



Pla Docent de l'Assignatura

Guia Docent Programació d'Activitats

Curs acadèmic: 2012-13 **Trimestre:** 2n i 3r
Nom de l'assignatura: Enginyeria de Programari
Codi assignatura: 21418
Estudis: Enginyeria en Informàtica
Nombre de crèdits ECTS: 8
Nombre total d'hores de dedicació: 200
Temporalització:
Curs: 2n curs
Període: 2n i 3r trimestre
Professorat: Anders Jonsson, Toni Urcola , a determinar
Grup: 1

Guia Docent

1. Dades descriptives de l'assignatura

- **Curs acadèmic:** 2012-2013
- **Nom de l'assignatura:** Enginyeria de Programari **Codi:** 21418
- **Tipus d'assignatura:** Troncal
- **Titulació / Estudis:** Enginyeria en Informàtica
- **Nombre de crèdits ECTS:** 8
- **Nombre total d'hores de dedicació a l'assignatura:** 200
- **Temporalització:**
 - Curs: 2n
 - Període: 2n i 3r trimestre
- **Coordinació:** Anders Jonsson
- **Departament:** Departament de Tecnologies de la Informació i les Comunicacions
- **Professorat:** Anders Jonsson, Toni Urcola, ???
- **Grup:** 1
- **Llengua de docència:** Castellà, català
- **Edifici on s'imparteix l'assignatura:** Roc Boronat
- **Horari:**
 - Grup 1 Dilluns 18:30 a 20:30
 Dimecres 16:30 a 18:30
 Dijous 14:30 a 16:30

2. Presentació de l'assignatura

L'enginyeria de programari s'encarrega de desenvolupar i de fer el manteniment dels sistemes de programari perquè és comportin eficientment i s'hi pugui confiar. També es dedica a desenvolupar programari de manera assequible per tal de satisfer els requeriments dels clients i dels usuaris que han definit i especificat la solució. És una disciplina important per l'espectre i l'impacte ampli que té el programari en les activitats econòmiques de la humanitat i la societat moderna i pel paper que juga el programari en sistemes crítics i en la seguretat de moltes aplicacions. La disciplina integra mètodes formals de les matemàtiques, de les ciències de la computació i de les pràctiques i orígens de l'enginyeria.

Així doncs, l'enginyeria de programari és l'aplicació sistemàtica, disciplinada, quantificable i rigorosa d'enfocaments de desenvolupament, operació, manteniment i implantació del programari. Evidentment, l'estudi dels enfocaments que condueixen a la producció de programari de qualitat és la matèria central de l'enginyeria de programari. Simplement, l'enginyeria de programari es preocupa per construir correctament el programari. El programari no ha de fallar perquè, avui en dia, es controlen, es fan servir o es regulen milers d'activitats a través de sistemes que executen programari.

La enginyeria de programari engloba una sèrie de coneixements, definicions de processos, pràctiques, eines i estàndards. Tot aquest coneixement l'utilitzen els equips de treball que porten a terme tasques que produeixen productes de programari d'alta qualitat i que compleixen les necessitats dels usuaris. Les tasques inclouen l'anàlisi i l'especificació, el disseny, la implementació, la verificació, la realització de proves, el manteniment i la gestió dels projectes. Tot això a temps i dins del pressupost.

L'enginyeria de programari moderna també té un enfocament important perquè els projectes produeixin sistemes sostenibles i que el seu impacte al medi ambient sigui mínim i els seus objectius considerin també principis socials i ètics. Els enginyers de programari apliquen la tecnologia per tal de desenvolupar nous sistemes, construir nous instruments i utilitzar la tecnologia per al progrés i el benefici general, el mateix temps que es garanteix la qualitat dels resultats.

3. Competències a assolir en l'assignatura

S'han de tenir capacitats per al treball en equip, la solució de problemes, l'organització, la planificació, el disseny conceptual i les habilitats cognitives.

Competències generals	Competències específiques
Instrumentals 1.-Les habilitats cognitives (comprensió, interrelació d'idees i pensaments, anàlisi i síntesi) són molt importants per a la resolució de problemes, la planificació, l'anàlisi i el disseny. 2.- Sentit comú. De manera semblant, aquesta competència és una eina per resoldre els problemes de disseny i donar prioritat a restriccions i requeriments. 3.- Organització del temps. Ser capaç de resoldre un problema complex de forma sub-òptima, donades les restriccions de temps. Planejar anticipadament la gestió del temps és fonamental en totes les activitats d'enginyeria. 4.-Investigació documental i ús de fonts i referències. Com que el disseny és principalment una resolució conceptual de problemes, el fet d'utilitzar el coneixement existent és fonamental en el desenvolupament del programari de qualitat. Interpersonals 1.- Capacitat de comunicació. El context de treball de l'enginyeria de programari és normalment una col·laboració en equip on s'han de comunicar problemes, solucions i criteris, tant en un llenguatge tècnic com en un llenguatge per a usuaris que no són tècnics. La comunicació ha de ser efectiva tant de forma escrita com oral, en presentacions a clients o en equips de treball petits. 2.- Capacitat de comunicació amb experts d'altres àmbits. Els	Les competències específiques es resumeixen en tres grans capacitats. 1.- Coneixement de les eines, dels mètodes, de les pràctiques i de les eines de la enginyeria de programari. 2.- Capacitat per analitzar i descobrir requeriments d'un problema complex, la solució del qual involucra el desenvolupament d'un sistema de programari. 3.- Capacitat de dissenyar una solució per al problema complex que s'adhereixi als principis de l'enginyeria de programari i que satisfaci els requeriments funcionals al mateix temps que compleix amb especificacions d'alta qualitat i que s'implementa en el temps i el pressupost anticipats. Si es desglossen aquestes capacitats i competències, a continuació obtindrem els detalls següents sobre les competències específiques. 1.- Coneixement de les diferents metodologies de l'enginyeria de programari. 1.a.- Coneixement dels diferents tipus del "cicle de vida" del programari. 1.b.- Capacitat d'escollir una metodologia concreta entre les existents segons el cas concret. 1.c.- Capacitat de comprendre com el procés d'anàlisi i les decisions de disseny afecten el producte final. 2.- Capacitat d'analitzar i de descriure els requeriments d'un problema complex. 2.a.- Capacitat d'extreure els requeriments: identificar les parts interessades, els actors, els casos d'ús, classificar requeriments segons si són

enginyers de programari interactuen amb altres enginyers en projectes complexos, així com amb usuaris i consumidors del sistema de programari final. Les solucions són efectives si s'obté la informació clara i els acords necessaris. 3.- Lideratge. Ser capaç de prendre decisions en un grup de treball o de guiar el procés de pensament a solucions efectives. Sortir de la situació de falta de productivitat és essencial. 4.- Ús d'instruments tecnològics. Per poder aplicar diferents tecnologies, enfocaments i patrons, s'han d'utilitzar eines de manera idònia. 5.- Capacitat d'aplicar la teoria a la pràctica. La construcció de sistemes de programari de magnitud real requereix fonaments i l'aplicació de conceptes teòrics a la pràctica sòlida i sistemàtica.	funcionals, no funcionals o del domini. 2.b.- Capacitat d'expressar requeriments de forma organitzada i tècnicament concreta y unívoca. 2.c.- Domini d'algun llenguatge de modelització estàndard (per exemple UML) per expressar requeriments. 2.d.- Capacitat de treball en equip per capturar requeriments. 2.e.- Capacitat de comunicació de requeriments a clients, de forma clara per obtenir retroalimentació i validar requeriments. 3.- Capacitat de dissenyar una solució per a un problema complex d'enginyeria de programari a partir de requeriments. 3.a.- Capacitat de comprendre els requeriments donats. 3.b.- Capacitat d'organització i de planificació de diverses fases i tasques de desenvolupament. 3.c.- Presa de decisions de disseny. Per aconseguir-ho, s'han de seguir dos passos: a) Identificar les forces involucrades i trobar la millor solució que obtingui un balanç entre els criteris i les prioritats. b) Verificar que la solució tingui components i mòduls amb un mínim d'acoblament i màxima cohesió. 3.d.- Capacitat de comunicar de forma convincent i clara les decisions preses en la resolució d'un cas pràctic. 3.e.- Optimització del temps: capacitat de trobar una aproximació suficientment bona a la solució d'un problema complex. 3.f.- Capacitat de treball en equip. 3.g.- Coneixement de tecnologies orientades a objectes. Per això, s'ha de tenir: a) domini d'un llenguatge orientat a objectes (per exemple, C++)
Sistèmiques 1.- Gestió de la informació. Un problema complex es presenta amb molta informació, a vegades inconsistent o incompleta. L'enginyer de programari ha d'organitzar la documentació de requeriments, de gestió del projecte i del producte. 2.- Anàlisi de punts forts i febles. Ser capaç d'identificar el punt més dèbil d'un sistema és fonamental per jutjar la seva qualitat i seguretat. 3.- Presa de decisions. A partir d'una anàlisi prèvia, poder optar a les millors opcions disponibles. 4.- Introducció de millores. És molt important poder analitzar una solució i poder suggerir millores. 5.- Capacitat de generar idees noves. Per resoldre un problema complex, s'ha de disposar de la	

creativitat suficient per generar alternatives. 6.- Interès per la qualitat. Aquesta competència és fonamental per a l'aplicació de solucions tecnològiques a un problema. 7.- Disseny i direcció de projectes. S'ha de ser capaç de crear solucions que satisfacin els requeriments i de gestionar els processos que implementen el disseny. 8.- Anticipació de riscos. Cada projecte té riscos i possibles canvis. Per això, s'ha de fer una planificació que pugui ajustar solucions i dissenys a canvis o modificacions en els requeriments. 9.-Redisseny de processos. Cada procés que presenta poca eficiència o proporciona productes de baixa qualitat ha de ser revisat. 10.- Capacitat d'adaptació a solucions noves. Els canvis causen ajustaments i s'ha de tenir la capacitat d'adaptació. 11.-Capacitat d'aprenentatge. És molt important estar disposat a assimilar conceptes nous i noves tecnologies en una era de canvi tecnològic constant.	b) Coneixement d'un llenguatge estàndard de modelització (per exemple UML) c) Capacitat d'aplicar i utilitzar patrons de disseny. 3.h.- Capacitat d'implementar el disseny. Per això, s'ha de tenir capacitat d'obtenir retroalimentació, de revisar el codi i de fer proves per ajustar el disseny. 3.i.- Capacitat de realitzar proves i de validar la implementació d'una solució.
---	---

4. Contingut

La descripció breu del contingut és:

- Cicle de vida del programari
- Modelització visual i ús de llenguatges com UML
- Anàlisi i disseny orientat a objectes
- Ús i aplicació de patrons de l'enginyeria de programari
- Eines avançades d'implementació

La descripció detallada és la següent:

4.1. Part I: L'enginyeria de programari i els seus processos

Bloc de continguts 1: Cicle de vida i metodologies

Conceptes:

- Noció d'enginyeria de programari
- Relació de l'enginyeria de programari amb els àmbits professionals de la informàtica i amb altres professions
- Cicle de vida dels productes de programari i els seus models

Procediments:

- Identificació de les diverses parts o processos del cicle de vida genèric i de les activitats que es realitzen en cada fase
- Anàlisi dels avantatges i els desavantatges dels diferents esquemes per a un cicle de vida i en relació amb el context d'un projecte determinat

Actituds:

- Interès per l'enginyeria de programari i per la seva importància en la majoria dels àmbits de la professió informàtica
- Esperit crític per avaluar els avantatges i els inconvenients dels diversos models d'un cicle de vida i de les metodologies de l'enginyeria de programari

Bloc de continguts 2: Enginyeria de requeriments

Conceptes:

- Definició i classificació de tipus de requeriments
- Elements d'un document de requeriments i formalismes associats
- Cicle de vida de la enginyeria de requeriments

Procediments:

- Classificació dels requeriments funcionals, no funcionals i del domini
- Identificació de les parts interessades i d'actors

Actituds:

- Utilització de diagrames de casos d'ús en UML per analitzar i descobrir requeriments

- Redacció de documents de requeriments com a resultat d'una anàlisi d'un cas pràctic
- Manteniment d'un document de requeriments i gestió de la seva actualització com a resultat de canvis

Bloc de continguts 3: Metodologies

Conceptes:

- Descripció de diversos exemples de metodologies
- Mètodes àgils en oposició als tradicionals
- Metodologies basades en processos en oposició a metodologies basades en persones

Procediments:

- Identificació de beneficis i inconvenients de diverses metodologies
- Aplicacions de les pràctiques i processos d'algunes de les metodologies

Actituds:

- Actitud crítica per avaluar una metodologia

[4.2. Part II: Anàlisi i disseny orientat a objectes](#)

Bloc de continguts 4: La modelització amb UML

Conceptes:

- Avantatges de la modelització visual
- Definició i història breu de l'UML
- Els principals diagrames UML i els elements inclosos

Procediments:

- Utilitat dels diversos diagrames UML i el seu ús en diverses fases del cicle de vida
- Comprensió dels diagrames UML i la seva aplicació a un problema concret
- Utilització dels diagrames UML més importants (diagrames de casos d'ús estàtics i de seqüència) en la resolució d'un problema i, en menor mesura, d'alguns altres tipus de diagrames

Actituds:

- Acceptació de la importància d'utilitzar un formalisme estàndard a l'hora d'utilitzar diagrames

Bloc de continguts 5: Anàlisi i disseny orientat a objectes

Conceptes:

- Definició i propietats dels objectes
- Definició i propietats de les classes
- Les relacions d'herència, composició, agregació, associació i dependència
- El polimorfisme i les jerarquies d'herència
- El llenguatge C++ com a exemple d'un llenguatge orientat a objectes

Procediments:

- Divisió d'un problema complex utilitzant els conceptes i els procediments del paradigma orientat a objectes
- Utilització del llenguatge UML per modelar els sistemes orientats a objectes
- Utilització d'un llenguatge com l'UML per modelar els sistemes orientats a objectes
- Utilització d'un llenguatge com el C++ per implementar un disseny orientat a objectes
- Desenvolupament dirigit per proves

Actituds:

- Les bones pràctiques de programació (encasellament, no duplicació, mètodes curts i nomenclatura clara)
- Adequació a les convencions per a la codificació en l'equip de treball
- Adaptació al canvi, el codi que conforma una modelització en evolució constant
- Comprensió dels avantatges del desenvolupament dirigit per proves

Bloc de continguts 6: Patrons de programari**Conceptes:**

- Concepte de patró en el programari
- Catàleg de patrons i la seva utilitat

Procediments:

- Utilització de l'UMP per entendre un patró de disseny concret
- Utilització dels catàlegs de patrons per identificar un patró com a solució genèrica d'un problema concret
- Utilització de patrons en el disseny d'una solució per a un cas concret

Actituds:

- No s'ha de tornar a inventar la roda. La gran majoria dels problemes que apareixen en l'àmbit pràctic ja estan solucionats i la solució està disponible en forma d'un patró que segurament té característiques de qualitat superiors a allò que podem desenvolupar en una primera iteració aïllada

Bloc de continguts 7: Eines avançades d'implementació**Conceptes:**

- Aprendre a utilitzar eines avançades d'implementació
- Qt: llibreria per implementar interfícies gràfiques d'usuari
- Templates: programació genèrica

Procediments:

- Aplicació pràctica de Qt per implementar una aplicació gràfica
- Desenvolupament de programes amb templates

Actituds:

- Eines típiques de desenvolupament de software que cal dominar

5. Avaluació

5.1 Criteris generals d'avaluació

S'avaluaran tots els tipus de competències de la secció 3, així com també les competències específiques a través de les següents activitats que s'avaluen i són obligatòries:

- Quatre exàmens (al mig i al final de cadascú dels dos trimestres), corresponents als següents quatre temes:
 - El cicle de vida i les metodologies
 - Modelització visual i UML
 - Els patrons de disseny
 - La implementació amb C++
- Problemes pràctics resolts al laboratori (en parelles) i que poden formar part d'un projecte global durant el transcurs del curs en dos trimestres
- Participació i discussió durant els seminaris del material de la classe, les sessions de seminari i els exercicis lliurats després dels seminaris

La nota final de l'assignatura es calcula així:

- Quatre exàmens ($4 \times 10\% = 40\%$)
- 8 sessions de problemes al laboratori ($8 \times 5\% = 40\%$)
- 10 sessions de seminaris i lliurament d'exercicis pràctics ($10 \times 2\% = 20\%$)

Per aprovar l'assignatura s'ha d'aprovar per separat 1) cadascú dels quatre exàmens; 2) el projecte implementat a les sessions de pràctiques; i 3) l'assistència de seminaris i els exercicis lliurats després dels seminaris.

Les pràctiques i els seminaris són NO RECUPERABLES, doncs s'haurà d'aprovar aquestes parts durant el mateix curs.

Cada examen és recuperable per separat durant una prova adicional a juliol. No obstant això, per tenir dret a repetir un examen s'ha de complir els següents criteris: 1) aprovar al menys 2 dels 4 exàmens durant el curs; 2) aprovar la part pràctica; i 3) aprovar la part de seminaris.

5.2 Detall de l'avaluació

Quatre examens

A cada examen, es faran preguntes concretes sobre temes fonamentals de l'enginyeria de software, i es presentaran problemes reals que s'han de resoldre.

Què s'avaluarà?

Competències generals:

- Resolució de problemas
- Organització i planificació

Competències específiques:

- Capacitat d'analitzar i descobrir els requeriments d'un problema complex de l'enginyeria de programari
- Capacitat de dissenyar una solució per a un problema complex de programari a partir dels requeriments donats
- Capacitat d'aplicar l'UML
- Capacitat d'aplicar conceptes de l'anàlisi i el disseny orientat a objectes
- Capacitat d'aplicar i usar patrons de l'enginyeria de programari
- Capacitat de distingir entre diferents metodologies de l'enginyeria de programari
- Capacitat d'estructurar, escriure i llegir programes escrits a C++

Quan s'avaluarà?

- Al mig i al final de cada trimestre (2 i 3)

Sessions de problemes al laboratori

En el laboratori, s'introduiran eines que donen suport les metodologies de l'enginyeria de programari i que conformaran fases d'un projecte per realitzar en equip i on es posaran en pràctica les competències generals i específiques. S'ha de col·laborar amb una altra persona i implementar un projecte a C++, que es lliurarà de manera continua durant el llarg dels dos trimestres. El quart examen forma part de l'avaluació de les pràctiques, per assegurar que cada alumne hagi adquirit el coneixement esperat.

Què s'avaluarà?

Competències generals:

- Resolució de problemes
- Organització i planificació
- Treball en equip
- Comunicació efectiva i eficient

Competències específiques:

- Capacitat d'analitzar i descobrir els requeriments d'un problema complex de l'enginyeria de programari
- Capacitat de dissenyar una solució per a un problema complex de programari a partir dels requeriments donats
- Capacitat d'aplicar l'UML
- Capacitat d'aplicar conceptes de l'anàlisi i el disseny orientat a objectes
- Capacitat d'aplicar i usar patrons de l'enginyeria de programari
- Capacitat de distingir entre diferents metodologies de l'enginyeria de programari
- Capacitat d'estructurar, escriure i llegir programes escrits a C++

Quan s'avaluarà?

- Es reparteixen les 8 sessions de problemes durant els dos trimestres. Per tant, hi ha una sessió de problemes aproximadament cada 2-3 setmanes. Normalment, les activitats d'una sessió de problemes i laboratori s'hauran de completar durant i després de la sessió i s'hauran de lliurar els resultats d'una sessió de problemes abans de la sessió següent.

Participació en seminaris i discussió d'exercicis pràctics

Durant les sessions de seminari, es discutiran i es resoldran casos i exercicis pràctics. Els exercicis s'han de completar i lliurar després de cada seminari, típicament abans del seminari següent. L'objectiu és aprofundir en les capacitats generals i específiques i aconseguir una retroalimentació i avaluació contínua.

Què s'avaluarà?

Competències generals:

- Resolució de problemas
- Organització i planificació
- Treball en equip
- Comunicació efectiva i eficient

Competències específiques:

- Capacitat d'analitzar i descobrir els requeriments d'un problema complex de l'enginyeria de programari
- Capacitat de dissenyar una solució per a un problema complex de programari a partir dels requeriments donats
- Capacitat d'aplicar l'UML
- Capacitat d'aplicar conceptes de l'anàlisi i el disseny orientat a objectes
- Capacitat d'aplicar i usar patrons de l'enginyeria de programari
- Capacitat de distingir entre diferents metodologies de l'enginyeria de programari

Quan s'avaluarà?

- Es reparteixen les 10 sessions de problemes durant els dos trimestres, és a dir, 5 sessions cada trimestre. Normalment, les activitats d'una sessió de seminari consisteixen en la discussió de solucions de casos concrets i problemes específics i en la discussió col·lectiva o presentació per part dels estudiants de les seves solucions per debatre alternatives i obtenir una retroalimentació del facilitador del seminari. S'ha de lliurar la solució a cada exercici a dintre d'un determinat termini.

6. Metodologia

L'assignatura representa, per a l'estudiant, 18 sessions de teoria (2 hores cadascuna, que inclou dos dels examens), 8 sessions de problemes als laboratoris (2 hores cadascuna) i 10 sessions de seminari (2 hores cadascuna) durant dos trimestres.

Les sessions de teoria són classes magistrals on participen tots els estudiants i s'introdueixen els conceptes teòrics bàsics, les metodologies i es demostren els procediments adequats per a la resolució de problemes complexos amb eines i tècniques de l'enginyeria de programari.

Les sessions de seminari són en grups petits i consisteixen en la discussió de problemes concrets i en la retroalimentació immediata derivada de la discussió constructiva de solucions proposades per els estudiants i facilitadors d'exercicis concrets.

Les sessions de problemes són en grups mitjans però els estudiants treballen per parelles en la culminació de fases d'un projecte de programari aplicant també eines concretes per a la producció de programari en un context d'enginyeria de programari. Aquestes sessions de laboratori tenen com a objectiu una realització pràctica i una il·lustració concreta que permet utilitzar eines com les eines CASE, IDE o les de control de versions. El seu objectiu es posar els conceptes teòrics en pràctica i serveixen per complementar les competències i les capacitats en un sentit real, concret i pràctic.

7. Fonts d'informació i recursos didàctics

7.1. Bibliografia bàsica

- Fowler, Martin. *UML distilled: a brief guide to the standard object modeling language* / Martin Fowler with Kendall Scott Reading (Mass.) : Addison-Wesley, cop. 2000
- Meyer, Bertrand. *Construcción de software orientado a objetos* / Bertrand Meyer traducció: Miguel Katrib Mora, Rafael García Bermejo, Salvador Sánchez ; revisió tècnica: Jesús García. Madrid [etc.] : Prentice Hall, 1998
- Sommerville, Ian. *Software engineering* / Ian Sommerville Harlow : Addison-Wesley, 2009
- Larman. *Applying UML and patterns: an introduction to object-oriented analysis and design*

7.2. Bibliografia complementària

- Costal, Dolors. *Enginyeria del software especificació: especificació de sistemes orientats a objectes amb la notació UML* / Dolors Costal, M. Ribera Sancho, Ernest Teniente Barcelona: Edicions UPC, 2000
- Joyanes Aguilar, Luis. *Pla docent assignatura Estudis Programación orientada a objetos* / Luis Joyanes Aguilar Madrid: Osborne/McGraw-Hill, cop. 1998
- Peralta Giménez, Allen J. *Enginyeria del software: programació orientada a objectes* / Allen J. Peralta Giménez, Horacio Rodríguez Hontoria Barcelona: Edicions UPC, 1994 Upper Saddle River, N.J.: Prentice Hall PTR, cop. 1998
- Pressman, Roger S. *Ingeniería del software un enfoque práctico* / Roger S. Pressman adaptació: Darrel Ince traducció: Rafael Ojeda Martín [et al.] direcció, coordinació y revisió Madrid McGraw-Hill cop. 2002
- Schach, Stephen R. *Software engineering* / Stephen R. Schach Boston: Irwin, cop. 1993 Shtern, Victor
- Shtern, Victor. *Core C++: a software engineering approach* / Victor Shtern Upper Saddle River: Prentice Hall PTR, cop. 2000
- Stroustrup, Bjarne. *The C++ programming language* / Bjarne Stroustrup

7.3. Recursos didàctics

- Apunts en format lliure
- Col·lecció de problemes
- Web de l'assignatura al Moodle