Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 005.342
DOI: https://doi.org/10.24867/17OI01Djuricin

MIGRACIJA MONOLITNOG SISTEMA NA MIKROSERVISNU ARHITEKTURU

MIGRATING MONOLITHIC SYSTEM TO MICROSERVICE ARCHITECTURE

Sandra Đuričin, *Fakultet tehničkih nauka, Novi Sad*

Oblast – INŽENJERSTVO INFORMACIONIH SISTEMA

Kratak sadržaj – *Mikroservisna arhitektura postaje sve popularnija poslednjih nekoliko godina i sve se više kompanija opredeljuje za nju kao prvobitan izbor za kreiranje novih aplikacija ili prilikom unapređivanja postojećih monolitnih. Migracija sistema iz monolitne u mikroservisnu arhitekturu predstavlja veliki izazov i zato je potrebno posvetiti dosta vremena sagledavanju prednosti i mana, kao i odabiru metode kojom bi se ova migracija izvela.*

Ključne reči: *mikroservisi, mikroservisna arhitektura, migracija na mikroservisnu arhitekturu*

Abstract – *Microservice architecture has become increasingly popular in recent years. Therefore, more and more companies are opting for it as the initial choice for creating new applications or when upgrading existing monolithic ones. The migration of systems from monolithic to microservice architecture presents a great challenge, so it is necessary to devote a lot of time to consider the advantages and disadvantages of the transition to microservice architecture, as well as choosing the method by which this migration would be performed.*

Keywords: *microservices, microservice architecture, migration to microservice architecture*

1. UVOD

Monolitna arhitektura ima svoje prednosti i ponekad je dobar izbor za početničku arhitekturu ili kod sistema koji nemaju veliku mogućnost proširenja [1]. Kako sistem najčešće tokom svog životnog veka ima tendenciju da raste, njegovo održavanje postaje sve teže [1], [2]. Mikroservisna arhitektura postaje sve popularnija i sve se više malih i velikih preduzeća opredeljuje za nju [3]. Takođe je česta pojava da kompanije svoje monolitne sisteme migriraju u mikroservisne [3], dok pak postoji i mišljenje da je nekad bolje ostati na monolitnoj kako bi se izbegli tehnički i organizacioni izazovi koji dolaze sa promenom arhitekture [1]. Zbog toga izbor odgovarajuće arhitekture za određen sistem uvek predstavlja izazov za programere [3]. Uprkos prednostima koje mikroservisna arhitektura pruža, mnoge kompanije ipak odbijaju migraciju plašeći se potencijalnih problema ili jer još uvek nisu upoznate sa svim benefitima koje ova arhitektura pruža [3].

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Teodora Lolić, docent.

Ovaj rad razmatra nekoliko tehnika migracije koje su autori koristili kako bi izvršili migraciju na mikroservisnu arhitekturu, kao i njihove prednosti i mane.

2. TEORIJSKE OSNOVE

Teorijskim osnovama će biti obuhvaćeni opisi monolitne i mikroservisne arhitekture, kao i njihove osnovne odlike.

2.1. Monolitna arhitektura

Monolitna arhitektura je tradicionalan način za razvoj softvera koji su koristili svetski poznati Internet servisi kao što su Netflix, Amazon i eBay, a zalaže se za enkapsulaciju svih funkcija u jednu aplikaciju [2], [4]. Manje složene monolitne aplikacije imaju svoje prednosti: jednostavne su za razvoj, testiranje, primenu i skaliranje. Ipak, svaka uspešna aplikacija će usled rasta vremenom postati prevelika [2]. Iako je implementacija takve aplikacije laka, veliku monolitnu aplikaciju je često teško razumeti i održavati [4]. Osim toga, svaka promena u monolitnoj aplikaciji dovodi do ponovnog pokretanja cele aplikacije.

2.2. Mikroservisna arhitektura

Mikroservisna arhitektura pomaže u rešavanju problema složenosti velikih aplikacija tako što ih dekomopozira u manje nezavisne delove – mikroservise [4]. Mikroservisne aplikacije se sastoje iz skupa malih, izolovanih i autonomnih servisa razvijenih zasebno, sa jasno definisanim zadatkom koji obavljaju [3], [1], [5], [4]. Mikroservisi komuniciraju putem poruka, koristeći standardne formate podataka i protokole [4].

3. SISTEMATSKI PREGLED LITERATURE

Pre odabira primarnih studija neophodno je izvršiti sistematski pregled literature kojim se analizira trenutno stanje iz oblasti migracije monolitnih sistema na mikroservise. Na početku sistematskog pregleda literature postavljaju se istraživačka pitanja čiji odgovori bi trebalo da omoguće izdvajanje različitih pristupa rešavanju problema migracije monolitinih sistema na mikroservise, kao i sumiranje prednosti i mana koje svaki pristup donosi sa sobom. Potom je potrebno definisati istraživačku strategiju koja podrazumeva odabir ključnih reči za pretragu, kao i kriterijuma inkluzije i ekskluzije za filtriranje radova pronađenih u odabranim bibliografskim bazama u cilju određivanja pogodnosti radova za kandidate za primarne studije.

3.1. Istraživačka pitanja

U nastavku su data istraživačka pitanja.

- Koje su prednosti i mane mikroservisne arhitekture?

- Na koje načine je moguće transformisati monolitnu arhitekturu sistema u mikroservisnu?
- Koje su prednosti i mane tehnika koje predlažu autori prilikom prelaska iz monolitne u mikroservisnu arhitekturu?

3.2. Istraživačka strategija

Ključne reči korišćene za pretragu su: dekompozicija monolitnih sistema; mikroservisna arhitektura; migracija monolitnih sistema; migracija na mikroservisnu arhitekturu.

U nastavku su dati kriterijumi selekcije (inkluzije i ekskluzije) koji su primenjeni prilikom odabira primarnih studija.

1) Inkluzija:
   - radovi iz naučnih časopisa i konferencija,
   - radovi objavljeni nakon 2015. godine,
   - radovi iz oblasti softverskog inženjerstva.
2) Ekskluzija:
   - radovi za koje je utvrđeno da se njihova tema ne poklapa sa temom ovog rada,
   - duplirani radovi koji se pojavljuju u obe bibliografske baze podataka,
   - radovi koji nisu napisani na engleskom jeziku, odnosno radovi kod kojih je samo abstrakt na engleskom jeziku.

3.3. Literaturni izvori

Za pretraživanje radova korišćene su indeksne baze *Scopus* i *Google Scholar*. . Pretraga je ograničena tako da prikazuje samo radove u kojima se ključne reči za pretragu nalaze u naslovima, abstraktima i ključnim rečima. Nakon što je definisana strategija pretraživanja, korišćenjem logičkih operatora *AND* i *OR* u kombinaciji sa ključnim rečima dobijeni su stringovi za pretragu u indeksnim bazama koji su prikazani u tabeli 1.

Tabela 1. *Stringovi za pretragu u indeksnim bazama*

Scopus	TITLE-ABS-KEY ((("decomposition" ("microservice" OR "microservices")) OR (("monolith" OR "monolithic") ("microservice" OR "microservices")) OR (("migration" OR "migrating") ("microservice" OR "microservices")))) AND PUBYEAR > 2015
Google Scholar	((“decomposition” (“microservice” OR “microservices”)) OR ((“monolith” OR “monolithic”) (“microservice” OR “microservices”)) OR ((“migration” OR “migrating”) (“microservice” OR “microservices”)))

Inicijalnom pretragom *Scopus* indeksne baze dobijeno je 575 rezultata, a *Google Scholar* baze oko 2.180. Radovi su sortirani po relevantnosti. U obzir je uzeto prvih 60 radova iz svake bibliografske baze. Na osnovu naslova i abstrakta radova izvršena je dalja selekcija gde su izbačeni radovi koji nisu sa konferencija ili iz naučnih časopisa, koji nisu na engleskom jeziku ili kod kojih je bio dostupan samo abstrakt. Takođe su izbačeni i radovi koji nisu bili iz oblasti softverskog inženjerstva. Nakon toga je usledila selekcija radova na osnovu čitanja kompletnog rada. U ovom koraku su izbačeni svi radovi koji se nisu usko bavili migracijom monolitne arhitekture u mikroservisnu, gde tehnika migracije nije bila dovoljno detaljno i jasno opisana ili gde nisu bile navedene prednosti i mane mikroservisne arhitekture. Od 120 radova izabrano je 5 primarnih [2] – [6] koji ispunjavaju sve zahteve i kriterijume inkluzije i ekskluzije.

4. DISKUSIJA

Nakon odabira najpogodnijih kandidata za primarne studije, pristupljeno je njihovoj analizi u cilju definisanja prednosti i mana mikroservisne arhitekture, kao i različitih tehnika dekompozicije i njihovih prednosti i mana.

4.1. Prednosti i mane mikroservisne arhitekture

Autori [3] kao glavne prednosti migracije sistema na mikroservisnu arhitekturu navode lakoću održavanja softvera, skalabilnost i raspodelu odgovornosti između nezavisnih timova [3], [2], [6]. Mikroservisna arhitektura omogućava smanjivanje kompleksnosti monolitnog sistema dekompozicijom sistema na nezavisne servise, odnosno omogućava programerima da vrše izmene i testiraju servise nezavisno od drugih programera, što olakšava proces razvoja softvera [3]. Takođe, podela koda na manje delove, odnosno mikroservise doprinosi lakšem razumevanju i čitanju koda [3], [6]. Skaliranje mikroservisne arhitekture je mnogo lakše od monolitne arhitekture [3].

Mikroservisna arhitektura omogućava da svaki mikroservis bude postavljen na različit server ili pak bude napisan na različitom programskom jeziku [3], [6], [5]. Mikroservisi se mogu razvijati i održavati od strane različitih timova, nezahtevajući koordinaciju između timova [3], [6], [5]. Svaki tim može biti odgovoran za svoj deo, odnosno mikroservis, a na taj način je moguće veliki kod podeliti u nekoliko manjih celina sa jasnim granicama i tako olakšati rad i povećati produktivnost programera [3].

Na manjim celinama je lakše otkloniti kvar, pri čemu se kvar nekog mikroservisa vrlo često otklanja njegovom zamenom sa istim mikroservisom prethodne verzije bez potrebe da se restartuje ceo sistem [3]. Mikroservisi kao manje celine se brže razvijaju i lakše je na njima isprobavati nove tehnike i alate.

Mikroservisna arhitektura pored svojih dobrih strana poseduje mane i izazove [2]. Jedan od najvažnijih je kako efikasno izvršiti dekompoziciju monolitne arhitekture u mikroservise [2]. Odabir loše tehnike dekompozicije dodatno otežava ovaj proces i donosi dodatne izazove [2]. Ukoliko monolitni sistem iz nekog razloga prestane da radi, to se vrlo brzo može uočiti jer ceo sistem prestaje da radi. U mikroservisnoj arhitekturi ukoliko dođe do pada nekog servisa, drugi servisi nastavljaju svoj normalan rad što dovodi do malo otežanog uočavanja problema [6].

Takođe, vrlo često se dizajniranje ovakvih sistema vrši intuitivno i na osnovu prethodnog iskustva programera – ukoliko dođe do pogrešne dekompozicije to može rezultovati dodatnim troškovima [4]. Svaki mikroservis mora da komunicira preko mreže, što otežava kompleksnost implementacije [3]. Procena potrebnog vremena za razvoj mikroservisnog sistema mnogo je manje precizna u odnosu na procenu za monolitni sistem [3].

4.2. Tehnike dekompozicije korišćene u primarnim studijama

Autori primarnih studija su koristili različite tehnike dekompozicije monolitnog sistema na mikroservise. Kako je jedan od ciljeva istraživanja sumiranje prednosti i mana različitih tehnika, neophodno je izvršiti njihovu detaljnu analizu.

1) Tehnika dekompozicije korišćena u radu [5]

Autori [5] su u svom radu opisali metodu za dekompoziciju monolitnog sistema na mikroservise koja se zasniva na podeli monolitnog sistema na tri glavna dela: korisnički, serverski i bazu podataka. Takođe se pretpostavlja da je sistem struktuiran na manje logičke celine i da svaka od njih poseduje svoju poslovnu logiku. Formalno su sistem opisali koristeći tri skupa F, B i D. F je skup "fasada" , odnosno ulaznih tačaka u sistem, koje se koriste za komunikaciju korisničkog sa serverskim delom. Skup B predstavlja poslovnu logiku u serverskom delu i zavisi od tabela u bazi podataka (skup D). Takođe je potrebno naglasiti da je sistem podeljen na nekoliko poslovnih oblasti, gde je svaka odgovorna za jedan poslovni proces. Svakoj poslovnoj oblasti odgovara jedan podsistem.

U prvom koraku je izvršeno mapiranje tabela baze podataka sa odgovarajućim podsistemima. U drugom koraku je kreiran graf zavisnosti kod kog čvorovi predstavljaju fasade, poslovne funkcije i tabele u bazi podataka. Veze u grafu su pozivi od fasada ka poslovnim funkcijama, pozivi između poslovnih funkcija i pristupi od strane poslovnih funkcija ka tabelama. U trećem koraku se vrši identifikacija parova fasada i svih tabela sa kojima su one povezane preko poslovnih funkcija. U četvrtom koraku se vrši odabir parova iz prethodnog koraka na osnovu kriterijuma pripadnosti tabela nekom od podsistema definisanih u koraku jedan. U petom koraku se vrši indentifikacija potencijalnih kandidata za mikroservise tako što se za svaki par iz prethodnog koraka analizira kod fasada i poslovnih funkcija koje se nalaze na putanji od čvora fasade do čvora tabele tog para. U šestom koraku se kreira *Application Programming Interface* (API) mrežni prolaz kako bi migracija ka mikroservisima bila potpuno transparetna za korisnike.

2) Tehnika dekompozicije korišćena u radu [2]

Autori [2] su imali za cilj da smanje kompleksnost dekompozicije monolitnog sistema na mikroservise koristeći poluautomatizovani pristup. Pristup se zasniva na korišćenju dijagrama toka podataka iz poslovne logike kako bi se definisali mikroservisi. U prvom koraku je potrebno ručno konstruisati prečišćen dijagram toka podataka (DTP) — on se fokusira samo na operacije, dok isključuje sporedne informacije poput skladišta podataka i eksternih entiteta. U drugom koraku, prečišćen DTP treba pretvoriti u razgradiv DTP. Nakon kreiranja razgradivog dijagrama vrlo lako se identifikuju svi kandidati po principu jedna operacija i njen izlazni podatak predstavljaju jedan mikroservis.

3) Tehnika dekompozicije korišćena u radu [4]

Autori [4] u svom radu navode da je pre identifikacije mikroservisa neophodno kreirati model sistema. Ovakav model se sastoji od konačnog skupa sistemskih operacija i sistemskog stanja prostora. Sistemske operacije su javne operacije sistema, poput onih od kojih se sastoji API i koje pružaju odgovor sistema na neki od spoljašnjih okidača. Stanje prostora je skup sistemskih promenljivih koje sadrže informacije koje se upisuju ili čitaju pri izvršavanju operacija.

Razgradnja na mikroservise se postiže particionisanjem sistemskih promenljivih stanja u nepovezane skupove tako da operacije svakog mikroservisa mogu direktno pristupati samo svojim lokalnim promenljivim. Ukoliko je potrebno da mikrosistem izvršava operacije (pisanje ili brisanje) nad promenljivom iz drugog stanja prostora, to se postiže isključivo pomoću API mikroservisa koji sadrži stanje prostora o kom je reč. Na ovaj način je moguće lako prepoznati šta bi bila dobra dekompozicija, odnosno dekompozicija kojom se postiže nizak nivo međusobne povezanosti mikroservisa i visok nivo kohezije pojedinačnih mikroservisa. Sistemske operacije i sistemske promenljive stanja se definišu na osnovu slučaja upotrebe, a u tabelu se unose odnosi između svake sistemske operacije i promenljive nad kojom se ona koristi. U ovoj tabeli svaka ćelija pokazuje da li operacija vrši pisanje, čitanje ili ne radi ništa od ta dva nad promenljivom stanja.

Prvi korak u definisanju operacija i promenljivih je pronalaženje svih glagola u neformalnom opisu slučaja upotrebe (za operacije) ili imenica (za promenljive stanja). Putem grafa je moguće mnogo lakše identifikovati klastere sa gustim vezama koji su slabo povezani sa drugim takvim klasterima. Svaki klaster koji ispunjava te uslove se smatra dobrim kandidatom za mikroservis. To znači da je količina informacija koje takav mikroservis deli sa ostatkom sistema mala, što ga čini zaštićenim od promena u ostatku sistema, kao i ostatak sistema od promena u njemu. Pored toga, unutrašnje veze su kod takvih kandidata guste, što u većini slučajeva ukazuje na to da klaster predstavlja kohezivnu funkcionalnu jedinicu. Čvorovi grafa predstavljaju sistemske promenljive stanja i operacije, dok veze povezuju operaciju i promenljivu ako i samo ako operacija čita ili ažurira vrednost promenljive. Pored toga, svakoj vezi se dodeljuje težinski koeficijent pri čemu je on niži u slučaju da se vrši čitanje, a viši u slučaju da se vrši pisanje. Na ovaj način se vrši grupisanje podataka sa operacijama koje ih menjaju.

4) Tehnika dekompozicije korišćena u radu [6]

Pristup prikazan u radu autora [6] iziskuje proširenu verziju zapisa logova tokom korišćenja aplikacije. Za svaku korisničku operaciju izvršenu sa korisničkog interfejsa ili sa ulazne tačke sistema, sve aktivnosti moraju biti ispraćene u log fajlovima. Kompletno izvršavanje mora biti ispraćeno u celosti od ulazne tačke, preko pristupa bazi podataka do rezultata vraćenog korisniku. Svaki log mora sadržati događaje koji ukazuju na ulaz i izlaz iz funkcije i pristupanje bazi podataka. Svaka stavka u logu mora imati vremensku oznaku i identifikator sesije da bi se napravila razlika između korišćenja iste klase ili metoda od strane različitih korisnika. Kada se kreira log fajl pristupa se dekompoziciji sistema. U prvoj fazi se vrši analiza putanja izvršavanja pri čemu se identifikuju najčešće korišćene putanje. Autori predlažu upotrebu alata *DISCO*

za grafički prikaz poslovnih procesa definisanih na osnovu log fajlova. Rezultat ove analize je grafički prikaz procesa sa svim klasama i tabelama baze podataka korišćenim u poslovnom procesu, kao i sa strelicama koje povezuju svaku klasu na osnovu podataka u logu.

Druga faza je analiza učestanosti putanja izvršavanja gde debljina strelice ukazuje na učestanost pozivanja između klasa. Izlaz iz ovog koraka je tabela gde su prikazane različite putanje izvršavanja sa učestanostima njihove upotrebe.

Treća faza jeste otklanjanje cikličnih zavisnosti pronađenih analiziranjem putanja izvršavanja iz tabele generisane u prethodnom koraku. U slučaju da postoje ovakve zavisnosti, softver arhitekte moraju da prodiskutuju sa timom programera kako bi bilo najbolje prekinuti ih. Rezultat nakon ovog koraka je prerađena verzija tabele putanja izvršavanja. U četvrtoj fazi se identifikuju različite mogućnosti dekompozicije vizuelnom analizom grafova. Mikroservis kandidati mogu imati zajedničke putanje i upravo to omogućava postojanje većeg broja različitih rešenja dekompozicije.

4.3. Prednosti i mane tehnika dekompozicije

U tabeli 2 dat je prikaz prednosti i mana tehnika dekompozicije korišćenih u radovima [2], [4], [5], [6].

Tabela 2. *Prednosti i mane različitih tehnika dekompozicije*

Rad	Prednosti	Mane i izazovi
[5]	Relativno laka podela na osnovu identifikacije podsistema prema poslovnim oblastima.	Ukoliko se tabela nekog podsistema koristi u još nekom podsistemu ili je neka poslovna operacija vezana za više poslovnih podsistema, ne može se izvršiti dekompozicija u mikroservise.
	Pogodna tehnika za sisteme bez opisanih slučajeva upotrebe jer je potreban samo pristup kodu i bazi podataka.	
[2]	Laka implementacija na osnovu DTP kreiranog prema poslovnoj logici.	Nije moguće primeniti ukoliko dijagram toka podataka nije definisan.
[4]	Moguće je izvršiti dekompoziciju u klastere visokog nivoa kohezije i niskog nivoa međusobne povezanosti.	Kvalitet neformalnog opisa direktno utiče na kvalitet dekompozicije i identifikaciju pogodnih kandidata za mikroservise.
[6]	Metod se može koristiti za dekompoziciju monolitnih sistema i za kontinualno praćenje ponašanja postojećih mikroservisa pri radu čime bi se otkrili mogući propusti.	Iziskuje vreme za isprobavanje situacija pri kreiranju loga, pri čemu je lako preskočiti neki slučaj upotrebe, čime bi kvalitet dekompozicije mogao biti značajno pogoršan.
	Informacije o učestanosti korišćenja metoda pomažu u sprečavanju kreiranja prevelikih mikroservisa sličnih monolitnim sistemima.	Moguće je da korišćeni alati pogrešno identifikuju procese, pošto je kompleksnost prikupljanja podataka visoka.

5. ZAKLJUČAK

Mikroservisni sistemi imaju znatno veću skalabilnost i lakši su za održavanje u odnosu na monolitne sisteme. U ovom radu sagledane su prednosti i mane mikroservisne arhitekture, kao i tehnika za migraciju na mikroservisnu arhitekturu. Uočeno je da uprkos prednostima koje ovaj stil arhitekture nudi, postoje i mane kao i izazovi sa kojima se treba suočiti prilikom migracije monolitnog sistema na mikroservisnu arhitekturu.

Pre donošenja odluke u vezi sa migracijom sistema, potrebno je detaljno razmotriti benefite i izazove za svaki sistem pojedinačno i tek onda doneti konačnu odluku. Takođe je vrlo važno svakom sistemu pristupiti individualno prilikom odabira tehnike migracije jer će od nje zavisiti uspešnost dekompozicije čitavog sistema.

Planirano je da se budućim istraživanjima obuhvate slučajevi neuspešnih migracija na mikroservisnu arhitekturu, tehnike koje su prilikom njih korišćene kao i potencijalne greške i razlozi neuspešne dekompozicije.

6. LITERATURA

[1] M. Kalske, N. Mäkitalo, and T. Mikkonen, “Challenges when moving from monolith to microservice architecture,” Current Trends in Web Engineering, pp. 32–47, 2018.

[2] R. Chen, S. Li, and Z. Li, “From monolith TO Microservices: A DATAFLOW-DRIVEN APPROACH,” 2017 24th Asia-Pacific Software Engineering Conference (APSEC), 2017.

[3] D. Taibi, V. Lenarduzzi, and C. Pahl, “Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation,” IEEE Cloud Computing, vol. 4, no. 5, pp. 22–32, 2017.

[4] S. Tyszberowicz, R. Heinrich, B. Liu, and Z. Liu, “Identifying microservices using functional decomposition,” Dependable Software Engineering. Theories, Tools, and Applications, pp. 50–65, 2018.

[5] A. Levcovitz, R. Terra, and M. T. Valente, “Towards a technique for extracting microservices from Monolithic enterprise systems,” arXiv.org, 10-May-2016. [Online]. Dostupno na: https://arxiv.org/abs/1605.03175. [Pristupljeno: 06-Sep-2021].

[6] D. Taibi and K. Systä, “From monolithic systems TO Microservices: A DECOMPOSITION framework based on PROCESS MINING,” Proceedings of the 9th International Conference on Cloud Computing and Services Science, 2019.

Kratka biografija:

Sandra Đuričin rođena je u Senti 1997. god. Master rad na Fakultetu tehničkih nauka iz oblasti Inženjerstvo informacionih sistema odbranila je 2021.god
kontakt: sandradjuricin9@gmail.com