Sistemi Informativi:

Web Services

argomenti

- Cambio di prospettiva rispetto UML
- Evoluzione delle architetture SW
- Componentizzazione delle applicazioni
- Integrazione delle applicazioni
- Definizione di standard per l'integrazione tra cui WS
- Problemi

Strati di un sistema informativo

- 1 - Client
- 2 - Presentation layer
- 3 - Application logic layer
- 4 - Resource management layer

Approccio top-down allo sviluppo

- possibile in modo integrale per progetti che partono da scratch (sempre piu' raro)
  1. Canali di accesso e piattaforme cliente
  2. Formati di presentazione e protocolli cliente
  3. Definizione funzionalita' richiesta dal livello 2
  4. Sorgenti dati necessari alla logica applicazione

Approccio bottom-up

- Precedenza all' analisi dei legacy systems preesistenti per capire costo e fattibilita' di ricavare le informazioni necessarie attraverso wrappers (pattern adattatore), anziche' alla application logic
- L'ordine e' 1,4,3,2
- Il problema principale non e' tanto il disegno dei diagrammi delle classi (usando reverse engineering) quanto l'**integrazione** di componenti

Storia: architettura di un IS: 1-tier

- Applicazione monolitica su mainframe
- Interfaccia a dumb terminal
- No API's: interfaccia tramite cattura dati su terminale
- Nessun problema di incompatibilita'
- Massima efficienza
- Sviluppo non strutturato causa mancanza di vincoli interni e pochi esterni

Architettura di un IS: 2-tier

- Nato all'apparizione del PC (~1980 IBM)
- Lo strato di presentazione viene trasferito su di esso (grazie alle capacita' di elaborazione)
- Libera risorse nel server
- Possibili strati di presentazione diversi ed indipendenti per le varie utenze

Architettura di un IS: 2-tier

- Anche detta client-server
- Thin-clients con funzionalita’ minima
- Fat clients molto complessi
  - Presentazione sofisticata
  - Nel frattempo reti piu’ veloci
- Applicazione non piu’ monolitica
- Dialogo tramite RPC ed API: possibilita’ di implementare clienti disparati a patto di rispettare le API

Remote Procedure Call (RMI se OO)

- scambio di messaggi tra 2 macchine virtuali
- realizza indipendenza dalla macchina di esecuzione
- i parametri sono copie fisiche e non referenze

Remote Procedure Call (RMI se OO)

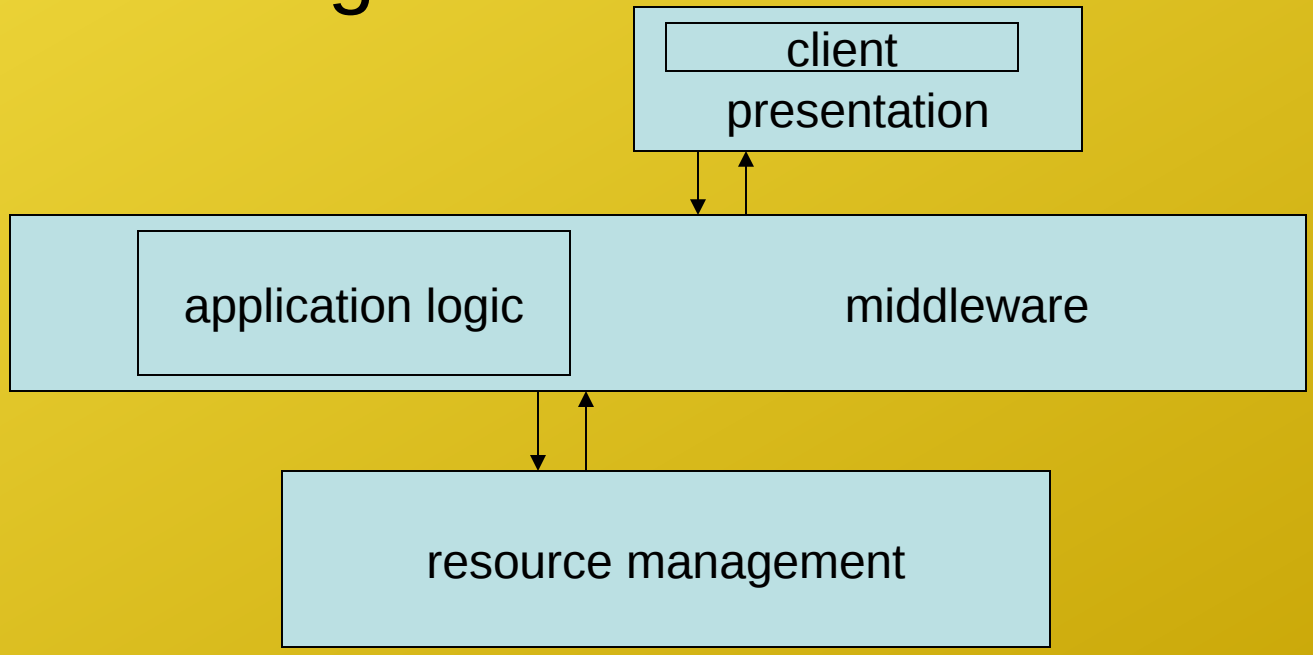
- il server si mette in ascolto ad un indirizzo ben conosciuto (eventualmente pubblicato su apposito servizio)
- dal lato client:
  - si intercetta l'invocazione del metodo remoto,
  - serializza i parametri sul canale di trasmissione
  - si inviano come dati generici
- dal lato server
  - i parametri vengono riespansi
  - il metodo locale viene invocato

Architettura di un IS: 2-tier

- Il cliente acquista vita propria
- Incremento di complessita’
- Il cliente e’ usato per accedere a piu’ server e rispettando le rispettive API
- Troppe interfacce: mantenibilita’ sempre decrescente

Architettura di un IS: 3-tier

- Risolve il problema di integrare piu' server creando uno strato indipendente per la application logic



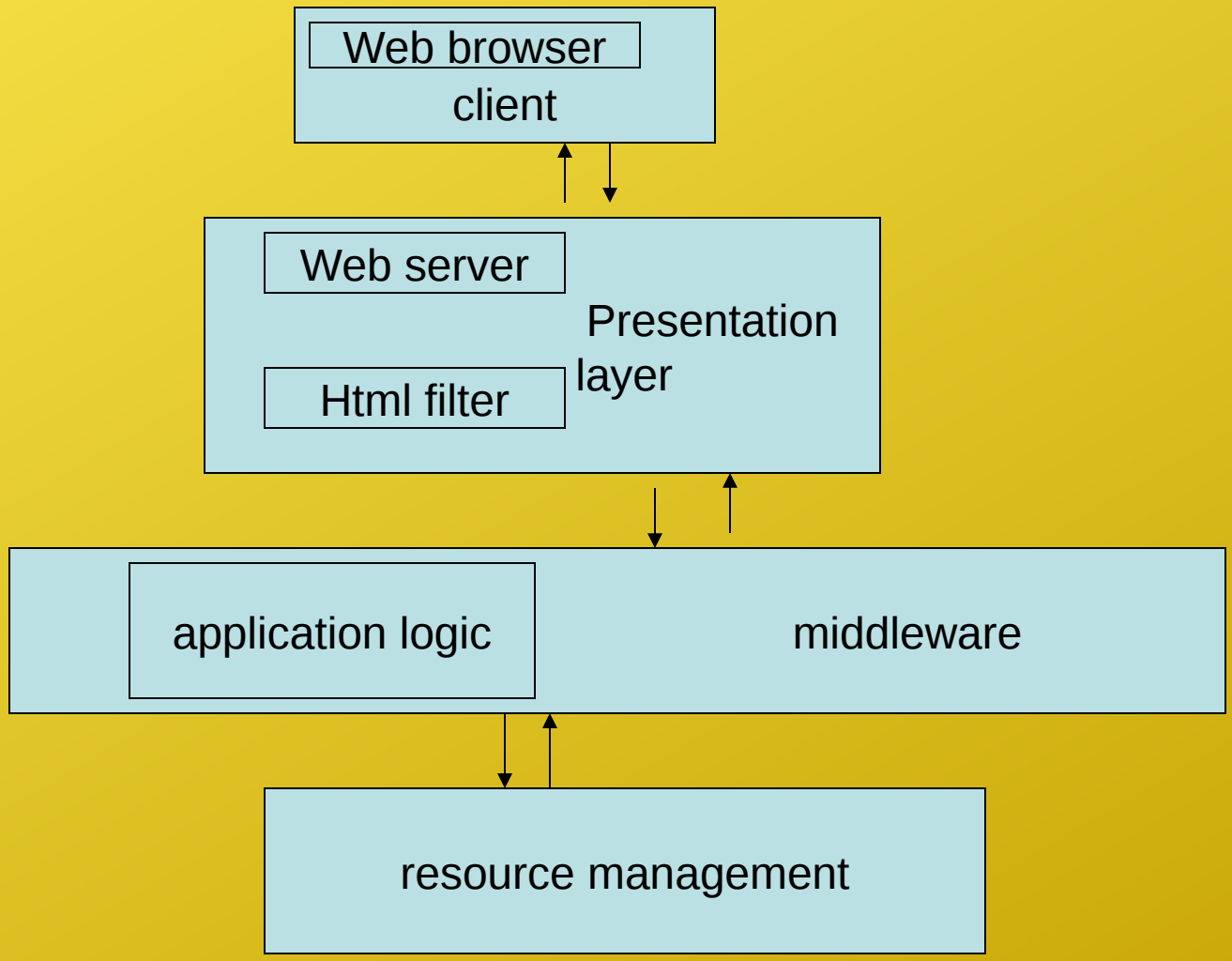
Architettura di un IS: 3-tier

- Definizione di interfacce chiare tra i vari componenti
- Maggior riutilizzabilita’
- Maggior flessibilita’ nella sostituzione e scalabilita’ dei componenti
- Sviluppo del middleware
  - Impacchettamento di funzioni sviluppate N volte

Middleware

- Un insieme di servizi di supporto allo sviluppo della logica delle applicazioni; connette tutti i vari componenti
  - Transazioni
  - Persistenza
  - Autenticazione
  - Logging
  - Replicazione
  - Conversione dati
  - Gestione code e messaggi

Architettura di un IS: N-tier



Evoluzione da 1 a N-tier

- Aumento della flessibilita’
- Aumento dell’ indipendenza
- Aumento della complessita di gestione’
- Aumento dell’overhead di comunicazione
- Reso possibile dall’incremento delle prestazioni HW

Da comunicazione sincrona (pro) …

- Il chiamante manda un messaggio ed aspetta (wait) la risposta
- Nell’intervallo non puo’ fare nulla
- Il sistema e’ piu’ facile da comprendere
- Il sistema e’ piu’ facile da programmare

Comunicazione sincrona (contro)

- Il sistema deve essere tutto on line perche' le chiamate abbiano successo
- Aggiornamento piu' complicato
- Ridotta tolleranza ai guasti

a comunicazione asincrona

- Il chiamante invia un messaggio senza attendere la risposta e prosegue
- Se il chiamato deve rispondere, usa lo stesso metodo

Comunicazione asincrona (pro)

- Maggior efficienza
- Maggior indipendenza dei componenti, facilita' di aggiornamento, fault tolerance
- I sistemi che gestiscono le code implementano funzionalita' prima gestite nelle singole applicazioni

comunicazione asincrona (contro)

- Maggior difficolta’ di programmazione
- I messaggi devono essere ora memorizzati dentro a code (componente aggiuntivo)
- Devono avere un **identificatore** di contesto
- La gestione dello stato di una sessione e' piu' complessa

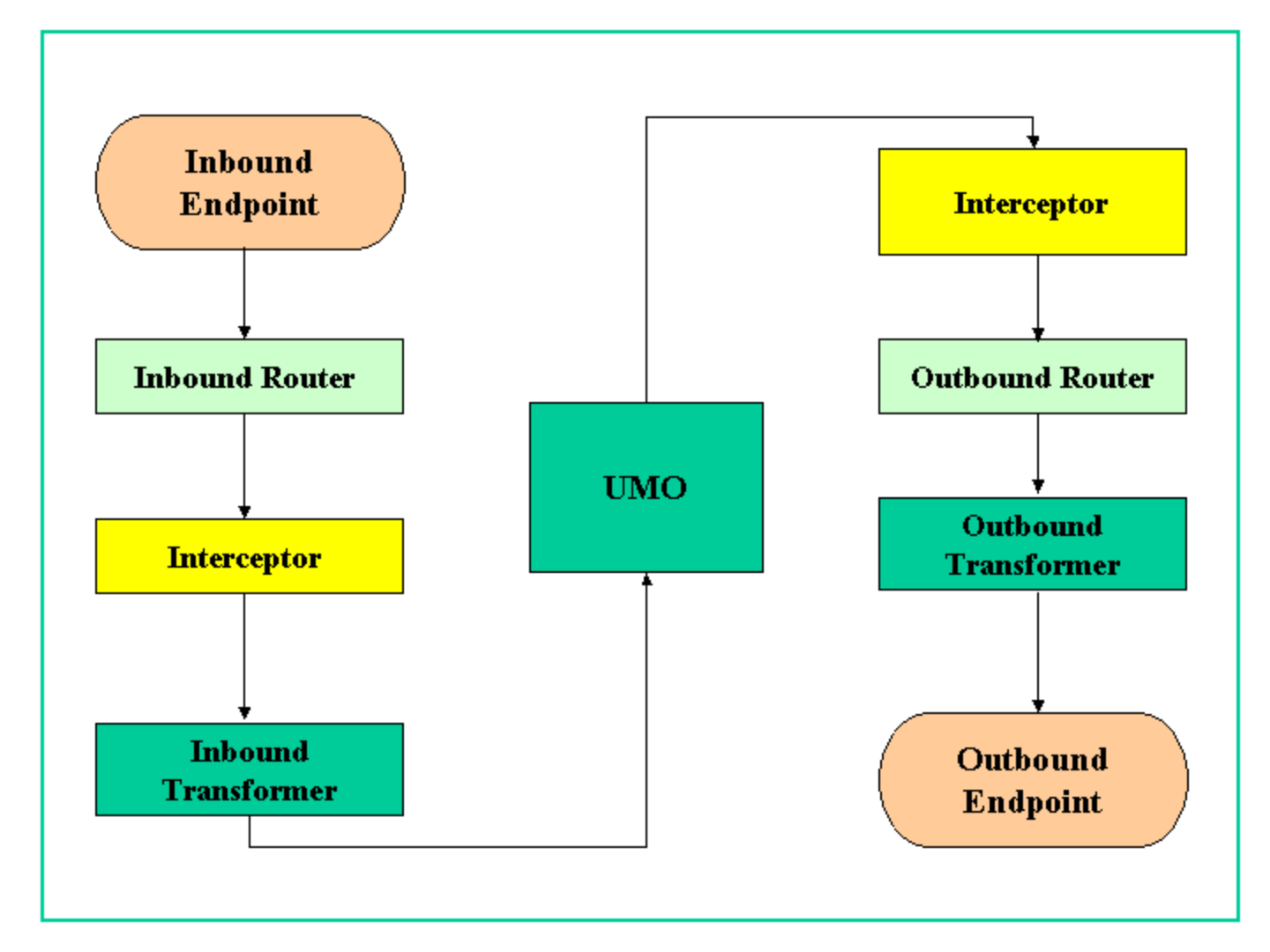
Message Brokers

- Una forma di Message Oriented Middleware
- Puo' trasformare e filtrare i messaggi da una coda all' altra
- Puo' scegliere il destinatario di un messaggio in base al suo contenuto
- Interazione di tipo publish-subscribe
- Affrontano i problemi di Enterprise Application Integration (EAI)

Enterprise Service Bus

- Una forma molto flessibile di MOM, aggiunge:
- Coordinazione di Business process
- Interazione tra i protocolli di comunicazione
- Web Services

Enterprise Service Bus



Enterprise Service Bus (Mule)

- UMO (Universal Messaging Object)
  - Cliente Event-driven che riceve ed alabora un messaggio
- Inbound transformer
  - Componente che trasforma il messaggio dal formato di trasporto a quello del cliente prima dell'invocazione di UMO
- Outbound transformer
  - Componente che trasforma il messaggio dal formato cliente a quello di trasporto collocato tra message router e message endpoint

Enterprise Service Bus (Mule)

- Endpoint
  - Oggetto da configurare per specificare indizzamento informazioni di trasporto, filtri, transazioni...
- Router
  - Controlla il flusso di messaggi da endpoint a UMO e viceversa
- Interceptor
  - Fornisce servizi aggiuntivi: logging, statistiche...

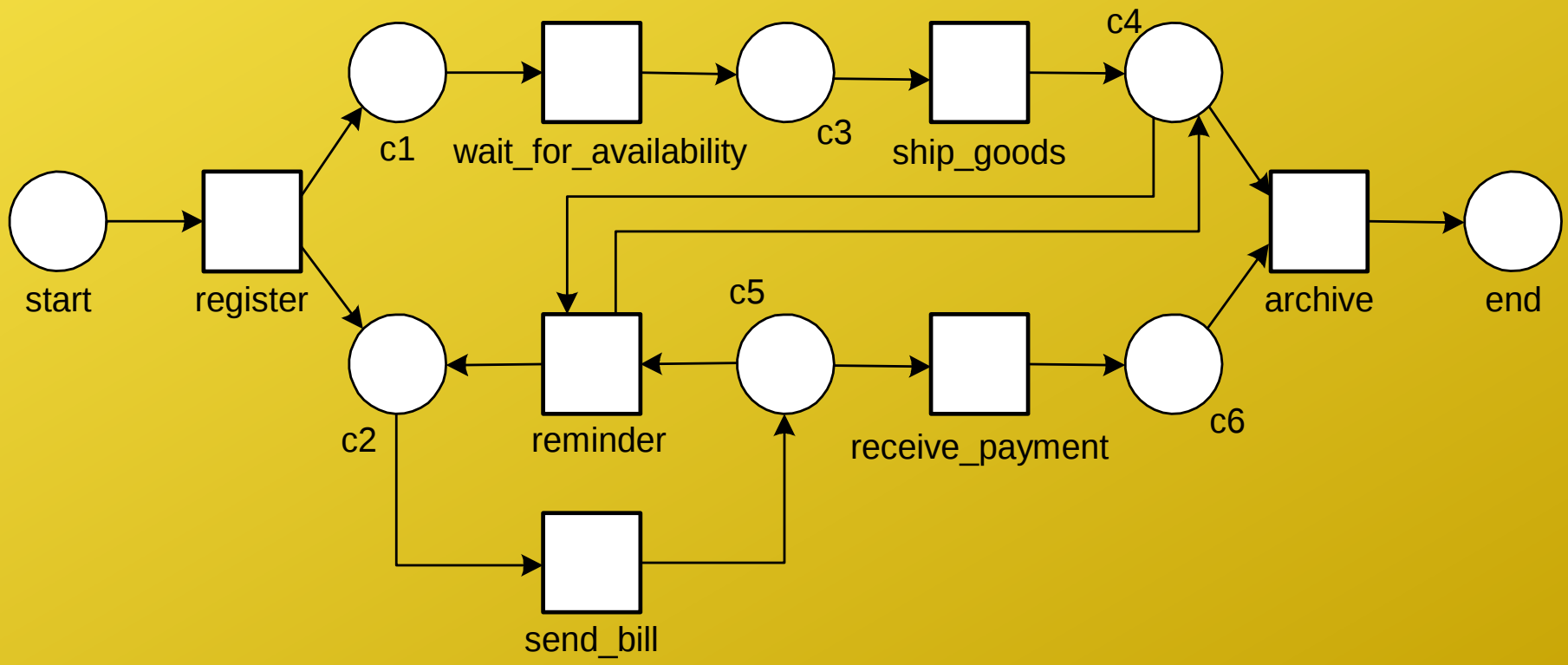
Workflow Management System

- Deriva dall' office automation
- Automatizza il percorso di una procedura aziendale attraverso le varie fasi (es pratica assicurativa)
- Graficamente basato sui **diagrammi di attivita'/reti di Petri/Algebra dei processi**
- Le singole attivita' sono in generale non singoli messaggi , ma intere applicazioni
- Consente permanenza, transazionalita', recupero dai crash

Reti di Petri (in quattro parole)

- sono un modello **teorico** per modellare il funzionamento di un computer
- assomigliano a diagrammi di attivita' con degli stati intermedi che fanno da contenitori dei dati prodotti dalle attivita'
- quando tutti i dati nei contenitori a monte sono pronti, una attivita' inizia e deposita i risultati nei contenitori a valle.
- modellano terminazione,correttezza, raggiungibilita' di stati etc.
- Fanno pubblicare tanto

Example of a process model: A Petri net modeling order processing

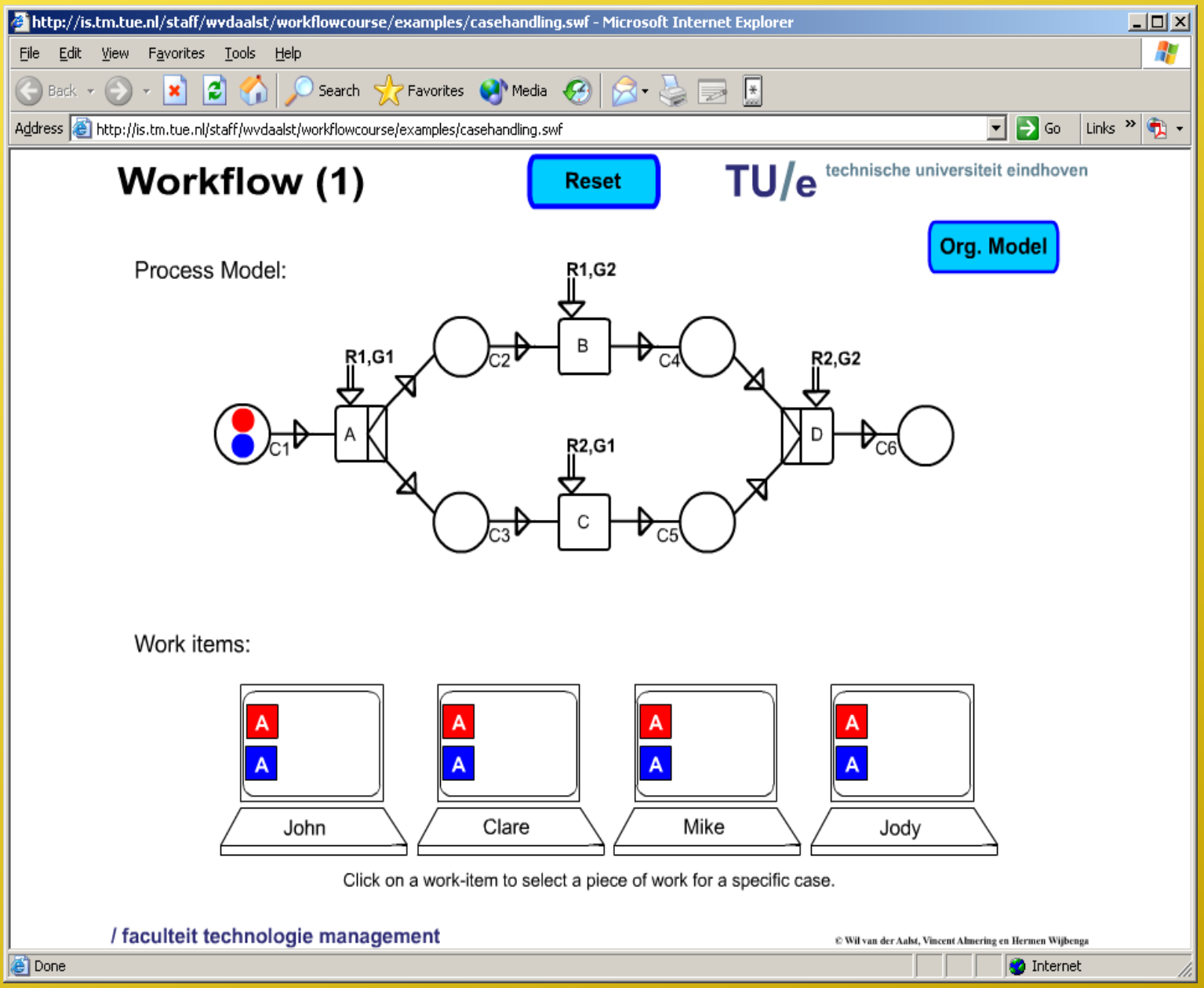


componenti di un workflow

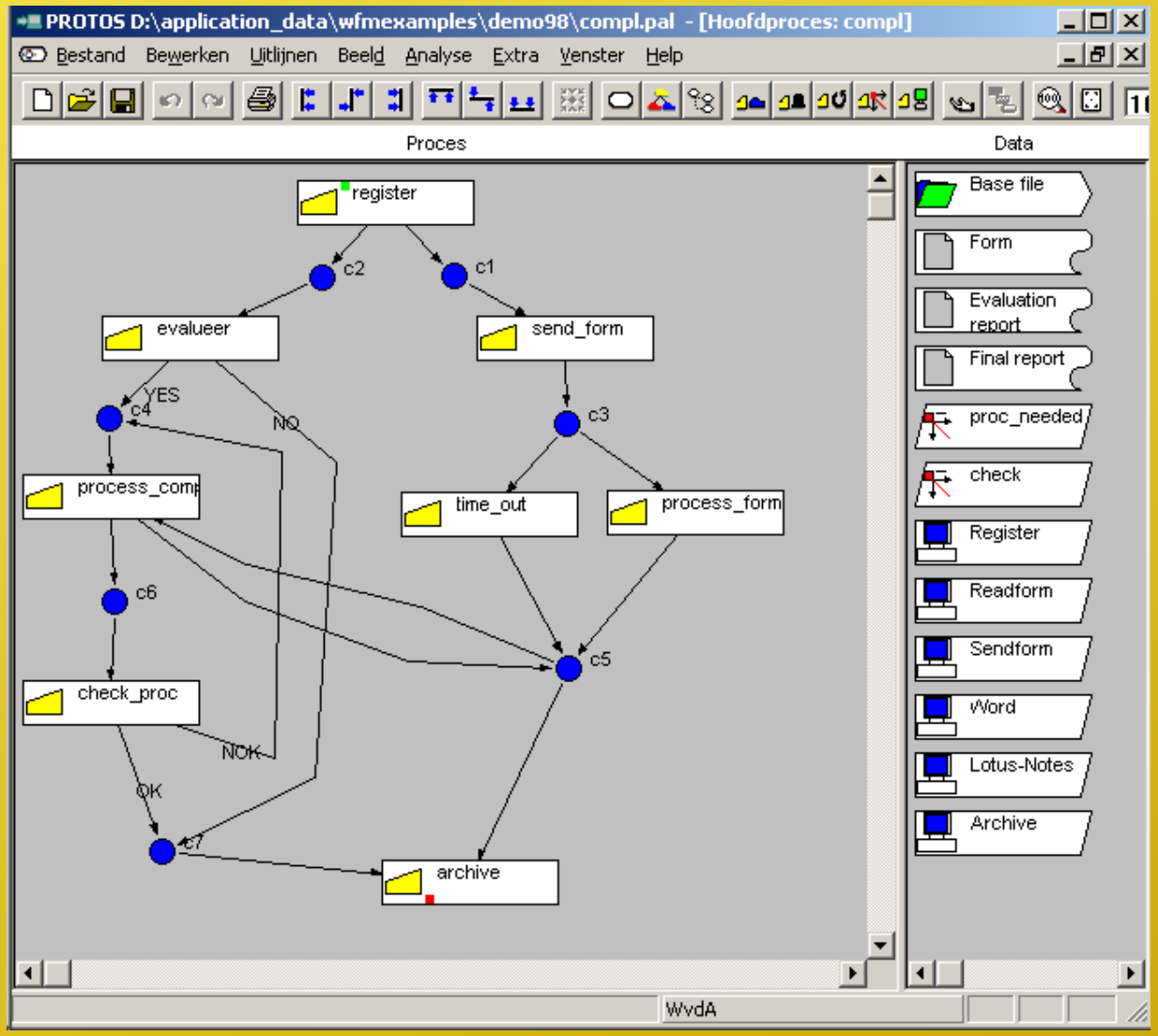
- Un insieme di costrutti di base tipo ciclo, condizione, selezione, fork, join
- dei nodi di attivita'
  - programmi (esecuzione automatica)
  - compiti (eseguiti da persone)
- un DB transazionale per la persistenza (esecuzione lunga)
- code di messaggi, attivita'/task, istanze, gestione ruoli
- un esecutore del WF

problemi dei Workflow Management System

- dipendenza dalle interfacce delle varie applicazioni invocate (procedura inalterata)
- se cambia la procedura (migrazione dei casi)
  - che fare se cambia il workflow, dei casi gia' presenti in esso?
  - possibile (in linea di principio) solo per alcuni tipi di modifiche
  - → vale la pena una soluzione il piu' generale possibile!!


http://is.tm.tue.nl/staff/wvdaalst/workflowcourse/examples/casehandling.swf - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://is.tm.tue.nl/staff/wvdaalst/workflowcourse/examples/casehandling.swf
Go Links
Workflow (1)
Reset
TU/e technische universiteit eindhoven
Org. Model
Process Model:
R1,G2
R1,G1
R2,G2
R2,G1
A
B
C
D
C1
C2
C3
C4
C5
C6
Work items:
A
A
John
Clare
Mike
Jody
Click on a work-item to select a piece of work for a specific case.
/ faculteit technologie management
© Wil van der Aalst, Vincent Ahmering en Hermen Wijbenga
Done
Internet

Process perspective: Protos (extended



Nascita degli application servers

- .net, j2ee (esempio ambiente netbeans)
- Incorporano i servizi del middleware piu' il supporto per interazione col WEB
- Supporto per:
  - Web browsers
  - Applicazioni normali
  - Device
  - E-mail
  - Web services

Quali problemi acquistano importanza?

- Abbiamo sempre piu' supporto per sviluppare una singola applicazione
- Tuttavia nel loro ciclo di vita i sistemi informativi devono poter interagire in modo non previsto a priori con altre applicazioni
- Queste applicazioni sono costruite in ambienti diversi con linguaggi ed interfacce incompatibili

Il problema

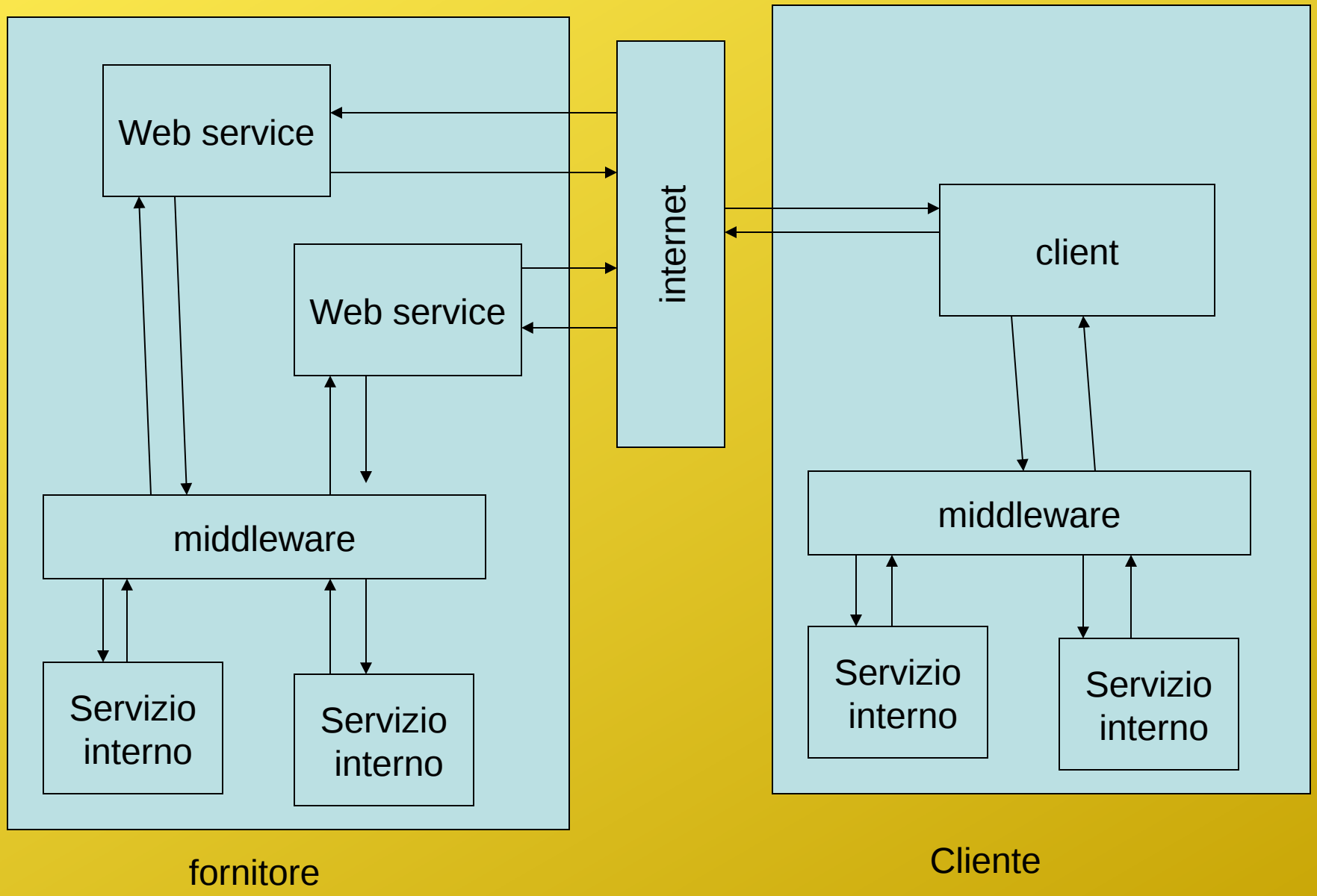
- Risolto relativamente all'interno di un impresa tramite sistemi Enterprise Application Integration
- Tra imprese diverse:
  - Interazioni/Transazioni di lunga durata bloccano risorse troppo a lungo
  - Dove mettere il middleware?
  - Quali API esporre?
  - Che livello di fiducia accordare?

Service Oriented Architecture

- Un servizio e' un programma/procedura con una interfaccia pubblica accessibile da un altro programma esterno
- I servizi sono autonomi, in quanto magari sviluppati/gestiti da compagnie diverse
- Manca un coordinatore centralizzato

Arrivano i Web Service

- Un Web Service e' un programma dotato di una interfaccia scritta in linguaggio WSDL (basato su XML)
- Altri programmi in rete interagiscono con un WS mandandogli messaggi di tipo SOAP (Simple Object Access Protocol)
- SOAP e WSDL possono essere mappati su tutti i linguaggi di programmazione


Web service
Web service
internet
client
middleware
middleware
Servizio interno
Servizio interno
Servizio interno
Servizio interno
fornitore
Cliente

implicazioni

- Occorre adottare degli standard per descrivere il comportamento di una applicazione:
  - Statica (interfacce) (WSDL)
  - Dinamica (flusso) ?
  - Semantica ?
  - Vincoli vari (contrattuali, sicurezza, finanziari..)
- La diffusione di Internet suggerisce di utilizzarla come canale di comunicazione

Statica: WSDL

- WSDL (Web Service Description Language) descrive in modo standard i messaggi che si possono inviare ad una applicazione
  - Basato su XML
  - Un port e' una collezione di messaggi (interfaccia)
  - Si fa riferimento ad un insieme di tipi di base universali comuni tipo intero, carattere etc.
  - Esistono strumenti automatici di conversione dal linguaggio origine a WSDL (e meno male!)
  - Due parti di definizione:
    - Astratta: sempre uguale
    - Concreta : specifica il binding ad una particolare implementazione

Altri standard

- Simple Object Access Protocol (SOAP)
  - Basato su XML, descrive il formato di un messaggio spedito tra applicazioni
- Universal Description Discovery Integration (UDDI)
  - Servizi pubblicati in registries
  - Fornisce API per la pubblicazione e la scoperta di un servizio (facile parallelo col browsing di un agente umano)
- fortunatamente, tramite gli ambienti integrati, questi standard si usano senza doverli conoscere

Frammento WSDL

In java:

interface HelloService {void SayHelloRequest(String firstName);}

In wsdl:

```
<?xml version="1.0" encoding="UTF-8"?>
    <definitions name="HelloService"
     targetNamespace="http://www.ecerami.com/wsdl/HelloService.ws
     dl" xmlns="http://schemas.xmlsoap.org/wsdl/"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <message name="SayHelloRequest">
        <part name="firstName" type="xsd:string"/>
    </message>

……..……
```

La prima parte definisce I “dizionari” , la terminologia del resto del documento

approccio REST

- Ogni risorsa e' un Unique Resource Identifier
- 4 operazioni HTTP: get put post delete
- ogni operazione e' vista come navigazione da una informazione all'altra
- Ogni invocazione di un servizio diventa un accesso ad un URL differente

Dinamica (1)

- Occorre specificare/pubblicare in qualche modo la sequenza corretta di invocazione delle operazioni di un servizio
- op1: login
- op2: browse OR order
- ….
- opN pay
- …

Dinamica (2)

- Se pensiamo ad una collezione di applicazioni che interagiscono possiamo rappresentarle con un workflow / activity diagram/ macchina a stati/ rete di Petri/ sistema a regole…
- Tra le varie proposte :
  - WSCI – specifica la sequenza di scambio messaggi
  - BPEL un ambiente di esecuzione di Web services

Web Service Choreography Interface (WSCI)

- specifica una serie di scambi di messaggi WSDL tramite sequenze elementari
- send, receive, send-receive, receive-send
- in realta' ci sono molti piu' pattern: spedisci a tanti ed attendi solo la prima risposta…..
- a volte gli standard appaiono troppo presto, prima che la soluzione di un problema sia stata analizzata a sufficienza.

Semantica: il problema

- Se dobbiamo scambiare messaggi tra due applicazioni, occorre essere sicuri che il significato delle operazioni sia identico tra i due partecipanti
- Fatto in genere da un tecnico.
- Esempio: un oggetto Dirigente nel sistema A potrebbe essere simile ad un oggetto Manager nel sistema B
- Oppure Dirigente e Manager potrebbero essere assomigliare solo riguardo ad uno specifico insieme di proprieta' (che cosa vi ricorda?)
- Se per I nostri scopi sono sufficienti siamo a posto

Semantica

- Descrive il significato dei messaggi
- I linguaggi utilizzati descrivono tramite predicati logici le proprieta' e le relazioni tra le classi del dominio (Description Logic)
- Possono essere paragonati ad un soprainsieme di UML, dove si specificano proprieta' di esistenza, disgiunzione tra classi ed altre.
- In UML queste proprieta' non sono esplicite, ma realizzate tramite vincoli o algoritmi nella realizzazione del sistema
- Rendere "dichiarative" queste proprieta' permette l'uso di applicazioni che possano "dedurre" l'equivalenza che ci interessa

Semantica

- Class C = union (class A, Class B);
- Individual I1; Individual I2; i1 != I2;
- Relation R; Transitive (R);
- Questi linguaggi (OWL, DAML..) descrivono delle **ontologie**.
- imparentati con la logica del primo ordine, problemi analoghi
- Problema principale :
  - dato un concetto in una ontologia, cercarne uno equivalente in un altra (in modo automatico)

Equivalenza Semantica

- Il problema assomiglia a cercare, dati due class diagram, delle classi (concetti) equivalenti
- Se si usano linguaggi molto potenti il problema puo' essere **indecidibile (**Godel**)**
- La soluzione pragmatica (realistica) consiste in una progressiva standardizzazione delle descrizioni dettagliate dei servizi (Ebxml, RosettaNet)

Vincoli vari

- Se immaginiamo che due applicazioni di due aziende diverse possano interagire per realizzare qualche funzionalita' non banale, occorre descrivere tutto il contesto (legale, finanziario etc) in cui operano
- Esistono proposte di linguaggi (e standard relativi) per descrivere tutti questi aspetti.
- Piu' di 1000 pagine di documenti (e in continua crescita)

Uso intra-aziendale

- WS come come interfaccia standard tra applicazioni
- Incapsulamento di legacy applications
- Utilizzabile subito per la mancanza dei problemi semantici e dei vincoli al contorno

riassumendo

- Le architettture forniscono sempre piu' supporto allo sviluppo di un' applicazione tramite componenti e framework standard
- Questi componenti consentono interoperabilita' tra sistemi incompatibili
- Spesso oggi il successo di un IS consiste nella riuscita integrazione tramite middleware di qualche componente nuovo e di applicazioni preesistenti (legacy)