# Domain Driven Design

## Uvod u Domain Driven Design

**Domain Driven Design (DDD)** predstavlja pristup u razvoju softvera koji se fokusira na razvoj **modela domena nekog problema.** Pritom se teži razumevanju procesa i pravila samog domena.

DDD pruža principe i šablone za rešavanje složenih problema. Tako se dobija **jasan, čist, testabilan kod** koji predstavlja neki domen. Međutim, glavni cilj nije pisanje koda, već **rešavanje problema**. To zahteva puno rada sa klijentima kako bi se shvatile njihove potrebe i razumeo domen problema. Neophodan je neprestani rad tima za razvoj softvera sa domen ekspertima-ljudima koji su stručnjaci za određeni domen.

Jedan od glavnih principa prilikom rada sa DDD-em je: **“Divide and conquer“** – “Zavadi pa vladaj“. Ovaj princip je takođe poznat i kao: “**Separation of Concerns**“ – razdvajanje odgovornosti. Odnosno, potrebno je podeliti domen na manje delova i fokusirati se na deo po deo. Tako se dobijaju takozvani subdomeni (**subdomain**). Svaki od njih je potrebno **modelovati** i rešiti problem sa kojim se on susreće.

Benefiti DDD-a su:

* fleksibilan softver (lako dodavanje novih funkcionalnosti)
* jasna putanja prilikom rešavanja kompleksnog problema
* čist kod
* kod lak za održavanje
* biznis logika je na jednom mestu
* korisni design pattern-i

Iako Domain Driven Design pruža navedene benefite, treba se jako pažljivo koristiti. Nije uvek najbolje rešenje koristiti DDD. Kao što i sam **Eric Evans**, začetnik Domain Driven Desgn-a kaže, DDD se treba koristiti samo kod kompleksnih problema gde se i mogu uvideti pravi benefiti DDD-a. U suprotnom, rad sa DDD-em može postati **overengineering**-razvoj proizvoda koji je bespotrebno ukomplikovan.

S obzirom da je DDD jedan jako kompleksan pristup, postoji nešto što se zove DDD Mind Map. On prikazuje koncepte DDD-a, njihovu interakciju i položaj. Mind Map-u je moguće videti na sledećoj slici:

A diagram of a company

Description automatically generated