



Component based software engineering

Ingegneria del Software

Argomenti

- ~ Componenti e modelli di componenti
- ~ Processi CBSE
- ~ Composizione dei componenti

Sviluppo basato sui componenti

- ~ L'ingegneria del software basata sui componenti (CBSE) è un approccio allo sviluppo del software che si basa sul riutilizzo di entità chiamate "componenti software".
- ~ È emerso dal fallimento dello sviluppo orientato agli oggetti nel supportare un riutilizzo efficace. Le singole classi di oggetti sono troppo dettagliate e specifiche.
- ~ I componenti sono **più astratti** delle classi di oggetti e possono essere considerati come fornitori di servizi autonomi. Possono esistere come entità autonome.

Elementi essenziali del CBSE

- ~ Componenti indipendenti specificati dalle loro interfacce.
- ~ Standard dei componenti per facilitarne l'integrazione. Componenti conformi agli standard interagiscono in modo indipendente dal linguaggio in cui sono scritti.
- ~ Middleware che fornisce supporto per l'interoperabilità e l'integrazione dei componenti. Gestisce problemi di basso livello.
- ~ un processo di sviluppo orientato al riutilizzo che consente l'evoluzione dei requisiti in base alle funzionalità dei componenti a disposizione.

CBSE e principi di design

- ~ Oltre ai vantaggi del riutilizzo, il CBSE si basa su solidi principi di progettazione dell'ingegneria del software:
 - I componenti sono indipendenti e non interferiscono tra loro;
 - Le implementazioni dei componenti sono nascoste;
 - La comunicazione avviene attraverso interfacce ben definite;
 - Un componente può essere sostituito da un altro se la sua interfaccia viene mantenuta;
 - Le infrastrutture dei componenti offrono una serie di servizi standard.

Componenti standard

- ~ È necessario stabilire degli standard in modo che i componenti possano comunicare tra loro e interagire.
- ~ Sfortunatamente, sono stati creati diversi standard per i componenti in competizione tra loro:
 - Enterprise Java Beans di Sun
 - COM e .NET di Microsoft
 - CCM di CORBA
- ~ In pratica, questi molteplici standard hanno ostacolato l'adozione di CBSE. È impossibile che i componenti sviluppati con approcci diversi lavorino insieme.
 - soluzione -> il concetto di **servizio**

Software engineering orientata ai servizi

- ~ Un servizio eseguibile è un tipo di **componente indipendente**. Ha un'interfaccia "fornisce" ma non un'interfaccia "richiede".
- ~ Fin dall'inizio, i servizi sono stati basati su standard, quindi non ci sono problemi di comunicazione tra servizi offerti da fornitori diversi.
- ~ Le prestazioni del sistema possono essere più lente con i servizi, ma questo approccio sta sostituendo il CBSE in molti sistemi.
- ~ I servizi li vedremo più avanti



Componenti e modelli di componenti



Componenti

- ~ I componenti forniscono un servizio a prescindere dal luogo di esecuzione del componente o dal suo linguaggio di programmazione.
- ~ Un componente è un'entità eseguibile indipendente che può essere costituita da uno o più oggetti eseguibili;
- ~ L'interfaccia del componente è pubblicata e tutte le interazioni avvengono attraverso l'interfaccia pubblicata;

Definizione di componente

~ Councill and Heinmann:

- A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard.
- Un componente software è un elemento software conforme a un modello di componente e può essere distribuito e composto in modo indipendente senza modifiche secondo uno standard di composizione.

~ Szyperski:

- A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third-parties.
- Un componente software è un'unità di composizione con interfacce specificate contrattualmente e dipendenze esplicite dal contesto. Un componente software può essere distribuito in modo indipendente ed è soggetto a composizione da parte di terzi.

~ da entrambe le definizioni -> un componente è un elemento incorporato in un sistema invece di un servizio che viene richiamato da un sistema.

Caratteristiche dei componenti

Component Description characteristic

Componibile	Affinché un componente sia componibile, tutte le interazioni esterne devono avvenire attraverso interfacce definite pubblicamente. Inoltre, deve fornire accesso esterno alle informazioni che lo riguardano, come i suoi metodi e attributi.
Distribuibile	Per essere distribuibile, un componente deve essere autonomo. Deve essere in grado di operare come entità indipendente su una piattaforma di componenti che fornisca un'implementazione del modello del componente. Questo di solito significa che il componente è binario e non deve essere compilato prima di essere distribuito. Se un componente è implementato come servizio, non deve essere distribuito dall'utente di un componente. Viene invece distribuito dal fornitore del servizio.

Caratteristiche dei componenti

Caratteristica del Descrizione componente

Documentato	I componenti devono essere completamente documentati, in modo che i potenziali utenti possano decidere se i componenti soddisfano o meno le loro esigenze. La sintassi e, idealmente, la semantica di tutte le interfacce dei componenti devono essere specificate.
Indipendente	Un componente deve essere indipendente: deve essere possibile comporlo e distribuirlo senza dover utilizzare altri componenti specifici. Nelle situazioni in cui il componente ha bisogno di servizi forniti dall'esterno, questi dovrebbero essere esplicitamente definiti in una specifica di interfaccia "requires".
Standardizzato	La standardizzazione dei componenti significa che un componente utilizzato in un processo CBSE deve essere conforme a un modello di componente standard. Questo modello può definire le interfacce dei componenti, i metadati dei componenti, la documentazione, la composizione e la distribuzione.

Un componente come un fornitore di servizi

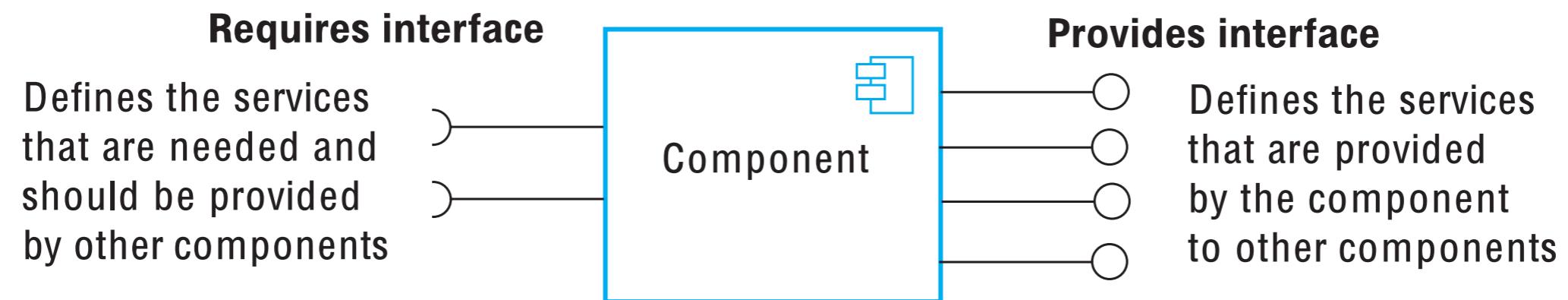
~ caratteristiche critiche:

- Il componente è un'entità indipendente ed eseguibile.
Non deve essere compilato prima di essere utilizzato con altri componenti.
- I servizi offerti da un componente sono resi disponibili attraverso un'interfaccia e tutte le interazioni del componente avvengono attraverso tale interfaccia.
- L'interfaccia del componente è espressa in termini di operazioni parametrizzate e il suo stato interno non è mai esposto.

Interfacce dei componenti

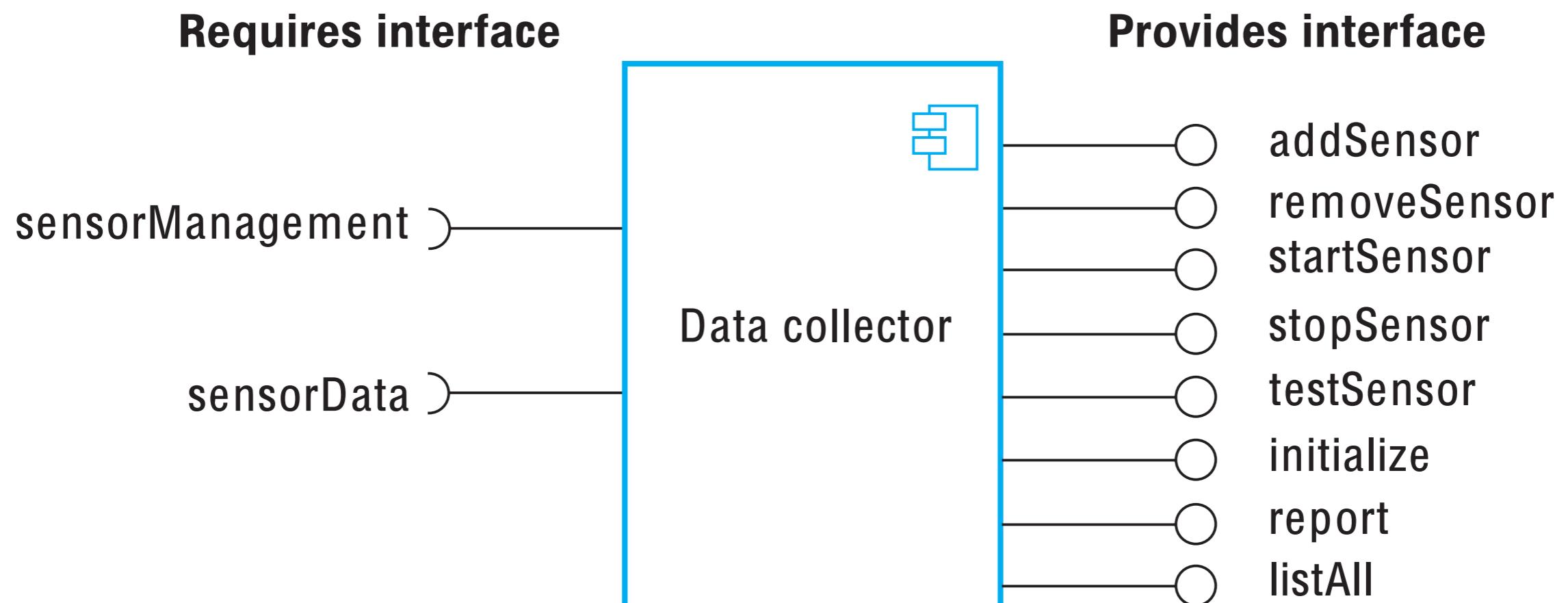
- ~ Tutti i componenti hanno due interfacce correlate (figura slide successiva)
- ~ Interfaccia Provides
 - Definisce i servizi forniti dal componente ad altri componenti.
 - Questa interfaccia, in sostanza, è l'API del componente. Definisce i metodi che possono essere richiamati da un utente del componente.
- ~ Interfaccia Requires
 - Definisce i servizi che specificano quali servizi devono essere resi disponibili affinché il componente venga eseguito come specificato.
 - Questo non compromette l'indipendenza o la distribuibilità di un componente, perché l'interfaccia "requires" non definisce come questi servizi debbano essere forniti.

Interfacce dei componenti



Note UML notation. Ball and sockets can fit together.

Esempio: data collector



Accesso ai componenti

- ~ L'accesso ai componenti avviene tramite chiamate di procedura remote (RPC).
- ~ Ogni componente ha un identificatore unico (di solito un URL) e può essere richiamato da qualsiasi computer in rete.
- ~ Pertanto, può essere chiamato in modo simile a una procedura o a un metodo in esecuzione su un computer locale.
- ~ Il componente chiamato usa lo stesso meccanismo per accedere ai componenti dei servizi richiesti

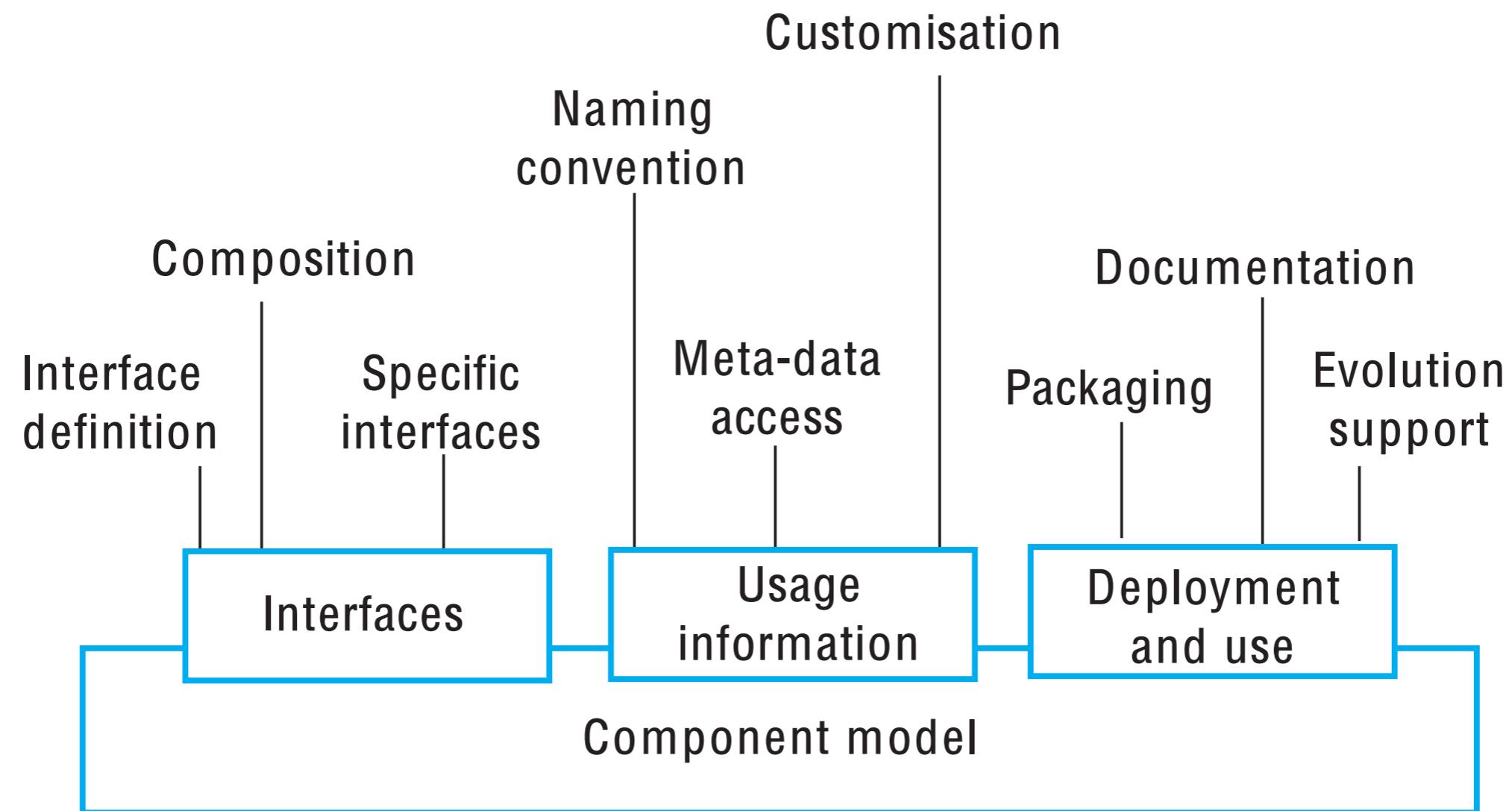
Componente come “servizio esterno”

- ~ una differenza importante tra componente come servizio esterno e componente come elemento di programma utilizzato tramite una chiamata di procedura remota:
 - & i servizi sono entità completamente indipendenti
 - & non richiedono altri componenti a supporto delle loro operazioni
 - & questi sono forniti internamente
 - & i programmi possono usare servizi senza bisogno di fornire supporto addizionale per la sua esecuzione.

Modelli di componenti

- ~ Un modello di componente è una definizione di standard per l'implementazione, la documentazione e la distribuzione dei componenti.
- ~ Esempi di modelli di componenti
 - Modello EJB (Enterprise Java Beans)
 - Modello COM+ (modello .NET)
 - Modello dei componenti Corba
- ~ Il modello dei componenti specifica come devono essere definite le interfacce e gli elementi che devono essere inclusi in una definizione di interfaccia.

Elementi di base di un modello di componenti



Elementi di un modello di componente

~ Interfacce

- I componenti vengono definiti specificando le loro interfacce. Il modello del componente specifica come devono essere definite le interfacce e gli elementi, come i nomi delle operazioni, i parametri e le eccezioni, che devono essere inclusi nella definizione dell'interfaccia.
- Per i servizi web si usano in genere linguaggi basati su XML, in EJB si usa Java per la definizione delle interfacce

~ Utilizzo

- Affinché i componenti possano essere distribuiti e accessibili da remoto, è necessario che ad essi sia associato un nome o un handle unico. Questo deve essere unico a livello globale.
- I metadati di un componente sono informazioni sul componente stesso, sulle interfacce e i suoi attributi. Sono importanti, consentono all'utente del componente di trovare servizi offerti e richiesti.
- Ricordiamo che i componenti sono entità generiche che devono essere personalizzate per il loro particolare sistema applicativo

~ Distribuzione (Consegna)

- Il modello dei componenti include una specifica di come i componenti debbano essere impacchettati per la distribuzione come entità indipendenti ed eseguibili.
- In seguito alla richiesta di nuovi requisiti, i componenti vanno modificati o sostituiti

Supporto Middleware

- ~ I modelli dei componenti sono la base del middleware che fornisce il supporto per l'esecuzione dei componenti. Implementa i servizi comuni dei componenti e ne fornisce le interfacce.
- ~ Le implementazioni dei modelli di componenti forniscono:
 - servizi di piattaforma che permettono ai componenti scritti secondo il modello di comunicare;
 - servizi di supporto che sono servizi comuni indipendenti dall'applicazione utilizzati da diversi componenti.
- ~ Per utilizzare i servizi forniti da un modello, i componenti vengono distribuiti in un contenitore.
 - Il contenitore è un'implementazione dei servizi di supporto più una definizione che un componente deve fornire per integrarsi con il contenitore.

Servizi middleware definiti in un modello di componenti

Support services

Component management

Concurrency

Transaction management

Persistence

Resource management

Security

Platform services

Addressing

Interface definition

Exception management

Component communications



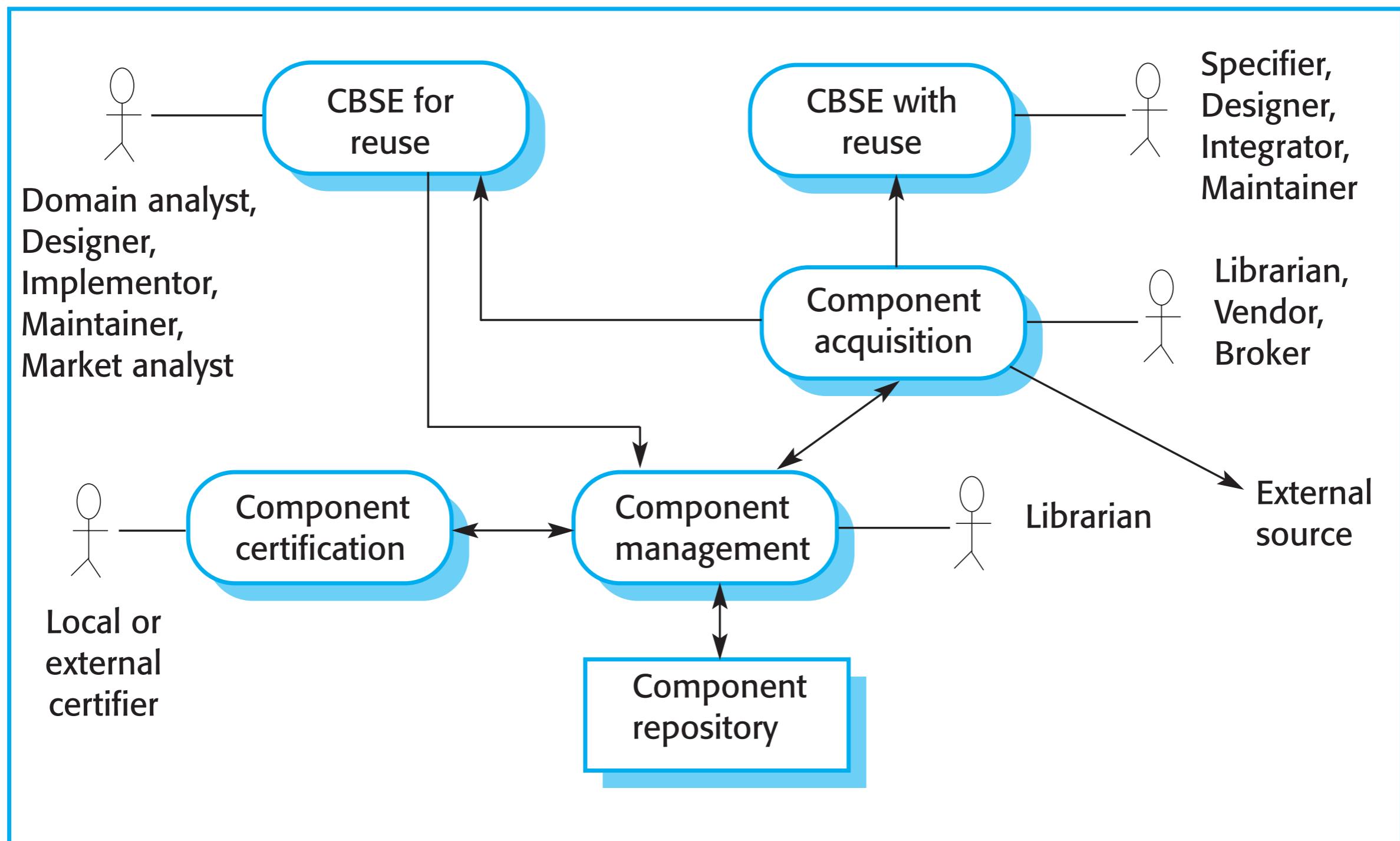
Processi CBSE

I processi CBSE

- ~ I processi CBSE sono processi software che supportano l'ingegneria del software basata sui componenti.
 - Essi tengono conto delle possibilità di riutilizzo e delle diverse attività di processo coinvolte nello sviluppo e nell'utilizzo di componenti riutilizzabili.
- ~ Sviluppo per il riutilizzo
 - Questo processo riguarda lo sviluppo di componenti o servizi che saranno riutilizzati in altre applicazioni. Di solito comporta la generalizzazione di componenti esistenti.
- ~ Sviluppo con riutilizzo
 - È il processo di sviluppo di nuove applicazioni che utilizzano componenti e servizi esistenti.

Processi CBSE

CBSE processes



I processi di supporto alla CBSE

- ~ L'acquisizione di componenti è il processo di acquisizione di componenti da riutilizzare o sviluppare in un componente riutilizzabile.
 - Può comportare l'accesso a componenti o servizi sviluppati localmente o il reperimento di tali componenti da una fonte esterna.
- ~ La gestione dei componenti si occupa di gestire i componenti riutilizzabili di un'azienda, assicurando che siano adeguatamente catalogati, conservati e resi disponibili per il riutilizzo.
- ~ La certificazione dei componenti è il processo di verifica di un componente e di certificazione della sua conformità alle specifiche.

CBSE per il riutilizzo

- ~ IL CBSE per il riutilizzo si concentra sullo sviluppo di componenti.
- ~ I componenti sviluppati per un'applicazione specifica di solito devono essere generalizzati per renderli riutilizzabili.
- ~ È più probabile che un componente sia riutilizzabile se è associato a un'astrazione di dominio stabile (oggetto aziendale).
 - Ad esempio, in un ospedale le astrazioni stabili del dominio sono associate al focus fondamentale dell'applicazione: infermieri, pazienti, trattamenti, ecc.

CBSE per il riutilizzo

- ~ Contrariamente a quanto ci si aspettava quando è nata la CBSE
 - c'è un solo produttore di componenti
 - il processo CBSE è principalmente usato all'interno delle organizzazioni che hanno scelto di sviluppare software secondo CBSE
 - Le organizzazioni hanno una base di componenti già sviluppati al loro interno
 - il riutilizzo potrebbe richiedere qualche modifica
 - il riutilizzo potrebbe richiedere costi -> ha senso analizzare quanto convenga estendere per fornire componenti riutilizzabile rispetto alla effettiva probabilità di riutilizzo

Sviluppo di componenti per il riuso

- ~ I componenti da riutilizzare possono essere costruiti appositamente generalizzando i componenti esistenti.
 - : Riutilizzabilità dei componenti
 - : Deve riflettere astrazioni stabili del dominio;
 - : Dovrebbe nascondere la rappresentazione dello stato;
 - : Deve essere il più possibile indipendente;
 - : Deve pubblicare le eccezioni attraverso l'interfaccia del componente.
- ~ C'è un compromesso tra riusabilità e usabilità
- ~ Più l'interfaccia è generale, maggiore è la riusabilità, ma è più complessa e quindi meno usabile.

Modifiche per rendere un componente riutilizzabile

- ~ Eliminare i metodi specifici dell'applicazione.
- ~ Modificare i nomi per renderli generali.
- ~ Aggiungere metodi per ampliare la copertura.
- ~ Rendere coerente la gestione delle eccezioni.
- ~ Aggiungere un'interfaccia di configurazione per l'adattamento dei componenti.
- ~ Integrare i componenti necessari per ridurre le dipendenze.

Problema della gestione delle eccezioni

- ~ I componenti non dovrebbero gestire autonomamente le eccezioni, perché ogni applicazione avrà i propri requisiti per la gestione delle eccezioni.
- ~ Piuttosto, il componente dovrebbe definire quali eccezioni possono verificarsi e pubblicarle come parte dell'interfaccia.
- ~ In pratica, però, ci sono due problemi:
 - La pubblicazione di tutte le eccezioni porta a interfacce pesanti e più difficili da capire. Ciò può scoraggiare i potenziali utenti del componente.
 - Il funzionamento del componente può dipendere dalla gestione delle eccezioni locali e la loro modifica può avere serie implicazioni per la funzionalità del componente.

Riutilizzabilità dei componenti

- ~ La riutilizzabilità dipende dal dominio di applicazione
 - se il dominio è generico ed il componente implementa una funzionalità standard —> è più riutilizzabile
 - diventa più complesso
- ~ C'è un compromesso tra riutilizzabilità e comprensibilità di un componente
- ~ Il costo di sviluppo dei componenti riutilizzabili può essere superiore al costo degli equivalenti specifici.
 - Questo costo aggiuntivo per il miglioramento della riusabilità dovrebbe essere un costo dell'organizzazione piuttosto che del progetto.
- ~ I componenti generici possono essere meno efficienti dal punto di vista dello spazio e avere tempi di esecuzione più lunghi rispetto ai loro equivalenti specifici.

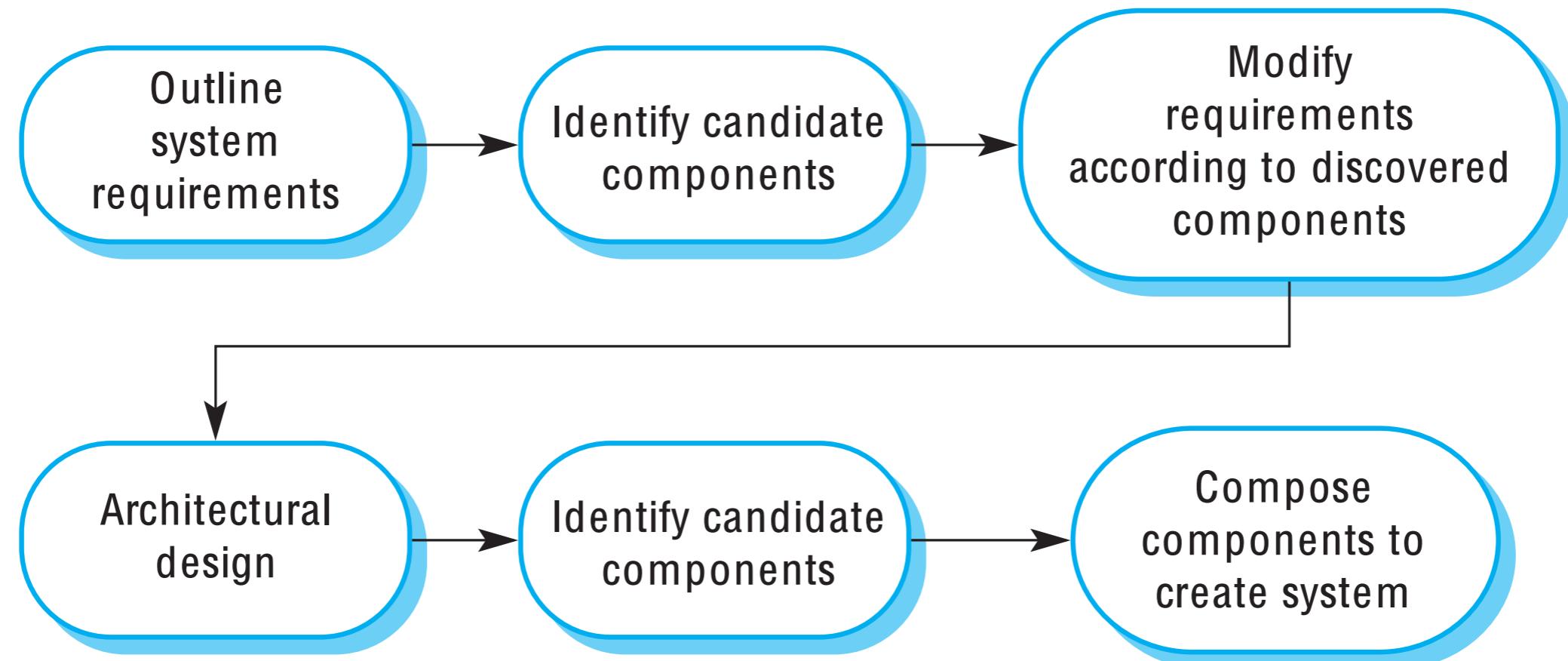
Riutilizzabilità dei componenti

- ~ La gestione dei componenti comporta la decisione di come classificare il componente in modo che possa essere individuato
 - renderlo disponibile in un repository (GitHub o Sourceforge) o come servizio
 - mantenere informazioni sull'uso del componente e di tenere traccia delle diverse versioni del componente.
- ~ Un'azienda con un programma di riutilizzo può effettuare una forma di certificazione del componente prima che questo sia reso disponibile per il riutilizzo.
 - La certificazione significa che qualcuno, oltre allo sviluppatore, controlla la qualità del componente.

CBSE con il riuso

- ~ Il CBSE con il processo di riutilizzo deve trovare e integrare componenti riutilizzabili.
- ~ Quando si riutilizzano i componenti, è essenziale fare dei compromessi tra i requisiti ideali e i servizi effettivamente forniti dai componenti disponibili.
- ~ Ciò comporta:
 - & Sviluppare requisiti di massima;
 - & Ricerca di componenti e modifica dei requisiti in base alle funzionalità disponibili.
 - & Ricercare nuovamente se ci sono componenti migliori che soddisfano i requisiti modificati.
 - & Comporre i componenti per creare il sistema.

CBSE con il riuso





CBSE con il riuso

- ~ i requisiti utente vengono identificati a grandi linee
- e i requisiti troppo specifici limitano il riutilizzo
- ~ i requisiti vengono perfezionati e modificati a seconda dei componenti disponibili
- ~ ulteriore attività di ricerca e perfezionamento dei componenti dopo che l'architettura è stata definita
- ~ integrazione dei componenti nell'infrastruttura

CBSE con il riuso

- ~ Primo passo: cercare i componenti disponibili all'interno dell'azienda
 - evita i rischi di uso dei componenti di altri fornitori
- ~ poi selezionare quelli più appropriati
 - alcuni soddisferanno direttamente uno o più requisiti
- ~ convalidare e verificare come si comportano i componenti

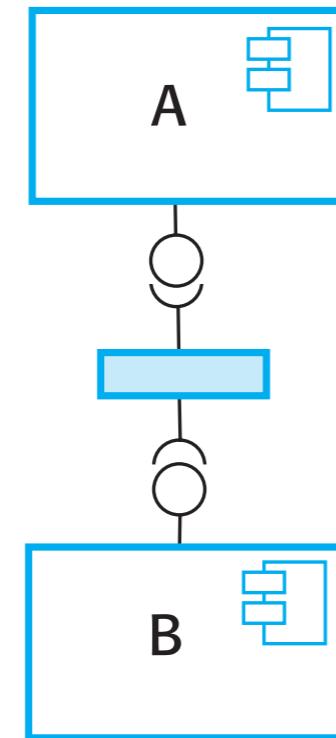


Composizione dei Componenti

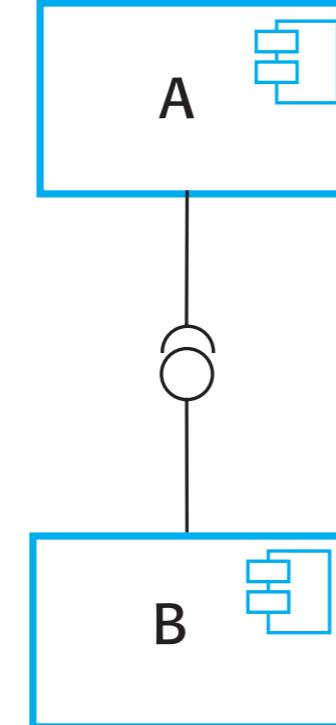
Composizione dei componenti

- ~ Il processo di assemblaggio dei componenti per creare un sistema.
- ~ La composizione comporta l'integrazione dei componenti tra loro e con l'infrastruttura dei componenti.
- ~ Normalmente è necessario scrivere del "codice collante" per integrare i componenti.

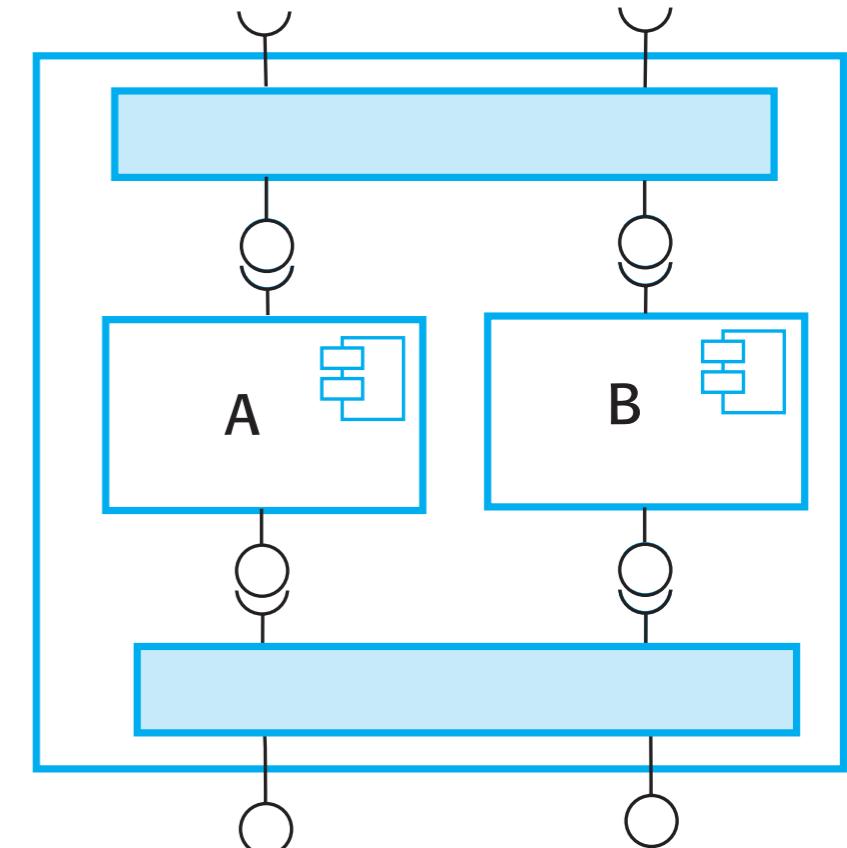
Tipi di composizione



(1)



(2)



(3)

~ Sequenziale

(1)

~ Gerarchica

~ Additiva

Tipi di composizione

- ~ Composizione sequenziale (1), in cui i componenti composti vengono eseguiti in sequenza. Ciò comporta la composizione delle interfacce di ciascun componente.
- ~ Composizione gerarchica (2), in cui un componente richiama i servizi di un altro. L'interfaccia provides di un componente viene composta con l'interfaccia requires di un altro.
- ~ Composizione additiva (3), in cui le interfacce di due componenti vengono messe insieme per creare un nuovo componente. Le interfacce provides e requires di un componente integrato sono una combinazione delle interfacce dei componenti costitutivi.



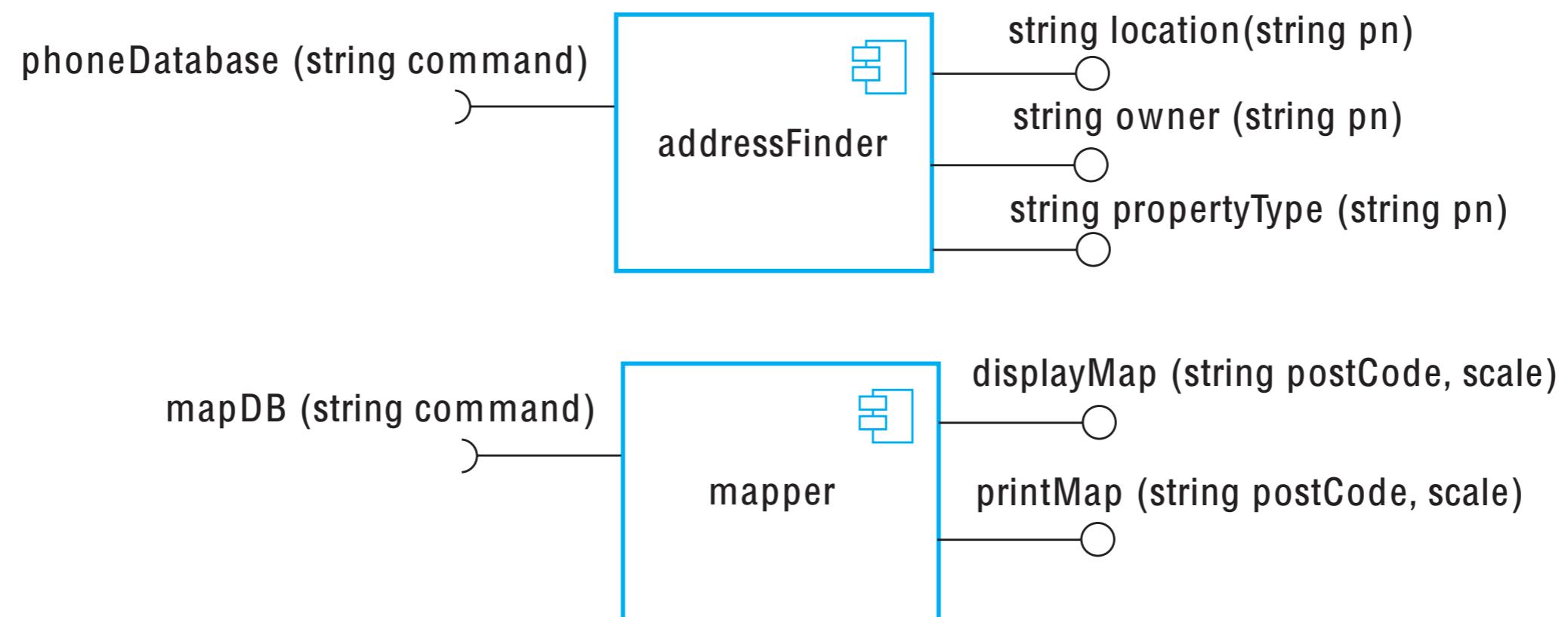
Codice colla

- ~ Codice che permette ai componenti di lavorare insieme
- ~ Se A e B sono composti in modo sequenziale, il codice collante deve chiamare A, raccogliere i suoi risultati e poi chiamare B usando questi risultati, trasformandoli nel formato richiesto da B.
- ~ Il codice colla può essere utilizzato per risolvere le incompatibilità di interfaccia.

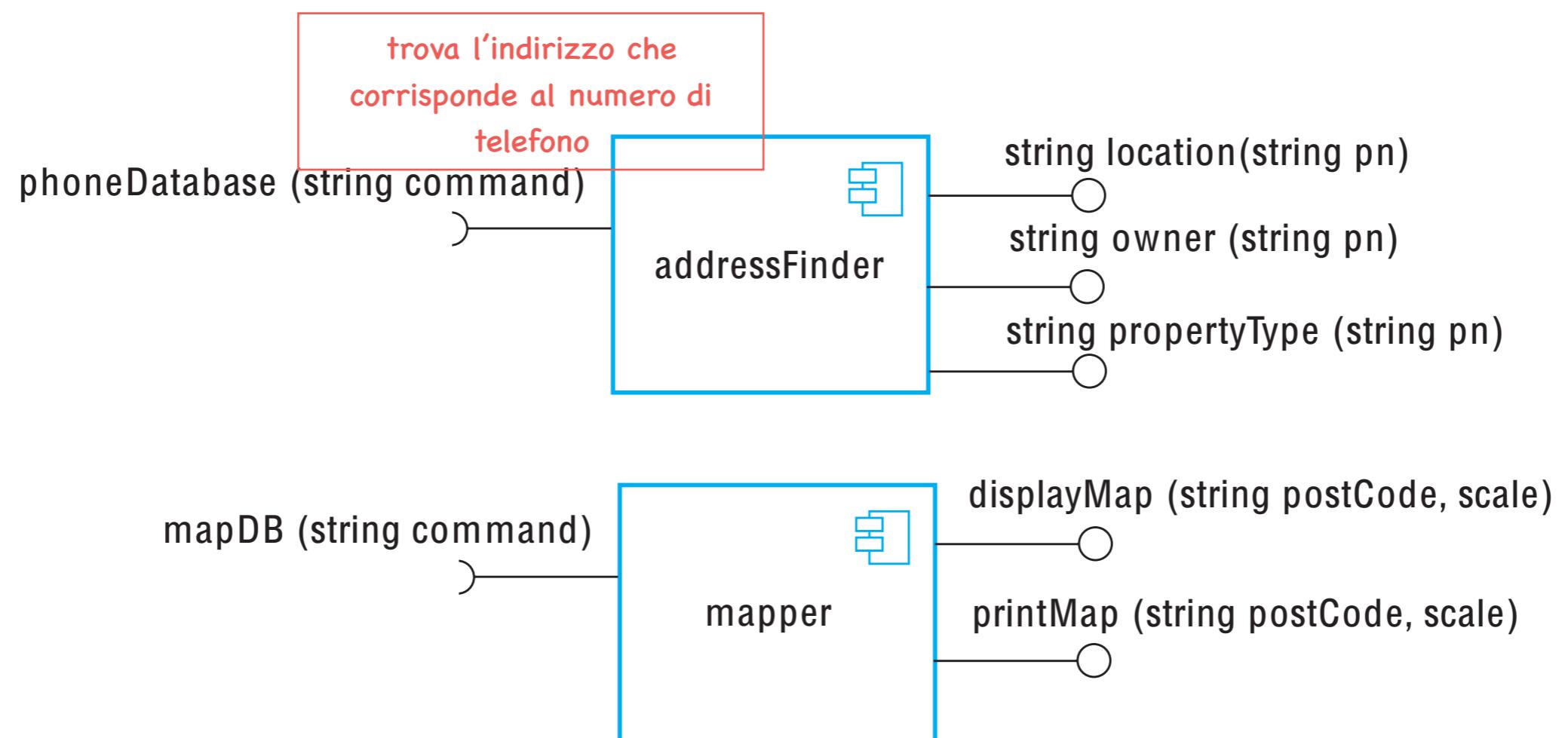
Incompatibilità di interfaccia

- ~ Incompatibilità dei parametri, quando le operazioni hanno lo stesso nome ma sono il tipo o il numero dei parametri sono diversi.
- ~ Incompatibilità delle operazioni, quando i nomi delle operazioni nelle interfacce composte sono diversi.
- ~ Incompletezza delle operazioni, quando l'interfaccia provides di un componente è un sottoinsieme dell'interfaccia requires di un altro.

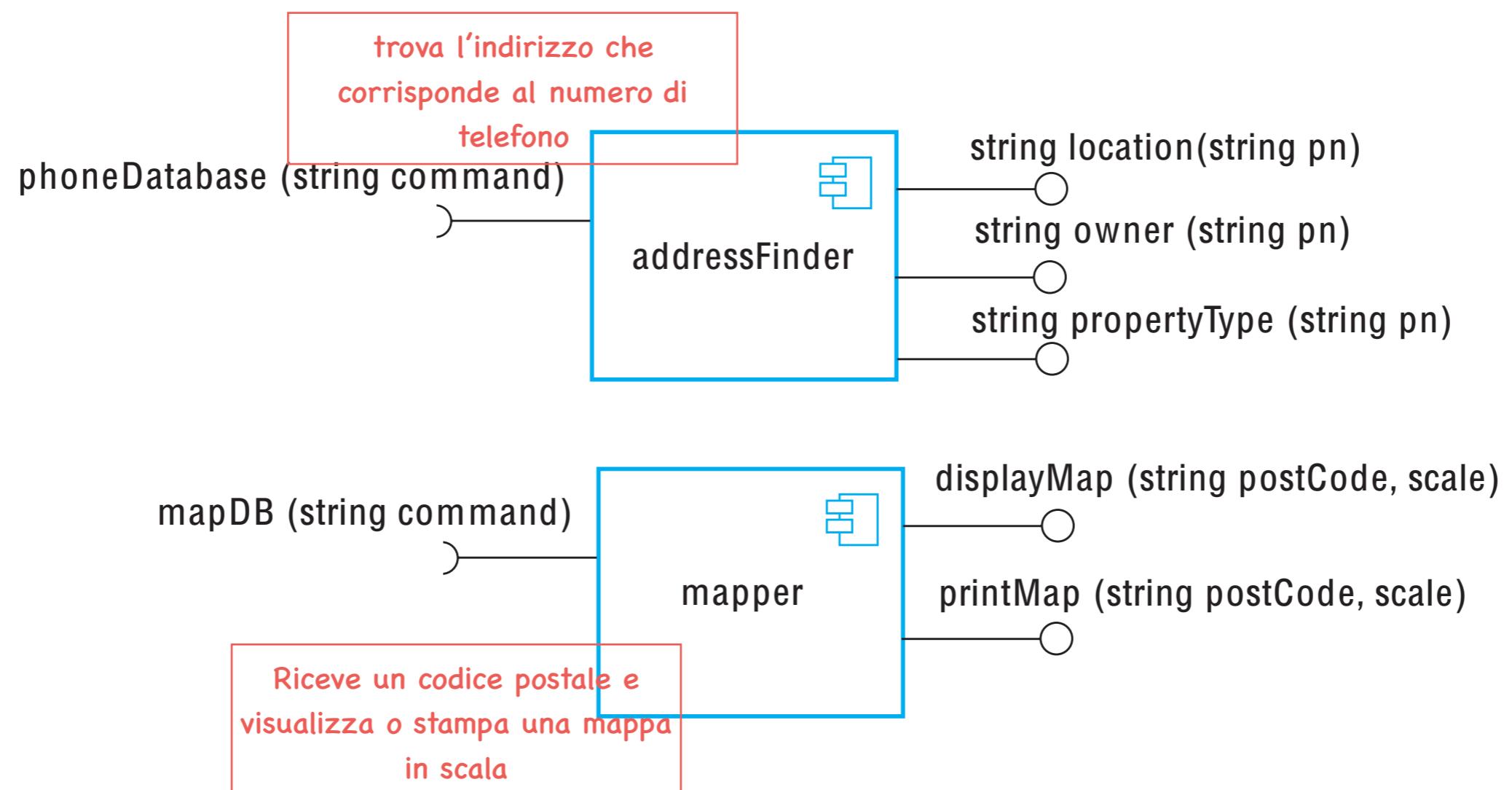
Incompatibilità di interfaccia



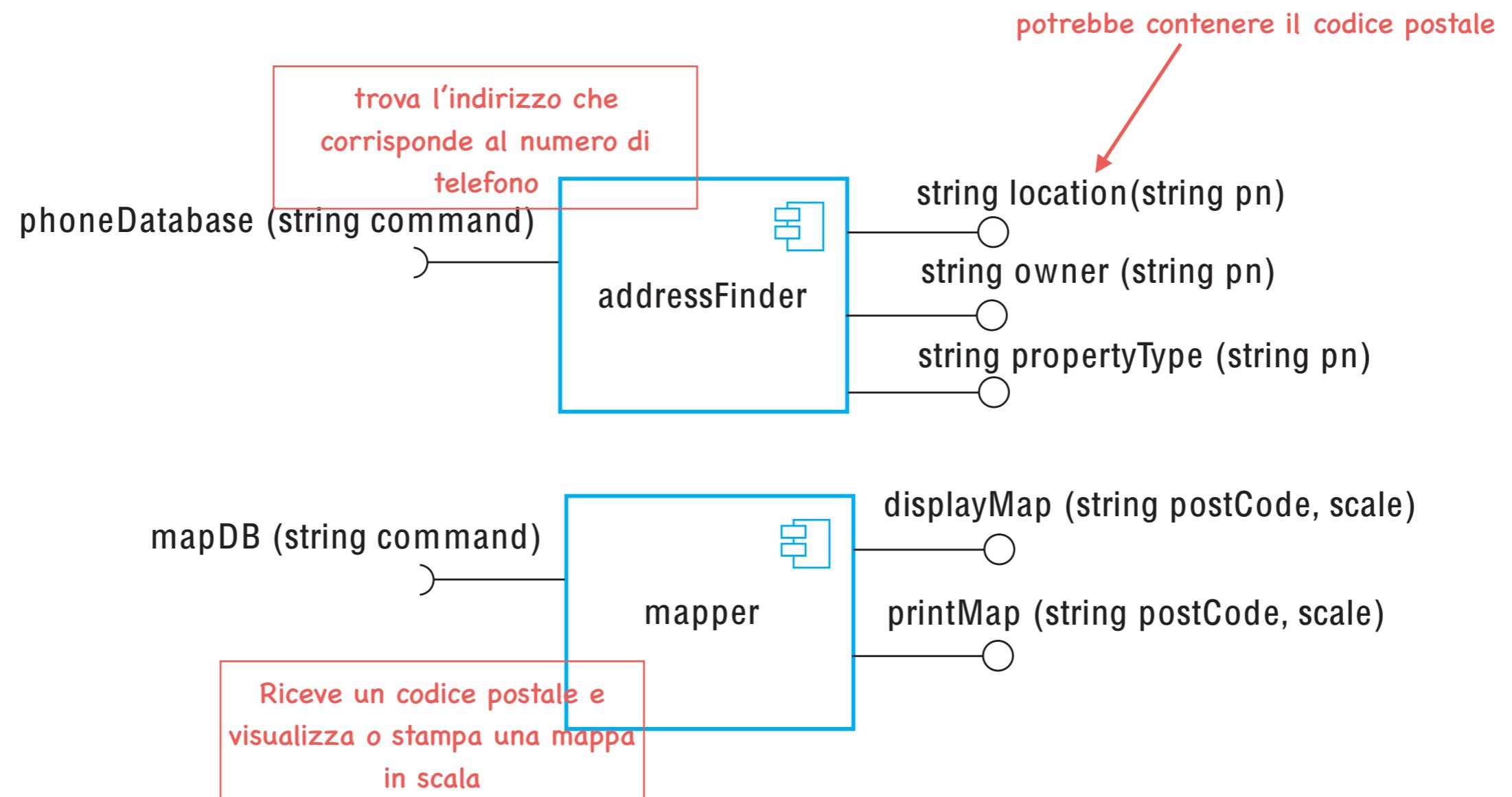
Incompatibilità di interfaccia



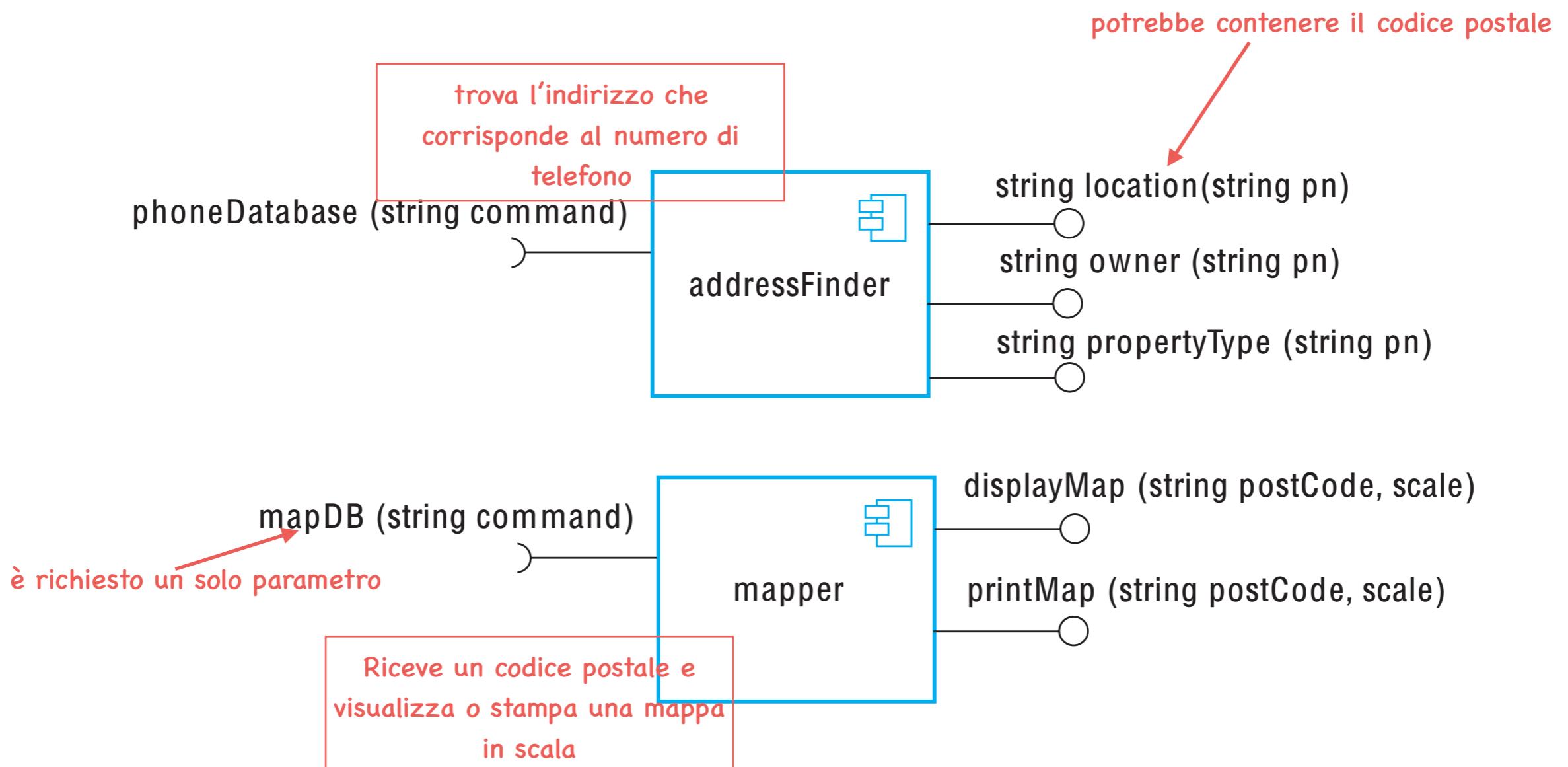
Incompatibilità di interfaccia



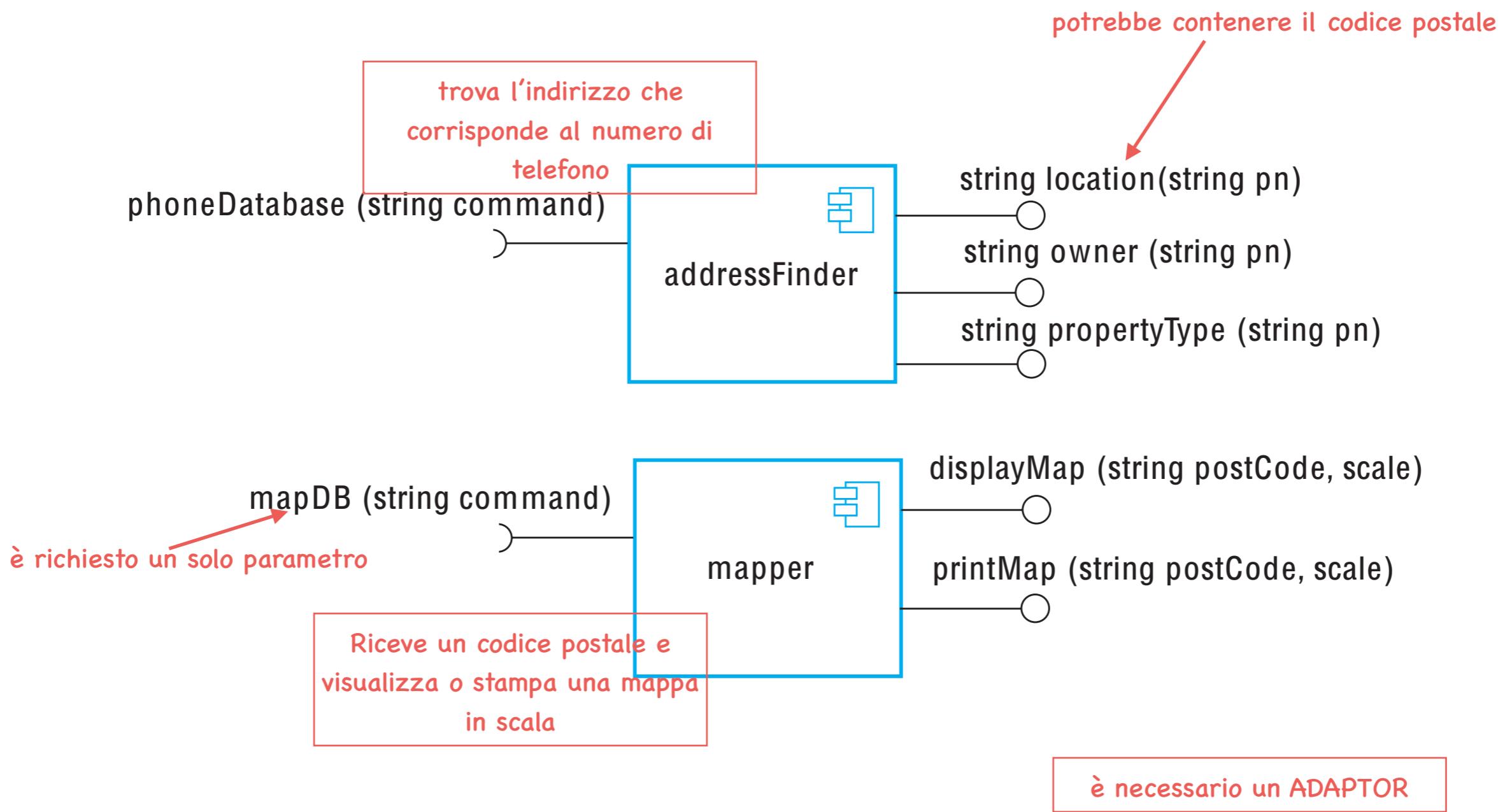
Incompatibilità di interfaccia



Incompatibilità di interfaccia



Incompatibilità di interfaccia





Componente ADAPTOR

- ~ Affronta il problema dell'incompatibilità dei componenti riconciliando le interfacce dei componenti che vengono composti.
- ~ A seconda del tipo di composizione, sono necessari diversi tipi di adattatori.
- ~ Un addressFinder e un componente mapper possono essere composti attraverso un adattatore che toglie il codice postale da un indirizzo e lo passa al componente mapper.

Composizione attraverso gli ADAPTOR

- ~ Il componente `postCodeStripper` è l'adattatore che facilita la composizione sequenziale dei componenti `addressFinder` e `mapper`.

```
address = addressFinder.location (phonenumbers) ;  
postCode = postCodeStripper.getPostCode (address) ;  
mapper.displayMap(postCode, 10000)
```

Key points

- ~ IL CBSE è un approccio basato sul riutilizzo per definire e implementare nei sistemi i componenti ad accoppiamento libero.
- ~ Un componente è un'unità software la cui funzionalità e le cui dipendenze sono completamente definite dalle sue interfacce.
- ~ I componenti possono essere implementati come elementi eseguibili inclusi in un sistema o come servizi esterni.
- ~ Un modello di componente definisce un insieme di standard che i fornitori e i compositori di componenti devono seguire.
- ~ I processi CBSE chiave sono
 - CBSE per il riutilizzo e CBSE con riutilizzo.

Key points

- ~ Durante il processo CBSE, i processi di ingegneria dei requisiti e di progettazione del sistema si intrecciano.
- ~ La composizione dei componenti è il processo di "cablaggio" dei componenti per creare un sistema.
- ~ Quando si compongono componenti riutilizzabili, di solito si devono scrivere adattatori per conciliare le diverse interfacce dei componenti.
- ~ Quando si scelgono le composizioni, bisogna considerare le funzionalità richieste, i requisiti non funzionali e l'evoluzione del sistema.