# RUP e XP a confronto

John Smith

White Paper del software Rational

TP 167, 5/01

Rational®
the software development company

# Indice

## Introduzione

Questo documento mette a confronto RUP (Rational Unified Process), un framework di processo perfezionato nel corso degli anni dal software Rational e utilizzato da numerosi progetti software, dai più piccoli ai più grandi, ed XP (Extreme Programming), un approccio di sviluppo software che viene sempre più riconosciuto come metodo efficace per la costruzione di sistemi più piccoli in un ambiente dai requisiti mutevoli.

Il confronto riguarda la natura e lo stile di entrambi e ne descrive il processo, la struttura interna, i presupposti e l'aspetto. Il documento esamina inoltre i principali obiettivi di ciascuno dei due approcci e le conseguenze del loro utilizzo.

La maggior parte dei processi hanno elementi in comune che rendono possibile un confronto sistematico. Richiedono sequenze o gruppi di attività, che vengono eseguite da ruoli (in genere singoli individui o team) per creare artefatti o prodotti di lavoro da consegnare tutti, o in parte, a un cliente. In gran parte dei processi le istanze del processo hanno una dimensione temporale, con un inizio ed una fine, oltre ad interessanti punti cardine intermedi che rappresentano lo svolgimento di attività significative (o cluster di attività) e la produzione di artefatti associati. In base a ciò, questo documento esplora le seguenti dimensioni di processo, utilizzandole ai fini del confronto:

- Assegnazione dei tempi e degli impegni — descrive come viene organizzato il processo in termini di tempo e quale tipo di impegno viene richiesto al personale interessato.
- Artefatti — confronta i prodotti di lavoro, ossia ciò che viene prodotto nel corso di progetti basati su RUP o XP.
- Attività — illustra in che modo ogni processo intende creare gli artefatti.
- Discipline — confronta i modi in cui XP e RUP delineano le principali problematiche nella progettazione di software.
- Ruoli — esplora le differenze tra i ruoli (che svolgono le attività) di RUP e quelli che caratterizzano un team di XP.

Lo scopo di questo documento è chiarire le rispettive posizioni di RUP e XP nello spettro del processo per osservare cosa offre l'uno all'altro e smentire la convinzione secondo cui XP è un'alternativa leggera, e quindi preferibile, alla pesantezza di RUP.

Le principali fonti di informazione su XP utilizzate per redigere questo documento sono tre libri della serie XP di Addison-Wesley:

- Extreme Programming Explained [Beck00]
- Extreme Programming Installed [Jeffries01]
- Planning Extreme Programming [Beck01]

La scelta è ricaduta su questi libri in quanto rappresentano i repository più ovvi di informazioni su XP dai quali inizieranno molti dei potenziali lettori interessati all'argomento. Erano significativi anche altri libri ma sfortunatamente non disponibili quando è stato scritto questo documento. La principale fonte di informazioni su RUP è lo stesso prodotto RUP di Rational Software Corporation.

Questo libro non intende essere un supporto a XP o RUP e sarà dunque più facilmente letto da coloro che conoscono bene questi approcci o che hanno accesso a parte del materiale di riferimento; ad esempio, [Beck00] nel caso di XP e [Kruchten00] di RUP.

## Assegnazione dei tempi e degli impegni

RUP riguarda progetti (per lo sviluppo di software) il cui ciclo di vita è suddiviso in fasi denominate Inizio, Elaborazione, Costruzione e Transizione. Ogni fase è a sua volta suddivisa in iterazioni, ciascuna delle quali può richiedere svariate build.

Nella maggior parte dei progetti di RUP, la durata di un'iterazione varia dalle due settimane ai sei mesi[1], mentre le iterazioni del ciclo di vita di un progetto sono da tre a nove. Quindi, in base a queste impostazioni predefinite, RUP interessa progetti la cui durata è compresa fra le sei settimane e i 54 mesi.

Le iterazioni di XP hanno una durata media di due settimane. I rilasci di XP sono di due mesi (o poco più); sono definiti dal cliente e rilasciati allo stesso. In termini di durata, i rilasci XP sono simili alle iterazioni di RUP (almeno durante l'Elaborazione o la Costruzione[2], in progetti adatti a XP; ossia, per quelli svolti da un team di circa 10 persone[3], e basati su riferimenti strutturali prestabiliti).[4]

Per illustrare tale concetto, prendiamo in considerazione un team di massimo 10 persone, definiamo qual è la dimensione massima di un progetto da affidare ad un team come questo e vediamo poi come RUP può gestire progetti di questa entità e anche più piccoli. Secondo COCOMO II[5] che utilizziamo come modello di stima, un team di 10 persone è solitamente responsabile di progetti la cui durata massima è di circa 15 mesi. Un progetto che dura più di 15 mesi richiederebbe normalmente un team con più di 10 persone. Dieci persone riuscirebbero a costruire sistemi più ampi soltanto con una pianificazione più lunga, ma solitamente la pianificazione è talmente importante da prevedere uno staff più consistente per garantire risultati soddisfacenti. Questo progetto è in grado di produrre approssimativamente da 40.000 a 50.000 righe di codice sorgente (sloc) (800-1000 punti di funzione in Java)[6] partendo da zero.

Secondo il modello, si tratta del progetto più ampio che ci si può realizzare con un team XP
in tempi brevi. Inoltre, vi sono forse altre ragioni per cui un team piccolo non riesce a costruire un ampio sistema in modo efficiente; ad esempio, la perdita di familiarità nel tempo con il codice già creato. (Un piccolo team dovrebbe fare in serie ciò che un grande team riesce a fare in parallelo. Così, arrivato il momento del test e del debug dell'integrazione, il team piccolo dovrà tornare sul codice scritto molto tempo prima).

Se prendiamo una serie di progetti di queste dimensioni, osserviamo una mappatura accettabile tra i rilasci XP e le iterazioni RUP. Gli esempi seguenti illustrano come pianificarli in RUP. I numeri sono soltanto indicativi, ma ragionevoli per progetti di queste dimensioni. Includono l'impegno e i tempi per la preparazione di tutti gli artefatti RUP richiesti che possono essere distribuiti oltre al codice, come le guide utenti, le informazioni di installazione e operazione e così via.

### Esempi di progetti XP

L'Esempio 1 illustra un progetto piccolissimo che consiste in 5.000 righe del nuovo codice Java e che richiede circa 12 persone/mese su un lasso di tempo di 7 mesi. [7]

[1] La durata di un'iterazione prevista in uno sviluppo e-business può essere più breve (da due a sei settimane).

[2] Le iterazioni della fase Inizio sono generalmente più brevi. Notare inoltre che il modello fase di RUP comprende un'ampia varietà di forme, cosicché in una lunga fase Transizione può essere contemplata una sequenza di iterazioni, ciascuna delle quali porta alla consegna del software al cliente, anche ad intervalli di due mesi, ma l'ingresso alla transizione implica che l'architettura sia completamente stabile, mentre le modifiche hanno lo scopo chiaro di adattare, perfezionare e correggere.

[3] Vedere [Beck00], secondo il quale non è possibile eseguire un progetto XP con 20 programmatori, mentre con 10 è "ampiamente fattibile".

[4] XP si focalizza soprattutto sulla consegna diretta del valore commerciale al cliente, in primo luogo come funzione. Dà poca importanza all'architettura e fà in modo che emerga durante una serie di refactoring del software, man mano che vengono aggiunte le funzioni.
Se l'architettura della soluzione non è ben stabile, si rischiano gravi rotture quando le funzioni aggiunte invalidano le precedenti ipotesi (e le soluzioni ad-hoc), creando problemi non riconducibili al refactoring locale.

[5] COCOMO II è una rilavorazione del modello classico COCOMO per la stima del costo software sviluppato in origine dal dr. Barry Boehm.
È calibrato per progetti piccoli come quelli da 2000 sloc. Consultare [Boehm00].

[6] I punti di funzione sono misurazioni della dimensione del software indipendenti dal codice sorgente, ottenute quantificando la funzionalità fornita all'utente , unicamente in base alla progettazione logica e alle specifiche funzionali. Questa definizione è stata presa dal sito Web del gruppo utenti internazionale del punto di funzione, all'indirizzo http://www.ifpug.org/.

[7] Anche se può sembrare lungo, osserviamo che copre l'intero ciclo di vita, da un punto di partenza, praticamente senza staff né requisiti (solo una minima idea) all'accettazione del progetto e alla liquidazione. Rappresenta anche un output del modello COCOMO II, che consente una notevole compressione della pianificazione, ma con l'inconveniente di maggiori costi e rischi. Tuttavia, secondo la pianificazione nella tabella, qualcosa di utile e funzionale dovrebbe essere disponibile più o meno tre mesi e mezzo dopo l'avvio del progetto (al termine della prima iterazione di Costruzione).

Esempio 1

| | Inizio | Elaborazione | Costruzione | Transizione |
|---|---|---|---|---|
| Staff | 2 | 1 | 1.5 | 2 |
| Durata in settimane | 3 | 6 | 18 | 3 |
| Numero di iterazioni (e durata in settimane di ciascuna) | 1 (3) | 1 (6) | 3 (6) | 1 (3) |

L'Esempio 2 riguarda un progetto piccolo di 10.000 righe del nuovo codice Java, che richiede circa 27 persone/mese per un periodo di 8 mesi.

Esempio 2

| | Inizio | Elaborazione | Costruzione | Transizione |
|---|---|---|---|---|
| Staff | 1.5 | 2.5 | 4 | 3 |
| Durata in settimane | 4 | 7 | 20 | 4 |
| Numero di iterazioni (e durata in settimane di ciascuna) | 1 (4) | 1 (7) | 3 (7) | 1 (4) |

L'Esempio 3 illustra un progetto di 40.000 righe del nuovo codice Java, che richiede circa 115 persone/mese per un periodo di 15 mesi.

Esempio 3

| | Inizio | Elaborazione | Costruzione | Transizione |
|---|---|---|---|---|
| Staff | 3 | 5 | 10 | 8 |
| Durata in settimane | 6 | 16 | 36 | 6 |
| Numero di iterazioni (e durata in settimane di ciascuna) | 1 (6) | 2 (8) | 4 (9) | 1 (6) |

Entro questi limiti (che ricoprono quasi l'intero processo di sviluppo), le iterazioni RUP si associano strettamente ai rilasci XP, sia in termini di scopo che di durata.

### Sviluppo di un ampio sistema non adatto a XP

In sviluppi molto ampi (che RUP sarà in grado di gestire, contrariamente a XP), le iterazioni RUP sono molto più lunghe. L'Esempio 4 mostra un progetto di 1.500.000 righe del nuovo codice Java, che richiede 4.600 persone/mese per un periodo di 45 mesi. [8]

Esempio 4

| | Inizio | Elaborazione | Costruzione | Transizione |
|---|---|---|---|---|
| Staff | 35 | 70 | 140 | 100 |
| Durata in settimane | 20 | 50 | 100 | 20 |
| Numero di iterazioni (e durata in settimane di ciascuna) | 2 (10) | 2 (25) | 3 (33) | 2 (10) |

[8] Si può obiettare che non si dovrebbe mai caratterizzare un progetto in tal modo o che un progetto di queste dimensioni deve essere suddiviso in molti progetti più piccoli (ciascuno dei quali sarebbe adatto a XP). Naturalmente, i progetti di queste dimensioni sono costruiti come sistemi o sottosistemi (mentre i progetti estremamente ampi come sistema di sistemi), ma quando i sistemi o i sottosistemi devono essere integrati in un singolo progetto, sarà da prendere in esame l'architettura e, in particolare, le interfacce tra i complessi software e i team che li producono. XP, nella sua forma attuale, non prende in considerazioni tali problematiche.

Chiaramente, in questo esempio il personale non è organizzato come un team singolo e monolitico (il progetto è composto da vari team, ciascuno dei quali lavora su sottosistemi che possono essere composti da sottosistemi, cosicché, ai livelli inferiori, sarà effettuato il planning su una scala simile a quella adoperata per progetti di dimensioni adatte a XP). Tuttavia, questo progetto deve rappresentare un sistema integrato affinché, al livello più alto, la pianificazione sarà richiesta per iterazioni di portata nettamente superiore a quelle adatte a XP (33 settimane). Tali iterazioni durano così a lungo a causa dell'inerzia relativa alla pianificazione di un progetto integrato di queste dimensioni. All'interno del piano di iterazione, RUP viene realizzato mediante piani build di integrazione (che possono esistere al livello dei sistemi e sottosistemi) realizzati su una scala simile a quella dei rilasci XP (per questo sistema molto grande) e, al livello più basso, su una scala simile a quella delle iterazioni XP (che sono di circa due settimane), specialmente nelle successive Elaborazione e Costruzione.

Così, tornando agli esempi dei sistemi più piccoli, le iterazioni di RUP sono più o meno equivalenti ai rilasci XP, e le build di RUP alle iterazioni XP.

### A cosa si associano le fasi di RUP in XP?

Almeno a prima vista, le fasi di RUP si associano ad aspetti del ciclo di XP non troppo ben delineati. XP sembra essere costruito, o almeno descritto, per essere adatto a una quantità stabile di rilasci (in armonia con un ciclo di business) contenenti iterazioni. Si riconosce che esiste una sorta di progetto da inizializzare (vedere "Scoping a Project" di [Beck01]) e che il grande piano costruito in questa fase sarà poi suddiviso in rilasci che, a loro volta, saranno divisi in iterazioni. [Beck01] sostiene che il grande piano ha il merito di aver deciso che non era poi così stupido investire nel progetto.

Così, esiste un punto in XP, prima del ciclo di rilasci e iterazioni, in cui è necessario decidere se un progetto è fattibile e quanto costerà; per fare questo, devono essere create alcune idee più precise sulla funzionalità richiesta. Si tratta di ciò che RUP chiama fase Inizio. In XP si dà l'impressione che ciò debba essere fatto velocemente, molto velocemente. In RUP diciamo che la durata dipende dalla prudenza rispetto ai rischi inerenti. Anche in XP leggiamo cose che sono state fatte a questo punto (da un numero ridotto di persone) come quelle che seguono:

- deduzione di alcuni requisiti (scrivere "grandi storie")
- pianificazione
- negoziazione con il cliente (definizione delle aspettative)
- factoring di eventuali vincoli di business

È facile osservare che ci vogliono pochi giorni per passare da un inizio frettoloso a una Visione condivisa (quelle che RUP chiama "grandi storie") e a uno scenario business (budget e ROI - Return On Investment - che giustificano il progetto), ma anche per avere un primo scorcio del Piano di sviluppo software.

In RUP, come in XP, vengono fatte numerose esplorazioni e pianificazioni all'avvio del progetto che consentono di stabilire quante probabilità di successo si hanno. Come mostrato in precedenza dall'Esempio 2, la pianificazione di RUP prevede un investimento (in termini di impegno) di circa 12.000 $ nella fase Inizio per giustificare spese del valore di circa 250.000 $. Ma dipende: nelle aree in cui i problemi sono tenuti sotto controllo, non occorreranno grandi sforzi e quindi l'inizio non sarà mai troppo precipitoso. In altre occasioni, ci si dovrà soffermare a lungo sulla fase iniziale in quanto il dominio è sconosciuto e richiede ricerche e approfondimenti.

Dopo la sua fase Inizio equivalente, XP va dritto alla pianificazione del primo rilascio. In RUP la fase Elaborazione segue l'Inizio, mentre nelle iterazioni di Elaborazione si stabilizza l'architettura della soluzione. In apparente contrasto, XP suggerisce che la pianificazione del rilascio sia function-oriented, affinché tutti i rilasci consegnino un valore commerciale al cliente. XP è abbastanza esplicito nel criticare ciò che chiama pianificazione delle infrastrutture: deve essere fatto lo stretto necessario per supportare la funzionalità scelta per quel tipo di rilascio. Se necessario, un'ulteriore infrastruttura viene poi aggiunta, nel caso in cui il sistema si sfaldi a causa dell'aggiunta di funzioni alle infrastrutture iniziali, poi viene eseguito il refactoring per far funzionare il codice.[9] L'idea è quella di non spendere troppo tempo a costruire qualcosa che non consegni alcun valore al cliente.

[9] Notare che il refactoring risulta comunque soltanto nei miglioramenti locali e non riguarda l'insieme del framework. Se tutto il framework cerca di risolvere il problema sbagliato, il refactoring non offrirà una soluzione corretta. L'unica soluzione è apportare modifiche radicali o, alle brutte, ricominciare. Ciò comporta ovviamente rischi legati alla pianificazione e di budget.

Per il confronto, è necessario affrontare due punti su RUP:

- Il concetto di architettura in RUP non si traduce semplicemente in infrastruttura

  Secondo RUP, "l'architettura di un sistema software (ad un certo punto) è l'organizzazione o la struttura dei componenti significativi del sistema, che interagisce mediante interfacce con componenti fatti di piccole parti e interfacce."

  Così, l'architettura riguarda l'intera soluzione (non solo l'infrastruttura) da una particolare serie di prospettive, ad un livello appropriato di astrazione.

- La nozione di RUP di architettura eseguibile consegna un valore commerciale al cliente.

  Consente inoltre la simulazione preliminare di una soluzione che soddisfi i requisiti non funzionali del cliente (come alcuni requisiti funzionali chiave del processo).

  In questo modo, la riduzione del rischio (e spesso i requisiti non funzionali sono quelli più rischiosi) è di per sé un valore commerciale per il cliente.

Perché riteniamo necessario descrivere il processo in questo modo? Perché RUP è in grado di allontanare i rischi, ma anche perché crediamo che vi sono classi di problemi e sistemi (solitamente quelli più ampi e complessi) che sono semplicemente non adatti a XP, il cui approccio consente all'architettura di emergere dal refactoring del codice. Un rischio è che con modifiche locali nel refactoring (l'ambito sarà quindi limitato), saranno prodotte ottimizzazioni locali che porteranno, nel complesso, a una soluzione inadatta e non ottimale. Questo non lo pensiamo solo noi: Kent Beck afferma la stessa cosa di [Beck00], nel capitolo intitolato Four Values, under the heading Courage. Purtroppo XP dice poco su come ideare modifiche strutturalmente significative (dice solo che ci vuole coraggio per applicarle!)

Poi è anche difficile fare il refactoring di alcune modifiche. Ad esempio, per inserire un sistema monoutente e monomacchina in sistemi multiutente e multiprocessore, può essere necessaria una nuova e lunga progettazione che magari permetterà di ottenere soltanto un sistema con molti vincoli.

L'enfasi di RUP sull'architettura interessa i casi in cui l'approccio iniziale è sbagliato e deve essere completamente riesaminato. Nei sistemi più piccoli, il rischio è inferiore (il refactoring può dare una ripulita al sistema e vi sono meno probabilità che l'architettura iniziale non sia corretta).

Notare anche che quando il rischio percepito è minimo (a causa di dimensione, nuova tecnologia, mancanza di familiarità, complessità, altre esigenze di prestazioni, affidabilità, sicurezza e così via) e la soluzione può essere creata in un framework esistente e ben conosciuto, la fase Elaborazione in RUP (in cui le problematiche strutturali vengono gestite) può essere abbastanza breve (e riguardare più da vicino la deduzione, il perfezionamento e la simulazione di requisiti funzionali importanti). In altre parole, il ciclo di vita di RUP si accorcia e assomiglia quasi a un ciclo di vita XP. Se XP è in grado di affrontare un problema con successo, potrà farlo anche lo scenario di sviluppo RUP in modo più "leggero" (anche se non uguale).

La fase Costruzione in RUP equivale alla serie di rilasci XP con la particolarità che in RUP devono essere consegnati i risultati di ogni iterazione di Costruzione al cliente per avere l'effetto prodotto da XP. XP non ha un vero e proprio equivalente della fase Transizione RUP, poiché ogni rilascio XP è già nelle mani del cliente. Al termine di un progetto XP troviamo ciò che [Beck00] chiama la morte del progetto (vedere le informazioni sotto l'intestazione Ciclo di vita di XP poco più giù), dove vengono realizzate alcune delle attività di liquidazione del progetto.

### A cosa si associano le fasi di XP in RUP?

Le fasi di XP (che si applicano sia ai rilasci sia alle iterazioni di XP) sono l'esplorazione, l'impegno e la guida. Al livello del rilascio, corrispondono a particolari gruppi di attività (chiamati in RUP dettagli di workflow) nella disciplina di gestione della progettazione RUP. Si tratta del piano per l'iterazione successiva (associato all'esplorazione e all'impegno) e del progetto di monitoraggio e controllo (associato alla guida). Le sezioni seguenti illustrano il concetto.

9 Nessuno negherebbe che il coraggio è una virtù - e, per antonomasia, una virtù di "gente per bene". Tuttavia, anche la persona più coraggiosa avrà bisogno, a volte di un metodo di lavoro ben organizzato e di un framework in cui si possono risolvere i problemi.

## Fasi XP

In un capitolo di [Beck00] sul ciclo di vita di un progetto XP ideale, i titoli del primo livello sono esplorazione, pianificazione, iterazioni al primo rilascio, produzione, manutenzione e morte . Queste sembrano essere le fasi di livello più elevato, ma non è proprio così. Un progetto XP sarà delimitato da una fase iniziale di esplorazione e dalla "morte"alla liquidazione del progetto. Ma l'elemento essenziale è il rilascio e ogni rilascio ha una fase di esplorazione. Quindi, la fase Esplorazione del progetto (che porta al primo rilascio) è un caso speciale in quanto si deve passare attraverso un ingresso (vedere Scoping a Project in [Beck01]) in cui viene presa la decisione se continuare o no. Sia i rilasci sia le iterazioni di XP hanno tre fasi (esplorazione, impegno, guida). La “Manutenzione” caratterizza la vera natura del progetto XP dopo il primo rilascio.

## Ciclo di vita XP

Grosso modo, il ciclo di vita XP relativo a un progetto con sette rilasci in 15 mesi assomiglia a quello descritto nella Figura 1.

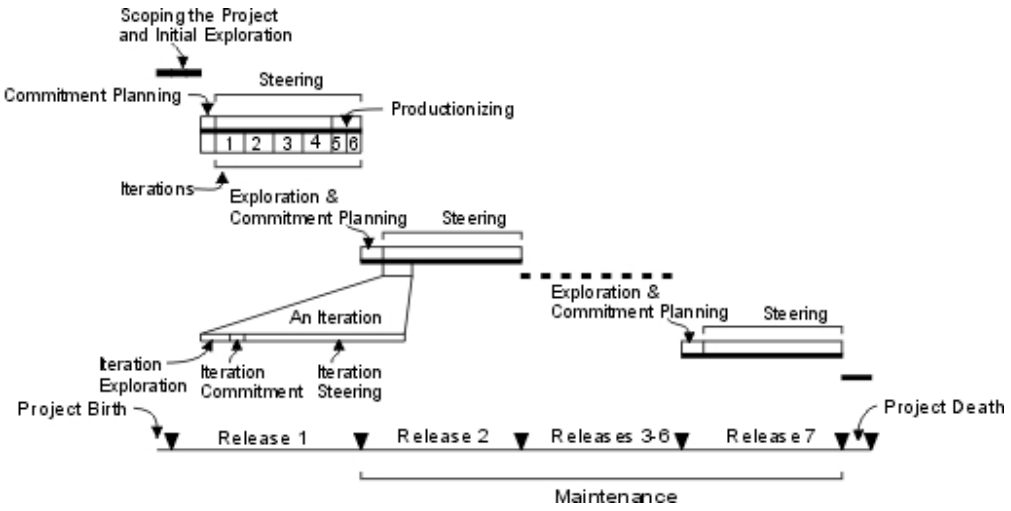


Figura 1

Ogni rilascio dopo il primo (di tre mesi) dura circa due mesi, mentre ogni iterazione circa due settimane, tranne nelle ultime fasi del rilascio (produzione - come viene chiamata in [Beck00]) quando l'andatura dell'iterazione accelera.

Che senso ha tutto ciò da un punto di vista di metrica della pianificazione? In questo esempio studiato, abbiamo cercato di seguire le linee guida suggerite da XP (secondo le quali i rilasci dovrebbero durare in media due mesi, di cui il primo tra i due e i sei mesi ), prendendo un team composto da meno di 10 persone. Se il progetto è come quello dell'Esempio 3 precedentemente illustrato, produrrà circa 40.000 righe di codice Java o 800 punti di funzione (con il fattore di conversione del modello COCOMO II) in totale. Se viene diviso in rilasci, ogni rilascio produrrà circa 115 punti di funzione.

La parte più difficile sarà il primo rilascio: la sfida consiste nello stabilire un punto di partenza (è stato designato un membro dello staff ma non sono ancora stati catturati i requisiti e nessun esame è stato realizzato) e nel consegnare infine al cliente un prodotto di ottima qualità. Anche se si punta ad un'alta produttività, diciamo 12 punti di funzione per staff al mese (basandoci sui dati industriali forniti dal David Consulting Group all'indirizzo http://davidconsultinggroup.com/indata.htm), non è possibile suddividere lo staff come arbitrariamente deciso; infatti, il problema dello spazio che stiamo affrontando qui (115 punti di funzione) è abbastanza piccolo da non poter essere suddiviso arbitrariamente in parti e affrontato da un grande team.

Tuttavia, se riteniamo fattibile questa ipotesi, occorrerà una squadra di circa quattro persone e il progetto terminerà il primo rilascio con un team di sette persone. Se uno o più membri dello staff vengono aggiunti al team per la durata del progetto, nel resto del progetto siamo nella modalità normale di XP (manutenzione) con un team di otto persone (che rientra nell'area di confort XP in termini di dimensioni). Man mano che cresce la dimensione diminuirà leggermente la produttività e l'intervallo di rilascio ridotto (due mesi) controbilancerà la crescita dello staff, in modo tale che ogni rilascio produrrà all'incirca lo stesso peso funzionale in più del primo. L'impegno globale del progetto sarà allora di 108 persone/mese - poco meno di quelle del precedente Esempio 3).

## Ciclo di vita predefinito di RUP

Al contrario, il ciclo di vita predefinito di un progetto RUP di analoghe dimensioni assomiglia a questo:

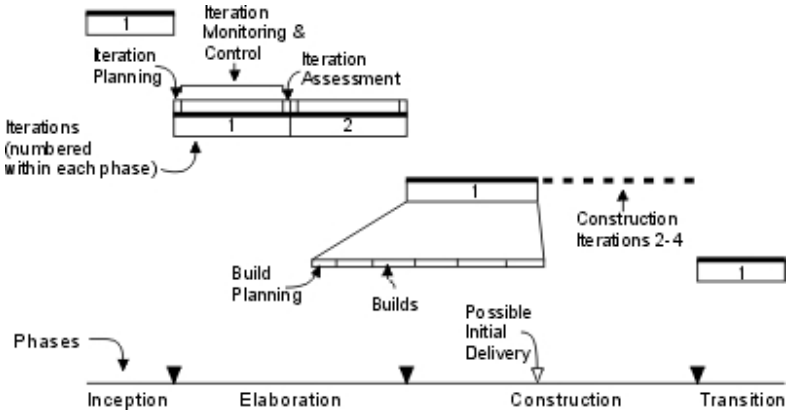


Figura 2

Nuovamente, questo piano si basa sull'Esempio 3. Le iterazioni RUP sono numerate in ogni fase. Utilizzando il ciclo di vita predefinito di RUP, molto probabilmente la principale opportunità di consegnare un prodotto di qualità al cliente sarebbe alla fine della prima iterazione di Costruzione, che dura sette mesi all'interno di un progetto. Tuttavia, in questo punto RUP può produrre circa un quarto, o 200 punti di funzione, della capacità totale (contro 115 punti di funzione XP dopo sette mesi). In genere, la prima consegna al cliente avviene al termine della Costruzione, che in questo caso durerebbe 13 mesi nel progetto, fase in cui tutta la capacità sarebbe presente, essenzialmente. Ciò non vuol dire che il cliente vede il prodotto per la prima volta; RUP lo invita infatti alle valutazioni di iterazione e ad osservare il prodotto in evoluzione durante il test.

Perché esiste questa differenza tra RUP e XP? Esiste perché, in questo caso predefinito, si presuppone che il rischio sia abbastanza notevole nell'architettura della soluzione (se ad esempio non ha precedenti, o la tecnologia è nuova o i requisiti non funzionali sono particolarmente onerosi) da richiedere due iterazioni nell'Elaborazione per renderlo stabile, oltre che per cercare di realizzare la funzionalità richiesta dal cliente. Questo vuol dire che solo i requisiti strutturalmente significativi saranno stati studiati a fondo entro la fine dell'Elaborazione.

XP funziona diversamente. Fà in modo di avere una serie di soluzioni complete e adeguate ad ogni rilascio, poiché presuppone che l'architettura non si sfalderà tra i vari rilasci e che il refactoring sarà comunque in grado di riparare ogni eventuale sfaldatura. Crediamo che ciò limiti in modo sensibile l'applicazione di XP a classi di sistema che possono essere costruite
su architetture esistenti di nota capacità.

## Ciclo di vita reale di RUP

In alcuni sistemi, il vero ciclo di vita di RUP sembra leggermente differente: la fase Elaborazione è probabilmente più corta e forse anche l'Inizio. Se scambiamo un'iterazione di Elaborazione con una di Costruzione, possiamo anticipare la fine della prima iterazione di costruzione di un paio di mesi, e farla quindi durare cinque mesi anziché sette.

Facendo questo si produrranno probabilmente meno dei 200 punti di funzione dell'esempio predefinito (poiché il numero di persone che lavoravano nell'iterazione di Costruzione precedente sarebbe ridotto), ma ci si avvicinerà al livello di agilità pianificato da XP (con un certo rallentamento al fine di ridurre notevolmente i rischi). In casi estremi, possiamo immaginare un approccio basato su RUP nella produzione iniziale di un sistema piccolo (diciamo 115 punti di funzione come nell'esempio di XP), seguito da svariati cicli completi di RUP (sequenze complete di Inizio, Elaborazione, Costruzione e Transizione) per poi produrre il resto del sistema in una modalità di manutenzione analoga a quella di XP.

Di nuovo, dopo circa sei mesi, la produzione iniziale può essere al limite di tempo estremo raccomandato per XP, poi però RUP ne prende atto e richiede di pianificare (assegnando quindi tempi e impegni) cose su cui XP sembra sorvolare, come la distribuzione dei rilasci.

Possiamo accennare altre restrizioni, a partire da queste osservazioni, sulla natura dei progetti adatti a XP. XP richiede esplicitamente una relazione molto intima tra il cliente e il team di sviluppo, in cui il cliente sia a tempo pieno sul sito per scrivere le storie e definire i rilasci. In XP, il cliente ha un suo ruolo nel team: la persona o il gruppo che svolge il ruolo può appartenere o meno ad una società richiedente, ma ha in ogni caso l'autorità di parlare per conto di chi commissiona il software. Essenzialmente, possiede i requisiti.

Questo tipo di relazione vale più spesso per i prodotti sviluppati all'interno della società che per quelli presi formalmente in appalto per clienti esterni. Il ciclo di un rilascio di due mesi può anche essere più accettabile in simili circostanze. Ogni due mesi, la distribuzione di un nuovo rilascio (con il rilevamento di nuove funzioni e difetti) all'interno della società può comportare spese logistiche eccessive.

Con questi presupposti (piccolo team, architettura già definita, sviluppo della produzione interna alla società con meno formalità) i cicli di vita RUP appositamente personalizzati avrebbero un effetto simile a quello dello sviluppo XP. La durata del rilascio iniziale può essere leggermente più lunga dell'optimum di XP, ma avrebbe meno rischi.

Nella direzione opposta (grande team, architettura senza precedenti, sviluppo appaltato in modo formale, con vincoli inerenti alla produzione e alla distribuzione) è difficile osservare come XP possa essere scalabile (e ad essere sinceri, le fonti di questo documento non credono ci riesca). RUP, d'altro canto, è progettato per affrontare le formalità che caratterizzano simili progetti. Forse un approccio simile a quello di XP (che utilizza cioè tecniche come pair programming e refactoring) può essere incorporato all'interno di un progetto RUP più ampio, ma soltanto una volta che l'architettura è stabile.

## Artefatti

RUP descrive più di 100 artefatti, è ciò ha sollevato obiezioni circa le spese burocratiche, non tollerate per i piccoli progetti. Questo punto di vista è sbagliato per quattro ragioni:

- I progetti non devono produrre tutti gli artefatti: la selezione e la personalizzazione di artefatti è una parte necessaria del processo. RUP fornisce linee guida su cosa può essere omesso e cosa personalizzato.
- Un artefatto (in RUP, la descrizione di un artefatto) è un'entità di specifica (precisa cioè le informazioni che un'attività deve produrre e, per farlo sistematicamente, può descrivere un artefatto in forma astratta). La caratterizzazione in RUP di artefatti come modelli, elementi di modello o documenti (circa la metà degli artefatti di RUP sono classificati come documenti) non ha lo scopo di suggerire una particolare realizzazione.

  Un modello UML può essere catturato e presentato utilizzando un tool basato sullo scopo (come Rational Rose) o altrimenti con diagrammi realizzati mediante un semplice tool di grafica o ancora con uno schizzo su una lavagna bianca - che rappresenterebbe comunque un artefatto in termini di RUP (sebbene si corra il rischio di perdere i dati con una lavagna bianca!).

  Il termine “documento” ha connotazioni storiche che portano ancora le persone a concepire un documento come discreto prodotto di lavoro su carta con una particolare (e richiesta) serie di titoli di paragrafi - che dovranno essere seguiti dal testo (e forse da diagrammi) per completare il documento. Ma questa è solo una realizzazione di un documento RUP.

  I documenti RUP sono gruppi di informazioni composti da testo e diagrammi. Ai fini dell'utilità, RUP precisa come devono essere tali informazioni e, per farlo in modo conveniente e sistematico, indica un modello che enumera e descrive le voci e le problematiche da considerare. È certamente possibile utilizzare tali modelli in modo diretto e formale per realizzare gli artefatti sotto forma di documenti (in senso generale), siano essi cartacei o elettronici. Riconoscendo che molti progetti intendono fare ciò, abbiamo scelto una serie di modelli da poter utilizzare in questo senso, più o meno direttamente. Ma non è essenziale farlo (non tutto ciò che viene presentato in un modello è necessariamente rilevante in un dato progetto) e RUP dà al progetto la libertà di scegliere una realizzazione (e il meccanismo di produzione) appropriata ad un artefatto. La cosa importante sono le informazioni che contiene.

- Abbiamo ritenuto essenziale identificare in modo chiaro tutti i potenziali prodotti di lavoro del processo affinché possano essere considerati dal progettista di processo e dal responsabile di progetto; in tal modo le decisioni relative alla loro omissione o personalizzazione saranno prese a ragion veduta e con la piena comprensione delle conseguenze in caso di omissione. Crediamo che se l'elenco degli artefatti viene reso completo ed esplicito si possa ottenere una certa oggettività nella configurazione del processo. Si può in questo modo aiutare il responsabile di progetto con meno esperienza invitandolo a prestare attenzione a cose come queste che un responsabile esperto capirebbe immediatamente anche se fossero sottintese, ma che un novizio non coglierebbe al primo impatto. Il fatto di avere una serie ben definita di documenti aiuta anche il cliente e il responsabile del progetto a raggiungere rapidamente un accordo su ciò che deve essere consegnato e in quale forma, per evitare le sgradevoli incomprensioni che spesso caratterizzano la liquidazione del progetto.
- RUP descrive alcuni artefatti composti e prosegue descrivendone i componenti come artefatti. Ciò favorisce una descrizione più dettagliata degli input e output dell'attività, se necessaria. RUP può interrompere la descrizione di un elemento composito, come il modello Progettazione, per soffermarsi sulle classi intese come parti del modello Progettazione e non come artefatti a pieno titolo. L'identificazione di queste classi come artefatti consente una descrizione precisa del relativo utilizzo e si conforma al tempo stesso con il metamodello del processo pur incrementando il numero totale di artefatti.

XP sembra immune dalla critica relativa al numero eccessivo di artefatti, questa idea è troppo semplicistica per le due ragioni seguenti:

- XP è già stato adattato a un particolare tipo di sviluppo, con un sottoinsieme di discipline RUP a un certo livello, quindi ci si aspetta che richieda pochi artefatti.
- XP enfatizza l'importanza delle storie dell'utente e del codice, ma gli altri prodotti di lavoro sono menzionati senza troppa enfasi durante la descrizione del processo, pertanto il numero di artefatti non è così insignificante come sembra a prima vista.

Considerando l'approccio di XP, non sorprende che nei tre libri utilizzati come riferimento per questo paragone (poiché sono il fiore all'occhiello del processo), i termini “artefatto” e “prodotto di lavoro” non compaiano nell'indice. Tuttavia, non è difficile leggere nel testo e cogliere riferimenti che sono in realtà artefatti. Eccone alcuni esempi:

- Storie
- Vincoli
- Attività
- Attività tecniche
- Test di accettazione
- Software-codice
- Rilasci
- Metafore
- La progettazione - bozze CRC, UML
- Documenti di progettazione - prodotti alla fine del progetto
- Standard di codifica
- Test delle unità
- Workspace (sviluppo e altre infrastrutture)
- Piano di rilascio
- Piano di iterazione
- Report e osservazioni sui meeting
- Piano complessivo - budget
- Relazione di avanzamento
- Stime riferite alle storie
- Stime delle attività
- Difetti (e dati associati)
- Documentazione supplementare dalla conversazione
- Documentazione di supporto
- Dati di test
- Tool del framework di test
- Tool per la gestione del codice
- Risultati di test
- Spike (soluzioni)
- Registrazioni del tempo di lavoro per attività
- Dati di metrica
  - Risorse
  - Ambito
  - Qualità
  - Tempo
- Altre metriche
- Tracciamento dei risultati

Poi vi sono circa 30 artefatti, alcuni dei quali compositi. L'elenco è soltanto indicativo, non esaustivo. I libri su XP non si soffermano molto sull'argomento e, in ragione del loro livello, non sarebbe giusto aspettarselo. Tuttavia, se un progetto deve realizzare degli artefatti, occorreranno ulteriori dettagli sui contenuti e le forme.

Il progetto può senz'altro descriverli immediatamente, ma ciò porta via del tempo prezioso al lavoro reale; in compenso, RUP fornisce in anticipo una guida che consente di risparmiare il tempo da dedicare al lavoro.

### Perché XP non ha bisogno di tutti gli artefatti RUP?

Una ragione è che l'ambito di XP non è lo stesso di RUP. Ciò è alquanto intenzionale: XP riguarda la programmazione in base alle necessità del business. Come nascono queste esigenze di business (e come sono modellate, catturate e interpretate) non rientra nell'ambito di XP.
Il membro del team XP responsabile del cliente presenta una distillazione dei requisiti sotto forma di storie al team di sviluppo XP, che ne stabilisce il valore commerciale. Come tali storie vengano poi espresse (o siano esprimibili) sotto questa forma non riguarda XP. Così, per esempio, ciò che RUP descrive nella sua disciplina di modellazione del business non rientra nell'ambito di XP (la modellazione del business in RUP comprende circa 14 artefatti). XP descrive un processo con regolari rilasci al cliente. La logistica della distribuzione di questi rilasci non rientra fra le problematiche di sviluppo, quindi la disciplina di distribuzione di RUP è ampiamente fuori dall'ambito di XP (la distribuzione in RUP comprende circa 9 artefatti). Pertanto, in un piccolo progetto adatto a XP, è concepibile omettere circa 23 artefatti quando si effettua la personalizzazione di RUP.

Un'altra ragione è che in XP la cattura dei requisiti e della progettazione può essere abbastanza semplice: i requisiti vengono catturati come storie degli utenti (che a volte dovrebbero essere suddivise), poi scomposte in attività (le quali rappresentano, in sostanza, la progettazione, tenendo a mente una metafora sul sistema). È possibile fare questo in tutti i sistemi? Decisamente no. È possibile farlo in alcuni sistemi? Certo, ma in tutta franchezza XP afferma che non lo si può fare in tutti i sistemi o, quantomeno, non lo dichiara apertamente.

Il comportamento auspicato di molti sistemi più ampi e complessi può essere difficile da articolare senza un certo approccio sistematico come quello dei casi d'uso. Non sarà neanche possibile fare pieno affidamento sul dialogo tra il cliente e lo sviluppatore per elaborare in modo consistente storie complesse degli utenti, poiché la memoria umana è spesso difettosa. Quindi, il miglioramento di strutture con un comportamento complesso nei grandi sistemi viene agevolato dall'abilità di creare e sviluppare varie viste astratte dell'architettura del sistema in questione. Gli artefatti che RUP descrive nei Requisiti (circa 14 artefatti) e nelle discipline di Analisi & Progettazione (circa 17) consentono a RUP di affrontare la varietà, la dimensione e la complessità di tali sistemi.

Nella roadmap dei progetti piccoli di RUP, il numero degli artefatti (in Requisiti e Analisi & Progettazione) si riduce a sette, anche grazie al riferimento all'artefatto composto. Non è un trucco: gli artefatti vengono descritti alla stessa maniera di XP. Ad esempio, riferendosi al modo in cui sarà probabilmente realizzato in un piccolo progetto, RUP dice questo del modello Progettazione:

> "Ci si attende che il modello Progettazione evolva in una serie di sessioni di brainstorming nelle quali gli sviluppatori utilizzeranno schede
> CRC e diagrammi ricavati a mano per esplorare e catturare la progettazione. Il modello Progettazione sarà mantenuto soltanto finché gli
> sviluppatori lo riterranno utile. Non sarà mantenuto coerente con l'implementazione, ma registrato per eventuali riferimenti."

Infine, RUP consente una maggiore formalità nella gestione del progetto, se necessaria, ma anche negli accordi relativi al contratto. Di nuovo, molti artefatti di gestione del progetto di RUP (ve ne sono 15 nella disciplina di gestione del progetto) fanno parte di artefatti composti e devono semplicemente essere realizzati come documenti separati quando la formalità del progetto lo richiede. Per esempio, in RUP il Piano di sviluppo software "contiene" artefatti come il Piano di gestione rischi e il Piano di accettazione prodotto. In progetti più piccoli e meno formali, sarebbe possibile affrontare le problematiche delineate in questi piani semplicemente con uno o due paragrafi nel Piano di sviluppo software. Nella roadmap dei progetti piccoli di RUP, il numero di
artefatti sulla gestione del progetto viene ridotto a sei.

### Confronto degli artefatti di un piccolo progetto

Quindi, quando si adatta RUP a un piccolo progetto, e si personalizzano anche i requisiti dell'artefatto, cosa succede nel complesso? Se si osserva la roadmap di RUP e si contano gli artefatti, se ne avranno in totale 30 (26, se vengono esclusi alcuni della distribuzione), quindi dopo tutto non ce ne sono poi troppi! RUP delinea semplicemente in maniera chiara ciò che XP accenna appena, consente di decidere cos'è realmente necessario e fornisce una guida su come effettuare la selezione. Ora, la granularità varia secondo le parti, ma il punto è che in RUP il numero di artefatti per piccoli progetti, come quelli che XP affronterebbe senza problemi, è più o meno uguale a quello degli artefatti di XP.

Mappatura orientativa da artefatti XP ad artefatti RUP

| XP | Roadmap dei progetti piccoli di RUP |
|---|---|
| Storie<br>Documentazione supplementare conversazione | Visione<br>Glossario<br>Modello del caso d'uso |
| Vincoli | Specifiche supplementari |
| Test di accettazione<br>Test delle unità<br>Dati di test<br>Risultati di test | Modello di test |
| Software-codice | Modello di implementazione |
| Rilasci | Prodotto (Product)<br>Note sul rilascio |
| Metafora | Documento dell'architettura software |
| Progettazione - bozze CRC, UML<br>Attività<br>Attività tecniche<br>Documenti di progettazione - prodotti alla fine del progetto<br>Supporto della documentazione | Modello di progettazione |
| Standard di codifica | Linee guida per la progettazione<br>Linee guida per la programmazione |
| Workspace (sviluppo e altre infrastrutture)<br>Tool del framework di test | Tool |
| Piano di rilascio<br>Stime riferite alle storie<br>Stime delle attività<br>Piano di iterazione | Piano di sviluppo software<br>Piano di iterazione |
| Piano complessivo - budget | Scenario business<br>Elenco dei rischi<br>Piano di accettazione del prodotto |
| Relazione di avanzamento<br>Registrazioni del tempo di lavoro per attività<br>Dati di metrica: risorse, ambito, qualità, tempo<br>Altre metriche<br>Tracciamento dei risultati<br>Relazioni e note delle riunioni | Valutazione stato |
| Difetti e dati correlati | Richieste di modifica |
| Tool per la gestione del codice | Piano di gestione configurazione<br>Repository del progetto<br>Workspace |
| Spike | Prototipi |
| | Caso di sviluppo<br>Modelli specifici per progetto |

### Linee guida

Oltre agli artefatti, RUP fornisce delle linee guida, ossia ulteriori informazioni sull'artefatto (significato, rappresentazione, relazioni con altri artefatti, contenuto ed utilizzo). Tali linee guida, se stampate, occuperebbero centinaia di pagine, ma non bisogna spaventarsi; si possono infatti leggere esclusivamente le informazioni sugli artefatti desiderati.

I libri XP contengono inoltre molte informazioni su pratiche e artefatti, senza cercare (o avere bisogno) di modellare le relazioni rigorosamente. La letteratura complessiva su XP non è poi così poca, comunque. I tre libri disponibili al momento sono di circa 600 pagine, e altri due di circa 700 pagine saranno presto sul mercato. Anche il libro sul refactoring [Fowler99] è consistente, più di 400 pagine.

Oltre a ciò, svariati siti Web si interessano a XP. Le informazioni non hanno tendenza a sovrapporsi, poiché XP viene esaminato da diversi punti di vista che si aggiungono semplicemente alle esperienze di base. Di fatto, ciò fa emergere un'altra differenza tra RUP e XP: RUP è un prodotto, XP non lo è. Non esiste un'unica fonte relativa a XP, anche se è utile iniziare dai libri. Il modo più rapido per “imparare” XP è seguire i corsi disponibili in commercio e dispensati da coloro che lo hanno ideato.[11]

[10] RUP è a volte definito “proprietà”, ma nessuno può comprarlo e utilizzare le idee contenute in esso o aggiungerne altre, né tanto meno cancellare parti che non riguardano da vicino un’organizzazione o un progetto e via dicendo; basta solo rispettare il copyright e le relative licenze. Anche la descrizione di XP sotto forma di libro è proprietà intellettuale degli autori e delle case editrici; l'unica differenza è il grado di comprensione a partire dalle fonti. RUP, come prodotto, cerca di essere completo, mentre i libri di XP necessitano dei supplementi (esercitazioni costanti o consulenze).

## Attività

RUP definisce formalmente il termine “attività” come lavoro svolto da un ruolo, utilizzando e trasformando artefatti di input e producendo artefatti di output nuovi e modificati. RUP prosegue poi enumerando queste attività e suddividendole in categorie secondo la “disciplina” o la principale “area di attenzione” all'interno del progetto (come definito da RUP). Tali discipline sono:

- Modellazione del business
- Requisiti
- Analisi & Progettazione
- Implementazione
- Test
- Distribuzione
- Configurazione & Gestione delle modifiche
- Gestione del progetto
- Ambiente

Le attività sono correlate al tempo attraverso gli artefatti che producono e consumano: un'attività può logicamente iniziare non appena i relativi input sono disponibili (e hanno raggiunto la maturità desiderata). Questo per dire che le coppie di attività produttore-consumatore possono sovrapporsi nel tempo, se lo stato dell'artefatto lo consente; non devono essere messe rigidamente in sequenza.

Le attività in RUP sono volte a rendere meno opaco il processo intellettuale di produzione di un artefatto (per dare informazioni importanti su come si deve produrre qualcosa). Le attività possono anche essere utilizzate per facilitare la pianificazione da parte del responsabile di progetto. Le "pratiche ottimali" si intrecciano in RUP, come descritto nel ciclo di vita, negli artefatti e nelle attività: i principi di progettazione del software hanno dimostrato di produrre software di qualità costruiti in base a pianificazione e budget prevedibili.

RUP, attraverso le attività e gli artefatti associati, supporta e realizza queste pratiche ottimali, temi ricorrenti in RUP. Notare che anche XP utilizza la nozione di “pratiche”, ma come vedremo questo concetto non combacia esattamente con quello di pratiche ottimali in RUP.

XP (vedere [Beck00], Capitolo 9) presenta una visione semplice e coinvolgente dello sviluppo software in quanto ha quattro attività basilari da applicare e strutturare secondo alcune pratiche di supporto:

- Codifica
- Test
- Ascolto
- Progettazione

In realtà, le attività XP si avvicinano, nell'ambito, alle discipline di RUP piuttosto che alle attività RUP e molte cose che accadranno in un progetto XP (oltre a codifica, test, ascolto e progettazione) saranno il frutto dell'elaborazione e dell'applicazione di tali pratiche.

### Esiste un equivalente XP delle attività di RUP?

Sì, esiste, ma le “attività” di XP non vengono formalmente identificate o descritte: ad esempio, nel Capitolo 4, User Stories di [Jeffries01], si leggerà il titolo “Define requirements with stories, written on cards”, mentre lungo tutto il capitolo sono contenute informazioni e descrizioni di processo sulle storie dell'utente e su come (e da chi) devono essere prodotte. Poi il libro prosegue, descrivendo XP nei suoi aspetti salienti (che in [Jeffries01] sono misti, alcuni focalizzati sull'artefatto, altri sull'attività); “cose fatte” e “cose prodotte” sono descritte con vari livelli di regolamentazione e precisione.

La regolamentazione apparente di RUP deriva dalla completezza e dalla maggiore formalità che lo caratterizzano nel trattamento sistematico di attività, inclusi input e output. In XP la regolamentazione non manca ma forse, nel tentativo di mantenersi “leggero”, omette semplicemente formalità e precisione. I dettagli presenti in RUP dovranno essere aggiunti quando si implementa XP in un progetto. La mancanza di specificità in XP non è una forza o una debolezza, ma non si deve confondere con la semplicità. Gli addetti al progetto devono sapere cosa fare e hanno bisogno in quel momento di informazioni specifiche.

## Discipline e workflow

RUP ha recentemente sostituito il termine “disciplina” con “workflow di base”. Una disciplina, in RUP, è un'insieme di attività (e di concetti correlati) che producono una particolare serie di artefatti, i quali costituiscono aspetti e problematiche importanti nell'ambito dello sviluppo software. Questa consuetudine si adatta abbastanza bene alla definizione del vocabolario di disciplina come ramo della conoscenza o dell'insegnamento.

Come osservato in precedenza, le discipline di RUP sono Modellazione del business, Requisiti, Analisi & Progettazione, Implementazione, Test, Distribuzione, Configurazione & Gestione delle modifiche, Gestione del progetto e Ambiente. Non coprono tutti gli aspetti relativi a cosa un'organizzazione o un'impresa devono fare quando assumono personale per sviluppare, distribuire, operare, supportare, vendere, immettere nel mercato o quantomeno esaminare sistemi ampiamente software. In generale, RUP non affronta la progettazione di sistemi, ad esempio. Non descrive nemmeno tutti i requisiti di alcuni standard internazionali di processo software come ISO 15504 (che traccia aspetti inerenti all'acquisizione di software e alla gestione di risorse umane). Ma la scelta è volontaria: questi ulteriori aspetti, sebbene importanti, sono al di fuori dell'interesse di progettazione di RUP.

XP è anche più ristretto: include quattro attività basilari (codifica, test, ascolto e progettazione, precedentemente considerate più pertinenti alle discipline di RUP), svolte mediante una serie di pratiche che richiedono l'esecuzione di altre attività; tali attività sono associate ad altre discipline di RUP. Le pratiche di XP, classificate alla lettera da [Beck00], sono:

- “Planning Game— determinare rapidamente l'ambito del rilascio successivo combinando priorità di business e stime tecniche. Quando la realtà impatta il piano, aggiornare il piano.
- Small Releases— iniziare rapidamente la produzione di un sistema semplice, quindi rilasciare nuove versioni su un ciclo molto breve.
- Metaphor— guidare ogni sviluppo con una semplice storia condivisa su come funziona l'intero sistema.
- Simple Design— il sistema deve essere progettato nella maniera più semplice possibile in un qualunque momento. La complessità in eccesso viene rimossa non appena individuata.
- Testing— i programmatori scrivono continuamente test di unità da eseguire in modo impeccabile ai fini dello sviluppo. I clienti scrivono test per dimostrare che le funzioni sono state ultimate.
- Refactoring— i programmatori ristrutturano il sistema senza modificarne il comportamento per eliminare ripetizioni, incrementare la comunicazione, semplificare le cose e renderle più flessibili.
- Pair programming— tutto il codice di produzione viene scritto da una coppia di programmatori su una sola macchina.
- Collective ownership— chiunque può modificare un codice in una qualsiasi parte del sistema e in ogni momento.
- Continuous integration— integrare e costruire il sistema più volte al giorno, ogni qual volta viene completata un'attività.
- 40-hour week— lavorare non più di 40 ore a settimana come regola. Mai fare straordinari per due settimane di seguito.
- On-site customer— inserire nel team un utente reale, disponibile a tempo pieno per rispondere alle domande.
- Coding standard— i programmatori scrivono tutto il codice secondo le regole, enfatizzando la comunicazione mediante il codice.”

Un particolare interessante nell'argomento Simple Design è che l'espressione “complessità extra” è in qualche modo soggettiva e difficile da definire, vista da un occhio di osservatore esperto.

Le attività svolte in seguito alla pratica Planning Game, ad esempio, saranno principalmente associate alla disciplina di gestione del progetto in RUP. Tuttavia, alcuni argomenti non rientrano nell'ambito di XP e, fra questi, la modellazione del business. La deduzione di requisiti è molto al di fuori dell'ambito di XP (il cliente, presente sul sito assieme al team, definisce e descrive i requisiti sotto forma di storie). La distribuzione del software rilasciato non viene descritta da XP. A causa del livello e dei tipi di sviluppo che tratta, XP può affrontare in modo molto leggero le problematiche relative alle discipline di Configurazione e Gestione delle modifiche, che RUP invece spiega nei dettagli.

## Utilizzo delle pratiche XP in RUP

Nelle discipline in cui XP e RUP si sovrappongono, alcune delle pratiche descritte in XP possono essere impiegate in RUP (e alcune lo sono già), per esempio:

- Pair programming: XP richiede che il codice di produzione sia creato dai programmatori che lavorano in coppia in una singola workstation, affermando che tale pratica ha effetti benefici sulla qualità del codice e, una volta acquisite le capacità, è anche gradevole. RUP non descrive la dinamica di produzione del codice in questo livello approfondito ed è certamente possibile utilizzare il pair programming in un processo basato su RUP. Alcune informazioni su questa pratica e sul test (test-first design e refactoring - vedere sotto) vengono ora fornite da RUP sotto forma di white paper. Ovviamente, non è obbligatorio utilizzare questa pratica in RUP e non bisogna neanche credere che sia necessario imporla.

  Le prove dei vantaggi su scala industriale sono scarse e aneddotiche; gli studi di ([Nosek98] e [Williams00]) hanno confrontato coppie con singoli individui (che lavorano isolati), ma i veri team non lavorano in questo modo. All'interno di un team, con la cultura della comunicazione aperta, in cui gli individui si sentono liberi di fare domande ai colleghi (ma anche ai responsabili) e lavorano ad un processo ben definito, si può ipotizzare che i vantaggi del pair programming (in termini di effetti sui costi dell'intero ciclo di vita) siano più difficili da individuare. In un buon team, le singole persone si riuniranno per discutere e risolvere problemi insieme senza essere obbligate a farlo.

  [Beck00] ha un'opinione abbastanza negativa sulle persone e sul processo quando dice: "Un'altra funzione caratteristica del pair programming è che alcune pratiche non funzionerebbero senza di esso. Sotto effetto dello stress, le persone regrediscono. Saltano i test di scrittura, rimandano il refactoring …". Si suppone che queste pratiche naturali portino a un risultato migliore - perché non applicarle dunque?

  La suddivisione in istanze di un buon processo non è inopportuna, mentre invece lo è l'idea di considerarla obbligatoria a un "micro" livello. Tuttavia, in alcune circostanze, il lavoro di coppia è ovviamente vantaggioso, poiché ci si può aiutare a vicenda, ad esempio:

  - nei primi giorni di addestramento dei team, quando i collaboratori cominciano a conoscersi
  - in team che non conoscono a fondo qualche nuova tecnologia
  - in team composti da esperti e da novizi

- Test-first design e refactoring: sono ottime tecniche da applicare nella disciplina Implementazione di RUP. Test-first design, in particolare, è un ottimo modo di chiarire i risultati a un livello dettagliato.
- On-site customer: molte attività di RUP traggono enormi vantaggi, in termini di efficienza, dal fatto di avere un cliente sul sito come membro del team. Con il cliente presente sul sito, l'esigenza di uno scambio di documenti è sempre minore.

  XP è restio nell'affermare che devono essere catturate altre cose oltre al codice e privilegia la conversazione come miglior mezzo di comunicazione. Ma tutto dipende dalla continuità e la familiarità che ne derivano. Quando un sistema subisce modifiche, bisognerà produrre altre cose, anche per sistemi piccoli.

  XP ammette questa pratica, ma sotto forma di aggiunta, come quando inserisce i documenti di progettazione alla fine del progetto. Infatti, XP non vieta la creazione di documenti o di altri artefatti (si mantiene cauto sull'argomento), ma dice che si dovrebbero produrre soltanto quelli realmente necessari, che verranno poi utilizzati. RUP è d'accordo su questo, ma va oltre, descrivendo ed elencando ciò di cui si ha bisogno quando la qualità e la familiarità non sono ottimali.

- Coding standard: RUP ha un artefatto (Linee guida per la programmazione) che può essere quasi sempre considerato obbligatorio (come lo è per molti profili dei rischi del progetto, da cui dipende gran parte della personalizzazione).
- Continuous integration: RUP supporta questa pratica mediante build a livello di sistema e di sottosistema (all'interno di un'iterazione); i componenti di unità controllati sono integrati e testati nel contesto del sistema emergente.

### Pratiche XP non scalabili

Alcune pratiche, tuttavia, non risultano scalabili (né XP afferma che lo siano) e il fatto di utilizzarle è soggetto a questa condizione in RUP; ad esempio:

- Collective ownership: è utile che tutti i membri di un team ristretto siano responsabili di un piccolo sistema o sottosistema di un sistema più grande e che abbiano familiarità con la totalità del codice. Attribuire a tutti i membri lo stesso potere di effettuare modifiche ovunque dipende dalla natura e dalla complessità del sistema o sottosistema in questione. Spesso sarà più rapido (e sicuro) avere un singolo addetto (o una coppia) per segmento di codice.
  Anche la familiarità con i codici scritti nel modo migliore diminuisce rapidamente col passare del tempo, specie quando si tratta di codici con algoritmi complessi.
- Refactoring:in un ampio sistema, un refactoring frequente non può sostituire la mancanza di architettura. [Beck00] afferma: "La strategia di progettazione XP somiglia a un algoritmo per "scalare una montagna". Si fa una semplice progettazione, poi si rende un po' più complessa, poi un po' più semplice, quindi un po' più complessa. Il problema degli algoritmi a livelli consiste nel raggiungimento di condizioni ottimali, in cui un piccolo cambiamento non è in grado di migliorare la situazione, mentre un grande cambiamento sì."
  In RUP, l'architettura fornisce l'idea del "raggiungimento della vetta", ossia la possibilità di rendere malleabili sistemi ampi e complessi.
- Metaphor: in sistemi ampi e complessi, ridurre l'architettura a una metafora non basta. RUP offre un framework descrittivo più ricco dell'architettura, che non riduce semplicemente (come fa [Beck00], con scarsa considerazione) a "grandi caselle e connessioni".
- Rapid releases: la frequenza con la quale un cliente può accettare e distribuire nuovi rilasci dipende da molti fattori, tra cui la dimensione del sistema, spesso correlata all'impatto commerciale. A
  Un ciclo di due mesi sembra troppo breve per alcune classi di sistema: la logistica della produzione può persino proibirlo.

Alcune pratiche, che a prima vista sembrano direttamente utilizzabili in RUP, devono invece essere ulteriormente elaborate e applicate con cauzione:

- Simple design: XP si basa molto sulla funzionalità: le storie dell'utente sono selezionate, suddivise in attività e successivamente implementate. Secondo [Beck00], la "giusta progettazione del software, in qualsiasi momento, è quella che:

1. esegue tutti i test

2. non ha una logica riprodotta…

3. descrive ogni proposito importante per i programmatori

4. ha il minor numero possibile di classi e metodi."

L'idea di XP è di non aggiungere niente che non sia strettamente indispensabile al momento. Esiste un problema, per certi versi simile a quello dell'ottimizzazione locale, nel descrivere una classe di requisiti detti "non funzionali" in RUP. Tali requisiti apportano un valore commerciale al cliente, ma sono molto difficili da descrivere come storie (alcune delle quali, chiamate "vincoli" da XP, rientrano in questa categoria).

RUP non esige che la progettazione vada oltre le necessità, ma che sia basata su un modello strutturale ben preciso rispondente ai requisiti non funzionali.

Quindi, RUP concorda con XP: la "semplice progettazione" deve svolgere soltanto test in grado di garantire che il software soddisferà i requisiti non funzionali. Di nuovo, ciò porta soltanto a una problematica più ampia, che si verifica quando aumentano la dimensione e la complessità del sistema, quando l'architettura non ha precedenti o i requisiti non funzionali sono onerosi. Ad esempio, quando bisogna disporre i dati per ordine (se si vuole lavorare in un ambiente distribuito eterogeneo) sembra che il codice diventi troppo complesso, ma è comunque una necessità.

- 40 ore settimanali: come XP, RUP suggerisce che non bisogna fare straordinari sistematicamente. XP non suggerisce un limite di 40 ore ed è abbastanza flessibile sul discorso dell'orario di lavoro. I progettisti software sono famosi per fare molti straordinari senza essere pagati, avendo però la soddisfazione di vedere qualcosa ultimato, e il responsabile non vede la necessità di far loro interrompere il lavoro.

  Ciò che il responsabile non deve fare è sfruttare questo fatto o imporre straordinari (ma raccogliere piuttosto dati sulle ore effettive di lavoro, anche se non pagate). Se l'annotazione delle ore lavorate presenta numeri elevati per un periodo troppo lungo, dovrebbero essere svolte delle indagini. Ma tali problematiche devono essere risolte, dal responsabile e dalla persona interessata, nelle circostanze particolari in cui si verificano, tenendo anche conto delle eventuali osservazioni fatte dal resto del team. Quaranta ore è solo un'indicazione (anche se rigida).

## Ruoli

In RUP, le attività sono svolte da alcune figure.[12] Queste figure hanno anche la responsabilità di artefatti specifici (il responsabile crea l'artefatto e assicura che i cambiamenti apportati da altri ruoli - se consentiti - non lo compromettano). Un ruolo in RUP può essere svolto da un individuo o da un gruppo di persone. Allo stesso modo, un individuo o un gruppo possono svolgere vari ruoli. Un ruolo non deve essere necessariamente associato a una singola posizione o “slot” in un'organizzazione; la mappatura dei ruoli alle unità organizzative può anche essere "molti a molti".

### Ruoli di RUP

RUP definisce in totale 30 ruoli: non esiste una mappatura esatta alle discipline poiché un ruolo (ad esempio, un architetto software) non corrisponde necessariamente a una sola disciplina, ma solo approssimativamente (situa quindi il ruolo in corrispondenza della principale disciplina che svolge):

- Modellazione del business: tre ruoli
- Requisiti: cinque ruoli
- Analisi & Progettazione: sei ruoli
- Implementazione : tre ruoli
- Test: due ruoli
- Distribuzione: quattro ruoli
- Configurazione & Gestione delle modifiche: due ruoli
- Gestione del progetto: due ruoli
- Ambiente: tre ruoli

### Perché i ruoli in RUP sono così tanti?

I ruoli in RUP vengono utilizzati come attività di suddivisione, sono poi utili a delineare capacità e competenze necessarie a specifiche attività e a selezionare lo staff che dovrà svolgerle. Il livello di suddivisione facilita anche l'identificazione di nuove posizioni organizzative quando cambia l'importanza di un ruolo. Ad esempio, in un progetto piccolo e più informale, i ruoli RUP di Responsabile di progetto e Responsabile di configurazione possono essere svolti dalla stessa persona; in un progetto formale e più ampio dell'aeronautica militare, il compito del Responsabile di configurazione sarà così specifico e oneroso da poter interessare un piccolo team. Descrivendo i ruoli in tal modo, RUP facilita la mappatura.

### Ruoli XP

[Beck00] identifica sette ruoli applicabili ad XP, poi prosegue descrivendo le relative responsabilità, le capacità e le caratteristiche richieste a coloro che li svolgeranno. Questi ruoli sono:

- Programmatore
- Cliente
- Tester
- Tracker
- Coach
- Consulente
- Big Boss

Riferimenti a tali ruoli sono presenti in altri libri XP, con descrizioni elaborate della attività svolte.

[12] Per essere più precisi (e specifici), le attività vengono eseguite da individui o team che svolgono i ruoli.

La differenza nel numero di ruoli XP e RUP è facile da spiegare:

- XP non copre tutte le discipline di RUP.
- I ruoli XP sono in realtà più vicini alle posizioni che ai ruoli di RUP; ad esempio, il programmatore di XP esegue molteplici ruoli di RUP (implementatore, integratore e revisore di codice) che richiedono competenze leggermente differenti.

Quando i ruoli di RUP riguardano un piccolo progetto (come nella roadmap dei progetti piccoli di RUP), il numero di ruoli simili a quelli XP, ossia le posizioni a cui sono abbinati, è di molto inferiore a 30. Nella roadmap di un piccolo progetto, le posizioni sono 5, come mostra la seguente tabella.

I ruoli di RUP associati a un piccolo progetto per ABC Company

| Titolo di lavoro di ABC Company | Ruolo RUP |
|---|---|
| Responsabile di progetto | Responsabile di progetto<br>Ingegnere di processo<br>Responsabile di distribuzione<br>Revisore dei requisiti<br>Revisore dell'architettura |
| Esecutivo di ABC Company | Revisore del progetto<br>Stakeholder<br>Revisore dei requisiti |
| Capo programmatore | L'analista di sistema<br>Lo specificatore di requisiti<br>Progettista dell'interfaccia utente<br>Architetto di software<br>Revisore della progettazione<br>Ingegnere di processo<br>Specialista del tool<br>Responsabile di configurazione<br>Responsabile del controllo della modifica<br>e, in misura minore, gli stessi ruoli del Programmatore |
| Programmatore | Progettista<br>Implementatore<br>Revisore del codice<br>Integratore<br>Progettista test<br>Tester |
| Assistente amministrativo | Responsabile di:<br>- manutenzione del sito Web del “piccolo progetto”<br>- assistenza al Responsabile del progetto nelle attività di pianificazione o di progettazione<br>- assistenza al Responsabile del controllo della modifica nel controllodelle modifiche agli artefatti<br>- eventuale collaborazione con altre figure, se necessaria |

## Conclusioni

XP e RUP non sono, nei fatti, la stessa cosa. RUP è un framework di processo da cui possono essere configurati particolari processi, poi suddivisi in istanze. RUP deve essere configurato; questa tappa è essenziale, come afferma lo stesso RUP. Nello specifico, si dovrebbe paragonare una versione personalizzata di RUP con una di XP, personalizzando RUP in base alle caratteristiche esplicitamente (o implicitamente) stabilite da XP. Un RUP così personalizzato può accogliere alcune pratiche XP (come pair programming, test-first design e refactoring), ma non sarebbe comunque identico ad XP in quanto dà molta importanza al ruolo di architettura, astrazione (nella modellazione) e rischio e presenta una struttura differente nel tempo (fasi, iterazioni).

RUP consentirà di costruire processi in grado di ospitare progetti esterni all'ambito di XP nel tipo e nella scala. RUP è più pesante solo perché è una descrizione completa di una famiglia di processi leggeri o pesanti (in termini di artefatti, componenti distribuibili, formalità, prescrizione, formalità o qualsiasi altra misura di “peso”) nell'implementazione, a seconda dei casi. XP è certamente leggero in quanto è stato espressamente ideato per implementare un processo (leggero), ma anche perché le descrizioni di XP (almeno nei libri) non sono poi così elaborate. Quindi, in un'implementazione di XP, occorrerà scoprire cose, inventarle o definirle immediatamente. Se paragonato a RUP, XP è senz'altro più leggero nel materiale descrittivo.

Le cose comunque cambieranno, forse già a partire dalla pubblicazione di altri due libri, uno dei quali, [Succi01], è di circa 512 pagine. Tuttavia, nella situazione attuale, il profilo dell'impegno di adozione è diverso nei due approcci. RUP punta molto sull'impegno preliminare, sia nei requisiti di addestramento sia nel processo di personalizzazione. Un organizzazione sarà portata a personalizzare RUP in base a un'applicazione adatta alle sue dimensioni per progetti di tipo e dimensione specifici, utilizzando poi i risultati in vari progetti. Con XP sarà richiesto un addestramento anticipato, ma il resto dell'impegno di adozione sarà concentrato sul progetto, che si espande e cattura tutte le cose indispensabili al funzionamento di XP. XP non spinge ovviamente a salvaguardare la “memoria aziendale”, rendendo così l'organizzazione che lo adotta (se questa non memorizza la sua esperienza di processo) vulnerabile alla rotazione del personale.

Definire RUP pesante e XP leggero senza ulteriori dettagli è doppiamente nocivo, in quanto ne oscura sia la natura sia la funzione. E quando lo si fa in modo peggiorativo, è persino irrilevante. Saranno le implementazioni di questi processi ad essere “leggere” o “pesanti”, in funzione della leggerezza o della pesantezza richiesta dalle circostanze.

XP non è una disciplina libera in cui tutto è permesso (si focalizza piuttosto su un aspetto specifico dello sviluppo software e sul valore che apporta, inoltre dà indicazioni chiare e rigorose su come raggiungere l'obiettivo prefissato).

La copertura di RUP è più ampia e profonda e ne spiega dunque la "dimensione" apparente. Tuttavia, al micro livello del processo, RUP offre occasionalmente alternative altrettanto valide, cosa che XP non fà, ad esempio il pair programming. Con ciò non intendiamo criticare XP, ma ci limitiamo ad illustrare come, anche in base al suo nome, abbia ristretto il proprio orizzonte.

## Riferimenti

[Beck00] Extreme Programming Explained, Kent Beck, Addison-Wesley, 2000

[Beck01] Planning Extreme Programming, Kent Beck, Martin Fowler, Addison-Wesley, 2001 [Boehm00] Software

Cost Estimation with COCOMO II, Barry W. Boehm et al, Prentice Hall PTR, 2000 [Fowler99] Refactoring:

Improving the Design of Existing Code, Martin Fowler et al, Addison-Wesley, 1999

[Jeffries01] Extreme Programming Installed, Ron Jeffries, Ann Anderson, Chet Hendrickson, Addison-Wesley, 2001

[Kruchten00] The Rational Unified Process, An Introduction, Second Edition, Philippe Kruchten, Addison-Wesley, 2000

[Martin01] Extreme Programming in Practice, Robert C. Martin, James W. Newkirk, Addison-Wesley, 2001 (non ancora pubblicato)

[Nosek98] The Case for Collaborative Programming, John T. Nosek, Comm. ACM, Vol. 41, No. 3, 1998, pp. 105 108

[Succi01] Extreme Programming Examined, Giancarlo Succi, Michele Marchesi, Addison-Wesley, 2001 (non ancora pubblicato)

[Williams00] Strengthening the Case for Pair Programming, Laurie Williams, Robert R. Kessler, Ward Cunningham, Ron Jeffries, IEEE Software, Vol. 17, No. 4, 2000, pp. 19-25

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