Introducció.

Les proves son necessàries al desenvolupament de projectes, com més aviat es detecti un error, menys costos comportarà, la fase de proves te un cost d’entre el 30% i el 50%.

Les proves en el cicle de vida d’un projecte.

A cada fase, caldrà que el treball dut a terme haurà de ser validat i verificat.

Procediments, tipus i casos de proves.

A cada fase se l’hi haurà de dedicar un temps considerable a desenvolupar les tasques. Els procediments relacionats amb les proves son un petit projecte dins del projecte principal.

Planificació de les proves.

Si l’errada triga en ser detectada, pot créixer exponencialment i comportar més costos imprevistos. Per aquest motiu es important planificar les proves, es a dir, saber quan i com dur a terme les proves.

Disseny de les proves. Tipus de proves.

Hi ha un munt de tipus de proves, proves unitàries, proves funcionals, proves d’integració, proves de sistemes i proves d’acceptació.

Proves unitàries.

Son les de nivell més baix, es duen a terme a mesura que es fa el projecte. Volen detectar errors en les dades i algorismes. Tenen un enfocament estructural o un enfocament funcional. Les més utilitzades son, caixa blanca o caixa negre.

Proves unitàries: Caixa blanca.

Se centren en l’estructura del programa, analitzant els camins d’execució. Hauríem de cercar casos per recórrer tots els camins possibles, hi ha dues tècniques, cobertura de flux de control i complexitat ciclomàtica.

Forma

Descripción generada automáticamente

Caixa blanca: cobertura de flux de control.

S’ha de dissenyar un cas de prova per a cada camí independent, també s’han de dissenyar prou casos de prova per a totes les condiciones del programa (true/false). Per últim s’han de dissenyar els casos de prova de manera que s’intenti executar un bucle en diferents situacions límit.

Caixa blanca: com dibuixar un diagrama de flux.

Agrupar instruccions en seqüencies, assenyalar sobre el codi les condiciones i assenyalar les iteracions.

Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

Caixa blanca: cobertura de flux de control.

Imagen que contiene Texto

Descripción generada automáticamente

Caixa blanca: complexitat ciclomàtica.

La complexitat ciclomàtica, es el nombre de camins independents d’un diagrama de flux. Per calcular-la es: CC = nombre de branques – nombre de nodes + 2.

En l’exemple, els nodes que intervenen son 1, 2, 3, 4, 5, 6, 7 i les branques son les línies que uneixen els nodes que en total son 8. CC = 8 – 7 + 2 = 3.

Això significa que s’hauran de dissenyar tres casos de prova. Camí 1: 1-2-3-4-5-6-2-7, Camí 2: 1-2-3-4-6-2-7, Camí 3: 1-2-7.

Proves unitàries: Caixa negra.

Proven la funcionalitat del programa, es dissenyen casos de prova que comprovin les especificacions del programa. Cal escollir els casos de prova, només els necessaris, per tal de que el temps d’execució sigui baix, buscant eficàcia. Son independents del llenguatge de programació utilitzat.

Tècniques.

* Classes d’equivalència: determinar les diferents entrades i sortides.
* Anàlisi dels valors límit: estudiar valors inicials i finals, ja que estadísticament detecten mes errors.
* Estudi d’errors típics: normalment, es repeteixen els mateixos errors en molts programes.

Classes d’equivalència.

Dissenyar els casos de forma eficient, s’ha de identificar les condicions, restriccions o continguts de les entrades i sortides, i a partir de les condicions, les classes d’equivalència de les entrades i sortides.

Recomanacions.

Les classes han de recollir tant dades valides com errònies. Si s’especifica un rang de valors s’ha de crear una classe valida i dos incorrectes (major, correcte i menor). Si s’especifica un valor vàlid i d’altres no vàlids, es crearà una classe valida i l’altra no valida. Si hi ha un conjunt de dades d’entrada concretes valides, es generarà una classe per a cada valor vàlid, i una altre per un no vàlid.

Ús d’interfícies gràfiques.

Les proves d’interfície gràfica han d’incloure, proves sobre finestres (icones de tancar, minimitzar...), proves sobre menús i us de ratolí, proves d’entrada de dades (quadre de textos, llistes desplegables...), proves de documentació i ajuda del programa.

Proves d’integració.

Impliquen una progressió ordenada, partint des dels components i acabant amb el sistema complet.

S’incrementa progressivament el nombre de mòduls fins a formar el programa complet, es podrà fer de forma ascendent (comença pels mòduls fulla) o descendent (comença pels mòduls arrel).

Es prova cada mòdul per separat, i després s’integren tots alhora i es prova el programa complet.

Proves d’integració: Integració incremental ascendent.

Es comença amb els mòduls de sota, es combinen els mòduls de baix nivell en grups que realitzen una funció especifica, buscant reduir el nombre de proves.

Per a cada grup s’escriu un mòdul impulsor que es un mòdul per simular la crida als mòduls i capturar els resultats. Es prova cada grup utilitzant el seu impulsor.

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

A la segona fase es barreja la prova d’unitat i la prova d’integraacio. La creació de mòduls impulsors normalment no es complicada.

Proves d’integració: Integració incremental descendent.

Aquesta estratègia comença al mòdul principal, i incorpora els mòduls de sota progressivament. Si hi ha seccions critiques, convé provar-los com més aviat millor. Els mòduls d’entrada/sortida s’han de provar com més aviat millor. Es pot fer en profunditat o en amplada.

El mòdul arrel es el primer, quan el mòdul arrel ja no dona cap defecte, es posa un mòdul subordinat, s’executen les proves corresponents.

La codificació de mòduls ficticis subordinats es mes complicada que la dels mòduls impulsors, poden nomes mostrar un missatge, mostrar els paràmetres que reben, que torni un valor independent del valor d’entrada, o mòduls que en funció dels paràmetres tornin un valor de sortida diferent.

Proves d’integració: Avantatges i inconvenients de les integracions descendent i ascendent.

L’ascendent es millor per trobar les errades grans a la part baixa del programa, més fàcil crear les entrades i més fàcil observar els resultats, però cal crear mòduls impulsors i el programa complet nomes apareix quan s’agrega l’últim mòdul.

La descendent es millor per trobar els errors greus dels mòduls alts de la jerarquia, mes fàcil de manipular els casos de prova, permet veure abans una estructura prèvia del programa per facilitar les demostracions. Per contrabanda cal crear mòduls ficticis, es difícil incorporar casos de prova, es mes difícil observar la sortida i pot retardar la prova de parts del programa.

Proves del sistema.

Es el procés d’un sistema integrat de SW i HW per veure si compleix els requisits especificats, compliment de tots els requisits funcionals, funcionament i rendiment de les interfícies de HW i SW, adequació de la documentació d’usuari i execució i rendiment de condicions límit i de sobrecarrega.

Els casos de prova s’obtenen de tres fonts diferents:

* Casos basats en els requisits utilitzant la tècnica de caixa negra aplicada.
* Casos necessaris per provar el rendiment del sistema i de la seva capacitat.
* Casos basats en el disseny d’alt nivell aplicant la tècnica de caixa de vidre.

Proves d’acceptació.

L’objectiu es comprovar si el producte esta preparat per ser implantat per al seu us a l’entorn de l’usuari. Les característiques principals d’aquestes proves son:

* Participació de l’usuari, normalment es l’usuari qui executa les proves.
* Esta enfocada cap a la prova dels requisits d’usuari.
* Esta considerada com la fase final del projecte.
* Els casos de prova poden procedir de casos de caixa negra, processament d’una carrega de treball típica, proves especials de fiabilitat per veure el nivell de degradació.