Príprava na skúšku z ASS

Zoznam vzorov a štýlov z prednášok na základe mailu od Poláška

Pre každý vzor treba:

- Obrázok
- Popis

Všetky opisy z POSA 4
http://cl.ly/1f322B0C0C1M
Prípadné nezrovnalosti naštudovať samostatne :)

P.

Štýly

starý doc na zápočet
https://docs.google.com/document/d/1tvY-lT7UvyimHhYdf0n4L3_jp6M2WLB4lY4MDEV6t40/edit#heading=h.d160mye1ekfn

Vzory

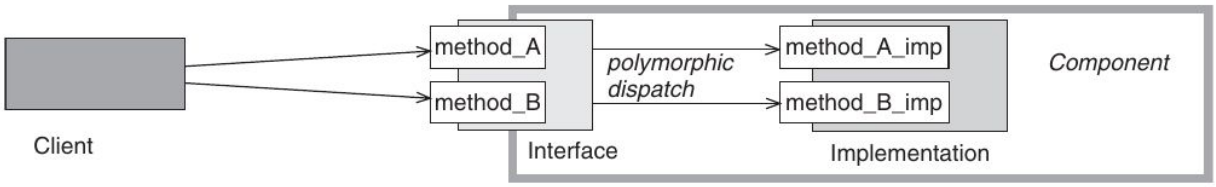
Distribution Infrastructure
Event Demultiplexing and Dispatching
Interface Partitioning
Component Partitioning
Application Control
Object Interaction
Adaptation and Extension
Modal Behaviour
Resource Management
MVC, Microkernel, Reflection
J2EE

Bonus
Database Access

Component Partitioning

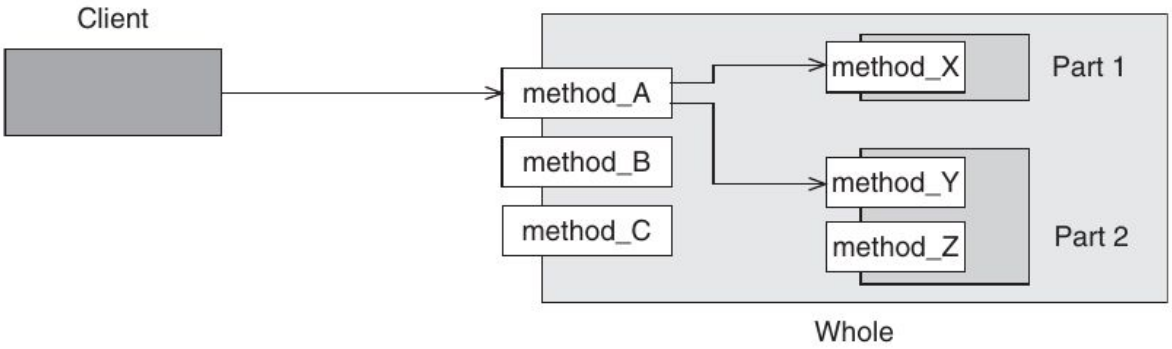
(ASS13POSA)

Encapsulated Implementation



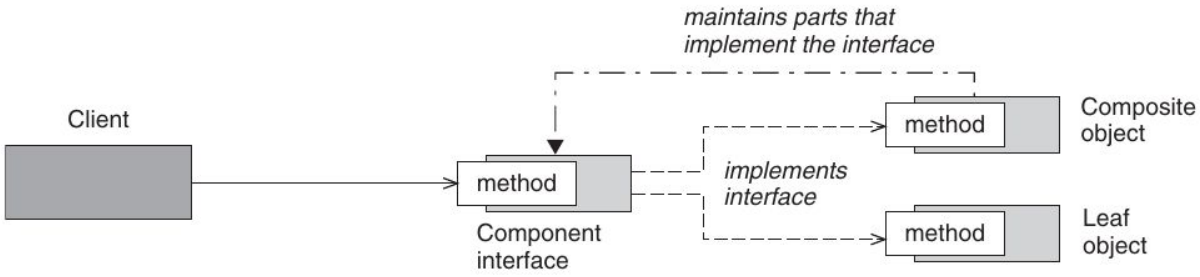
- rozhranie definuje poskytované služby a vlastnosti služieb
- skrývanie implementácie, konfigurácie, vytvárania častí, …

Whole Part



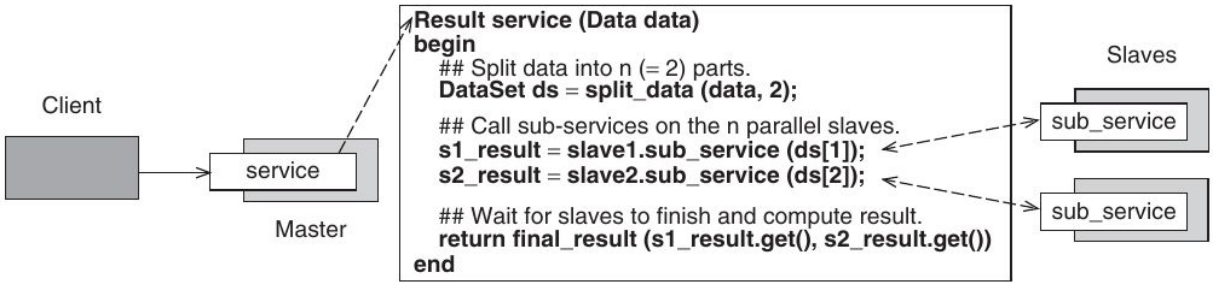
- zapuzdruje viac kompomentov, ktoré nemusia mať spoločné rozhranie
- zoskupenie jednotlivých častí ako jeden celok
- klienti nemajú info, ktorá časť je volaná, vedia iba o rozhraní

Composite



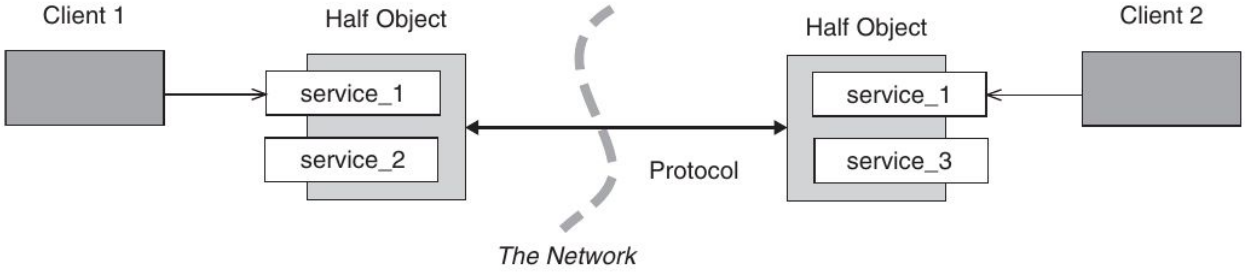
- komponent predstavuje hierarchiu zloženú z objektov podobných (pod)typov
- prístup k objektu alebo k hierarchii objektu je cez rovnaké rozhranie
- rekurzívne vyskladaná štruktúra
- štruktúra a prístup ostáva stabilná pri pridávaní ďalších častí

Master-Slave



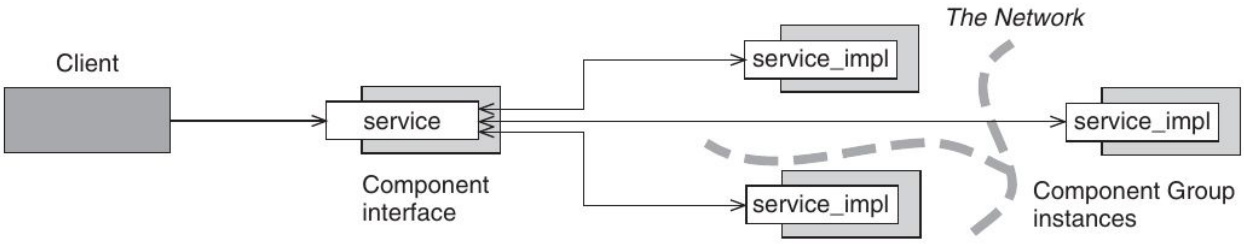
- delenie práce na menšie časti a poslanie na slaves
- možné spracovanie paralelne
- master čaká na výsledky, ktoré potom spracuje

Half-Object + Protocol



- usporiadanie logicky súvisiacich objektov vo viac adresnom priestore
- delenie na “pol-objekty” alebo “iný zlomok-objekty”, pre každý uzol časť objektu
- každá časť implementuje istú funkcionalitu
- objekt spracuje svoju časť a ak sa vyžaduje ďalšie spracovanie, vyvolá iný objekt cez sieť

Replicated Component Group

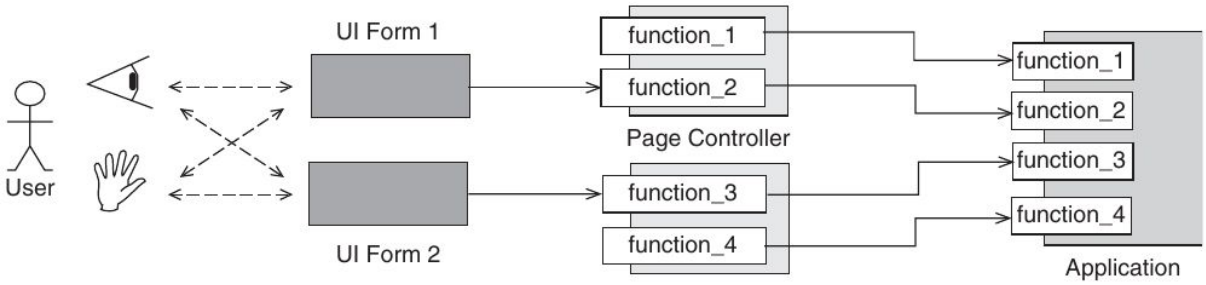


- namnožená štruktúra komponentu na viacerých uzloch v sieti

- klienti interagujú so skupinou cez jeden bod, ako keby bol komponent logicky jednotný
- poskytuje dostupnosť služby
- požiadavka je cez prístupový bod poslaná na všetky uzly a čaká sa na výsledok od nejakého uzla

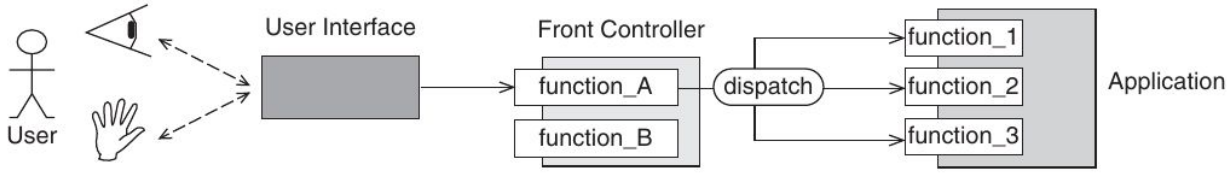
Application Control

Page Controller



- vstupný bod pre každý form UI
- spracovanie a vykonávanie requestov z form-ov a transformuje pre potreby aplikačnej vrstvy
- pre každý form môže samostatný controller

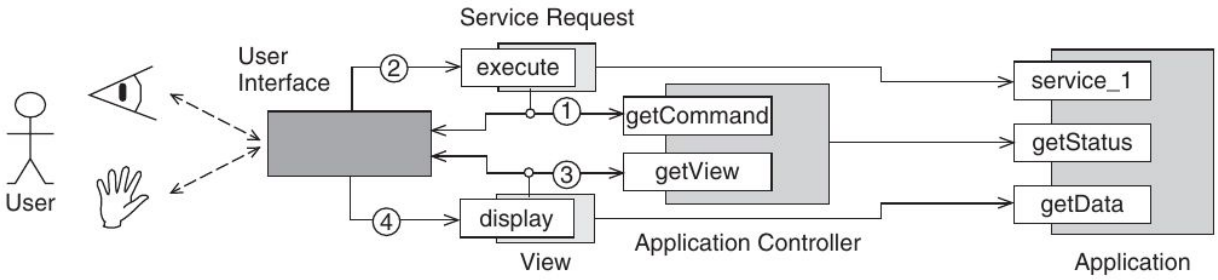
Front Controller



- jednotný vstupný bod aplikácie
- spracovávanie a vykonávanie requestov z UI
- vyvolá vyžadovanú funkcionalitu (často rovnakú pre každý request) a transformuje UI request do requestu pre aplikačnú vrstvu

Application Controller

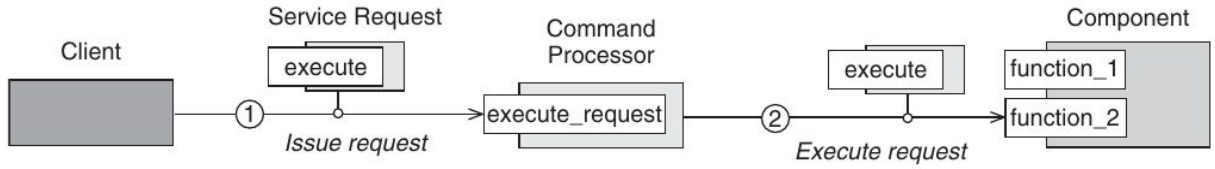
ii



- oddeľuje UI navigáciu od riadenia application workflow

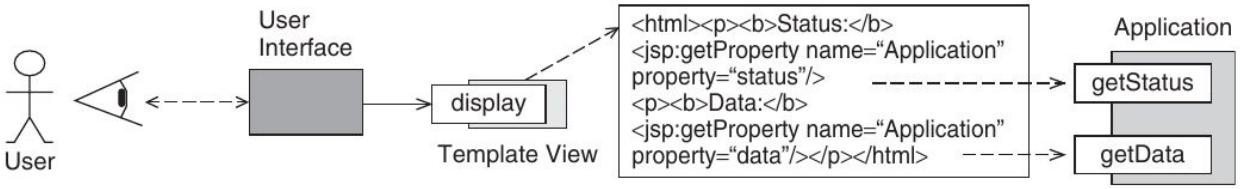
- prijíma requesty z UI a rozhoduje sa, ktoré služby vykoná na základe aktuálneho stavu workflow

Command Processor



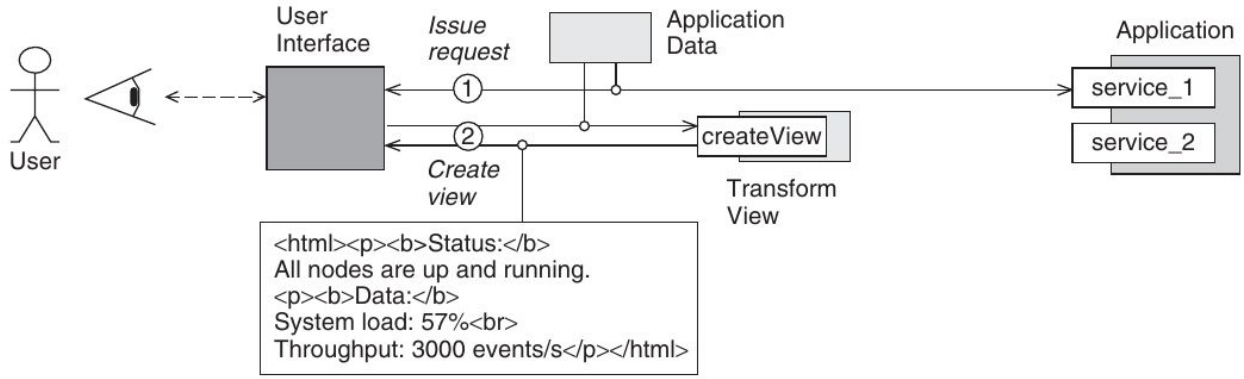
- oddeľuje request ako samostatný objekt
- naplánuje vykonanie, nemusí byť jasné, kedy sa má request vykonať
- možné doplniť o dalšiu funkcionalitu (log, undo, redo, …)

Template View



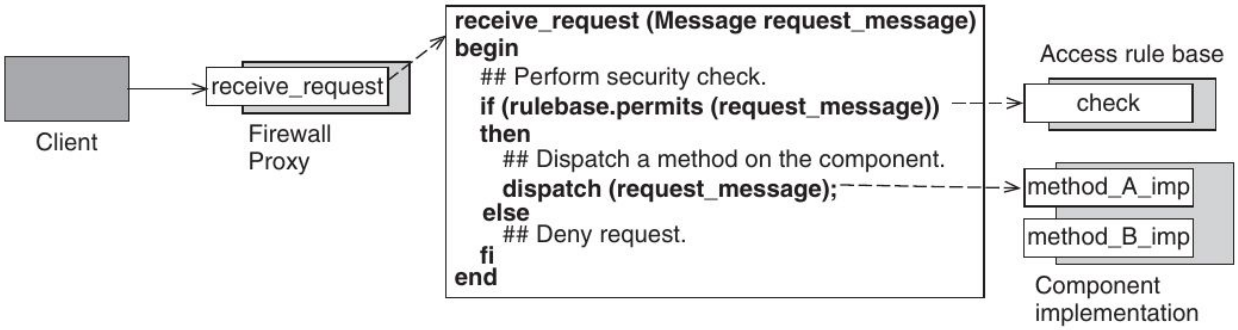
- preddefinovaný view pre každý view komponent
- pre dynamické časti
- spracovanie a zobrazenie variácií získaných dát v jednej implementácii

Transform View



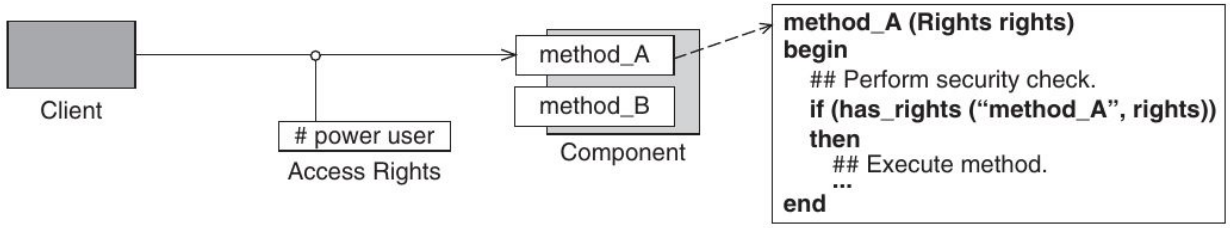
- dáta získané z aplikačnej vrstvy sú transformované na konkrétny form alebo view

Firewall Proxy



- chráni aplikáciu pred vonkajšími vplyvmi
- je to Proxy, ktorá kontroluje vstup, identifikuje a odstraňuje podozrivý obsah

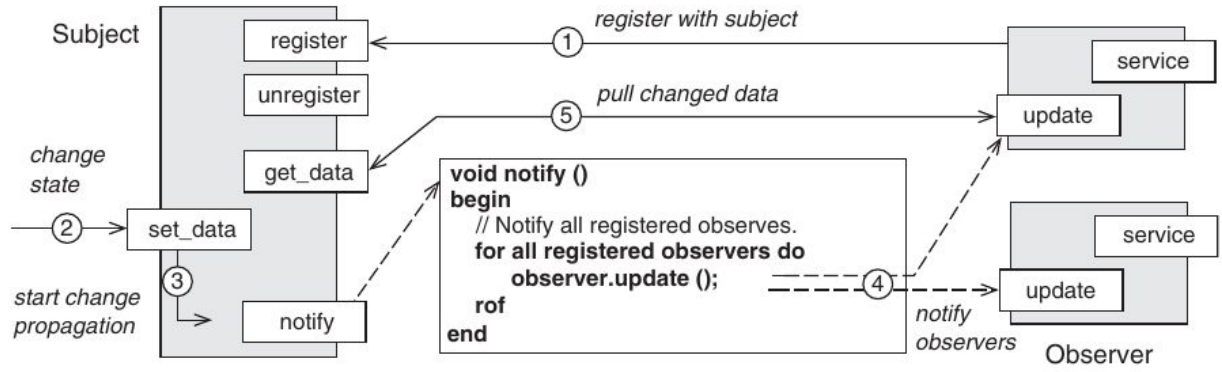
Authorization



- kontroluje vstupné práva, aby klient mohol vykonávať len určitú, pre neho určenú funkcionalitu

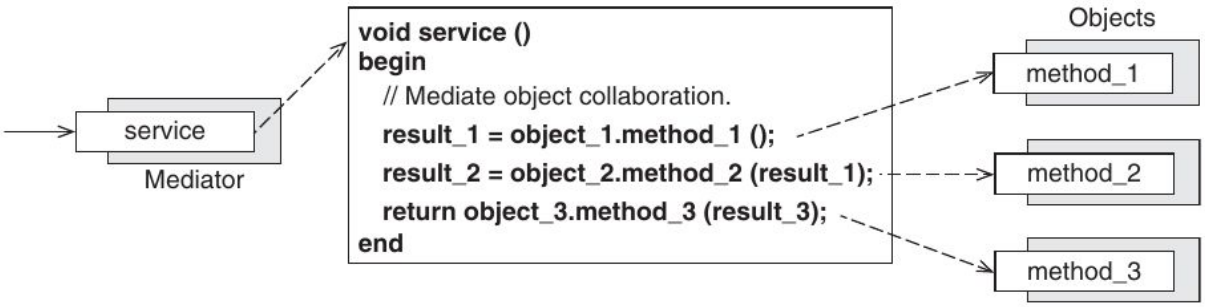
Object Interaction

Observer



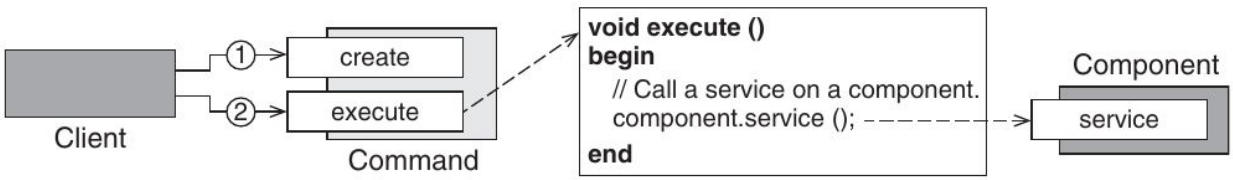
- synchronizovanie stavu kooperujúcich objektov - jednosmerné posielanie zmeny stavu
- observery sa registrujú k objektu, od ktorého berú informácie o stave
- objekt notifikuje všetky registrované observery, ak nastane zmena jeho stavu

Mediator



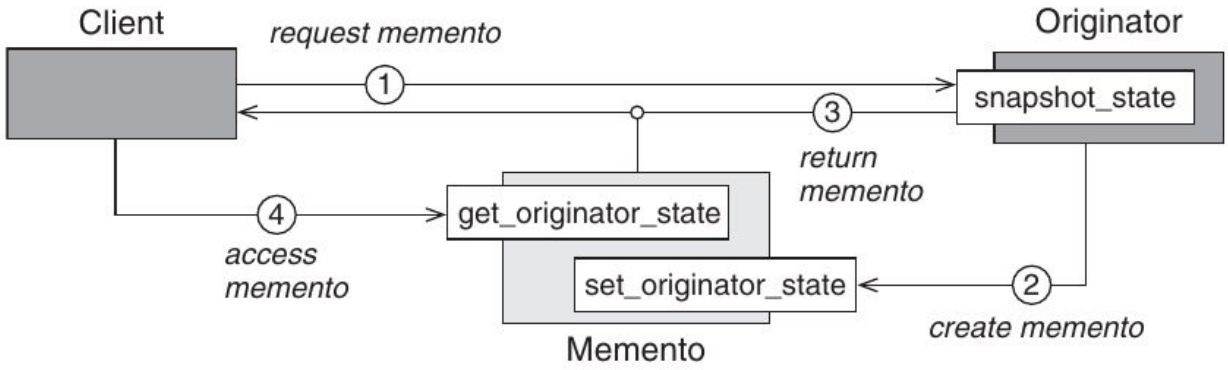
- zapuzdrenie, akým spôsobom interaguje množina komponentov
- jeden objekt spravuje vzťahy medzi objektami, namiesto vytvárania závislosti každý s každým

Command



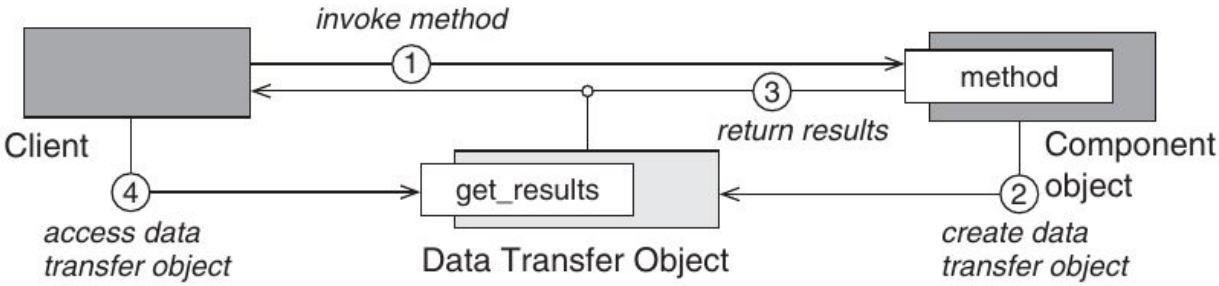
- zapuzdruje request ako objekt
- vyvolanie akcie nezávisle na komponente
- oddialenie volania

Memento



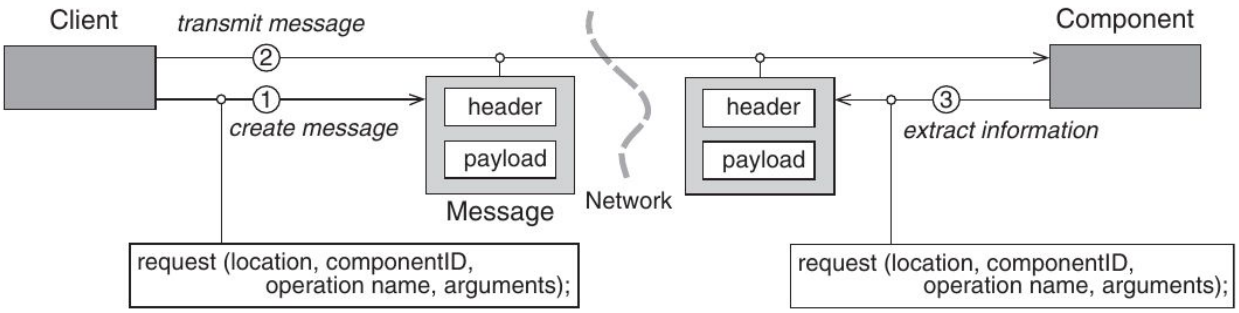
- zachytí stav objektu
- namiesto originálneho objektu sa pracuje s mementom
- s pôvodnym objektom sa môže pracovať inde

Data Transfer Object



- redukuje počet volaní a dopytov na vzdialené objekty
- obalí skupinu atribútov na jeden objekt, ktorý je posielaný jedným volaním
- vhodné aj pre nedistribuované systémy - zabraňuje malej granularite
- zmena DTO neovplyvní zmenu komponentov

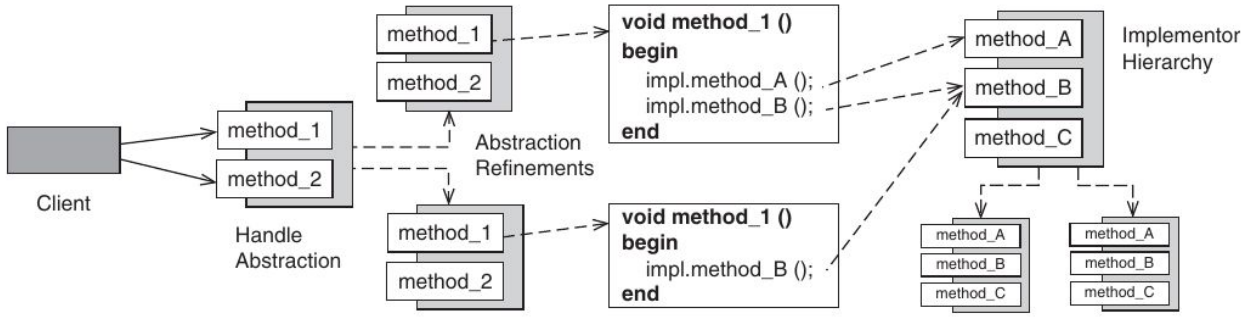
Message



- 2 komponenty aplikácie si vymieňajú informácie v jednej dátovej štruktúre naprieč sieťou
- spôsob vymieňania dát bez vytvorenia závislosti na konkrétne typy a rozhrania
- musí byť dohodnutý formát správy
- správa obsahuje hlavičku s informáciami o cieli, type a veľkosti správy ...

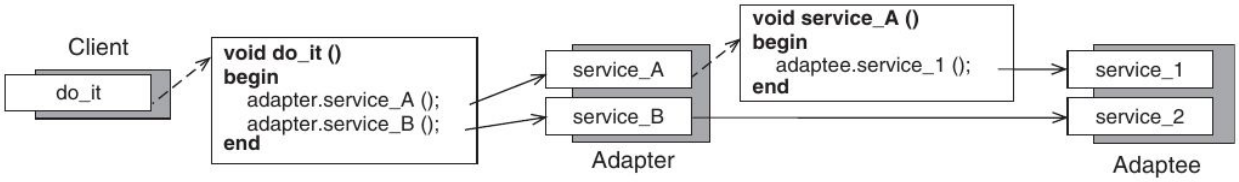
Adaptation & Extension

Bridge



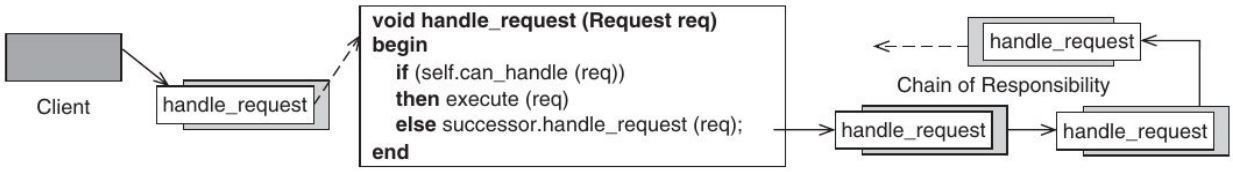
- variácie implementácií pre dané rozhranie na základe platformy alebo počas behu aplikácie
- rozhodovanie je skryté pred klientom

Object Adapter



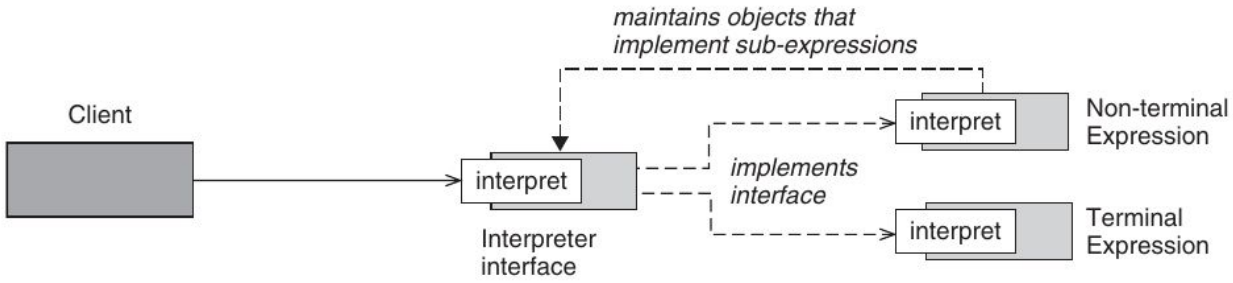
- jednotné rozhranie pre viacero existujúcich komponentov
- existujúce komponenty majú rôzne rozhrania
- výsledok volania adaptovaného objektu je upravený do návratového typu nového rozhrania
- zmena komponentu ovplyvní len Adapter

Chain of Responsibility



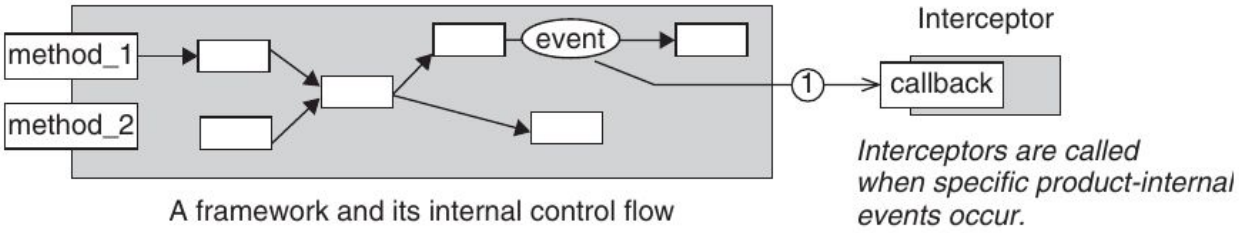
- veľa objektov, ktoré dokážu spracovať rovnaký request (vstup)
- klienta nezaujíma, ktorý konkrétny objekt vstup spracuje
- reťazenie objektov - každý objekt rozhodne, či vie spracovať vstup, ak nie vyvolá nasledovníka
- ak nikto nedokáže spracovať vstup -> chyba, prípadne **Null Object**

Interpreter



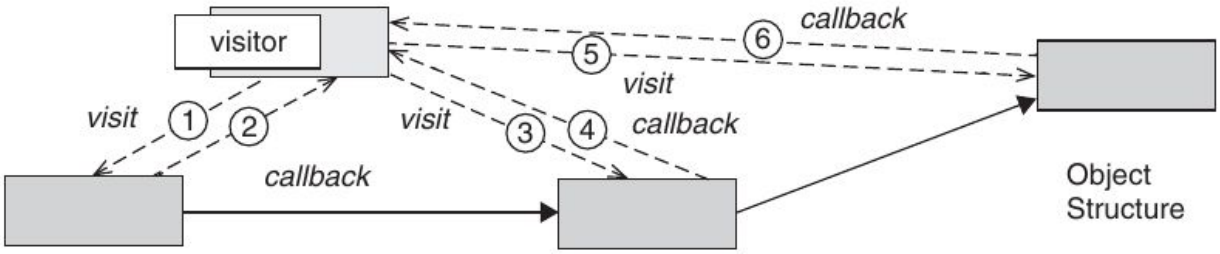
- spracovanie dát, skriptu, gramatiky a na základe toho vyvolanie nejakej služby

Interceptor



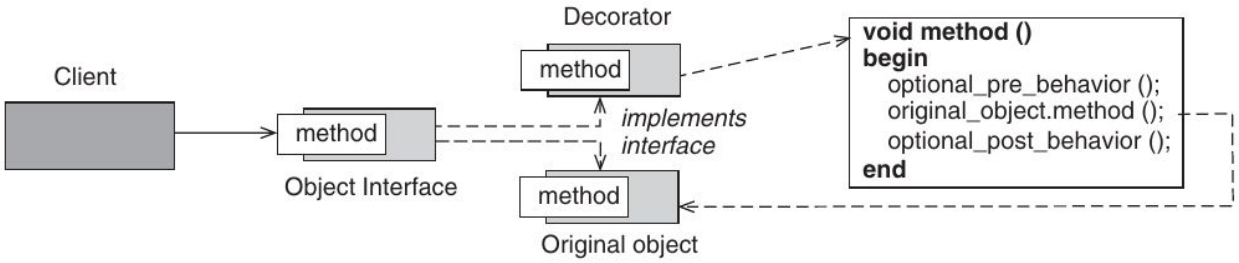
- pridanie nového správania bez upravovania existujúceho komponentu
- volanie, keď sa vyvolá istá udalosť
- registruje sa na daný komponent a je notifikovaný pri nastatí udalosti
- vyvoláva funkcionalitu mimo implementácie komponentu
- volanie interceptora nemusí nastať stále, ale len za určitých podmienok

Visitor



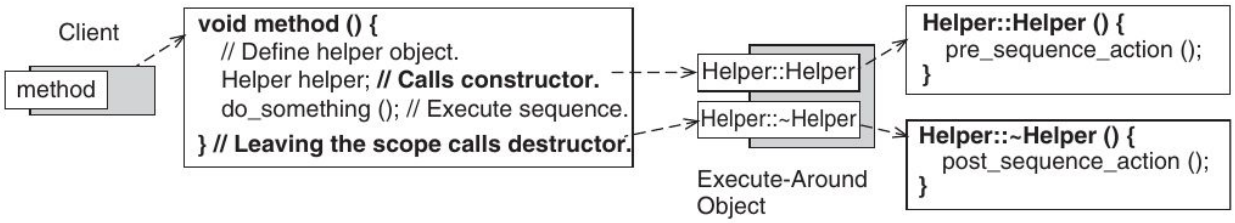
- služba operujúca nad agregovanou objektovou štruktúrou
- vyjadruje špecifiké správanie priradené pre každú triedu
- triedy akceptujú visitora zavolajú naspäť zodpovedajúcu metódu visit
- pri rožšírení o novú triedu sa upraví len visitor

Decorator



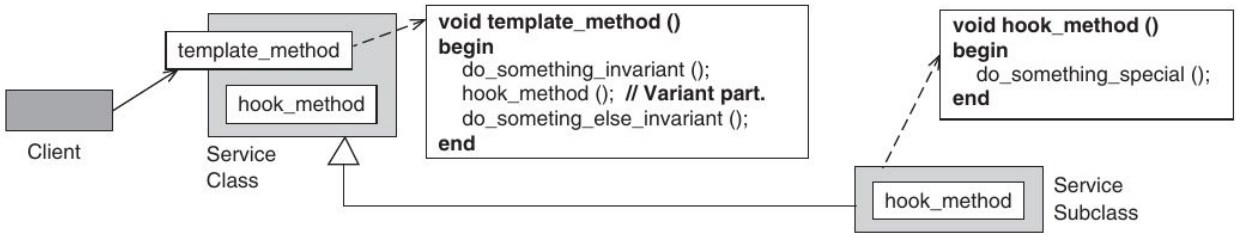
- rozšírenie zodpovednosti pre individuálny objekt
- zdieľa rovnaké rozhranie ako dekorovaný objekt
- doplní funkcionalitu a zavolá obaľovaný objekt
- možné kaskádové obaľovanie (rovnaké rozhranie)

Execute-Around Object



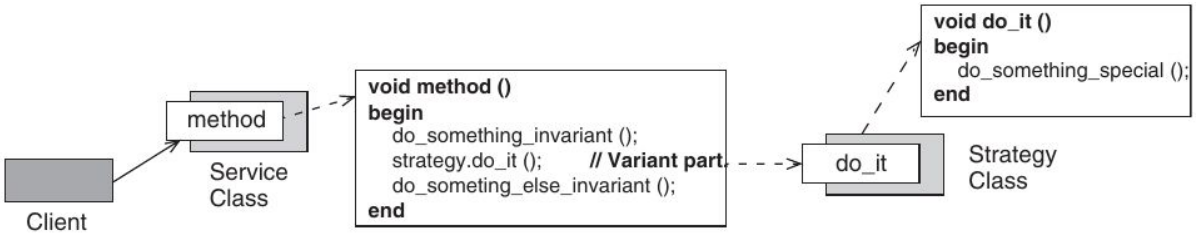
- C++ vzor
- vykonanie rovnakých akcií okolo sekvencie výrazov
- preprocessing/postprocessing ((de)alokácia pamäte)
- často ako Helper class
    - v konštuktore preprocessing
    - v deštruktore postprocessing
- Inymi slovami sa jedna o RAII
- helper object je na stack-u a teda volanie jeho konstruktora a destruktora je deterministicke, constructor pri vytvoreni objektu a destruktor je zavolany implicitne, ked sa object dostane mimo scopu. Toto poskytuje exception-safe spravu zdrojov, povedzme “automaticku spravu pamate” alebo automaticke uvolnenie lock-u ked uz nie je potrebny vylucny pristup (chyba programu, exception alebo dokoncenie prace)

Template Method



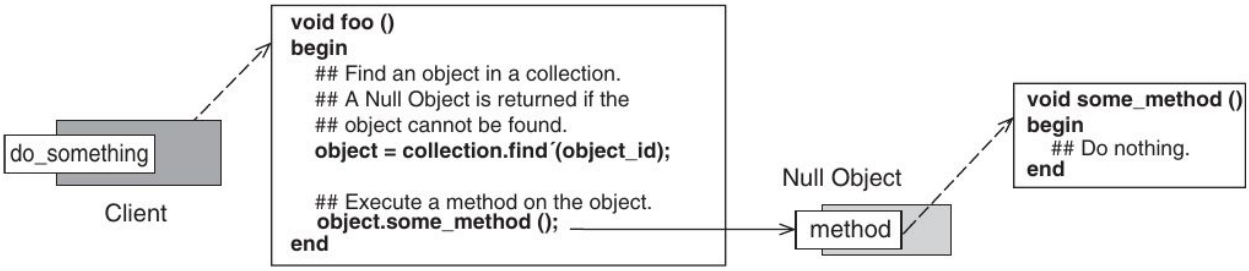
- objekty majú podobnú štruktúru a správanie, ale líšia sa v malej časti správania
- zabraňuje duplikovaniu kódu
- podtriedy doplňujú správanie definované v nadtriede
- definuje sa methoda, kde sa spravi core spravanie a potom sa tam spravi nejaky hook, ktory sa zavola, v pripade nadtriede to moze byt prazdna metoda, potom v podtriede sa tato metoda overidne a mame obohatenu metodu, problem je ale ten, ze temlate method by mala byt nepolymorficka a teda nedala by sa overridnut inak toto cele straca vyznam

Strategy



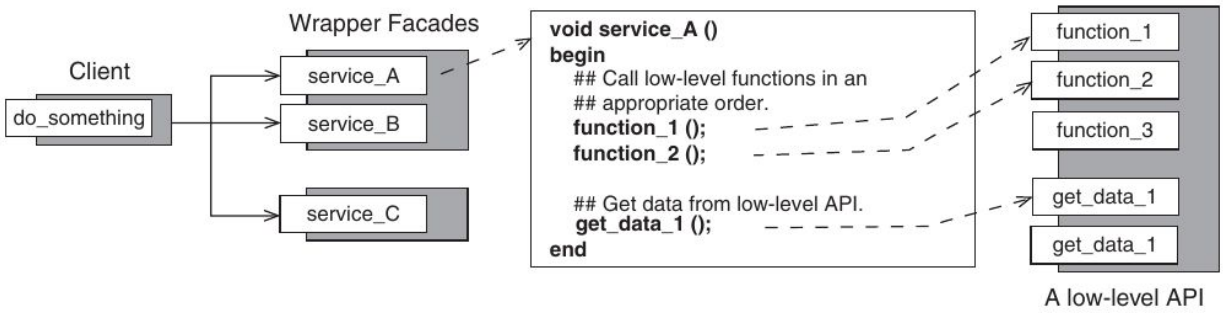
- objekty majú podobnú štruktúru a správanie, ale líšia sa vo viac častiach
- správanie sa líši od prípadu k prípadu
- zabraňuje duplikovaniu kódu
- služba definuje kód s istým správaním, v ktorej sa volá samostatná stratégia, ktorá vyjadruje odlišné správanie

Null Object



- vyjadrenie správania, keď “nič” nenastane
- namiesto **null** sa vráti prázdny alebo špeciálny objekt (default value)

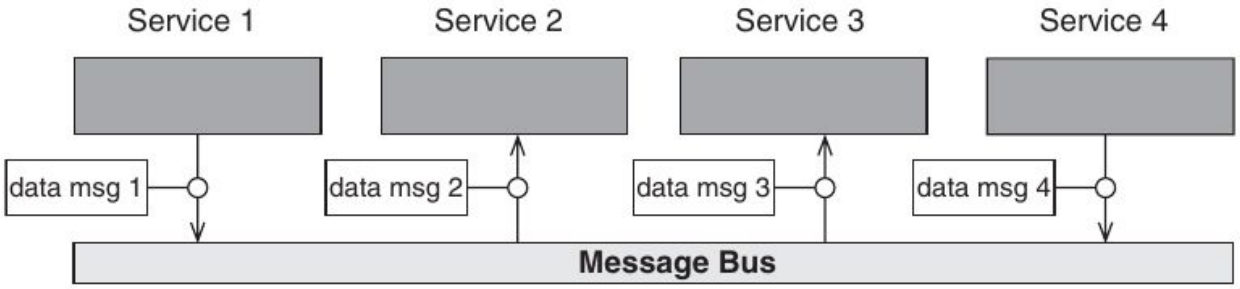
Wrapper Facade



- jednotné rozhranie pre prístup k nízko-úrovňovým API
- kód API by musel byť veľmi previazaný s ostatným kódom aplikácie
- prístup k API funkciám nepriamo cez fasádu
- chyby vratené z API sú transformované do zodpovedajúceho jazyka
- napriklad gtkmm pre gtk

Distribution Infrastructure

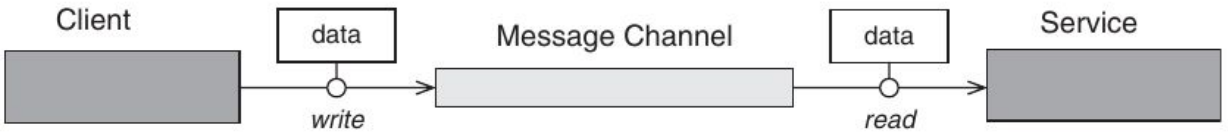
Messaging



- služby distribovaného systému si vymieňajú informácie <u>posielaním správ</u>
- správa ma svoj formát - obsahuje aj metadáta
- komunikácia N:1
- cez nejaký middleware, infraštruktúru

Tieto 4 veci môžme nazvať vzory - sú to realizácie Messaging-u

1. ***Message Channel***

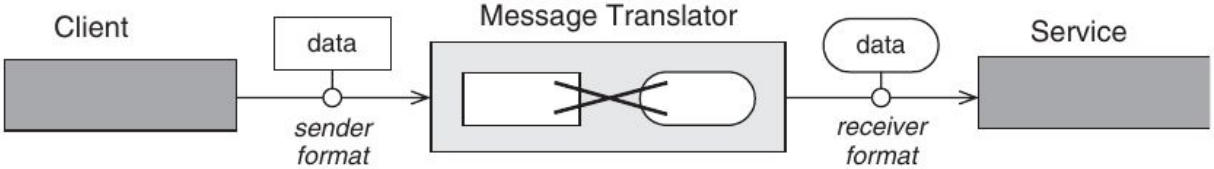


   - vytvorenie infraštruktúry - kanála, ktorým si vymieňajú správy
   - spoľahlivá komunikácia

2. ***Message Endpoint***

   - vytvorenie prístupového bodu, ktorý zapuzduje informácie dôležité pre vytvorenie spojenia
   - transformácia dát do správy sa vykonáva mimo klienta

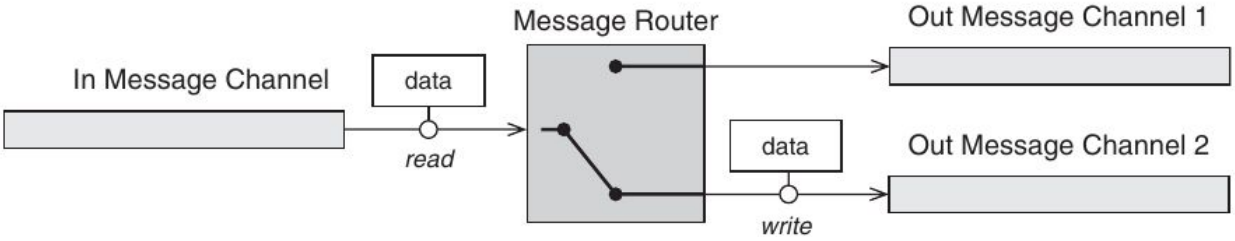
3. ***Message Translator***



   - odosielateľ a prijímateľ využívajú rôzne formáty správ
   - prekladá správy do správych formátov

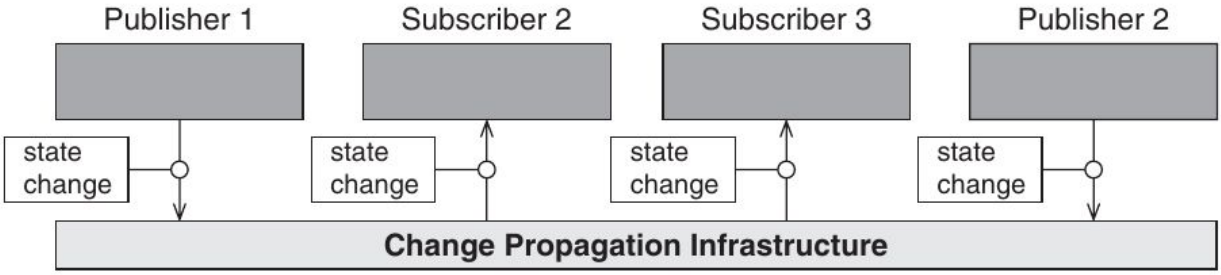
- odstraňuje sa, že by vysielateľ/prijímateľ musel v sebe obsahovať formáty každej správy pre každý uzol

4. Message Router



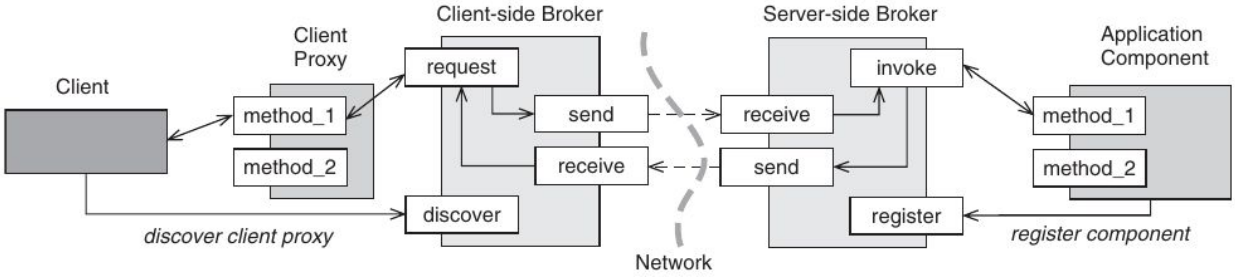
- posielanie správ medzi službami na základe rôznych podmienok siete
- router rozhoduje, akou cestou je správa preposlaná na koncový uzol
- prijímateľ ani vysielateľ správy nemá informáciu, ako sa správa pošle/prijme

Publisher-Subscriber



- založené na udalostiach
- komunikácia 1:N
- komponenty nemajú o sebe informácie, kto vyslal správu, ani kto ju spravoval
    - publisher pošle správu na všetkých a očakáva, že niekto správu spracuje, ale nie je dôležité kto
    - subscriber očakáva, že raz niekto vyšle správu, ale nie je dôležité kto
- prijatie správy nemusí implikovať okamžité vyvolanie udalosti

Broker

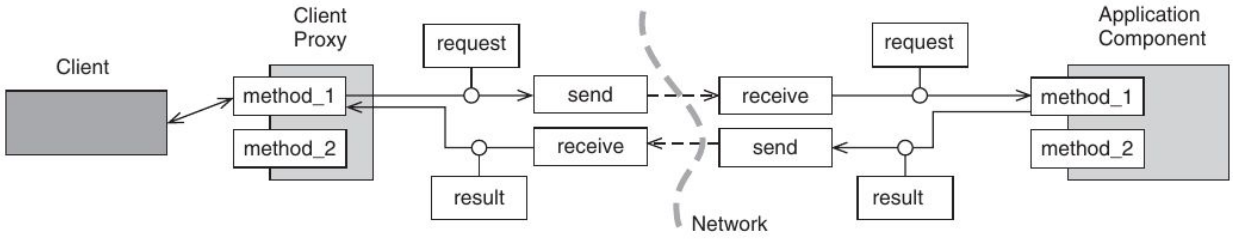


- komunikácie medzi komponentami vzdialeným volaním metód
- komunikácia 1:1

- model komunikácie pre komponenty nezávislé od prog. jazyka a operačného systému

Tieto veci sú vzory, realizácia Broker-a

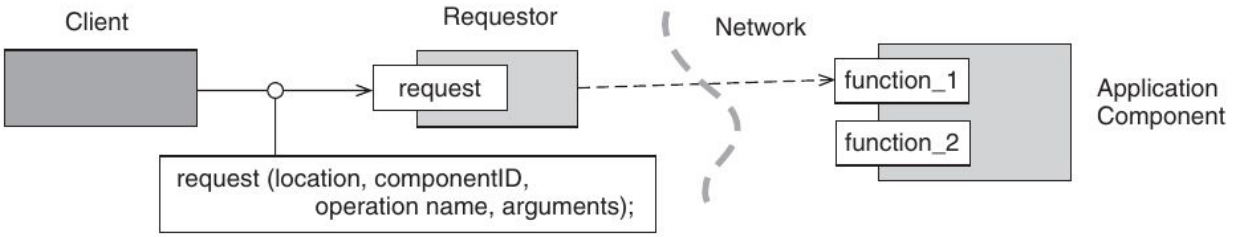
1. *Client Proxy*



- nahradzuje priamy prístup k vzdialenému komponentu
- odstraňuje závislosť na formáte a používanom protokole u klienta
- rovnaké rozhranie ako vzdialený komponent

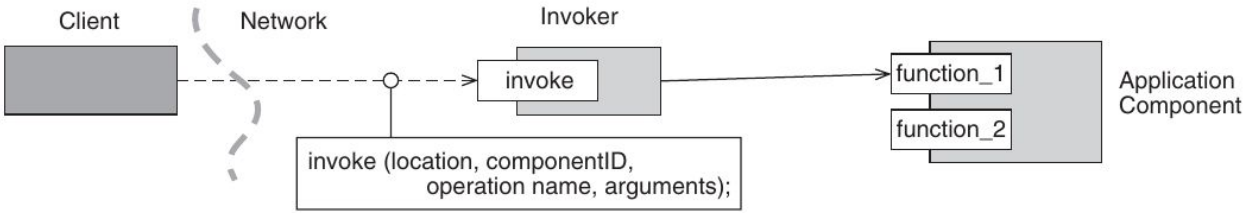
2.

Requestor



- vyvoláva požiadavky na vzdialený komponent, ktorý ma vykonať určité správanie na základe ním definovaných parametrov

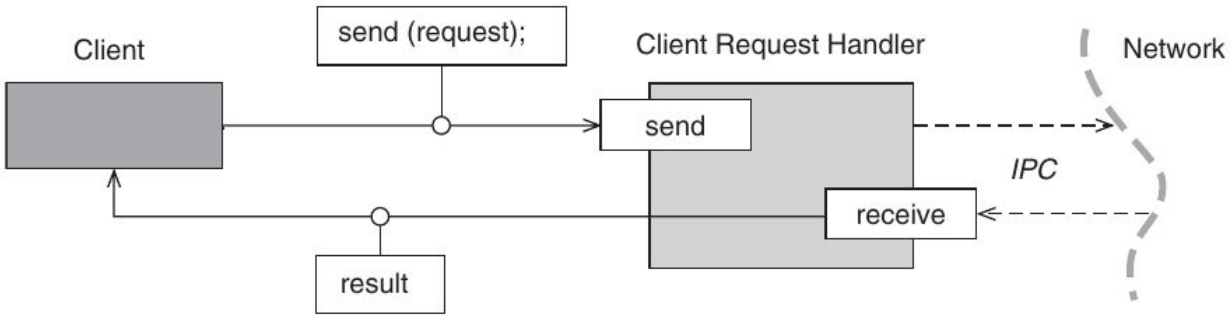
Invoker



- vykonáva konkrétnu operáciu na konkrétnom komponente na základe požiadaviek prijatých zo vzdialeného komponentu

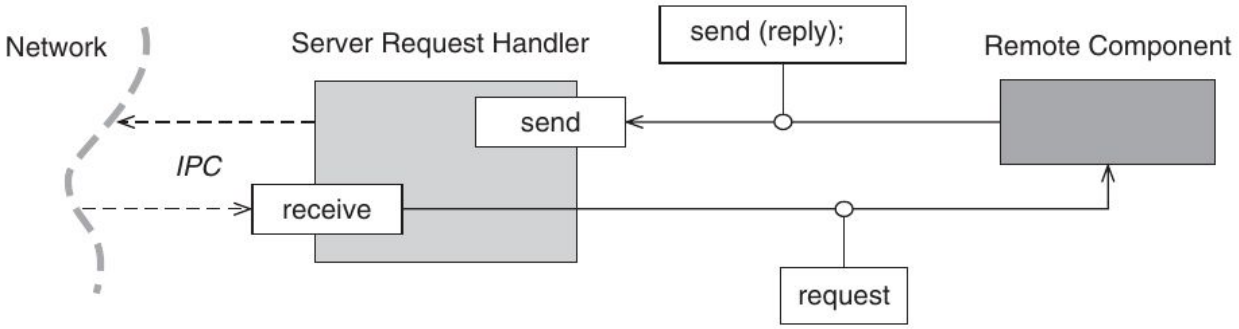
3.

Client Request Handler



- zapuzdruje a vykonáva všetky funkcie, ktoré sú potrebné pre poslanie správy z klientskej strany a na prijatie odpovede zo siete

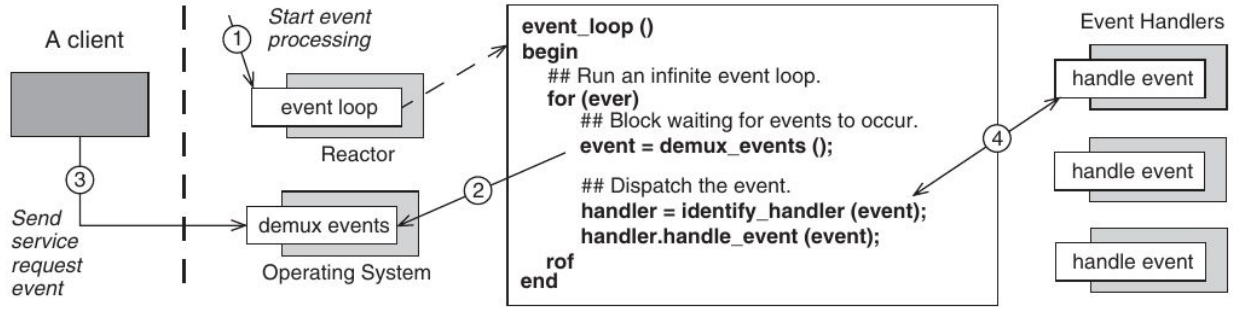
Server Request Handler



- zapuzdruje a vykonáva všetky funkcie, ktoré sú potrebné pre prijatie požiadaviek na serverovej strane a poslanie odpovede späť do siete

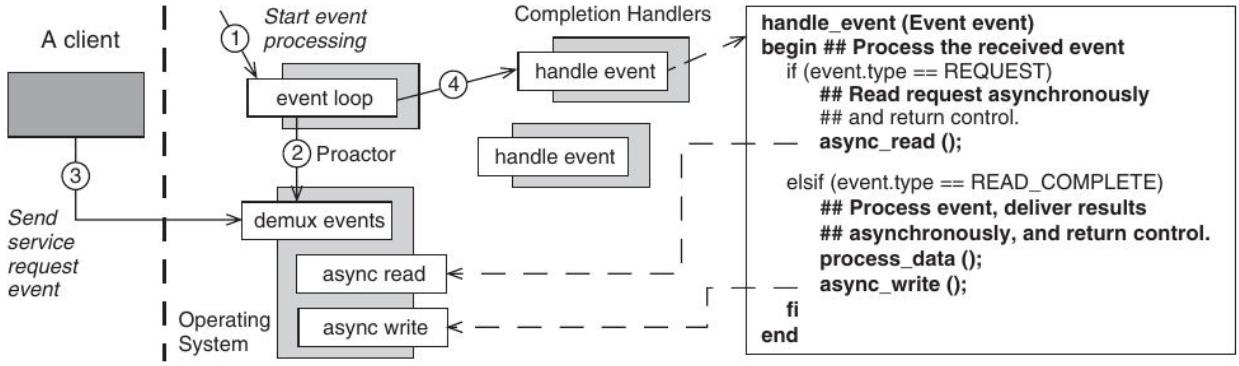
Event Demultiplexing & Dispatching

Reactor



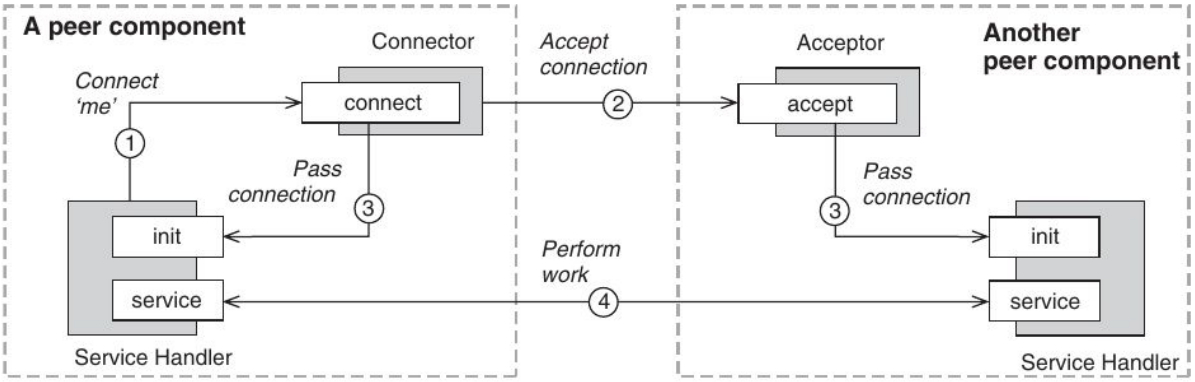
- spôsob, akým je riadené vykonávanie requestov z viacerých zdrojov, ktoré majú nastať simultánne tak, aby request bol v danom čase priradený na jeden zodpovedajúci handler
- krátkotrvajúce requesty

Proactor



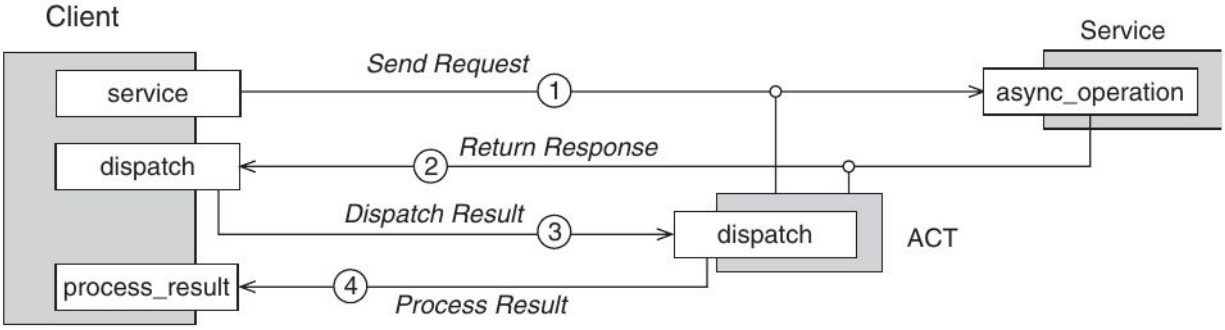
- dlhotrvajúce requesty, často blokujúce jeden druhého, ak by nastávali postupne
- snaha o ich vykonanie simultánne, aby sa nezdržovali navzájom a ich vykonanie asynchrónne
- vyhodnocovanie výsledkov asynchrónnych volaní, ktoré má systém pod kontrolou

Acceptor-Connector



- extrahuje logiku pripájania medzi navzájom komunikujúcimi rovnocennými (peer) komponentami

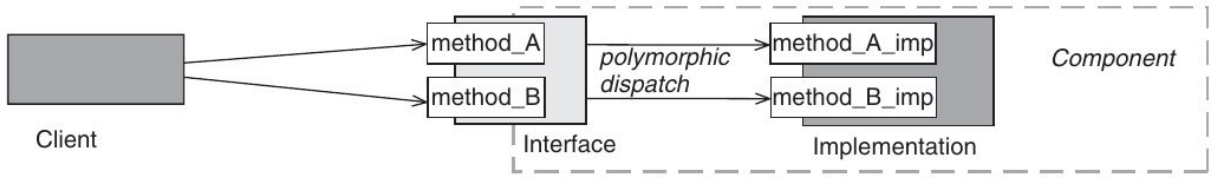
Asynchronous Completion Token



- udalosti sa vykonávajú asynchrónne a poradie prijatia odpovedí nemusí zodpovedať poradiu začatia vykonávania operácií
- token obsahuje minimálne množstvo operácií potrebných na určenie, v akom poradí sa spracujú prijaté informácie

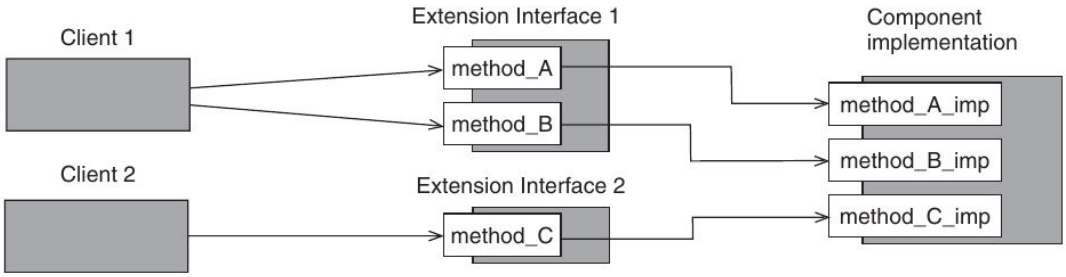
Interface Partitioning

Explicit Interface



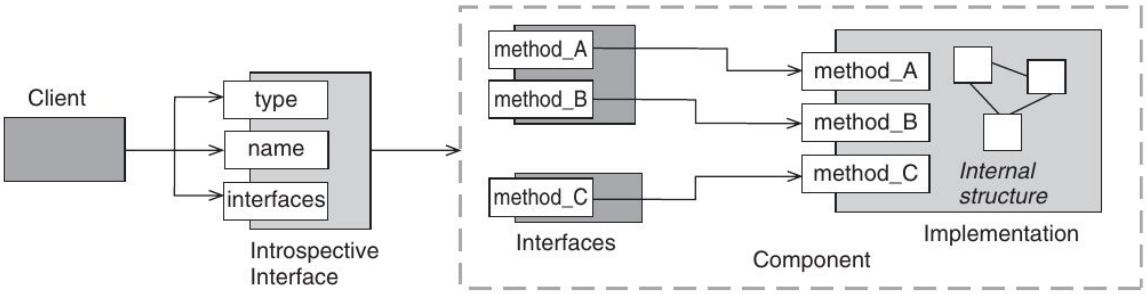
- definovanie rozhrania pre skrytie implementačných detailov komponentu
- rozhranie definuje služby, ktoré komponent poskytuje

Extension Interface



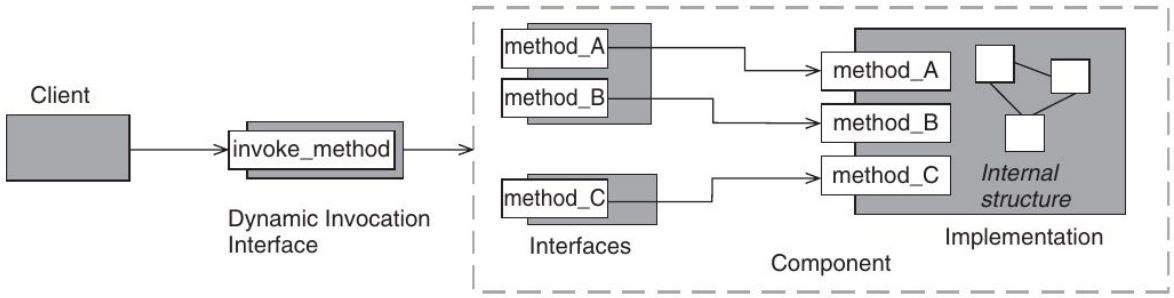
- nové rozhranie, ktoré sa pridáva pri vývine funkcionality komponentu, napri pri pridávaní novej funkcionality
- aj ak sa upravuje signatúra metódy, staré rozhranie sa ponechá s pôvodnou metódou a upravená sa vloží do nového rozhrania

Introspective Interface



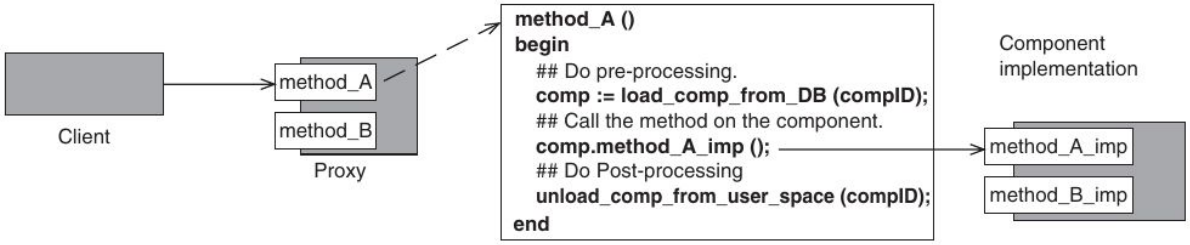
- samostatné rozhranie, ktoré umožňuje prístup k metadátam komponentu (typ, štruktúra, správanie, stav, …)
- slúži na monitorovanie efektívnosti a výkonnosti komponentu
- oddelenie od ostatných častí rozhraní, ktoré poskytujú funkcionalitu

Dynamic Invocation Interface



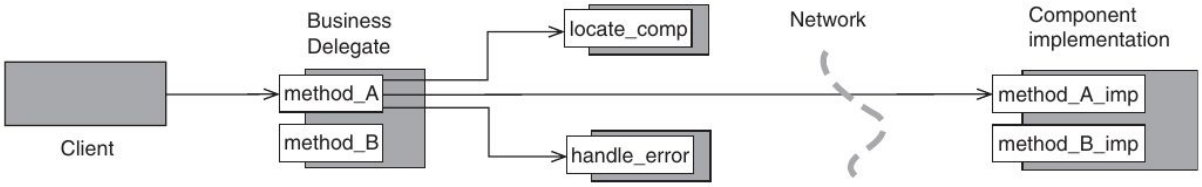
- rozhranie pre vyvolanie metód dynamicky bez priameho definovania závislostí u klienta

Proxy



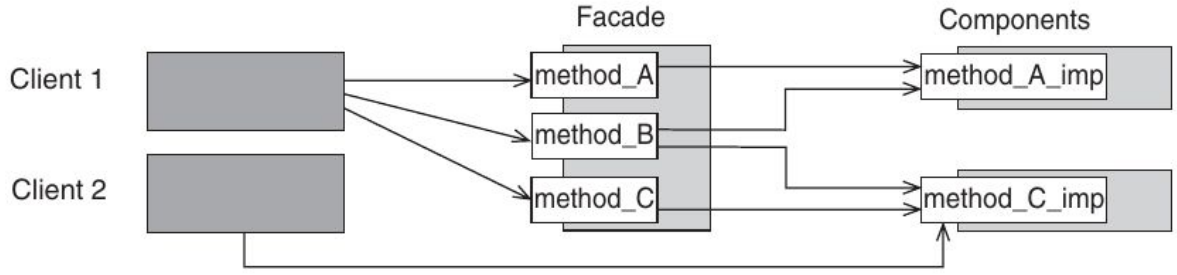
- oddialenie volania náročnej operácie na komponente poskytnutím náhrady s obmedzenou funkcionalitou dovtedy, kým všetky dáta nie sú potrebné
- rovnaké rozhranie ako komponent
- môže obsahovať kontrolu vstupných dát, kontrolu prístup. práv, …

Business Delegate



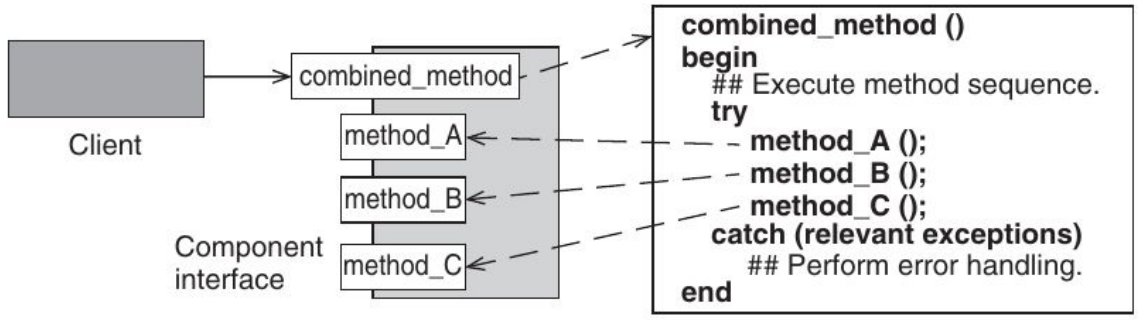
- zapuzdruje cestu s prístupom k vzdialenému komponentu
- poskytuje transparentné lokalizovanie pri volaní v distribuovanom systéme
- klienta nezaujíma, či volaný komponent je vzdialený alebo blízky, stačí výsledok
- spracuje všetky ostatné úlohy, ktoré sú potrebné pre vzdialené volanie (ošetrenie chýb, …)
- rovnaké rozhranie ako vzdialený komponent

Facade



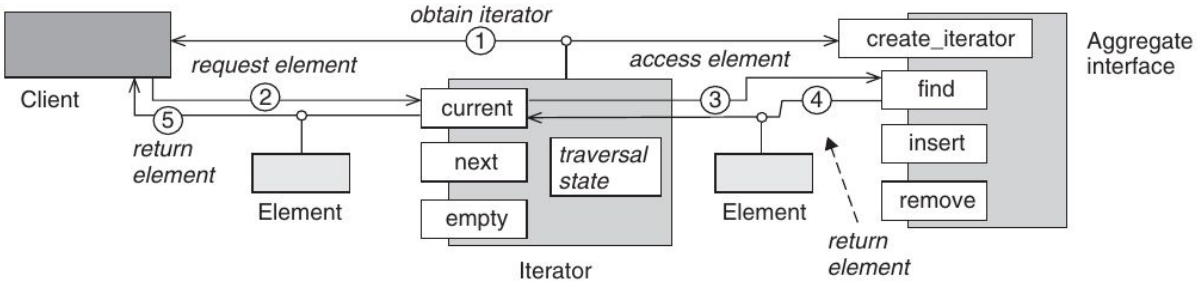
- jednotný prístupový bod pre skupinu komponentov
- usmerňuje volania metód od klienta na jednotlivé komponenty
- klient nemusí používať fasádu, môže si ponechať priamy prístup na komponent

Combined Method



- vykonanie viacerých metód v jednej operácií, ktoré sa vykonávajú často v rovnakom poradí
- minimalizuje sa vkladanie viacnásobného volania operácií v poradí v klientskom kóde
- kombinovaná metóda môže obsahovať jednotný spôsob ošetrovania chýb pre všetky metódy spoločne

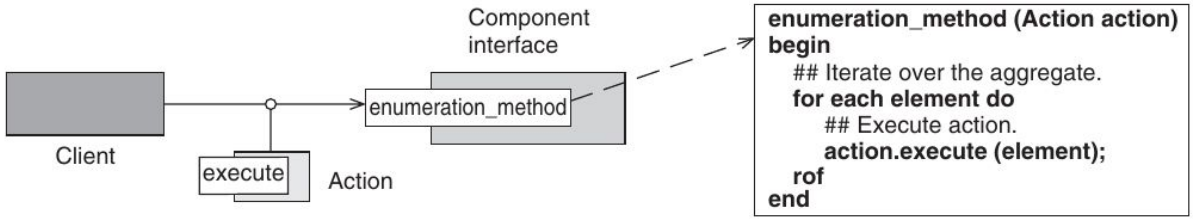
Iterator



- prístup k prvkom štruktúry, ktorý je v určitom poradí bez odhalenia jej vnútornej reprezentácie

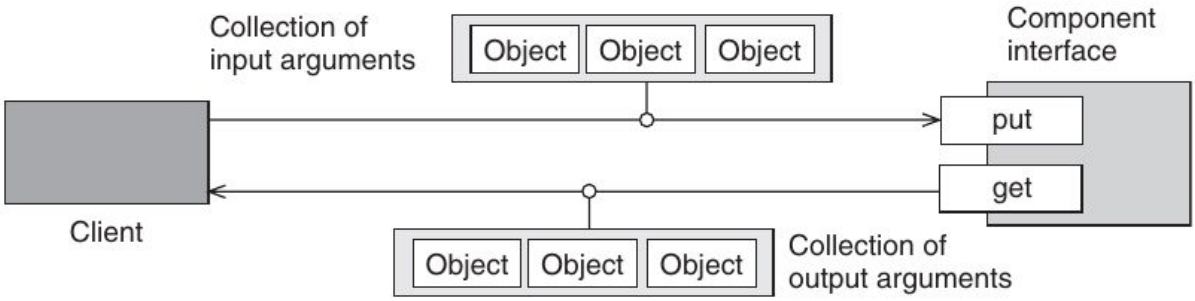
- štruktúra definuje postupnosť usporiadania prvkov

Enumeration Method



- volanie funkcionality každého prvku štruktúry naraz v jednej operácii, aby nebola spôsobená záťaž pri získavaní prvkov po jednom
- vždy sa prejde celá štruktúra

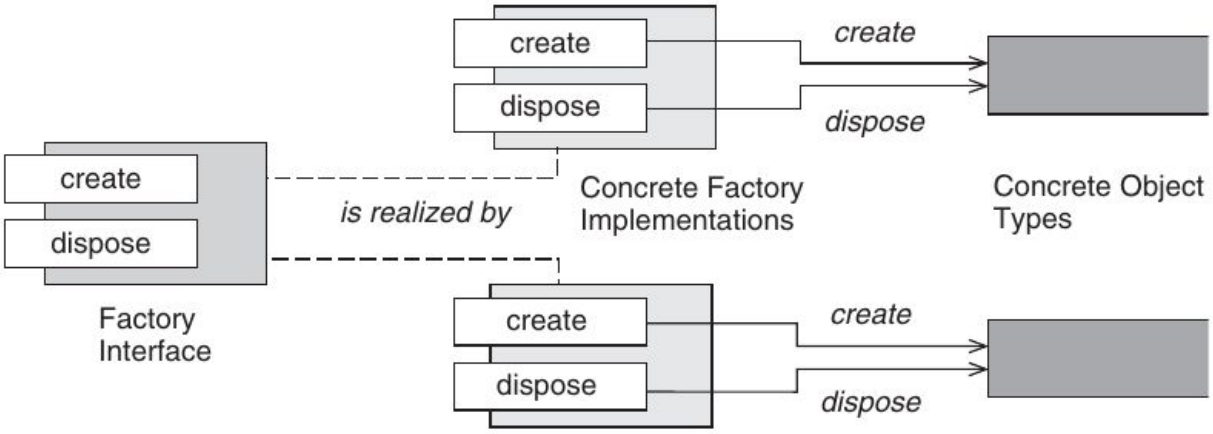
Batch Method



- redukuje viacnásobné zmeny prvkov štruktúry, ktoré by sa získavali a upravovali jednotlivo a spôsobili zbytočnú záťaž

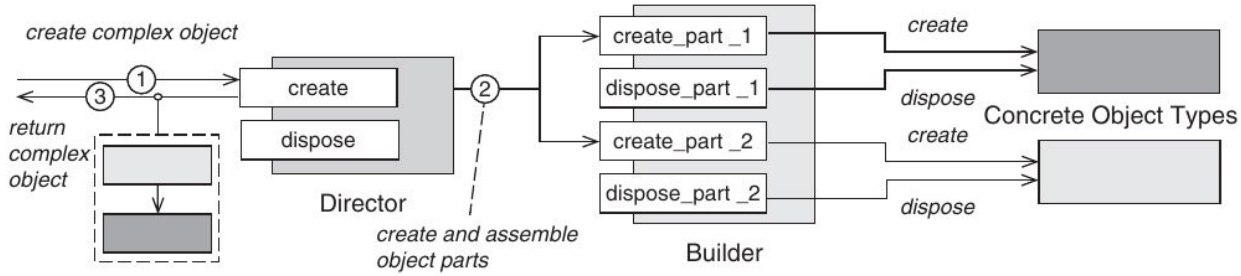
Resource Management

Abstract Factory



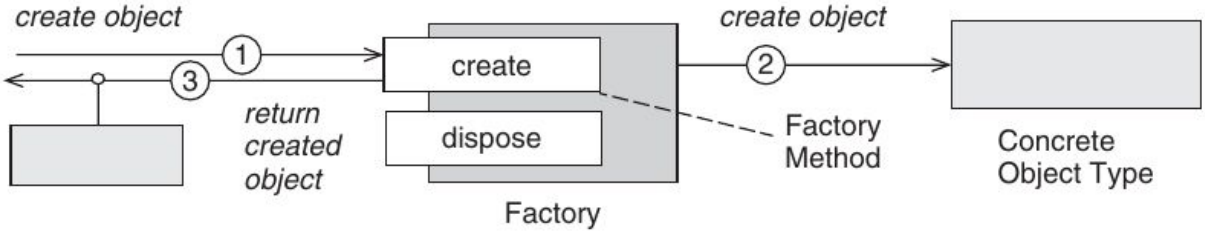
- oddelenie spôsobu vytvárania objektov rovnakého typu
- klient dostáva typ, nie aký objekt je vytvorený
- skrytie, ako sú príbuzné objekty vytvárané
- klientské rozhranie obsahuje konkrétny typ továrne
- skrytá štruktúra objektu

Builder



- vytváranie zložitých objektov
- vytváranie po krokoch
- z každého kroku môže byť vrátený objekt
- zapuzdruje, ako sa vytvára objekt
- zmena implementácie Buildera neovplyvňuje zvyšok

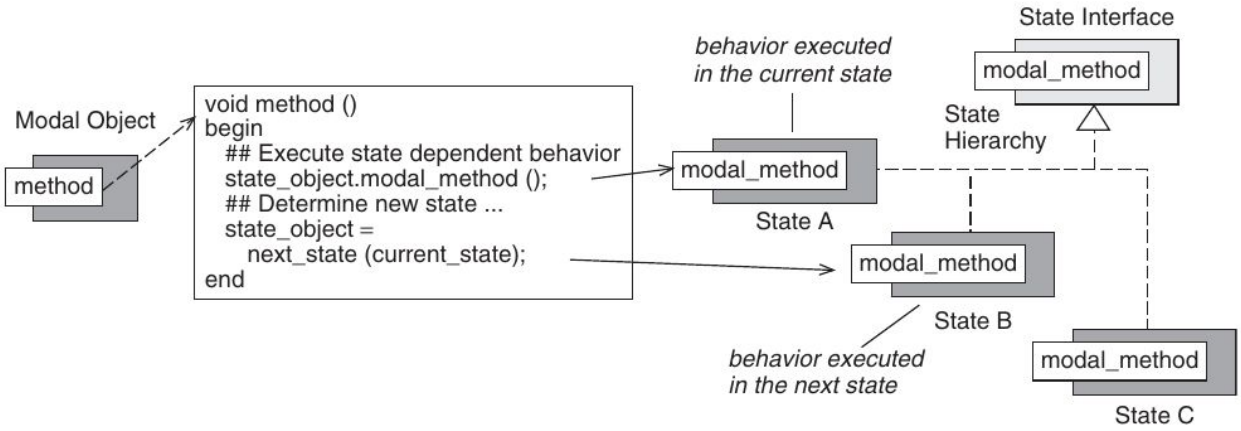
Factory Method



- skrývanie, ako sa vytvárajú objekty
- vytváranie niekedy závisí od typu iného objektu
- metóda môže obsahovať validáciu

Modal Behaviour

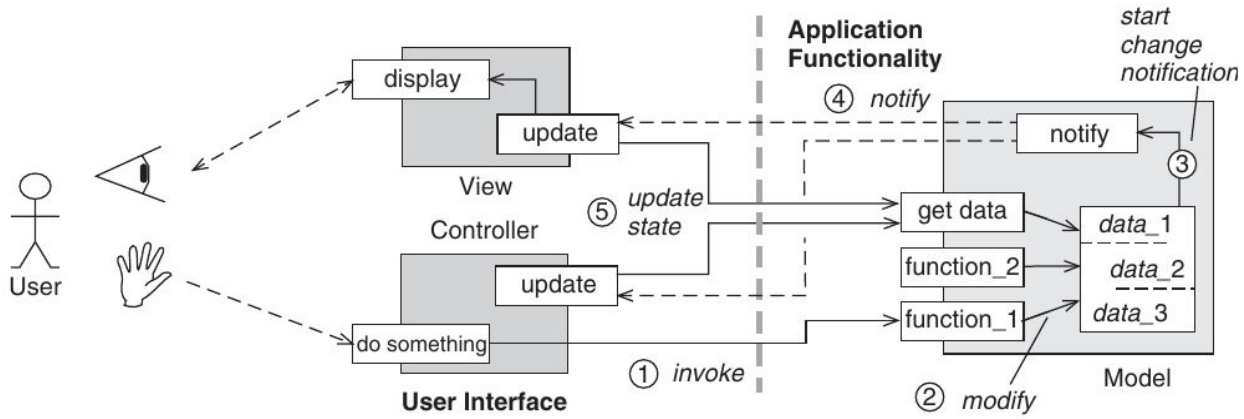
Objects for States



- podpora správania, ktoré sa zmení, keď sa zmení stav objektu
- rozdelí sa na 2 časti
    - objekty pre správanie
    - objekt pre dáta
- objekt na základe dát volá objekty, ktorý má vykonať správanie
- objekty pre správanie sú inštancie hierarchie tried, kde každá trieda predstavuje správanie v určitom stave
- zabraňuje rozvetveným podmienkam a veľkému switch-u pre jeden objekt, ktoré by museli byť v každej metóde správania

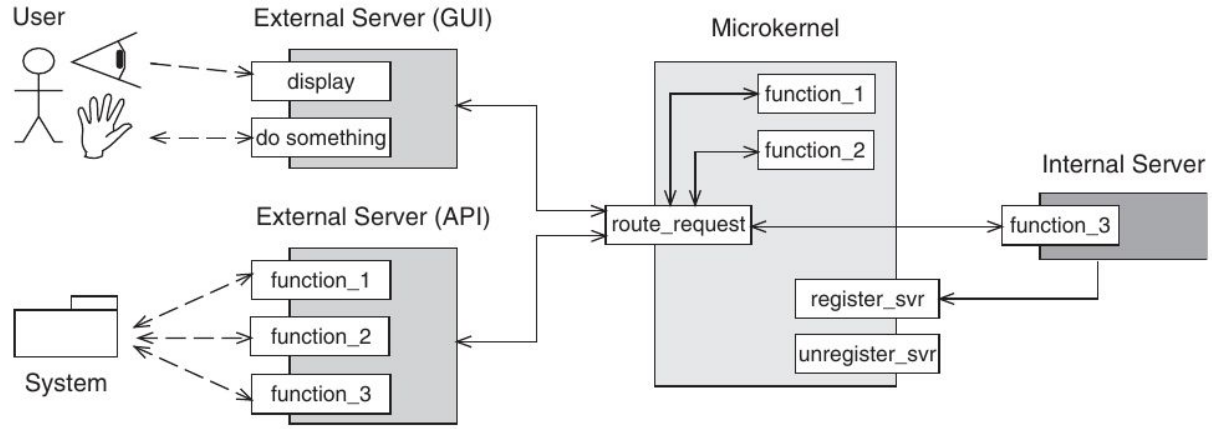
MVC, Microkernel, Reflection

MVC



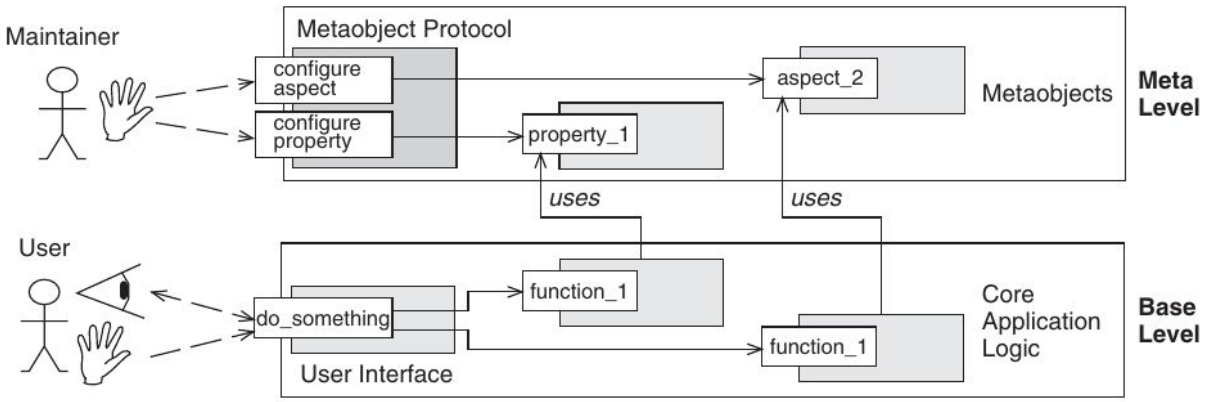
- pri aplikáciach, kde sa často mení UI a funkcionalita ostáva rovnaká
- 3 časti
    - model - základná funkcionalita a spracovanie dát
    - view - zobrazovanie výsledkov používateľovi
    - controller - spracováva vstupy z view a z modelu
- View + Controller => UI
- pri zmene modelu sú notifikované všetky views

Microkernel



- prispôsobovanie sa systému na meniace sa požiadavky
- minimálne funkčné jadro, ktoré obsahuje základnú funkcionalitu
- rožširovanie o nové časti a spolupráca medzi nimi
- podpora rôznych verzií s rovnakým funkčným základom

Reflection



- menenie štruktúry a správania dynamicky
- 2 časti
    - base level - vytvorená aplikačná logika
    - meta level - informácie o systémových vlastnostiach, sledovanie behu aplikácie
- pri dynamickom upravovaní štruktúry metalevelu sa ovplyvňuje base level

J2EE

Stateless Session

- informácie o pripojení sa stratia po vykonaní

- pri častom opakovaní je potrebné vytvárať náročne spojenie

- každá operácia je samostaná
- buď sa urobí commit alebo rollback
- možné vykonanie volania na inom serveri

Stateful Session

- stav o volaniach sa udržiava v pamäti, dáta potrebné na pripojenie na zdroje

- pamäťová náročnosť
- zvýšená réžia
- riešenie kolízií na jednom zdroji pri častom dopytovaní a udržiavaní viacerých spojení beanov

- pri opakovanom spojení netreba spojenie a dáta znovu získavat

Database Access

Database Access Layer

- definuje vzťahy medzi OO a DB
    - jednoduchšia práca s objektami v aplikácii
    - relačné dáta vhodnejšie pre ukladanie
- samostatná vrstva medzi aplikáciou a DB
- prístup k dátam pre aplikáciu
- mapovanie dátových štruktúr

Data Mapper

- skrýva spôsob usporiadania relačných dát
- manipulácia dát medzi aplikáciou a DB
- trasformácie dát a typov
- zmena DB nezmení aplikáciu a naopak

Row Data Gateway

- Rozhranie pre manipuláciu nad jedným záznamom
- dátová štruktúra zodpovedá záznamu v tabuľke
- skýva prístup k DB
- konverzie typov
- vhodné pre malé aplikácie
- jeden objekt pre jeden riadok

Table Data Gateway

- Manipulácie kolekcií dát
- obaľuje DB prístupový kód pre tabuľku
- rozhranie pre kolekcie rovnakého typu
- zapuzdruje implementačné detaily a transformácie typov pre tabuľku a objekty
- prístup po množstvách

Active Record

- Vyhnutie sa komplexnému mapovaniu
- jednoduché manipulačné metódy
- Rozhranie špecifické pre záznam
- zapuzdruje zodpovedajúce dáta
- zmena dát vedie k zmene záznamu
- pre málo zložité aplikácie