CICLE FORMATIU:	CFGS Desenvolupament d'aplicacions web (CFPS ICC0)
MÒDUL:	MP05. Entorns de desenvolupament
UNITAT FORMATIVA:	UF2. Optimització de programari
ACTIVITAT:	A2.Ptp2.1. Prova final d'unitat formativa
ALUMNE/A	Adria Moya

1- [1 punt] Marqueu si els següents continguts pertanyen (V) o no (F) al pla de proves.

Núm	Pregunta	V	F
1	Identificador del pla de proves	x	
2	Descripció del pla de proves	x	
3	Descripció del projecte		
4	Elements del programari que no s'han de provar	x	
5	Definició de la configuració del pla de proves	x	
6	Requisits del projecte		
7	Documents a lliurar	x	
8	Responsables i responsabilitats	x	
9	Parts interessades (o stakeholders)		
0	Calendari del pla de proves	x	
1	Calendari de la fase de desenvolupament		

2- [1 Punt] Relacioneu els tipus de proves amb les característiques que les defineixen.

Núm	Pregunta	Resposta
1	Són les encarregades de detectar els errors en la implementació dels requeriments d'usuari.	A
2	El mètode utilitzat és el de capsa blanca, el de disseny descendent (o top-down) i el de bottom-up.	D
3	La seva finalitat és detectar errors en l'assoliment dels requeriments.	C
4	El seu objectiu és la validació de l'aplicació per part dels usuaris.	B

A	Tipus de proves funcionals	**C**	Tipus de proves de sistemes
B	Tipus de proves d'acceptació	**D**	Tipus de proves d'integració

3- [1 punt] Completeu les definicions de conceptes relacionada amb els diferents tipus de proves.

Núm	Pregunta
1	Les proves [_D_] proven la funcionalitat del programa per al qual es dissenyen casos de prova que comprovin les especificacions del programa.
2	Les proves [_B_] se centren en la implementació dels programes per escollir els casos de prova. L'ideal seria cercar casos de prova que recorrin tots els camins possibles del flux de control del programa. Aquestes se centren en l'estructura interna del programa tot analitzant els camins d'execució.

A	de caixa blava	**B**	de capsa blanca
C	de capsa negra	**D**	de caixa negra
E	de funcions	**F**	de regressió
G	de sistema	**H**	d'integració

4- [1 Punt] Completeu les definicions de conceptes relacionada amb els diferents tipus de proves.

Núm	Pregunta
1	Un avantatge de les proves de [_A_] és que són independents del llenguatge o paradigma de programació utilitzat, de manera que són vàlides tant per a programació estructurada com per a programació orientada a objectes.
2	L'enfocament estructural o les proves de [__B_], dins les proves unitàries, serveix per analitzar el codi en totes les seves estructures, en tots els seus camins del programari. Però existeix un altre tipus de proves que es basen en un enfocament més funcional, anomenades proves de [_A_]

A	de caixa negra	**B**	de capsa blanca
C	de capsa negra	**D**	de caixa blanca
E	de funció	**F**	de sistema

5- [1 Punt] Marqueu si els següents continguts pertanyen (V) o no (F) a la planificació de proves

Núm	Pregunta	V	F
1	Tenen per objectiu arribar a la creació d'un pla d'actuació que es refereixi a quan i com es duran a terme les proves.	V	
2	Una anàlisi minuciosa del sistema i dels seus elements.	V	
3	El pla de proves ha de contenir totes les funcions, les estratègies, les tècniques i els membres de l'equip de treball implicats.	V	
4	Una anàlisi parcial de les estratègies, les tècniques i els membres de l'equip de treball implicats		F

6- [1 Punt] Trieu les opcions correctes escollint els tipus de proves que es desenvoluparan durant les proves del sistema.

Núm	Pregunta	S	Núm	Pregunta	S
1	Proves de rendiment		**6**	Proves d'acceptació	
2	Proves de robustesa		**7**	Proves de resistència	
3	Proves de depuració		**8**	Proves de seguretat	
4	Proves d'usabilitat		**9**	Proves d'instal·lació	
5	Proves de capsa blanca		**0**	Proves d'"alcohol"	

1. Tipus de proves de sistema
2. Proves de sistema
3. proves d'integració
4. proves de sistema
5. Tipus de proves de regressió
6. Tipus de proves de càrrega
7. proves de sistema
8. proves de sistema
9. proves de sistema

7- [1 Punt] Relacioneu les descripcions amb el seu significat

Núm	Descripció	Clau
1	És l'identificador que s'assignarà al pla de proves. És important per poder identificar fàcilment quin abast té el pla de proves. Per exemple, si es volen verificar les interfícies i procediments relacionats amb la gestió de clients, el seu pla de proves es podria dir PlaClients	A
2	Defineix l'abast del pla de proves, el tipus de prova i les seves propietats, així com els elements del programari que es volen provar.	B
3	Defineix les taques necessàries per preparar i executar les proves. Però hi ha algunes tasques que tindran un caràcter especial, per la seva importància o per la seva dependència amb d'altres. Per a aquest tipus de tasques, serà necessari efectuar una planificació més detallada i determinar sota quines condicions es duran a terme.	G
4	També és important definir els elements que no s'hauran de tenir en compte al pla de proves.	D
5	Defineix els documents que cal lliurar durant el pla de proves i en finalitzar-lo. Aquesta documentació ha de contenir la informació referent a l'èxit o fracàs de les proves executades amb tot tipus de detall. Alguns d'aquests documents poden ser: resultats dels casos de proves, especificació de les proves, subplans de proves…	H
6	Es defineix el responsable de cadascuna de les tasques previstes en el pla.	K
7	Defineix la tècnica a utilitzar en el disseny dels casos de prova, com per exemple la tècnica de capsa blanca o de capsa negra, així com les eines que s'utilitzaran o, fins i tot, el grau d'automatització de les proves.	E
8	En el calendari queden descrites les tasques que s'hauran d'executar, indicant les seves dependències, els responsables, les dates d'actuació i la durada, així com les fites del pla de proves. Una eina molt utilitzada per representar aquest calendari del pla de proves és el Diagrama de Gantt	L
9	Determina els elements del programari que s'han de tenir en compte en el pla de proves, així com les condicions mínimes que s'han de complir per dur-ho terme.	C
10	Per a cada tasca definida dins el pla de proves, s'haurà d'assignar un o diversos recursos, que seran els encarregats de dur-la a terme.	J
11	Defineix les circumstàncies sota les quals el pla de proves podrà ser alterat, finalitzat, suspès o repetit. Quan s'efectuïn les proves, s'haurà de determinar quin és el punt que provoca que se suspenguin, ja que no tindria gaire sentit continuar provant el programari quan aquest es troba en un estat inestable. Una vegada els errors han estat corregits, es podrà continuar efectuant les proves; és possible que s'inicïin des del principi	F

	del pla o des d'una determinada prova. Finalment, es podrà determinar la finalització de les proves si aquestes han superat un determinat llindar.	

A	Identificador del pla de proves	**B**	Descripció del pla de proves
C	Elements del programari a provar.	**D**	Elements del programari que no s'han de provar
E	Estratègia del pla de proves	**F**	Definició de la configuració del pla de proves
G	Tasques especials.	**H**	Documents a lliurar
J	Recursos	**K**	Responsables i Responsabilitats
L	Calendari del pla de proves.		

8- [1 Punt]Completeu els espais d'aquestes asseveracions relacionades amb la refactorització.

Núm	Pregunta
1	La refacció és [_G_] el codi font mantenint intacte el seu [_E_]
2	La refacció es considera un aspecte molt important per al desenvolupament d'aplicacions mitjançant programació [_A_]
3	Les proves i els casos de proves són bàsics per indicar fins a quin punt es pot [_E_] una tècnica de refacció sobre un codi font determinat.

A	Extremar	**B**	Externa
C	Transformar	**D**	Indicar
E	Optimitzar	**F**	Aplicar
G	Programara	**H**	Comportament

9- [1 Punt] Les proves i els seus casos de proves són bàsics per indicar si el codi font desenvolupat funcionarà o no funcionarà. Però també ajudaran a saber fins a quin punt es poden aplicar tècniques de *refactorització[1] de* codi font o no. Com caldrà implementar la *refactorització*? Posa en ordre les següents fases de la metodologia:

Clau	Fase de la metodologia	Ordre
1	Analitzar el codi font	2
2	Executar les proves.	5
3	Dissenyar les proves unitàries i funcionals.	3
4	Definir una estratègia d'aplicació dels canvis.	7
5	Desenvolupar el codi font	1
6	Modificar el codi font.	8
7	Analitzar canvis a efectuar.	6
8	Implementar les proves.	4
9	Execució de les proves.	9

10- [1 Punt] La utilització d'una eina d'automatització de les proves ofereix la possibilitat de generar els casos de proves, executar-los i comparar els resultats obtinguts amb els resultats esperats. Per tant, ens ofereixen, com a punts forts, una contraposició a la repetició dels errors a l'hora de desenvolupar les proves. Un altre punt fort de les eines d'automatització de les proves són les funcionalitats que ofereixen a tall de resum i que permeten tenir més informació tant de les proves com del codi desenvolupat. Ara bé, les eines d'automatització, no estan exents de punts febles.

Trieu de la següent llista els punts febles de les eines d'automatització.

1. El temps que cal dedicar a aprendre a fer servir correctament aquest programari.
2. Cada aplicació informàtica té les seves característiques i les seves especificacions a l'hora de ser utilitzada. S'hauran de conèixer bé per treure'n el màxim profit.
3. La correcta configuració i una bona selecció de les proves, els resultats obtinguts de les eines en la realització de les proves tampoc no seran
4. La fiabilitat de les proves, donat que es podria donar per bons uns resultats que no ho són.
5. La fiabilitat de les aplicacions és inversament proporcional al grau d'expertesa en el seu correcte ús.
6. Dins el projecte, serà recomanable tenir una persona que es dediqui de forma exclusiva a aquesta tasca.
7. El temps necessari a conèixer a consciència una eina informàtica com el que es dedicaria a efectuar les proves de forma manual.

11- [1 Punt] Marqueu si els següents continguts és una (A) avantatge o un (I) inconvenient de la tècnica de la refactorització. (opcional)

Núm	Descripció	A	I
1	Prevenció de l'aparició de problemes habituals a partir dels canvis provocats pels manteniments de les aplicacions.	X	
2	Excessiva dedicació de temps a la refactorització, provocant efectes negatius		X
3	Limitacions degudes a les bases de dades, interfícies gràfiques		X
4	Major enteniment de les estructures de programació	X	
5	Possibles problemes de comunicació		X
6	Detecció més senzilla d'errors.	X	
7	Repercussions en la resta del programari i de l'equip de desenvolupadors quan un d'ells aplica tècniques de refactorització.		X
8	Permet agilitzar la programació.	X	
9	Excés d'obsessió per aconseguir el millor codi font		X
10	Ajuda a augmentar la simplicitat del disseny.	X	
11	Personal poc preparat per utilitzar les tècniques de refactorització		X
12	Genera satisfacció en els desenvolupadors.	X	

Anotacions

[1] **Refaccionant codi font**

En enginyeria del programari, el terme *refactoring*, s'empra per a descriure el fet de modificar codi font sense canviar-ne el comportament observat des de l'exterior. La refacció sovint és una part del cicle de vida del programari: els desenvolupadors alternen entre la producció de noves funcionalitats i la refacció de codi ja existent per tal de millorar-ne la consistència interna i la claredat. Les proves de regressió asseguren que la refacció no ha canviat el comportament del codi observat des d'altres mòduls del programa.

El terme és una analogia amb la factorització de nombres i de polinomis. Per exemple, $x^2 - 1$ factoritza a $(x + 1)(x - 1)$, revelant una estructura interna que abans no era visible (les dues arrels +1 i -1). De forma similar quan es refacciona el programari apareixen estructures que restaven amagades en el codi original.

Aquests processos de refacció s'han produït de forma desorganitzada des dels inicis de la programació. Les metodologies modernes de programació, com ara la programació extrema, preveuen fases en el cicle de vida del programari on únicament es fan processos de refacció.

Una referència clàssica de la refacció del codi font és el llibre de Martin Fowler anomenat *Refactoring*. Tot i que, com s'ha dit, la refacció s'ha dut a terme informalment durant anys, l'article de 1993 de William F. Opdyke és el primer a analitzar la refacció.

La refacció és un concepte d'una importància cabdal, tant és així que ha estat qualificada com una de les innovacions més importants del programari

Alguns mètodes de refacció es troben amb problemes en ser utilitzats. Refer la capa de negoci d'una base de dades és difícil o impossible, a causa de la transformació de l'esquema i la migració de dades que ha d'ocórrer mentre el sistema pot estar sent utilitzat fortament. Finalment, la refacció que afecta a una interfície pot causar dificultats a menys que el programador tingui accés a tots els usuaris de la interfície. Per exemple, un programador que canviï el nom d'un mètode en una interfície ha d'editar totes les referències al nom vell a tot el projecte o mantenir un stub amb el nom vell. Aquest stub llavors cridaria al nou nom del mètode.