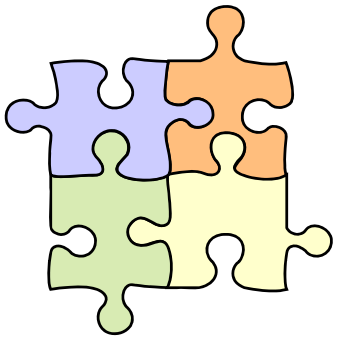
Luca Cabibbo

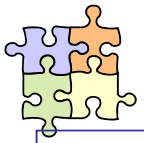


Architetture Software

Punto di vista delle Informazioni

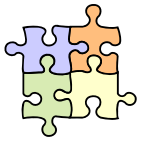
Dispensa AS 17

ottobre 2008



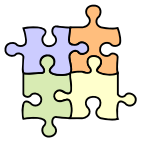
- Fonti

- [SSA] Chapter 17, The Information Viewpoint



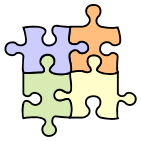
- Obiettivi e argomenti

- Obiettivi
  - descrivere il punto di vista delle Informazioni
- Argomenti
  - punto di vista delle informazioni
  - interessi
  - modelli
  - problemi e insidie



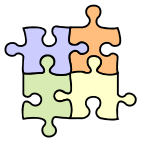
* Punto di vista delle Informazioni

- Il **punto di vista delle Informazioni** descrive il modo in cui l'architettura memorizza, manipola, gestisce e distribuisce le informazioni
  - interessi
    - contenuto e struttura delle informazioni; flussi di dati; ownership dei dati; tempestività (timeliness), latenza ed età (age); riferimenti e corrispondenze; gestione delle transazioni; qualità dei dati; volumi dei dati; conservazione dei dati e degli archivi; leggi e regolamenti
  - modelli
    - struttura statica dei dati; flussi di informazioni; ciclo di vita delle informazioni; ownership dei dati; analisi di qualità dei dati; metadati; modelli volumetrici
  - applicabilità
    - sistemi che richiedono una gestione dei dati non banale



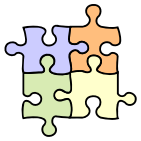
Vista delle informazioni

- La *vista delle informazioni*
  - definisce il modo in cui l'architettura gestisce le informazioni
  - in particolare, descrive
    - i tipi di informazioni gestiti dal sistema
    - le relazioni tra questi tipi di informazioni e i vari elementi software dell'architettura – compresi i flussi di informazioni scambiati tra di essi
    - le relazioni tra questi tipi di informazioni e l'ambiente – ovvero, con le "informazioni vere"
- Rilevanza
  - le informazioni sono un patrimonio significativo di molte organizzazioni e sistemi informatici



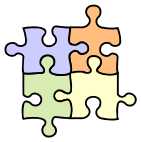
Livello di dettaglio

- Enfasi delle informazioni nella definizione dell'architettura
  - concentrarsi sugli aspetti architetturalmente significativi della struttura statica e del flusso dinamico delle informazioni
    - non analisi e progettazione dei dati di dettaglio
  - per rispondere a domande architetturalmente importanti
    - come vengono memorizzate, manipolate, gestite e distribuite le informazioni tra gli elementi dell'architettura?



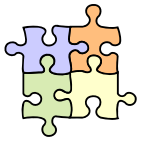
* Interessi

- Interessi affrontati dalla vista delle informazioni
  - contenuto e struttura delle informazioni
  - allocazione delle informazioni
  - flussi di dati
  - possesso (ownership) dei dati
  - tempestività (timeliness), latenza ed età (age)
  - riferimenti e corrispondenze tra entità
  - gestione delle transazioni
  - qualità dei dati
  - volumi dei dati
  - leggi e regolamenti
  - conservazione (retention) dei dati e degli archivi



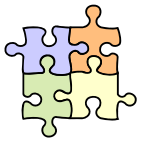
Un'osservazione

- Ci sono altri interessi relativi alla gestione dei dati e delle informazioni – che però non compaiono nel precedente elenco
  - ad esempio, replicazione e backup
  - non sono in questo elenco perché l'interesse prevalente è – secondo [SSA] – di altri punti di vista (in particolare, di deployment) o prospettive (ad es., della sicurezza e della disponibilità)



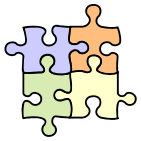
Contenuto e struttura delle informazioni

- *Contenuto e struttura delle informazioni*
  - la vista delle informazioni è interessata alle macro-entità più importanti che il sistema deve gestire, nonché alle macro-relazioni tra di esse
    - non è progettazione di dettaglio dei dati
  - bisogna concentrarsi sulle entità più significative per le parti interessate, con una struttura interna complessa, accedute e/o modificate frequentemente



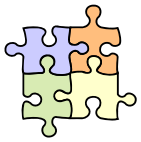
Allocazione delle informazioni

- Nel descrivere contenuto e struttura delle informazioni, due punti di vista rilevanti
  - il punto di vista “globale” – quali sono le informazioni gestite complessivamente dal sistema?
  - il punto di vista dell’allocazione delle informazioni agli elementi architetturali – quali le informazioni gestite da ciascun elemento (funzionale, di concorrenza, di deployment, ...) dell’architettura?
    - si noti che, di solito, le informazioni non vengono semplicemente “partizionate” tra i vari elementi
    - la scelta di come allocare informazioni agli elementi è non banale, e può avere impatto sulle qualità dell’intero sistema



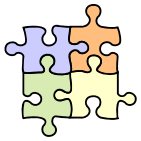
Problemi da identificare e risolvere

- Il fatto che le informazioni gestite complessivamente dal sistema sono in effetti allocate e mosse tra i vari elementi software dell'architettura (funzionali, di concorrenza, di deployment, ...) pone una serie di problemi di gestione – che devono essere identificati e risolti
  - quale allocazione delle informazioni? quali movimenti ammessi per le informazioni?
  - quali le strategie di accesso alle e sincronizzazione delle informazioni? quali conseguenze sulle qualità del sistema?



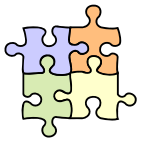
Flussi di dati

- *Flussi di dati*
  - le informazioni mosse tra gli elementi funzionali possono essere importanti tanto quanto quelle memorizzate dai singoli elementi
  - domande importanti
    - dove vengono creati i dati?
    - dove vengono consumati i dati? da dove sono acceduti e modificati?
    - come vengono cambiati i dati mentre si muovono nel sistema?
  - analisi allo stesso livello di dettaglio di "contenuto e struttura delle informazioni"
  - in alternativa, i flussi di dati possono essere analizzati nel contesto della vista funzionale



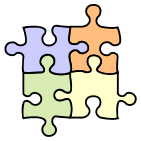
Possesso dei dati (1)

- *Possesso (ownership, proprietà) dei dati*
  - i dati sono spesso distribuiti in più elementi/data store
    - ad es., i dati possono essere replicati in più elementi/data store
    - ad es., laddove bisogna integrare dei sistemi esistenti
  - alcune domande importanti (per identificare le problematiche e scegliere le strategie di gestione)
    - qual è la copia più aggiornata di un particolare dato?
    - come sono sincronizzate le diverse copie di un dato?
    - come sono sincronizzate le diverse copie di un dato, se le modifiche possono avvenire su copie diverse? come riconciliare i conflitti?
    - la validazione dei dati avviene in un solo punto o deve essere ripetuta in più elementi?



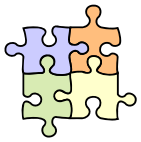
Possesso dei dati (2)

- Un caso (semplice) – ma non sempre possibile
  - ogni dato ha un data owner unico – e possibilmente più data consumer
    - ogni dato ha un data store che ne è il proprietario principale
    - relazioni descritte da un modello di ownership dei dati
  - è richiesta un'interfaccia per far comunicare questi elementi
    - va indicata nella vista funzionale
  - va definito un protocollo che regola il flusso di dati tra questi elementi



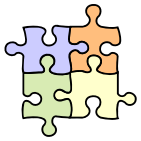
Tempestività, latenza ed età

- *Tempestività (timeliness), latenza ed età (age)*
  - si tratta di qualità importanti soprattutto se possono esistere più copie (repliche) di uno stesso dato – se invece c'è una sola copia, l'accesso è normalmente sincrono alla copia primaria
  - se esistono più repliche di uno stesso dato, bisogna capire
    - se e per quanto tempo possono essere non allineate – *latenza* – il periodo di tempo tra quando un valore viene aggiornato nella sorgente (primaria) per quel dato e quando il valore aggiornato è disponibile a tutte le parti del sistema
    - se è accettabile che un data consumer acceda ad una replica non aggiornata di un dato – *tempestività*
  - inoltre, bisogna capire se e quando scadono certe informazioni – *age*
    - ad es., il tempo in cui informazioni sulle quotazioni in borsa sono utili potrebbe essere di qualche secondo/minuto



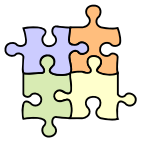
Riferimenti e corrispondenze tra entità

- *Riferimenti e corrispondenze tra entità*
  - spesso è importante poter identificare univocamente i dati nell'ambito dell'intero sistema
    - ad es., mediante chiavi primarie, oid (identificatori d'oggetto) o, più in generale, *riferimenti*
- Alcuni possibili situazioni
  - oggetti in memoria che rappresentano dati in una base di dati
    - quando cambia l'oggetto in memoria, devo aggiornare la base di dati
  - dati presenti in più basi di dati
    - in ciascuna base di dati, potrebbero avere chiavi primarie diverse
    - come capire quando due entità rappresentano lo stesso oggetto del mondo reale?



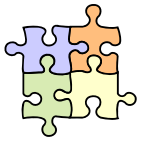
Gestione delle transazioni

- *Gestione delle transazioni*
  - una *transazione* è un'unità di elaborazione atomica, che deve essere completata nella sua interezza oppure non essere eseguita affatto
    - proprietà – atomicità, consistenza (i dati sono lasciati in uno stato consistente), isolamento, persistenza
    - commit e rollback
  - la gestione delle transazioni può riguardare un singolo elemento/database – oppure più elementi/database
  - sono possibili transazioni "lunghe"? come gestirle? con quale impatto (ad es.) sulla latenza?
  - sagas, transazioni annidate e transazioni compensatrici



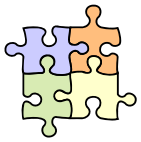
Qualità dei dati

- *Qualità dei dati*
  - una misura di quanto i dati all'interno del sistema corrispondono (o meno) ai dati "veri" nel "mondo reale"
    - ad es., i dati potrebbero essere incompleti oppure non corretti (ad es., non accurati, o non precisi o non aggiornati)
  - particolarmente importante se bisogna integrare o far interoperare più sorgenti informative
  - problemi
    - come valutare la qualità dei dati?
    - quali i livelli di qualità minimi accettabili ?
    - come migliorare la qualità dei dati? automaticamente? manualmente?
    - quale l'effetto di dati di scarsa qualità su altri dati?
  - importante comprendere le qualità effettivamente possedute dai dati nel sistema



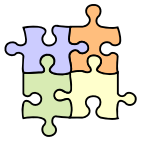
Volumi dei dati

- *Volumi dei dati*
  - quale il volume delle informazioni memorizzate? il volume delle informazioni mosse nel sistema? i tassi di crescita attesi di questi volumi?
  - quale le implicazioni sull'architettura?
    - ad es., se grandi volumi di dati devono essere necessariamente elaborati in periodi di tempo vincolati



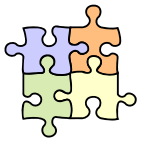
Leggi e regolamenti

- *Leggi e regolamenti*
  - la gestione di alcune informazioni può essere regolata da leggi e regolamenti
    - ad es., privatezza dei dati
  - quali leggi e regolamenti? quale il loro impatto sull'architettura?



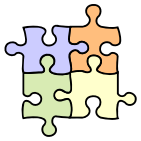
Conservazione dei dati

- *Conservazione (retention) dei dati e degli archivi*
  - per quanto tempo vanno memorizzate le informazioni?
    - può variare da tipologia a tipologia di informazione
  - quali le gerarchie di memorizzazione richieste? come muovere dati tra i diversi livelli? come garantire la conservazione nel tempo?



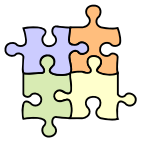
* Modelli

- Tipi di modelli per la vista delle informazioni
  - modelli per la struttura statica dei dati
  - modelli per i flussi di informazioni
  - modelli per il ciclo di vita delle informazioni
  - altri tipi di modelli

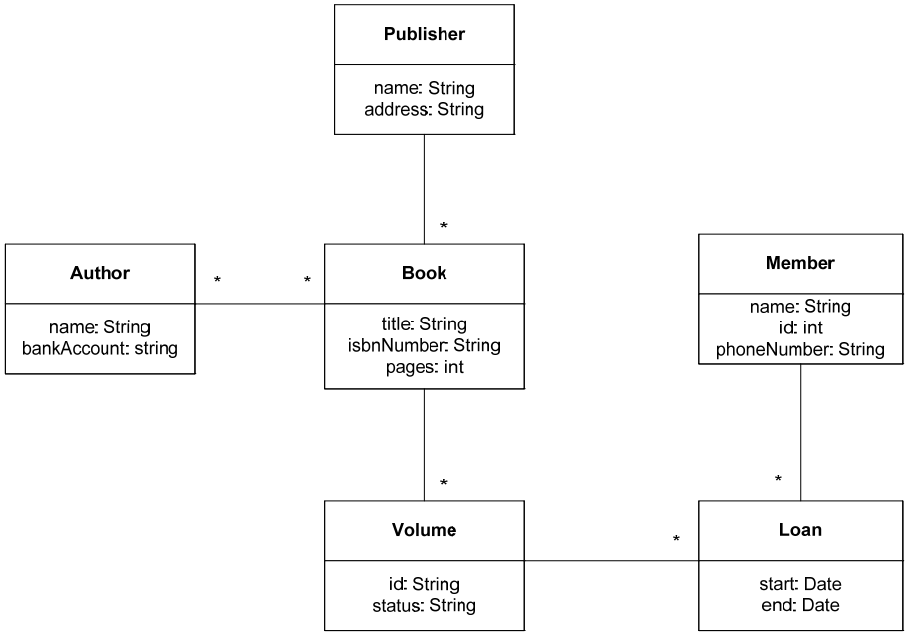


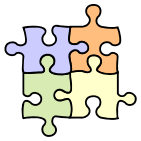
- Modelli per la struttura statica dei dati

- Ad esempio
  - il modello Entity-Relationship
  - diagrammi delle classi di UML
- Utile ragionare a livello di macro-entità e macro-relazioni – architetturalmente significative



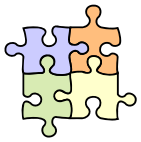
Struttura statica dei dati – esempio (UML)





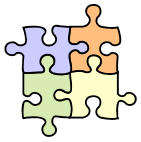
Struttura statica dei dati – esempio

- Alcuni modelli per la struttura statica dei dati utilizzati in UML Components
  - Business Concept Model
    - modello concettuale delle informazioni relative al dominio di business del sistema – può comprendere anche tipi di dati che il sistema non deve gestire
  - Business Type Model
    - modello concettuale delle informazioni che devono essere gestite dal sistema (nel suo complesso)
  - Interface Information Model
    - modello delle informazioni che dovranno essere gestite dal componente che implementerà una certa interfaccia
  - data type
    - un tipo di dato che compare nelle interfacce dei componenti – rappresenta un'unità di flusso di dati tra componenti

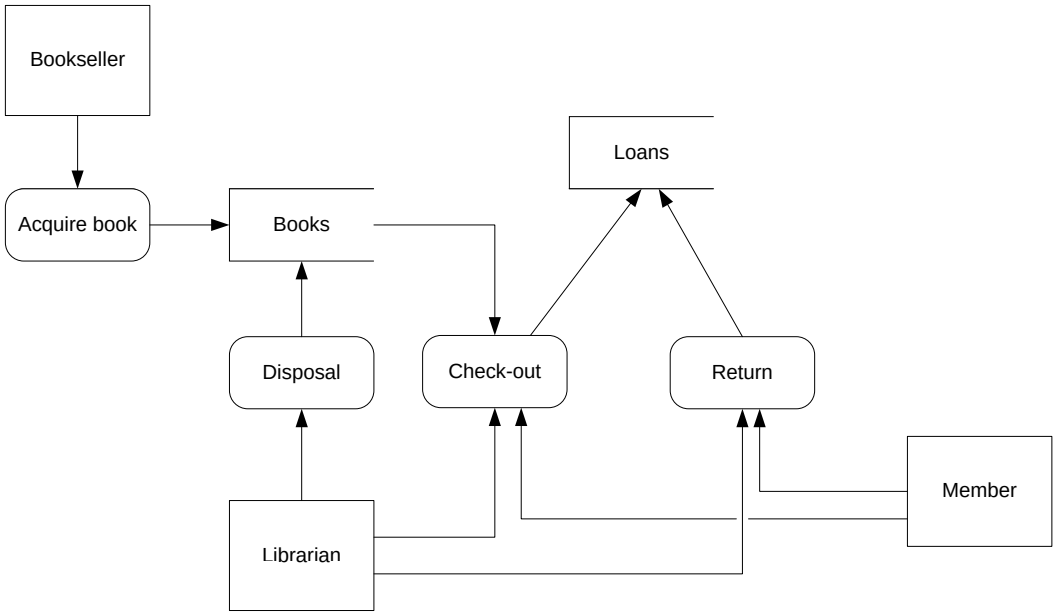


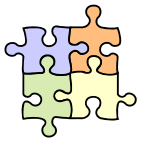
- Modelli per i flussi di informazioni

- I modelli per i flussi di informazioni hanno lo scopo di descrivere/analizzare il movimento dei dati
  - con il mondo esterno al sistema
  - tra gli elementi interni del sistema
- Movimenti di dati descritti da *flussi di dati*
  - con un contenuto/struttura
  - con un verso
  - con un volume e/o una frequenza di trasferimento
- Varie opzioni possibili
  - linguaggi per la descrizione di processi – ad es.,
    - DFD – diagrammi di flussi di dati
    - diagrammi di attività di UML
  - XML e linguaggi associati – ad es., DTD o XML-Schema



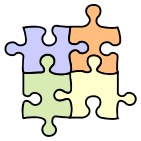
Flussi di dati – esempio (DFD)





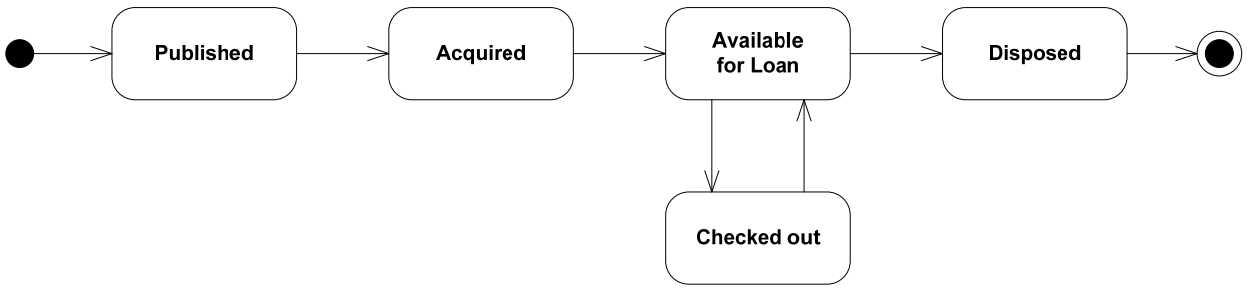
- Modelli per il ciclo di vita delle informazioni

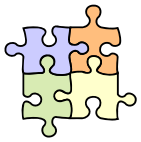
- I modelli per il ciclo di vita delle informazioni descrivono il modo in cui i valori dei dati cambiano (possono cambiare) nel tempo
  - questi modelli enfatizzano i seguenti aspetti
    - quali gli stati in cui può trovarsi un certo tipo di dati?
    - quali le transizioni possibili tra stati?
- Esempio – libri in una biblioteca
  - un libro viene creato quando viene *pubblicato* (almeno dal punto di vista della biblioteca), poi può essere *acquistato* dalla biblioteca, e ripetutamente *prestato* e *restituito*, fino a quando non viene *eliminato*
  - un libro è inizialmente *Pubblicato*, poi può essere *Acquistato* dalla biblioteca, ed essere in alternanza in uno stato di *Disponibile per il prestito* o *Prestato*, fino a quando non è *Eliminato*



Esempio – diagramma di macchina a stati

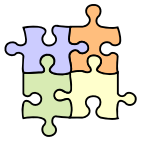
- Il ciclo di vita di molti oggetti/documenti può essere descritto da un automa a stati finiti
  - ad esempio, da un diagramma di macchina a stati di UML





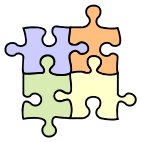
- Modelli di ownership dei dati

- Modelli di ownership dei dati
  - ad es., matrici entità-sistema/elemento funzionale
  - possibili ruoli – un elemento può anche avere più ruoli
    - *owner* o *master* – possiede la copia principale
    - *creator* – crea nuove istanze
    - *updater* – modifica istanze esistenti
    - *deleter* – cancella istanze esistenti
    - *reader* – può leggere ma non modificare istanze esistenti
    - *copy* – detiene una copia read-only di dati
    - *validater* – esegue validazione
  - possibile specificare sotto quali circostanze (ad es., da quali gruppi di utenti) sono permesse certe operazioni
  - consente di identificare i conflitti possibili – in tal caso va stabilita un'opportuna strategia di risoluzione dei conflitti



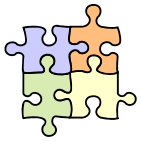
Matrice di ownership dei dati – esempio

Data Item Component	Customer	Product	Order	Fulfillment
Catalog	-	*owner*	-	-
Purchasing	*reader*	*updater*	*owner*	*creator*
Delivery	*copy*	*reader*	*reader*	*updater*
Customer	*owner*	*reader*	*reader*	*reader*



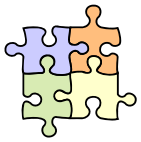
- Analisi di qualità dei dati

- La qualità dei dati può essere descritta/analizzata in termini di
  - *sorgenti* (cause) di dati di scarsa qualità
  - *principi e strategie* per la gestione di dati di scarsa qualità
- Possibili strategie
  - ignora la scarsa qualità dei dati
    - quando il costo della riparazione eccede i benefici che possono essere ottenuti
  - correggi automaticamente i dati di scarsa qualità
    - necessario saper riconoscere automaticamente tali dati
  - scarica i dati di scarsa qualità
  - gestisci manualmente i dati di scarsa qualità
    - l'opzione più costosa



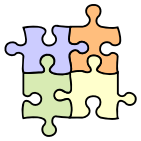
- Metadati

- I *metadati* sono “dati sui dati”
  - i metadati sono le informazioni e i documenti che rendono i dati comprensibili e ne consentono la condivisione tra utenti e sistemi
    - ad es., catalogo della base di dati
- Ad es., metadati in un data warehouse
  - per l’utente
    - quali sono i dati a disposizione? quale il significato di quella dimensione/attributo? quale il periodo temporale dei dati presenti nel data warehouse?
  - per l’area di preparazione dei dati
    - quale la sorgente di dati per una dimensione? quali le trasformazioni da effettuare? quando l’ultimo aggiornamento effettuato?



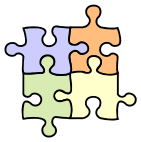
* Problemi e insidie

- Alcuni problemi e insidie nella definizione di una vista delle informazioni
  - incompatibilità dei dati
  - difficoltà nello stabilire corrispondenze tra dati
  - qualità dei dati scarsa
  - updater multipli
  - cattiva latenza delle informazioni
  - complessità delle interfacce
  - gestione non adeguata dei volumi



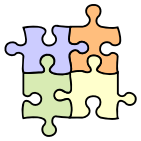
Incompatibilità dei dati

- Incompatibilità dei dati
  - informazioni rappresentate diversamente in sistemi/elementi differenti
    - incompatibilità della codifica
      - ad es., codici, unità di misura, semantica
    - incompatibilità della struttura o dei vincoli di integrità
- Tecniche di riduzione del rischio
  - sviluppa un modello comune per le informazioni gestite dal sistema – conforma i modelli locali agli elementi con il modello comune
  - definisci uno strato software sopra le sorgenti di dati per nascondere le incompatibilità con altre parti del sistema
  - sviluppa un modello comune per le informazioni scambiate nel sistema – garantisci (anche usando degli intermediari) che i dati scambiati tra gli elementi siano conformi a questo modello comune



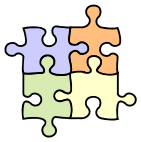
Difficile stabilire corrispondenze tra dati

- Difficoltà nello stabilire corrispondenze tra dati
  - importante poter identificare univocamente i dati nell'ambito dell'intero sistema – quando presenti in parti diverse del sistema
- Tecniche di riduzione del rischio
  - usa gli stessi identificatori per dati presenti in più elementi del sistema
  - se non è possibile, assicurati di poter stabilire le corrispondenze tra elementi – ad es., introducendo degli elementi funzionali appositi



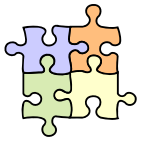
Qualità dei dati scarsa

- Qualità dei dati scarsa
  - possibili difetti nei dati – inconsistenza, incompletezza, mancanza di accuratezza, ...
  - è più grave una qualità dei dati che è *inaspettatamente* scarsa
- Tecniche di riduzione del rischio
  - fissa una qualità minima per i dati – e garantiscila
  - concentrati sulla qualità dei dati più importanti (per le parti interessate)
  - usa strumenti per la qualità dei dati
  - definisci strategie per la gestione di dati la cui qualità è scarsa



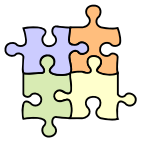
Updater multipli

- Updater multipli
  - non sempre è possibile garantire che i dati vengano aggiornati in uno ed un solo posto
    - ad es., legacy system, sistemi disconnessi, ...
- Tecniche di riduzione del rischio
  - sviluppa un modello di ownership dei dati accurato – e identifica i dati con più updater
  - determina (con le parti interessate) quali updater sono più importanti
  - identifica le possibili inconsistenze che possono verificarsi
  - identifica strategie per gestire queste inconsistenze



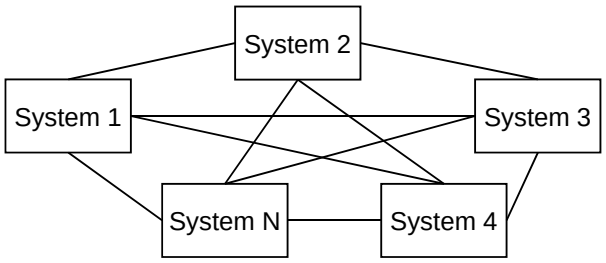
Cattiva latenza delle informazioni

- Cattiva latenza delle informazioni
  - cause della latenza
    - cause interne – architettura troppo complessa o volumi eccessivi
    - cause esterne – dati arrivano al sistema in modo discreto (ad es., settimanalmente), in modo prevedibile o meno (ad es., sistemi disconnessi)
  - è più grave una latenza dei dati che è *inaspettatamente* scarsa
- Tecniche di riduzione del rischio
  - identifica e stima presto la latenza attesa dei dati
  - comprendi (con le parti interessate) se la latenza è un problema
  - raggiungi (con le parti interessate) un accordo ragionevole sui requisiti di latenza

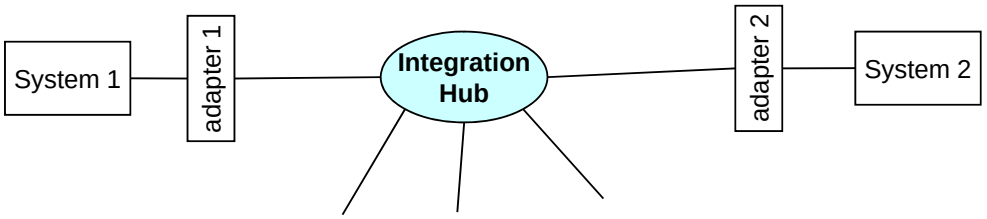


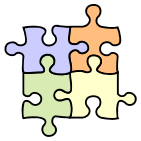
Complessità delle interfacce

- Complessità delle interfacce
  - se N elementi devono scambiare informazioni tra loro, potrebbero essere richieste fino a N(N-1)/2 interfacce



- Tecniche di riduzione del rischio
  - se necessario, usa un *integration hub*





Gestione non adeguata dei volumi

- Gestione non adeguata dei volumi
  - un sistema progettato per gestire migliaia di transazioni al giorno probabilmente non è in grado di gestire milioni di transazioni al giorno
    - utile conoscere il carico atteso del sistema
- Tecniche di riduzione del rischio
  - garantisci che informazioni sui volumi siano raccolte, riviste e approvate dalle parti interessate
  - assicurati che i volumi attesi siano realistici
  - stabilisci corrispondenze tra volumi di business e volumi di dati
    - ad es. quanti scontrini fiscali e voci negli scontrini in un anno?
  - tieni conto di possibili espansioni future
  - convalida (ad es., con dei prototipi) la capacità dell'architettura di gestire i volumi attesi