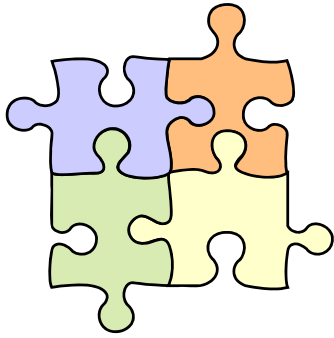


Luca Cabibbo



Architetture Software

Punto di vista dello Sviluppo

Dispensa AS 19

ottobre 2008

1

Punto di vista dello Sviluppo

Luca Cabibbo – SwA



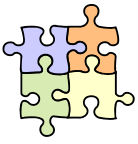
- Fonti

- [SSA] Chapter 19, The Development Viewpoint

2

Punto di vista dello Sviluppo

Luca Cabibbo – SwA



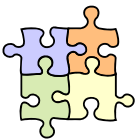
- Obiettivi e argomenti

□ Obiettivi

- descrivere il punto di vista dello Sviluppo

□ Argomenti

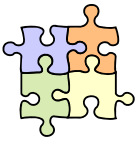
- punto di vista dello sviluppo
- interessi
- modelli
- problemi e insidie



* Punto di vista dello Sviluppo

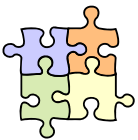
- Il **punto di vista dello Sviluppo** descrive l'architettura che sostiene il processo di sviluppo del software

- interessi
 - organizzazione in moduli; elaborazione comune; standardizzazione del progetto; standardizzazione del testing; strumentazione; organizzazione del codice sorgente
- modelli
 - modelli della struttura dei moduli; modelli di progetto comuni; modelli del codice
- parti interessate
 - sviluppatori e verificatori
- applicabilità
 - tutti i sistemi



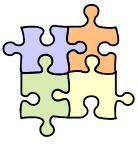
Vista dello sviluppo

- La *vista dello sviluppo*
 - descrive la struttura e l'organizzazione del codice, la gestione delle dipendenze, la gestione delle configurazioni, e l'uso di linee guida e politiche comuni nel progetto
 - per fornire un approccio omogeneo all'implementazione e alla verifica del software
- Rilevanza
 - la vista dello sviluppo costituisce il punto di partenza per il lavoro successivo di progettazione, implementazione e test
 - i moduli sono solitamente l'unità di base del lavoro di sviluppo e rilascio



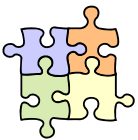
* Interessi (1)

- Organizzazione in moduli
 - l'organizzazione del codice in *moduli* strutturati aiuta a comprendere meglio il codice e le dipendenze tra i suoi elementi
 - ragionando sulle dipendenze è possibile capire quanto un sistema è comprensibile, costruibile, modificabile
- Elaborazione comune
 - opportuno separare aspetti comuni nel codice, e garantirne una gestione uniforme
 - ad es., logging e parametri di configurazione
 - evita ripetizioni e inconsistenze



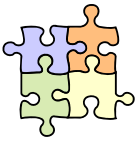
Interessi (2)

- Standardizzazione del progetto
 - l'uso di standard e di convenzioni migliora la comprensibilità e la modificabilità del codice, riducendo il tempo di sviluppo
 - ad es., uso di design pattern, convenzioni per l'avvio e l'arresto di moduli/processi
- Standardizzazione del testing
 - quali approcci, tecnologie e convenzioni seguire per le diverse attività di verifica del software – test unitari, di integrazione, di carico, usabilità, ...



Interessi (3)

- Strumentazione
 - per *strumentazione* si intende la pratica di aggiungere codice speciale
 - ad es., logging per favorire il debugging e la profilazione
 - questo codice deve per poter essere successivamente disattivato e/o eliminato
- Organizzazione del codice sorgente
 - organizzazione del codice (sorgente ed eseguibile) nel file system o in un CVS
 - convenzioni di codifica
 - un'organizzazione omogenea favorisce le attività legate all'implementazione



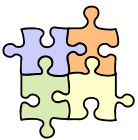
* Modelli

- Modelli per la vista dello sviluppo
 - modelli per la struttura dei moduli
 - ad es., diagrammi dei package di UML
 - modelli di progetto comuni – descrivono
 - come gestire l'elaborazione comune
 - ad es., logging, internazionalizzazione, strumentazione, avvio e arresto di operazioni, sicurezza, transazioni, ...
 - approcci di progetto standard
 - ad es., design pattern per il collegamento tra i diversi elementi del progetto
 - software comune
 - ad es., librerie, framework, middleware
 - modelli per l'organizzazione del codice
 - ad es., struttura del file system o del CVS

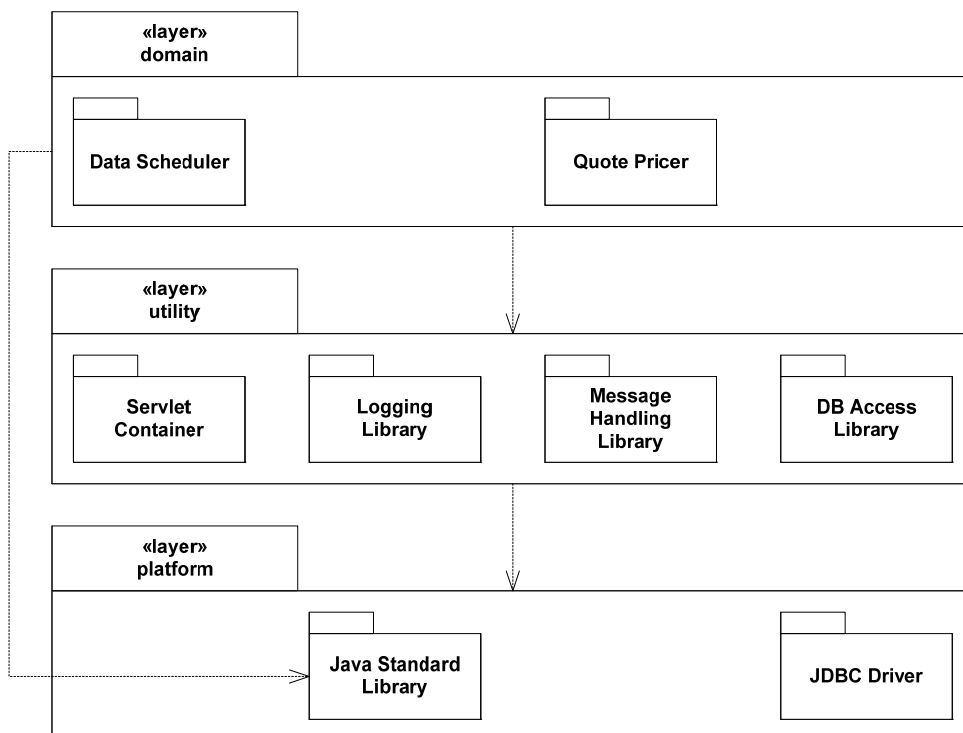
9

Punto di vista dello Sviluppo

Luca Cabibbo – SwA



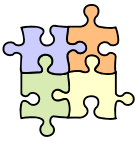
Esempio- diagramma dei package di UML



10

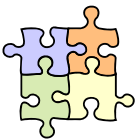
Punto di vista dello Sviluppo

Luca Cabibbo – SwA



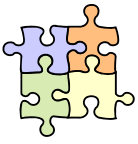
Esempio - progettazione di package

- Alcune linee guida – vedi [Applicare UML e i pattern]
 - package per responsabilità funzionalmente coese
 - package per una famiglia di interfacce funzionalmente correlate
 - nessun ciclo nei package
 - esistono tecniche di partizionamento/accorpamento di package per ridurre l'accoppiamento complessivo
 - riduci le dipendenze verso package non stabili
 - i package più responsabili hanno l'interfaccia più stabile
 - separa tipi indipendenti
 - usa factory per ridurre le dipendenze tra package concreti
- Per approfondimenti
 - R.C. Martin, Agile Software Development



* Problemi e insidie

- Troppo dettaglio
 - bisogna concentrarsi su aspetti architetturealmente significativi
- AD sovrappesantita
 - i dettagli importanti vanno forse riportati separatamente in un'appendice dell'AD
- Focalizzazione non uniforme
 - con il rischio di concentrarsi su aspetti semplici – tralasciando magari quelli più complessi come l'avvio e l'arresto dei moduli



Problemi e insidie (2)

- Mancanza di attenzione allo sviluppatore
 - lo sviluppatore è la parte interessata più importante per questa vista – che va rivolta a lui
- Mancanza di precisione
 - descrizioni imprecise possono essere ignorate o travisate dagli sviluppatori
- Problemi con l'ambiente utilizzato
 - le tecnologie cambiano rapidamente
 - alcune indicazioni valide per un progetto potrebbero non essere le più opportune in un progetto successivo