МАТФ
Универзитет у Београду
Математички факултет

11001 БЕОГРАД
Студентски трг 16
П.П. 550
Телефон: 011 20 27 801
Факс: 011 26 30 151
matf@matf.bg.ac.rs
www.matf.bg.ac.rs

Информациони системи

Софтверска архитектура

Дара Милојковић
dara_milojkovic@matf.bg.ac.rs

2. децембар 2022.

Садржај

Садржај

Шта је софтверска архитектура?

- Архитектура софтверског система је скуп структура потребних да се о том систему резонује. У овај скуп спадају елементи система, њихови међусобни односи као и особине ових елемената и односа.
- Архитектура софтвера је основна организација система која обухвата његове компоненете, њихове међусобне односе и односе са окружењем у ком се систем налази, као и све принципе који су утицали на дизајн и еволуцију система.
- Укључује низ одлука и свака од њих може имати значајан утицај на квалитет, перформансе, одрживост и општи успех апликације.
- Ово је највиши ниво разлагања система на делове.

Шта је софтверска архитектура? (2)

- Архитектура представља мост између апстрактних пословних циљева и конкретног резултујућег система.
- Одређује да ли ће систем моћи да покаже жељене особине (високе перформансе, безбедност, скалабилност,...).
- Евалуацијом архитектуре могуће је предвидети које особине може имати систем.

Садржај

Дизајнирање архитектуре софтвера

- Софтверска архитектура директно одређује баланс између особина система.
- Често не постоји најбоља архитектура већ само архитектура са баланосом који одговара потребама.
- Основни захтеви који су постављени пред архитектуру:
  - Функционални захтеви - шта систем треба да ради
  - Захтеви о квалитету - брзина одзива, лакоћа коришћења, безбедност система, пропустност система...

Дизајнирање архитектуре софтвера (2)

- Посао архитекте софтвера је да разуме квалитете и мане који сваки појединачан избор архитектуре доноси и да у складу са тим изабере најприкладнију архитектуру.
- Избор архитектуре се тешко мења.
  - Најчешћи исход је поновно писање целог система.

Основни принципи дизајна архитектуре

- **Раздвајање одговорности** - сваки део система би требало да има јединствену одговорност.
- **Јединствена одговорност** - сваки део система би требало да обавља само једну функцију.
- **Принцип најмањег знања** - ниједна копонента система не би требало да зна имплементационе детаље других компоненти.
- **Лењо пројектовање система** - треба дизајнирати само компоненте система које су потребне.

Додатна разматрања при дизајну архитектуре

- Имплементације компоненти треба да буду конзистенте.
- Треба преферирати композицију објеката уместо наслеђивања.
- Не мешати логичке одговорности у оквиру компоненти.
- Издвојити пропратне компоненте што даље од пословне логике.
- Јасно дефинисати употребу и комуникацију компоненти.

Избори које треба направити при дизајнирању архитектуре

- Тип апликације (апликација за телефон, Веб апликација, апликација за персонални рачунар, ...)
- Технологије имплементације (одабир је често вођен ограничењима инфраструктуре, политиком организације, вештинама, ...)
- Мере квалитета (безбедност, перформансе, употребљивост, ...)
- Пропратне компоненте (механизам логовања, ауторизација и аутентикација, управљање изузецима, ...)

Садржај

Архитектонски шаблони

- У неким случајевима, елементи архитектуре су укомпоновани тако да решавају одређене проблеме.
- Такве композиције су се показале корисним током времена и у различитим доменима због чега су документоване и називамо их архитектноски шаблони (енгл. *Architectural Patterns*).
- Шаблон описује типове елемената и њихов начин интеракције у решавању проблема.
- Софтверска архитектура је често комбинација више стилова који чине читав систем.

Неки архитектонски шаблони

- Слојевите архитектуре (енгл. *Layered Pattern*)
- Надовезивање са филтрирањем (енгл. *Pipe and Filter Pattern*)
- Архитектура заснована на догађајима
- Архитектура заснована на микрокернелима
- Архитектура заснована на микросервисима

Слојевите архитектуре

- Најчешће коришћен стил.
- Како би се постигло раздвајање одговорности, софтвер је подељен на слојеве који представљају главне компоненте.
- Подаци теку од највиших слојева ка најнижим (на дну се обично налази база података).
- Чест пример је трослојна архитектура где постоје презентациони слој, апликативна логика и складиште података.

Надовезивање са филтрирањем

- Многи системи имају задатак да трансформишу токове података. Многе трансформације се често понављају и пожељно је направити их као посебне делове који се многу поново искористити.
- Делови су слабо спрегнути, независни и могу се извршавати паралелно.
- Подаци добијени на улазу обрађују се кроз низ трансофрмација које извршавају филтери (енгл. *filters*) повезани преко канала (енгл. *pipes*).
- Филтери не морају да знају ништа једни о другима.
- Један од примера је UNIX терминал.

Архитектура заснована на догађајима

- Дистрибуирана архитектура
- Асинхроне природе
- Апликација је организована тако да неке компоненте генеришу догађаје док се друге региструју да буду обавештене када се неки догађај деси.
- Користи се за графички кориснички интерфејс, дебагер.

Архитектура заснована на микрокернелима

- ”Plug-in” архитектура
- Постоји неколико основних операција (микрокернел) које чине језгро система и користе се често.
- Остале операције (додаци, енгл. *plug-in*) користе микрокернел и имплементирају логику над њим.

Архитектура заснована на микросервисима

- Креирање великог броја мањих програма (севиса) уместо једног великог.
- Додавање нове функционалности постиже се креирањем новог малог сервиса.
- Користи се када су различити задаци лако раздвојиви.

Литература

- *L. Bass, P. Clements, R. Kazman* - **Software Architecture in Practice (Third edition)**
- *F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, M. Stal* - **Pattern-oriented Software Architecture: A System of Patterns**
- *M. Shaw, D. Garlan* - **Software architecture**