

BiblioTech - Personal Digital Library

Albana Gaba Alessandro Pegoraro Mirco Bocedi
Fabio Giuseppe Strozzi

Gruppo 8

Obiettivo

Creare un software efficiente per la catalogazione di documenti digitali in categorie personalizzabili dall'utente.

Funzionalità principali

- i documenti possono essere arricchiti da meta-dati.
- si possono creare delle relazioni navigabili tra i documenti.
- si possono effettuare delle ricerche sul catalogo o sul web.
- i doppioni vengono riconosciuti.
- supporto per qualunque tipo di file.
- portabile: funziona sia su OSs Windows che Linux.
- scalabile: ottime prestazioni anche con un elevato numero di documenti.
- le funzionalità possono essere estese e/o automatizzate tramite l'uso di **plugin**.

Funzionalità principali

- i documenti possono essere arricchiti da **meta-dati**.
- si possono creare delle relazioni navigabili tra i documenti.
- si possono effettuare delle ricerche sul catalogo o sul web.
- i doppioni vengono riconosciuti.
- supporto per qualunque tipo di file.
- portabile: funziona sia su OSs Windows che Linux.
- scalabile: ottime prestazioni anche con un elevato numero di documenti.
- le funzionalità possono essere estese e/o automatizzate tramite l'uso di **plugin**.

Funzionalità principali

- i documenti possono essere arricchiti da meta-dati.
- si possono creare delle **relazioni navigabili** tra i documenti.
- si possono effettuare delle ricerche sul catalogo o sul web.
- i doppioni vengono riconosciuti.
- supporto per qualunque tipo di file.
- portabile: funziona sia su OSs Windows che Linux.
- scalabile: ottime prestazioni anche con un elevato numero di documenti.
- le funzionalità possono essere estese e/o automatizzate tramite l'uso di **plugin**.

Funzionalità principali

- i documenti possono essere arricchiti da meta-dati.
- si possono creare delle relazioni navigabili tra i documenti.
- si possono effettuare delle **ricerche** sul catalogo o sul web.
- i doppioni vengono riconosciuti.
- supporto per qualunque tipo di file.
- portabile: funziona sia su OSs Windows che Linux.
- scalabile: ottime prestazioni anche con un elevato numero di documenti.
- le funzionalità possono essere estese e/o automatizzate tramite l'uso di **plugin**.

Funzionalità principali

- i documenti possono essere arricchiti da meta-dati.
- si possono creare delle relazioni navigabili tra i documenti.
- si possono effettuare delle ricerche sul catalogo o sul web.
- i **doppioni** vengono riconosciuti.
- supporto per qualunque tipo di file.
- portabile: funziona sia su OSs Windows che Linux.
- scalabile: ottime prestazioni anche con un elevato numero di documenti.
- le funzionalità possono essere estese e/o automatizzate tramite l'uso di **plugin**.

Funzionalità principali

- i documenti possono essere arricchiti da meta-dati.
- si possono creare delle relazioni navigabili tra i documenti.
- si possono effettuare delle ricerche sul catalogo o sul web.
- i doppioni vengono riconosciuti.
- supporto per **qualsunque tipo di file**.
- portabile: funziona sia su OSs Windows che Linux.
- scalabile: ottime prestazioni anche con un elevato numero di documenti.
- le funzionalità possono essere estese e/o automatizzate tramite l'uso di **plugin**.

Funzionalità principali

- i documenti possono essere arricchiti da meta-dati.
- si possono creare delle relazioni navigabili tra i documenti.
- si possono effettuare delle ricerche sul catalogo o sul web.
- i doppioni vengono riconosciuti.
- supporto per qualunque tipo di file.
- **portabile**: funziona sia su OSs Windows che Linux.
- scalabile: ottime prestazioni anche con un elevato numero di documenti.
- le funzionalità possono essere estese e/o automatizzate tramite l'uso di **plugin**.

Funzionalità principali

- i documenti possono essere arricchiti da meta-dati.
- si possono creare delle relazioni navigabili tra i documenti.
- si possono effettuare delle ricerche sul catalogo o sul web.
- i doppioni vengono riconosciuti.
- supporto per qualunque tipo di file.
- portabile: funziona sia su OSs Windows che Linux.
- **scalabile**: ottime prestazioni anche con un elevato numero di documenti.
- le funzionalità possono essere estese e/o automatizzate tramite l'uso di **plugin**.

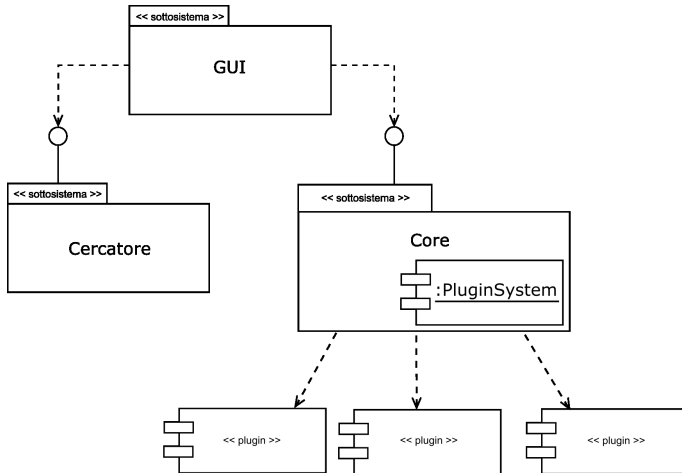
Funzionalità principali

- i documenti possono essere arricchiti da meta-dati.
- si possono creare delle relazioni navigabili tra i documenti.
- si possono effettuare delle ricerche sul catalogo o sul web.
- i doppioni vengono riconosciuti.
- supporto per qualunque tipo di file.
- portabile: funziona sia su OSs Windows che Linux.
- scalabile: ottime prestazioni anche con un elevato numero di documenti.
- le funzionalità possono essere estese e/o automatizzate tramite l'uso di **plugin**.

Scelte tecniche

- linguaggio di programmazione Java.
- librerie grafiche SWT.
- tools di sviluppo: Eclipse, CVS, Inkscape, Umbrello e LaTeX.
- largo utilizzo di componenti esistenti: Hsqldb, JUnit, JDom, Google API, Fast MD5, ecc.

Architettura



Vantaggi

La modularità dell'architettura comporta i seguenti vantaggi:

- è possibile sostituire la GUI senza perdere le funzionalità, ad esempio con:
 - servlet Java o PHP e HTML.
 - UI testuale (console Linux).
 - Java Swing.
- lo sviluppatore della GUI utilizza delle primitive molto semplici del Core e del ricercatore: non deve preoccuparsi dello stable storage, astrazione dei documenti simile agli *Entity EJB*.
- facilità di manutenzione.
- suddivisione più semplice delle responsabilità dei membri del team e possibilità di sviluppo concorrente.

Vantaggi

La modularità dell'architettura comporta i seguenti vantaggi:

- è possibile **sostituire la GUI** senza perdere le funzionalità, ad esempio con:
 - servlet Java o PHP e HTML.
 - UI testuale (console Linux).
 - Java Swing.
- lo sviluppatore della GUI utilizza delle primitive molto semplici del Core e del cercatore: non deve preoccuparsi dello stable storage, astrazione dei documenti simile agli *Entity EJB*.
- facilità di manutenzione.
- suddivisione più semplice delle responsabilità dei membri del team e possibilità di sviluppo concorrente.

Vantaggi

La modularità dell'architettura comporta i seguenti vantaggi:

- è possibile sostituire la GUI senza perdere le funzionalità, ad esempio con:
 - servlet Java o PHP e HTML.
 - UI testuale (console Linux).
 - Java Swing.
- lo sviluppatore della GUI utilizza delle primitive molto semplici del Core e del ricercatore: non deve preoccuparsi dello stable storage, **astrazione** dei documenti simile agli *Entity EJB*.
- facilità di manutenzione.
- suddivisione più semplice delle responsabilità dei membri del team e possibilità di sviluppo concorrente.

Vantaggi

La modularità dell'architettura comporta i seguenti vantaggi:

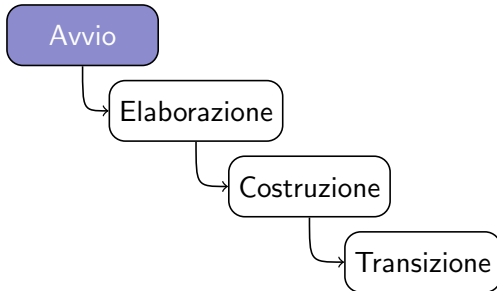
- è possibile sostituire la GUI senza perdere le funzionalità, ad esempio con:
 - servlet Java o PHP e HTML.
 - UI testuale (console Linux).
 - Java Swing.
- lo sviluppatore della GUI utilizza delle primitive molto semplici del Core e del ricercatore: non deve preoccuparsi dello stable storage, astrazione dei documenti simile agli *Entity EJB*.
- facilità di **manutenzione**.
- suddivisione più semplice delle responsabilità dei membri del team e possibilità di sviluppo concorrente.

Vantaggi

La modularità dell'architettura comporta i seguenti vantaggi:

- è possibile sostituire la GUI senza perdere le funzionalità, ad esempio con:
 - servlet Java o PHP e HTML.
 - UI testuale (console Linux).
 - Java Swing.
- lo sviluppatore della GUI utilizza delle primitive molto semplici del Core e del ricercatore: non deve preoccuparsi dello stable storage, astrazione dei documenti simile agli *Entity EJB*.
- facilità di manutenzione.
- suddivisione più semplice delle **responsabilità** dei membri del team e possibilità di sviluppo concorrente.

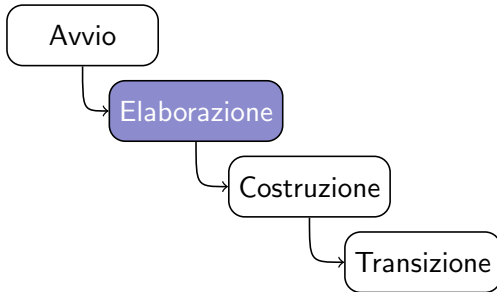
Processo Waterfall con prototipazione



Flusso di lavoro:

- revisione e analisi dei requisiti
- inizio del prototipo
- stesura di un piano operativo
- stime dei costi e dei rischi

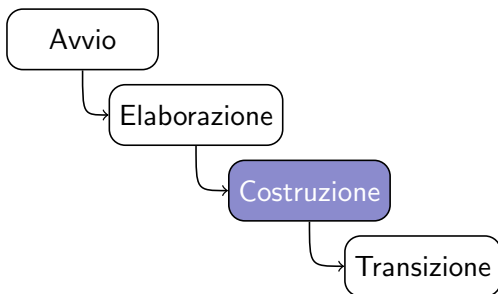
Processo Waterfall con prototipazione



Flusso di lavoro:

- progettazione
- prime implementazioni

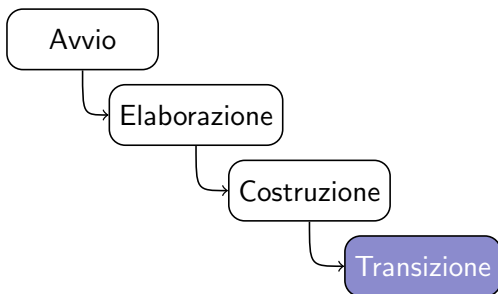
Processo Waterfall con prototipazione



Flusso di lavoro:

- implementazione del Core
- il prototipo si evolve nella GUI
- test delle unità

Processo Waterfall con prototipazione



Flusso di lavoro:

- test finali, validazione
- completamento della documentazione
- pubblicazione

Modifiche al piano di processo

Ragioni che hanno portato al ritardo nello sviluppo:

- LOC richieste per implementare il progetto maggiori di quelle previste, quindi
- molte ore dedicate ai test (*evitare l'integration hell*), più di quante ne aveva previste COCOMO.
- errata pianificazione temporale dei task, in data 31/1 non erano ancora ultimati:
 - alcuni dei plugin previsti nelle specifiche.
 - il componente cercatore.
 - l'interfaccia grafica: l'inesperienza con le librerie SWT era tra i principali rischi previsti.

Sforzo e costo in base a COCOMO II

Previsione iniziale

Circa **5000 LOC** per uno sforzo totale di **12 mp** e un costo tra 12000 e 27000 \$.

Risultato finale

Circa 12000 LOC (inclusi i plugin ma esclusi i test e la documentazione), 714 ore totali di lavoro. Se 1 mp equivale a 152 ore di lavoro (secondo COCOMO), lo sforzo complessivo è di 4,7 mp.

Sforzo e costo in base a COCOMO II

Previsione iniziale

Circa 5000 LOC per uno sforzo totale di 12 mp e un costo tra 12000 e 27000 \$.

Risultato finale

Circa **12000 LOC** (inclusi i plugin ma esclusi i test e la documentazione), **714 ore** totali di lavoro. Se 1 mp equivale a 152 ore di lavoro (secondo COCOMO), lo sforzo complessivo è di **4,7 mp**.

A posteriori...

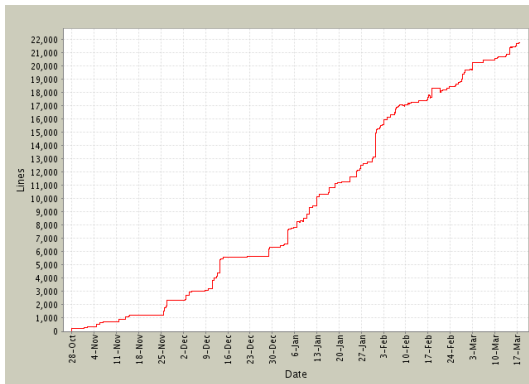
Terminato il progetto abbiamo ricalcolato lo sforzo con COCOMO II usando dei parametri più ottimistici e in particolare:

- ridotta l'influenza dei rischi;
- aumentata la flessibilità e l'esperienza del team;
- ridotta la complessità del progetto (da alta a normale);

Anche se le linee di codice ora sono 12000...

Il coefficiente di sforzo si riduce da 0,65 a 0,15 e lo sforzo finale è stimato a 4,5 mp.

Crescita del software in termini di LOC



55% LOC del sistema, 24% LOC dei test, 21% LOC della documentazione.

Il ciclo di vita del software non è terminato

Il progetto è stato registrato su SourceForge, verrà rilasciato con
licenza GPL:

`http://yadoda.sourceforge.net`

Yet Another Document Oriented Desktop Application

Correzioni al processo di sviluppo

- ripetere il modello a cascata per l'introduzione di ogni nuova funzionalità: il processo nell'insieme diventa a spirale.
- pianificare delle revisioni tecniche formali.
- utilizzare JML e il *design by contract* per generare automaticamente i test di tipo *black-box*.
- utilizzare il servizio di *bug tracking* di SourceForge.
- migliorare la pianificazione dei rilasci.

Correzioni al processo di sviluppo

- ripetere il modello a cascata per l'introduzione di ogni nuova funzionalità: il processo nell'insieme diventa a **spirale**.
- pianificare delle revisioni tecniche formali.
- utilizzare JML e il *design by contract* per generare automaticamente i test di tipo *black-box*.
- utilizzare il servizio di *bug tracking* di SourceForge.
- migliorare la pianificazione dei rilasci.

Correzioni al processo di sviluppo

- ripetere il modello a cascata per l'introduzione di ogni nuova funzionalità: il processo nell'insieme diventa a spirale.
- pianificare delle **revisioni tecniche formali**.
- utilizzare JML e il *design by contract* per generare automaticamente i test di tipo *black-box*.
- utilizzare il servizio di *bug tracking* di SourceForge.
- migliorare la pianificazione dei rilasci.

Correzioni al processo di sviluppo

- ripetere il modello a cascata per l'introduzione di ogni nuova funzionalità: il processo nell'insieme diventa a spirale.
- pianificare delle revisioni tecniche formali.
- utilizzare JML e il *design by contract* per generare automaticamente i test di tipo *black-box*.
- utilizzare il servizio di *bug tracking* di SourceForge.
- migliorare la pianificazione dei rilasci.

Correzioni al processo di sviluppo

- ripetere il modello a cascata per l'introduzione di ogni nuova funzionalità: il processo nell'insieme diventa a spirale.
- pianificare delle revisioni tecniche formali.
- utilizzare JML e il *design by contract* per generare automaticamente i test di tipo *black-box*.
- utilizzare il servizio di *bug tracking* di SourceForge.
- migliorare la pianificazione dei rilasci.

Correzioni al processo di sviluppo

- ripetere il modello a cascata per l'introduzione di ogni nuova funzionalità: il processo nell'insieme diventa a spirale.
- pianificare delle revisioni tecniche formali.
- utilizzare JML e il *design by contract* per generare automaticamente i test di tipo *black-box*.
- utilizzare il servizio di *bug tracking* di SourceForge.
- migliorare la **pianificazione dei rilasci**.

Modifiche e nuove funzionalità

- migliorare il meccanismo di ricerca, ad esempio permettendo l'utilizzo di shell pattern (o regular expression) sui meta-value.
- migliorare il meccanismo di deduzione dei tipi di file (abbandonare il mime-type).
- creare nuovi plugin per dedurre le categorie di catalogazione, eventuali relazioni.
- creare plugin intelligenti per la ricerca dei doppioni, sfruttando ad esempio le tecniche usate per combattere lo spam.
- creare nuove viste grafiche, ad esempio per mezzo di grafi.