenti di avere buoni voti con un minimo sforzo.

Poi improvvisamente il corso accelera verso la fine del semestre, il professore affronta gli argomenti più complessi in modo superficiale prima dell'esame finale. A questo punto tutti gli studenti cedono alla pressione, sostenendo un esame insoddisfacente. La cosa peggiore è che, se non si sostituisce il professore, questo disastro si ripeterà anno dopo anno e classe dopo classe. Una scelta più intelligente è di concentrare il 60% del lavoro nella prima sessione e di fornire il materiale più difficile. La correlazione con la gestione di un progetto iterativo è di non sprecare tempo prezioso all'inizio per risolvere non-problemi ed eseguire compiti irrilevanti. Il motivazione più comune per il fallimento tecnico all'avvio è che si spende tutto il tempo nel fare cose semplici."

Insidia: nascondere la testa sotto la sabbia

Si tende a dire "Questo è un problema delicato, ci vorrà un pò di tempo per risolverlo. Rimandiamone la soluzione, così avremo più tempo per pensarci". Il progetto si impegna così ad affrontare i compiti più semplici, senza dedicare la dovuta attenzione ai veri problemi. Quando si arriva al punto in cui è necessaria una soluzione, si devono prendere decisioni affrettate, o il progetto deraglia. Si desidera affrontare la questione immediatamente. Se per qualche ragione un progetto fallisce, estinguerlo velocemente per evitare sprechi di denaro, impegno e tempo."

Insidia: dimenticare i nuovi rischi

Nell'attuare un'analisi del rischio alla fase iniziale e utilizzarla per la pianificazione, ci si può dimenticare dei rischi che emergono in seguito nel progetto, che certamente si presenteranno. I rischi dovrebbero essere rivalutati costantemente, ogni settimana o ogni mese. L'elenco dei rischi stilato all'inizio serve solo come punto di partenza. È solo quando il team inizia con uno sviluppo concreto (principalmente con il software) che si scopriranno molti altri rischi.

Disaccordi dovuti a diversi modelli di ciclo di vita

Il responsabile di un progetto iterativo vedrà spesso dei contrasti tra il proprio ambiente e altri gruppi, come la gestione principale, i clienti e gli appaltatori, che non hanno adottato, o addirittura compreso, la natura dello sviluppo iterativo. Questi si aspettano degli artefatti completi e congelati al punto cardine principale, non vorrebbero riesaminare i requisiti volta per volta, sono spaventati dalla rielaborazione e non comprendono lo scopo o il valore di alcuni prototipi strutturali eseguiti male. Ritengono che l'iterazione sia senza senso, che si diverta con la tecnologia, che sviluppi il codice prima che le specifiche siano stabilite e che sviluppi un codice con noncuranza.

Come minimo, rendere chiaramente visibili i propri piani ed intenzioni. Se lasciamo l'approccio iterativo in mente e su poche righe scritte su lavagne condivise con il team, si incorrerà in successivi problemi.

Il responsabile di progetto deve proteggere il team da attacchi esterni e politici al fine di evitare disordini e distrazioni da parte del mondo esterno, deve fare da ammortizzatore. Per mantenere il controllo della situazione il responsabile di progetto deve costruire un'immagine di credibilità e affidabilità per l'esterno. Per cui, la visibilità e "tracciare il piano" restano importanti, specialmente alla luce del "piano" visto da molti come qualcosa di non convenzionale. Infatti è in realtà molto più importante.

Insidia: diversi gruppi che operano sulle proprie pianificazioni

È più semplice e facile che tutti i gruppi (o team o subappaltatori) operino secondo la stessa fase e piano di iterazione. I responsabili di progetto ritengono di ottimizzare il tempo sintonizzando la pianificazione di ogni team individuali, ciascuna delle quali finisce per avere il proprio piano di iterazione. Se ciò accade, tutti i vantaggi ottenuti andranno perduti in seguito e i team saranno costretti a sincronizzarsi col gruppo più lento. Per quanto fattibile, posizionare tutti sulla stessa lunghezza d'onda.

Insidia: offerta di prezzo fisso durante la fase Inizio

Molti progetti sono sottoposti a offerte per lo sviluppo contrattuale troppo presto, a metà della fase Inizio. In uno sviluppo iterativo, il momento migliore per le offerte di tutte le parti è al punto cardine LCA (alla fine dell'elaborazione). Non esiste una ricetta magica: è necessaria una conoscenza e una negoziazione degli stakeholder, che devono mostrare i vantaggi di uno sviluppo iterativo ed eventualmente un processo di offerta a passo doppio.

Calcolare i progressi è differente

Il sistema tradizionale di valore acquisito per calcolare il progresso è diverso, poiché gli artefatti non sono sempre completi e congelati, ma sono rilavorati in vari incrementi. Se un artefatto ha un certo valore nel sistema di valore assorbito e si è guadagnato credito per questo nella prima first iterazione, in cui è stato creato, la valutazione di progresso è allora oltremodo ottimistica. Se si ottiene credito solo quando diventa stabile in due o tre iterazioni successive, la misura di progresso diventa invece troppo pessimistica. Quando si utilizza questo tipo di approccio per monitorare il progresso, gli artefatti devono essere divisi in diverse parti, ad esempio documento iniziale (40%), prima revisione (25%), seconda revisione (20%), documento finale (15%). Ad ogni parte deve essere dato un valore. È possibile utilizzare il sistema di valore assorbito senza aver completato ogni elemento.

Un'alternativa può essere quella di organizzare il valore assorbito sulle iterazioni stesse e calcolare il valore secondo criteri di valutazione. Quindi i momenti di traccia intermedi (in genere mensili) riportati nella Valutazione di stato saranno costruiti sul Piano di iterazione. Questo richiede una traccia di artefatti più precisa delle specifiche di requisiti tradizionali, di progettazione, ecc., poiché si sta eseguendo la traccia del completamento di vari casi d'uso, casi di test e così via.

Come dice Walker Royce: “Un responsabile di progetto dovrebbe essere più concentrato sulla misurazione e monitoraggio delle modifiche nei requisiti, nella progettazione, nel codice, piuttosto che nel contare le pagine dei testi e delle righe del codice. (Vedere “Ulteriori letture" riportate in seguito). Joe Marasco aggiunge: “Attenti non solo alle modifiche, ma anche al contenutos. Le cose che cambiano più volte per poi ritornare allo stesso punto di partenza sono il sintomo di problemi più profondi.”

Da un punto di vista positivo, un software concreto che esegue in modo anticipato può essere utilizzato da un responsabile di progetto saggio per ottenere una maggiore credibilità. Può mostrare progressi in modo più significativo rispetto a pratiche revisionate con centinaia di caselle selezionate. Inoltre, gli ingegneri preferiscono “dimostrazioni di come funzioni" a “documentazioni su come dovrebbe funzionare”. Prima dimostrare, poi documentare.

Stabilire il numero, la durata e il contenuto delle iterazioni

Cosa fare prima? Un responsabile nuovo allo sviluppo iterativo ha spesso difficoltà nel decidere il contenuto delle iterazioni. Inizialmente, tale progettazione è guidata dal rischio, tecnico e programmatico e dalla criticità delle funzioni del sistema in costruzione (RUP fornisce le linee guida per decidere il numero e la durata delle iterazioni). I criteri evolvono anche durante il ciclo di vita. In costruzione, la pianificazione mira a completare certe funzioni o certi sottosistemi, in transizione mira invece a risolvere problemi e ad aumentare la solidità e la prestazione.

Insidia: spingere troppo nella fase Inizio

Abbiamo parlato in precedenza del fatto di non affrontare i problemi difficili per primi. D'altro canto, esagerare nella direzione opposta è ugualmente una motivo di fallimento. Esiste una tendenza a voler affrontare troppi problemi e coprire troppi campi nelle prime iterazioni, ma manca qui un riconoscimento di altri fattori: un team deve essere formato (istruito), deve acquisire nuove tecniche e nuovi strumenti, spesso il dominio del problema è nuovo per molti sviluppatori. Ciò causa spesso gravi sovraccarichi nella prima iterazione, che potrebbero screditare l'intero approccio iterativo oppure l'aborto dell'iterazione dichiarata completa quando niente è in esecuzione, il che significa dichiarare "vittoria" in un punto in cui non si può trarre nessuna lezione,
mancando i vantaggi dello sviluppo iterativo.

Nel dubbio, o nell'affrontare queste crisi, sminuirle in qualche modo (ciò riguarda il problema, la soluzione, il team). Ricordare che la completezza è una questione di un ciclo di vita successivo. L'“incompletezza adeguata" dovrebbe essere il primo problema del ciclo di vita del responsabile. Se la prima iterazione contiene troppi scopi, suddividerli in due iterazioni e poi dare priorità in modo fermo
agli obiettivi da raggiungere per primi.

È meglio mirare prima a obiettivi più semplici e tradizionali nel progetto. Notare che non abbiamo detto facili. Raggiungere un risultato concreto all'inizio del progetto aiuta a sollevare il morale. Molti progetti che mancano del primo punto cardine, non potranno recuperarlo, quelli che mancano di molti punti cardine sono condannati nonostante molti sforzi eroici, nel pianificare dunque, assicurarsi di non aver saltato un primo punto cardine.

Insidia: troppe iterazioni

Per prima cosa un progetto non dovrebbe confondere le costruzioni quotidiane o settimanali con le iterazioni. Poiché esiste un limite stabilito nella pianificazione, monitoraggio e valutazione di un'iterazione, un'organizzazione che non ha familiarità con quest'approccio non dovrebbe tentare l'iterazione con alta frequenza per questo primo progetto. La durata di un'iterazione dovrebbe anche tenere in conto la misura dell'organizzazione, il grado di distribuzione geografica e il numero delle diverse organizzazioni coinvolte. Rivisitare la nostra regola pratica “sei più o meno tre”.

Insidia: iterazioni in sovrapposizione

Un'altra insidia comune è che le iterazioni si sovrappongano troppo. Pianificare la nuova iterazione nell'ultimo quinto dell'attuale iterazione mentre si tenta di avere una significativa sovrapposizione di attività (ad esempio iniziare un'analisi dettagliata, progettare e codificare la successiva iterazione prima di terminare quella attuale e di apprendere da questa) può sembrare interessante quando si guarda un grafico GANTT, ma arrecherà dei problemi. Ma qualcuno non si impegnerà a eseguire e terminare il proprio contributo all'attuale iterazione, non saranno sensibili al fatto di stabilire le cose o decideranno di prendere in considerazione ogni feedback solo nell'iterazione successiva. Alcune parti del software non saranno pronte a supportare il lavoro che è stato posticipato ecc. Sebbene sia possibile indirizzare forza lavoro per eseguire lavori non collegati all'iterazione attuale, è consigliabile farlo in modo eccezionale. Tale problema è spesso creato da un ristretto numero di skill dei membri di un'organizzazione o da una molto rigida: Joe è un analista e questa è l'unica cosa che sa o vuole fare, non vuole partecipare alla progettazione, all'implementazione o alla verifica. Altro esempio negativo: un vasto progetto di controllo e comando ha iterazioni talmente sovrapposte che in pratica sono in esecuzione parallelamente e allo stesso momento richiedendo alla gestione di dividere l'intero staff nelle iterazioni, impedendo l'integrazione di lezione apprese da quelle precedenti.

Vedere Figura 3 per alcuni esempi di pattern di iterazione non produttivi.

Team non sincronizzati -> confusione

Prima iterazione troppo carica -> panico

Troppa sovrapposizione -> no feedback

Figura 3: Alcuni esempi di pattern di iterazione pericolosi

Un buon Responsabile di progetto e un buon Architetto

Per avere successo, un progetto software necessita sia di un buon Responsabile di progetto che di un buon Architetto. La migliore gestione possibile e lo sviluppo iterativo non porteranno ad un prodotto di successo senza una buona architettura. Viceversa, un'architettura fantastica fallirà se il progetto non è ben gestito. È dunque una questione di equilibrio, focalizzarsi unicamente sulla gestione del progetto non produrrà risultati. Il responsabile di progetto non può semplicemente ignorare l'architettura: è necessaria sia l'abilità di architettura che di dominio per determinare il 20% delle cose che devono andare nelle prime iterazioni.

Insidia: utilizzare la stessa persona da responsabile di progetto e architetto

Utilizzare la stessa persona come responsabile di progetto e architetto funzionerà solo per piccoli progetti (5-10 persone). Per compiti più vasti, la stessa persona che svolge i due ruoli avrà come conseguenza un progetto che non sarà ben gestito né ben strutturato. Per prima cosa, il ruolo richiede diverse skill. Secondariamente, i ruoli sono più di un lavoro a tempo pieno. Di conseguenza, il responsabile di progetto e l'architetto devono coordinate quotidianamente, comunicare tra loro e

fare compromessi. I ruoli sono simili a quelli di regista e produttore. Entrambi lavorano per un obiettivo comune ma sono responsabili di aspetti totalmente differenti. Se la stessa persona svolge due ruoli, il progetto difficilmente riesce.

Conclusioni

A questo punto ci si potrebbe sentire scoraggiati: così tanti problemi, così tante insidie in cui cadere. Se risulta così difficile pianificare e eseguire uno sviluppo iterativo, perché scoraggiarsi? Non perdetevi d'animo, esistono alcune ricette e tecniche per affrontare tutti questi inconvenienti e i risultati sono maggiori dei lati negativi in termini del raggiungimento prodotti software di maggiore qualità e affidabili. Alcuni spunti chiave: “Affronta i rischi in modo efficace o loro attaccheranno te" (Attack the risks actively or they will attack you). (Dal libro di Tom Gilb elencato nei Riferimenti e Ulteriori letture.) Il software ha la priorità. Riconoscere gli scarti e la rilavorazione. Scegliere un responsabile di progetto e un architetto che lavorino insieme. Sfruttare i vantaggi dello sviluppo iterativo.

Il modello waterfall è semplificativo per il responsabile e difficoltoso per il team di progettazione. Lo sviluppo iterativo è molto più allineato con il lavoro degli ingegneri software, ma a discapito della facilità della gestione. Considerato che molti team sono composti da una rapporto di 5 ingegneri per 1 responsabile (o più), questo è un grande compromesso.

Sebbene lo sviluppo iterativo risulterà più duro di altri approcci tradizionali la prima volta che verrà utilizzato, genera dei concreti vantaggi a lungo termine. Una volta presa la mano, si potrà constatare di essere un responsabile più capace e sarà più semplice gestire progetti complessi e vasti. Una volta ottenuto un intero team che capisce e pensa in modo iterativo, il metodo crescerà molto più velocemente rispetto ad un approccio tradizionale.

Nota: John Smith, Dean Leffingwell, Joe Marasco e Walker Royce mi sono stati d'aiuto nella stesura di questo documento, mettendo a disposizione le loro esperienze nel campo della gestione dei progetti iterativi. Parte di questo documento è inserito nel Capitolo 6 del nuovo libro del collega Gerhard Versteegen sullo sviluppo software (vedere Riferimenti e Ulteriori letture).

Informazioni sull'autore

Philippe Kruchten è entrato a far parte di Rational Software nel 1987 ed è attualmente un membro di Rational, con sede a Vancouver, B.C. Già Direttore e Responsabile Generale del Dipartimento Process Business, ha guidato lo sviluppo del Rational Unified Process. Oltre a focalizzarsi sull'architettura e progettazione software, è impegnato nelle pratiche di progettazione software e nel processo di sviluppo. È laureato in ingegneria meccanica e ha seguito un dottorato in informatica presso istituzioni francesi.

Riferimenti e ulteriori letture

1. Rational Unified Process 2000 , Rational Software, Cupertino, Ca., 2000.

2. Barry W. Boehm, “A Spiral Model of Software Development and Enhancement,” Computer, maggio 1988, IEEE, pp.61-72.

3. Tom Gilb, Principles of Software Engineering Management, Addison-Wesley, 1988.

4. Philippe Kruchten, The Rational Unified Process—An Introduction, Addison Wesley Longman, 1999.

5. Walker Royce, Software Project Management—A Unified Approach, Addison Wesley Longman, 1999.

6. Gerhard Versteegen, Projektmanagement mit dem Rational Unified Process, Springer-Velag, Berlino, 2000.

Rational®
the software development company

Sedi principali:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Numero verde: (800) 728-1212
E-mail: info@rational.com
Sito Web: www.rational.com
Sito internazionale: www.rational.com/worldwide