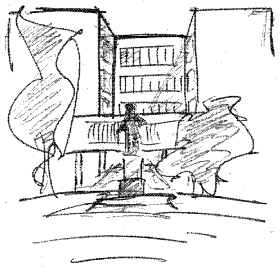
[P271]
Информациони системи

10

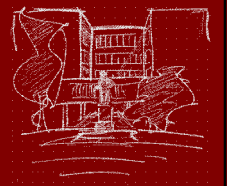


Саша Малков

Универзитет у Београду
Математички факултет

2023/2024

[P271]
Информациони системи



Саша Малков

Тема 14

Архитектура информационих система

Архитектура информационих система

- Разматра се у контексту ширег окружења али и имплементације
- Морамо да имамо у виду
  - **Пословну архитектуру (*Enterprise Architecture*)**
    - Архитектура комплетног ИТ сектора
  - **Архитектуру информационог система**
    - Архитектура информационог система, у оквирима домена
  - **Архитектуру софтвера**
    - Архитектура софтверских елемената информационог система
- Ако је домен ИС целокупно предузеће, онда се оквири ИС и предузећа, као и њихове архитектуре, углавном подударају

Архитектура информационих система (2)

- Архитектуру ИС чине
  - **Информациона архитектура**
    - назива се и Архитектура података
    - дефинише главне податке ИС и односе међу њима
  - **Апликативна архитектура**
    - дефинише апликације потребне за управљање подацима и обављање послова, као и односе међу њима
  - **Технолошка архитектура**
    - одређује технологије и инфраструктуру за имплементацију апликација

Кључни концепти архитектуре ИС

- Пословни процес
- Информациони ентитет
- Апликативна компонента
- Инфраструктурна компонента

Кључни концепти архитектуре ИС

- **Пословни процес**
  - скуп активности које обавља једна класа корисника
- Информациони ентитет
- Апликативна компонента
- Инфраструктурна компонента

Кључни концепти архитектуре ИС

- Пословни процес
- **Информациони ентитет**
  - особа, предмет, место или концепт значајан и релевантан у пословном смислу, о коме/чему се чувају информације
- Апликативна компонента
- Инфраструктурна компонента

Кључни концепти архитектуре ИС

- Пословни процес
- Информациони ентитет
- **Апликативна компонента**
  - целовит скуп операција које се обављају над подацима
  - често се назива и апликативни блок или блок ИС
- Инфраструктурна компонента

Кључни концепти архитектуре ИС

- Пословни процес
- Информациони ентитет
- Апликативна компонента
- **Инфраструктурна компонента**
  - целовит скуп инфраструктурних операција
  - често се назива и технолошки блок или блок ИТ

Кључни концепти архитектуре ИС

- Пословни процес
- Информациони ентитет
- Апликативна компонента
- **Инфраструктурна компонента**
  - целовит скуп инфраструктурних операција
  - често се назива и технолошки блок или блок ИТ
  - разликујемо три подврсте
    - **блок ИТ инфраструктуре**
      - физички структурални концепти, попут чворова, мреже и сл.
    - **блок ИТ платформе**
      - скуп сервиса неопходних за функционисање, повезивање и испоручивање апликативних компоненти
    - **ИТ-блок апликације**
      - представља технолошке елементе имплементације апликативних блокова, попут слојева презентације, података, логике и сл.

Други значајни концепти архитектуре ИС

- **Сервиси**
  - скуп операција које пружа (путем неког интерфејса) апликативни блок
  - разликујемо:
    - пословне сервисе
    - сервисе ИС
    - технолошле сервисе
- **Операције**
  - апстрактни описи активности које подржава један сервис

Пројектовање архитектуре ИС

- Пројектовање архитектуре ИС почива на
  - препознавању и одређивању кључних компоненти
  - одређивању основних технолошких принципа њихове имплементације
  - одређивању начина њиховог повезивања

Пројектовање архитектуре ИС

- Највише пажње се посвећује инфраструктурним и основним апликативним компонентама
  - њихове одговорности и односи дефинишу оквир архитектуре
- Мање пажње се посвећује конкретним пословним процесима и информационим ентитетима
  - они су елементи конкретне имплементације
  - али без њиховог оквирног сагледавања архитектура лако може да буде погрешно постављена

Архитектура софтвера и обрасци повезивања

- **Архитектура**
  - виши ниво апстракције
  - има глобалнији карактер
  - одређује одговорности и односе међу компонентама
    - укључујући и начине повезивања компоненти
- **Образац повезивања**
  - нижи ниво апстракције
  - има локалнији карактер
  - представља начин остваривања односа између компоненти
- Често се ова два појма мешају
- Свака архитектура почива на једном или више образаца повезивања компоненти
- Размотрићемо прво обрасце повезивања, па затим и архитектуре

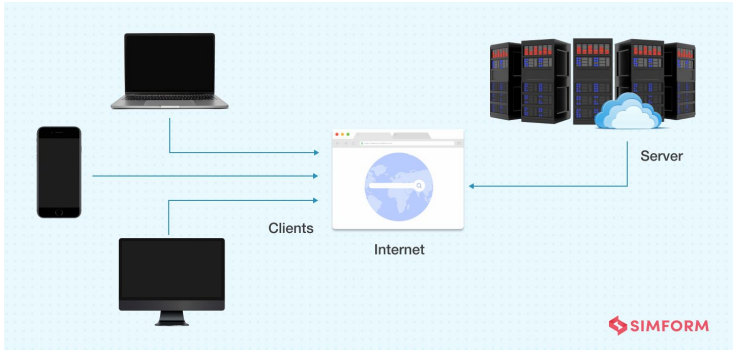
Обрасци повезивања компоненти

- Неки од најчешће коришћених образаца повезивања компоненти су:
  - Клијент-сервер
  - Главни-подређени
  - Цеви са филтерима
  - Брокери
  - Равноправно повезивање

Образац повезивања Клијент-сервер

- Одређује однос двеју компоненти које имају различите улоге
  - **Клијент** – компонента која искључиво користи услуге сервера
  - **Сервер** – компонента која искључиво пружа услуге клијенту
- Релативно једноставан образац повезивања
  - клијент зависи од сервера
    - зависност само у једном смеру
  - релативно ниска спрегнутост путем јасно дефинисаног интерфејса
    - од ширине, сложености и стабилности интерфејса зависи и степен спрегнутости

Образац повезивања Клијент-сервер



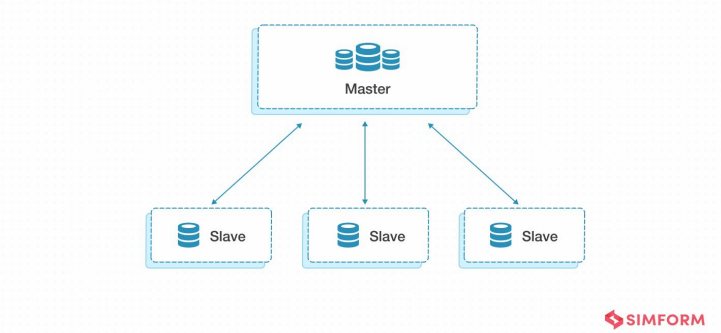
Образац повезивања Клијент-сервер / Особине

- Висока флексибилност у једном смеру
  - Интерфејс сервера је обично предефинисан и стабилан
  - Лако се додају нови клијенти или мењају постојећи, зато што то нема утицаја на сервер
- Једноставно испоручивање
  - Клијенти и сервер могу да се независно испоручују
- Једноставан развој и одржавање клијената
- Потенцијално незгодно одржавање сервера
  - Сложено ако се мења интерфејс
- Перформансе углавном зависе само од сервера
- Умерена скалабилност
  - Своди се на вертикално скалирање сервера
  - Сервер обично има сложено стање и тешко се скалира хоризонтално

Образац повезивања Главни-подређени

- Компоненте у функционалном и оперативном смислу имају веома сличне улоге
- Ипак, разликују се
  - Само једна главна компонента
    - сви корисници (клијенти) комуницирају само са главном компонентом
    - она по потреби дистрибуира захтеве подређеним компонентама
  - Једна или више подређених компоненти
    - углавном не комуницирају са клијентима, већ само међусобно
    - испуњавају задатке које добију од главне компоненте

Образац повезивања Главни-подређени



Образац повезивања Главни-подређени (2)

- Користи се углавном као вид репликације ради сигурности
  - У раду се користи само главна компонента
  - Све промене се са главне пресликавају на подређене компоненте
  - Ако дође до прекида у раду главне компоненте, онда једна од подређених може да се (привремено или трајно) прогласи за нову главну компоненту
- Може да се прилагоди тако да и подређене компоненте могу да се користе за неке операције
  - Обично за операције читања података или прављења извештаја
    - Посебно је згодно за аналитичке операције, које не захтевају пуну стабилност података (тј. могу да читају без закључавања)
  - На тај начин се подижу и перформансе неких операција

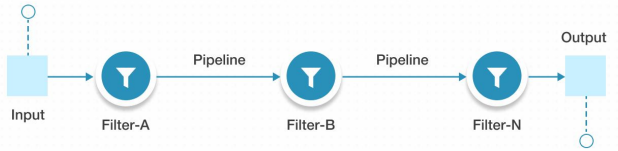
Образац Главни-подређени / Особине

- Умерена флексибилност
  - Обично су компоненте веома јако спрегнуте, дотле да се ради о јединственој имплементацији која се користи у више инстанци и у различитим оперативним режимима
- Умерено једноставно испоручивање
  - Све компоненте се морају испоручити у исто време, али то није превише сложено, зато што се ради о више инстанци исте компоненте
- Углавном једноставан развој и одржавање
- Перформансе зависе од начина употребе
  - Све "главне" операције обавља једна главна компонента, која представља уско грло
  - Само неке операције могу да се пребаце на подређене компоненте
- Умерена скалабилност
  - Главна компонента може да се скалира само вертикално
  - Подређене компоненте могу да се скалирају и хоризонтално, али уз ограничену функционалност

Образац повезивања Цеви са филтерима

- Компоненте се повезују у једносмеран ланац, који дефинише ток података, тзв. цев
  - Излаз сваке од компоненти (осим последње) представља улаз у наредну компоненту
- Овај образац није универзално применљив
  - Не могу сви послови да се поделе на секвенцијалне кораке
  - Али тамо где могу ово је веома згодан образац
- Перформансе су одређене најспоријом кариком у ланцу
  - Али (бар начелно) свака компонента може да се независно скалира

Образац повезивања Цеви са филтерима



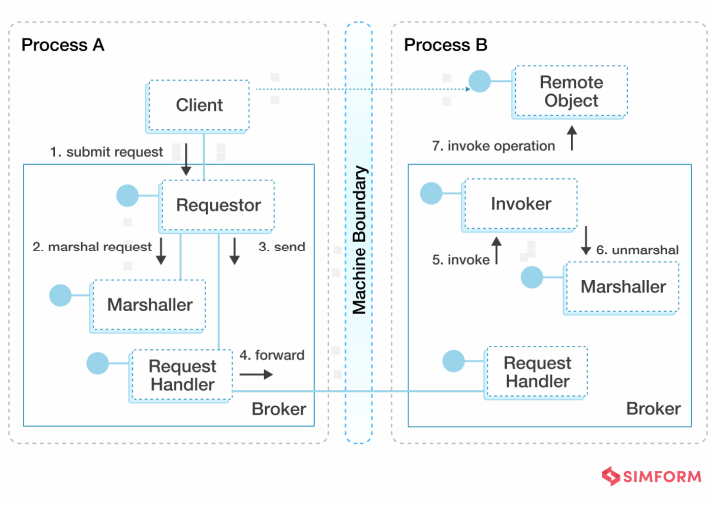
SIMFORM

Образац Цеви са филтерима / Особине

- Релативно ниска флексибилност
  - Умерена спрегнутост
  - Није лако додавати нове или значајно мењати постојеће кораке у постојећем ланцу
  - Додавање значајно другачијих компоненти у ланац често захтева промену апликативног интерфејса или чак и одговорности постојећих компоненти
- Релативно сложено испоручивање
  - Испоручивање сваке од компоненти може да утиче на рад преосталих компоненти у ланцу
  - Ако се мења интерфејс или део интерфејса, онда сви елементи ланца морају да се заједно испоручују
- Умерено једноставан развој и одржавање
  - Ако су улоге компоненти јасно заокружене и изоловане, онда нема проблема
  - Ако нису, онда тестирање и дебаговање могу да буду отежани
- Углавном солидне перформансе
  - Перформансе су одређене најспоријом кариком у ланцу
  - Али (бар начелно) свака компонента може да се независно скалира
- Релативно висока скалабилност
  - Већ сама подела обраде на кораке представља вид хоризонталног скалирања
  - Свака компонента за себе има потенцијал за хоризонтално скалирање
    - мада то није увек једноставно

Образац повезивања Брокер

- Користи се за повезивање компоненти у дистрибуираним системима
  - Али може и када нема дистрибуирања, тј. када се оно само апстрахује
- Брокер је компонента / механизам за повезивање
- Овај образац захтева да делови механизма брокера раде у процесима сваке од компоненти које се повезују



Образац повезивања Брокер (2)

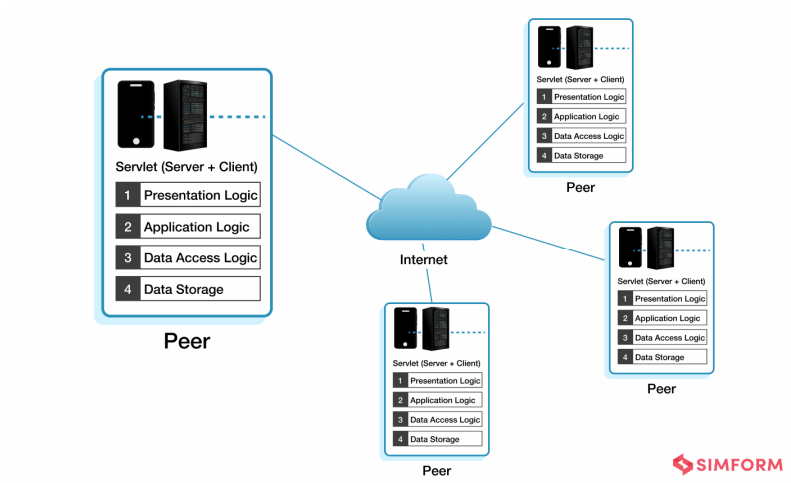
- Сваки пут када клијент захтева неку услугу од неке друге удаљене компоненте
  - брокер (тј. део уграђен у клијента) преводи захтев у одговарајуће формате и протоколе
  - шаље их до удаљене компоненте
  - тамо их поново преводи у облик разумљив тој компоненти и покреће операцију
  - по потреби се одговор враћа на сличан начин у супротном смеру
- При томе брокери могу да комуницирају
  - **непосредно** - брокер непосредно шаље поруку другој компоненти (њеном брокеру)
  - **кроз дељени канал везе** - стављају поруку у ред порука и очекују да ће прималац сам да препозна, преузме и обради одговарајуће поруке
  - **путем посредника** - брокер шаље поруку посреднику, а он проналази примаоца и преноси му поруку

Образац повезивања Брокер / Особине

- Релативно висока флексибилност
  - Једном дефинисан интерфејс брокера се не мења лако, али обично нема ни потребе
  - У систем могу да се додају (и мењају) веома разнолике компоненте
- Релативно сложено испоручивање
  - Испоручивање сваке од компоненти може да мора да буде праћено испоручивањем свих одговарајућих компоненти које она користи
  - Посебно је сложено у случају непосредне комуникације, када свака компонента мора да зна коме шаље поруке
- Углавном једноставан развој и одржавање
  - Ако су улоге компоненти јасно заокружене и изоловане, онда нема проблема
  - Ако нису, онда тестирање и дебаговање могу да буду отежани
- Умерене перформансе
  - Режијски трошкови могу да буду релативно високи
  - Перформансе зависе од техничких аспеката комуникације и грануларности компоненти
  - Ако повезивање није непосредно, уско грло могу да представљају канал везе или посредник
- Релативно висока скалабилност
  - Свака од компоненти има потенцијал за хоризонтално скалирање
  - Осим ако се остварује непосредно повезивање

Образац Равноправно повезивање

- Користи се за уопштене видове повезивања начелно равноправних компоненти
  - Свака компонента може да има улогу сервера или клијената у односу на одговарајуће услуге
  - Посебно се користи при повезивању више компоненти са истим или веома блиским одговорностима
    - дистрибуиране базе података
    - дистрибуирани системи датотека
    - дистрибуиране комуникационе мреже (VOIP, Skype и слично)
- У односу на друге обрасце повезивања
  - децентрализација у односу на Клијент-сервер
  - уопштавање комуникације у односу на Брокер
  - уопштавање улога у односу на Главни-подређени



Образац Равноправно повезивање / Особине

- Релативно висока флексибилност
  - Обично се ради о више инстанци исте компоненте, па је слобода у њеном развоју врло велика
- Потенцијално сложено испоручивање
  - Све док се не мења интерфејс порука, начелно је могућа комуникација између инстанци различитих верзија компоненте
  - Размена порука (и података и операција) између различитих верзија компоненте може да буде проблематична
- Умерено једноставан развој и одржавање
  - Зависи од предмета и начина дистрибуирања
- Перформансе зависе од случаја до случаја
  - У питању је вид дистрибуиране обраде, па има простора за високе перформансе, али и за проблеме
- Релативно висока скалабилност
  - Број инстанци начелно није ограничен
  - Али технологија њиховог повезивање може да има ограничења

Основни обрасци архитектуре ИС

- Слојевита архитектура
- Архитектура вођена догађајима
- Архитектура микро-језгра
- Сервисно-оријентисана архитектура (COA)
- Архитектура микросервиса
- Архитектура заснована на простору

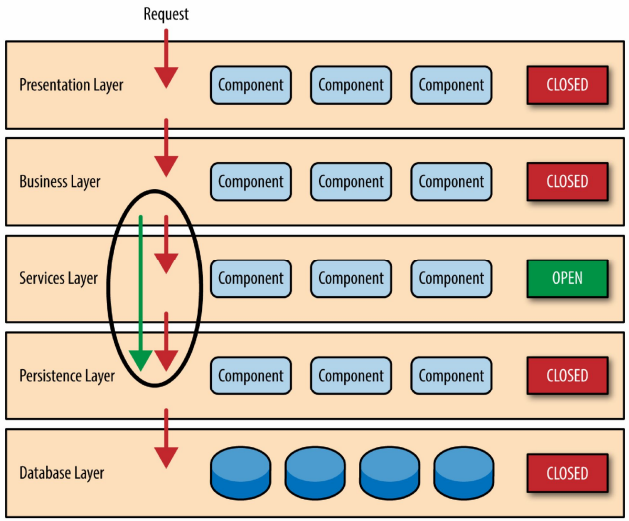
Слојевита архитектура

- Подразумева слојевито раздвајање - од физичких записа података до корисника
- Обично има три "главна" слоја:
  - слој презентације
  - слој пословне логике
  - слој података
- Може да има и више слојева, ако се неки слојеви додатно раздвоје
- Може да има и мање слојева, ако се неки слојеви додатно интегришу

Слојевита архитектура (2)

- Већи број слојева обично долази од
  - раздвајања слоја података на
    - слој перзистенције
      - пружа интерфејс према подацима за више слојеве
      - обично OP-пресликавање, РЕСТ интерфејс и сл.
    - слој физичке репрезентације
      - представља имплементацију записивања података
      - обично база података, фајл систем и сл.
  - додавања слоја сервиса између слојева пословне логике и перзистенције
    - представља корак према СОА
    - тај додатни слој, или неки његови делови, често се деле између више апликација/система/подсистема
    - може и додатно да се дели, према локацији

Слојевита архитектура (3)



Слојевита архитектура (4)

- Мањи број слојева може да постоји из више разлога
  - интеграција презентационог слоја и слоја пословне логике
    - има смисла ако апликација/систем/подсистем нема физичке крајње кориснике, већ се користи само од стране других апликација (на пример, B2B)
  - интеграција пословне логике и слоја података
    - може да има смисла ако апликација/систем/подсистем представља проширење неког другог система, који чува податке и испоручује непосредно слоју пословне логике у облику који је ту употребљив
- Мањи број слојева практично значи да није у питању потпуни систем већ само део или допуна неког система
  - или нема крајње кориснике и презентациони слој
  - или нема своје податке и слој података

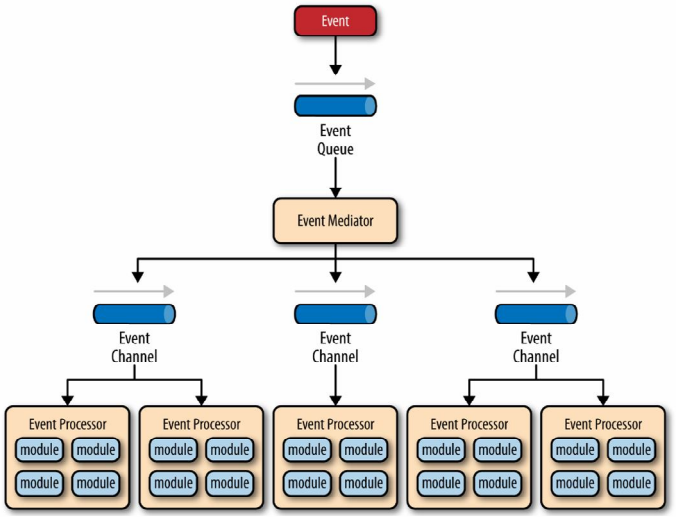
Слојевита архитектура / Особине

- Релативно ниска флексибилност
  - Неке врсте промена могу да буду ограничене на појединачне слојеве, али углавном се промене простиру кроз више слојева
  - Измене се лако планирају, али је њихово имплементирање дистрибуирано
- Сложено испоручивање
  - Слојеви су често монолитни или састављени од више комплексних целина
  - Често више слојева мора да се испоручује заједно
- Добра модуларност
  - Обично је чиста подела одговорности
  - Развој је углавном лакши него код других архитектура
  - Релативно лако се тестира и дебагује
- Перформансе дискутабилне
  - Начелно добре
  - Осетљиве на уска грла сваког од слојева
  - Сваки захтев пролази кроз све слојеве у два смера, што некада може да буде ограничење, ако су потребне врло високе перформансе обраде неких захтева
- Ограничена скалабилност
  - Сваки од слојева је јединствен и не може да се једноставно реплицира
  - Уска грла морају да се решавају вертикалним скалирањем, што је скупо и ограничено

Архитектура вођена догађајима

- Идеја је да се раздвоје јединице за обраду појединачних догађаја од места настајања тих догађаја
- Између места настајања и места обраде мора да постоји компонента која управља распоређивањем догађаја
- Постоје два основна обрасца ове архитектуре
  - **заснована на посреднику**
  - **заснована на брокеру**
- У оба случаја постоје две основне врсте догађаја
  - **иницијални** - они који потичу од оригиналног извора
  - **процесни** - они које посредник производи током обраде других догађаја

Архитектура вођена догађајима заснована на посреднику



АВД заснована на посреднику / Елементи

- Ред догађаја
- Посредник
- Канали догађаја
- Јединица за обраду догађаја

АВД заснована на посреднику / Елементи

- **Ред догађаја**
  - прихвата све догађаје и испоручује их посреднику
  - може да буде и обично и има више редова догађаја, за сваку врсту догађаја или сваки подсистем посебно
- Посредник
- Канали догађаја
- Јединица за обраду догађаја

АВД заснована на посреднику / Елементи

- Ред догађаја
- **Посредник**
  - задужен за управљање обрадом догађаја
  - за сваки преузет догађај, емитује један или више процесних догађаја којима се тај догађај имплементира
    - иако може да личи, то није пословна логика, него механизам препознавања надлежних јединица
- Канали догађаја
- Јединица за обраду догађаја

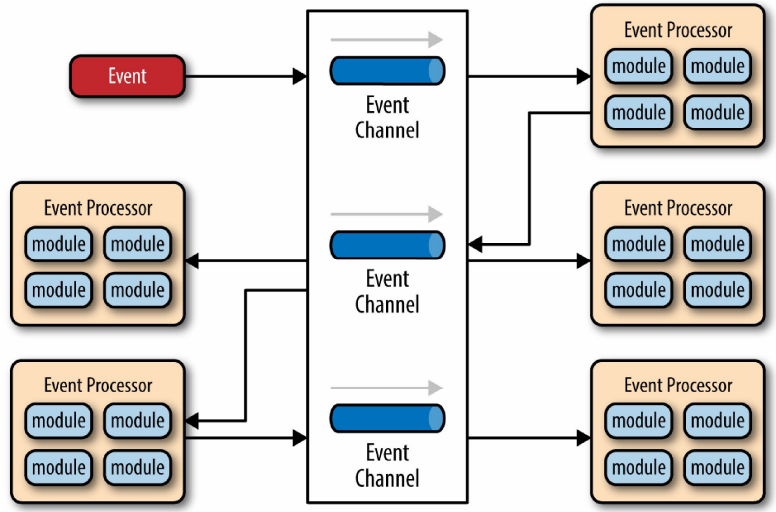
АВД заснована на посреднику / Елементи

- Ред догађаја
- Посредник
- **Канали догађаја**
  - прихватају процесне догађаје од посредника
  - испоручују их јединицама за обраду
  - један канал може да опслужује једну или више јединица за обраду
- Јединица за обраду догађаја

АВД заснована на посреднику / Елементи

- Ред догађаја
- Посредник
- Канали догађаја
- **Јединица за обраду догађаја**
  - обухвата пословну логику догађаја
  - мора да буде целовита и раздвојена од других јединица, колико је то могуће
  - грануларност може да буде различита, тј. једна јединица може да буде задужена за само једну врсту догађаја или и за више сродних врста догађаја

Архитектура вођена догађајима заснована на брокеру



АВД заснована на брокеру / Елементи

- Не постоји посредник
- Јединствени канал догађаја
  - сви догађаји се додају у јединствени канал догађаја
  - сваки догађај у каналу се "ставља на располагање" свим јединицама
- Јединице за обраду догађаја
  - јединице за обраду виде све догађаје
  - преузимају само оне за које су надлежне
  - обрађују их и по потреби производе и додају у канал нове процесне догађаје

Арх. вођена догађајима / Примена

- Ова архитектура је сложенија од слојевите
- Може да имплементира целу апликацију/систем/подсистем, или само неки од слојева
- Ако имплементира целу апликацију, онда међу изворним догађајима имамо пре свега догађаје корисничког интерфејса, а међу процесним догађајима имамо догађаје који одговарају обради на нивоу пословне логике али и оне који одговарају обради на нивоу слоја података
  - на пример:
    - изворни догађај може да буде попуњавање формулара за уписивање предмета
    - одговарајућа јединица за обраду анализира податке, проверава ауторизацију и сл. и прави процесни догађај одговарајуће пословне логике
    - одговарајућа јединица за обраду уписа ће направити одговарајуће догађаје појединачних измена података и старати се о томе да се они одвијају као целовита трансакција
    - одговарајуће јединице за обраду догађаја на подацима ће непосредно мењати садржај базе података
    - ...

Арх. вођена догађајима / Примена (2)

- Ако се овом архитектуром имплементира само појединачан слој слојевите архитектуре, онда су врсте догађаја више уједначене, па ова архитектура има ограничен опсег примене, али је управљање и организовање далеко једноставније
- COA може да се моделира на начин који подсећа на Арх. вођену догађајима
  - Код COA имамо слично успостављање комуникације, иако у средишту пажње обично нису *догађаји* него *захтеви*

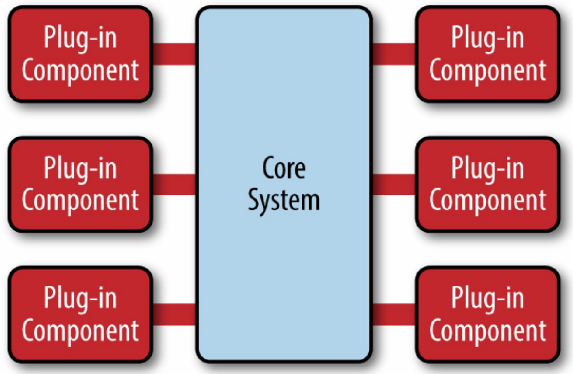
Архитектура вођена догађајима / Особине

- Висока флексибилност
  - Појединачне јединице за обраду имају релативно уске и изоловане одговорности, па се релативно лако мењају и додају
- Једноставно испоручивање
  - Свака јединица за обраду представља јединицу испоручивања
  - У случају измена је често довољно испоручити појединачну јединицу
- Отежан развој
  - Обично је веома чиста и јасна подела одговорности али образац почива на асинхроним механизмима
  - Развој је због тога углавном нешто сложенији
  - Релативно су сложени тестирање и дебаговање
- Перформансе су углавном високе
  - Перформансе зависе пре свега од квалитета централизованог механизма управљања порукама
    - али он обично није превише захтеван (осим ако имамо огроман број малих догађаја)
  - Остали делови система су прилично слабо спрегнути и могу да раде асинхроно и паралелно
- Висока скалабилност
  - Појединачне јединице се лако реплицирају
  - Осетљив је само део за управљање порукама

Архитектура микро-језгра

- Уобичајена за апликације које се израђују и испоручују као "монолитне" целине које могу да се допуњавају по потреби
- Флексибилност се остварује путем уметака (енгл. plug-in)
- Две основне компоненте су
  - **језгро система**
    - минималне функционалности (зато се зове "микро"-језгро)
    - омогућава повезивање и покретање уметака
    - имплементира логику употребе и повезивања али не и логику појединачних послова
  - **умеци**
    - представљају самосталне независне компоненте
      - могу да користе једни друге али и да буду потпуно независни
    - имплементирају специфичне послове

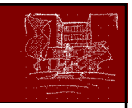
Архитектура микро-језгра



Архитектура микро-језгра / Примена

- Језгро система обично има регистар расположивих уметака
  - СОА са регистром сервиса се моделира на сличан начин, мада су код СОА много интензивније међусобне зависности појединачних компоненти
- Често се користи за део ИС, а не за цео ИС
  - на пример, подсистем за обраду догађаја може да се имплементира применом архитектуре микро-језгра

Архитектура микро-језгра / Особине



- Висока флексибилност
  - Појединачни умеци се лако додају и мењају, без много утицаја на друге компоненте
- Углавном једноставно испоручивање
  - Као што се одвојено развијају, тако умеци могу и да се одвојено испоручују
- Умерено једноставан развој
  - Низак ниво спрегнутости и висок ниво изолованости појединачних уметака начелно омогућава релативно лако развијање, тестирање и дебаговање
  - Ипак, потенцијално висок степен уопштења врста уметака може да има велики утицај на порast сложености језгра и архитектуре повезивања, а тиме и на сложеност развоја и језгра и уметака
- Перформансе су углавном високе
  - Језгро система обично нема велике процесне и протоколарне захтеве па систем има потенцијал за високо искоришћење ресурса
- Умерена или ниска скалабилност
  - Језгро обично није скалабилно или је умерено скалабилно, зато што се ради о издвојеној и релативно целовитој компоненти
  - У зависности од технике повезивања уметака, начелно би могла да се оствари виша скалабилност, али је у пракси углавном ограничена

Сервисно-оријентисана архитектура

- ...обрађено раније...

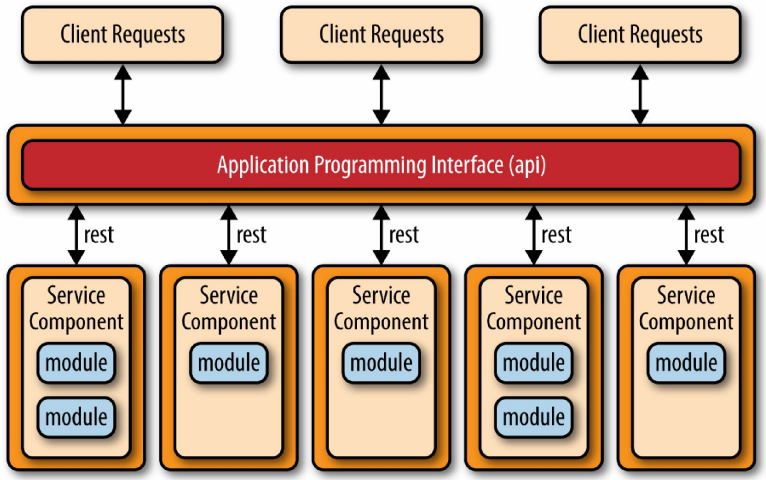
Архитектура микро-сервиса

- Базирана је на концептима
  - одвојеног испоручивања функционалних јединица
  - основу развоја и моделирања чине сервисне компоненте
    - функционалне јединице и сервисне компоненте нису исто
      - ФЈ су јединице развоја и испоручивања
      - СК су јединице које пружају заокружену услугу
      - једна СК може да обухвата више ФЈ
  - у основи представља дистрибуирану архитектуру
    - компоненте су ниско спрегнуте и комуницирају неким мрежним протоколима
    - грануларност може бити веома различита
    - поједностављена координација и повезивање, у односу на СОА

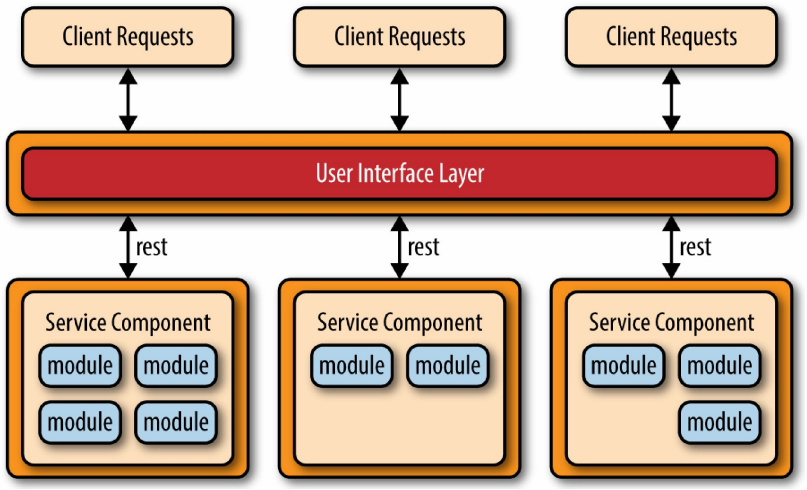
Архитектура микро-сервиса / Топологија

- Архитектура је веома флексибилна и пружа слободу при избору начина повезивања
- Уобичајене су топологије
  - **РЕСТ-АПИ**
    - самостални заокружени сервиси који се повезују питем РЕСТ-АПИ-ја
    - обично са фином гранулацијом сервиса
    - уобичајено за веб-апликације
  - **РЕСТ-апликација**
    - сервиси имају нешто виши ниво и користе се непосредно од стране ГКИ
    - нешто грубља гранулација - мањи број већих сервиса који раде веће појединачне послове
    - уобичајено за ИС средње сложености
  - **централизовано управљање порукама**
    - слично као РЕСТ-апликације, али се између ГКИ и сервиса додаје нови слој за обраду порука
      - налик на обраду догађаја са брокером
      - слој порука нема улогу контролисања или управљања већ само представља слој комуникације
    - уобичајено за велике или сложене ИС
    - лакше скалирање путем реплицирања компоненти

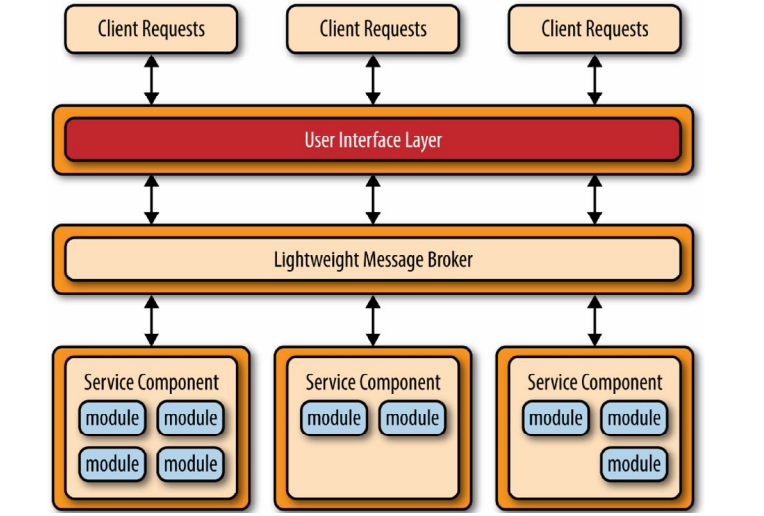
Архитектура микро-сервиса / Топологија РЕСТ-АПИ



Архитектура микро-сервиса / Топологија РЕСТ-апликација



Архитектура микро-сервиса / Топологија централизованог управљања порукама



Арх. микро-сервиса / Управљачке компоненте

- Један од основних мотива за ову архитектуру је избегавање сложених зависности и процедура управљања или контролисања рада
  - Међутим, ако има јако много сервиса (на пример РЕСТ-АПИ топологија примењена у сложеним случајевима), онда може да постане неопходно да се граде неки управљачки механизми
  - Зато су у неким сложеним случајевима друге топологије препоручљивије

Архитектура микро-сервиса / Особине

- Висока флексибилност
  - Сервиси су заокружене целине које су слабо спрегнуте, па се лако и независно додају и одржавају
- Једноставно испоручивање
  - Јединица испоручивања је СК или чак ФЈ
  - Одређена сложеност може да буде последица потенцијално великог броја елемената који се испоручују
- Једноставан развој
  - Све фазе развоја, од прављења, преко тестирања па до дебаговања су релативно једноставне, у односу на друге архитектуре
  - Изолованост и заокруженост ФЈ и СК имају пресудну улогу при томе
- Перформансе су умерене
  - Због поделе система на већи број компоненти, које комуницирају мрежним протоколима, режијски трошкови могу да постану прилично значајни
  - У пракси то обично није велики проблем због добре скалаблности
- Висока скалабилност
  - СК се релативно лако реплицирају, посебно у случају топологије са централизованим протоком порука

Архитектура микросервиса и СОА

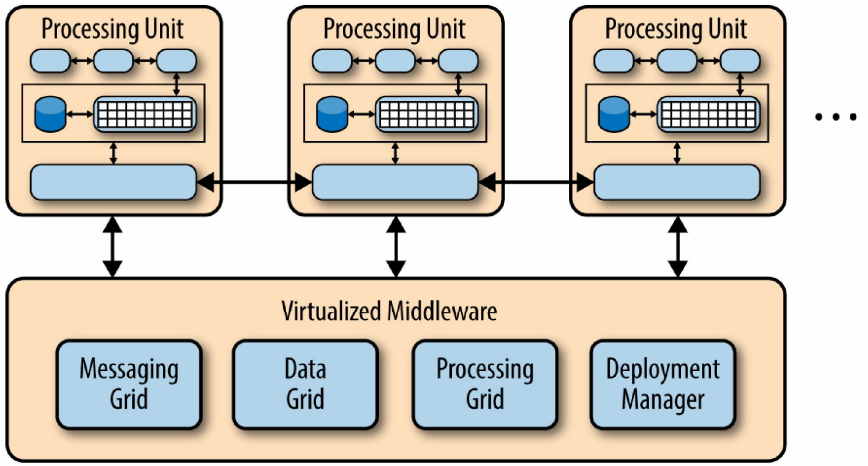
- Основни концепти су веома слични, али има битних разлика:
  - дизајн целина / сервиса
    - микросервиси су често мањи, ситније гранулисани него сервиси код СОА
    - у СОА, сервис је и јединица функционалности и јединица дистрибуирања и јединица употребе
    - код МС, јединица употребе често није сервис него група сервиса, због ситније гранулације
  - стандардизација и формализација
    - СОА захтева уједначену технологију интерфејса у целом систему и често додатну контролу (регистре и сл.)
    - МС често користе различите технологије и протоколе
  - природа комуникације
    - МС често захтевају комуникацију без стања
    - СОА нема ограничења у том смислу

Архитектура заснована на простору

- Назива се и архитектура у облаку, али се примењује и без облака
- Основни циљ је да омогући добро скалирање на свим нивоима, од корисничког интерфејса, преко пословне логике, па све до базе података
- Основу чине две велике целине
  - процесне јединице
  - виртуализована апликативна инфраструктура (*middleware*)
- Процесне јединице, у одређеном смислу, одговарају концепту сервиса или микросервиса
  - обухватају и јединице корисничког интерфејса (па и веб компоненте), али и компоненте позадинске пословне логике
  - чине их апликативни модули, подаци који се чувају у меморији и подсистеми за перзистенцију

Архитектура заснована на простору (2)

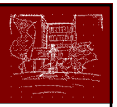
- Виртуализована апликативна инфраструктура је практично контролер система
- Она садржи четири основне целине:
  - подсистем за размену порука (*messaging grid*)
  - подсистем за синхронизацију и репликацију података у меморији (*data grid*)
  - подсистем за обраду (*processing grid*)
  - управљач испоручивањем (*deployment manager*)



Архитектура заснована на простору (2)

- Виртуализована апликативна инфраструктура је практично контролер система
- Она садржи четири основне целине:
  - **подсистем за размену порука (*messaging grid*)**
    - служи да пристигле поруке (захтеве) проследи одговарајућој слободној (или бар најмање оптерећеној) процесној јединици
    - сложеност зависи од величине система и алгоритма распоређивања
  - подсистем за синхронизацију и репликацију података у меморији (*data grid*)
  - подсистем за обраду (*processing grid*)
  - управљач испоручивањем (*deployment manager*)

Архитектура заснована на простору (2)



- Виртуализована апликативна инфраструктура је практично контролер система
- Она садржи четири основне целине:
  - подсистем за размену порука (*messaging grid*)
  - **подсистем за синхронизацију и репликацију података у меморији (*data grid*)**
    - има централну улогу, зато што у овој архитектури већина активности ради над подацима у меморији
    - стара се о синхронизацији података између процесних јединица
      - све морају да имају исту верзију података
      - мада у пракси то не мора да се усклађује синхроно, већ може и асинхроно, ако је довољно брзо
  - подсистем за обраду (*processing grid*)
  - управљач испоручивањем (*deployment manager*)

Архитектура заснована на простору (2)

- Виртуализована апликативна инфраструктура је практично контролер система
- Она садржи четири основне целине:
  - подсистем за размену порука (*messaging grid*)
  - подсистем за синхронизацију и репликацију података у меморији (*data grid*)
  - **подсистем за обраду (*processing grid*)**
    - опциона компонента која управља разменом порука између процесних компоненти у случају дистрибуирања послова
  - управљач испоручивањем (*deployment manager*)

Архитектура заснована на простору (2)

- Виртуализована апликативна инфраструктура је практично контролер система
- Она садржи четири основне целине:
  - подсистем за размену порука (*messaging grid*)
  - подсистем за синхронизацију и репликацију података у меморији (*data grid*)
  - подсистем за обраду (*processing grid*)
  - **управљач испоручивањем (*deployment manager*)**
    - покреће и зауставља процесне јединице и стара се да оне користе усклађене верзије софтверских компоненти

Архитектура заснована на простору / Особине

- Ово је сложена архитектура и скупа за имплементацију
  - Није посебно добро прилагођена класичним апликацијама са великим релационим базама података
  - Обично се користи са нерелационим системима
- Веома је флексибилна
  - Процесне јединице су међусобно независне и могу да се додају и мењају без много ограничења
- Једноставно испоручивање
  - када се једном дефинише инфраструктура, свака јединица се лако испоручује
- Сложен развој
  - и инфраструктура и повезивање процесних јединица могу да буду веома сложени
  - тестирање и дебаговање могу да буду проблематични због сложености окружења
- Веома висока скалабилност
  - процесне јединице су независне и лако се реплицирају

Литература за тему

- Richards, ***Software Architecture Patterns***, *O'Reilly, 2015*
- Vasconcelos, Sousa, Tribolet, ***Information System Architectures: Representation, Planning and Evaluation***, *J. of Systemics, Cybernetics and Informatics, Vol.1 N.6, 2004.*
- *https://www.simform.com/blog/software-architecture-patterns/*