Un sistema di sviluppo aziendale

Ing. Davide Bolcioni

Amministratore di Sistema
3D Informatica Srl

Scalare il processo di sviluppo

- Scalare in quantità
  - Aumentano gli sviluppatori
    - Rif. “The Mythical Man-Month”. F. Brooks.
  - Aumentano gli artefatti sviluppati
    - Ircocervi e dinosauri
- Scalare in qualità
  - Strumenti migliori
  - Problemi più complessi
- Scalare in latenza

Cosa si chiede allo sviluppo

- Rapidità di consegna
  - Tempi di modifica ridotti
  - Tempi di realizzazione ridotti
- Efficacia della soluzione
  - Usabilità
  - Performance
- Validità dell'investimento
  - Drenaggio post-vendita

Lo scenario di partenza

- Ogni sviluppatore è un isola
  - Duplicazioni e incompatibilità
- Dipendenza da IDE
  - Non visibilità delle fasi di *build* e *deploy*
  - Debug invece di design
- Il mondo sulla scrivania
  - Da me funziona
  - Cambia la configurazione
  - Sistema uguale e sistema dedicato

La strategia

1. Introdurre strumenti appropriati
   - CVS, Eclipse, unit testing
   - Autotools, ant, pacchettizzazione
   - Regressione, sorveglianza, autoriparazione
2. Acquisire le pratiche che ne scaturiscono
   - Versioni e rami, design patterns, refactoring
   - Standard, interfacce, moduli e dipendenze
3. Iterare
   - Con la competenza ci si giova degli strumenti sofisticati
   - Eclipse e gli eterni principianti

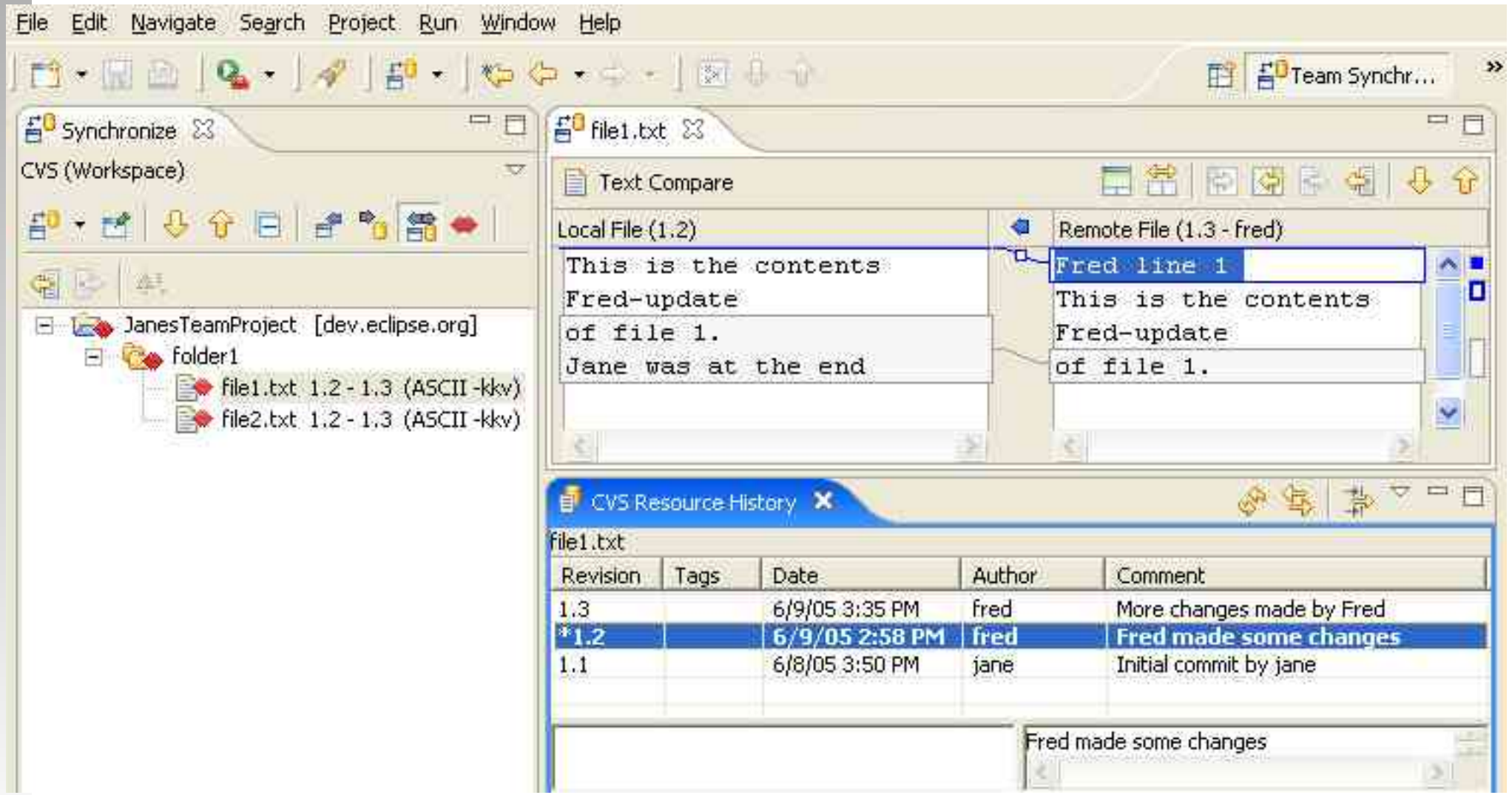
Strumenti - CVS

- Soluzione immediata a
  - Lavoro di gruppo
  - Backup
- Consente
  - Versioni
  - Rami
- Motivazioni specifiche
  - Solido, diffuso e documentato
  - Ampia dotazione di strumenti accessori
  - Nota su RCS, Subversion e realtà distribuite

Strumenti – Eclipse (1)

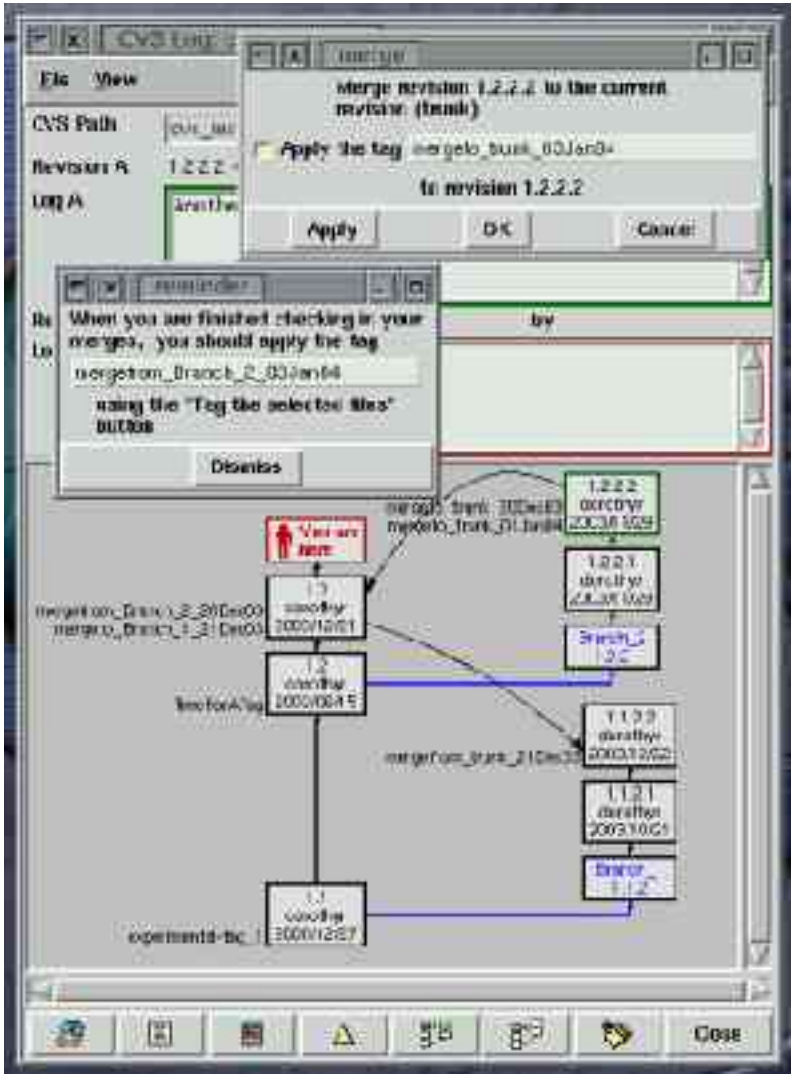
- Soluzione immediata per
  - Accesso a CVS
  - Rudimentale sviluppo Java (creazione .jar)
- Consente
  - Utilizzo di ant
  - Utilizzo di makefile
- Motivazioni specifiche
  - Offre funzionalità che serviranno dopo
  - Architettura suscettibile di ampia evoluzione
  - Aggrega e smista strumenti specializzati

Strumenti – Eclipse (2)



Strumenti – TkCVS

- Soluzione immediata per
  - Accesso a CVS
- Consente
  - Fusione assistita dei rami
  - Riparazioni
- Motivazioni specifiche
  - Visualizza rami
  - Tool invece che IDE
  - Semantica di `cvs(1)`



Strumenti – cvsweb/ViewCVS

- Soluzione immediata per
  - Consultazione CVS da piattaforme legacy
  - Esplorazione rami (CvsGraph)
- Consentono
  - Scaricamento tarball di una versione
- Motivazioni specifiche
  - Cvsweb Perl
  - ViewCVS Python

Pratiche – Versioni e Rami

- Marcatura delle versioni
  - Questa funziona
  - Questa l'ho consegnata
- Creazione di rami
  - Cambiamenti destabilizzanti
    - Sincronizzazione periodica
    - Stabilizzazione sul ramo
  - Release
    - Release candidate prima della release
    - Patch *minimali* dopo la release
    - Riportare sul tronco

Strumenti – autotools

- Soluzione immediata per
  - Configurazione della fase di *build*
  - Portabilità del codice
- Consentono
  - Riproducibilità della fase di *build*
  - Pacchettizzazione
- Motivazioni specifiche
  - Test della situazione effettiva
    - Librerie GNU su piattaforma Solaris
  - Metadati e strumenti, non linguaggi
  - Invocabili e pilotabili

Strumenti – ant

- Soluzione immediata per
  - Parametrizzazione della fase di *build* Java
- Consente
  - Riproducibilità della fase di *build*
  - Pacchettizzazione
- Motivazioni specifiche
  - Invocabile e pilotabile

Pratiche – Standard e interfacce

- Gli autotools facilitano
  - Con `AC_LIBOBJ`, ad esempio `AC_FUNC_MKTIME`
- Gli autotools costringono
  - Senza `configure`, niente `make`
- Gli autotools esplicitano
  - Le interfacce sono fissate
  - Le implementazioni cambiano
  - I programmi funzionano
- Per Java il bisogno è minore

Scenario intermedio

- Basta sviluppatori isolati
  - Con CVS si coordina il lavoro sul codice
  - Si esamina l'esistente invece di riscriverlo
- Minore dipendenza da IDE
  - La fase di *build* è esplicita
  - Dal CVS al CD con certezza
    - Non basta: possibili modifiche presso il cliente
- Un mondo oltre la scrivania
  - Funziona anche su un sistema diverso
  - Dipendenze: si comincia a vedere cosa ci vuole

Strumenti – Pacchettizzazione

- Soluzione immediata per
  - Installazione, aggiornamento e rimozione
- Consentono
  - Soluzioni per composizione
  - Listino
  - Traccia dell'installato
  - Verifica di deviazioni rispetto all'installato
- Motivazioni specifiche
  - I pacchetti funzionano, i setup no
  - Non devastano il sistema

Pratiche – Moduli e dipendenze

- Installazione per moduli
  - Si traccia cosa è installato
  - Si installa il necessario
- Esplicitazione delle dipendenze
  - Si rimpiazza un modulo e funziona comunque
  - Si aggiorna il sistema e funziona comunque
- **Riduzione del drenaggio**
  - Debug “a forcella”
    - Seguendo i rami in CVS
  - Quel che non c'è non si può rompere
  - Si scoprono deviazioni sull'installato

Strumenti – Documentazione

- Strumenti specializzati
  - JavaDoc per Java
  - Doxygen per C++
  - Manca per lo sviluppo Web
- Non basta documentare il codice
- Da produrre in fase di *build*

Strumenti – Unit Testing

- Strumenti specializzati
  - JUnit, HttpUnit, JwebUnit per Java/Web
  - CppUnit per C++
- Soluzione immediata per
  - Specifiche vaghe e generiche
    - Moduli privi di specifica
- Consente
  - Sostituzione dell'implementazione
  - Test di integrazione e regressione
- Motivazioni specifiche
  - Previsti in Eclipse

Pratiche - Refactoring

- Possiamo riorganizzare il codice
  - Lavorando su un ramo separato
  - Superando gli unit test
  - Costruendo un eseguibile da provare
- Possiamo progettare le soluzioni
  - Usando i design patterns
  - Usando UML
- **Efficacia della soluzione**
  - Posso produrre codice performante
  - Posso produrre una soluzione usabile
  - In breve, posso intervenire nel codice

Considerazioni finali

- Non è indolore come sembra
  - Ogni strumento ha i suoi bachi
  - Ogni pratica ha le sue deviazioni
- Funziona molto bene
  - Potenza della modularità
  - I moduli sono gestibili e comprensibili, le IDE meno
- Da solo non basta
  - I programmi li scrivono gli sviluppatori
- Prima vittima dei brevetti
  - Costo degli strumenti di sviluppo e Open Source