



# Service oriented software engineering

Ingegneria del Software

# Argomenti

- ~ Architetture orientate ai servizi
- ~ Servizi RESTful
- ~ Ingegneria dei servizi
- ~ Composizione di servizi

# Web service

- ~ Un servizio web è un'istanza della nozione più generale di servizio:
  - "un atto o una prestazione offerta da una parte a un'altra. Sebbene il processo possa essere legato a un prodotto fisico, la prestazione è essenzialmente intangibile e non comporta normalmente la proprietà di alcuno dei fattori di produzione".
- ~ L'essenza di un servizio, quindi, è che la fornitura del servizio è indipendente dall'applicazione che lo utilizza.
- ~ I fornitori di servizi possono sviluppare servizi specializzati e offrirli a una serie di utenti di organizzazioni diverse.

# Servizi riusabili

- ~ I servizi sono componenti riutilizzabili che sono indipendenti (non richiedono un'interfaccia) e che sono accoppiati in modo lasco.
- ~ Un servizio web è:
  - è un componente software riutilizzabile, debolmente accoppiato, che incapsula funzionalità discrete, che possono essere distribuite e a cui si può accedere tramite programma. Un servizio web è un servizio a cui si accede utilizzando protocolli standard basati su Internet e XML.
- ~ I servizi sono indipendenti dalla piattaforma e dal linguaggio di implementazione

# Vantaggi dell'approccio orientato ai servizi

- ~ I servizi possono essere offerti da qualsiasi fornitore di servizi all'interno o all'esterno di un'organizzazione, in modo che le organizzazioni possano creare applicazioni integrando i servizi di una serie di fornitori.
- ~ Il fornitore di servizi rende pubbliche le informazioni sul servizio, in modo che qualsiasi utente autorizzato possa utilizzarlo.
- ~ Le applicazioni possono ritardare il binding dei servizi fino a quando non vengono distribuiti o fino all'esecuzione. Ciò significa che le applicazioni possono essere reattive e adattare il loro funzionamento ai cambiamenti dell'ambiente di esecuzione.

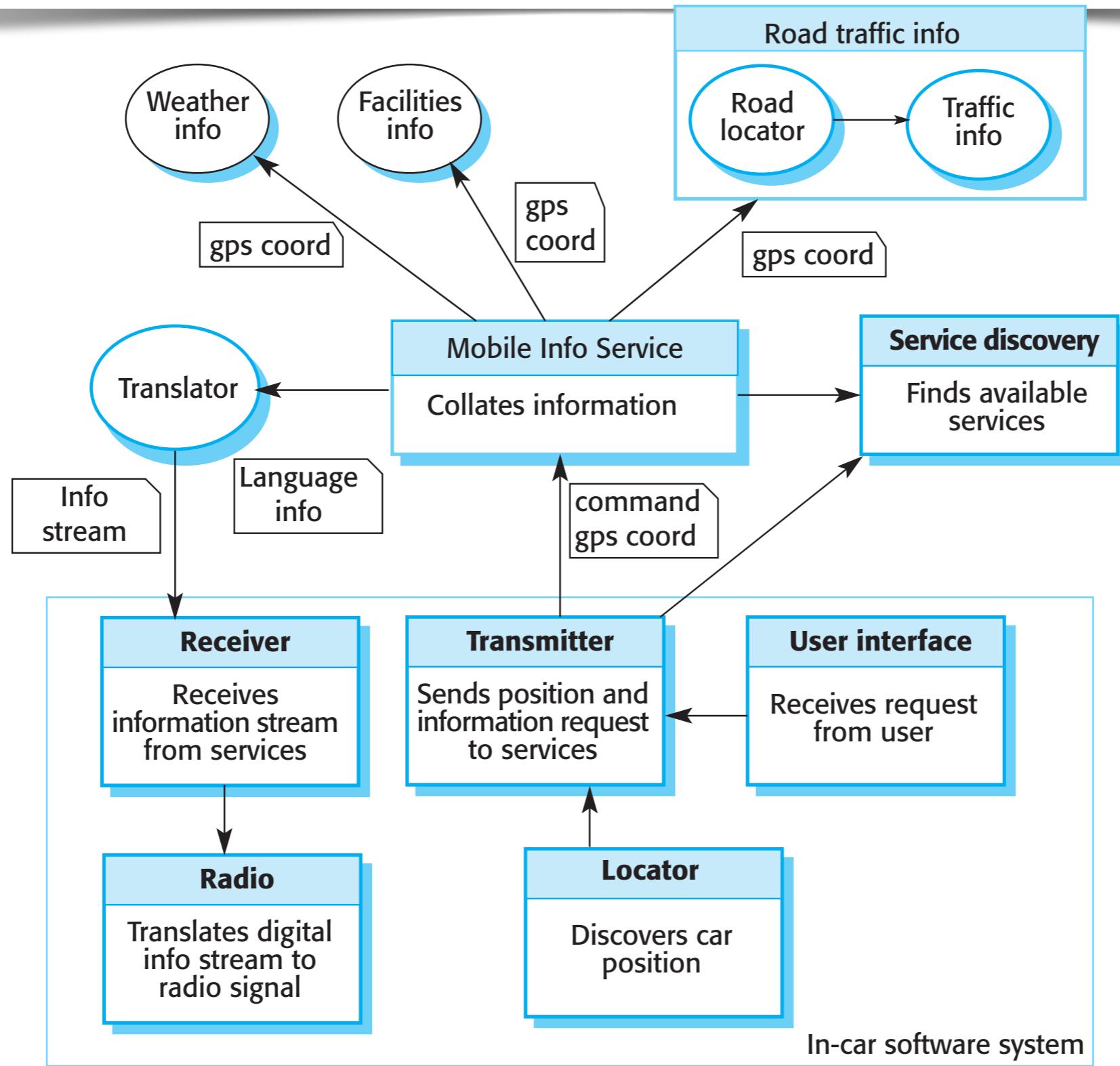
# Vantaggi dell'approccio orientato ai servizi

- ~ È possibile la costruzione opportunistica di nuovi servizi. Un fornitore di servizi può riconoscere i nuovi servizi che possono essere creati **collegando i servizi esistenti** in modo innovativo.
- ~ Gli utenti dei servizi possono **pagare i servizi** in base al loro utilizzo piuttosto che alla loro fornitura. Invece di acquistare un componente usato raramente, gli sviluppatori dell'applicazione possono utilizzare un servizio esterno che verrà pagato solo quando necessario.
- ~ Le applicazioni possono essere ridotte, il che è particolarmente **importante per i dispositivi mobili** con capacità di elaborazione e memoria limitate. L'elaborazione ad alta intensità di calcolo può essere scaricata su servizi esterni.



# Uno scenario

- ~ un sistema informativo per auto fornisce al conducente informazioni sul tempo, sulle condizioni del traffico stradale, sulle informazioni locali, ecc. È collegato al sistema audio dell'auto in modo che le informazioni vengano trasmesse come segnale su un canale specifico.
- ~ L'auto è dotata di un ricevitore GPS per rilevare la sua posizione e, in base a questa, il sistema accede a una serie di servizi informativi. Le informazioni possono essere fornite nella lingua specificata dal conducente.



# I vantaggi di SOA per questa applicazione

~ Tra i vantaggi di questo approccio

- Non è necessario decidere, al momento della programmazione o dell'implementazione del sistema, quale fornitore di servizi debba essere utilizzato o a quali servizi specifici si debba accedere.
- Mentre l'auto si muove, il software di bordo utilizza il servizio di individuazione dei servizi per trovare il servizio di informazione più appropriato e si lega a questo.
- Grazie all'uso di un servizio di traduzione, il software può muoversi oltre i confini nazionali e quindi rendere disponibili le informazioni locali a chi non parla la lingua locale.

# Service oriented software engineering

- ~ E' uno sviluppo significativo come quello orientato agli oggetti.
- ~ La costruzione di applicazioni basate sui servizi consente alle aziende e alle altre organizzazioni di cooperare e di utilizzare le funzioni aziendali degli altri.
- ~ Le applicazioni basate sui servizi possono essere costruite collegando i servizi di vari fornitori utilizzando un linguaggio di programmazione standard o un linguaggio di workflow specializzato.

# Service oriented software engineering

- ~ Inizialmente le compagnie svilupparono i propri standard (come nel caso dei componenti)
  - s. standard differenti e concetto di architettura orientata ai servizi
  - s. gli standard proposti erano complessi e presentavano overhead di esecuzione
- ~ Approccio architettura alternativo
  - s. servizi RESTful

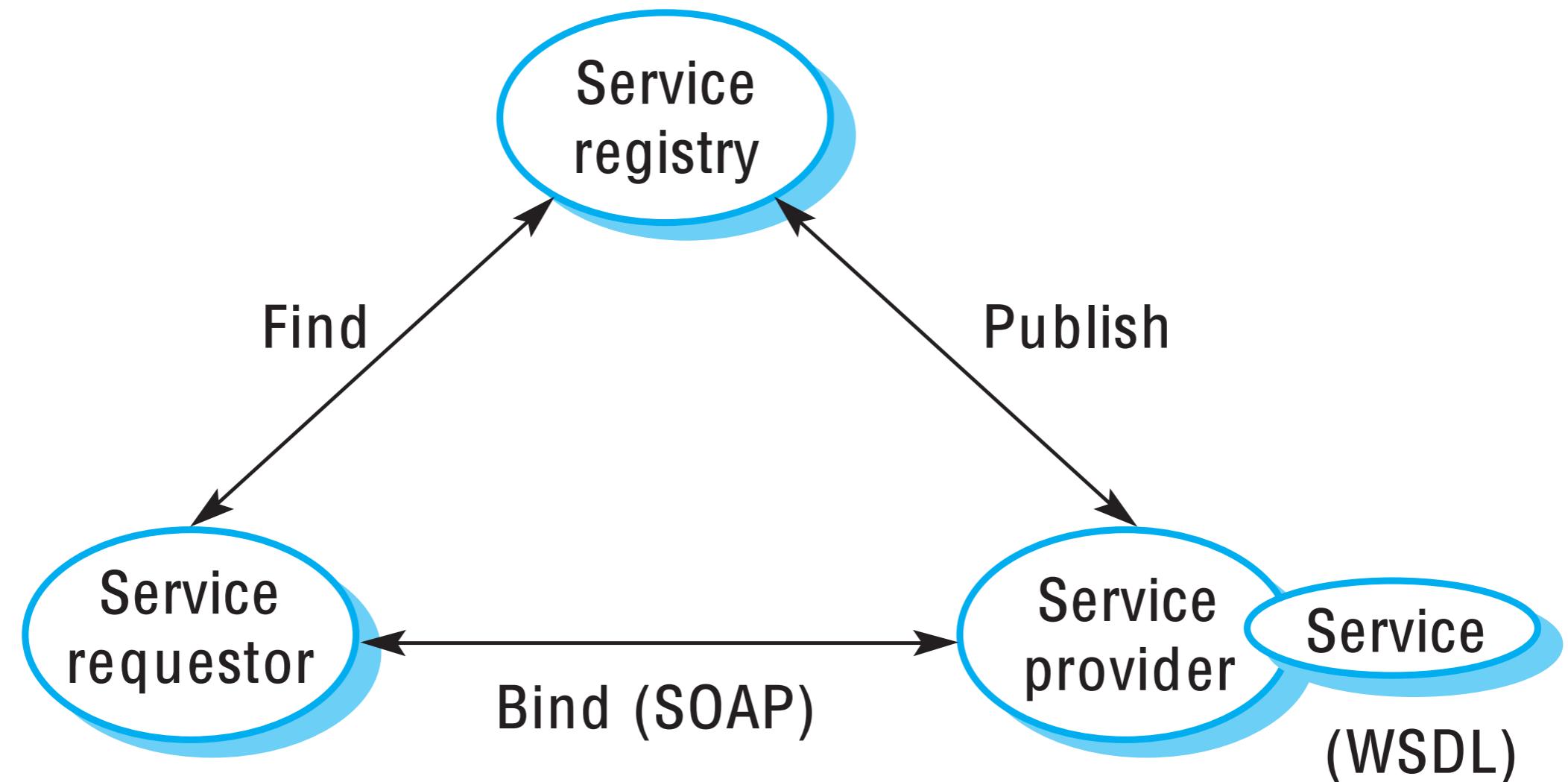


# Service oriented architecture

# Service oriented architecture

- ~ Un mezzo per sviluppare sistemi distribuiti in cui i componenti sono servizi autonomi.
- ~ I servizi possono essere eseguiti su computer diversi da diversi fornitori di servizi.
- ~ I servizi eseguibili possono essere inclusi nelle applicazioni —> le applicazioni posso decidere quale servizio sia più appropriato
- ~ Sono stati sviluppati protocolli standard per supportare la comunicazione dei servizi e lo scambio di informazioni.

# Service oriented architecture





# Vantaggi di SOA

- ~ I servizi possono essere forniti localmente o esternalizzati a fornitori esterni
- ~ I servizi sono indipendenti dal linguaggio
- ~ Gli investimenti nei sistemi preesistenti possono essere preservati
- ~ L'informatica inter-organizzativa è facilitata dallo scambio di informazioni semplificato

# Standard per i servizi web

~ SOAP

- è uno standard di scambio di messaggi che supporta la comunicazione di servizi

~ WSDL (linguaggio di definizione dei servizi web)

- Questo standard consente di definire l'interfaccia di un servizio e i suoi binding.

~ WS-BPEL

- uno standard per i linguaggi di flusso di lavoro utilizzati per definire la composizione dei servizi.

# Standard per i servizi web

XML technologies (XML, XSD, XSLT, ....)

Support (WS-Security, WS-Addressing, ...)

Process (WS-BPEL)

Service definition (UDDI, WSDL)

Messaging (SOAP)

Transport (HTTP, HTTPS, SMTP, ...)

# Service oriented software engineering

- ~ Gli approcci esistenti all'ingegneria del software devono evolversi per riflettere l'approccio orientato ai servizi allo sviluppo del software.
- ~ Ingegneria dei servizi. Lo sviluppo di servizi affidabili e riutilizzabili.
- ~ Sviluppo di software per il riutilizzo
- ~ Sviluppo di software con servizi. Lo sviluppo di software affidabile in cui i servizi sono i componenti fondamentali.
- ~ Sviluppo di software con riuso

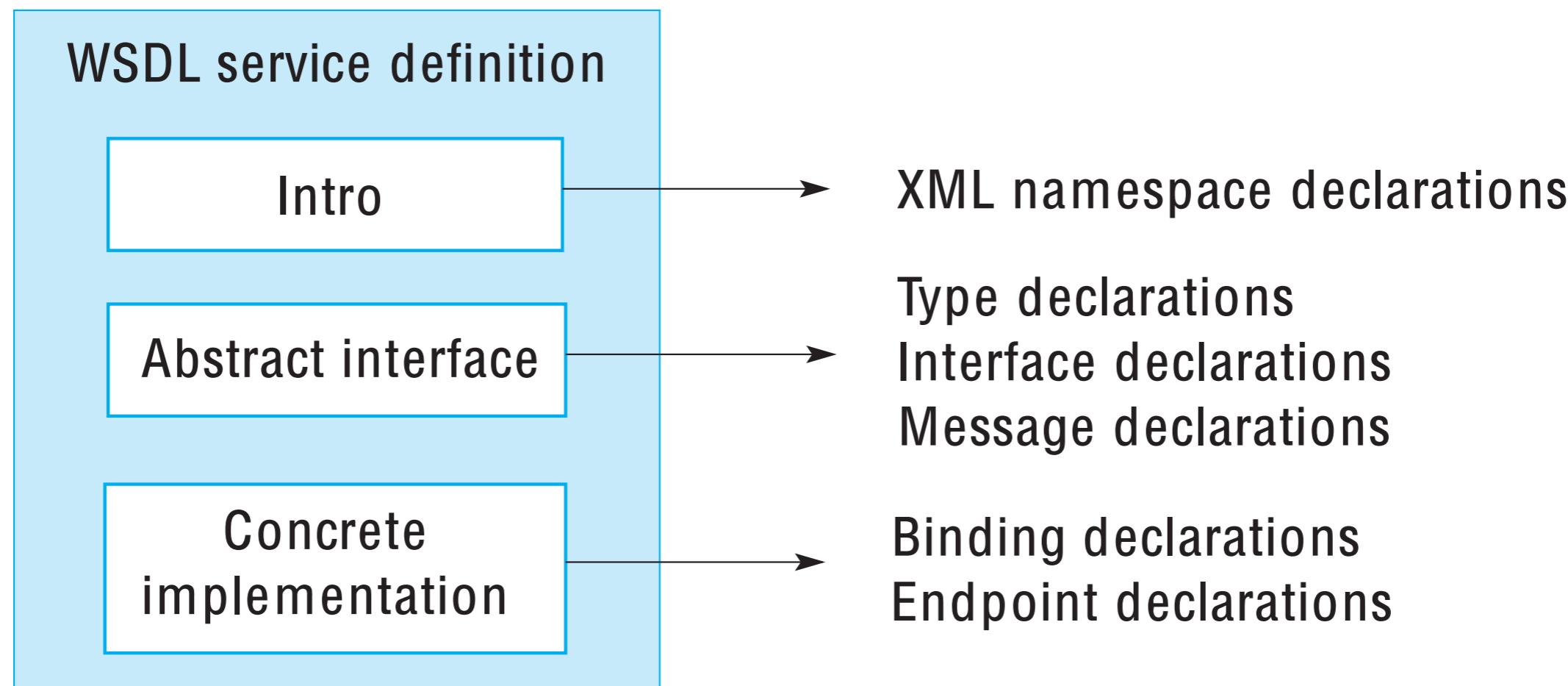
# I servizi come componenti riutilizzabili

- ~ Un servizio può essere definito come:
  - Un componente software riutilizzabile e debolmente accoppiato che incapsula funzionalità discrete che possono essere distribuite e accessibili via programma. Un servizio Web è un servizio a cui si accede utilizzando protocolli standard basati su Internet e XML.
- ~ Una distinzione fondamentale tra un servizio e un componente, come definito nel CBSE, è che i servizi sono indipendenti.
  - I servizi non hanno un'interfaccia "richiede".
  - I servizi si basano su una comunicazione basata su messaggi con messaggi espressi in XML.

# Web service description language

- ~ L'interfaccia del servizio è definita in una descrizione del servizio espressa in WSDL (Web Service Description Language).
- ~ La specifica WSDL definisce
  - Le operazioni supportate dal servizio e il formato dei messaggi inviati e ricevuti dal servizio.
  - Come si accede al servizio - cioè, il binding mappa l'interfaccia astratta su un insieme concreto di protocolli.
  - Dove si trova il servizio. Di solito è espresso come un URI (Universal Resource Identifier).

# Organizzazione di una specifica WSDL



# Componenti delle specifiche WSDL

- ~ La parte "cosa" di un documento WSDL, chiamata interfaccia, specifica quali operazioni supporta il servizio e definisce il formato dei messaggi inviati e ricevuti dal servizio.
- ~ La parte "come" di un documento WSDL, chiamata binding, mappa l'interfaccia astratta in un insieme concreto di protocolli. Il binding specifica i dettagli tecnici di come comunicare con un servizio web.
- ~ La parte "dove" di un documento WSDL descrive la posizione di una specifica implementazione del servizio web (il suo endpoint).

# Parte di una descrizione WSDL per un web service

- Definizione alcuni dei tipi utilizzati. Si supponga che il prefisso dello spazio dei nomi 'ws' si riferisca all'URI dello spazio dei nomi per gli schemi XML e che il prefisso dello spazio dei nomi associato a questa definizione sia weathns.

```
<types>
  <xs: schema targetNamespace = "http://.../weathns"
    xmlns: weathns = "http://.../weathns" >
    <xs:element name = "PlaceAndDate" type = "pdrec" />
    <xs:element name = "MaxMinTemp" type = "mmtrec" />
    <xs: element name = "InDataFault" type = "errmess" />

    <xs: complexType name = "pdrec"
      <xs: sequence>
        <xs:element name = "town" type = "xs:string"/>
        <xs:element name = "country" type = "xs:string"/>
        <xs:element name = "day" type = "xs:date" />
      </xs:complexType>
      Definitions of MaxMinType and InDataFault here
    </schema>
  </types>
```

# Parte di una descrizione WSDL per un web service

- ~ Ora si definiscono l'interfaccia e le sue operazioni. In questo caso, c'è una sola operazione per restituire le temperature massime e minime.

```
<interface name = "weatherInfo" >  
  <operation name = "getMaxMinTemps" pattern = "wsdlIns: in-out">  
    <input messageLabel = "In" element = "weathns: PlaceAndDate" />  
    <output messageLabel = "Out" element = "weathns:MaxMinTemp" />  
    <outfault messageLabel = "Out" element = "weathns:InDataFault" />  
  </operation>  
</interface>
```



# Servizi RESTful

# Web service RESTful

- ~ Gli attuali standard dei servizi web sono stati criticati come standard "pesanti", troppo generici e inefficienti.
  - Notevole attività di elaborazione per creare, trasmettere ed integrare messaggi XML
  - potrebbe esserci bisogno di hardware aggiuntivo per ottenere la qualità del servizio richiesto
- ~ REST (REpresentational State Transfer) è uno stile architettonico basato sul trasferimento di rappresentazioni di **risorse** da un server a un client.
- ~ Questo stile è alla base del web nel suo complesso ed è più semplice di SOAP/WSDL per l'implementazione dei servizi web.
- ~ I servizi RESTful comportano un overhead inferiore rispetto ai cosiddetti "grandi servizi web" e sono utilizzati da molte organizzazioni che implementano sistemi basati sui servizi.

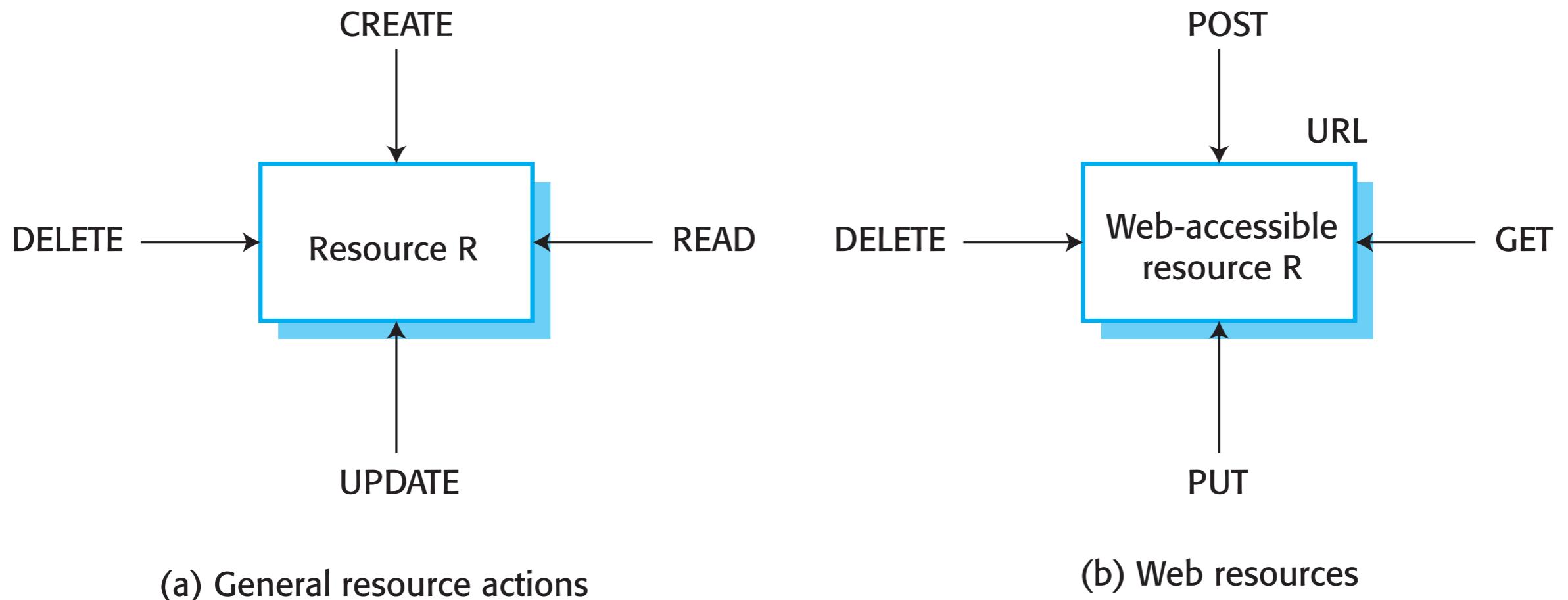
# Risorsa

- ~ L'elemento fondamentale di un'architettura RESTful è una risorsa.
- ~ In sostanza, una risorsa è semplicemente un elemento di dati come un catalogo, una cartella clinica o un documento, come il capitolo di questo libro.
- ~ In generale, le risorse possono avere più rappresentazioni, cioè possono esistere in diversi formati (la risorsa logica sottostante è sempre la stessa)
  - MS WORD
  - PDF
  - Quark XPress

# Operazioni principali

- ~ In un'architettura RESTful tutto è rappresentato come risorsa -> hanno un identificatore unico che è il loro URL
- ~ Le risorse sono simili agli oggetti -> quattro operazioni polimorfe fondamentali
  - & Creare - creare la risorsa.
  - & Leggere - restituire una rappresentazione della risorsa.
  - & Aggiornare - modificare il valore della risorsa.
  - & Eliminare - rende inaccessibile la risorsa.
- ~ Il Web è un esempio di sistema che ha un'architettura RESTful. Le pagine web sono risorse e l'identificatore unico di una pagina web è il suo URL.

# Risorse e azioni



# Funzionalità operative

- ~ POST viene utilizzato per creare una risorsa. Ha dati associati che definiscono la risorsa.
- ~ GET è usato per leggere il valore di una risorsa e restituirlo al richiedente nella rappresentazione specificata, come XHTML, che può essere resa in un browser web.
- ~ PUT è usato per aggiornare il valore di una risorsa.
- ~ DELETE è usato per eliminare la risorsa.

# Accesso alle risorse

- ~ Quando si utilizza un approccio RESTful, i dati sono esposti e vi si accede utilizzando il loro URL.
- ~ Pertanto, i dati meteorologici per ogni località del database potrebbero essere accessibili utilizzando URL come:
  - & <http://weather-info-example.net/temperatures/boston>
  - & <http://weather-info-example.net/temperatures/edinburgh>
- ~ Invoca l'operazione GET e restituisce un elenco di temperature massime e minime.
- ~ Per richiedere le temperature per una data specifica, si utilizza una query URL:
  - & [http://weather-info-example.net/temperatures/edinburgh?  
date=20140226](http://weather-info-example.net/temperatures/edinburgh?date=20140226)

# Risultati della query

- ~ La risposta a una richiesta GET in un servizio RESTful può includere URL.
- ~ Se la risposta a una richiesta è un insieme di risorse, si può includere l'URL di ciascuna di esse.
  - & <http://weather-info-example.net/temperatures/edinburgh-scotland>
  - & <http://weather-info-example.net/temperatures/edinburgh-australia>
  - & <http://weather-info-example.net/temperatures/edinburgh-maryland>

# Svantaggi dell'approccio RESTful

- ~ Quando un servizio ha un'interfaccia complessa e non è una semplice risorsa, può essere difficile progettare un insieme di servizi RESTful per rappresentarlo.
- ~ Non esistono standard per la descrizione delle interfacce RESTful, quindi gli utenti dei servizi devono affidarsi a una documentazione informale per comprendere l'interfaccia.
- ~ Quando si utilizzano i servizi RESTful, è necessario implementare la propria infrastruttura per il monitoraggio e la gestione della qualità e dell'affidabilità del servizio.

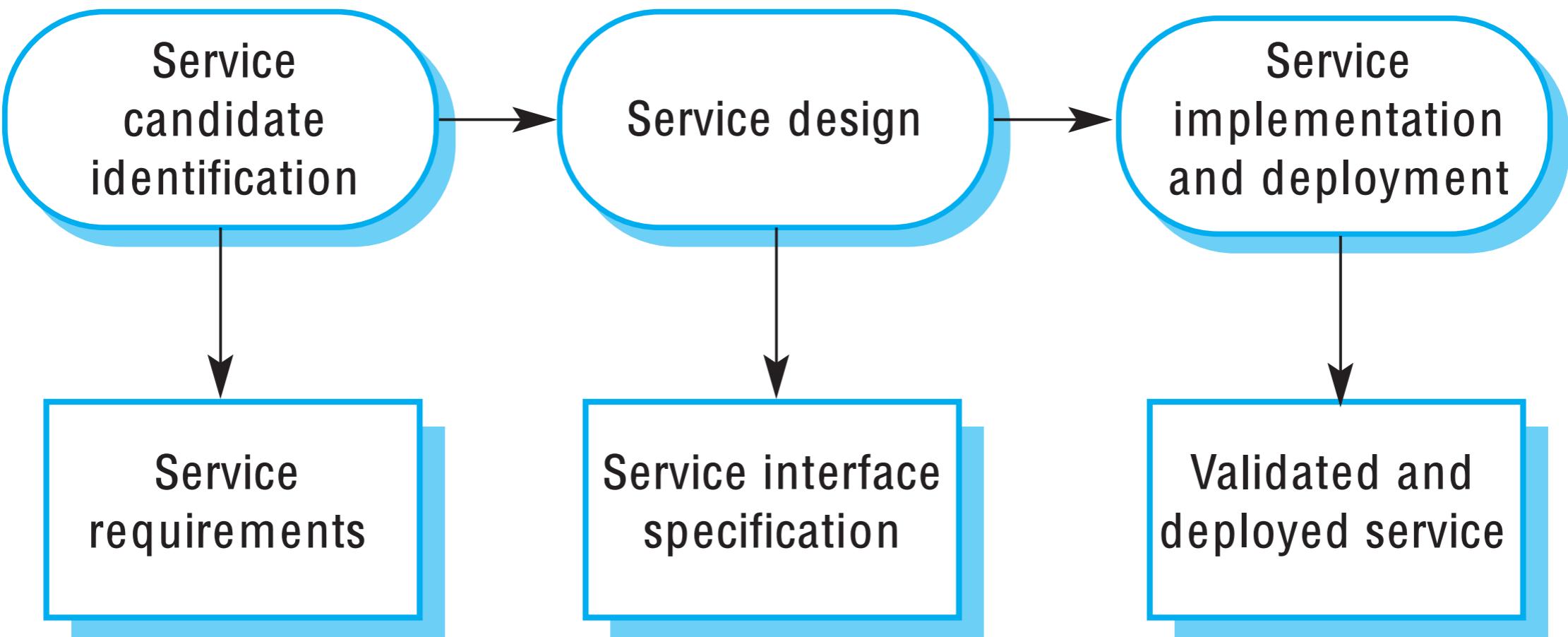


# Service Engineering

# Service Engineering

- ~ Il processo di sviluppo di servizi per il riutilizzo in applicazioni orientate ai servizi
- ~ Il servizio deve essere progettato come un'astrazione riutilizzabile che può essere utilizzata in diversi sistemi.
- ~ Devono essere progettate le funzionalità generalmente utili associate a tale astrazione e il servizio deve essere robusto e affidabile.
- ~ Il servizio deve essere documentato in modo da poter essere scoperto e compreso dai potenziali utenti.

# Il processo di ingenerai dei servizi



# Fasi dell'ingegneria del servizio

- ~ Identificazione dei candidati al servizio, in cui si identificano i possibili servizi che potrebbero essere implementati e si definiscono i requisiti del servizio.
- ~ Progettazione del servizio, in cui si progetta l'interfaccia logica del servizio e le sue interfacce di implementazione (SOAP e/o RESTful).
- ~ Implementazione e distribuzione del servizio, in cui si implementa e si testa il servizio e lo si rende disponibile per l'uso.

# Identificazione del candidato al servizio

- ~ I servizi devono supportare i processi aziendali.
- ~ L'identificazione dei candidati al servizio implica la **comprendere dei processi aziendali** di un'organizzazione per decidere quali servizi riutilizzabili potrebbero supportare tali processi.
- ~ Tre tipi fondamentali di servizi
  - Servizi di utilità che implementano funzionalità generali utilizzate da diversi processi aziendali.
  - Servizi di business che sono associati a una specifica funzione aziendale, ad esempio, in un'università, la registrazione degli studenti.
  - Servizi di coordinamento che supportano processi composti come il servizio di ordinazioni (inviare ordini ai fornitori, verificare il ricevimento delle merci, effettuare pagamenti).

# Servizi orientati ai compiti e alle entità

- ~ I servizi orientati al compito sono quelli associati a qualche attività.
- ~ I servizi orientati alle entità sono come gli oggetti. Sono associati a un'entità aziendale, come un modulo di domanda di lavoro.
- ~ I servizi di utilità o aziendali possono essere orientati alle entità o ai compiti, mentre i servizi di coordinamento sono sempre orientati ai compiti.

# Classificazione dei servizi

	Utility	Business	Coordination
Compito	Convertitore di valuta Localizzatore di dipendenti	Convalidare il modulo di richiesta di rimborso Controllare lo stato del credito	Elaborare le richieste di rimborso spese Pagare il fornitore esterno
Entità	Controllore di stile del documento Convertitore da modulo web a XML	Modulo per le spese Modulo di iscrizione per studenti	

# Identificare i servizi

~ Alcune domande che potrebbero servire per identificare i servizi:

- Il servizio è associato a una singola entità logica utilizzata in diversi processi aziendali?
- Il compito è svolto da diverse persone all'interno dell'organizzazione? Si può adattare a un modello RESTful?
- Il servizio è indipendente?
- Il servizio deve mantenere uno stato? È necessario un database?
- Il servizio può essere utilizzato da clienti esterni all'organizzazione?
- È probabile che i diversi utenti del servizio abbiano diversi requisiti non funzionali?

# Esempio di identificazione di servizi

- ~ Una grande azienda, che **vende apparecchiature informatiche**, ha concordato prezzi speciali per alcune configurazioni approvate per alcuni clienti. Per **facilitare gli ordini automatizzati**, l'azienda desidera produrre un servizio di catalogo che permetta ai clienti di **selezionare** le apparecchiature di cui hanno bisogno. A differenza di un catalogo per i consumatori, **gli ordini** non vengono effettuati direttamente attraverso l'interfaccia del catalogo. **I prodotti** vengono invece ordinati attraverso il sistema di approvvigionamento basato sul web di ogni azienda che **accede al catalogo** come servizio web. La ragione di questo è che la maggior parte delle aziende ha le proprie procedure di budgeting e di approvazione degli ordini e deve seguire il proprio processo di ordinazione.

# Servizio di catalogo

- ~ Tipo di servizio orienta alle entità —> risorsa sottostante è il catalogo
- ~ Creato da un fornitore per mostrare quali beni possono essere ordinati da altre aziende.
- ~ Requisiti funzionali del servizio
  - Deve essere creata una versione specifica del catalogo per ogni cliente
  - Il catalogo deve essere scaricabile
  - Possono essere confrontati le specifiche e i prezzi di un massimo di 6 articoli
  - Devono essere fornite funzioni di navigazione e ricerca
  - Deve essere fornita una funzione che consenta di prevedere la data di consegna degli articoli ordinati.
  - Devono essere supportati gli ordini virtuali che riservano la merce per 48 ore per consentire l'invio di un ordine aziendale.

# Servizi di catalogo - req. non funzionali

- ~ L'accesso sarà limitato ai dipendenti delle organizzazioni accreditate.
- ~ I prezzi e le configurazioni offerte a ciascuna organizzazione saranno riservati.
- ~ Il catalogo sarà disponibile dalle 7.00 alle 11.00.
- ~ Il catalogo deve essere in grado di elaborare fino a 10 richieste al secondo.

# Progettazione dell'interfaccia dei servizi

- ~ Consiste nel pensare alle operazioni associate al servizio e ai messaggi scambiati.
- ~ Il numero di messaggi scambiati per completare una richiesta di servizio deve essere normalmente ridotto al minimo.
- ~ È possibile che nei messaggi debbano essere incluse informazioni sullo stato del servizio.

# Descrizioni funzionali delle operazioni del servizio di catalogo

Operation	Description
CreaCatalogo	Crea una versione del catalogo su misura per un cliente specifico. Include un parametro opzionale per creare una versione PDF scaricabile del catalogo.
Visualizza	Visualizza tutti i dati associati a un elemento del catalogo specificato.
Ricerca	Questa operazione prende un'espressione logica e cerca nel catalogo in base a tale espressione. Visualizza un elenco di tutti gli elementi che corrispondono all'espressione di ricerca.

# Descrizioni funzionali delle operazioni del servizio di

Operation	Description
Confronta	Fornisce un confronto tra un massimo di sei caratteristiche (ad esempio, prezzo, dimensioni, velocità del processore, ecc.) di un massimo di quattro articoli del catalogo.
DataConsegna	Restituisce la data di consegna prevista per un articolo se ordinato quel giorno.
CreaOrdineVirtuale	Riserva il numero di articoli da ordinare da un cliente e fornisce informazioni sugli articoli per il sistema di approvvigionamento del cliente.

# Fasi di progettazione dell'interfaccia

## ~ Progettazione dell'interfaccia logica

- Inizia con i requisiti del servizio e definisce i nomi delle operazioni e i parametri associati al servizio. Vanno definite anche le eccezioni

## ~ Progettazione dei messaggi (SOAP)

- Per i servizi basati su SOAP, progettare la struttura e l'organizzazione dei messaggi di ingresso e di uscita. Notazioni come UML sono una rappresentazione più astratta di XML.
- La specifica logica viene convertita in una descrizione WSDL.

## ~ Progettazione dell'interfaccia (REST)

- Progettare come le operazioni richieste si adattano alle operazioni REST e quali risorse sono necessarie.

## ~ Dopo una definizione logica formale di quello che il servizio dovrebbe fare

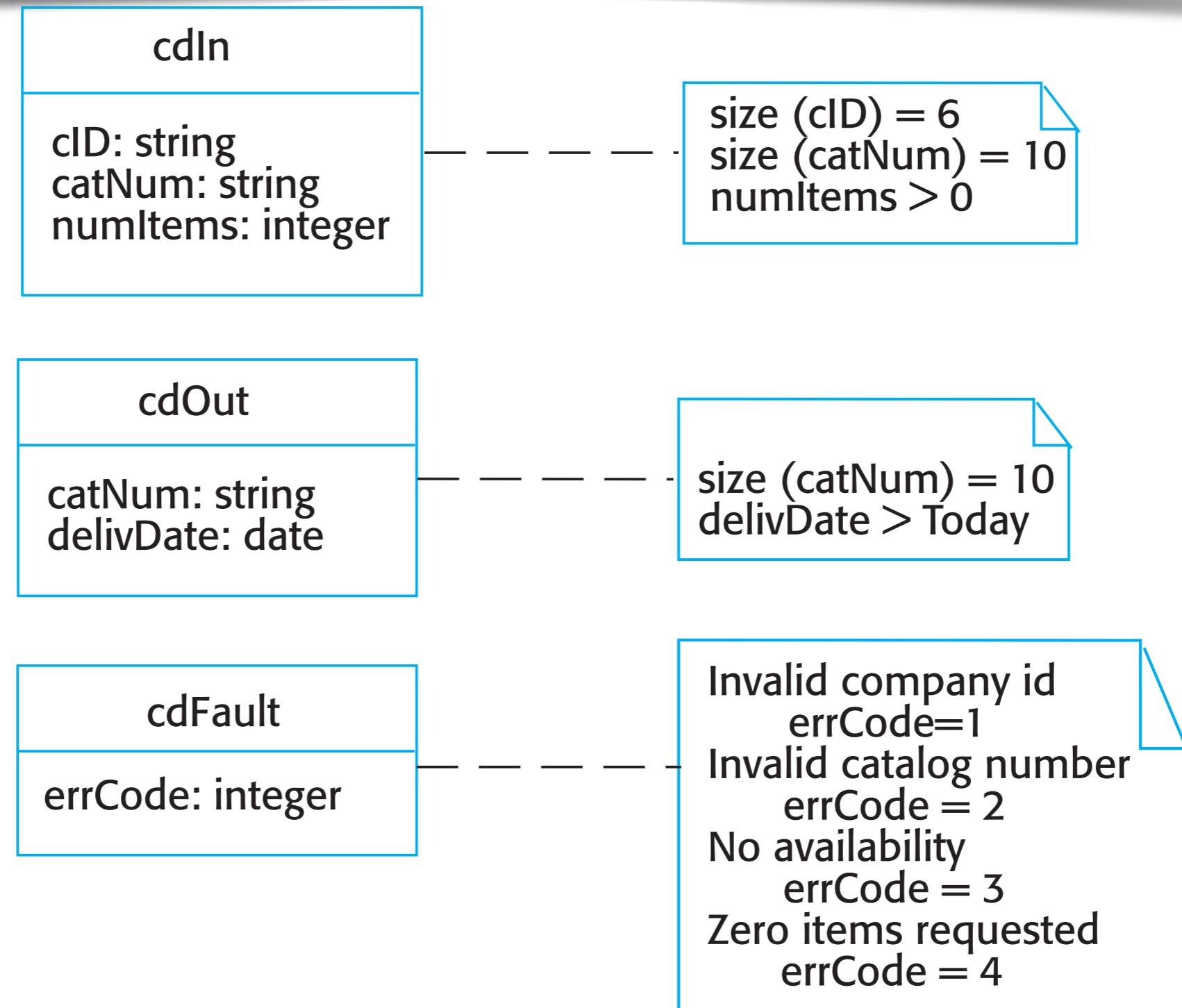
- si inseriscono i dettagli agli input e agli output

Operazione	Input	Output	Eccezioni
CreaCatalogo	<i>ccln</i> ID dell'azienda Flag PDF	<i>ccOut</i> URL del catalogo per quell'azienda	<i>ccFault</i> ID azienda non valido
Visualizza	<i>visIn</i> URL del catalogo Numero di catalogo	<i>visOut</i> URL della pagina con le informazioni sull'articolo	<i>visFault</i> Numero di catalogo non valido
Ricerca	<i>ricIn</i> URL del catalogo Stringa di ricerca	<i>ricOut</i> URL della pagina web con i risultati della ricerca	<i>ricFault</i> Stringa di ricerca errata
Confronta	<i>confIn</i> URL del catalogo Caratteristiche (fino a 6) Numero di catalogo (fino a 4)	<i>confOut</i> URL della pagina con la tabella dei confronti	<i>confFault</i> ID azienda non valido Numero di catalogo non valido Caratteristica sconosciuta
DataConsegna	<i>dataIn</i> ID dell'azienda Numero di catalogo Numero di articoli ordinati	<i>dataOut</i> Data di consegna stimata	<i>dataFault</i> ID azienda non valido Nessuna disponibilità Articoli ordinati zero
CreaOrdineVirtuale	<i>cordIn</i> ID dell'azienda Numero di catalogo Numero di articoli ordinati	<i>cordOut</i> Numero di catalogo Numero di articoli ordinati Data di consegna stimata Prezzo unitario stimato Prezzo totale stimato	<i>cordFault</i> ID azienda non valido Numero di catalogo non valido Articoli ordinati zero

# Progettazione delle interfacce

- ~ In alcuni casi basta una descrizione testuale delle interfacce
- ~ Altre volte occorre un progetto più dettagliato
  - per una descrizione dettagliata può essere usata una notazione grafica (diagrammi UML o formato leggibile JSON)
  - possono anche essere aggiunte delle annotazioni

# Definizione UML dei messaggi di ingresso e di uscita



# Interfaccia RESTful

- ~ Deve esistere una risorsa che rappresenti un catalogo specifico dell'azienda. Questa dovrebbe avere un URL della forma <base catalog>/<nome azienda> e dovrebbe essere creata con un'operazione POST.
- ~ Ogni elemento del catalogo deve avere un proprio URL del tipo:
  - <base catalog>/<nome azienda>/<identificatore articolo>.
- ~ L'operazione GET viene utilizzata per recuperare gli elementi.
  - La ricerca è implementata utilizzando l'URL di un elemento del catalogo come parametro GET.
  - La ricerca è implementata utilizzando GET con il catalogo aziendale come URL e la stringa di ricerca come parametro della query. Questa operazione GET restituisce un elenco di URL degli elementi che corrispondono alla ricerca.

# Interfaccia RESTful

- ~ L'operazione **Compare** può essere implementata come una sequenza di operazioni GET, per recuperare i singoli articoli, seguita da un'operazione POST per creare la tabella di confronto e da un'operazione GET finale per restituirla all'utente.
- ~ Le operazioni **CheckDelivery** e **MakevirtualOrder** richiedono una risorsa aggiuntiva, che rappresenta un ordine virtuale.
- ~ Si utilizza un'operazione POST per creare questa risorsa con il numero di articoli richiesti. L'id dell'azienda viene utilizzato per compilare automaticamente il modulo d'ordine e viene calcolata la data di consegna. Questa può essere recuperata con un'operazione GET.

# Implementazione e distribuzione del servizio

- ~ Programmare i servizi utilizzando un linguaggio di programmazione standard o un linguaggio di workflow.
- ~ I servizi devono poi essere testati creando messaggi di input e verificando che i messaggi di output prodotti siano quelli attesi.
- ~ La distribuzione comporta la pubblicizzazione del servizio e la sua installazione su un server web. I server attuali forniscono il supporto per l'installazione dei servizi

# Servizi di sistema legacy

- ~ I servizi possono essere implementati realizzando un'interfaccia di servizio con i sistemi legacy esistenti.
- ~ I sistemi legacy offrono ampie funzionalità e questo può ridurre il costo dell'implementazione del servizio.
- ~ Le applicazioni esterne possono accedere a queste funzionalità attraverso le interfacce di servizio.

# Descrizione dei servizi

- ~ Informazioni sulla vostra attività, dettagli di contatto, ecc. Questo è importante per motivi di fiducia. Gli utenti di un servizio devono avere la certezza che non si comporterà in modo malevolo.
- ~ una descrizione informale delle funzionalità fornite dal servizio. Questo aiuta i potenziali utenti a decidere se il servizio è quello che vogliono.
- ~ una descrizione di come utilizzare i servizi basati su SOAP e RESTful.
- ~ Informazioni sulla sottoscrizione che consentono agli utenti di registrarsi per ricevere informazioni sugli aggiornamenti del servizio.

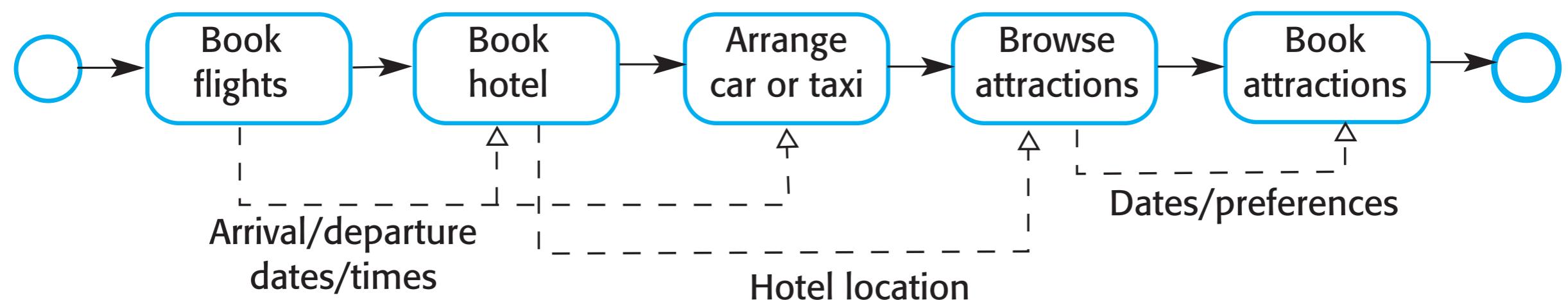


# Composizione dei servizi

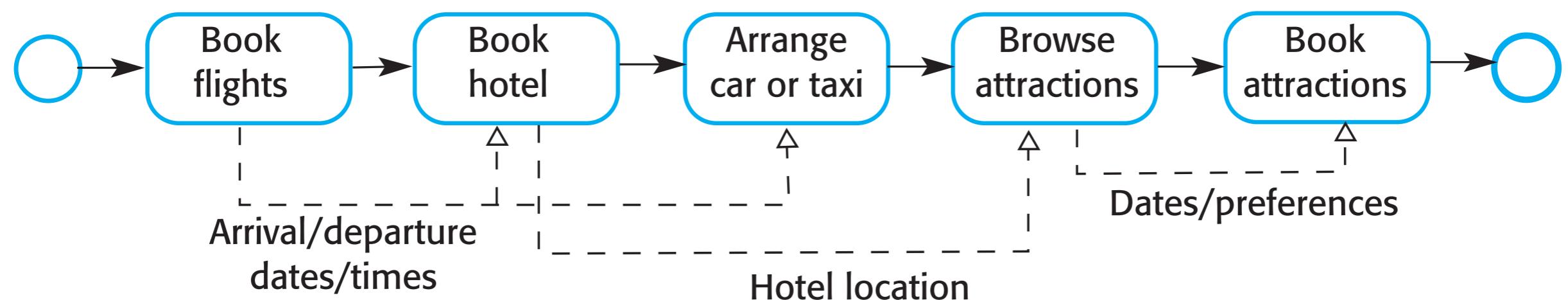
# Sviluppo software con servizi

- ~ I servizi esistenti vengono composti e configurati per creare nuovi servizi e applicazioni compositi.
- ~ La base per la composizione dei servizi è spesso un flusso di lavoro.
- ~ I flussi di lavoro sono sequenze logiche di attività che, insieme, modellano un processo aziendale coerente.
- ~ Ad esempio, fornire un servizio di prenotazione di viaggi che permetta di coordinare voli, noleggio auto e prenotazioni alberghiere.

# Workflow di un pacchetto vacanze

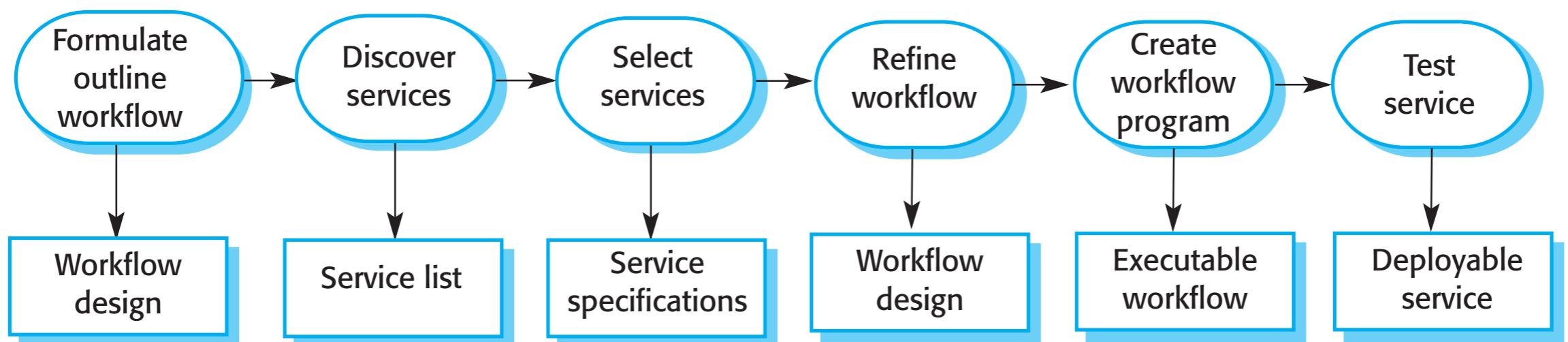


# Workflow di un pacchetto vacanze



*azioni di compensazione*

# Costruzione di un servizio tramite composizione



# Costruzione mediante composizione

- ~ Formulare un flusso di lavoro di massima
  - In questa fase iniziale della progettazione del servizio, si utilizzano i requisiti del servizio composito come base per creare un progetto di servizio "ideale".
- ~ Scoprire i servizi
  - In questa fase del processo, si cercano i registri o i cataloghi dei servizi per scoprire quali servizi esistono, chi li fornisce e i dettagli della fornitura del servizio.
- ~ Selezionare i servizi possibili
  - I criteri di selezione includono ovviamente la funzionalità dei servizi offerti. Possono anche includere il costo dei servizi e la qualità del servizio (reattività, disponibilità, ecc.) offerto.

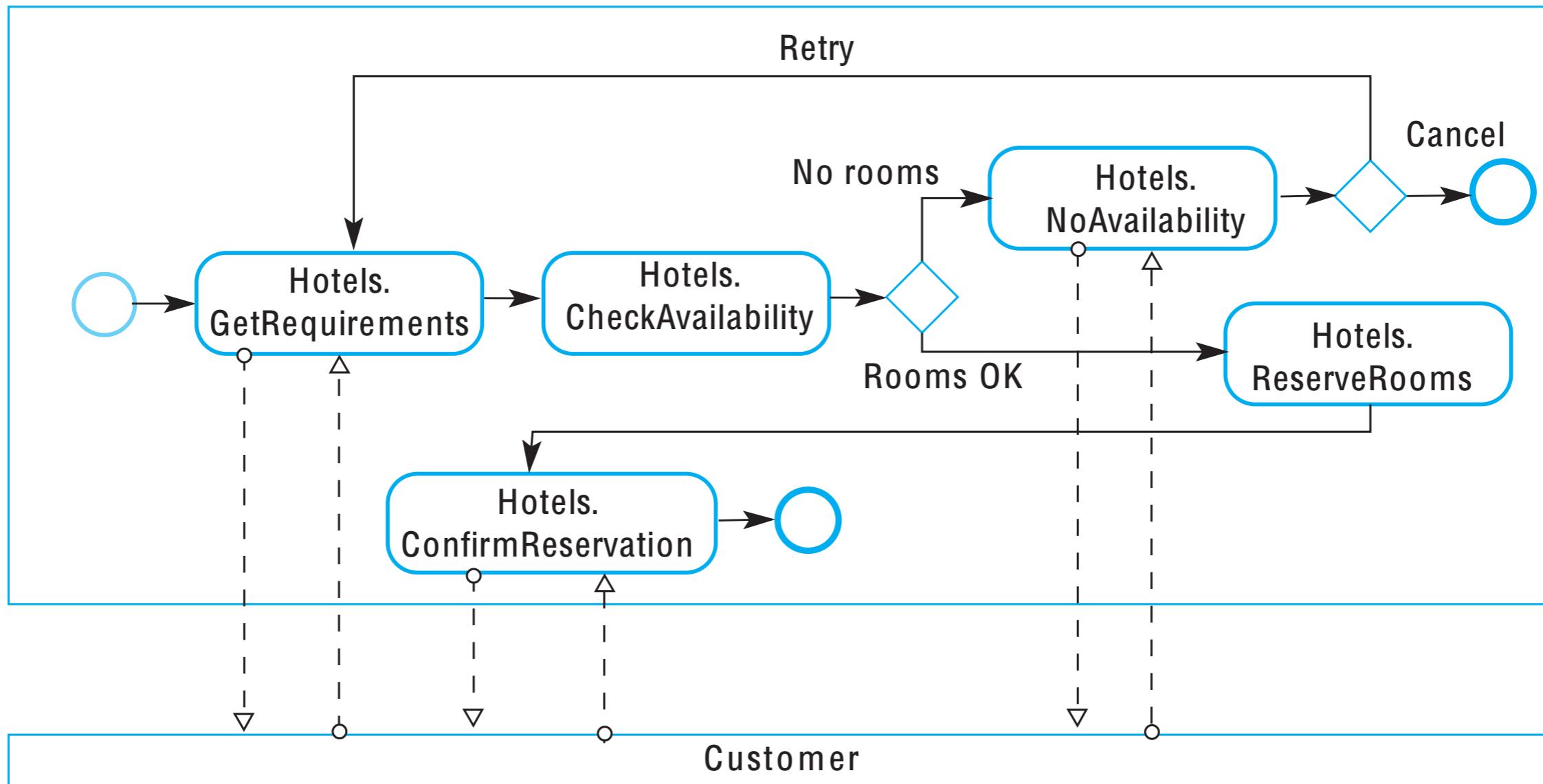
# Costruzione mediante composizione

- ~ Affinare il flusso di lavoro.
  - Questo comporta l'aggiunta di dettagli alla descrizione astratta e forse l'aggiunta o la rimozione di attività del flusso di lavoro.
- ~ Creare il programma del flusso di lavoro
  - In questa fase, il progetto astratto del flusso di lavoro viene trasformato in un programma eseguibile e viene definita l'interfaccia del servizio. Si può usare un linguaggio di programmazione convenzionale, come Java, o un linguaggio di workflow, come WS-BPEL.
- ~ Test del servizio o dell'applicazione completati
  - Il processo di collaudo del servizio composito completato è più complesso del collaudo dei componenti in situazioni in cui si utilizzano servizi esterni.

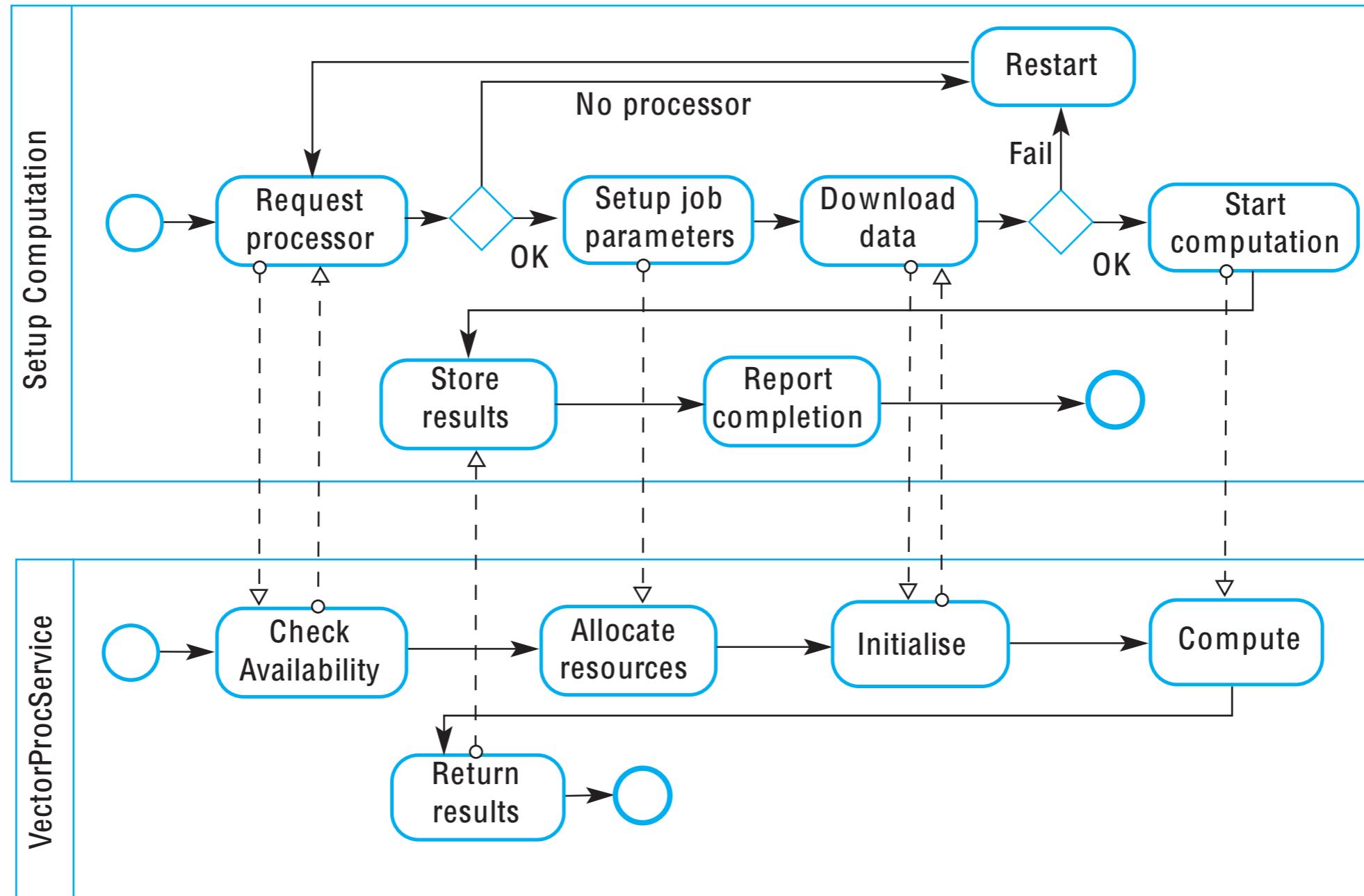
# Progettazione e implementazione di un workflow

- ~ WS-BPEL (Business Process Execution Language) è uno standard XML per le specifiche dei flussi di lavoro. Tuttavia, le descrizioni di WS-BPEL sono lunghe e illeggibili.
- ~ Le notazioni grafiche dei flussi di lavoro, come BPMN, sono più leggibili e WS-BPEL può essere generato da esse.
- ~ Nei sistemi inter-organizzativi, vengono creati flussi di lavoro separati per ciascuna organizzazione e collegati attraverso lo scambio di messaggi.
- ~ I flussi di lavoro possono essere utilizzati con servizi basati su SOAP e RESTful.

# Un frammento del flusso di lavoro di una prenotazione



# Interazione tra workflow



# Key points

- ~ L'architettura orientata ai servizi è un approccio all'ingegneria del software in cui i servizi riutilizzabili e standardizzati sono gli elementi di base dei sistemi applicativi.
- ~ I servizi possono essere implementati all'interno di un'architettura orientata ai servizi utilizzando un insieme di standard per i servizi web basati su XML. Questi includono standard per la comunicazione dei servizi, la definizione delle interfacce e l'implementazione dei servizi nei flussi di lavoro.
- ~ In alternativa, si può utilizzare un'architettura RESTful, basata su risorse e operazioni standard su tali risorse.
- ~ Un approccio RESTful utilizza i protocolli http e https per la comunicazione dei servizi e mappa le operazioni sui verbi standard http POST, GET, PUT e DELETE.

# Key points

- ~ I servizi di utilità forniscono funzionalità di uso generale; i servizi aziendali implementano parte di un processo aziendale; i servizi di coordinamento coordinano l'esecuzione del servizio.
- ~ L'ingegneria dei servizi comporta l'identificazione dei servizi candidati all'implementazione, la definizione delle interfacce dei servizi e l'implementazione, il collaudo e la distribuzione dei servizi.
- ~ Lo sviluppo di software che utilizza i servizi comporta la composizione e la configurazione dei servizi per creare nuovi servizi e sistemi composti.
- ~ I linguaggi grafici per i flussi di lavoro, come il BPMN, possono essere utilizzati per descrivere un processo aziendale e i servizi utilizzati in tale processo.