Lekcija 6 - Principi dizajna

Dva opšta tipa problema

- Uvrnuti (*wicked*) – osnovne karakteristike su:
  - zahtevi su poznati tek nakon što se završi implementacija, tj. problem je "definisan" samim rešenjem
  - ne postoji izlazni kriterijum, tj. ne zna se kada treba stati u traženju rešenja
  - rešenja nisu tačna/netačna već manje/više korisna (dobra)
  - svaki problem je priča za sebe
  - name jasnih limitiranih alternativnih ruta
  - dva pristupa rešavanju (linearna i oportunistička) na slici su superponirana jedna na drugoj:

- Pitomi (*tame*) – tipični problemi s kojima se računarska nauka bavi. Izazovni ali dobro definisani problemi. Na primer, razni algoritmi, strukture podataka, protokoli, itd.

Elektroenergetski softverski inženjering – Razvoj EE softvera - 2016

2

Većina problema s kojima se bavi softversko inženjerstvo je uvrnuto (po ovoj šemi). Razlog tome je složenost problema, koji se rešava. Ta složenost je, s druge strane, uzrokovana ne tehničkom već više domenskom/sociološkom komponentom. Softverski sistemi se mogu klasifikovati u 3 različite vrste: S, P i E (https://en.wikipedia.org/wiki/Lehman%27s_laws_of_software_evolution). EES spada definitivno u E tip. Ovde na softver utiče okruženje u veoma velikoj meri, i fluktuacija zahteva ka sistemu je konstantna. Zato je jako teško odrediti apsolutni skup zahteva, koje softver treba da zadovolji.

Pored toga, nije očigledno kako balansirati nefunkcionalne zahteve. Ova problematika je poprilično sužena u slučaju striktno teoretskih (matematičkih) modela (algoritama). Neki algoritmi čak imaju za pretpostavku da je memorija neograničena. U softverskom inženjerstvu ne samo da je ograničena, već treba brinuti i o različitim tipovima memorija i njihovim osobinama. Efikasno rukovanje memorijama (pogotovo u distribuiranim sistemima) je samo po sebi već dovoljno uvrnuto.

Većina ljudi pristupa rešavanju uvrnutih problema linearnim pristupom. Drugi pravac je tzv. oportunistički način, gde se probaju više alternativa i uvek odabira jedna, koja dovodi bliže ka rešenju (koliko god to maglovito visilo u vazduhu).

Za rešavanje uvrnutih problema moramo raspolagati sa nizom heuristika. Tu dolazi do izražaja iskustvo eksperata, koji većinu ovakvih problema “napadaju” koristeći svoje bogato iskustvo i intuiciju.

Karakteristike procesa dizajna

- Ono je komplikovano – ako se pogleda oportunistički pristup, onda izgleda kao da je nastao haos. Međutim, progres se ogledava sukcesivnim odabirom naredne povoljne solucije. Na taj način se zapravo prolazi kroz lavirint pokušaja.
- Ono je bazirano na kompromisima i prioritetima – potrebno je balansirati skup zahteva, koje će se isporučiti klijentu u datom roku. Takođe, treba balansirati nefunkcionalne atribute da bi se došlo do prihvatljivog kvaliteta.
- Ono je osnovano na heuristikama – iskusni dizajneri koriste umnogome svoju intuiciju da bi razrešili sijaset problema, za koje iskustveno znaju šta je rešenje. Ovo je slično kao i kad velemajstor na osnovu iskustva ne razmatra sve moguće poteze, nego samo one, za koje zna da u datoj poziciji ima smisla razmatrati.
- Ono evoluira tokom vremena – tu je veoma bitno ugraditi u arhitekturu saznanja o procenjenim vektorima promena.

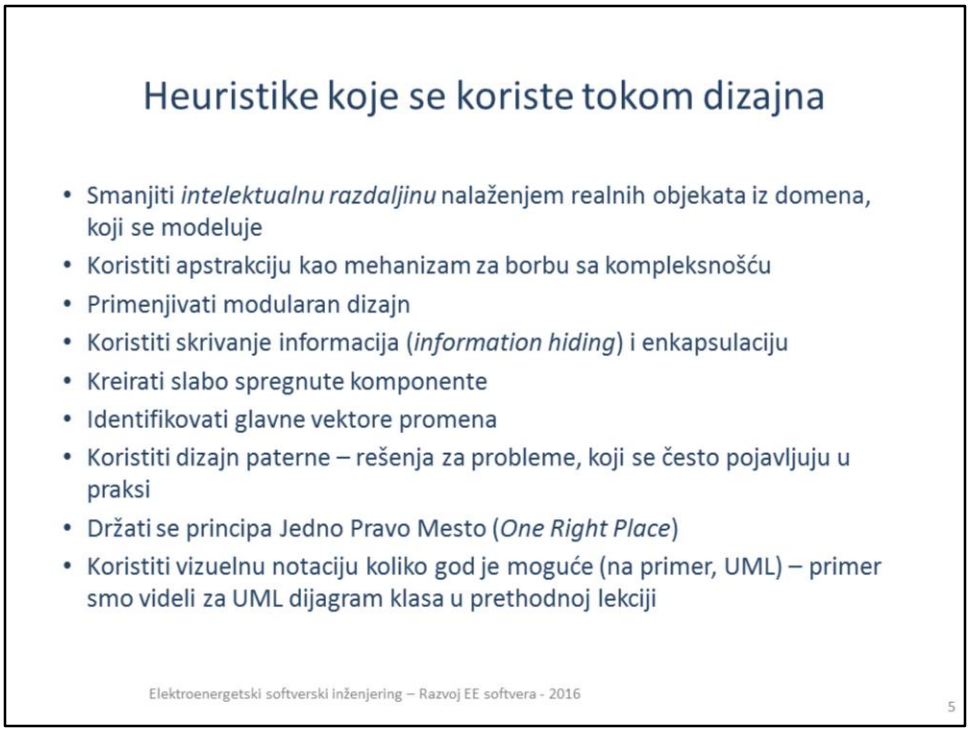
Na primeru MVC-a, možemo videti da je tu izgled (view) korisničkog interfejsa potpuno odvojen od poslovne logike (modela). Na taj način je vektor promene kontrolisan i ugrađen u samu arhitekturu.

U svojoj legendarnoj knjizi *De architectura* čuveni rimski arhitekta Vitruvius je naveo sledeće 3 najbitnije karakteristike dizajna (arhitekturalnog): čvrstoća, korisnost i lepota. Mnogi su kasnije govorili da *nešto* mora biti lepo da bi uopšte radilo, i obrnuto, *to* sigurno radi jer je lepo.

Da li treba da je dizajn lak za ponovno korišćenje (reusable)? Ovo je zanimljivo pitanje i ne može se dati jednostavan odgovor. Naime, komponenta namenjena za ponovno korišćenje mora biti dizajnirana za to. Dalje, treba da se raspolaže sa dovoljno informacija o domenu radi procene koje komponente ima smisla generalizovati, i kako to uraditi.

Jedna vrsta ponovnog korišćenja dizajna su paterni. Druga forma su softverski okviri (frameworks).

Održiv dizajn je bitan za vršenja izmena na dizajnu. One će uslediti ne samo zbog primećenih grešaka (pa treba korekcija), novih/izmenjenih zahteva, već i zbog dubljeg razumevanja postojećih zahteva od strane samog dizajnera. Sve u svemu, proces dizajna je iterativan pa su promene na njemu potpuno prirodne pojave.



Princip *One Right Place* je donekle sličan *Single Point of Control*-u, mada nije isti. Kod ovog drugog naglasak je na akcijama, koje se mogu izvršiti u sistemu za kontrolu njegovog ponašanja, dok je prvi vezan za samu realizaciju sistema (svaka funkcija je implementirana na veoma uskom opsegu i nije raštrkana svuda po sistemu).

U svakom slučaju dizajniranje je kreativan proces i ne može se automatizovati. Veliki uticaj na krajnji rezultat ima i personalni stil samog dizajnera.

Neke osobine iskusnih dizajnera

- Bogat asortiman dizajn paterna
- Veština vladanja razvojnim alatima
- Afinitet ka jednostavnosti
- Sposobnost sagledavanja stvari iz ugla klijenta i iz ugla programera
- Mogućnost predviđanja glavnih vektora promena
- Dobre komunikacione sposobnosti
- Iskustvo u praksi i na bazi neuspešnih projekata (iz ovih neuspeha se najviše nauči)
- Veštinu rukovanja kompleksnošću (dobar moć apstrakcije)