Softversko inženjerstvo

Elektronski fakultet Niš

RUP metodologija

RUP *(Rational Unified Process)*

- **RUP** je skup parcijalno uređenih koraka namenjenih osnovnom cilju - da se efikasno i u predvi|enim okvirima korisniku isporuči sistem koji u potpunosti zadovoljava njegove potrebe
- **Inkrementalan** i **iterativan** proces
- Proces proizvodnje sotverskog proizvoda je **planiran** i **kontrolosan**
- Proces je **dokumentovan**
- Baziran je na **UML**-u

Elektronski fakultet u Nišu

RUP - Karakteristike

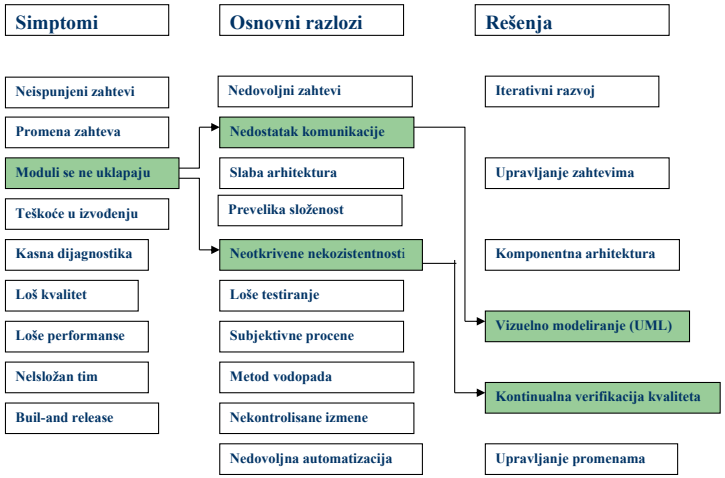
- RUP je inkrementalan i iterativan proces. To znači da se do konačne verzije sistema stiže kroz niz iteracija, a da se u svakoj iteraciji inkrementalno povećava funkcionalnost sistema sve dok se ne stigne do konačnog sistema. Na taj način, stalno se dobijaju izvršne verzije sistema sa sve većim brojem realizovanih funkcija, čime se tokom trajanja razvoja sistema, polako smanjuje rizik.
- Dobra osobina RUP metodologije je što je proces jako dobro dokumentovan (postoji i Web-tutor koji vodi korisnika kroz proces), dobro definisan - tačno je definisano šta se od proizvoda (modeli i dokumenti) u kojoj fazi dobija i u potpunosti podržan softverskim alatima (pre svega kompanija *Rational (sada IBM)* sa svojim softverskim alatima) i šablonima (templejtima) proizvoda. Sve ovo jako pomaže korisniku da dođe do konačnog cilja - **što boljeg softverskog proizvoda.**

Elektronski fakultet u Nišu

Osnovni problemi u razvoju SW-a

- Potrebe korisnika ili poslovnog sistema nisu zadovoljene
- Promena zahteva
- Moduli nisu integrisani
- Teškoće u realizaciji
- Kasno otkrivanje grešaka
- Loš kvalitet ili neiskustvo korisnika
- Loše performanse pri opterećenju
- Nekoordinisan rad tima

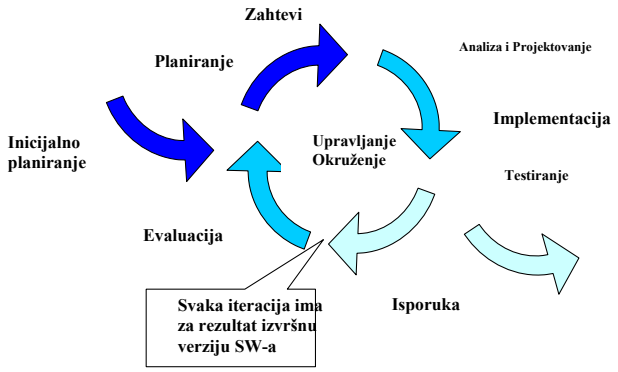
Simptomi, razlozi i rešenja



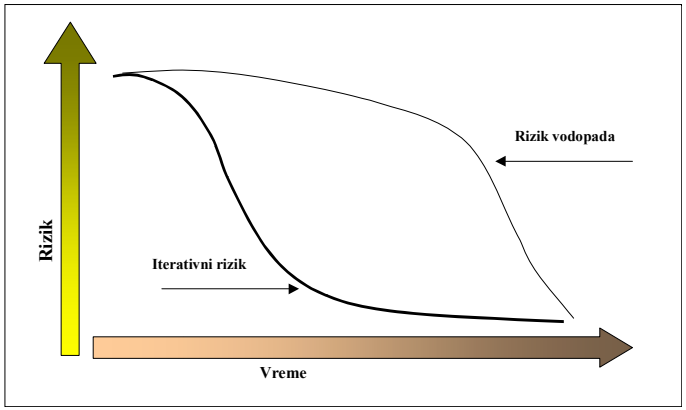
6 osnovnih principa za uspešan razvoj SW-a po RUP-u

- Iterativni razvoj SW-a
- Upravljanje zahtevima
- Komponentna arhitektura SW-a
- Vizuelno modeliranje
- Kontinualna verifikacija kvaliteta
- Upravljanje promenama

Iterativni razvoj



Smanjenje rizika



Upravljanje zahtevima

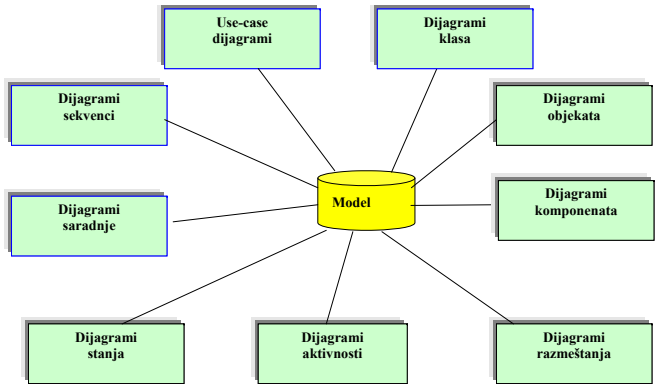
- Upravljanje zahtevima znači prevođenje zahteva korisnika u skup njihovih potreba i funkcija sistema.
- Ovaj skup se kasnije pretvara u detaljnu specifikaciju funkcionalnih i nefunkcionalnih zahteva.
- Detaljna specifikacija se prevodi u test procedure, projekat i korisničku dokumentaciju.
- Potrebno je definisati proceduru u slučaju promene zahteva korisnika.

Komponentna arhitektura

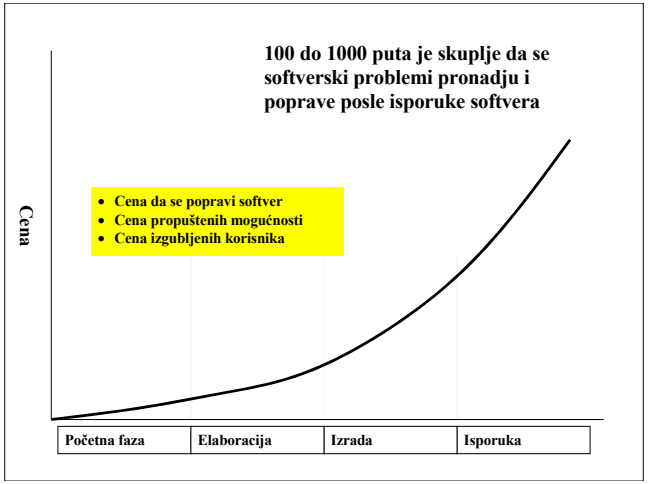
- Zadovoljava i trenutne i buduće zahteve
- Poboljšava proširljivost
- Obezbedjuje višestruko korišćenje
- Enkapsulira zavisnosti

Vizuelno modeliranje

- Koristi se UML (Unified Modeling Language) – Objedinjeni jezik za modeliranje



Kontinualna verifikacija kvaliteta



Upravljanje promenama

- Upravljanje promenama zahteva (*CRM-Change Request Management*)
- Izveštavanje o statusu proizvoda
- Upravljanje konfiguracijom (*CM-Configuration Management*)
- Praćenje promena
- Odlaganje izvornog koda i kontrola verzija

RUP metodologija

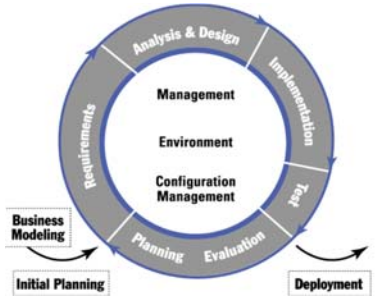
- RUP je metodologija za razvoj SW-a.
- RUP definiše korake koji dovode do proizvoda i ko je za njih odgovoran.
- Pomaže da se kontroliše projekat i da se smanji konfuzija.
- Pomaže rukovodstvu projekta u obezbeđenju resursa, planiranju i merenju napretka.
- Smanjuje rizik.
- Čini razvoj softvera predvidivim, ponovljivim i merljivim.

Osnovni koncepti RUP metodologije

- Faze, Iteracije → **Kada se nešto događa?**
- Tokovi procesa
  - Aktivnosti, koraci → **Šta se događa?**
- Proizvodi
  - modeli
  - izveštaji, dokumenti → **Šta se proizvodi?**
- Ušesnici
  - Projektant, programer → **Ko to radi?**

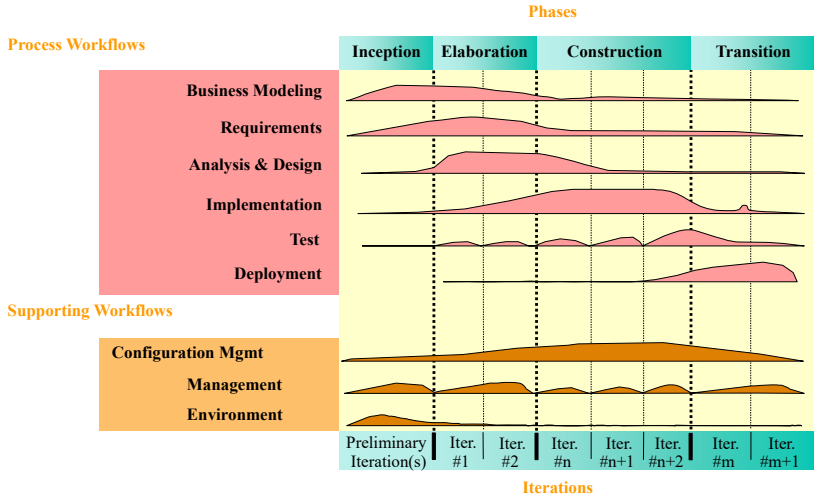
RUP faze

1. **Početna faza (*Inception*)**
2. **Faza razrade (*Elaboration*)**
3. **Faza izrade (*Construction*)**
4. **Faza isporuke (*Transition*)**

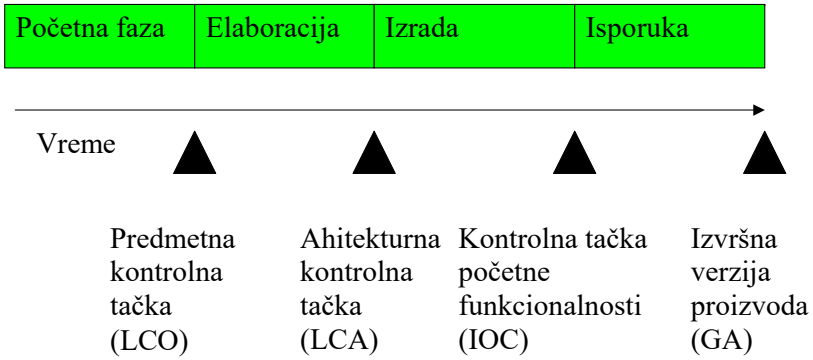


Svaka faza može imati proizvoljan broj iteracija i svaka iteracija (osim, naravno, početne) treba da rezultira izvršnom verzijom koja se može testirati.

RUP proces



Kontrolne tačke na kraju faza

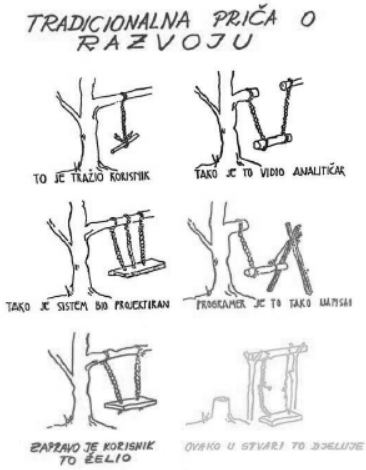


Početna faza

- Analiza problema
- Razumevanje potreba (potencijanih) korisnika
- Generalno definisanje sistema
- Upravljanje kod promena korisničkih zahteva

Rezultat ove faze je dokument ***Vizija sistema***.

Analiza problema i razumevanje potreba korisnika



Vizija sistema

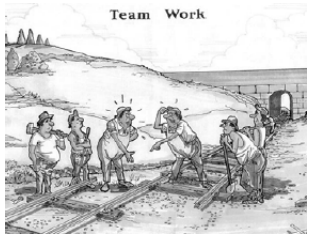
- Piše se bez mnogo tehničkih detalja tako da bude razumljiva i korisnicima i razvojnom timu.
- Koriste se samo blok dijagrami za šematski prikaz sistema.

Vizija sistema

- Pozicioniranje proizvoda
- Opis korisnika
- Opis proizvoda
- Funkcionalni zahtevi
- Nefunkcionalni zahtevi
- Ograničenja
- Kvalitet

Faza elaboracije

- Izrada plana projekta
- Organizacija i ekipni rad
- Detaljna definicija zahteva
- Definisanje arhitekture sistema

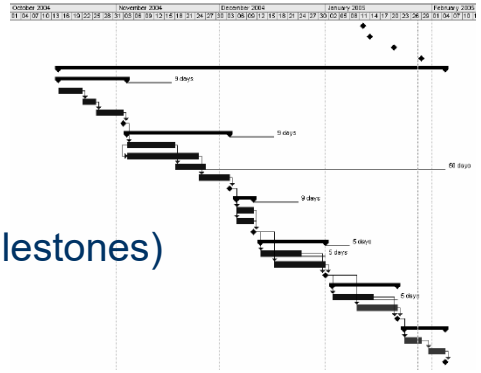


Rezultati ove faze su:

Plan projekta
Use-case specifikacija
Arhitekturni projekat sistema

Plan projekta

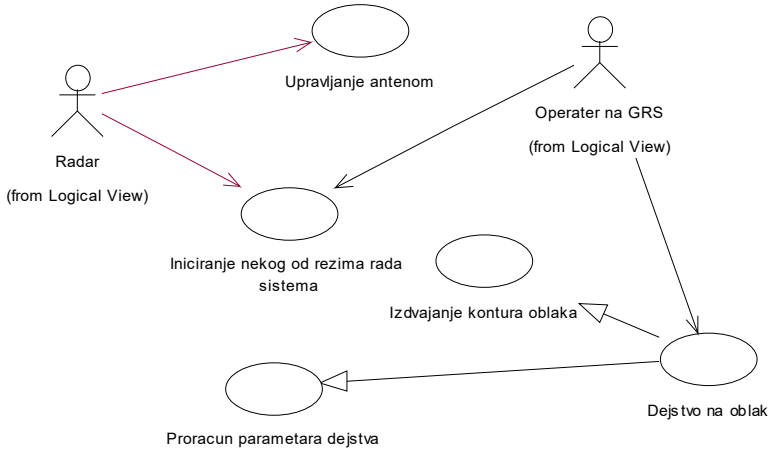
- Plan faza
- Plan izrade
- Rezultati projekta
- Kontrolne tačke (milestones)
- Resursi
- ...



Use-case specifikacija

- Opis slučajeva korišćenja
- Definisanje aktera sistemu
- Određivanje arhitekturno najznačajnijih slučajeva korišćenja

Slučajevi korišćenja



Opis slučajeva korišćenja

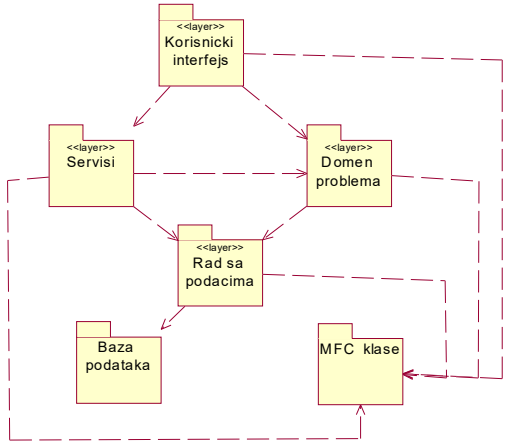
Naziv	Proračun elevacija za dejstvo
Akteri	Operater PRS1
Svrha algoritma	Proračun elevacija za lansiranje raketa
Opis	Nakon unosa novog sinoptičkog biltena u sistem potrebno je startovati izračunavanje novih elevacija za lansiranje raketa. Izračunavanje se vrši samo jednom nakon svake promene sinoptičkog biltena.

R. br. doga|aja	Akcija aktera	Reakcija sistema
1.	Ovaj slučaj korišćenja inicira operater PRS1 tako što izda komandu za izračunavanje elevacija.	
2.		Sistem daje informaciju o dužini trajanja proračuna a zatim vrši izračunavanje i na kraju izdaje poruku o završetku izračunavanja.

Arhitekturni projekat sistema

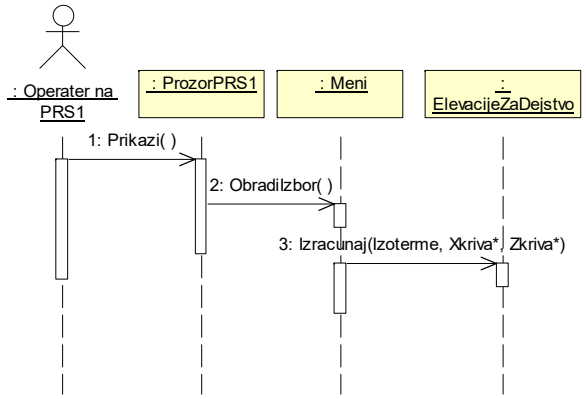
- Definisanje arhitekture sistema
- Definisanje najbitnijih klasa
- Realizacija arhitekturno najznačajnijih slučajeva korišćenja
- UML dijagrami klasa

Arhitektura sistema



Realizacija slučajeva korišćenja

- Dijagrami sekvenci



Faza izrade

- Realizacija sistema
- Testiranje

Rezultati ove faze su:

Plan testiranja
Test specifikacija
Detaljni projekat sistema
Softverski proizvod

Plan testiranja

- Zahtevi testiranja
- Strategija testiranja
- Tehnike testiranja
- Alati za testiranje
- Resursi
- Proizvodi
- Kontrolne tačke
- ...

Test specifikacija

- Test-case-ovi

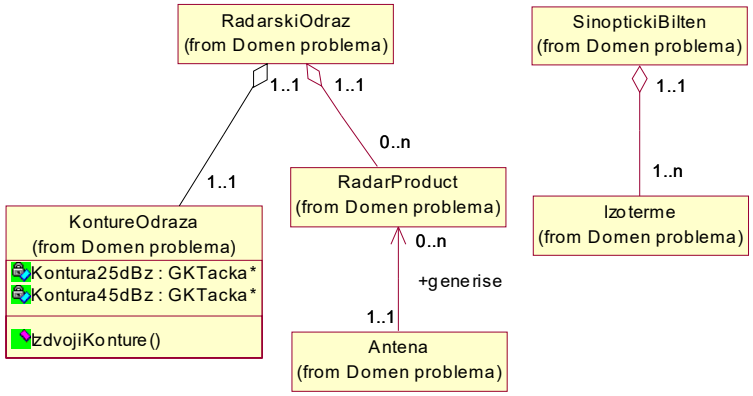
Test case:

- Opis
- Radnje-ulazi
- Očekivani odzivi-izlazi
- Završne radnje

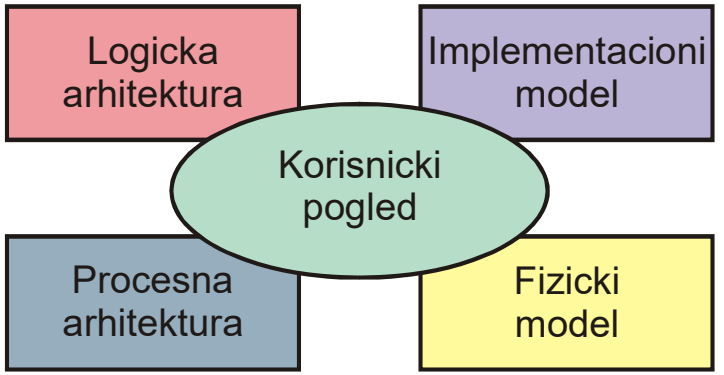
Detaljni projekat sistema

- Arhitekturni projekat razvijen u detalje
- Dijagrami klasa
- “4+1” model sistema

Dijagrami klasa



“4+1” Model sistema



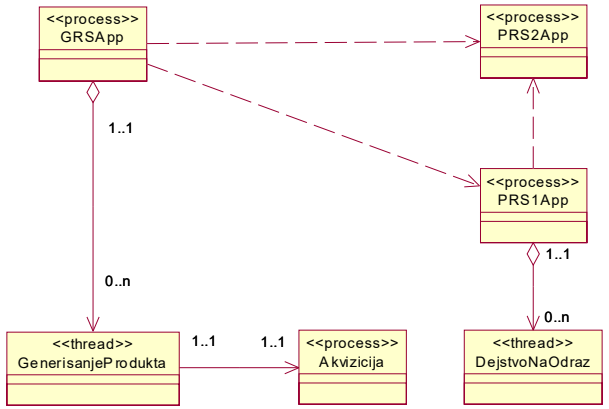
Logička arhitektura

- Logička arhitektura sistema opisuje najvažnije klase u sistemu, njihovu organizaciju u pakete i podsisteme kao i organizaciju paketa i podsistema u nivoe (*layers*)
- Za predstavljanje logičke arhitekture se koriste **dijagrami klasa**
- Mogućnost automatskog generisanja koda na osnovu dijagrama klasa

Procesna arhitektura

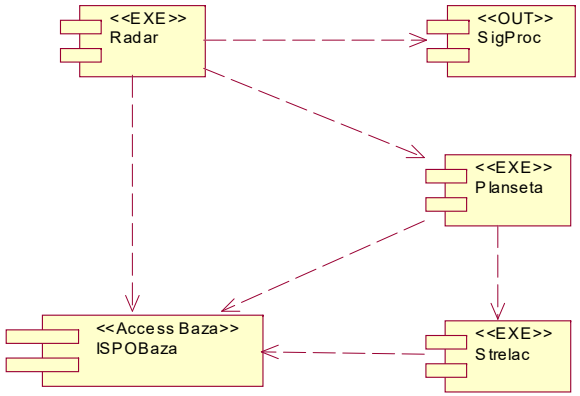
- Procesna arhitektura sistema opisuje najvažnije procese i niti (*thread*s) u sistemu i njihovu organizaciju. Procesi se izvršavaju u nezavisnim adresnim prostorima računara, dok su niti procesi koji se izvršavaju paralelno sa procesima ili drugim nitima ali u adresnom prostoru nekog od procesa.
- Za procesne arhitekture se koriste **dijagrami klasa**

Procesna arhitektura



Implementacioni model

- Za prikaz implementacionog modela se koriste **dijagrami komponenti**

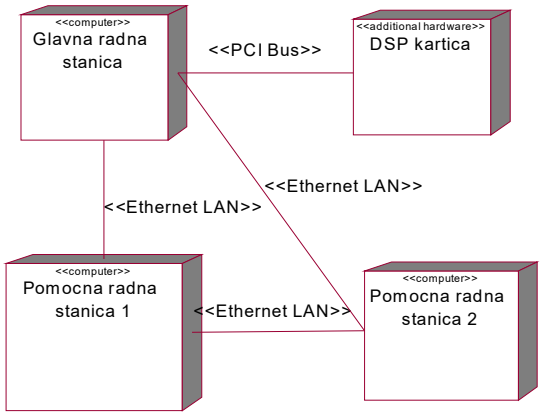


Fizički model

- Fizički model opisuje fizičke čvorove u sistemu i njihov razmeštaj u prostoru
- Za prikaz fizičkog modela se koriste **dijagrami razmeštaja**

Elektronski fakultet u Nišu

Fizički model



Elektronski fakultet u Nišu

Faza isporuke

- Finalizacija softverskog sistema
- Alfa (beta) testiranje
- Izrada korisničke dokumentacije (uputstva)
- Obuka korisnika
- Uvođenje sistema kod korisnika

Elektronski fakultet u Nišu

Faza isporuke

Rezultati ove faze su:

Test izveštaji
Korisničko uputstvo
Instalacija sistema

Test izveštaj

- Pregled rezultata testiranja
  - Generalna procena testiranog SW-a
  - Uticaj test okruženja
  - Predložena poboljšanja
- Rezultati izvršenja test-case-ova

Test log

ИЗВЕШТАЈ ТЕСТИРАЊА

Пројекат	ИОРП–1	Одобрио:	Дејан Ранчић
Тип Теста	Тестер		Датум
Black Box тестирање	Дачић Душан		01.04.2003

Ознака теста	Прошао (ДА/НЕ)	Тежина грешке	Коментар	Оцена лакоће коришћења
001/01	ДА			1
001/02	ДА			1
001/03	ДА			1
001/04	ДА			1
001/05	ДА			1
001/06	ДА			2
001/07	ДА			2
001/08	ДА			3
001/09	ДА			3
001/10	ДА			3
001/11	ДА			2
001/12	ДА			2
001/13	ДА			2
001/14	ДА			2
001/15	ДА			2

Kriterijumi testiranja

001/36	ДА			2
001/37	ДА			2
001/38	ДА			2
001/39	ДА			2
001/40	ДА			1
001/41	ДА			1

Оцене тежине грешке:	Оцене лакоће коришћења:
1 – Пад програма 2 – Неправилан рад програма 3 – Неслагање са спецификацијом 4 – Неодговарајући интерфејс	1 – Лако 2 – Интуитивно 3 – Уз стриктно поштовање упутства

© 2003 Електронски факултет, Ниш

Primena RUP alata i templejta?

Primena alata?

Za male projekte se ne moraju primenjivati, ali za bilo koji ozbiljniji rad moraju se koristiti alati (RationalRose, RationalSoDA i sl).

RUP je univerzalno rešenje?

Svaki projekat ima svoje specifičnosti. RUP pruža okvir za proces razvoja. Neki koraci mogu biti nepotrebni, neki su možda nedovoljno definisani. U zavisnosti od projekta, RUP se može prilagođavati sopstvenim potrebama.

RUP templejti?

Način da se formalizuje proces. U suštini, sam format templejta je najmanje važan, važniji je sadržaj dokumenata. Da bi se članovima razvojnog tima olakšao posao, kao i da bi se obezbedile sve neophodne informacije za naredne faze moraju se koristiti templejti. Naravno, i oni mogu biti podložni promenama u skladu sa potrebama.