**Što je validacija a što verifikacija?**

Validacija odgovara na pitanje gradimo li ispravan sustav, tj. zadovoljava li sustav funkcijske zahtjeve. Validacija se provodi ispitivanjem sustava.

Verifikacija odgovara na pitanje gradimo li sustav ispravno, tj. zadovoljava li sustav zahtjeve na ispravan način. Verifikacija uključuje provjeru poželjno zasnoavnu na formalnim matematičkim logičkim metodama.

**Tehnike verifikacije programa**

Statička verifikacija – nadzor izvornog koda, analizatori programa, sintaksne i semantičke greške

Dinamička verifikacija – pokretanje programa s realnim podacima i traženje grešaka

Formalna verifikacija

**Vrste ispitavanja**

Ispitivanje komponenti

Integracijsko ispitivanje

Ispitivanje sustava – ispitivanje prihvatljivosti, ispitivanje performansi, ispitivanje instalacije

**Kvar**

Proglašeni uzročnik problema, unosi se pri oblikovanju ili programiranju

**Pogreška**

Manifestacija kvara, uzrokuje zatajenje

**Zatajenje**

Neispravno ponašanje sustava s obzirom na zahtjeve, problem vidljiv izvan sustava

**Paretov princip**

Mali broj pogrešaka dovodi do velikog broja zatajenja (20/80)

**Regresijsko ispitivanje**

Ponovljeno ispitvanje nakon promjene ili popravka koda s ciljem potvrđivanja ispravnosti promjena i nepostanje negativnog utjecaja na nepromijenjene dijelove programa

**V model ispitivanja**

Ispitivanje se provodi nakon implementacije

**V model s ranom pripremom**

Priprema ispitivanja paralelno s razvojem

**W model**

Ispitivanje paralelno s razvojem aplikacije

**Ispitni slučaj**

Uređeni par (I,O) ispitnih podataka i očekivanog izlaza koji je zabilježen prije provođenja ispitivanja. Uspoređujemo stvarni izlaz s očekivanim pomoću kriterija prolaza ispitnog slučaja. Primjer glasovanje <18, >18

**Potpuno ispitivanje**

Ispitivanje svih mogućih vrijednosti varijabli, svih mogućih kombinacija ulaza, svih mogućih sekvenci izvođenja programa, svih mogućih HW/SW konfiguracija, svih mogućih naćina uporabe programa

**Ispitivanje komponenti**

Ispitivanje koda na pogreške u algoritmima, podacima i sintaksi.

Provodi se u kontekstu specifikacije zahtjeva

Verificira rad programskih dijelova koje je moguće neovisno ispititati, pojedinačne funkcije ili metode unutar objekta, klase objekata s više atributa i metoda, složene komponente s definiranim sučeljem

Postoji pristup programskom kodu – white box

**Slučajno ispitivanje**

Identificirati operacije primjenjive na razred

Definirati ograničenja na njihovo korištenje

Identificirati minimalni ispitni slučaj

Generirati niz ispravnih slučajnih ispitnih sekvenci

**Tipovi sučelja programskih komponenti**

Parametarsko – podaci i funkcije se prenose pozivima procedure

Dijeljena memorija

Preceduralno sučelje – komponente obuhvaćaju skup procedura koje pozivaju ostali podsustavi

Sučelje zasnovano na porukama – podsustavi traže usluge od ostalih podsustava slanjem poruke (klijent poslužitelj)

**Integracijsko ispitivanje**

Proces verifikacije interakcije programskih komponenti.

Cilj osigurati zajednički rad grupe komponenti prema specifikaciji zahtjeva

Osnovni problem: lokalizacija pogrešaka zbog složenih interakcija

**Pristupi integracijskom ispitivanju**

**Veliki prasak**

Integrirati sve komponente bez prethodnog ispitivanja, problem otkrivanja mjesta pogreške

**Poboljšani veliki prasak**

Integracija svih komponenti nakon provedenog ispitivanja, idalje prisutan problem

**Inkrementalni pristup**

Integracija i ispitivanje sustava dio po dio, uobičajen pristup u praksi, efikasan u lokalizacija mjesta pogreške

**Top down – Inkrementalno integracijsko ispitivanje**

Razviti kostur sustava i popuniti ga komponentama, ne treba razvijati upravljačke programe

**Bottom up**

Prvo integrirati neke komponente (najvažnije i najčešće funkcionalnosti) te dodati preostale, ne treba razvijati prividne komponente

**Funkcijska integracija**

Integriranje komponente u konzistentne funkcije bez obzira na hijerarhijsku strukturu, kombinacija, najčešće

**Ispitivanje sustava**

Proces ispitivanja inačice namijenjene distribuciji korisniku

Osnovni cilj provjera podudarnosti s funkcijskim zahtjevima

**Strategije ispitivanja sustava**

**Iscrpno ispitivanje**

Svih mogućih scenarija, moguće samo za ograničene primjere

**Slučajno ispitivanje**

Odabir ispitnih slučajeva temeljen vjerojatnosti

**Sistematsko ispitivanje**

Podjela ulaznih podataka u podddomene i odabir ispitnih slučajeva temeljem različitih svojstava

**Ispitivanje pod pritiskom**

Namjena određivanje stabilnosti sustava, istjerivanje pogrešaka (robusnost, raspoloživost i obrada pogrešaka na graničnim vrijednostima opterećenja)

**Ispitivanje prihvatljivosti**

Provjerava ponašanje sustava u odnosu na zahtjeve naručitelja, provodi se zajednički s timom naručitelja

Funkcijsko ispitivanje, black box

**Instalacijsko**

Provodi se nakon ispitivanja prihvatljivosti na instalaciji u radnoj okolini

**Alfa i beta ispitivanje**

Program pokusno upotrebljava ciljana skupina korisnika

**Funkcijsko ispitivanje – black box testing**

Pretpostavlja da nema znanja programskog koda ili oblikovanja sustava, koncentracija na U/I ponašanje

Ispitivanje samo prema zahtjevima i specifikacijama

**Strukturno ispitivanje – white box testing**

Ispitivanje očekivanog ponašanja zasnovano na svojstvima programa ili oblikovanju

Cilj ispitivanje je pokrivanje izvođenja svih mogućih naredbi i uvjeta najmanje jednom

Prikaz programa: dijagram toka, graf tijeka programa

**Selektivno ispitivanje**

Ispitivanje temeljnih putova, uvjeta, petlji, protoka podataka

**Ispitivanje temeljnih putova**

Skup putova koji minimalno jednom pokrivaju izvođenje svih naredbi i uvjeta

**Ciklomatska složenost**

Broj neovisnih putova u temeljnom skupu

CV(G) = Lukovi – Čvorovi + 2\*P

**Pokrivanje koda**

Predstavlja pokrivenost izvornog koda programa provedenim ispitivanjima

**Automatizacija**

Testiranja koja obavljaju programi.

Brže i jeftinije, poboljšanje točnosti, stabilno ispitivanje kvalitete, automatizirano dokumentiranje prijava pogrešaka, smanjenje ljudskog rada

**Strukturni dijagrami:**

Dijagram razreda, objekata, komponenti, razmještaja, paketa

**Ponašajni dijagrami:**

Obrasci uporabe, dijagram stanja, aktivnosti, interakcije: sekvencijski, komunikacijski

**Statički**

Diagram obrazaca uporabe, dijagram razreda, objekata, komponenti, razmještaja

**Dinamički**

Sekvencijski, komunikacijski, stanja, aktivnosti

**Razlika sekvencijski – komunikacijski**

Sekvencijski opisuje vremenske odnosa između podražaja, tj. modelira sustav u stvarnom vremenu. Koristimo životne linije kojih u komunikacijskom dijagramu nema.

Komunikacijski se koristi za opis struktura interakcije i usredotočen je na učinke instanci, ne prikazuje vremenske odnose

**Komunikacijski**

Komunikacijski se koristi za opis struktura interakcije i usredotočen je na učinke instanci, ne prikazuje vremenske odnose već redoslijed poruka

Objekti, veze, poruke

Tko šalje kome

**Dijagram stanja**

Elementi: početno i konačno stanje/a, stanje, prijelaz, uvjetno grananje, račvanje i skupljanje, složeno stanje

Stanje je zaobljeni pravokutnik koji u gornjem dijelu sadrži naziv stanja, a u donjem aktivnosti koje se događaju u tom stanju

Opisuje dinamičko ponašanje jednog objekta u vremenu

Prikazuje sekvencu stanja objekta te prijelaze iz jednog stanja u drugi temeljene na događajima, poput automata s memorijom

Pogodni za modeliranje događajima poticanog ponašanja sustava.

Objekt se promatra izolirano od ostalih

Izlaz ne ovisi samo o trenutnim ulazim nego i o povijesti

Entry/exit akcije, do aktivnost

Prijelaz: [uvjet] događaj/akcija, mogu prenositi parametre

Događaj koji izaziva prijelaz: interakcija: asinkroni prijem signala, sinkroni poziv objekta

Vremenski: proteklo prijeme

Dinamičko uvjetno grananje – uvjeti se računaju pri izlaska iz stanja

Konkurentna stanja – paralelna stanja

Fork i join – u slučaju vremenski paralelnog odvijanja stanja i prijelaza, višedretvenost

**Razlika dijagram aktivnosti/komunikacije/stanja**

Način iniciranja pojedinog koraka a posebice kako koraci dobivaju ulazni signal ili podatke.

U komunikacijskom to su poruke, u dijagramu stanja to su događaji.

**Razlika dijagrama aktivnosti/stanja**

U dijagramu aktivnosti su stanja zapravo aktivnosti.

Npr. dijagram stanja za korištenje dizala bio bi:

Početak,Ulazak,stajanje,izlazak,cilj

Dijagram aktivnosti bi bio:

Ulazak u dizalo, pritisak tipke za pravi kat, izlazak iz dizala.

Na prijelazima ne navodi događaje koji su prouzročili prijelaz zato se i koristi kada ponašanje ne ovisi o velikom broju vanjskih događaja.

**Razlika dijagrama obrazaca upotrebe i dijagrama aktivnosti**

Dijagram obrazaca uporabe je statički i bazira se na akcijama dionika sustava

Dijagram aktivnosti je dinamički i bazira se na aktivnostima, ne prikazuje aktore.

**Dijagram aktivnosti**

Elementi: čvorovi: akcije, upravljački,objekti

Veze

Značke – kreću se od izvorišta prema odredištu

Koristi se kada ponašanje ne ovisi o velikom broju vanjskih događaja

Unutar aktivnosti izvodi se samo jedna aktivnost, dakle ne postoje složene aktivnosti

**Dijagram komponenti**

Prikazuje komponente sustava i njihove međusobne odnose

Komponenta je zasebna cjelina programske potopore s vlastitim sučeljem

Zajedno s dijagramom razmještaja – fizički dijagrami

Pridruživanje/veza komponenti – jedna zahtijeva uporabu druge radi potpunog ostvarenja

Jednosmjerna ovisnost „B ovisi o A“ – klijent ovisi o isporučitelju

Sučelje: kolekcija operacija određuju usluge komponenti, tj. imenovan skup javno vidljivih atributa i apstraktnih operacija

Eksportirano sučelje: koje neka komponenta realizira kao uslugu za druge komponente

Importirano: koje neka komponenta koristi

Stereotipovi: executable, library, table, file, document

**Razlike komponenta razred**

Razina apstrakcije: razred je logička apstrakcija, komponenta fizički artifakt

Funkcionalnost: razredi imaju vlastite atribute i operacije, komponente imaju pristup samo operacijama kroz definirana sučelja

**Razlika dijagram komponenata i dijagram razreda**

Komponentni dijagram pomaže u modeliranju fizičkih cjelina kao što su izvršne datoteke, programske biblioteke, tablice, datoteke itd. Dok dijagram razreda prikazuje razrede i njihove odnose te njihove atribute i metode.

**Dijagram razmještaja**

Opisuje statičke odnose s fizičkog aspekta implementacije

Prikazuje sklopovlje i programsku podršku potrebnu za implementaciju sustava u stvarnom radnom okruženju

Elementi: čvorovi – fizički, izvršavaju komponente(poslužitelj npr)

Komponente – logički aspekt, sudjeluju u izvršavanju sustava(izvršne datoteke, stolne, mobilne, mrežne aplikacije)

Veze – jednosmjerne dvosmjerne ovisnost

Stereotipovi: processor, device, execution environment

Može prikazati pojedince(podcrtani nazivi)