Varianti di sistema

HÕkan Dyrhage

White paper del software Rational

TP 155

Rational®
the software development company

Indice

Introduzione

Questo documento descrive le varianti di sistema e come gestirle. Non occorre leggerlo per comprendere Rational Unified Process (al contrario, rappresenta soltanto un approfondimento di RUP). L'ultima sezione illustra brevemente come Rational Unified Process viene influenzato dall'aggiunta di varianti e di variabilità.

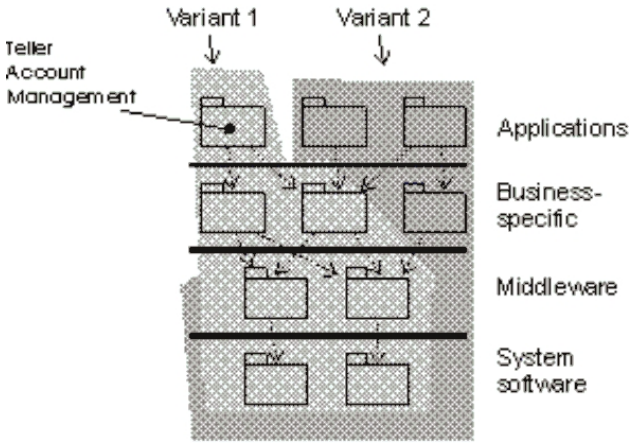
Si tratta di un settore in cui Rational Unified Process avrà modo di svilupparsi in futuro. Questo documento ne offre una breve panoramica.

Varianti del sistema

Molti sistemi sono prodotti con più di una variante. In altre parole, il sistema viene configurato, imballato ed installato in modi diversi a seconda dei clienti (e delle classi di clienti). A volte, si realizzano varianti diverse semplicemente installando e adattando il sistema in un altro modo. Altre volte la variabilità si ottiene consegnando alcune parti del sistema a clienti diversi. Le seguenti sezioni contengono alcuni esempi di varianti.

Varie parti del sistema

Varie parti del sistema completo sono consegnate a classi di clienti diverse. Ad esempio, un sistema bancario viene consegnato come due prodotti distinti. Diciamo che la variante 1 del sistema contiene tutto ciò che riguarda la telefonia, mentre la variante 2 ciò che attiene alla gestione dei conti agli sportelli. I programmi eseguibili sono definiti nei sottosistemi a livello delle applicazioni. Più precisamente, una variante (variante 1 nella figura sotto) è una build per il sottosistema TAM (Teller Account Management) e tutti i sottosistemi che deve compilare ed eseguire, ossia quelli che riguarda direttamente o indirettamente.



Un sistema bancario sviluppato con due varianti.

Varie lingue

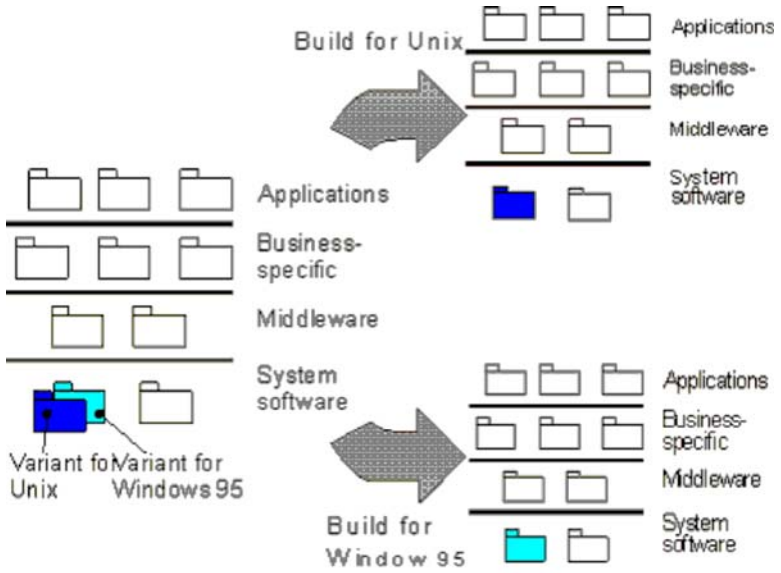
Se il sistema viene prodotto in varie lingue, ad esempio in inglese, francese e giapponese, si dovrà produrre una variante del sistema per ogni lingua. La differenza tra le varianti è che tutto il testo, nei menu e nelle guide, deve essere in una lingua specifica.

Un modo per gestire il problema delle lingue è raccogliere tutti i testi in un file e avere un file per ogni lingua. Produrre un sistema per ogni lingua significa produrre tutto, incluso il file che contiene il testo con la lingua specifica. Questo file è poi letto dal software al momento dell'avvio e vengono inizializzate tutte le principali variabili.

Piattaforme multiple

Se il sistema supporta piattaforme multiple incompatibili, è necessaria una variante del sistema per ogni piattaforma. Ad esempio, se un sistema esegue sia Windows NT sia UNIX, vengono prodotte due varianti del sistema.

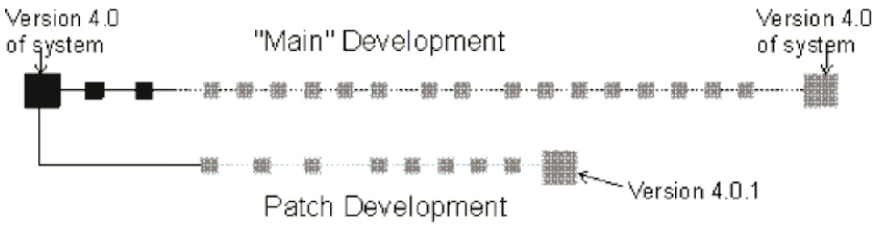
Nell'esempio che segue, il codice specifico alla piattaforma è situato in un sottosistema. In questo caso, sono sviluppate due varianti del sottosistema. Un file di compilazione, “makefile”, specifica quali versioni di ogni file di codice sorgente devono essere compilate assieme. È anche possibile affermare che un makefile specifica quali varianti di ogni sottosistema faranno parte della build. Quindi, è necessario un makefile per ogni piattaforma.



Un sistema viene sviluppato per essere eseguito su varie piattaforme.

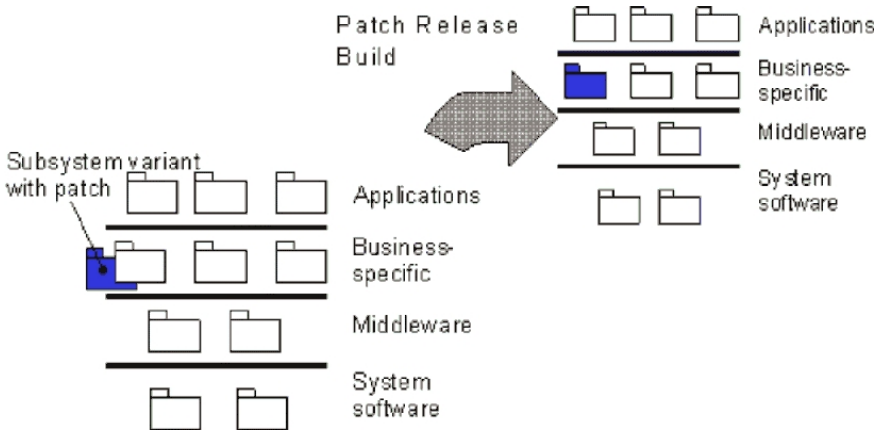
Rilasci provvisori

A volte è necessario sviluppare un rilascio patch del sistema. Lo si fa di solito parallelamente allo sviluppo di un progetto, ossia il rilascio patch è un'altra variante del sistema ed esiste contemporaneamente alla versione "principale" del sistema.



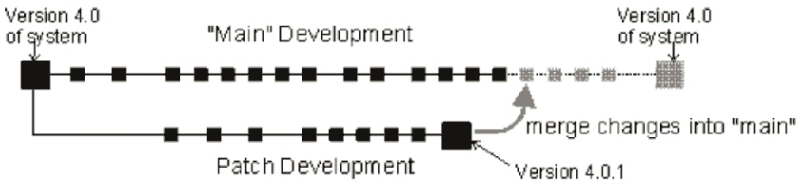
Un rilascio patch viene sviluppato parallelamente al principale progetto di sviluppo. I riquadri grigi indicano futuri rilasci e linee di base.

Le modifiche da apportare sono situate in uno o più sottosistemi di implementazione. Poiché un normale progetto di sviluppo procede in parallelo, occorre sviluppare varianti di questi sottosistemi nell'ambito dello stesso impegno di sviluppo. Per creare una build, si specifica in un makefile quali varianti di ciascun sottosistema devono far parte della build.



Una build di rilascio patch viene creata con una variante del sottosistema che contiene il patch.

Questo tipo di variante ha solitamente una vita breve. Una volta rilasciato il patch, la variante non viene ulteriormente sviluppata e tutte le modifiche di valore del codice rientrano nel percorso principale di sviluppo.

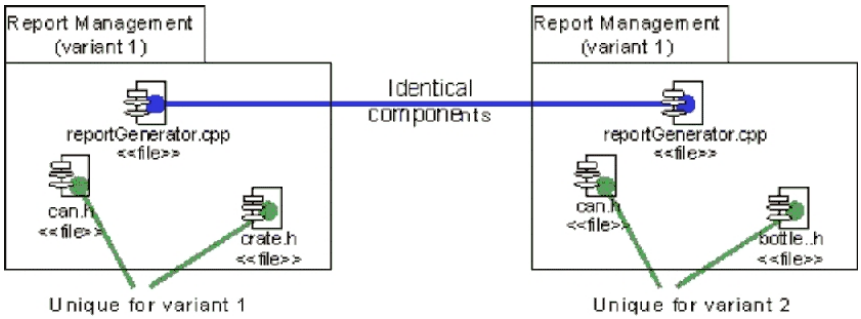


I cambiamenti rilevanti nel rilascio patch sono riuniti nel principale percorso di sviluppo quando

occorre.

Varianti di sottosistemi

Gli esempi precedenti illustrano che servono spesso una o più varianti di uno specifico sottosistema. Queste varianti avranno alcuni componenti in comune, mentre altri componenti sono specifici ad ogni variante di sottosistema.



Due varianti di un sottosistema di implementazione chiamato Gestione del report in cui un componente deve essere lo stesso in entrambi le varianti. Gli altri componenti sono specifici ad ogni variante.

Sovente, ogni variante di un sottosistema viene sviluppata da un implementatore e le varianti del sottosistema sono sviluppate in parallelo. Se il componente viene modificato da un implementatore che sviluppa una variante, la modifica deve essere estesa alle altre varianti. Per fare ciò, è necessario il supporto di un tool di gestione della configurazione e controllo della versione (CMVC) in grado di gestire i componenti che saranno identici in entrambe le varianti.

Meccanismi di variabilità

Esistono diversi meccanismi per creare varianti di sistema. Alcuni sono stati menzionati nelle sezioni precedenti. Ogni meccanismo ha caratteristiche diverse. Normalmente, viene utilizzata una combinazione di questi meccanismi per creare variabilità nel sistema.

- File di compilazione e link. Specificare in un file di compilazione (un makefile in un ambiente Unix) quali varianti di ogni file di codice sorgente devono essere compilate e unite all'interno di programmi eseguibili.
- Componenti caricati in modo dinamico. Sviluppare parti del sistema come librerie dinamiche di link, applet o componenti Active X che possono essere collegati nel programma in esecuzione al runtime. Questi componenti vengono gestiti da un tool CMVC, che rende possibile la consegna di un sottoinsieme di tali componenti al cliente.
- File di avvio. Utilizzare file contenenti informazioni che il software legge quando il sistema viene avviato per inizializzare il sistema. Ad esempio, file di risorsa in Windows, file di avvio o di inizializzazione, che sono letti dal software all'avvio del sistema e che configurano il sistema in modo diverso. Utilizzarli, ad esempio, se il sistema deve essere personalizzato in diverse lingue, nel qual caso si inseriscono tutti i testi in file di testo che vengono letti all'avvio del sistema.
- Suddividere il sistema in programmi eseguibili. Sviluppare il sistema come una serie di programmi eseguibili, ad esempio file .exe. Varie combinazioni possono essere consegnate al cliente. Le combinazioni di programmi eseguibili sono gestite da un tool CMVC.

Effetto su Rational Unified Process

Questa sezione descrive brevemente come Rational Unified Process viene influenzato dall'aggiunta di varianti.

Effetto sul workflow di implementazione

Il workflow di implementazione deve essere esteso alle seguenti aree:

- Aggiungere le seguenti fasi all'attività Definizione della Vista Implementazione:
- Come decidere quali varianti del sistema sviluppare
- Come stabilire quali sottosistemi devono avere varianti
- Come definire il meccanismo di variabilità da utilizzare per ottenere una variabilità
- Nell'attività Pianificazione di integrazione del sistema, devono essere considerate delle variabili e si deve pianificare come integrare varianti di sistema diverse.
- Occorre descrivere nei dettagli lo sviluppo parallelo "tra" varianti di un sottosistema. Ad esempio, cosa succede se un componente “identico” viene staccato? Come devono essere uniti i componenti? Questo discorso riguarda varie attività dell'implementatore.

Altre attività di implementazione possono essere ugualmente interessate.

Effetto su altri workflow

Lo sviluppo di varianti, o di famiglie di sistemi, influenza tutti i workflow. Nelle precedenti sezioni sono stati affrontati alcuni effetti del workflow di implementazione. Spieghiamo ora in breve come verranno influenzati gli altri workflow:

- Cattura dei requisiti— descrive come identificare le varianti del sistema. Cosa vuole ciascuna classe di clienti.
- Analisi & Progettazione— illustra come modellare varianti nel modello di progettazione, come progettare varianti di sottosistema e come definirle.
- Test— spiega come testare una famiglia di sistemi (varianti). Testare ogni variante del sistema come un sistema a parte. Le varianti incidono anche sul test di integrazione.
- Gestione— occorre organizzare il progetto e si può anche stabilire un team per ogni variante di sottosistema. In un ampio progetto, può anche esserci un responsabile di progetto per ogni variante di sistema.
- Ambiente— occorrono tool che facilitino la gestione delle varianti di sistemi.
- Distribuzione— è molto più complessa in quanto esistono varie classi di clienti a cui consegnare il prodotto finale.

Rational
the software development company

Sedi principali:

Software Rational
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