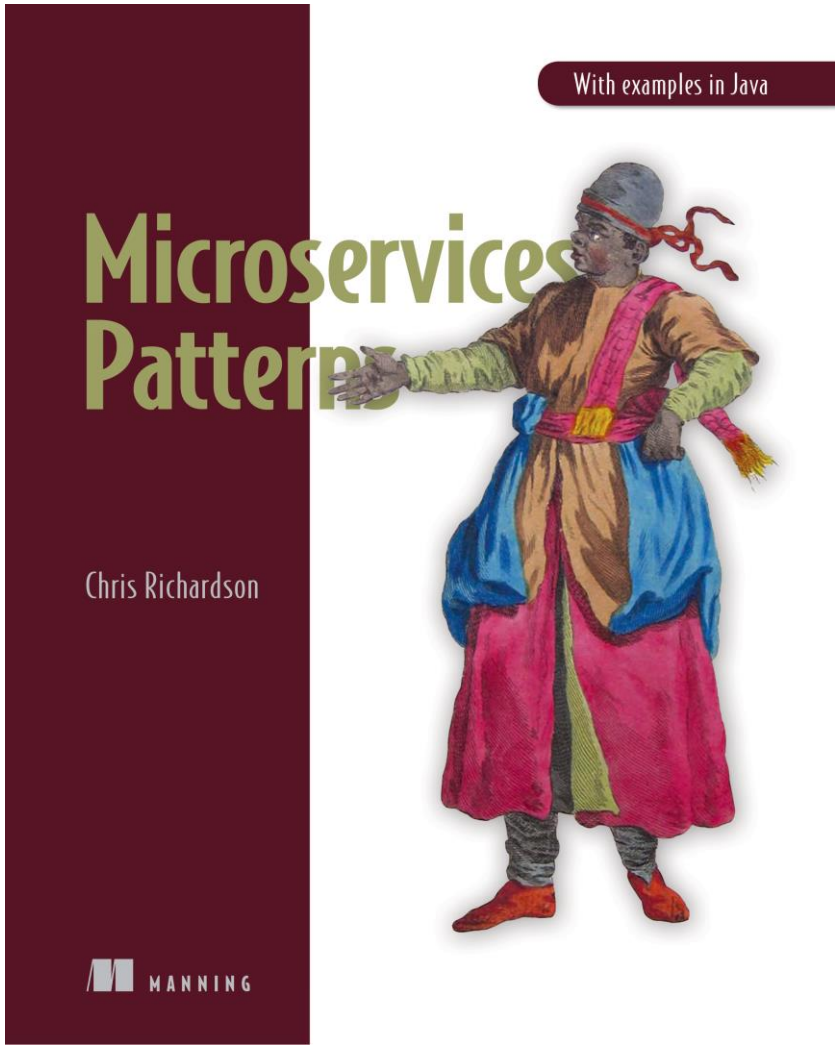
```
...
  }
</style>

<title>PUSGS | Programiranje u Smart Grid sistemima </title>
<span> Poznato i kao: Web 2</span>

<h1>Napustiti monolitne arhitekture</h1>

<div class="toolbar" role="banner">
...
```

Agenda

- **Literatura**
- **Kontekst imaginarnog softvera**
- **Simptomi i problemi monolita**
- **Karakteristike i prednosti mikroservisa**
- **Mikroservisni šabloni i zašto ih koristiti**

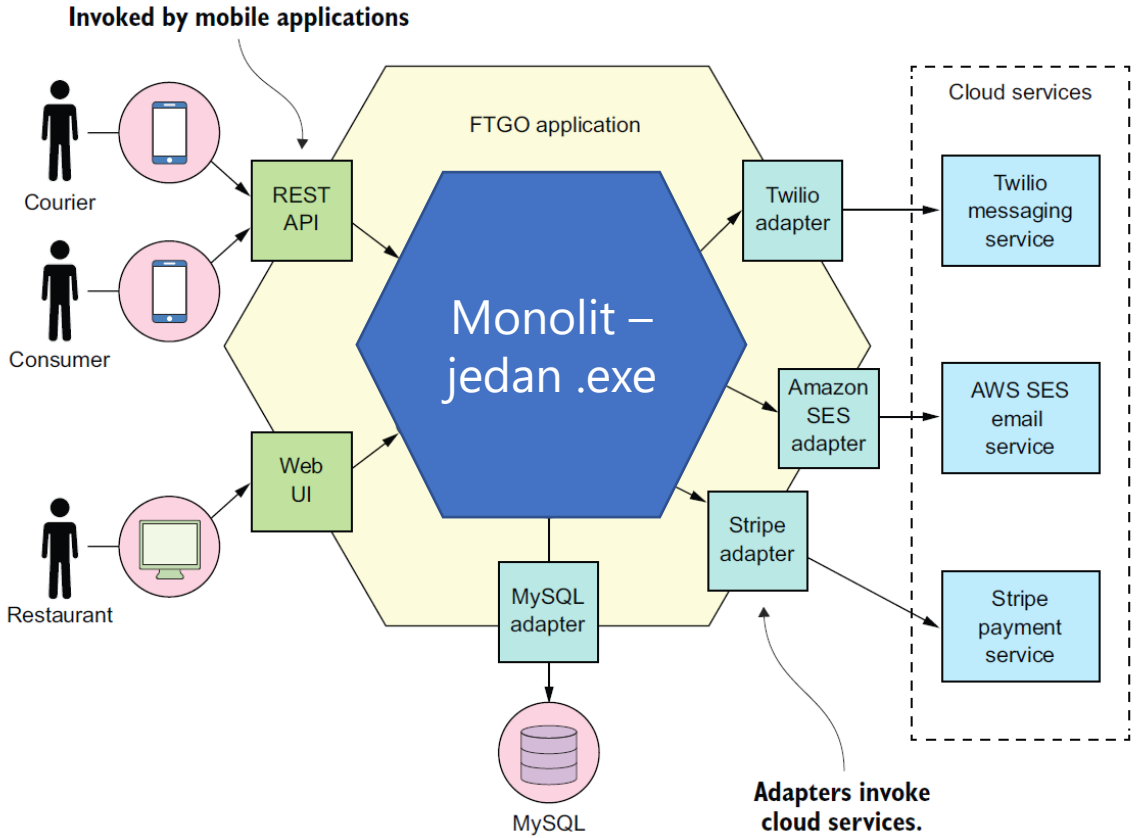
Literatura

- **Chris Richardson**
  - Web stranica
  - Knjiga ->
- Ostale reference će biti na slajdovima po potrebi
- Besplatna verzija za .NET:
  - Microservices architecture e-book



Kontekst

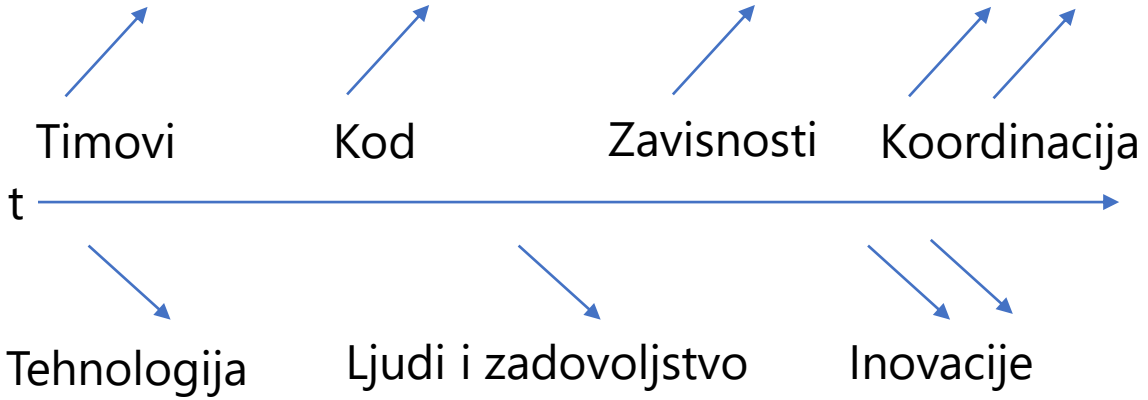
- **Moramo imati primer za razmatranje**
- **Hrana za poneti (HZP):**
  - Web sajt ili mobilni za poručivanje hrane
  - HZP koordiniše mrežu dostavljača
  - Plaća kurire i restorane
  - Restorani koriste HZP za menjanje menija
  - HZP koristi eksterne servise za plaćanje, poruke (twilio), kao i email
- **Monolitna struktura (jedan WAR/dll)**



Simptomi i problemi monolita

Monolit – dobre i loše strane

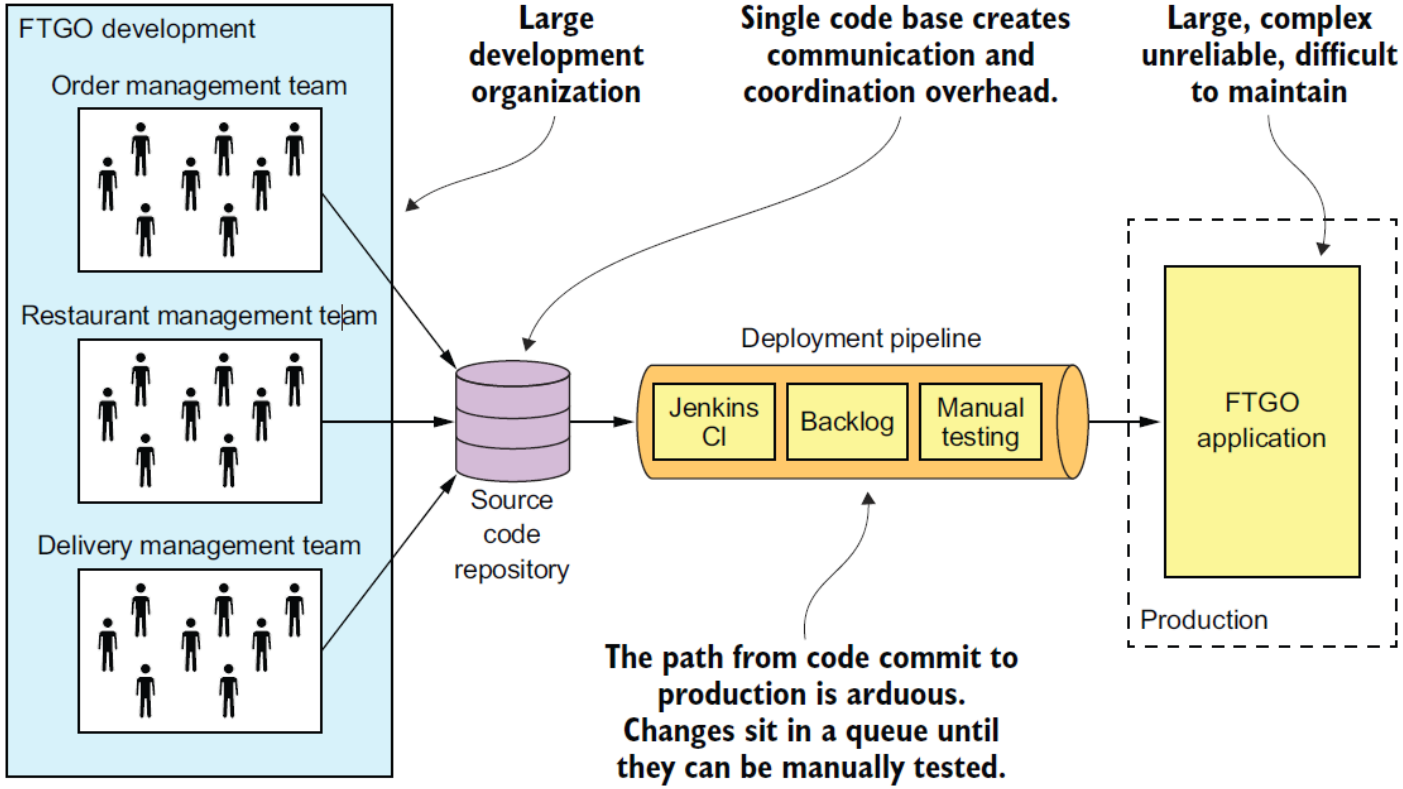
- Jednostavne za razvoj
- Jednostavno za radikalne izmene
- Testiranje (E2E). Selenium, REST API…
- Postavka – exe (WAR) se kopira
- Skaliranje. Stavi se više instanci iza balansera.



- Agilni razvoj i objavljivanje softvera postaju praktično nemoguće
  - Razvoj vrlo spor i sa posledicama nezadovoljstva kod zaposlenih
- Monolitne aplikacije su dobre za početak, ali posao preraste monolitne arhitekture

Propadanje organizacija zbog monolita

- Aplikacija postala kompleksna =>
- Niko je više ne razume =>
- Prave se nove greške =>
- Začarani krug počinje:
  - Spor razvoj
  - Put od komita do postavke
  - Vreme da se aplikacija podigne se uvećava
  - Kako postići zeleni dashboard?
  - CI zahteva ceo test
- Skaliranje (neki servisi CPU, neki memorija zbog in-memory baze podataka)
- Pouzdanost (ako jedna otkaže, sve otkazuju)
- Tehnologija zastareva (ne možemo probati malo druge tehnologije)



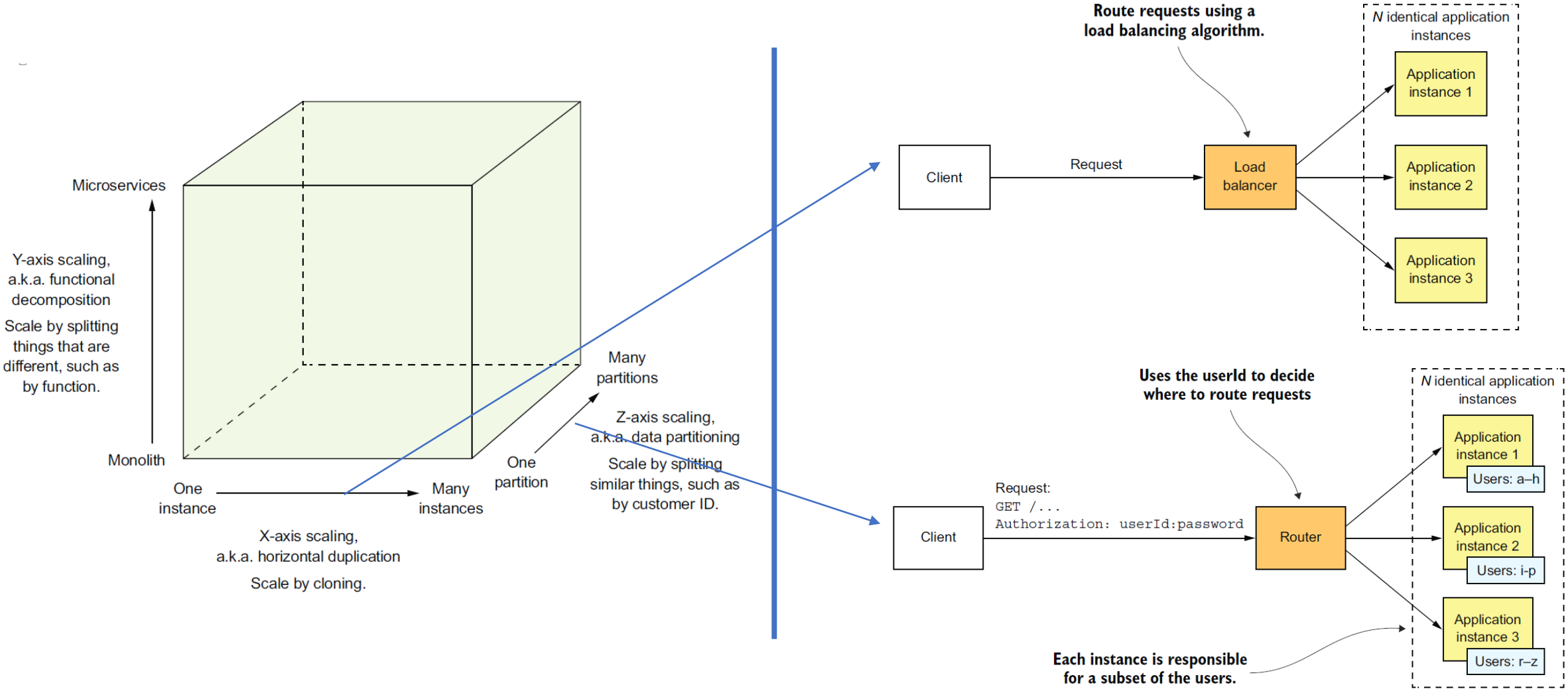
Kako se suočiti sa kompleksnošću?

Divide each difficulty into as many parts as is feasible and necessary to resolve it.

Rene Descartes (1596-1650)

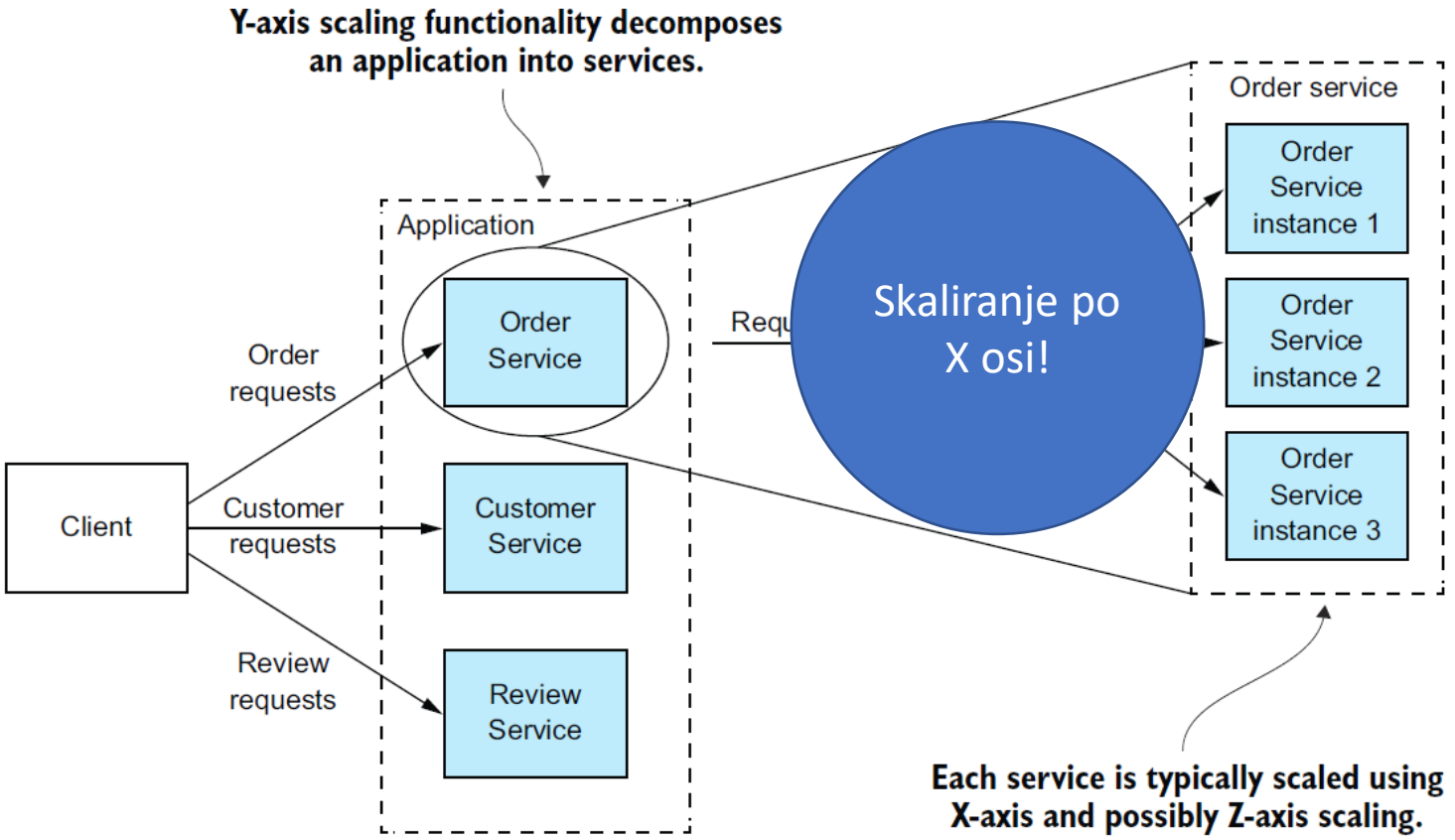
Karakteristike i prednosti mikroservisa

Kako skalirati (monolit)

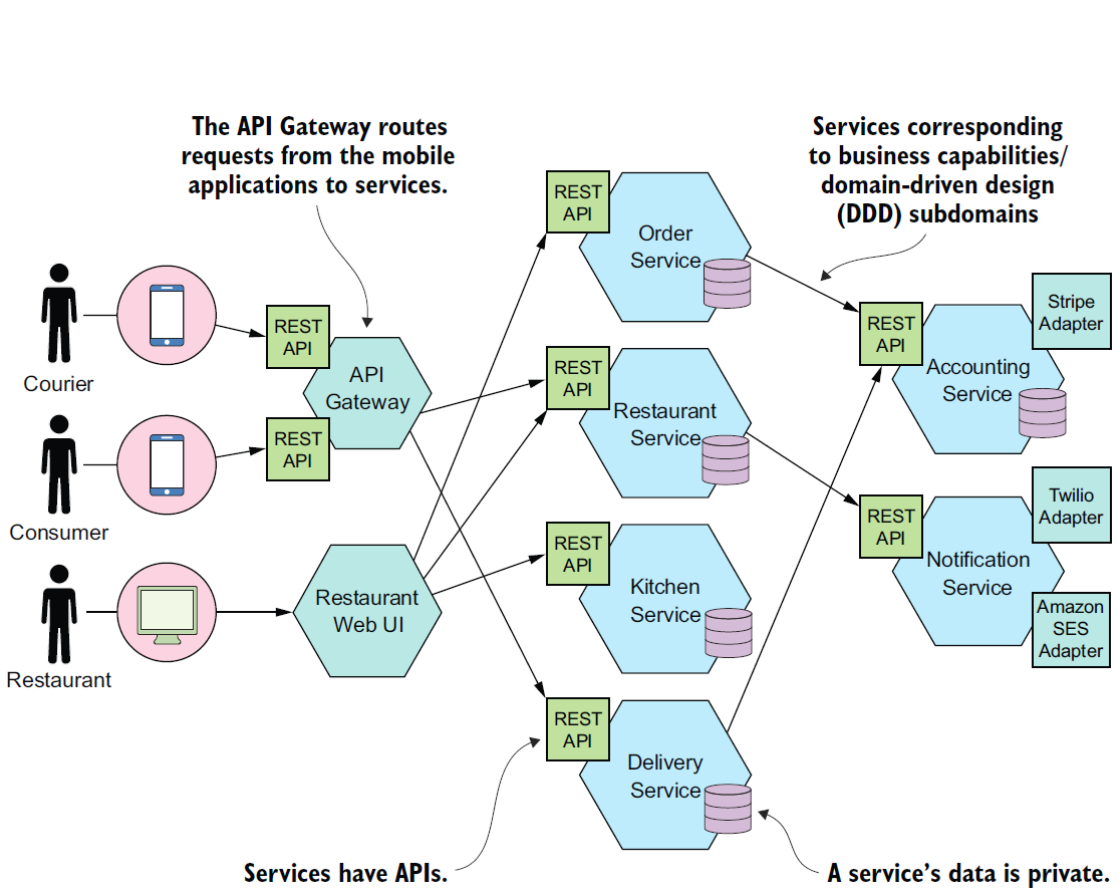


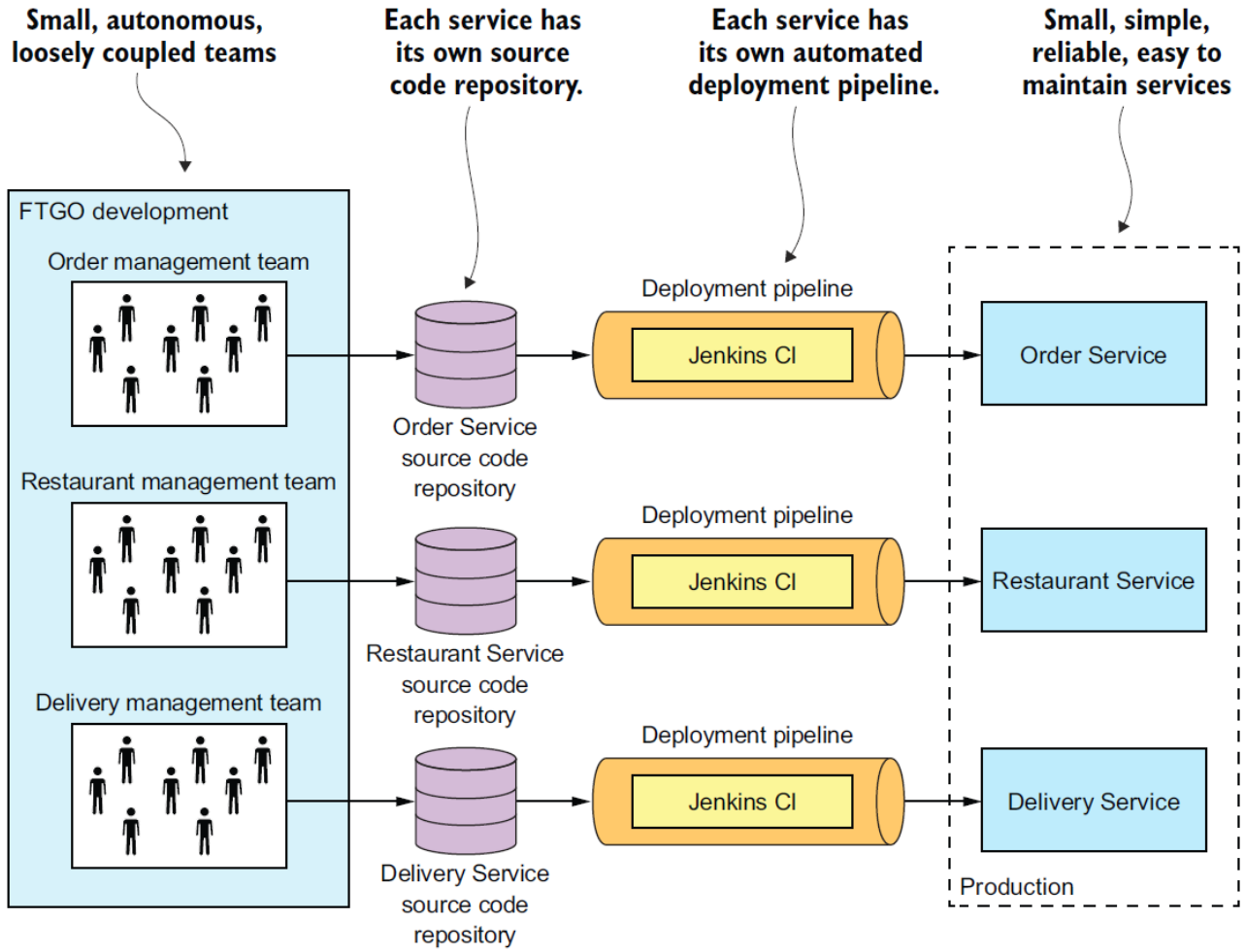
Kako skalirati (dekompozicijom)

- **Mikroservis kao modularnost**
  - Monolit koristi biblioteke I pakete, ali DevOps?
- Mikroservis ⇔ baza podataka
  - Ne sme jedan servis da blokira drugi!



HZP kao mikroservisna arhitektura





Prednosti i mane mikroservisa

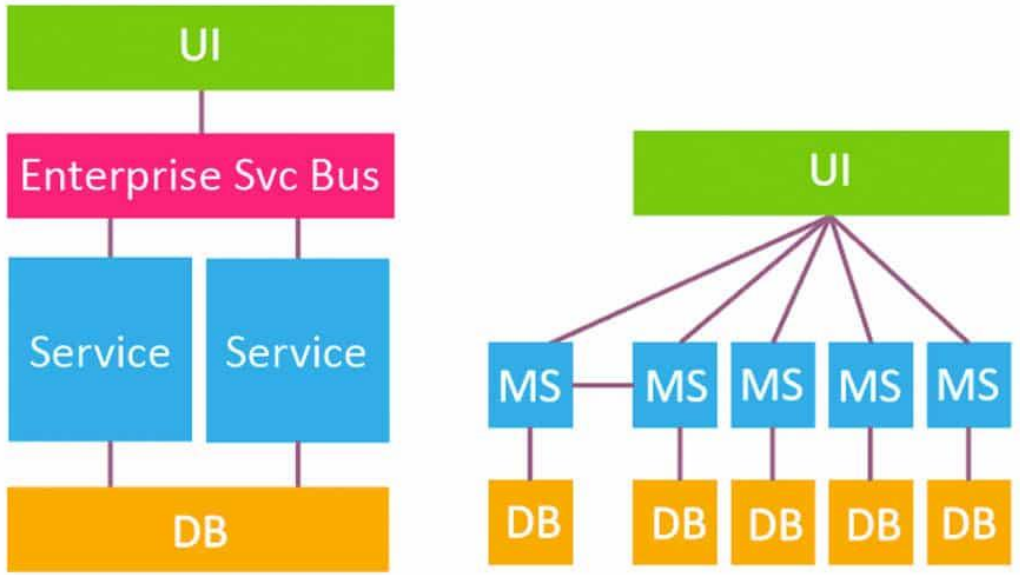
- **Prednosti:**
- Omogućava CI, lakšu postavku
- Mali servisi laki za održavanje
- Nezavisnost postavke
- Nezavisnost skaliranja
- Timovi autonomni
- Eksperimentisanje i usvajanje novih tehnologija
- Izolacija otkaza

- **Mane:**
- Kako podeliti servise, tj. dekompozicija?
  - Distribuirani monolit (kada više mikroservisa postane zavisno)
- Dodatna kompleksnost distribuiranih sistema
  - Interprocesna komunikacija
  - Transakcije
  - Upiti nad vise servisa
  - Postavka i upravljanje mikroservisima
- Kada krenuti

SOA ili mikroservisi?

- **SOA:**

1. Eksplicitne granice (jasni i publikovani interfejsi – WSDL)
2. Autonomija servisa (postavka i verzionisanje)
3. Servisi dele interfejse (kontrakte) i šeme, ne tipove (klase)
4. Kompatibilnost je određena polisama – komunikacioni protokoli i bezbednost



- SOAP ili WS* bazirani protokoli
- ESB za integraciju komunikacije
- Globalna baza podataka
- Servisi po veličini odgovaraju monolitima

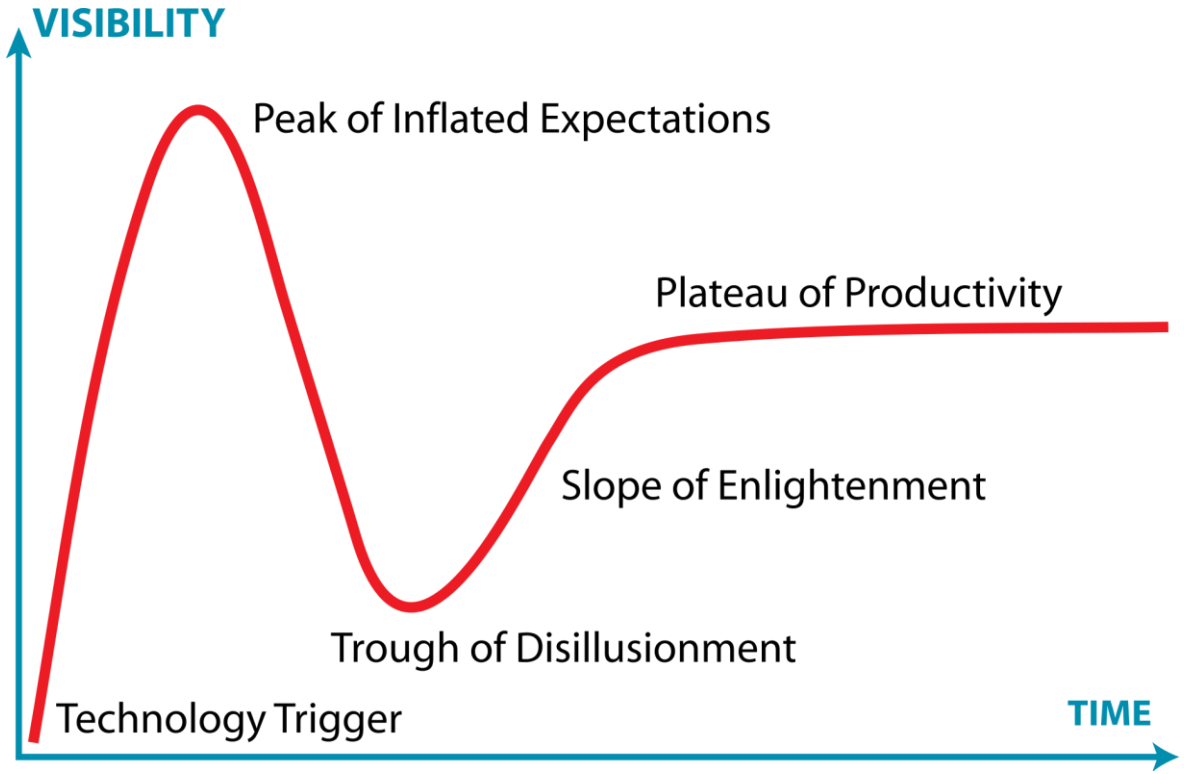
<- SOA
Mikroservisi ->

- REST ili drugi laki protokoli
- Direktna komunikacija
- Model po servisu
- Manje celine, obično jedan do dva manja tima

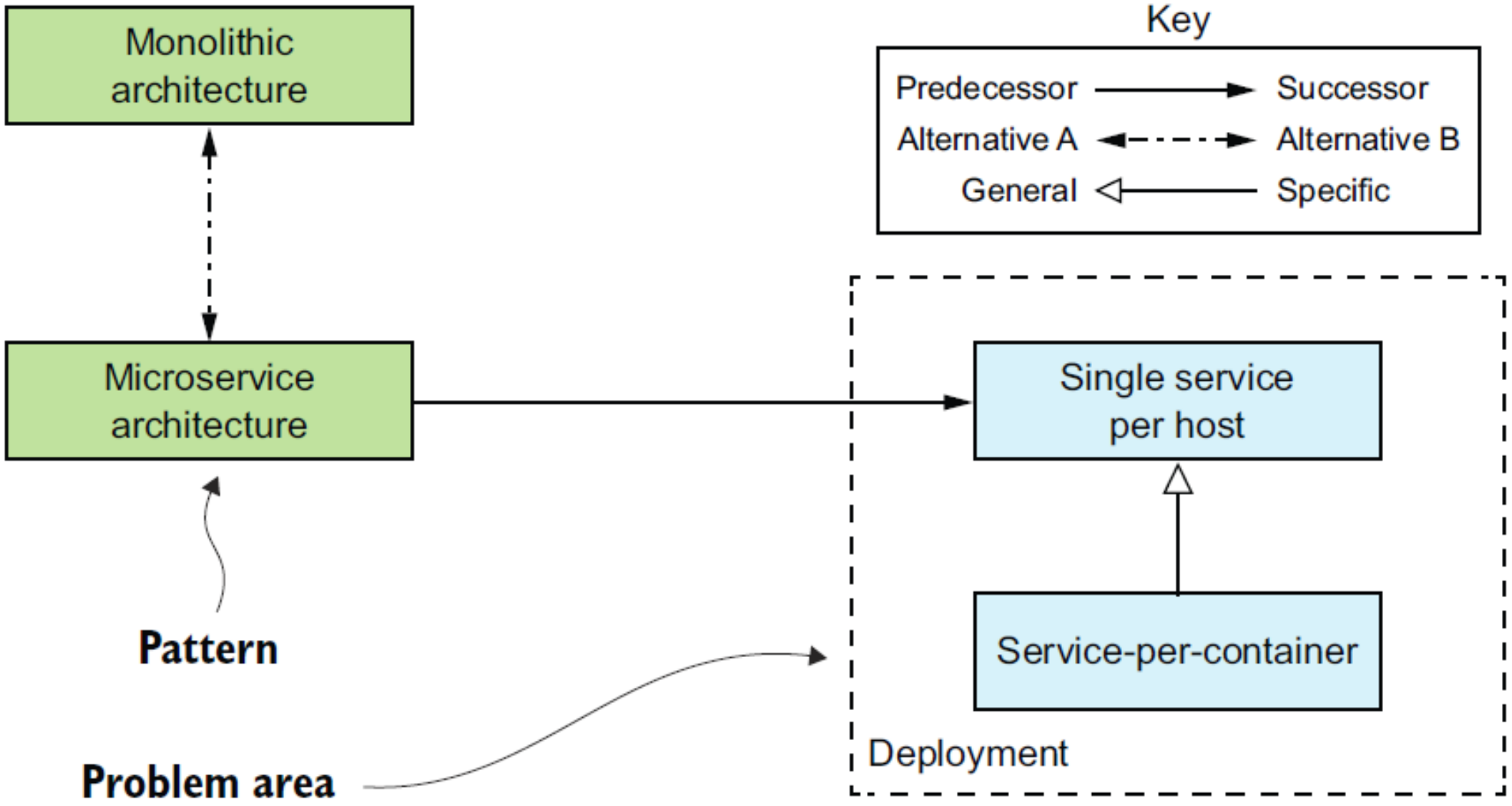
Mikroservisni šabloni i zašto ih koristiti

Arhitektura je odluka!

- Ne postoji jedno rešenje
- Monolit ili mikroservisna arhitektura
  - **Odluka zavisi od problema**
- Šabloni – Gang of Four za objektno?
- **Struktura šablona:**
  - Problemi (engl. *Forces*)
    - Npr. asinhroni (performanse ali i težina) ili sinhroni kod
    - Npr. višeorganizacijsko svojstvo
  - Rezultat primene (engl. *Resulting context*)
    - Prednosti, mane i problemi
  - Povezani šabloni



Povezani šabloni



Mikroservisni šabloni

Infrastrukturni:

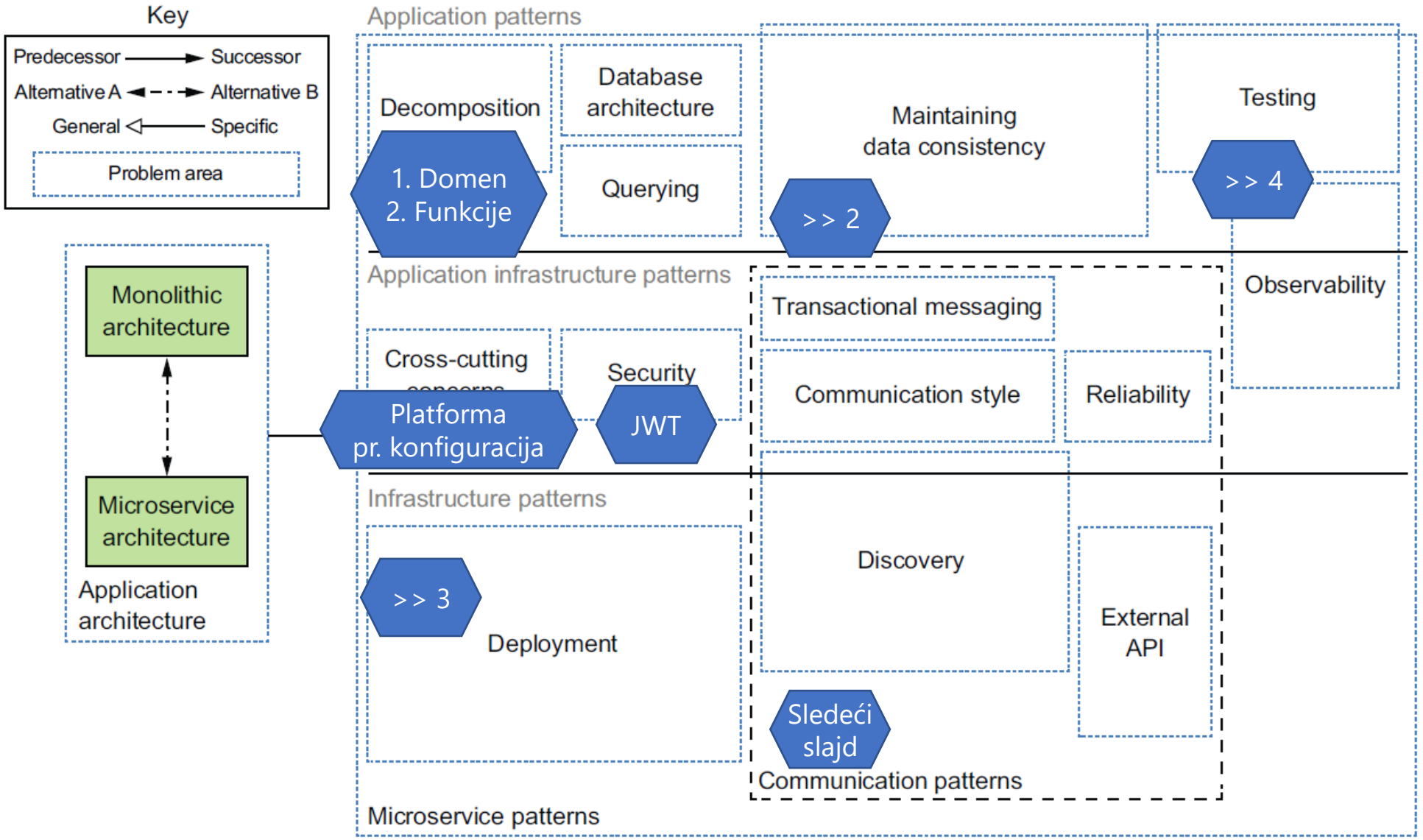
- Problemi sa infrastrukturom, nevezani za razvoj

Infrastruktura aplikacije:

- Problemi infrastrukture koji utiču na razvoj

Aplikativni šabloni:

- Rešavaju probleme inženjera softvera



Komunikacioni šabloni

Komunikacioni stilovi:

- Kako procesi međusobno komuniciraju?
- Ključno pitanje za nastavak

Otkrivanje servisa:

- Koji je IP servisa? Azure, docker, kubernetes…

Pouzdanost:

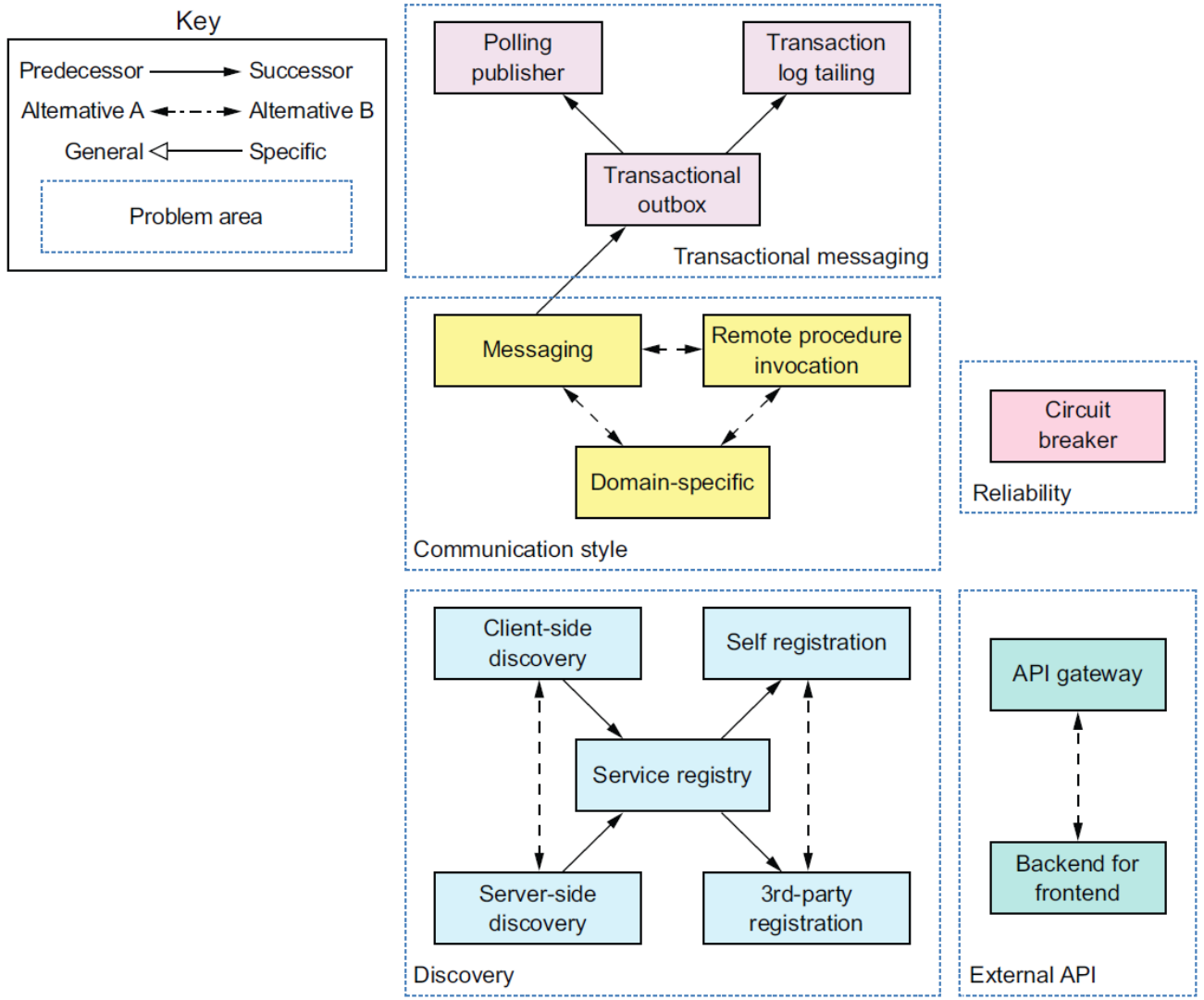
- Šta kada servisi nisu dostupni?

Transakcione poruke:

- Međuservisna komunikacija i transakciono pomeranje iz stanja u stanje? 2PC? Saga?

Eksterni API:

- Kako klijenti kontaktiraju naše servise?



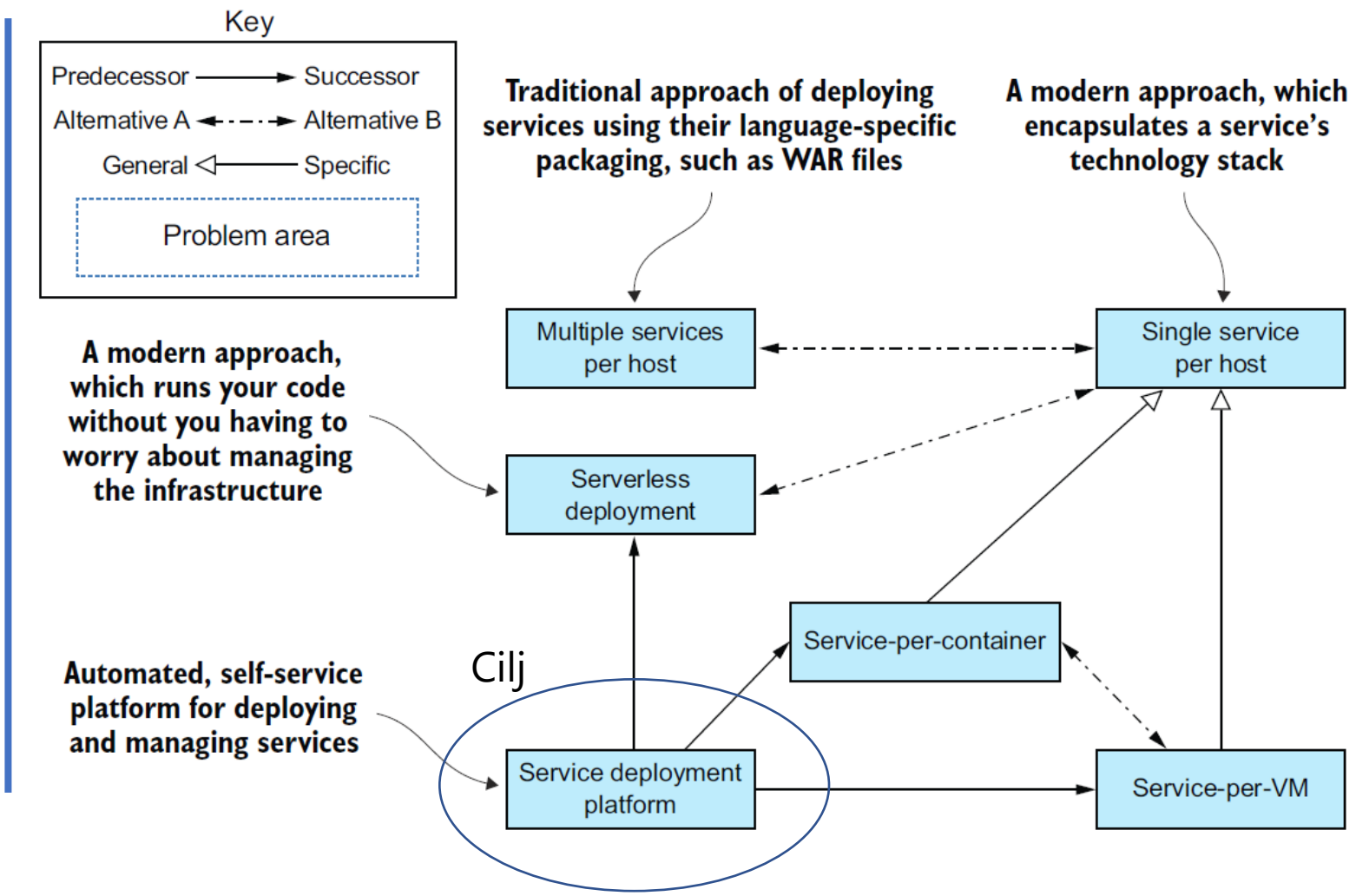
Šabloni postavke

Tradicionalno

- često manuelno i lako jer,
- postavlja se jedan .exe/dll/WAR

Mikroservisi

- vrlo komplikovano, zato postoje automatizovane platforme
- problem: odlučiti se za pravu?



Nadgledanje i testiranje

Pojasniti: SaaS ili tradicionalno

Provera zdravlja servisa preko posebnog API

Agregacija logova

- *Centralni log, alarmi, pretraga.*

Distribuirano praćenje

- *Svaki eksterni zahtev mora imati ID, i pregled šta se dešavalo po zahtevu*

Praćenje grešaka

- *Slično kao iznad, samo za programske greške*

Metrike aplikacije za tablu (TV)

Logovanje za potrebe inspekcije i neporecivosti (engl. Audit)

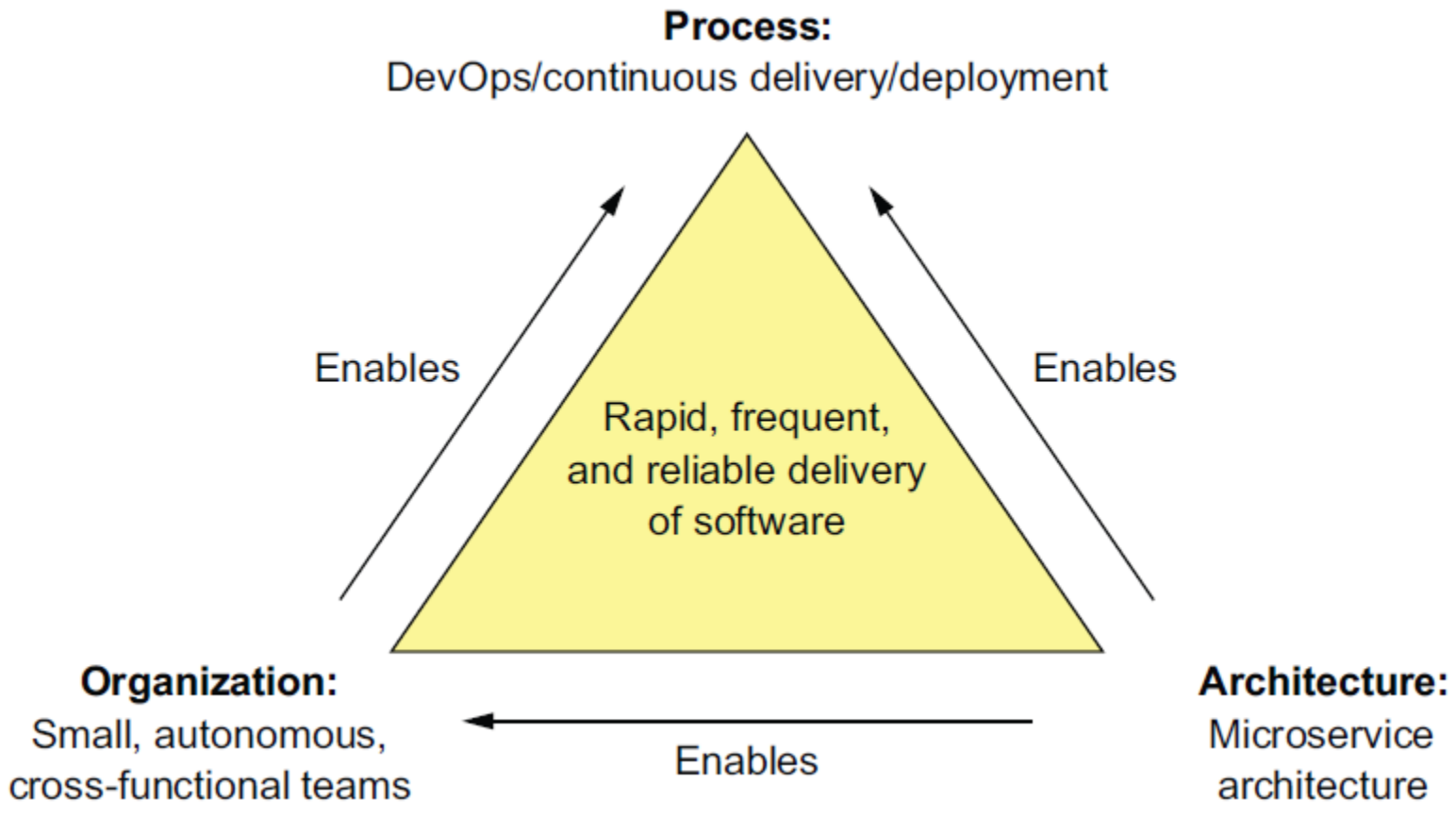
- *Pojasniti problematiku geopolitike kod ovakvih logova*

Testiranje:

1. Prema ugovoru sa klijentom (Cela aplikacija)
    1. „Use-cases“ testiranje
    2. SLA testiranje
2. Da li je servis uopšte dostupan klijentu
    1. Da li je u lancu zavisnosti u redu
3. Testovi samog servisa
    1. Kao nezavisne celine

Moramo imati širu sliku

1. Agilno (SCRUM) ili Tradicionalno (Vodopad)
2. Timovi koji rastu naspram produktivnosti je N ~ O(N2).
3. Tim za dve pice. Tim mora da zna sve da bi se smanjila zavisnost.
4. Konvejov zakon.
5. Vodopad i mikroservisi – gube se benefiti, poenta je da je softver uvek spreman za novu verziju.
6. **Problem:** arhitekturalni prelazak, kako utiče na organizaciju!



Zaključak

1. Monoliti ⇔ jedan .exe za postavku
2. Mikroservisi ⇔ nezavisni, svaki sa svojom bazom
3. Monoliti su u redu za male organizacije, kada kompleksnost krene da raste, treba razmotriti mikroservise
4. Fokusirati se na razvojnu organizaciju, mikroservisi omogućavaju skalabilnost timova
5. Mikroservisi ⇔ kompleksni, potrebna platforma, postoje nedostaci (brbljivi servisi)
6. Mikroservisni šabloni olakšavaju analizu za donošenje pravih odluka u slučaju novih aplikacija
7. Ljudski faktori za usvajanje nove arhitekture su važni