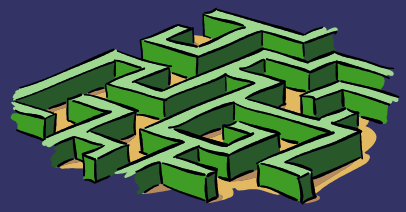
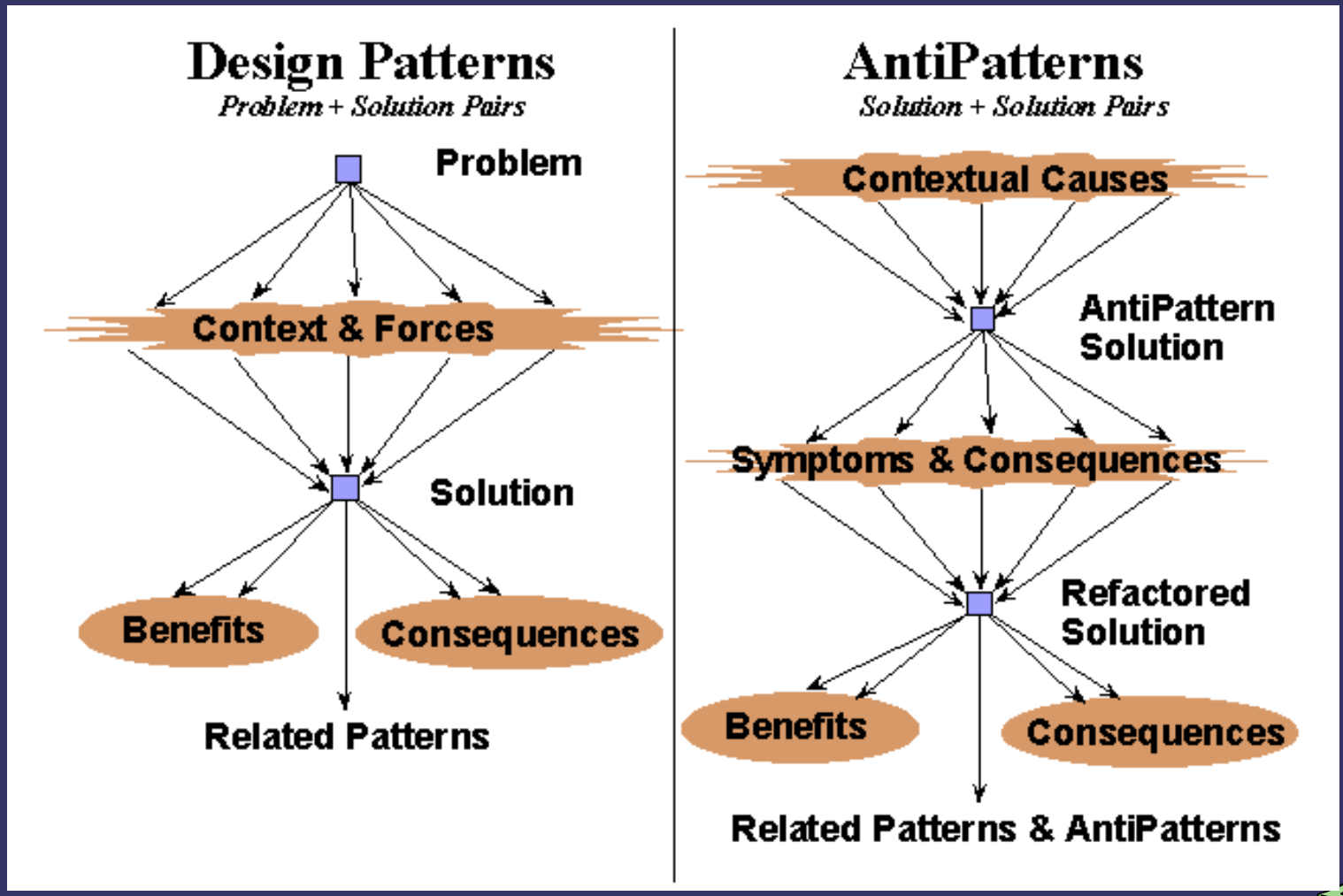
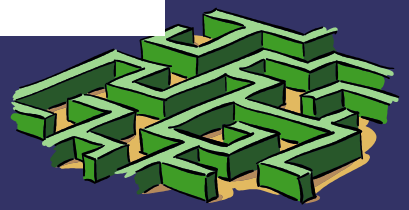
Antipaterni – referentni model

Profesor: Vladimir Filipović
Student: Milomir Radojević



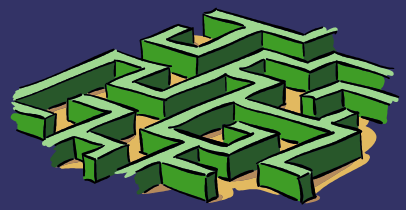
Uvod


Design Patterns
Problem + Solution Pairs
Problem
Context & Forces
Solution
Benefits
Consequences
Related Patterns
AntiPatterns
Solution + Solution Pairs
Contextual Causes
AntiPattern Solution
Symptoms & Consequences
Refactored Solution
Benefits
Consequences
Related Patterns & AntiPatterns



Uvod

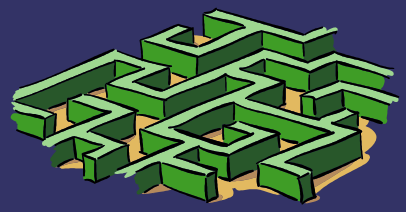
- Patern može da postane Antipatern
- Potrebno je refaktorisanje
- I Patern i Antipatern se piše počevši od rešenja
- Refaktorisano rešenje nije jedinstveno
- Patern pretpostavlja programiranje iz početka, dok Antipatern započinje postojećim (lošim) rešenjem



Osnovni koncepti

Referentni model je baziran na tri teme:

- Uzroci (greške koje stvaraju probleme)
- Primal Forces (faktori pri donošenju odluka)
- Softverski model nivoa dizajna (SDLM - Software Design−Level Model)

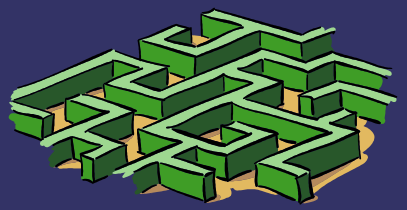


Uzroci

- Žurba (Haste)
- Apatija
- Ograničenost (Narrow−Mindedness)
- Lenjost
- Pohlepa
- Neznanje (Ignorance)
- Ponos

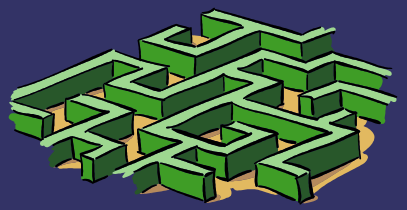
Žurba (Haste)

- Kompromisi u kvalitetu
- Prihvatanje svega što izgleda kao da radi u slučaju probijanja rokova
- Žrtvovanje testiranja



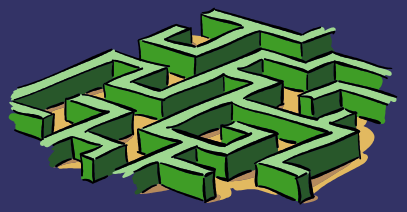
Apatija

- Nebriga o rešavanju poznatih problema
- U OO arhitekturi vodi ka problemu u particionisanju
- Stabilni deo sistema ostaje uvek u sistemu dok se individualne komponente menjaju
- Zanemarinje ovog particionisanja znači da promene u podsistemima utiču na ceo sistem
- Apatija vodi ka slaboj podršci za izmene



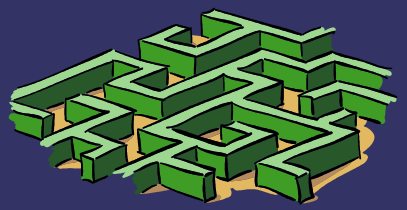
Ograničenost (Narrow−Mindedness)

- Odbijanje usvajanja koncepata koji su prepoznati kao dobri
- Metapodaci



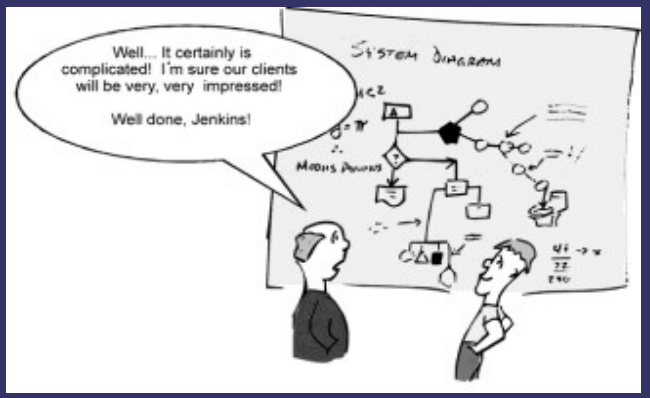
Lenjost

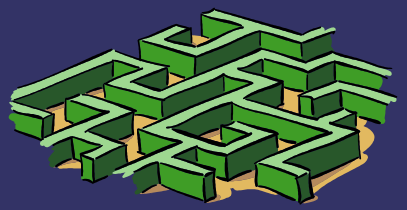
- Donošenje loših odluka zbog lakih rešenja
- Česta izmena interfejsa dovodi do greha lenjosti – manjka kontrole
- Posledica je da se u razvoju i održavanju softvera veliki deo potroši na razumevanje kako sistem radi



Pohlepa

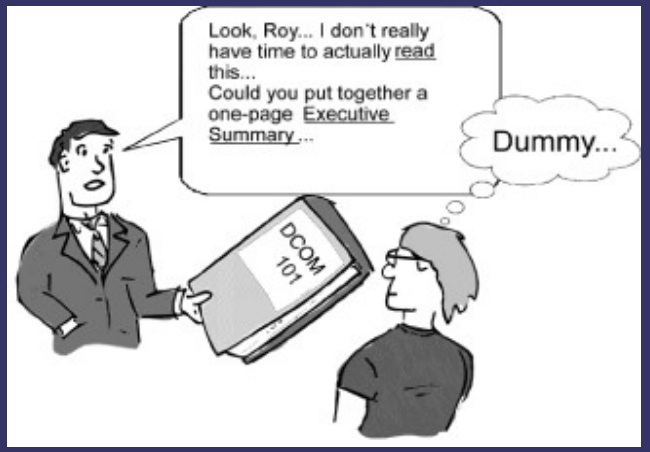
- Arhitekturna pohlepa znači modelovanje sa previše detalja
- Posledica je prevelika složenost
- Povećani troškovi razvoja, testiranja, održavanja...
- Može da dovede to propasti projekta

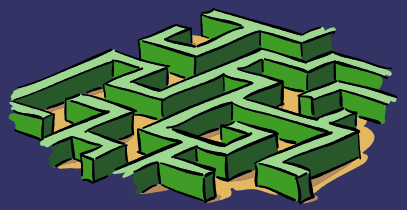




Neznanje (Ignorance)

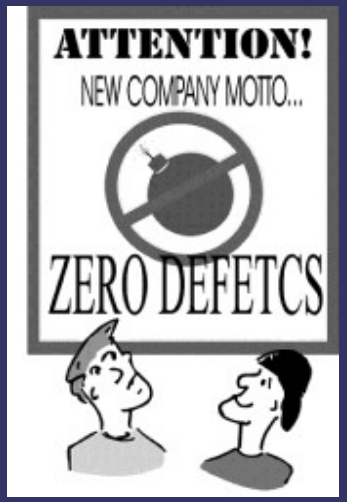
- Dešava se kada ne postoji želja za razumevanjem
- Stvara dugotrajne softverske probleme

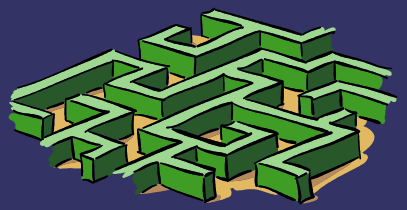




Ponos

- “Not invented here” sindrom
- Nepotrebno se razvijaju novi dizajn i proizvodi koji već postoje ili mogu da se nabave
- Ponovno izmišljanje već postojećeg softvera dovodi do neprotrebnih troškova i rizika

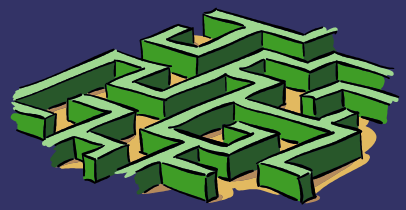




Primal Forces

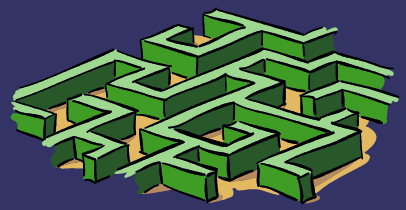
Ključni izbori tokom dizajna softverske arhitekture:

- Koje detalje otkriti a koje apstrahovati
- Koja svojstva uključuti a koja isključiti
- Koje aspekte učiniti fleksibilnim i proširivim
- Koja svojstva garantovati



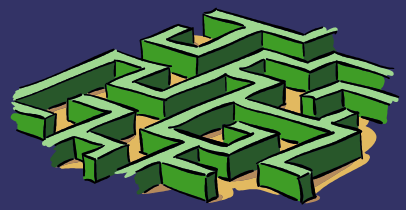
Primal Forces

- Odluke u dizajnu softvera su često složene
- Potrebno je razjasniti kontekst u kom se donose odluke
- Načini razjašnjavanja:
  - Razdvajanja faktora u odnosu na nivoe
  - Uspostavljanje prioriteta
- Stalno prisustvo rizika:
  - Svaki treći projekat bude otkazan
  - Pet od šest projekata se smatra neuspešnim
  - Nove tehnologije ne poboljšavaju statistiku



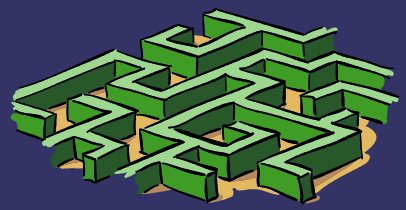
Primal Forces

- Forces su faktori koji se razmatraju pri donošenju odluka
- Za svaki softverski problem, dosta faktora se razmatra
- Izborom nekog rešenja neki od faktora su više razmotreni a neki manje
- Primal Forces su osnovna razmatranja koja je potrebno obaviti u cilju ostvarivanja uspešne arhitekture i razvoja



Primal Forces

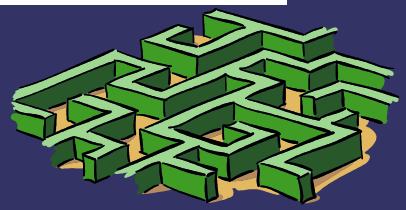
- Upravljanje funkcionalnošću (ispunjavanje zahteva)
- Upravljanje performansama (obezbeđivanje određene brzine izvršavanja operacija)
- Upravljanje složenošću (definisanje nivoa apstrakcije)
- Upravljanje promenama (kontrola evolucije softvera)
- Upravljanje IT resursima (kontrola ljudi, hardvera, softvera)
- Upravljanje transferom tehnologija (kontrola promena u tehnologiji)



Primal Forces

- Važnost faktora u odnosu na nivo modela

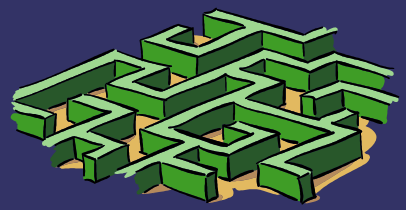
	Global Industry	Enterprise	System	Application
Management of Functionality	unimportant	marginal	important	critical
Management of Performance	important	important	critical	critical
Management of Complexity	important	critical	important	marginal
Management of Change	unimportant	critical	critical	important
Management of IT Resources	unimportant	critical	important	marginal
Management of Technology Transfer	critical	important	important	marginal



Primal Forces

- Upravljanje funkcionalnošću:
  - Ispunjavanje zahteva korisnika
  - Funkcionalnost se ostvaruje prevođenjem objekata iz sveta korisnika u svet računarskih tehnologija

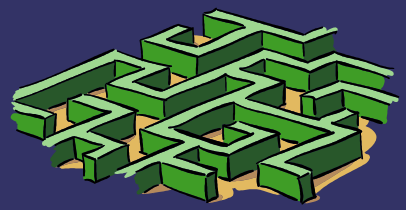
- Upravljanje performansama:
  - Ponekad zanemaren faktor
  - Osim funkcionalnost, sistem mora da ispuni potrebe korisnika vezane za performanse



Primal Forces

- Upravljanje složenošću:
  - Potrebno je voditi računa o dobroj softverskoj apstrakciji
  - Apstrakcija vodi ka jednostavnijim interfejsima, jasnijoj arhitekturi...
  - Nedostatak apstrakcije povećava složenost sistema

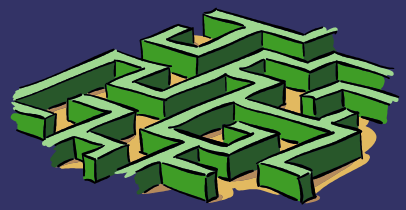
- Upravljanje promenama:
  - Prilagodljivost je često poželjna osobina
  - Sistem koji podržava promene u implementaciji je mnogo prilagodljiviji od onog koji mora da se modifikuje svaki put kada se dodaje neka nova komponenta
  - Važna je portabilnost



Primal Forces

- Upravljanje IT resursima:
  - Upravljanje resursima kompanije
  - Resursi su različite vrste hardvera, softvera, ljudstvo
  - Takođe upravljanje bezbednošću je veoma bitno

- Upravljanje transferom tehnologija:
  - Van okvira kompanije
  - Formalne i neformalne veze ostvarene korišćenjem i transferom softvera i drugih tehnologija
  - Širenje i deljenje tehnologija i znanja putem interneta
  - Mogućnost pravljenja i vršenja uticaja na standarde



SDLM

- Nivo objekata i klasa
- Nivo mikroarhitekture
- Nivo makroarhitekture
- Nivo aplikacije
- Nivo sistema
- Nivo kompanije
- Globalni nivo