Sistemi Informativi:

Il processo software

Attivita' di sviluppo software

Requirements Analysis — **Qual e' il problema?**

System Design — **Qual e' la soluzione?**

Problem Domain

Program Design — **Quali meccanismi implementano meglio la soluzione?**

Program Implementation — **Com'e' costruita La soluzione?**

Implementation Domain

Testing — **E' risolto il problema?**

Delivery — **La soluzione e' usabile dall'utente?**

Maintenance — **Occorrono migliorie?**

Software di supporto al progetto (Non superare le dosi raccomandate)

Requisiti: Repository

Progetto: UML

Sviluppo,Testing : IDE

Integrazione: ?

Configurazione,errori,

Manutenzione: Repository

Gestione: vari tools

Integrated Development Environment (IDE)

Software che assiste lo sviluppatore:

Curva di apprendimento meno ripida dovuta alla GUI /funzionalita' visibile

Minor numero di cambi di contesto per l'esecuzione dei diversi task

Standardizzazione metodologia di sviluppo

Aumenta la utilizzabilita' di alcuni linguaggi limitando l'apprendimento della sintassi

Integrated Development Environment (IDE)

Svantaggi:

Se una funzionalita' non e' prevista o ben progettata operazioni banali diventano difficili

- (occorre capire quello che prima si era felicemente ignorato)

Spesso semplici "utility" esterne sono incompatibili col modo di funzionamento dell' IDE

Apprendimento oneroso (es Eclipse),

A volte, costosa migrazione applicazioni tra i vari IDE's

Integrated Development Environment (IDE)

Fattori di successo:

- Automazione di task preesistenti
- Automazione di tecnologie standard
- Non imporre altro che standard accettati
- Rinunciare a fare tutto

Funzionalita’ di un IDE

Editing del codice sorgente

Compilazione del sorgente

Evidenziare la sintassi del linguaggio

Code-completion

Esame della struttura delle classi

Funzionalita’ di un IDE

Formattazione del codice

Accesso alla documentazione

Controllo interattivo degli errori

Creazione facilitata di classi etc.

Integrazione con repositories

Creazione dello scheletro di applicazioni / integrazione

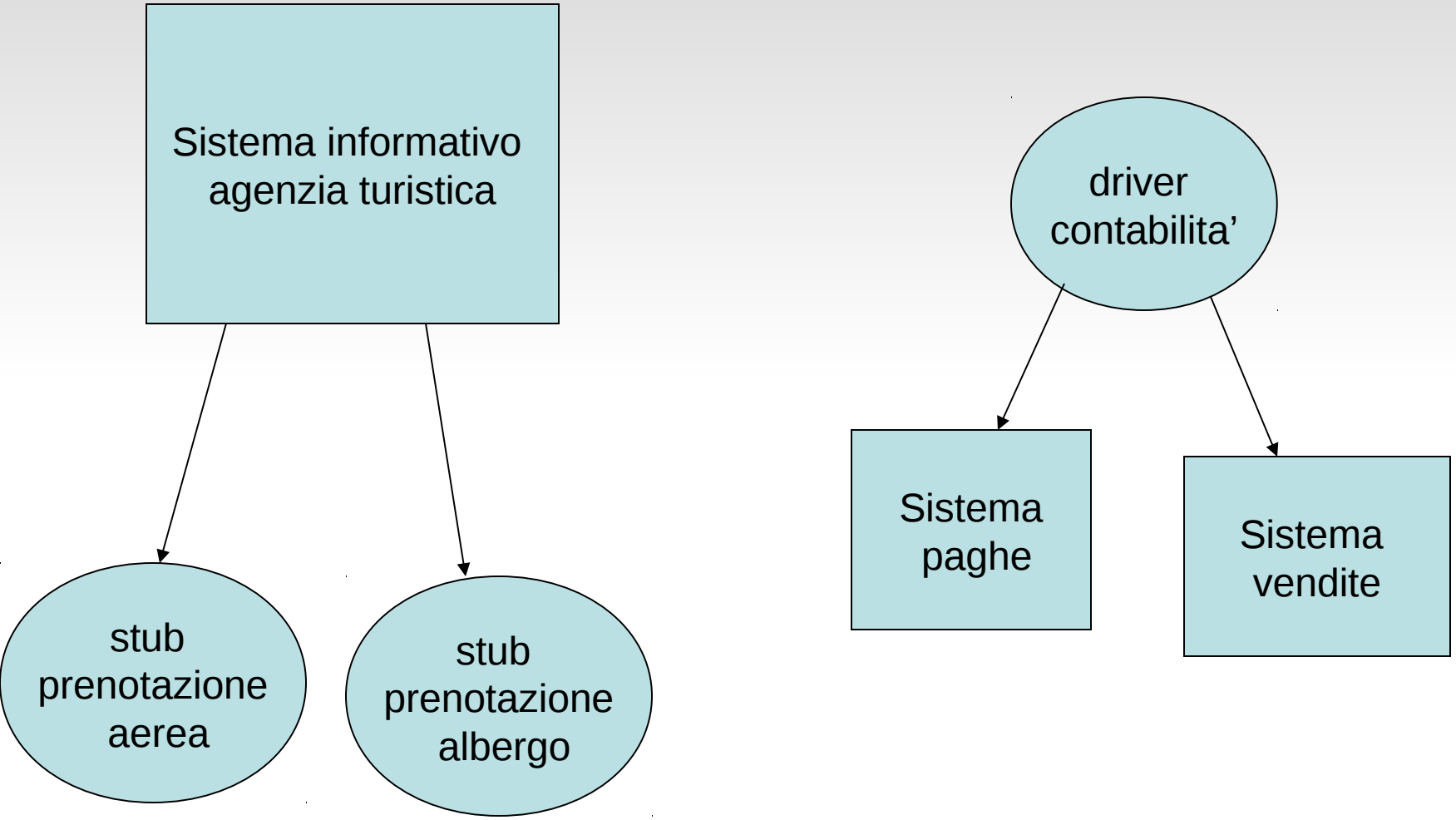
Profiling

Altre fasi: Integrazione, testing, manutenzione

Problemi dell' integrazione:

- Incrementalita’
- Interdipendenza tra moduli richiede:
  - Stubs
  - Drivers
- Progettare il sistema per una facile integrazione
- Distribuzione della funzionalita’ nei sistemi Object-Oriented

Integrazione: stub e driver



Integrazione, testing, manutenzione

Il testing:

- Dura tutto il ciclo di vita/ precoce rilevazione errori
- Piano di testing / lista dei test
- Collegamento tra casi d'uso e relativi test
- Gruppo indipendente di controllo qualita' (SQA)
- Tipi di test:
  - Esame del codice
  - Esecuzione del codice:
    - Test dei cammini del codice (white box)
    - Test rispetto alle specifiche (black box)
- Test di regressione
- Strumenti per simulare l'interazione dell' utente

Integrazione, testing, manutenzione

Tipi di manutenzione:

- Routine
- Adattamento
- Modifica

Fine del ciclo di vita:

- Le modifiche non bastano piu’
- Le modifiche hanno effetti imprevedibili
- Mancanza di documentazione/turnover
- Sostituzione della piattaforma

Prototipo e sviluppo rapido

"Rapid Prototyping" non e' un buon termine perche' confonde *prototipazione con "sviluppo rapido"*

- Il Prototipo e' un aspetto tecnico: e' un particolare modello nel ciclo di vita del progetto
- "sviluppo rapido e' una questione di management: un modo particolare di gestire il progetto.

Se non ristretta, la prototipazione puo' proseguire indefinitamente

- Prototipazione "Time-boxed"

Versione, Revisione, Release

Versione:

- E' una release iniziale o successiva di un oggetto della configurazione associato ad una completa ricompilazione. Versioni differenti hanno funzionalita' differenti.

Revisione:

- Cambio di una versione che corregge errori nel codice o progetto ma non modifica le funzionalita' documentate

Release:

- Quando una versione approvata viene formalmente distribuita

Approcci allo sviluppo software

Inizialmente:

- Programmi procedurali batch ad esecuzione predeterminata, funzioni semplici
- Linguaggi poco sofisticati
- Limitata o nulla interazione con l'utente (modo **batch**)
- *Dissociazione tra dati e funzioni*
- **Approccio strutturato**

In seguito:

- Programmi interattivi reagiscono ad eventi
- Interagiscono con altri sistemi
- Utente in controllo
  - *Associazione tra dati e funzioni*
- **Approccio Object Oriented**

Approccio Strutturato

Modelli usati:

Diagrammi di flusso dei dati (DFD),parzialmente simile a diagramma attivita' e workflow

Disegna dati e processi, le frecce mostrano il flusso dei dati

Diagramma entita’ - relazioni (ERD)

Il modello relazionale, no processi

Problemi:

Sequenziale

Poco flessibile/estendibile

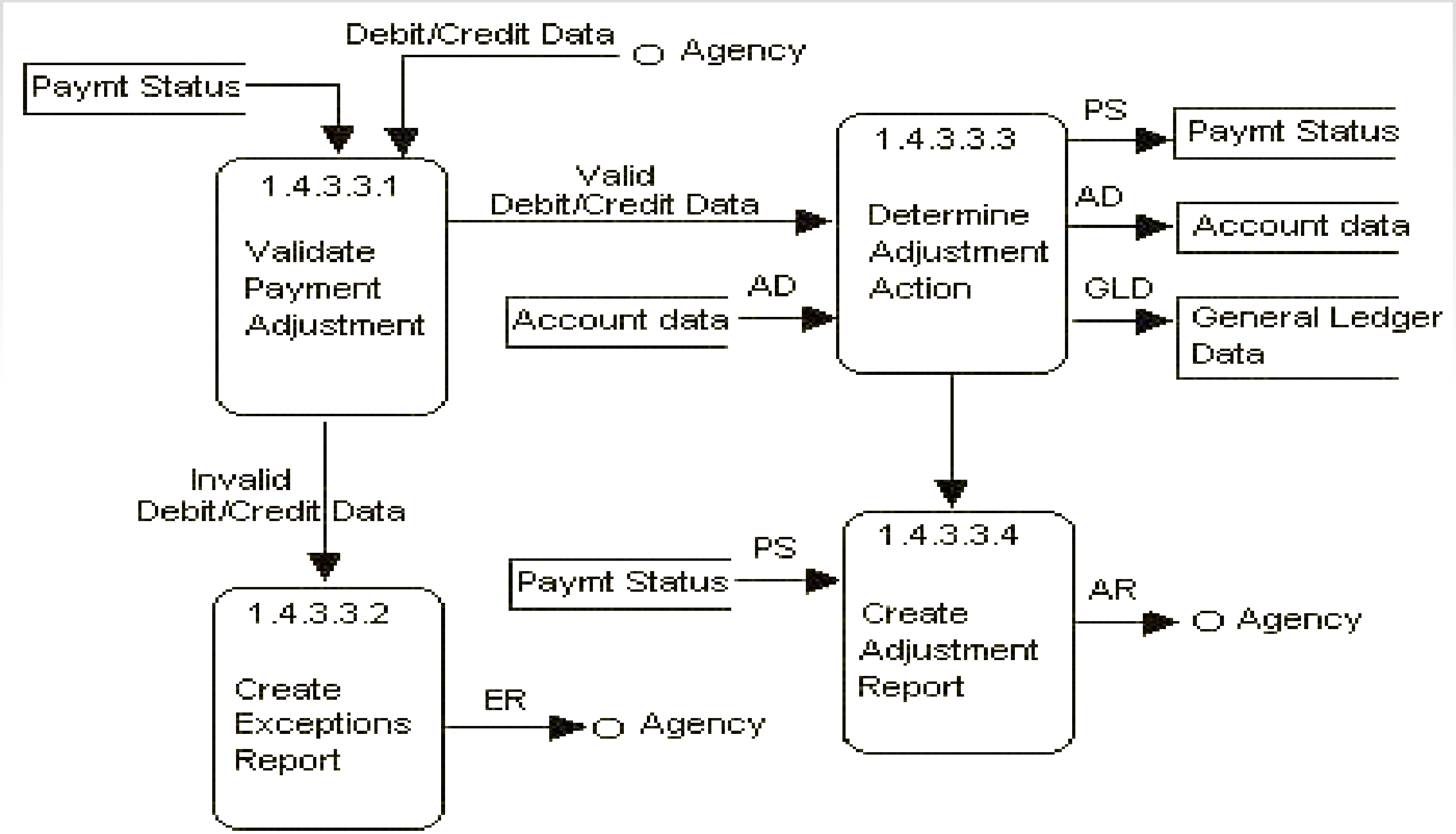
Poco riutilizzo (componenti elementari)

I dati sono potenzialmente modificati in molti punti

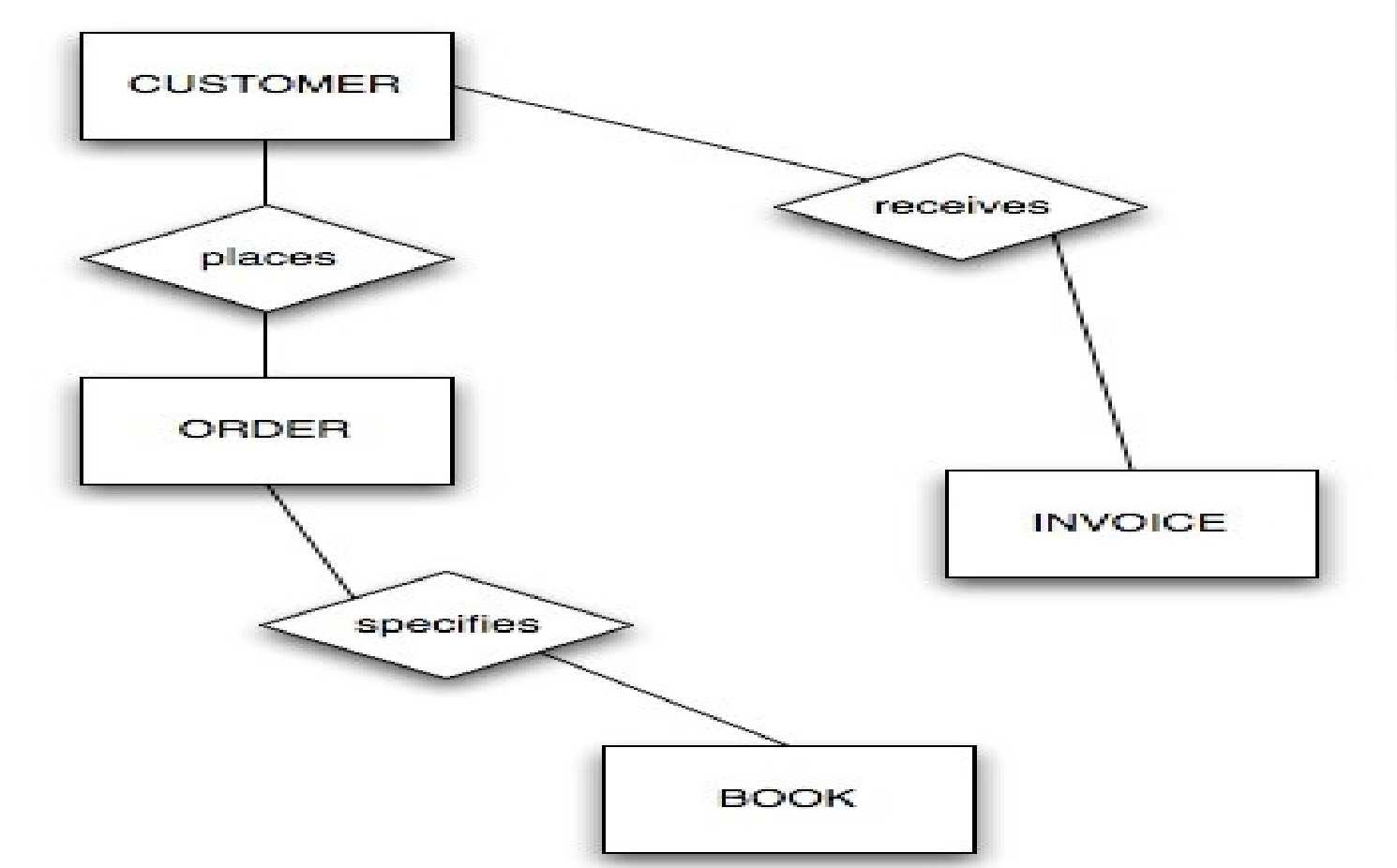
Soprattutto poca varieta' di diagrammi

Notare come la modellazione dei dati sia piu' facile che quella della dinamica

DFD


Debit/Credit Data
Agency
Paymt Status
1.4.3.3.1
Validate Payment Adjustment
Valid Debit/Credit Data
1.4.3.3.3
Determine Adjustment Action
PS
Paymt Status
AD
Account data
AD
Account data
GLD
General Ledger Data
Invalid Debit/Credit Data
1.4.3.3.4
Create Adjustment Report
PS
Paymt Status
AR
Agency
1.4.3.3.2
Create Exceptions Report
ER
Agency

ERD diagram



Approccio Object-Oriented

Un oggetto associa dati e operazioni

Processo iterativo/incrementale

Adatto alle applicazioni attuali

Problemi:

- Difficolta' di traduzione da una modellazione semanticamente ricca al modello relazionale
- Piu' difficile misurare il progresso di un progetto per la maggior fluidita' -> frequenti release
- Permette soluzioni complesse quindi di gestione difficile

Le misurazioni

Parte del progetto e della gestione del processo,

avanzamento lavori,

capacita' della ditta

Misurare passato per pianificare futuro

Misure sul SW :

- Complessita': numero classi, metodi..
- Dimensione codice

Misure del progresso del progetto:

- Volatilita' requisiti prima/dopo documentazione
- Dimensione della documentazione
- Numero di test necessari
- …

Attributi (metriche) di un prodotto software

- Correttezza
  - Fare quello che i suoi requisiti richiedono..
- Mantenibilità
  - Facilità di far evolvere il software per andare incontro ad esigenze (requisiti) che possono cambiare.
  - MTTC (mean time to change).
- Integrità
  - Abilità a resistere ad attacchi (accidentali o intenzionali).
  - Il software non deve causare un danno fisico od economico in caso di attacco/fallimento.

Attributi (metriche) di un prodotto software

- Efficienza
  - Il software non deve sprecare risorse del sistema
- Usabilità
  - Il software dovrebbe avere una interfaccia utente appropriata (che dipende dalle capacità richieste) ed essere ben documentato.

Prodotti Software

– due categorie:
  - Prodotti generici (Off The Shelf). Sistemi che sono prodotti da una azienda di sviluppo software e venduti ad un qualsiasi cliente (MSWord)
  - Prodotti customizzati. Sistemi che sono commissionati da uno specifico committente e sviluppati in maniera specifica

Importanza delle caratteristiche di un prodotto

- La relativa importanza delle caratteristiche di un prodotto dipende dal prodotto e dall'ambiente in cui deve essere usato
- In alcuni casi, alcuni attributi dominano
  - In sistemi real-time in cui è critica la sicurezza (es. Controllo traffico aereo) gli attributi chiave sono l'integrità e l'efficienza
- Costo cresce in maniera molto elevata se sono richiesti livelli molto alti di un particolare attributo