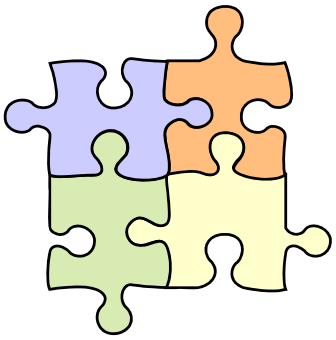
Luca Cabibbo

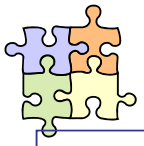


Architetture Software

Punto di vista Funzionale

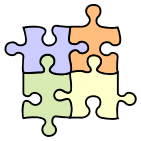
Dispensa AS 16

ottobre 2008



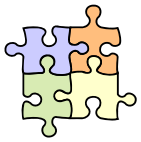
- Fonti

- [SSA] Chapter 16, The Functional Viewpoint



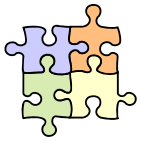
- Obiettivi e argomenti

- Obiettivi
  - descrivere il punto di vista Funzionale
- Argomenti
  - punto di vista funzionale
  - interessi
  - modelli e attività
  - problemi e insidie



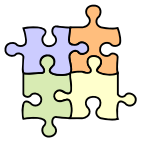
* Punto di vista Funzionale

- Il **punto di vista Funzionale** descrive gli elementi funzionali a runtime del sistema, le loro responsabilità, interfacce e interazioni primarie
  - interessi
    - capacità funzionali; interfacce esterne; struttura interna; filosofia di progettazione
  - modelli e attività
    - modello della struttura funzionale
    - progettazione della struttura funzionale
  - parti interessate
    - tutte le parti interessate
  - applicabilità
    - tutti i sistemi



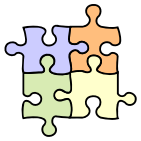
Vista funzionale

- La **vista funzionale**
  - definisce gli elementi funzionali del sistema – con le loro responsabilità, interfacce e interazioni
  - complessivamente, dimostra come il sistema può offrire le funzionalità richieste
- Rilevanza
  - la vista funzionale è la prima (l'unica?) che molte parti interessate provano a leggere
  - guida la definizione delle altre viste architetturali
- Una rivisitazione della *vista logica* di Kruchten



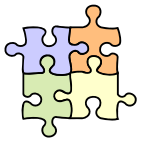
Livello di dettaglio

- Attenzione al livello di dettaglio
  - concentrarsi solo su ciò che è "architetturalmente significativo"
  - un interesse, problema o elemento del sistema è *architetturalmente significativo* se ha un impatto ampio sulla struttura del sistema o su una sua qualità importante – come prestazioni, scalabilità, sicurezza, ...
  - il resto va lasciato ai "progettisti"



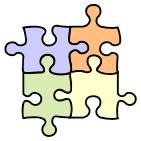
Contesto della definizione dell'architettura

- Ricordiamo il contesto della definizione dell'architettura
  - viene identificato un insieme di scenari significativi (funzionali e non funzionali) – ovvero, le funzionalità e qualità che devono essere fornite/possedute dal sistema
  - le qualità prioritarie sono usate come criterio di selezione per uno o più stili architetturali rilevanti – che sostengono tali qualità
  - ciascuno stile architetturale fornisce dei criteri di decomposizione dell'architettura, ovvero criteri di scelta degli elementi (funzionali, in particolare) e delle relazioni tra di essi
  - la definizione delle viste architetturali va svolta in modo iterativo ed evolutivo – analizzando e valutando le scelte fatte a fronte degli obiettivi di qualità



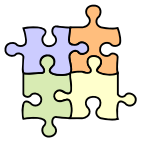
Importanza dell'architettura e delle viste

- La decomposizione effettuata (in questa vista come in altre viste) ha una forte influenza sulle qualità del sistema
  - le principali qualità richieste guidano la scelta dello stile architetturale
  - lo stile architetturale guida la decomposizione – ovvero, la scelta degli elementi
  - la decomposizione va validata rispetto a (tutte) le qualità richieste



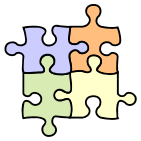
* Interessi

- Interessi affrontati dalla vista funzionale
  - capacità funzionali
  - interfacce esterne
  - struttura interna
  - filosofia di progettazione



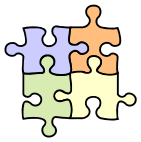
Capacità funzionali

- *Capacità funzionali*
  - la vista funzionale è interessata a comprendere/descrivere che cosa il sistema deve fare
    - in modo complementare, anche che cosa non deve fare, perché fuori dalla portata del sistema
  - legame forte con i requisiti funzionali



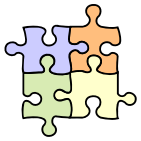
Interfacce esterne

- *Interfacce esterne*
  - la vista funzionale è interessata alle interfacce (flussi di controllo e di dati, entranti e uscenti) del sistema – verso i suoi attori e verso altri sistemi
    - flusso di controllo entrante – è richiesto al sistema di eseguire un compito
    - flusso di controllo uscente – il sistema richiede ad un altro sistema di eseguire un compito
    - flusso di dati entrante – da elaborare, o per cambiare lo stato del sistema
    - flusso di dati uscente – una risposta o una notifica
  - queste informazioni sono solitamente mostrate da un *diagramma di contesto*
    - mostra il sistema, gli attori che interagiscono con il sistema, i relativi flussi di collegamento



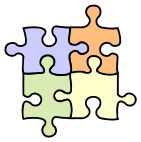
Struttura interna

- *Struttura interna*
  - la vista funzionale deve fornire una decomposizione del sistema in *elementi funzionali*
    - con una caratterizzazione di che cosa fanno questi elementi – quali le loro responsabilità funzionali
    - descrivendo come questi elementi interagiscono/collaborano per fornire le funzionalità complessive
  - in generale, sono possibili numerose scelte per sostenere gli stessi requisiti funzionali – anche sulla base di stili architetturali differenti
  - la struttura interna può avere un impatto profondo sulle qualità (non funzionali) del sistema
    - poiché le qualità richieste potrebbero essere contrastanti tra loro, la struttura interna ne determina un compromesso



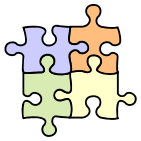
Filosofia di progettazione

- *Filosofia di progettazione*
  - esistono diversi principi di progettazione, che sostengono qualità interne differenti
    - ad esempio, costruibilità, verificabilità, evolvibilità, ...
  - su quali principi di progettazione fondamentali si desidera basare l'architettura del sistema?



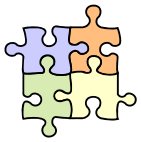
Principi e qualità (1)

- *Separazione degli interessi*
  - in che misura ciascun elemento interno è responsabile di una parte distinta di funzionalità? in che misura dell'elaborazione comune è eseguita in un solo posto?
  - un'alta separazione sostiene facilità di costruzione ed evoluzione – ma potrebbe influire negativamente su prestazioni e scalabilità



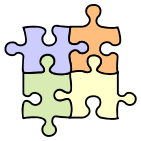
Principi e qualità (2)

- *Coesione*
  - in che misura le funzioni fornite da un elemento sono correlate tra loro?
  - una coesione alta porta a progetti più semplici, con elementi più facili da comprendere e riusare
- Attenzione, esistono varie forme di coesione – quali forme di coesione intendo sostenere?
  - coesione per pura coincidenza – elemento che contiene operazioni/servizi non correlati
  - coesione logica – elemento che contiene parti logicamente correlate, ma implementate in modo indipendente
  - coesione temporale – elemento che contiene operazioni usate circa nello stesso tempo
  - coesione di comunicazione – elemento che contiene operazioni che devono accedere agli stessi dati o dispositivi
  - coesione sequenziale – elemento che contiene operazioni usate in un ordine particolare
  - coesione funzionale – elemento che contiene operazioni che complessivamente svolgono una singola funzione
  - coesione dei dati – elemento che implementa o gestisce un tipo di dato



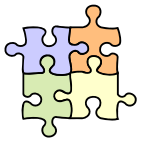
Principi e qualità (3)

- *Accoppiamento*
  - quanto sono forti le relazioni tra elementi? in che misura un cambiamento in un elemento influisce su altri elementi?
  - sistemi debolmente accoppiati sono più semplici da costruire e modificare – ma potrebbero essere peggiori in termini di prestazioni e scalabilità
- Attenzione, esistono varie forme di accoppiamento – quali intendo evitare/minimizzare?
  - accoppiamento di dati interni – un elemento modifica direttamente lo stato di un altro elemento
  - accoppiamento mediante dati globali – due o più elementi condividono dati globali
  - accoppiamento di controllo – una elemento esegue operazioni in un ordine fissato, ma l'ordine è controllato altrove
  - accoppiamento temporale
  - accoppiamento per composizione – un elemento controlla le operazioni svolte da altri elementi
  - accoppiamento per uso – un elemento richiede l'esecuzione di operazioni ad altri elementi



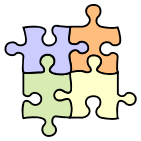
Principi e qualità (4)

- *Volume dell'interazione tra elementi*
  - in che proporzione i vari passi di elaborazione richiedono interazione tra elementi, anziché essere localizzati su un singolo elemento?
  - la comunicazione tra certi tipi di elementi può influire negativamente su prestazioni ed affidabilità



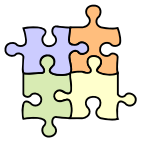
Principi e qualità (5)

- *Flessibilità funzionale*
  - in che misura il sistema è in grado di sostenere cambiamenti nelle funzionalità?
  - i sistemi progettati per essere più facili da cambiare possono essere più difficili da costruire, e possono avere prestazioni peggiori



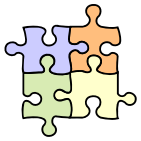
Principi e qualità (6)

- *Coerenza complessiva*
  - la struttura interna “sembra giusta”?
  - chiamata anche *integrità concettuale*
    - l’integrità concettuale implica che vengano usate un numero limitato di “forme” di progetto, e che vengano usate in modo uniforme [Davis]
    - I will contend that conceptual integrity is the most important consideration in system design – it is better to have a system omit certain anomalous features and improvements, but to reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas [Brooks]
  - se un’architettura “non sembra giusta”, allora è difficile da comprendere – il che potrebbe essere il sintomo di problemi sottostanti – l’architettura potrebbe essere difficile da realizzare



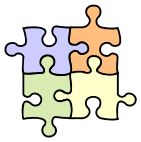
* Modelli

- I modelli per strutture funzionali contengono *tipicamente* i seguenti tipi di elementi
  - elementi funzionali
  - interfacce
  - connettori
  - entità esterne
- *Osservazione*
  - *in alcuni casi è opportuno usare modelli diversi o più ricchi di quello mostrato qui di seguito*



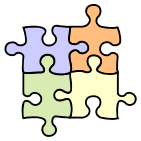
Elementi funzionali

- Elementi funzionali
  - un *elemento funzionale* è una parte ben definita del sistema runtime che ha responsabilità specifiche ed espone interfacce ben definite che consentono di collegarlo ad altri elementi
    - ad es., un modulo, un package, un data store, un sottosistema completo



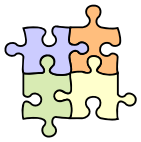
Interfacce

- Interfacce
  - un'*interfaccia* è un meccanismo ben definito che consente che le funzioni di un elemento possano essere accedute da altri elementi
  - un'interfaccia può essere definita in termini di
    - un insieme di operazioni (stile procedurale) – ciascuna con per ogni operazione, input, output, semantica (contratto)
    - un insieme di tipi di messaggi accettati (stile orientati ai documenti) – ciascuno con una descrizione dei dati accettati e della semantica (contratto) associata alla ricezione di un messaggio
  - un elemento può "fornire" oppure "richiedere" interfacce



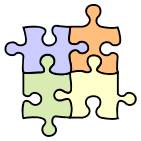
Connettori

- Connettori
  - i *connettori* sono i pezzi dell'architettura che collegano gli elementi e gli consentono di interagire
    - ad es., un protocollo
  - un connettore definisce le interazioni tra gli elementi che lo usano e consente di considerare la natura dell'interazione separatamente dalla semantica dell'operazione invocata
  - la rilevanza dei connettori dipende dalle circostanze
    - talvolta i connettori sono considerati semplicemente delle relazioni di connessione tra elementi
    - tuttavia, in alcuni casi sono considerati della stessa importanza degli elementi funzionali



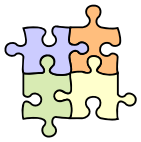
Entità esterne

- Entità esterne
  - le *entità esterne* rappresentano attori, altri sistemi o programmi *esterni* con cui il sistema comunica
  - da descrivere nel diagramma di contesto



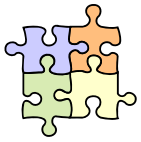
Notazioni

- Per la vista funzionale sono possibili numerose notazioni – più o meno formali
  - diagrammi delle classi e dei package di UML
  - diagrammi dei componenti di UML
  - altre notazioni formali
    - ad es., diagrammi dei flussi di dati
  - diagrammi a “rettangoli e linee”
  - abbozzi



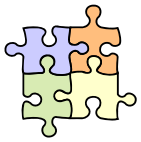
- Diagrammi dei componenti di UML

- [UML]
  - un *componente* è una parte modulare di un progetto di un sistema che nasconde la sua implementazione dietro un insieme di interfacce esterne
- Definizione molto generale
  - ad es., un modulo, un package, un data store, un sottosistema completo

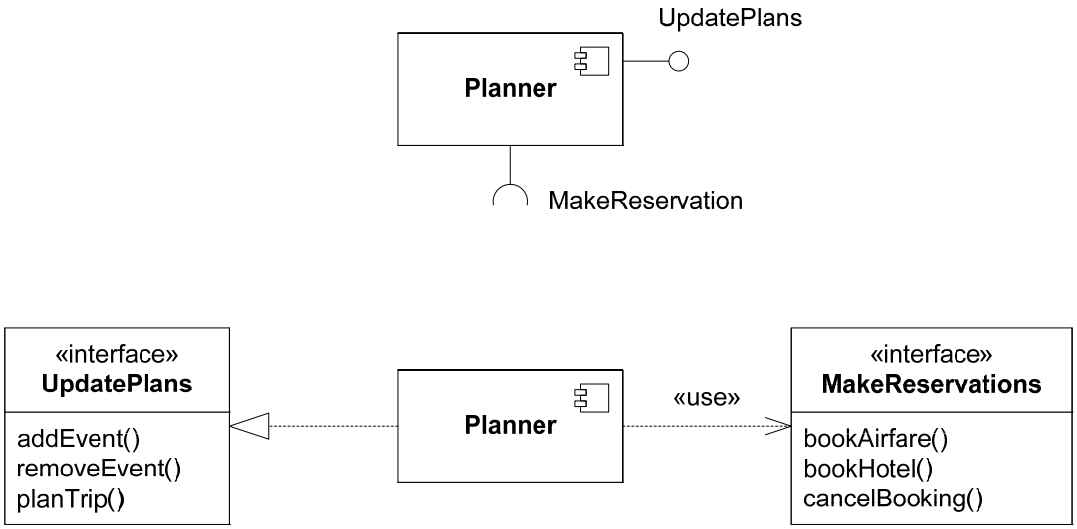


Notazione - componenti

UML 1.x



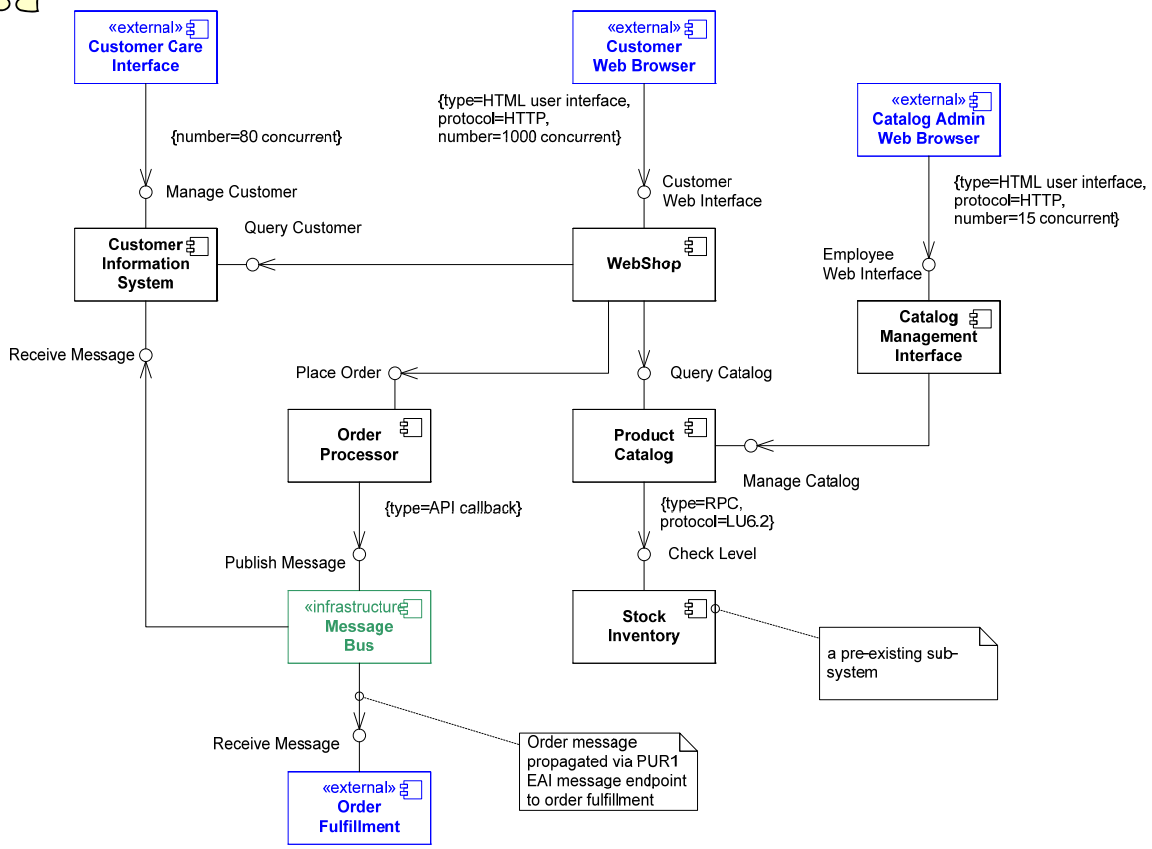
Notazione – interfacce fornite e richieste

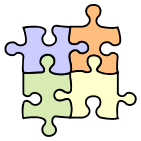


Esempio – WebShop

- WebShop è un sistema per l'acquisto di prodotti online
  - quattro entità esterne
    - tre attori principali – clienti, rappresentanti nel customer care, amministratori del catalogo
    - un sistema esterno per l'elaborazione degli ordini
  - sei componenti funzionali principali
    - [SSA] non fornisce la descrizione dettagliata dei componenti e delle interfacce
    - Stock Inventory è pre-esistente
  - un'infrastruttura funzionale – un bus di messagging – da sviluppare
  - alcune indicazioni di carico

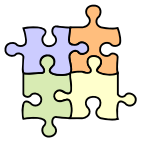
WebShop



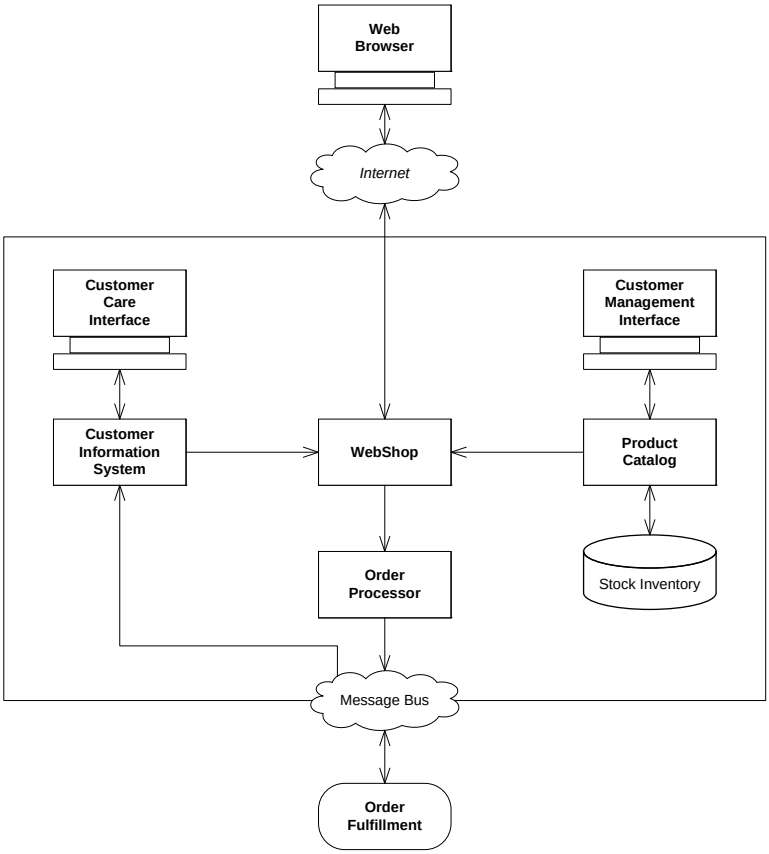


Webshop

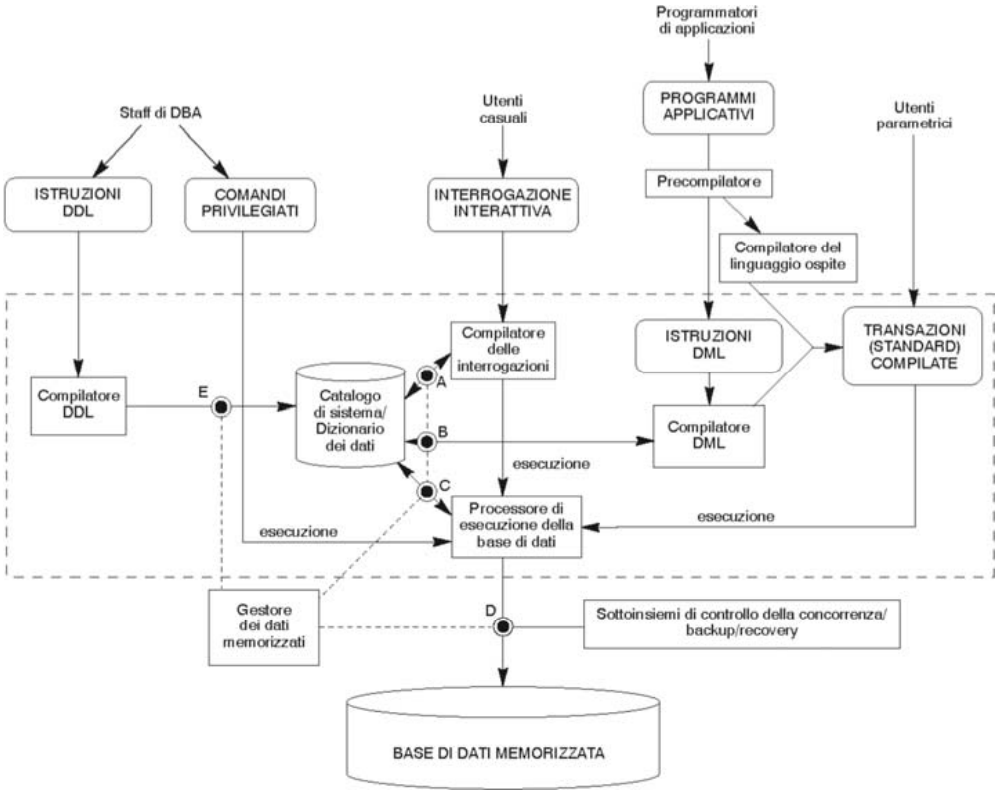
- Limitazioni di questa vista funzionale
  - quali sono le responsabilità dei componenti?
  - quali le interfacce (di dettaglio) dei componenti?
  - come collaborano i componenti per gestire i vari casi d'uso?
- Necessaria ancora
  - una descrizione testuale delle responsabilità dei componenti
  - modelli per descrivere la struttura dinamica della vista funzionale – le interazioni/collaborazioni



Esempio – Webshop a "rettangoli e linee"



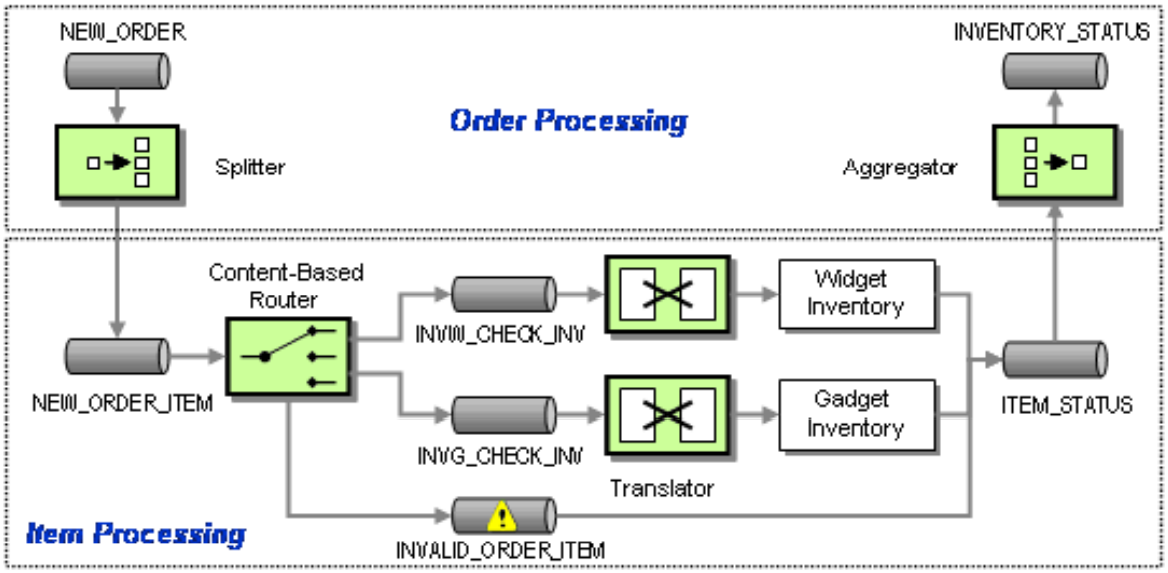
Esempio – vista funzionale di un DBMS



Interazione procedurale e messaggi

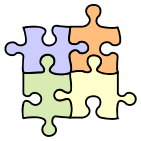
- Le interazioni tra componenti possono essere
  - in stile procedurale
    - operazioni relative all'erogazione di un servizio – con un protocollo richiesta/risposta, sincrono
  - in stile orientato ai messaggi/documenti
    - scambio asincrono di messaggi
    - è richiesta una notazione diversa, che mostra i tipi di messaggi scambiati e le connessioni tra elementi

Esempio



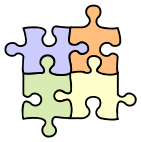
* Attività

- Attività nella definizione di una vista funzionale
  - identificazione degli elementi
  - assegnazione di responsabilità agli elementi
  - progettazione delle interfacce
  - verifica della tracciabilità delle funzioni
  - prova degli scenari comuni
  - analisi delle interazioni
  - analisi della flessibilità



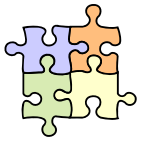
Identificazione degli elementi (1)

- Un possibile approccio (molto generale) per la scelta degli elementi funzionali
  - considera i requisiti funzionali, ed identifica le responsabilità fondamentali
  - identifica elementi funzionali candidati a cui assegnare queste responsabilità
    - in alcuni casi, potrebbero essere elementi già esistenti (librerie o componenti)
  - valutazione degli elementi scelti rispetto ai criteri di progettazione desiderabili
    - ad es., coesione e accoppiamento
  - itera, raffinando la struttura funzionale fino a quando non è considerata adeguata



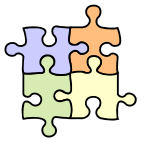
Identificazione degli elementi (2)

- L'identificazione degli elementi dipende anche da scelte di livello più alto
  - stile di decomposizione
    - la decomposizione può essere orientata agli oggetti, procedurale, basata su componenti, ...
  - stile di controllo
    - centralizzato, delegato, disperso, ...
  - stile architetturale
    - l'adozione di uno stile architetturale può influire sulla natura degli elementi e delle relazioni ammissibili tra elementi
      - ad esempio, se ho scelto Pipes and Filters, gli elementi funzionali saranno filtri oppure pipe
      - si noti che uno stile architetturale descrive anche i criteri di scelta per i relativi elementi



Identificazione degli elementi (3)

- La scelta degli elementi dipende anche dalle qualità desiderate – si veda “filosofia di progettazione” – ecco alcuni esempi un po’ generici
  - modificabilità – elementi con responsabilità coese a grana fine
  - prestazioni – pochi elementi con responsabilità a grana grossa
  - disponibilità – presenza di elementi replicati



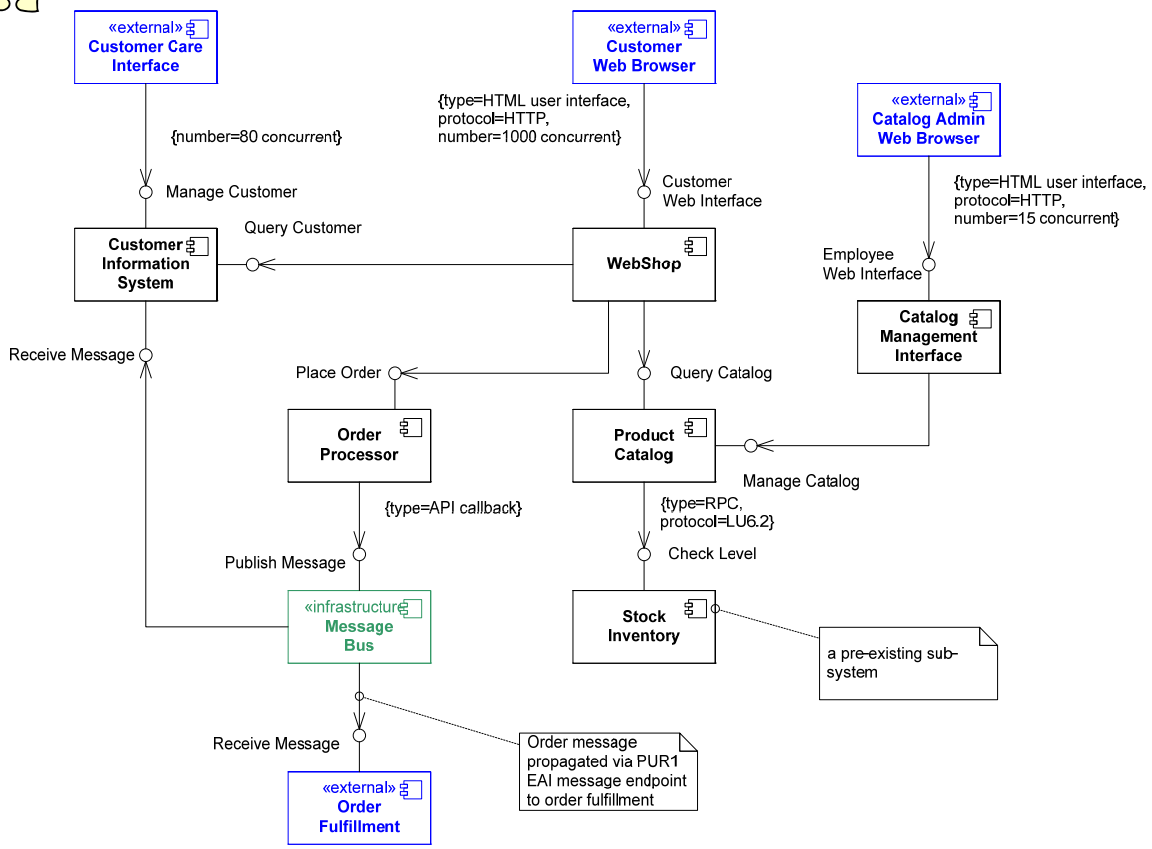
Identificazione degli elementi (4)

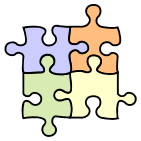
- Alcuni possibili raffinamenti comuni nella definizione degli elementi funzionali
  - decomposizione
    - decomponi un elemento con molte responsabilità funzionali in più elementi
  - compressione – accorpamento
    - ad es., per ridurre il volume delle interazione tra elementi
  - generalizzazione
    - identifica responsabilità comuni a più elementi ed assegnale ad un singolo nuovo elemento
  - replicazione
    - di un elemento o di una responsabilità

Assegnazione di responsabilità

- La vista funzionale deve assegnare *responsabilità* agli elementi funzionali
  - responsabilità di conoscere – gestione di informazioni
  - responsabilità di fare – ad es., trasformare informazioni o controllare l'esecuzione di operazioni ed attività
  - l'assegnazione di responsabilità deve essere chiara ed esplicita

Esempio – WebShop

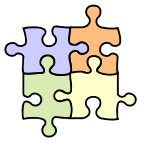




Esempio - WebShop

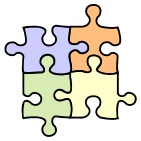
- WebShop
  - presenta ai clienti del sistema un'interfaccia utente unificata basata su HTML – accessibile mediante un browser web su Internet
  - gestisce tutto lo stato delle sessioni con i clienti
  - non implementa direttamente le funzionalità viste dall'utente – piuttosto, interagisce con gli altri elementi funzionali del sistema per consentire ai clienti di accedere al catalogo dei prodotti, di conoscere la disponibilità dei prodotti, di acquistare prodotti e di visualizzare le loro informazioni

- Customer Information System
  - gestisce tutte le informazioni persistenti sui clienti del sistema
  - consente ad un utente (tramite WebShop) di accedere alle proprie informazioni
  - fornisce un'interfaccia (API) per consentire di definire applicazioni per la gestione di informazioni su clienti
  - fornisce un'interfaccia event-driven per accettare dettagli su ordini posti dai clienti e sui cambiamenti di stato relativi a questi ordini



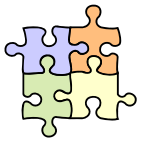
Progettazione dell'interfaccia

- I servizi offerti da ciascun elemento funzionale devono poter essere acceduti tramite un'interfaccia ben definita
  - possibili diversi approcci
    - segnatura delle operazioni, con parametri di input e output, pre-condizioni, post-condizioni ed invarianti – dunque, sintassi e semantica
    - formato dei messaggi ricevuti e trasmessi – per gli elementi che comunicano tramite messaggi
    - altri dettagli – ad es. protocollo e tecnologia, operazione/messaggio sincrono o asincrono
  - possibili diverse notazioni
    - ad es., sintassi dei PL, linguaggi formali, IDL, XML-Schema, ...



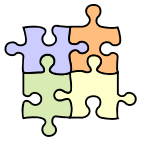
Altre attività

- Valutazione della tracciabilità delle funzioni
  - verifica di mutua completezza/consistenza con i requisiti funzionali
- Prova di scenari comuni
  - dimostrare come gli elementi interagiscono per soddisfare gli scenari più significativi
    - dapprima, si considerano gli scenari funzionali – il sistema è in grado di erogare le funzionalità richieste? come?
    - vanno considerati anche gli scenari non funzionali – il sistema garantisce le qualità richieste? come?
  - attività fondamentale di progettazione dinamica



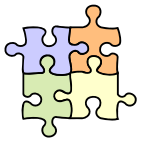
Altre attività

- Analisi delle interazioni
  - revisione delle qualità statiche (ad es., modularità) e dinamiche (volume delle informazioni scambiate) della struttura proposta
- Analisi della flessibilità
  - analisi what-if dell'impatto di possibili cambiamenti attesi



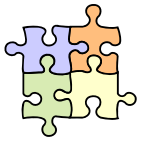
* Problemi e insidie

- Alcuni problemi e insidie comuni nella definizione di una vista funzionale
  - responsabilità comprese male
  - interfacce definite male
  - infrastruttura modellata come elementi funzionali
  - viste sovraccariche
  - diagrammi senza definizione degli elementi
  - difficoltà nel riconciliare i bisogni di diverse parti interessate
  - livello di dettaglio non appropriato
  - elementi “dio”
  - troppe dipendenze



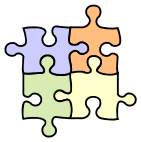
Responsabilità comprese male

- Responsabilità comprese male
  - importante capire *esattamente* cosa ciascun elemento funzionale deve fare – o non fare, perché fuori portata
  - ad es., difficile fare una verifica di completezza e consistenza rispetto ai requisiti funzionali se le responsabilità non sono specificate esplicitamente
    - ci potrebbero essere responsabilità funzionali non assegnate o assegnate a più elementi
- Tecniche di riduzione del rischio
  - definisci le responsabilità degli elementi in modo chiaro e presto
  - la progettazione di un elemento non deve iniziare se le responsabilità dell’elemento non sono state definite, riviste ed approvate
  - gli sviluppatori devono comprendere i confini dei vari elementi
  - tutti i requisiti devono essere stati mappati su elementi che li implementeranno



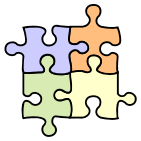
Interfacce definite male

- Interfacce definite male
  - le interfacce sono importanti perché descrivono le modalità di interazione tra elementi
  - se definite male, possibili rischi di incomprensioni e difficoltà di integrazione
- Tecniche di riduzione del rischio
  - definisci le interfacce in modo chiaro e presto
  - rivedi le interfacce e assicurati che siano ben comprese
  - non considerare completa la definizione di un elemento se l'interfaccia non è definita
  - assicurati che la definizione delle interfacce comprendano operazioni, la loro semantica, ed esempi di uso



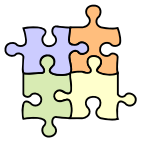
Infrastrutture come elementi funzionali

- Infrastrutture modellate come elementi funzionali
  - infrastrutture – elementi di middleware come, ad es., application server e database server
    - non vanno mostrati nella vista funzionale – possono generare confusione
    - ma, ad es., le basi di dati sono elementi funzionali
  - le infrastrutture vanno mostrate nella vista di deployment
  - da mostrare nella vista funzionale se l'infrastruttura deve essere implementata appositamente nel sistema in discussione
- Tecniche di riduzione del rischio
  - evita di modellare elementi dell'infrastruttura sottostante
  - metti in dubbio la necessità di elementi il cui nome non è correlato al dominio del problema affrontato dal sistema
  - affronta problemi infrastrutturali in altre viste



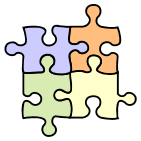
Vista sovraccarica

- Vista sovraccarica
  - c'è sempre la tentazione di aggiungere alla vista funzionale dettagli o annotazioni su aspetti normalmente di interesse per altre viste
  - una tale descrizione è normalmente difficile da comprendere – e quindi poco utile
- Tecniche di riduzione del rischio
  - rimuovi tutto dalla vista funzionale – tranne gli elementi funzionali e le loro relazioni
  - crea altre viste per descrivere altri aspetti dell'architettura
  - sviluppa le altre viste in parallelo – introduci riferimenti incrociati tra viste



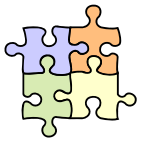
Diagrammi senza definizione degli elementi

- Diagrammi senza definizione degli elementi
  - diagrammi con elementi con una definizione incompleta e imprecisa
- Tecniche di riduzione del rischio
  - definisci ciascun elemento aggiunto alla vista (nel momento in cui viene aggiunto alla vista) – e rivedine la definizione con le altri parti interessate
  - un modello non è completo fino a quando ogni elemento non ha una buona definizione



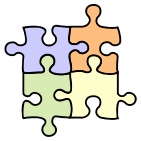
Difficoltà nel riconciliare bisogni diversi

- Difficoltà nel riconciliare i bisogni di diverse parti interessate
  - la vista funzionale normalmente interessa tutte le parti interessate – deve consentire la comunicazione con ciascuna di esse
  - difficile trovare una notazione e uno stile che va bene per tutte le parti interessate
- Tecniche di riduzione del rischio
  - usa modelli differenti con parti interessate diverse
  - importante un insieme di modelli “tecnici” per comunicare con le parti interessate tecniche
  - usa modelli semplificati per le parti interessate non tecniche



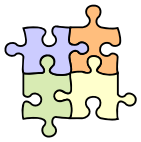
Livello di dettaglio non appropriato

- Livello di dettaglio non appropriato
  - un problema comune nella definizione della vista architetturale – dove finisce la definizione dell’architettura ed inizia la progettazione dei suoi elementi?
- Tecniche di riduzione del rischio
  - ci sono segni di dettaglio eccessivo, che si riconoscono con l’esperienza – numero eccessivo di decomposizioni, numero eccessivo di elementi, troppi dettagli sulla struttura o sul funzionamento interno degli elementi



Elementi "dio"

- Elementi "dio"
  - presenza di un elemento che ha troppe responsabilità ed è accoppiato con molti altri elementi – i quali hanno poche responsabilità
  - un tale elemento è difficile da comprendere, sviluppare e mantenere
  - rende difficile risolvere le varie qualità richieste
- Tecniche di riduzione del rischio
  - le principali responsabilità devono essere distribuite uniformemente tra i principali elementi del sistema



Troppe dipendenze

- Troppe dipendenze
  - il problema opposto a quello degli elementi "dio" – responsabilità distribuite in modo eccessivo, che richiede un accoppiamento elevato tra gli elementi del sistema
- Tecniche di riduzione del rischio
  - comprimi (accorpa) elementi
  - riorganizza la scelta degli elementi – anche sulla base di obiettivi di accoppiamento e coesione