21. REDIZAJN SOFTVERA

Redizajn softvera, tj. reinženjering softvera se može definisati kao pristup kojim se analiziraju postojeći softver i njegova primena u praksi, uz nastojanje da se unapredi efikasnost korišćenja (eksploatacije) primenom fundamentalnih i radikalnih pristupa izmenama ili eliminaciji funkcija i opcija, njihovim unapređivanjem, te poboljšanjem strukture i performansi.

Softverski reinženjering se definiše kao „ispitivanje i zamena postojećeg sistema novim sistemom koji ga rekonstituiše (drugačije je osmišljen i drugačje implementiran u odnosu na prvobitni sistem).

Potreban je sistematski pristup analizi i redizajnu softverskih funkcija, tokova podataka i strukture organizacije, kako bi se unapredio kvalitet usluga i/ili proizvoda, uz smanjenje troškova i vremena rada.

Softverski reinženjering je blisko povezan sa sledećim pojmovima:

- "Forward Engineering" – tradicionalni proces koji teče od visokog nivoa apstrakcije i logičkog, implementaciono-nezavisnog dizajna ka fizičkoj implementaciji sistema. Obično se iz modela generiše (primenom CASE alata) i kasnije modifikuje, preciznije implementira programski kod.
- "Reverse Engineering" (Reverzni inženjering) – proces analize nekog softvera i identifikacija komponenti i njihovih međusobnih veza i kreiranje prezentacije sistema u drugoj formi ili na višem nivou apstrakcije. Obično se reverznim inženjeringom iz programskog koda ili baze podataka dobijaju modeli, primenom CASE alata.
- Restrukturiranje – transformacija iz jedne forme prezentacije u drugu, na istom nivou apstrakcije, uz očuvanje eksternog ponašanja (funkcija i semantike). Kod se često restrukturira kada je loše napisan i ima loše performanse, u cilju da bude bolji za održavanje. Najčešće je ovde reč o refaktorisanju programskog koda – strukturnim izmenama koje ne utiču na izgled ili ponašanje (funkcije) softvera.

Najčešće se softverski reinženjering realizuje u procesu od nekoliko koraka:

1. **reverzni inženjering** ,
2. **restrukturiranje** - modifikacija apstraktne forme nastala analizom u reverznom inženjeringu (npr. modela) u drugi oblik modela,
3. **forward inženjering** – implementacijom novog sistema koji će imati osobine u skladu sa modifikacijama i biće implementiran na drugačiji način,
4. (po potrebi) **dodatne modifikacije** unapređenjem sistema u skladu sa novim zahtevima, koji nisu podržani u prethodnom sistemu.

Tradicionalni pristup reinženjeringu softvera uključuje reverzni inženjering, izmenu (koja može uključiti restrukturiranje i-ili dodavanje novih funkcija, tj. realizacija novih zahteva koji nisu bili uključeni u prethodno rešenje) i forward inženjering (najčešće primenom novijih tehnologija implementacije).

Pristupi softverskom reinženjeringu su:

- Pristup po principu „Veliki prasak (Engl. „Big Bang") – izmena celog starog sistema novim sistemom u jednom trenutku.
- Inkrementalni (fazni) pristup – izmena starog sistema novim, deo po deo, postepeno uvođenje novog sistema zamenom pojedinih delova starog sistema, jedan po jedan.
- Evolutivni pristup – postepena zamena starog sistema novim, funkcija po funkciju (razlika u odnosu na inkrementalni pristup, jer je tamo bila orjentacija na komponente, a ne funkcije).

Razlozi za softverski reinženjering:

- Pojava novih tehnologija, zastarela tehnologija softverskog rešenja koji ne funkciniše na novom hardveru;
- Konkurencija - konkurenti imaju softver urađen na savremen način i sa boljim karakteristikama;
- Povećanje zahteva za kvalitetom - staro rešenje se ne može više popravljati;
- Izmena kadrovske strukture (npr. odlazak kadrova koji su poznavali prethodnu tehnologiju, nedostatak kadrova koji poznaju tu tehnologiju, ali znaju noviju);
- Nemogućnost održavanja postojećeg rešenja (iz kadrovskih razloga - nedovoljno kadrova, preveliki troškovi održavanja prethodnog sistema).

Postoje mnogi **aspekti koje treba uzeti u obzir u dizajnu/redizajnu softvera** i njegovih delova. Značaj svakog treba da odražava ciljeve koje softver pokušava da postigne. Neki od ovih aspekata su:

- **Kompatibilnost** - softver je **u stanju da radi sa drugim proizvodima** ili može biti kompatibilan sa **starijom verzijom istog** softvera.
- **Rastegljivost - nove mogućnosti se mogu dodati softveru** bez značajnih promena u osnovnoj arhitekturi.
- **Greška tolerancije** - softver je otporan i s**posoban da se oporavi od neuspeha izvršavanja komponenti.**
- **Sposobnost snabdevanja** - mera **koliko lako se ispravke grešaka** ili funkcionalnih modifikacija može postići. Visoka mogućnost održavanja može biti proizvod modularnosti i proširenja.
- **Modularnost** - dobijeni softver obuhvata dobru definisanost, ima adekvatan stepen kohezije/spregnustosti i nezavisne komponente koje dovode do boljeg održavanja. Komponente bi mogle da budu realizovane i testirane u izolaciji pre nego što budu integrisane da se formira željeni sistem softvera. Ovo omogućava podelu rada u razvoju softverskog projekta.
- **Pouzdanost** - softver je u **stanju da obavi potrebnu funkciju** pod navedenim uslovima za određeni vremenski period.
- **Upotrebljivosti** - delovi ili ceo softver **može da se koristi i u drugim projektima bez ili samo sa neznatnim, malim modifikacijama.**
- **Nedostatak tolerancije** - softver je **u stanju da radi pod stresom ili toleriše nepredvidljiv ili nepravilan unos**. Treba da ima otpornost na uslove rada niske memorije.
- **Bezbednost** – softver je **otporan na pokušaje prijave neregistrovanih korisnika**, preuzimanja poverljivih podataka i sl.
- **Upotrebljivost** - **korisnički interfejs softvera mora biti koristan i dopadljiv** za osnovni cilj korisnika.
- **Performanse** - softver **obavlja svoje zadatke u okviru korisnički prihvatljivog vremena**. Softver ne troši previše memorije.
- **Prenosivost** - **upotrebljivost istog softvera u različitim okruženjima** (OS).
- **Skalabilnost** - softver se **dobro prilagođava povećanju podatka ili broja korisnika.**