Princípy tvorby softvéru
Architektúra softvérového systému

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

Čo je to architektúra?

"Software architecture encompasses the set of significant decisions about the organization of a software system including the selection of the structural elements and their interfaces by which the system is composed; behavior as specified in collaboration among those elements; composition of these structural and behavioral elements into larger subsystems; and an architectural style that guides this organization. Software architecture also involves functionality, usability, resilience, performance, reuse, comprehensibility, economic and technology constraints, tradeoffs and aesthetic concerns."

Philippe Kruchten, Grady Booch, Kurt Bittner, and Rich Reitman

Čo je to architektúra?

"The highest-level breakdown of a system into its parts; the decisions that are hard to change; there are multiple architectures in a system; what is architecturally significant can change over a system's lifetime; and, in the end, architecture boils down to whatever the important stuff is."

Martin Fowler

Čo je to architektúra?

"The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. Architecture is concerned with the public side of interfaces; private details of elements—details having to do solely with internal implementation—are not architectural."

Len Bass, Paul Clements, Rick Kazman

SW architektúra

Architektúra by mala:

- Odhaliť štruktúru systému a skryť implementčné detajly
- Realizovať všetky use casy a scenáre.
- Brať do úvahy požiadavky všetkých stakeholderov.
- Zaoberať sa nie len funkčnými ale aj nefunkčnými, najmä kvalitatívnymi požiadavkami.

Odhaliť štruktúru systému: Ako popísať SW systém?

4 + 1 view model:

- Logický pohľad
- Procesný pohľad
- Vývojársky pohľad
- Fyzický pohľad

- +1 Use case pohľad

Diagram so závislosťami

Čo je teda súčasťou architektúry systému?

Use case pohľad

- Prehľad spôsobov použitia systému.
- Všetci aktori.
- Ktoré scenáre / use casy chceme analyzovať/implementovať ako prvé.

Čo je teda súčasťou architektúry systému?

Fyzický pohľad

- Podmienky nasadenia (OS, DB server, WEB server, spôsob komunikácie medzi nodami), použité technologie.
- V prípade aplikácie pracujúcej na mnohých počítačoch, často potrebujeme zhlukovať nody typov.

Príklad: Fyzický pohľad

Architektonický pattern 3-Tier architektúra:

- Prezentačná "Tier"
- Business logic "Tier"
- Dátová "Tier"

Príklad: Veľká webová aplikácia - Prezentačná vrstva

- Štruktúra nodov vo vrstve: nezávislé, komunikujúce s logickou vrstvou (ktorá z pohľadu prezentačnej vrstvy vyzerá ako jeden node).
- Podmienky nasadenia, zvolené technológie: Browser, Javascript, ...
- Použité komunikačné kanály: internet, HTTPS

Príklad: Veľká webová aplikácia - Logická vrstva

- Štruktúra nodov vo vrstve: Load balancer, nezávislé servery komunikujúce s datovou vrstvou.
- Podmienky nasadenia, zvolené technológie: kontainerizovana aplikácia, Java
- Použité komunikačné kanály a protokoly: internet, HTTPS

Príklad: Veľká webová aplikácia - Dátová vrstvna

- Štruktúra nodov vo vrstve: Load balancer pre čítanie, single entry node pre zápis, replikujúce sa nody pre čítanie, ...
- Podmienky nasadenia, zvolené technológie: . . .
- . . .

Príklad: Logický pohľad

- Rozdenelie funkcionality do komponentov a definícia interfacov medzi nimi

Interface
Interface

je kontrakt medzi komunikujúcimi stranami:

- Volajúca strana (vyžaduje/requires interface) má adresu volaného (web / memory) a dohodnutým protokolom (HTTP, C calling convention) volá funkciu interfacu.
- Volaná strana (poskytujúca/implements interface) vykoná nejakú akciu a prípadne dohodnutým protokolom vráti návratovú hodnotu.

Komunikácia medzi rozličnými nodami/komponentami má prebiehať iba za použitia striktne zvolených interfaceov.

- Stateless interface nevyžaduje poznať stav druhej strany.
- Inak súčasťou interfacu aj usage protocol (prechody medzi stavmi a v ktorom stave môže byť ktorá funkcia volaná)

Procesný pohľad

- Logický pohľad je statický a často nie je dostatočný pre vysvetlenie, ako má fungovať systém
- Procesný pohľad ponúka (kde je to potrebné) dinamický pohľad na správanie sa systému
- Kto má pod kontrolou vykonávanie programu? Kde je aký thread? Inversion of control?

Development pohľad

- Organizácia vývoja (VCS, package, názvy zdrojákov, programovací jazyk)
- Ktoré časti systému chceme analyzovať/implementovať ako prvé.

Poradie je pri tvorbe aplikácie dôležité

Ktoré časti systému chceme typicky analyzovať/implementovať ako prvé

- Veci, ktoré sú nejasné/neznáme/neisté (hlavné riziká)
- Veci, ktoré potrebuje používať zvyšok systému a ktorých zmena má veľký dosah na zvyšok systému (cross-cutting concerns: logovanie, autentifikácia, perzistencia, continious delivery)
- Najhodnotnejšie use casy

Ako nájsť architektúru?

Základné otázky?

- Kto sú aktori nášho systému a ako ho budú používať?
- Ako bude aplikácia nasadzovaná a spravovaná?
- Ake sú kvalitatívne atribúty aplikácie ako: bezpečnosť, výkon, konkurentnosť, internacionalizácia, konfigurovateľnosť?
- Ako možno prirodzene rozdeliť aplikaciu na menšie časti?
- Ako nadizajnovať aplikáciu aby bola flexibilná?
- Aké sú súčastné súčastné a budúce trendy? (obľúbené technológie, . . .)

Ako nájsť architektúru?

Ďalšie otázky

- Ktoré základné časti architektúry predstavujú najväčšie riziko v prípade nesprávej voľby riešenia?
- Ktoré časti architektúry sa majú veľký predpoklad meniť, dizajn ktorých častí možno odkladať bez veľkých dôskedkov na neskôr?
- Čo sú kľúčové predpoklady v projekte a ako ich plánujeme testovať?
- Za akých podmienok bude nevyhnutné refaktorovať dizajn?

Ako nájsť architektúru?

Je potrebné určiť

- Application Type (mobile app, rich client, rich internet, service, web app, cloud, ...)
- Deployment Strategy
- Appropriate Technologies
- Quality Attributes
- Crosscutting Concerns (authentication, logging, caching, ...)

Nie je možné sa sústrediť na úplne všetko.

Kľúčový scenár:

- Reprezentuje výraznú neznámu oblasť alevo oblasť kde je známe výrazné riziko.
- Reprezentuje architektonicky signifikantný use case.
  - Business Critical.
  - Vysoký dopad.
- Reprezentuje prienik kvalitatívnych atribútov a funkcionality.
- Reprezentuje trade-off medzi kvalitatívnymi požiadavkami.

Architektúra - rozklad na menšie časti

Ako rozbíjať veci na menšie časti?

- Dizajnové princípy ako
  - separation of concerns, single responsibility, principle of least knowledge, DRY, ...
- nemixovať v rámci celkov (layers/components/...) rôzne veci (design patterns, component types, data types).
- Je potrebné explicitne pooísať ako jednotlivé štrukturálne časti budú navzájom komunikovať.
- Jasný kontrakt medzi jednotlivými časťami.

Princípy SW architektúry

- Build to change instead of building to last.
- Znižovanie rizika.
- Používať modelovanie a vizualizáciu ako komunikačný nástroj.
- Identifikuj a zapíš kľúčové rozhodnutia, aj so stručným zdôvodnením.

Princípy SW architektúry - príklad

Ako byť flexibilný pri voľbe riešenia na ukladanie dát?

Encapsulate what varies:

- Vytvoriť komponent zodpovedný čisto za ukladanie dát.
- Interface prispôsobený potrebám klientov.
- Implementácia naviazaná na konkrétne riešenie (file system, databáza).
- V prípade zmeny, stačí upravovať tento malý komponent

Princípy SW architektúry

- Nesnažte sa mať všetko správne na prvýkrát (to sa môže týkať aj DÚ, ktoré vám zadám)
- Nadizajnovať na začiatok môže stačiť iba toľko aby ste mohli dizajn testovať.
- Iteratívne pridávajte detajly počas viacerých prechodov aby ste sa uistili, že pri veľkých rozhodnutiach sa rýchlo priblížite k správnym riešeniam.

Testing the architecture

- What assumptions have I made in this architecture?
- What explicit or implied requirements is this architecture meeting?
- What are the key risks with this architectural approach?
- What countermeasures are in place to mitigate key risks?
- In what ways is this architecture an improvement over the baseline or the last candidate architecture?

Architektonické štýly a patterny

- Provide abstract framework for a family of systems
- Help communication

Typy of architektonických štýlov

Je v tom riadny bordel. Architektúra pokrýva veľa rôznych concernov a pohľadov.

- Deployment
- Structure
- Communication
- Domain
- ...

Key architectural styles and patterns

- Client/Server
  Segregates the system into two applications, where the client makes requests to the server. In many cases, the server is a database with application logic represented as stored procedures.
- Peer-to-peer
- Component-Based Architecture
  Decomposes application design into reusable functional or logical components that expose well-defined communication interfaces.
- Domain Driven Design
  An object-oriented architectural style focused on modeling a business domain and defining business objects based on entities within the business domain.

Key architectural styles and patterns

- Layered Architecture
  Partitions the concerns of the application into stacked groups (layers).
- N-Tier / 3-Tier
  Segregates functionality into separate segments in much the same way as the layered style, but with each segment being a tier located on a physically separate computer.

Key architectural styles and patterns

- Service-Oriented Architecture (SOA)
- Message Bus

  An architecture style that prescribes use of a software system that can receive and send messages using one or more communication channels, so that applications can interact without needing to know specific details about each other.
- Microservices
- Model-View-Controller
- Object-Oriented

Key architectural styles

Benefits of respective styles

User interface ako komponent aplikácie

- User interface sa často mení
- Oddeliť user interface ako vlastný komponent úplne dokonale? (encapsulate what varies)

Resources

Microsoft Application Architecture Guide