Màster universitari en **Formació del Professorat d'Educació Secundària Obligatòria i Batxillerat, Formació Professional i Ensenyament d'Idiomes**

Treball de fi de màster

Títol: Estudi i introducció del Pensament Computacional al mòdul de Programació de CF d'informàtica

Cognoms: Gregorio García

Nom: Julián

Titulació: Màster en Formació del Professorat d'Educació Secundària Obligatòria i Batxillerat, Formació Professional i Ensenyament d'Idiomes

Especialitat: FP

Director/a: Emma Rollón Rico

Data de lectura: 21 de juny, 2018

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Índex

Figures

Acrònims

ASIX: Administració de Sistemes i Xarxes

CA: Criteri d'Avaluació

CF: Cicle Formatiu

CFGS: Cicle Formatiu de Grau Superior

DAM: Desenvolupament d'Aplicacions Multiplataforma

DAW: Desenvolupament d'Aplicacions Web

PC: Pensament Computacional

STEM: Science, Technology, Engineering and Mathematics

RA: Resultat d'Aprenentatge

UF: Unitat Formativa

1 Introducció

Aquest treball està motivat per la meva experiència docent a dos centres de Cicles Formatius de Grau Superior (CFGS) d'informàtica, on vaig impartir el mòdul de programació (M03).

Vaig experimentar els entrebancs que els alumnes trobaven en el moment de resoldre les activitats que se'ls plantejava. Tenia converses directes amb cada alumne, on em plantejaven els seus dubtes. De forma intuïtiva, els preguntava per tal de saber quins dubtes guardaven a les seves ments o, en altres paraules, quin model mental havien construït a partir de les classes magistrals i les activitats resoltes. Vaig tenir la sensació que alguns d'ells estaven molt perduts i que no tenien les eines per trobar el camí de sortida. Les seves dificultats em recordaven a les que sentien alguns dels meus companys en classe de primària quan s'enfrontaven a les matemàtiques: una mena d'incomprensió d'un món aliè al qual es parla un altre idioma i del qual, finalment, t'acabes desconnectant.

Vaig pensar que calia preguntar-se si les competències i estratègies associades a la programació dels Cicles Formatius estan prou desenvolupades al currículum i reben l'atenció necessària a les aules.

El Pensament Computacional (PC) és un concepte aplicat a l'ensenyament obligatori de moltes escoles arreu del món amb l'objectiu d'ensenyar els conceptes computacionals, afins a la programació, per a alumnes que no necessàriament han d'acabar sent programadors. Aquest concepte posa l'accent en el tot el procés del pensament previ a la codificació, justament la part on sentia que els alumnes tenen més dificultats.

Aquest treball vol comprovar si el PC pot ser una estratègia que ajudi a resoldre les problemàtiques que pateixen els alumnes que m'he trobat.

2 Definició i context del problema

La problemàtica que s'explica a continuació es refereix al mòdul de programació dels CFGS de la família d'informàtica. En particular, el que se sol impartir al primer curs dels dos que componen un cicle superior de Formació Professional, i que és comú a tots els cicles superiors de la família.

Per començar, s'analitzen els resultats acadèmics dels mòduls de programació dels CFGS d'informàtica d'un centre (centre 1) al qual vaig impartir classes aquest curs 2017-18. Seguidament, es comenten les metodologies d'aula per aquest centre i per un altre, del curs 2016-17 (centre 2), on vaig impartir classes del mòdul de programació.

2.1 Resultats acadèmics

A continuació es mostra una anàlisi dels resultats acadèmics del centre 1 on vaig impartir el mòdul de programació. Aquest centre imparteix tres CFGS:

- Tres grups de Desenvolupament d'aplicacions multiplataforma (DAM).
- Un grup de Desenvolupament d'aplicacions web (DAW).
- Un grup d'Administració de sistemes i xarxes (ASIX).

Cada CFGS té els seus propis mòduls en funció del seu currículum, però el primer curs del mòdul de programació és idèntic per a tots. Té tres Unitats Formatives: programació estructurada (UF1), disseny modular (UF2) i fonaments de gestió de fitxers (UF3).

Quan un alumne no assoleix la qualificació de 5, ha de repetir la Unitat Formativa. La següent anàlisi (figura 1) compara les repeticions d'UFs del mòdul de programació respecte de tot el cicle. Són dades per a dos cursos del centre 1: 2015-16 i 2016-17.

2015-16				CURS			MÒDUL PROGRAMACIÓ		
grup	alumnes	UFs del cicle	total de UFs	UFs repetides	% UFs repetides	nota mitjana	UFs repetides	% UFs repetides	nota mitjana
DAM 1	26	15	390	91	23,33%	5,97	31	39,74%	5,08
DAM 2	24	15	360	120	33,33%	4,98	33	45,83%	4,99
DAM 3	24	15	360	32	8,89%	6,46	13	18,06%	6,83
DAW	29	19	551	107	19,42%	5,29	45	51,72%	4,23
ASIX	27	15	405	123	30,37%	4,27	30	37,04%	4,56
TOTAL	**130**	**79**	**2066**	**473**	**22,89%**		**152**	**38,97%**	
2016-17				**CURS**			**MÒDUL PROGRAMACIÓ**		
grup	alumnes	UFs del cicle	total de UFs	UFs repetides	% UFs repetides	nota mitjana	UFs repetides	% UFs repetides	nota mitjana
DAM 1	19	15	285	54	18,95%	5,84	18	31,58%	5,42
DAM 2	20	15	300	110	36,67%	4,69	29	48,33%	4,37
DAM 3	32	15	480	57	11,88%	5,79	22	22,92%	6,19
DAW	30	19	570	89	15,61%	5,56	17	18,89%	5,86
ASIX	29	15	435	120	27,59%	5,03	44	50,57%	4,47
TOTAL	**130**	**79**	**2070**	**430**	**20,77%**		**130**	**33,33%**	

Figura 1: % de UFs repetides per al total del curs i per al mòdul de programació

Com es pot veure a la figura 1, el % d'UFs repetides per al mòdul de programació és sempre superior al % d'UFs repetides per a tot el curs. Per certs grups (2015-16: DAW o 2016-17: ASIX), fins i tot dobla el percentatge. En el global de cada any lectiu, tenim 22,89% versus 38,97% (2015-16) i 20,77% versus 33,33% (2016-17). Això suposa un increment de les repeticions al mòdul de programació del 70% i el 60%, respectivament.

Pel que fa a les notes mitjanes del curs i del mòdul, no hi ha gaires diferències. Per tant, i com en general hi ha més suspesos al mòdul de programació, podem concloure que entre els aprovats del mòdul de programació hi ha millors notes. És a dir, no és que el nivell general sigui pitjor, sinó que els resultats estan més polaritzats: és més habitual o bé no superar la UF o fer-ho amb millor nota.

Per tant, podem concloure que les competències associades a la programació s'estan assolint per sota de la resta del curs i que cal preguntar-nos quins problemes d'aprenentatge estan succeint.

2.2 Metodologia d'aula i problemes trobats

He tingut dues experiències docents. Al primer centre (centre 2) vaig treballar en un grup de reforç en llenguatge de programació C. Es tractava dels alumnes amb dificultats de programació per a dos grups del centre. Ells havien de lliurar una sèrie de problemes plantejats i em tenien a la seva disposició per resoldre dubtes durant la classe. Al segon centre (centre 1) vaig impartir programació en llenguatge Java. En particular, la segona Unitat Formativa (Disseny modular), quan els alumnes ja tenen uns certs fonaments de programació. Vaig preparar els meus propis continguts per a dues àrees de la programació molt específiques: els algorismes d'ordenació i la recursivitat. La recursivitat és un tema central a la programació i un dels més complicats d'ensenyar i aprendre, segons els professors (Lahtinen, 2005).

Les activitats que es proposaven als alumnes tenien un enunciat clar, i se'ls demanava que programessin una solució. Si atenem a la fase de reconeixement del problema, podríem parlar de tres tipus de problemes (Pretz, Naples i Sternberg, 2003):

- Els problemes que es **presenten**: el problema està definit i espera la solució. Aquest tipus és el que ens ocupa: el professor planteja el problema a l'alumne. Els altres dos tipus estan més relacionats amb el pensament crític i la creativitat.

- Els que es **descobreixen**. Necessiten una definició per part del descobridor.

- Els que es **creen**. Cal definir-los, i això pot ser un repte, ja que la seva necessitat és una invenció d'algú.

Per entendre les dinàmiques d'aula, parlaré de les observacions i intervencions que vaig fer al centre 1. Les classes magistrals són curtes i es fan al començament de la sessió. Se solen explicar conceptes generals de programació i se sol preparar el camí a les activitats pràctiques que es fan a continuació. Durant les classes magistrals, els professors interactuen amb els alumnes en major o menor mesura, segons el perfil docent. La part pràctica sol ser de treball autònom i individual. Els alumnes van fent preguntes al professor, que s'acosta a les seves taules i les resol. Si la pregunta és d'interès general, segons el perfil docent del professor, es fa una explicació a tota la classe, tot i que és complicat fer-los atendre un cop ja han començat les activitats.

En els dos centres, el canal per recollir les dificultats dels alumnes va ser la meva instrucció directa: quan em demanaven ajuda, teníem una conversa on els clarificava els seus dubtes. La meva actitud era sempre intentar donar-los eines per trobar les solucions, en lloc de proporcionar-les directament. Les preguntes em permetien entendre quin progrés havien fet i quina informació els permetia seguir avançant.

Un exemple d'actitud de l'alumne davant de la resolució de problemes podria ser aquest: l'alumne escriu codi directament al seu ordinador, utilitzant el llenguatge de programació que el centre decideix com a conductor del seu aprenentatge. Com que hi ha mancances en la comprensió dels conceptes fonamentals de la programació, aquest procés d'escriptura acaba sent de prova i error (Schulte i Carsten, 2013, p.19). Els demano que intentin treball abans el disseny, deixant de banda l'ordinador i, per exemple, utilitzin paper. Però el suggeriment té poca eficàcia.

Aquestes són alguns tipus de dificultats que vaig constatar després del contacte amb els alumnes dels dos centres:

- Dificultats per **entendre el problema**. L'alumne no comprèn què es demana a l'enunciat i es queda bloquejat.

- Dificultats per **pensar en la solució del problema**. L'alumne entén el problema, però no sap com dirigir-se a la solució, quines estratègies utilitzar i com dissenyar la solució.

- Dificultats per **entendre el funcionament** de la màquina abstracta que hi ha darrere. L'alumne es fa un model mental amb tot el que va aprenent, però aquest model pot no ser viable. En parlarem de la **màquina nocional** (du Boulay, 1986).

- Dificultats amb la **notació del llenguatge** (C, Java). L'alumne comet errors semàntics i de sintaxi quan escriu el seu codi. La formalitat del llenguatge és el repte.

Tot i ser molt diversos, els entrebancs que es trobaven els alumnes de programació es podrien classificar en dos grans grups:

1. Els relacionats amb el **pensament** previ a codificar, que permet trobar una solució algorísmica al problema plantejat.

2. Els relacionats amb la **codificació** de la solució mitjançant del llenguatge de programació.

2.3 Objectius

Els objectius que es planteja aquest treball són:

- Estudiar els motius que hi ha darrere dels problemes d'aprenentatge del mòdul de programació dels CFGS de la família d'informàtica.

- Estudiar el PC dins del context dels CFGS d'informàtica: què és i per què incorporar-lo al mòdul de programació.

- Incorporar el PC al mòdul de programació i avaluar la seva efectivitat en termes de resultats acadèmics i motivació dels estudiants.

3 Estat de l'art

Existeixen molts estudis i publicacions pedagògics que fan referència a les dificultats dels novells amb la programació. A continuació en faig un recull, per després identificar aquelles estratègies que puguin ser aplicables dins del context dels CFGS.

3.1 Les dificultats de la programació

Per a Harel (2004, p.401), la resolució de problemes mitjançant algorismes planteja tres tipus de complexitats a les quals cal fer front:

- La **computacional**, associada a trobar solucions eficients per als nostres problemes perfectament definits.

- La de **comportament** o sistema, ja que volem dissenyar sistemes computeritzats extremadament complexos, concurrents, distribuïts, etc.

- La **cognitiva**, ja que moltes vegades s'involucra un cert nivell de coneixement i intel·ligència de tipus humà de què es pretén dotar a la solució.

Aquestes complexitats inherents a la computació suposen obstacles que qualsevol alumne haurà d'enfrontar amb les eines que adquireixi durant l'exposició als mètodes computacionals.

Per tal de classificar els tipus de dificultats per àmbits, podem veure una descomposició del concepte de "programació" en subtasques (BlackWell, 2002): **entendre el problema**, **dissenyar**, **codificar** i **fer el manteniment**. Totes, menys la darrera tasca (el manteniment) són activitats que formen part del currículum dels CF (Ensenyament, 2013).

A continuació exposem les dificultats en l'aprenentatge de la programació per cadascuna de les tres primeres subtasques.

3.1.1 Entendre el problema

El problema pot estar aparentment ben plantejat, però no és suficient per entendre'l. Els problemes, habitualment, es proporcionen a l'alumne mitjançant enunciats de text. Whitten i Graesser (2003) identifica cinc nivells de representació del text:

- El **codi superficial**, o text literal, que no es memoritza, habitualment.

- La **forma proposicional** del text conté els significats de les paraules en forma de proposicions, però ignora les paraules exactes.

- El **model mental** és una representació de la realitat externa per tal d'explicar com funciona, i es construeix mitjançant interaccions amb el coneixement previ.

- El **nivell comunicatiu** relaciona els missatges del text dins del context que l'emmarca.

- El **gènere discursiu** assigna el text a una o diverses categories retòriques. En el nostre cas no varia: sempre es tracta de l'enunciat d'un problema.

El lector necessita les habilitats corresponents per tal de processar els codis (de comunicació) darrere del text, i ho ha de fer per als cinc àmbits simultàniament. L'exactitud de la solució dependrà del nivell de comprensió de cadascuna d'aquestes representacions del text.

3.1.2 El disseny de la solució

Mayer (1998) explica que els problemes no rutinaris plantegen un repte a l'hora de dissenyar una solució, i com és de difícil la transferència de l'aprenentatge d'habilitats de resolució de problemes. En parla d'allò que requereix el bon solucionador:

- **Habilitats bàsiques** relacionades amb el context del problema.
- Habilitat per a controlar i orquestrar els processos cognitius, el que denomina "**metahabilitat**". Permet decidir quan cal utilitzar una certa habilitat bàsica.
- **Motivació**, basada en l'interès en la tasca, el convenciment de la capacitat d'un mateix per a realitzar-la i el sentit de les atribucions que es fan als èxits i als fracassos.

Segons Pretz, Naples i Sternberg (2003), un cop que hem reconegut o identificat el problema, necessitem **definir** l'àmbit i objectius del problema i **representar** mentalment la informació de forma que estableixi un camí viable cap a la solució. Per a aconseguir aquest propòsit, el coneixement basat en experiències prèvies influeix en la nostra habilitat, però si el context és molt diferent, l'analogia és gairebé impossible. És a dir, el coneixement està molt lligat al context on es va crear.

3.1.3 La programació

Pane (2001) estudia quins són els errors habituals dels nous programadors, des que pensen una solució en paper, i abans de codificar-la. Aquestes són algunes conclusions:

- Hi ha la dualitat declarativa / procedimental: els alumnes volen escriure el codi amb estils declaratius, orientats a esdeveniments o a regles que començaven amb "if" o "when", quan els llenguatges que s'utilitzen als ensenyaments són procedimentals (pas a pas).
- Els costa escriure condicions senzilles, i poques vegades utilitzen la negació, que simplificaria les expressions.
- Als llenguatges orientats a objecte, els objectes no són tan intuïtius com semblen: com s'itera o s'accedeix a ells, el polimorfisme, l'herència, etc.
- El llenguatge de programació és formal, i l'alumne intenta interpretar-lo com si fos llenguatge natural. Passa especialment amb les fórmules matemàtiques, les assignacions, o els operadors booleans.
- Els costa seguir el progrés i recordar l'estat del programa. S'obvia molt freqüentment l'ús de variables, perquè són difícils d'entendre per a un novell.
- La confusió entre el món real i els objectes programats. Cal assumir que un programa dicta quin serà el comportament, i si oblidem actuar sobre un objecte, no passarà res.

Aquests errors visualitzen el xoc que es produeix entre el llenguatge natural amb el qual ens expressem i l'aspecte formal i abstracte de la programació, amb la seva alta càrrega cognitiva.

Lahtinen (2005) explica que l'aprenentatge de llenguatges formals representa un repte per als alumnes, que sovint provoca alts ràtios d'abandonament. A continuació s'exposen algunes característiques específiques de la programació que contribueixen a aquesta situació (Blackwell, 2002; Jenkins, 2002):

- S'involucren diverses habilitats i processos: resolució de problemes, matemàtiques, disseny d'algorismes, codificació, etc.
- És una disciplina abstracta: per a una certa representació d'un problema, cal extraure només aquells aspectes que segueixen cert patró, una aproximació oposada al principi de cooperativisme que regeix el raciocini humà, on tot el que experimentem contribueix de forma rellevant (Stanovich, 2003).

- S'introdueixen elements notacionals per a representar les abstraccions. Per tant, no hi ha una relació causal entre el problema real y la seva representació, sinó una convenció no intuïtiva que hem de memoritzar.

- Es tracta d'una novetat educacional radical (Dijkstra, 1988), i per tant no serveixen els paradigmes de l'ensenyament que s'utilitzen per adreçar-se a la majoria de disciplines.

- No té el benefici de la manipulació directa, que permet tenir una representació mental sobre la qual actuar, i per tant no existeixen uns límits a l'efecte d'actuar sobre el model.

- La programació és més una habilitat que un cos de coneixement, i no funciona la dualitat d'aprenentatge superficial versus aprenentatge profund, sinó que calen tots dos simultàniament: el superficial, per a detalls com la sintaxi o la precedència d'operadors, i el profund, per a desenvolupar de forma completa la competència.

A totes aquestes qüestions s'afegeix que la instrucció se sol realitzar en grups nombrosos i heterogenis, no contemplant el tracte en la diversitat. A més, el ritme està condicionat per l'avaluació i no per la diversitat dels alumnes. Sembla evident la necessitat d'enfrontar-se amb eines i mètodes que específicament es dirigeixin a aquestes circumstàncies.

3.2 Estratègies pedagògiques

A continuació veurem algunes estratègies pedagògiques no vinculades directament al PC, i que s'utilitzen des de fa temps amb l'objectiu de millorar l'aprenentatge de la programació.

Holvikivi (2010) parla de tres formes possibles de millorar l'ensenyament de la computació:

1. El desenvolupament d'eines d'aprenentatge.
2. Els mètodes pedagògics.
3. El currículum.

Tot i acceptar que hi ha eines desenvolupades específicament per a la didàctica i que tenen efectivitat per a contexts concrets, volem tractar les estratègies més intemporals associades als **mètodes pedagògics** i el **currículum**. Ho farem per aquests dos aspectes cabdals:

- La **resolució de problemes**, en el sentit més genèric, aplicable a altres disciplines.

- Les particularitats de **l'ensenyament de la programació**.

3.2.1 Resolució de problemes

Quan les respostes habituals no són satisfactòries a l'hora de resoldre un problema, cal millorar el repertori de processos de pensament o, dit d'una altra manera, cal desenvolupar les habilitats metacognitives. Blakey (1990) descriu algunes estratègies per a aconseguir-ho:

- Al començament de l'activitat, l'alumne ha d'identificar el que sap i el que no.

- Durant les activitats de resolució de problemes, els professors han de pensar en alt, per tal que els alumnes puguin seguir els seus processos mentals. Alternativament, es pot utilitzar la resolució de problemes per parelles.

- L'alumne pot mantenir un diari, reflectint els seus coneixements, les ambigüitats i com s'han resolt les dificultats.

- Assumir la responsabilitat de planificar la teva feina, organitzant les activitats, fent estimacions de temps i temporitzant-les.

- Quan es tanquen les activitats, cal preguntar-se pel procés de pensament que s'ha realitzat, per tal de fer-lo conscient.

- Autoavaluar-se, i així els alumnes poden reconèixer experiències amb similituds i transferir-les.

El context també és molt important. La instrucció ha de contemplar la pràctica de la resolució de problemes dins d'un context concret, per exemple, treballant en problemes reals. Això genera metahabilitats que es podran transferir a altres problemes (Mayer, 1998).

3.2.2 Programació

Els programadors novells aprenen programació mitjançant l'aprenentatge d'un llenguatge concret, que els servirà per aplicar-lo en la construcció d'artefactes computacionals. Dos llenguatges habituals són C i Java, tot i que la tendència ha anat canviant cap a altres llenguatges com Python (Guo, 2014).

Per a Robins (2003, p.140), la majoria dels ensenyaments de programació per a novells es dirigeixen mitjançant coneixement (contingut i estructura), i s'oblida un altre aspecte essencial: les **estratègies necessàries** per a aplicar aquest coneixement. També valora que el paradigma de programació escollit pot ajudar a la comprensió de certs conceptes, però no hi ha un consens universal de quina és la millor solució. A "Computer Science Curricula" (ACM i IEEE, 2013) es parla de la tendència actual d'utilitzar llenguatges més segurs o més dinàmics. Per altra banda, els llenguatges de més baix nivell permeten entendre millor l'execució de la màquina que hi ha al darrere, però resulten molt més complexos en un curs introductori.

Pea i Kurland (1984) descriuen els efectes cognitius d'aprendre a programar. Expliquen que hi ha la idea errònia que la programació consisteix a acumular dades i fets, i que només cal aprendre el llenguatge en termes de semàntica i sintaxi. Esgrimeixen que els alumnes assimilen millor en referència a un model que induint-lo (erròniament) des de l'exploració. Du Boulay (1989) va introduir el concepte de màquina nocional: el model idealitzat d'un ordinador implicat per les construccions del llenguatge de programació. Pensa que, per tal de facilitar el seu aprenentatge, cal que els llenguatges es caracteritzin per la seva **simplicitat** i **visibilitat**.

Als CFGS d'informàtica no és possible escollir el llenguatge només tenint en compte aquestes característiques. L'objectiu és preparar als alumnes per al mercat, i el mercat no admet ni llenguatges didàctics ni senzills. El que sí que podem fer és aproximar-nos, amb els llenguatges que hi ha al mercat, a escenaris que afavoreixin aquestes qualitats:

- **Simplicitat**: el llenguatge ha de contenir poques parts que interactuïn de forma intel·ligible. Es tractaria de trobar màquines nocionals simplificades, com la màquina nocional imperativa simplificada (Sorva, 2013, p.8:24). És a dir, reduir l'àmbit visible del llenguatge a aquelles funcions que ens resultin útils per a l'ensenyament.

- **Visibilitat**: el llenguatge ha de contenir mètodes per observar les parts i els processos en acció, la visió dinàmica que es contraposa a l'estàtica del codi. Això ho podem aconseguir de diferents maneres: dibuixant visualitzacions en paper, fent jocs de rol de la màquina nocional o utilitzant els depuradors dels entorns de desenvolupament (IDE).

Segons Hidalgo-Céspedes (2016) (figura 2), els alumnes construeixen els seus models mentals sobre la màquina nocional mitjançant els mètodes i materials dels cursos de programació. L'avantatge d'aquesta aproximació és que no cal dominar la màquina real, i per tant es redueix la càrrega cognitiva.

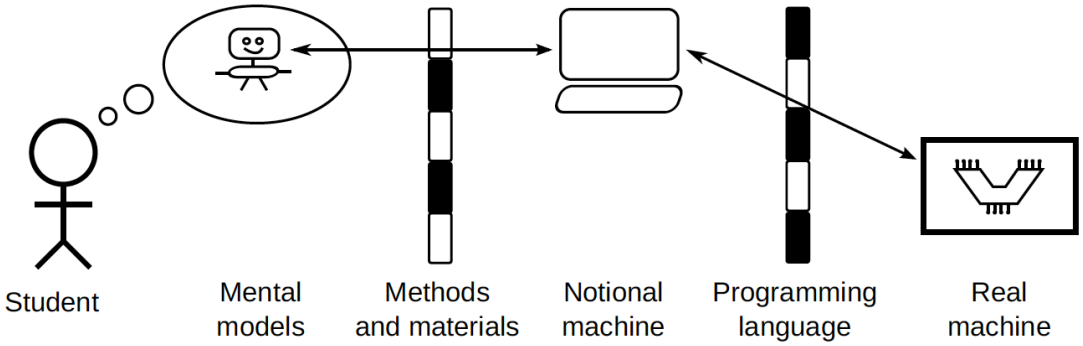


Figura 2: Procés de crear models mentals mitjançant màquines notacionals (Hidalgo-Céspedes, 2016)

Sorva (2013) recomana utilitzar un model conceptual a l'hora d'explicar la màquina nocional, amb metàfores i analogies senzilles, i que la instrucció del model sigui prèvia a l'explicació de les abstraccions associades a la programació.

Robins (2003) pensa que un curs introductori sobre programació ha de ser realista en les seves expectatives i sistemàtic en el seu desenvolupament: començar amb uns continguts, models i regles senzilles, i només expandir-les i refinar-les a mesura que l'alumne guanya en experiència. Cal deixar de banda les noves funcionalitats del llenguatge, només afegir-les en el context de solucions a problemes específics, i centrar-se més en l'aspecte del disseny bàsic d'un programa.

4 El Pensament Computacional

Pot ser el Pensament Computacional una estratègia vàlida a l'hora de millorar l'aprenentatge de la programació? Per tal de respondre aquesta pregunta, començarem amb un recull dels estudis que han tractat de definir aquest concepte.

4.1 Definicions

El Pensament Computacional (PC) és un tema que acapara molta atenció des de l'article que va escriure la Jeanette Wing (2006), i que ha portat les ciències computacionals als ensenyaments obligatoris de molts països. El terme va ser utilitzar per primer cop per Papert i Seymour (1980) al seu llibre "Mindstorms", referint-se a la necessitat de tenir escoles de computació per a tothom. Aquestes escoles tindrien la responsabilitat de construir coneixement amb una eina didàctica, Logo, un llenguatge de programació abastable per als novells i precursor de Scratch, un llenguatge visual de programació amb una àmplia comunitat activa a tot el món.

En la seva publicació, Wing (2006) explica que el PC és **la capacitat de pensar com un científic computacional**, i afirma que aquesta capacitat és fonamental per a tothom. Seguidament, Wing fa una llarga enumeració d'aquells aspectes que defineixen el pensament d'un científic computacional, com per exemple:

- L'ús de l'abstracció i la descomposició per emprendre tasques o dissenyar sistemes grans i complexos.
- La separació d'interessos, per tal de separar un artefacte computacional en parts amb un focus o interès delimitat.
- Trobar la representació apropiada per a un problema.
- Modelar els aspectes rellevants d'un problema per tal que sigui tractable.
- Utilitzar invariants per descriure declarativament i precisa el comportament d'un sistema.
- Tenir la certesa que podem utilitzar, modificar i influir en un sistema gran i complex sense entendre cada detall.

- Fer alguna cosa modular, anticipant el comportament o l'ús que se'n farà en el futur.

Per què pensa que el PC és una capacitat **fonamental**? Wing (2010) defineix els dos possibles destins d'aquest coneixement i la raó per la qual ho són:

- Per a **tothom**:
    - Entendre quins aspectes d'un problema es poden resoldre amb computació.
    - Reconèixer com es corresponen les eines computacionals amb les corresponents tècniques, i entendre les limitacions d'aquestes eines.
    - Aplicar o adaptar una eina computacional a un nou ús.
    - Aplicar les estratègies computacionals, com la de dividir i vèncer, a qualsevol domini.
- Per a **científics**, **enginyers** i **altres professionals**:
    - Aplicar els mètodes computacionals als seus problemes.
    - Reformular problemes per resoldre'ls amb computació.
    - Descobrir nova ciència mitjançant l'anàlisi de dades massives.
    - Enfrontar-se a nous problemes que, per la seva escala, eren inabordables sense computació.

La definició del PC ha estat sempre controvertida, però és completament necessària si volem establir uns resultats d'aprenentatge i uns criteris que puguin avaluar-los.

Aho (2011) té una definició de PC que ha estat utilitzada per Wing (2010): el PC són "**els processos del pensament involucrats en la formulació de problemes de tal forma que les seves solucions es puguin representar com a passos computacionals i algorismes**". I afegeix un aspecte important: cal trobar els models computacionals apropiats on formular el problema i derivar les solucions.

Selby (2013) també proporciona la seva definició, i ho fa seleccionant una sèrie de conceptes al voltant dels processos mentals de l'abstracció i la descomposició. Diu: "**és una activitat, habitualment orientada a un producte, associada a la resolució de problemes, tot i que no limitat a aquest aspecte**". I les habilitats del pensament que inclou aquesta activitat serien:

- Abstracció: la idea d'amagar la complexitat, definint diferents nivells d'abstracció per capes.
- Descomposició: la idea de trencar un problema en parts més petites, senzilles i assumibles.
- Algorísmica: el disseny precís i no ambigu d'algorismes. Té un terme anàleg: el pensament lògic.
- Avaluació: la idea que hem d'avaluar la nostra creació en funció d'allò que esperàvem, per tal de refinar la idea si cal. Introdueix el concepte d'autoavaluar-se, que és part del procés d'aprendre a aprendre.
- Generalització: l'habilitat que permet trobar una aplicabilitat més general de les solucions. Aquí també podem relacionar-ho amb el reconeixement de patrons.

4.2 La implantació del PC

La implantació del PC a les escoles d'ensenyament obligatori de tot el món ha estat un fenomen creixent els darrers anys. Els plantejaments de Wing (2006) van tenir molta repercussió en un moment que es va començar a prendre seriosament l'educació STEM (Science, Technology,

Engineering i Mathematics) i es volia incloure la computació a la seva definició. Va aprofitar la seva posició a la NSF (National Science Foundation) per tal d'influir en la inclusió del PC a l'educació primària dels Estats Units (Tedre i Denning, 2016). Algunes altres organitzacions com la CSTA (Computer Science Teachers Association), CAS (Computing at School, UK) i ACARA (Australian Curriculum, Assessement, and Reporting Authority) van presentar els seus propis marcs pedagògics de PC. Un cop el PC s'ha fet un tema popular a l'ensenyament primari, hi ha hagut moltes publicacions que han impulsat les seves pràctiques a les escoles.

A continuació, analitzem tres marcs pedagògics que ens permeten veure diferents visions de com introduir el PC als currículums dels ensenyaments. Els tres comparteixen una estructura semblant, amb les dimensions dels **conceptes** i les **pràctiques**.

4.2.1 Marc pedagògic del College Board

El College Board (2017) redacta un marc pedagògic per al currículum d'un curs anomenat "AP Computer Science Principles", que pertany al programa "Advanced Placement". Aquest té l'objectiu de formar a alumnes de secundària per preparar-los per la universitat en ciències computacionals. El que resulta interessant d'aquest marc és que ha estat dissenyat amb el Pensament Computacional com a element vertebrador. La seva estructura té dues dimensions: els set conceptes o grans idees, i les sis pràctiques.

Els set **conceptes** o **grans idees** són els principis fonamentals que es consideren essencials en la computació. Són els següents:

- I1. **Creativitat**: la computació es considera una activitat creativa que permet crear artefactes interessants i rellevants.
- I2. **Abstracció**: tècnica que redueix la informació i els detalls per tal de facilitar l'enfocament als conceptes rellevants.
- I3. **Dades i informació**: permeten la creació de coneixement mitjançant la seva traducció, procés i visualització.
- I4. **Algorismes**: s'utilitzen per a desenvolupar i expressar solucions a problemes computacionals.
- I5. **Programació**: permet la resolució de problemes, l'expressió humana i la creació de coneixement.
- I6. **Internet**: com a mitjà dels nostres temps per a estendre la computació a tot arreu.
- I7. **Impacte global**: la computació ha canviat com pensa, viu i juga la gent.

Les sis **pràctiques** de PC: s'enumeren els aspectes de la feina dels científics computacionals per als quals els alumnes hauran d'assolir les competències. Són les següents:

- P1. **Connexions amb la computació**:
    - Identificar els impactes de la computació en la societat.
    - Descriure la connexió entre persones i ordinadors.
- P2. **Creació de programes**:
    - Crear un programa amb una orientació pràctica, personal o social.
    - Seleccionar les tècniques apropiades per desenvolupar-ho.
- P3. **Abstracció**:
    - Explicar com es representen les dades, la informació o el coneixement per ser computats.

    - Explicar com s'utilitzen les abstraccions en computació o modelització.
    - Identificar abstraccions.
- P4. **Anàlisi de problemes i programes**:
    - Avaluar una solució proposada per a un problema.
    - Trobar i corregir errors.
    - Explicar com funciona un programa.
- P5. **Comunicació**:
    - Explicar el significat d'un resultat a un context.
    - Descriure la computació amb llenguatge, notació i visualitzacions acurades i precises.
- P6. **Col·laboració**:
    - Col·laborar amb un altre alumne per resoldre un problema computacional i per produir un programa.
    - Formar part d'un esforç col·lectiu des de la contribució individual.
    - Fomentar un clima col·laboratiu, resolent conflictes i facilitant la contribució d'altres.
    - Intercanviar coneixement i retroacció amb altres.

Tot i que al marc hi ha pràctiques que no són específiques de la programació (Internet, impacte global, etc.), el fet d'executar-les dins del context de la programació afavoreix la transferència de coneixement (Tedre i Denning, 2016).

Aquest marc pedagògic és una plantilla que permet utilitzar o no un llenguatge de programació. Allà on apareix el concepte "artefacte computacional" pot referir-se a un programa, una imatge, un àudio, una presentació o una pàgina web. En el nostre cas, l'hem substituït per "programa", l'artefacte computacional que ens ocupa.

Aquests són els **aspectes a destacar** del marc:

- Fa èmfasi en la relació de la computació i la societat, relacionant-la amb problemes reals que enfronten els alumnes, i com a via de motivació personal.
- Potencia l'expressió creativa dels alumnes.
- Demana que els alumnes interactuïn amb el món real, modelant-lo, fent simulacions i analitzant dades massives.
- Afegeix la necessitat de la comunicació escrita i oral per explicar la computació.
- Demana la col·laboració amb altres alumnes per resoldre problemes i produir programes.

4.2.2 Marc pedagògic de Brennan i Resnick

Brennan i Resnick (2012) defineixen el seu marc pedagògic a partir de les seves experiències amb programadors novells, descrivint tres dimensions essencials que han de ser presents a l'hora d'incorporar el PC a un projecte educatiu: els conceptes computacionals, les pràctiques computacionals i les perspectives computacionals.

Els **conceptes computacionals** (el què): és la dimensió que inclou tots els conceptes essencials que apareixen a la majoria dels llenguatges de programació. Són els següents:

- Seqüències: com, per expressar una tasca, ens expressem mitjançant passos individuals executables per un ordinador.

- Bucles: es tracta de mecanismes que ens permeten repetir seqüències múltiples vegades.

- Paral·lelisme: la possibilitat d'expressar la necessitat que hi hagi més d'una seqüència d'instruccions executant-se simultàniament.

- Esdeveniments: com un esdeveniment pot causar que passi una altra cosa.

- Condicions: l'habilitat de prendre decisions en funció de certes condicions per tal d'obtenir diferents resultats.

- Operadors: podem realitzar manipulació de números i cadenes de caràcters mitjançant les operacions matemàtiques, lògiques o de cadenes.

- Dades: involucren la manipulació de valors, que es poden desar, recuperar i actualitzar.

Les **pràctiques computacionals** (el com): descriu el procés de construir, pensar i aprendre, utilitzant els conceptes computacionals abans descrits. Són les següents:

- Ser incremental i iteratiu: aquest aspecte ens diu que dissenyar un programa requereix arribar a la solució en passos petits.

- Depurar i fer proves: ho necessitem per tal d'arribar a la solució que havíem pensat, ja que no podem anticipar absolutament tots els problemes que puguin sorgir.

- Reutilitzar i remesclar: aquest aspecte permet rebre noves idees, potenciar la capacitat crítica de llegir codi i fer-se preguntes sobre la propietat i autoria del treball.

- Abstraure's i modularitzar: la idea de construir una cosa gran a partir de parts més petites és la base del disseny i la resolució de problemes.

Les **perspectives computacionals**: permeten descriure els canvis de perspectiva que es poden observar als participants en la pràctica computacional. Són les següents:

- Com podem utilitzar la computació per a expressar-nos personalment.

- Com ens pot permetre connectar amb altres persones i practicar habilitats socials.

- La capacitat per fer-nos preguntes respecte dels aspectes computacionals que ens envolten a tots.

Aquests són els **aspectes a destacar** del marc:

- La part dels conceptes és de baix nivell en comparació als altres dos marcs. Potser un nivell no gaire pràctic per a la profunditat que es pretén en aquest treball.

- Inclou les perspectives computacionals com a tercera dimensió. Als altres dos marcs també apareixen aquestes idees, però integrades als conceptes i les pràctiques.

- És el marc de referència que s'utilitza a la revisió de l'ensenyament de PC mitjançant la programació que analitzem en aquest treball (Lye i Koh, 2014).

4.2.3 Marc pedagògic de Grover i Pea

Grover i Pea (2018) també descriuen el seu marc pedagògic, en aquest cas de dues dimensions: **conceptes** i **pràctiques**.

Els **conceptes computacionals** que es consideren essencials al PC són:

- La lògica i el pensament lògic: serveixen per analitzar situacions i prendre decisions o arribar a conclusions.

- Els algorismes i el pensament algorísmic: defineixen de forma precisa, i pas a pas, els plans o procediments per tal d'arribar a l'objectiu o resoldre un problema.

- Els patrons i el reconeixement de patrons: permeten arribar a solucions que permetin ser automatitzades i aplicades a casos generals.

- L'abstracció i la generalització: és el fet d'amagar els detalls en una capsa negra per tal de només concentrar-se en les entrades i les sortides.

- L'avaluació: qualsevol solució s'ha d'avaluar per la seva correcció i adequació basant-se en els objectius i les restriccions originals.

- L'automatització: la idea que la solució ha de ser executada per una màquina.

Les **pràctiques computacionals** que els científics computacionals farien servir per assolir els conceptes són:

- La descomposició de problemes en parts més petites, per tal de fer el problema més tractable i més fàcil de gestionar.

- La creació d'artefactes computacionals, que puguin finalment executar-se a una màquina.

- Fer proves i depurar, per tal d'avaluar el funcionament i resoldre errors mitjançant un procés d'aïllament.

- Refinament iteratiu (o desenvolupament iteratiu), una estratègia comuna a la programació per tal de fer créixer la solució amb proves i depuració freqüents per incloure millores.

- Col·laboració i creativitat, competències essencials del futur model d'ensenyament.

Els conceptes d'aquest marc pedagògic són més generals que els de Brennan i Resnick, i no hi ha la idea de les perspectives.

Aquests són els **aspectes a destacar** del marc:

- La divisió entre conceptes i pràctiques és molt clara.

- Inclou un concepte de disseny de software que no té els altres: el refinament iteratiu.

- No hi ha la idea de les perspectives del marc de Brennan i Resnick, tot i que dues perspectives apareix a les pràctiques: col·laboració i creativitat.

- És un bon marc en referència al concepte de programació, però sense baixar al nivell del marc de Brennan i Resnick.

4.2.4 Inclusió de la programació al marc pedagògic

Per acabar, considerarem un dels aspectes més rellevants a l'hora de definir un marc pedagògic per a la implantació del PC: **cal utilitzar o no un llenguatge de programació?**

Per situar-nos, veiem la definició de PC que fa Cuny (Wing, 2010), i a la que Wing fa referència: "és el procés mental implicat en la formulació de problemes i les seves solucions per tal que les solucions puguin representar-se i dur-se a terme **per un agent que processa informació**". Aquest agent al qual es refereix es pot interpretar de dues formes:

- Una **persona**, i llavors estem referint-nos al PC sense programació. Aquesta opció està relacionada amb iniciatives com la de CS Unplugged, un portal que treballa conceptes

de computació sense utilitzar ordinadors. Curzon i McOwan (2016) parlen extensament de jocs, màgia i puzles com a forma de convertir-se en un pensador computacional.

- Un **ordinador**, i llavors estem executant la nostra solució al mitjà computacional natural del PC. Grover i Pea (2018) asseguren que hi ha poques activitats que involucrin tants conceptes i pràctiques de PC com la programació, i per tant donen suport a la seva utilització. També és la forma que es vol considerar en aquest treball, ja que es pretén estudiar la inclusió del PC i el seu efecte a l'ensenyament de la programació.

4.3 Resultats de la implantació del PC

El marc pedagògic que hem vist de Brennan i Resnick (2012) és utilitzat per Lye i Koh (2014) per analitzar 27 estudis d'intervenció per al desenvolupament del PC mitjançant la programació. Aquest és el resum de les estratègies que contribueixen en positiu a la implantació efectiva del PC:

- Respecte de la **incorporació de la programació**, la pràctica és utilitzar llenguatges "low-floor" (fàcils per començar) i "high-ceiling" (amb recorregut per fer coses complexes).

- Respecte del **rendiment i els resultats**, es parla de la incidència a les tres àrees:

    - **Conceptes** computacionals: per als alumnes més joves, es millora el rendiment amb el treball de blocs visuals i l'ús de jocs. Per als més grans, els factors favorables són la programació en parelles amb metàfores, els mapes mentals, els projectes col·laboratius i utilitzar mètodes multi-sensorials.

    - **Pràctiques** computacionals: són favorables els mitjans que afavoreixen la visualització del programa, les pràctiques incrementals i iteratives, i les proves i la depuració.

    - **Perspectives** computacionals: la part més favorable és la dels alumnes que s'expressen a si mateixos.

- Respecte de les **estratègies d'intervenció**, es destaca el valor de les següents:

    - El reforç dels conceptes computacionals mitjançant jocs o e-learning.

    - La reflexió sobre el rendiment de la programació, amb revisions de codi propi, del company o del professor.

    - Estratègies de procés de la informació mitjançant metàfores, solucions a completar, mapes mentals, visualització de programes o conflictes cognitius.

    - Construcció de programes amb bastiment, sempre amb el suport necessari, evitant el pur descobriment, i en el seu lloc fer un descobriment guiat i estructurat.

Totes aquestes estratègies s'han identificat com a positives per potenciar l'aprenentatge del PC a l'ensenyament obligatori, que és l'objecte d'aquest estudi.

4.4 Crítica

Denning (2017) posa en dubte les premisses en què diu que es motiva el PC:

- El PC prepara millor a qualsevol jove per viure en un món cada cop més digitalitzat. Ell contesta que, encara que s'utilitzin eines computacionals, això no els fa desenvolupar el PC.

- Els pensadors computacionals seran solucionadors de problemes superiors en tots els camps. Ell contesta que el PC només beneficia als dissenyadors de computacions.

Respecte d'aquesta segona afirmació, Perkins i Salomon (1988) constaten que no hi ha resultats empírics que suportin que la instrucció de la programació pugui impactar les habilitats cognitives generals. Com es comenta a la seva teoria de la generalització de l'aprenentatge, la programació

és només un context per practicar el PC, però si volem que es produeixi la transferència cap a altres contextos, cal enfocar l'ensenyament en habilitats generals.

Denning (2017) considera que les definicions de PC tenen dues fonts d'ambigüitat associades:

- No mencionen el model computacional on els nostres pensaments finalment treballaran: tota l'abstracció, descomposició, representació, etc. ha d'estar orientada a fer una feina sobre un model concret.

- Suggereixen que qualsevol seqüència de passos constitueix un algorisme, sense tenir en compte que han de controlar un cert model, i per tant no hi ha d'haver cap judici humà.

Tedre i Denning (2016) fa una llista molt interessant dels riscos que comporta reivindicar el PC com la solució universal per a la resolució de problemes:

- Pèrdua d'ambició respecte de tot el coneixement produït prèviament en relació al PC, tractant de reinventar-lo.

- Dogmatisme, pensant que el PC és la millor eina per pensar i resoldre problemes.

- Conèixer versus fer: cal evitar l'avaluació del coneixement, i centrar-se a avaluar les competències.

- Reivindicacions exagerades, com la transferència d'habilitats del domini de la computació a altres, que no han estat mai substanciades.

- Visió estreta de la computació, deixant-ho només en aspectes tan concrets com la codificació.

- Massa èmfasi a la formulació, quan segurament caldria parlar més de disseny.

- Pèrdua de vista dels models computacionals: els nostres algorismes tenen un context, moltes vegades associat a una màquina física. No podem dissenyar sense tenir aquesta referència.

Finalment, comenta que hem d'evitar l'afirmació que tot es pot reduir a computació, o que el PC impulsa la revolució, quan és la revolució la que impulsa el PC.

5 Estudi dins del context del CF d'informàtica

Hem vist alguns marcs pedagògics de referència que expliquen com implantar el PC als ensenyaments, i també les millors estratègies per fer-ho. La pregunta que se'ns planteja ara és: té sentit incorporar-lo també al mòdul de programació de CFGS d'informàtica?

Per tant de contestar-la, necessitem un marc pedagògic de referència. Això ens permetrà valorar l'encaix del marc dins del currículum del mòdul de programació.

5.1 Marc pedagògic de referència

El marc pedagògic de referència (figura 3) s'ha construït amb els següents criteris:

- S'ha agafat el marc de Grover i Pea com a base, ja que té un bon equilibri de conceptes i pràctiques i també inclou competències transversals.

- S'han revisat els conceptes i pràctiques identificats com a positius a l'apartat 4.3 "Resultats de la implantació del PC", i s'ha escollit una pràctica que no era al marc de Grover i Pea: la visualització del model computacional (P8). Als alumnes els afavoreix realitzar visualitzacions de la condició dinàmica i l'estat dels programes mitjançant les eines i tècniques adequades.

- S'han afegit alguns aspectes més fronterers a la computació provinents del marc del College Board:

- Les dades i la informació (C7): cal integrar les dades i la informació provinents de l'acció humana als conceptes essencials que permeten imaginar noves solucions.
- L'impacte global (C8): necessitem una consciència de l'impacte que provoquem a la nostra societat amb els canvis que provoca la computació, tenint en compte els aspectes ètics.
- Comunicació (P6): cal potenciar la capacitat de comunicar-se dels alumnes, tan oralment com escrita.
- Connexió amb la computació (P7): els alumnes han d'estudiar els efectes de la computació a la societat i experimentar noves formes de connectar persones i màquines.

- S'ha comprovat que el marc està dins del context de les vuit competències bàsiques que ha d'assolir un alumne segons defineix el Decret 143/2007 d'ordenació dels ensenyaments de l'educació secundària obligatòria (figura 4) (Ensenyament, 2007). A continuació es mencionen els conceptes i pràctiques que ho corroboren:
  1. Comunicativa lingüística i audiovisual: P6 (comunicació).
  2. Artística i cultural: P5 (creativitat).
  3. Tractament de la informació i competència digital: C7 (dades i informació).
  4. Matemàtica: C1 (pensament lògic), C2 (algorismes).
  5. Aprendre a aprendre: C3, C4, P1 (resolució de problemes).
  6. Autonomia i iniciativa personal: P5 (col·laboració), P6 (comunicació).
  7. Coneixement i interacció amb el món físic: P7 (connexió amb la computació).
  8. Social i ciutadana: C8 (impacte global).

Conceptes	**Pràctiques**
C1. Pensament lògic	P1. Descomposició de problemes
C2. Algorismes	P2. Creació d'artefactes computacionals
C3. Reconeixement de patrons	P3. Fer proves i depurar
C4. Abstracció i generalització	P4. Refinament iteratiu
C5. Avaluació	P5. Col·laboració i creativitat
C6. Automatització	P6. Comunicació
C7. Dades i informació	P7. Connexió amb la computació
C8. Impacte global	P8. Visualització del model computacional

Figura 3: Conceptes i pràctiques del marc pedagògic de referència

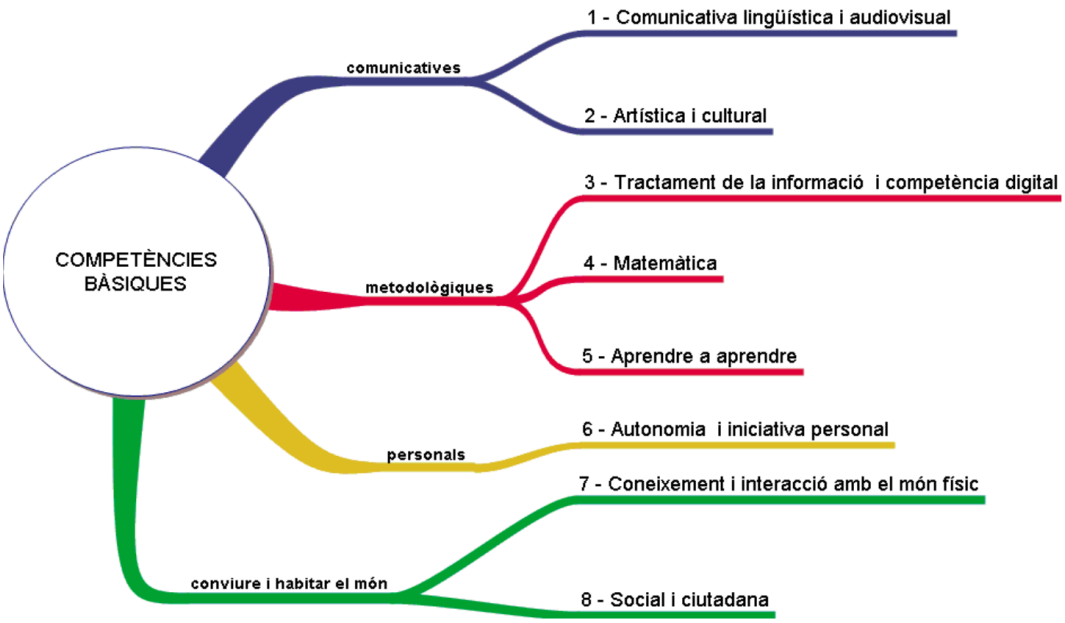


Figura 4: Les vuit competències bàsiques (Ensenyament, 2007)

5.2 Anàlisi dels problemes trobats

Per poder prendre la decisió d'incorporar el PC al mòdul, analitzarem si les dificultats dels alumnes descrites a l'apartat 2.2 "Metodologia d'aula i problemes trobats" d'aquest treball s'adrecen mitjançant els conceptes i les pràctiques del marc pedagògic de referència. Aquesta anàlisi es pot veure a la figura 5.

Dificultats trobades al CF	**Conceptes**	**Pràctiques**	**Anàlisi**
Entendre el problema	C1, C3, C4	Totes	Pot millorar construint un bon model mental, amb el coneixement provinent de les pràctiques.
Pensar la solució del problema	C1, C2	P1, P5	Els conceptes del pensament lògic i algorísmic són fonamentals, aconseguits mitjançant les pràctiques de descomposició de problemes, la col·laboració amb les idees d'altres companys i la creativitat.
Entendre el funcionament de la màquina abstracta	C1, C4, C6	P2, P3, P4, P8	Aquesta dificultat requereix moltes pràctiques i està associada a molts conceptes, i aquest marc en té uns quants per tal d'adreçar-la.
Notació del llenguatge	C6	P2	La notació està directament relacionada amb la pràctica reiterada de la programació.

Figura 5: Anàlisi de les dificultats en referència al marc pedagògic del PC

Com podem veure, tenim conceptes i pràctiques computacionals del marc pedagògic que s'adrecen a les dificultats que es van trobar a l'aula. Per tant, això ens fa pensar que incorporar el PC al mòdul de programació del CFGS d'informàtica pot incidir positivament en la resolució de les problemàtiques identificades.

Alguns conceptes (C5, C7 i C8) i pràctiques (P6 i P7) no es referencien directament a l'anàlisi. Tanmateix, la incorporació del PC al mòdul de programació es considera una bona oportunitat

per incloure'ls i fer de l'aprenentatge de la programació una experiència més creativa, social i solidària.

5.3 Currículum del CFGS

Ara estudiarem el contingut del currículum del CFGS del mòdul de programació, i veurem com podem encaixar-hi els conceptes i les pràctiques computacionals del marc pedagògic de referència. Si trobem carències, podrem donar suport a la integració del PC al currículum amb l'esperança que els problemes trobats a l'aula puguin ser adreçats.

Segons la web de la XTEC (Ensenyament, 2018), hi ha set Cicles Formatius de Grau Superior (CFGS) d'informàtica i comunicacions. S'hi poden distingir tres tipologies base, que són les significatives per l'àmbit d'aquest treball. El mòdul de programació d'aquests tres perfils base es pot veure comparat a la figura 6.

Cicle / Decret	**Mòdul de programació**	**Contingut**
Administració de sistemes informàtics en la xarxa DECRET 197/2013	MP3: Programació bàsica 165 hores (sense HLLD)	UF1: programació estructurada. 85 hores UF2: disseny modular. 50 hores UF3: fonaments de gestió de fitxers. 30 hores
Desenvolupament d'aplicacions multiplataforma DECRET 260/2013	MP3: Programació 297 hores (HLLD: 33h)	UF1: programació estructurada. 85 hores UF2: disseny modular. 50 hores UF3: fonaments de gestió de fitxers. 30 hores UF4: programació orientada a objectes (POO). Fonaments. 35 hores UF5: POO. Llibreries de classes fonamentals. 35 hores UF6: POO. Introducció a la persistència en BD. 29 hores
Desenvolupament d'aplicacions web DECRET 199/2015	MP3: Programació 297 hores (HLLD: 33h)	UF1: programació estructurada. 85 hores UF2: disseny modular. 50 hores UF3: fonaments de gestió de fitxers. 30 hores UF4: programació orientada a objectes (POO). Fonaments. 35 hores UF5: POO. Llibreries de classes fonamentals. 35 hores UF6: POO. Introducció a la persistència en BD. 29 hores

Figura 6: Comparació dels mòduls de programació de CFGS d'informàtica

Tenim que els CFGS anomenats de "Desenvolupament" tenen 297 hores, i el d'Administració, 165. De fet, tots tres tenen les tres primeres UF idèntiques, però el d'administració, més orientat a equips hardware i la seva gestió, conté una versió reduïda del mòdul amb només les tres primeres UFs. D'aquí el sufix "bàsica" al nom del mòdul.

Tot i que és decisió de cada centre, la distribució suggerida per Ensenyament a les "Orientacions als centres per a organitzar el cicle formatiu" (Ensenyament, 2016) és començar al primer curs amb les UFs 1, 2 i 3. Al segon curs (només per als dos CFGS de desenvolupament) es fan les altres tres: 4, 5 i 6. Això fa que el primer curs del mòdul de programació tots tres cicles sigui igual.

Per tal de fer consideracions sobre el currículum dels CFGS i valorar-lo en referència a altres ensenyaments semblants, necessitem seleccionar aquells continguts que puguin ser considerats els fonaments de l'ensenyament de la programació. En principi, les tres primeres UFs serien candidates a constituir aquests fonaments de la programació, però penso que la UF3 és un assumpte, la gestió de fitxers, que no es troba a la columna vertebral de la programació per la

seva especificitat. Fins i tot es pot considerar un tema obsolet en aquests currículums que daten del 2013/2015. Com a mínim, té massa atenció per a un curs introductori de programació. Per tant, les UFs que es consideraran fonamentals seran la 1 i la 2.

Les tres UFs del segon curs de programació (UF4, UF5 i UF6) se centren en la programació orientada a objectes, un dels paradigmes de programació més presents al mercat. Aquest és un dels debats més habituals a l'ensenyament dels llenguatges de programació: quin paradigma de programació és més convenient als cursos introductoris. Crida l'atenció que, una qüestió tan opinable, estigui directament com a contingut al currículum.

Per entendre millor el currículum, cal anar fins al detall dels resultats d'aprenentatge i els criteris d'avaluació que hi ha al Decret (Ensenyament, 2013). A la figura 7 es mostren els tres resultats d'aprenentatge de les dues primeres UFs.

UF	Resultats d'aprenentatge	Hores
1	**RA1**: Reconeix l'estructura d'un programa informàtic, identificant i relacionant els elements propis del llenguatge de programació utilitzat. **RA2**: Utilitza correctament tipus de dades simples i compostes emprant les estructures de control adients.	85h
2	**RA1**: Escriu i prova programes senzills reconeixent i aplicant els fonaments de la programació modular.	50h

Figura 7: resultats d'aprenentatge de les UFs de programació del CF

A la figura 8 es mostra l'anàlisi dels criteris d'avaluació dels tres resultats d'aprenentatge, relacionant-los amb els conceptes i les pràctiques del marc pedagògic de referència, per tal de veure quines mancances, en relació al PC, hi ha al mòdul de programació.

Criteris d'avaluació	Conceptes	Pràctiques
UF1 / RA1: Reconeix l'estructura d'un programa informàtic, identificant i relacionant els elements propis del llenguatge de programació utilitzat.		
1.1 Identifica els blocs que componen l'estructura d'un programa informàtic.	C6	
1.2 Crea projectes de desenvolupament d'aplicacions i utilitza entorns integrats de desenvolupament.	C6	P2
1.3 Identifica els diferents tipus de variables i la utilitat específica de cadascun.	C4	
1.4 Modifica el codi d'un programa per crear i utilitzar variables.	C4	P2
1.5 Crea i utilitza constants i literals.	C6	P2
1.6 Classifica, reconeix i utilitza en expressions els operadors del llenguatge.	C4	P2
1.7 Comprova el funcionament de les conversions de tipus explícites i implícites.	C4	P2
1.8 Introdueix comentaris en el codi.	C6	P2
UF1 / RA2: Utilitza correctament tipus de dades simples i compostes emprant les estructures de control adients.		

Criteris d'avaluació	Conceptes	Pràctiques
2.1 Descriu els fonaments de la programació.	C1, C2	
2.2 Escriu algorismes simples.	C1, C2, C3, C4	P1
2.3 Analitza i dissenya els possibles algorismes per la resolució de problemes.	C1, C2, C3, C4	P1
2.4 Escriu i prova programes senzills reconeixent i aplicant els fonaments de la programació.	C1, C3, C4	P1, P2
2.5 Utilitza estructures de dades simples i compostes.	C4	P2
2.6 Escriu i prova codi que faci ús de les estructures de selecció.	C4	P1, P2
2.7 Utilitza correctament les diferents estructures de repetició disponibles.	C4	P1, P2
2.8 Reconeix les possibilitats de les sentències de salt.	C4	P1, P2
2.9 Realitza operacions bàsiques, compostes i de tractament de caràcters.	C6	P2
2.10 Revisa i corregeix els errors apareguts en els programes.	C6	P3
2.11 Comenta i documenta adequadament els programes realitzats.	C6	P2, P6
2.12 Utilitza un entorn integrat de desenvolupament en la creació i compilació de programes simples.	C6	P2
UF2 / RA1: Escriu i prova programes senzills reconeixent i aplicant els fonaments de la programació modular.		
3.1 Analitza els conceptes relacionats amb la programació modular.	C3, C4	
3.2 Analitza els avantatges i la necessitat de la programació modular.	C3, C4	
3.3 Aplica el concepte d'anàlisi descendent en l'elaboració de programes.	C3, C4, C6	P2
3.4 Modula correctament els programes realitzats.	C3, C4	
3.5 Realitza correctament les crides a funcions i la seva parametrització.	C3, C4	
3.6 Té en compte l'àmbit de les variables en les crides a les funcions.	C4	
3.7 Prova, depura, comenta i documenta els programes.	C6	P3, P6
3.8 Defineix el concepte de llibreries i la seva utilitat.	C4	
3.9 Utilitza llibreries en l'elaboració de programes.	C6	P2

Criteris d'avaluació	Conceptes	Pràctiques
3.10 Coneix les nocions bàsiques de la recursivitat i llurs aplicacions clàssiques.	C2, C3, C4, C6	P1, P2

Figura 8: anàlisi dels criteris d'avaluació de les UFs de programació del CF

Aquestes són les conclusions de l'anàlisi dels criteris d'avaluació:

- El currículum està massa orientat a aprendre a programar, quan veritablement la programació hauria de ser un vehicle per assolir els conceptes computacionals mitjançant les pràctiques computacionals corresponents.

- Tot i que no és obligatori, l'ordre de la UF1 sol respectar-se als centres. Es comença per l'estructura d'un programa, en lloc de parlar dels veritables fonaments: el pensament lògic, els algorismes i la descomposició de problemes. Els currículums haurien de referir-se a l'ordre dels continguts que més afavoreix l'aprenentatge.

- Els continguts i els criteris d'avaluació tenen una correspondència un a un entre ells, i es barregen conceptes i pràctiques. Alguns criteris d'avaluació tenen conceptes associats però cap pràctica. Seria possible una altra forma de definir els criteris d'aprenentatge, tot i que no és l'objecte d'aquest treball.

- El pensament lògic (C1), tot i que s'ha inclòs a la taula (assumint que hauria de ser-hi implícitament), no és present explícitament, i sobta, sent un dels pilars de la programació. És important donar-li més rellevància al currículum.

- Hi ha poques pràctiques definides. La majoria són de creació d'artefactes computacionals (programes), i les altres dos que apareixen puntualment són la descomposició de problemes i les proves i depuració. Això no vol dir que no hi hagi pràctiques finalment al mòdul, però no estan suportades des del currículum, sinó des del sentit comú del professor. Aquest és el punt més feble del currículum.

- Falten conceptes com les dades i la informació (C7) i l'impacte global (C8) que, tot i no ser consubstancials a la programació, són rellevants en el context social de l'alumne, i el poden ajudar en la seva motivació i realització personal.

6 Incorporació al mòdul de programació

L'objectiu final d'aquest projecte és la proposta per a incorporar el PC al mòdul de programació. Aquesta UF incorporaria les conclusions que veiem al final del capítol "Estudi dins del context del CF d'informàtica", amb l'anàlisi que determina quines mancances caldria resoldre.

6.1 Normativa per a crear una UF

A continuació es descriu la normativa que permet afegir una nova UF als mòduls de programació.

El document "Orientacions als centres per a organitzar el cicle formatiu" dels CFGS d'informàtica (Ensenyament, 2016), al capítol 8, "Organització del currículum en unitats formatives", explica que les hores de lliure disposició que marca la llei per cada mòdul del CF es poden adjudicar lliurement pel centre mitjançant dos mètodes:

1. Distribuint-les entre una o més unitats formatives del mateix mòdul.

2. Incorporant-les en una nova unitat formativa del mateix mòdul (amb els resultats d'aprenentatge i continguts corresponents).

Dels CFGS d'informàtica existents, tots disposen de 33 hores de lliure disposició al mòdul de programació exceptuant el mòdul ASIX (Administració de Sistemes i Xarxes). Per tant, el segon mètode només seria aplicable a DAM i DAW. Pel que fa al mòdul ASIX, formalment no es podria crear aquesta nova UF, ja que no està previst a la normativa. L'alternativa seria recollir hores de

lliure disposició d'altres mòduls del mateix cicle. Tot i que no està descrita oficialment, aquesta opció és habitual als centres per tal de reestructurar els CF segons les prioritats curriculars que estableixen.

Un altre aspecte a resoldre és la temporització de la nova UF. Les orientacions (Ensenyament, 2016) permeten qualsevol distribució: al començament del curs, al final, al mig, en paral·lel, etc. tal com es veu a la figura 9. La decisió dependrà del centre, tot i que al següent apartat es fan unes recomanacions al respecte.

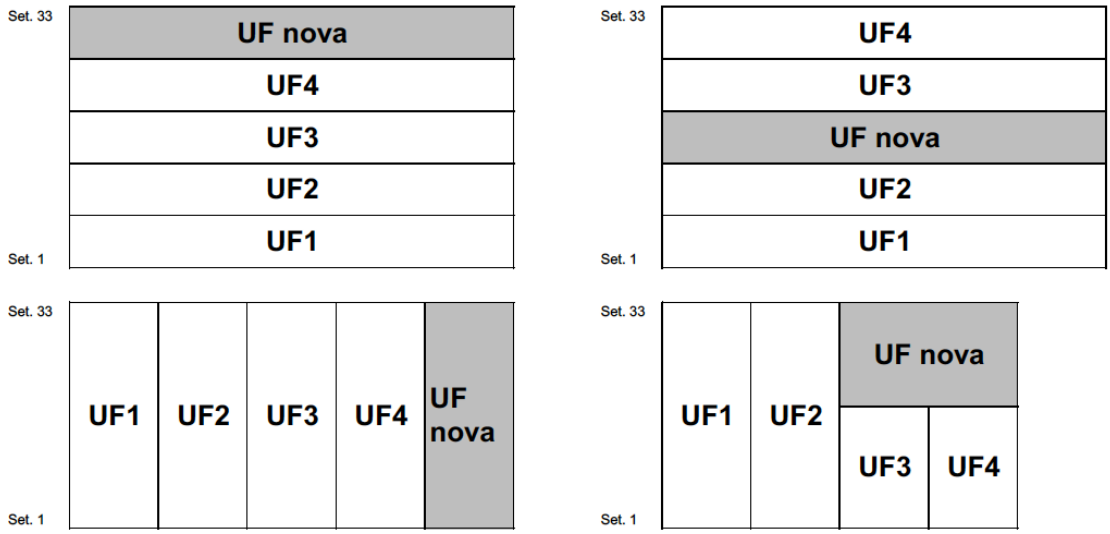


Figura 9: temporització d'una nova UF (Ensenyament, 2016)

Finalment, tal com s'explica al segon mètode, ens caldrà definir el resultat d'aprenentatge i els continguts corresponents per aquesta nova UF.

6.2 Disseny de la nova UF

Per a realitzar el disseny de la UF, s'han seguit les pautes descrites al document "Guia de programació de mòduls professionals (LOE) (Servei d'Ordenació de la FPI, 2011).

A continuació s'indiquen els apartats que es troben habitualment al Decret que defineix el currículum del CF. Aquests apartats són els que haurem de definir per a la nova UF:

- Els **resultats d'aprenentatge**: expressen les competències que ha d'adquirir l'alumnat per a poder desenvolupar funcions o processos i obtenir productes o resultats.
- Els **criteris d'avaluació**: indiquen les accions i els continguts de l'activitat o condicions que permeten valorar si s'ha aconseguit el resultat d'aprenentatge establert.
- Els **continguts**: són el conjunt de coneixements fonamentals que l'alumnat ha d'assimilar per assolir una determinada capacitat professional.

6.2.1 Resultats d'aprenentatge

La UF tindrà un sol resultat d'aprenentatge, derivat de la definició que fa Aho (2011) del PC: "**assolir les competències necessàries per a formular problemes de tal forma que les seves solucions es puguin representar com a passos computacionals i algorismes**".

6.2.2 Criteris d'avaluació

Aquests són els criteris d'avaluació, derivats dels conceptes computacionals descrits al marc pedagògic de referència:

1. Utilitza els conceptes lògics i l'àlgebra booleana en el disseny de solucions.

2. Explica i desenvolupa un algorisme per a ser implementat a un programa, i analitza les seves característiques d'eficiència, complexitat, correcció i claredat.

3. Troba patrons i prova hipòtesis sobre informació processada digitalment.

4. Descriu i utilitza la varietat d'abstraccions que permeten representar dades i escriure programes, així com els models i simulacions que representen fenòmens i proven hipòtesis.

5. Avalua la correcció d'un programa en relació als objectius o funcionalitats pels quals ha estat dissenyat, a l'estil de programació i a la solidesa del seu funcionament.

6. Desenvolupa un programa per a l'expressió personal, per curiositat, per crear coneixement o per resoldre un problema, de forma individual i en grup.

7. Utilitza la computació per processar, explorar i descobrir connexions de la informació i les dades massives, analitzant les implicacions de la transmissió i l'emmagatzematge.

8. Explica com les innovacions computacionals afecten la comunicació, la cognició, la innovació a altres camps i altres contextos socials, tant en positiu com en negatiu.

6.2.3 Continguts

Aquests són els continguts, derivats de les pràctiques computacionals descrites al marc pedagògic de referència:

1. Resolució de problemes mitjançant la descomposició.

2. Creació de programes a partir de dissenys abstractes.

3. Realització de proves i depuració de programes.

4. Utilització del refinament iteratiu com a estratègia de creació de programes.

5. Col·laboració en projectes col·lectius i d'expressió creativa.

6. Comunicació oral, abstracta i escrita sobre la computació.

7. Implicacions de la connexió de persones i computadors de diversos propòsits.

8. Tècniques de visualització de l'estat d'un programa.

6.3 Recomanacions per a programar la UF

A continuació es fan unes recomanacions per als professors que hagin de fer la programació d'aula per aquesta UF, i per tant hagin de dissenyar les activitats d'ensenyament i aprenentatge i la seva avaluació.

Les activitats han de definir-se en referència als continguts de la UF, que reflecteixen les pràctiques computacionals. Aquestes han de crear un coneixement en els alumnes que s'avaluarà mitjançant els criteris d'avaluació, que reflecteixen els conceptes computacionals.

Convé repassar el contingut de l'apartat 4.3 "Resultats de la implantació del PC" d'aquest treball, on s'explica quines estratègies funcionen en la incorporació del PC. Un dels aspectes comentats és la necessitat d'escollir un llenguatge de programació "low-floor" i "high-ceiling". Una possibilitat seria introduir-los a un llenguatge previ, més senzill, al que s'utilitzi a la resta de les UFs del mòdul. Un suggeriment és utilitzar Phyton amb IDLE (Grandell, Peltomäki, Back i Salakoski, 2006), un llenguatge present al mercat però amb una combinació interessant de senzillesa i potència.

En aquest procés d'aprenentatge és essencial generar **coneixement metacognitiu**, ja que habitualment està relacionat amb la transferència d'aprenentatge. L'ensenyament del coneixement metacognitiu ha de ser explícit per part del professor, encara que alguns alumnes

puguin assolir-ho de forma implícita. Per tant, ha d'haver-hi pautes a les classes per ensenyar-lo, i fer-ho etiquetant-lo explícitament. Una estratègia efectiva de la instrucció és que el professor pensi en veu alta mentre resol els problemes. Això fa que el grup classe pugui canviar el seu vocabulari i comunicar-se i intercanviar les seves estratègies cognitives, reflexionar sobre el propi aprenentatge i jutjar el seu autoconeixement (Pintrich, 2002).

Kirschner, Sweller i Clark (2006) comenten, respecte a la instrucció guiada, que és superior a altres basades en el descobriment quan es tracta d'alumnes novells. Per tant, és convenient que les activitats es dissenyin amb propòsits molt concrets relacionats amb les competències que es volen treballar.

Pel que fa a **l'avaluació**, Pintrich (2002) comenta que es pot aprofitar la classe per fer una avaluació informal del coneixement metacognitiu dels alumnes, per tal de calibrar la instrucció. Brennan i Resnick (2012) fan una sèrie de recomanacions per a avaluar PC mitjançant la programació:

- Intentar sempre que l'avaluació sigui útil per als alumnes.
- Aprofitar els dossiers d'aprenentatge per avaluar el desenvolupament d'un alumne al llarg del temps.
- Deixar que els alumnes parlin sobre el que han après. Fins i tot entrevistar-los, individualment.
- Avaluar als punts intermedis: no només el producte és important, també el procés.
- Evitar avaluar els conceptes, i veure altres formes de mesurar el coneixement: quan analitzen el treball d'un altre, o el reutilitzen, que el puguin explicar, criticar-lo, etc.
- Aprofitar quan hi hagi altres possibles avaluadors.

Pel que fa a la **temporització**, se suggereixen aquestes dues aproximacions:

- Fer la UF al començament del mòdul, com a prerequisit. Té l'avantatge que permetria arrossegar coneixement cap a les UFs posteriors, però com a experiment té més riscos, ja que el primer contacte amb l'alumnat podria no anar del tot bé.
- Fer-la en paral·lel amb la resta d'UFs. És menys arriscada, permet anar creant coneixement en paral·lel, però caldria plantejar-se si cal sincronitzar les activitats amb la resta de les UFs, o si cal reformular el seu contingut per no repetir continguts.

Pel que fa a la **formació**, caldria plantejar-se la realització de formacions de curta durada per als professors del mòdul de programació, per tal que sàpiguen com comunicar el PC als alumnes de forma efectiva i tinguin unes estratègies metacognitives adequades. A més, de retruc, aquesta formació podria fer que els professors reconsideressin el disseny de les activitats d'ensenyament i aprenentatge de la resta del mòdul, afegint aspectes relacionats amb el PC.

7 Una experiència en forma de taller

En aquest capítol s'explica una experiència en forma de taller. Abans d'entrar en el detall del taller, es fan algunes consideracions teòriques que defensen la conveniència de la integració dels jocs al procés d'aprenentatge.

7.1 Integració dels jocs a l'aprenentatge

El desenvolupament de jocs pot ajudar a l'aprenentatge del PC i de la programació. Van Eck (2006) explica els factors que fan els jocs interessants com a eines d'aprenentatge:

- L'interès creixent dels educadors i els investigadors sobre la matèria, amb dotzenes d'estudis publicats.
- La generació nativa digital que se sent desconnectada de la instrucció tradicional.

- La popularitat creixent dels videojocs.

Hi ha diversos conceptes relacionats amb aquesta qüestió que cal no confondre. L'un és la **gamificació**, que és el concepte més llunyà al que ens ocupa. La gamificació no és una eina d'ensenyament: el que pretén és motivar als jugadors mitjançant mecàniques de joc, com la competició, les històries, l'assoliment d'objectius, els nivells o els premis. De retruc, això pot ajudar al procés d'aprenentatge.

Un altre més proper és **l'aprenentatge basat en jocs**: el joc es dissenya al voltant d'uns resultats d'aprenentatge que ha d'assolir el jugador, al qual se li dona una retroacció personalitzada per ajudar-lo en el seu camí. Tampoc es va considerar aquesta opció per al taller.

Van Eck (2006) identifica tres aproximacions que tenen els educadors per a integrar els jocs al procés d'aprenentatge:

- Integració de jocs comercials a la classe. És la més pràctica, tot i que, com els jocs no estan dissenyats per a aprendre, cal fer-ho amb cura, ja que els temes poden ser incomplets o poc ajustats a les necessitats.

- Els educadors o programadors construeixen jocs educacionals des de zero per educar als alumnes. És una tendència a l'alça: els anomenats jocs seriosos.

- **Els alumnes construeixen jocs des de zero**. Aquesta és l'aproximació que volem tractar.

Muratet, Torguet, Jessel i Viallet (2009) expliquen que hi ha una tendència dels alumnes de programació a perdre l'interès i finalment deixar els estudis, i que està provocat per la falta de sentit i rellevància de la seva feina. Entre les aproximacions per enfrontar-ho es parla dels videojocs en dos sentits: jugar o implementar-ne un, fent ús d'un marc de desenvolupament com a bastiment. Hi ha un treball interessant de Gestwicki i Sun (2008) que tracta justament d'implementar un joc utilitzant una llibreria 2D, utilitzant els patrons de disseny associats al desenvolupament de jocs.

Els patrons de disseny, dins de la programació general, tracten de donar les millors respostes a les situacions o problemes més habituals amb que es troben els programadors i que, al llarg del temps, s'han anat recollint i anomenant per la seva alta freqüència d'aparició al codi. En el cas del desenvolupament de jocs hi ha també una particularització dels patrons, molt ben recollida en el llibre de Nystrom (2014) "Game programming patterns".

Aquest mètode permet a l'alumne endinsar-se als processos d'abstracció utilitzant els patrons de disseny associats als jocs. Són idees molt properes als conceptes i les pràctiques computacionals. El taller que es descriu a continuació es basa en aquesta aproximació.

7.2 Disseny del taller

La motivació d'aquest taller és introduir als participants al PC. S'ha fet integrant els jocs a l'aprenentatge dels alumnes mitjançant la seva construcció (Van Eck, 2006). És un assumpte que els interessa i en un context que els resulta familiar i que per tant els pot motivar.

El taller es diu "**El Pensament Computacional per mitjà del desenvolupament de jocs en Java**". Aquest és el guió i els seus objectius:

1. **Enquesta prèvia** per copsar el punt de partida dels alumnes.

2. **Introducció al PC**, la part més teòrica. En format de classe magistral, però amb interacció contínua amb els alumnes, fent-los preguntes per introduir-los als conceptes i a les estratègies que els desenvolupen.

3. **Reflexió sobre una llista de jocs coneguts**. Se'ls demana que expliquin com pensen que s'han dissenyat i com s'ha resolt la implementació dels diferents aspectes funcionals. Després de cada reflexió, se'ls explica quins patrons de disseny

es podrien haver utilitzat. L'objectiu és que tota la classe pugui participar i pensar en veu alta per exercitar la metacognició.

4. **Desenvolupament del joc "Pong"**, amb cinc activitats guiades a partir d'un codi de bastiment en Java. Se'ls demana que treballin, si és possible, en parelles de programació. Classe totalment interactiva, on es resolen els dubtes segons van demanant ajuda, i s'aprofita per parlar dels conceptes computacionals.

5. **Reflexió conjunta** final sobre la jornada.

6. **Enquesta posterior** per copsar l'impacte del taller.

El taller té una durada de dues hores: 45 minuts de classe magistral i reflexió sobre els jocs, 1h de desenvolupament del joc i la resta per fer les enquestes i la reflexió.

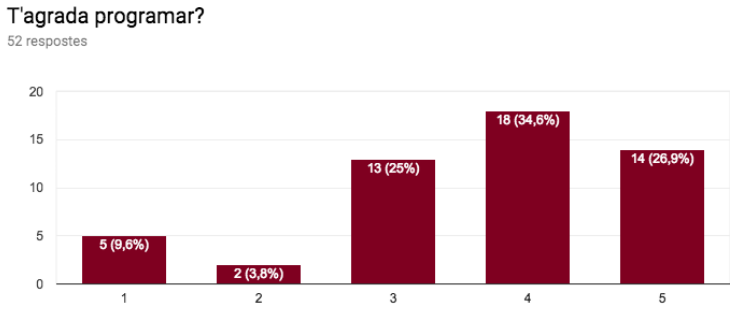
7.3 Impartició del taller

Es van realitzar dues sessions del taller amb alumnes del primer curs del mòdul de programació del centre 1. La primera sessió va ser amb alumnes del meu grup i la segona amb alumnes d'altres professors. Es va programar a mitjans de la UF2 (Disseny modular), quan els alumnes ja havien arribat als continguts essencials de la programació.

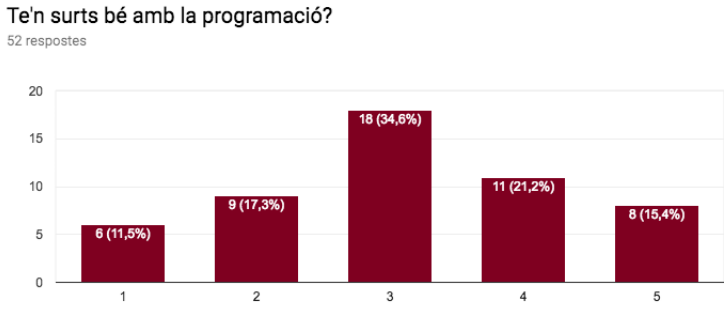
L'assistència, obligatòria, va ser d'uns 30 alumnes, la capacitat d'una aula. Tots eren nois.

7.3.1 Enquesta prèvia

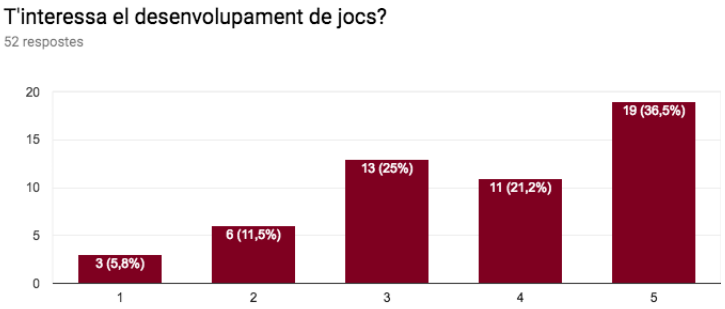
A continuació es comenten les respostes a les cinc preguntes d'escala Likert de cinc nivells (1: poc, 5: molt). Van contestar 52 alumnes.



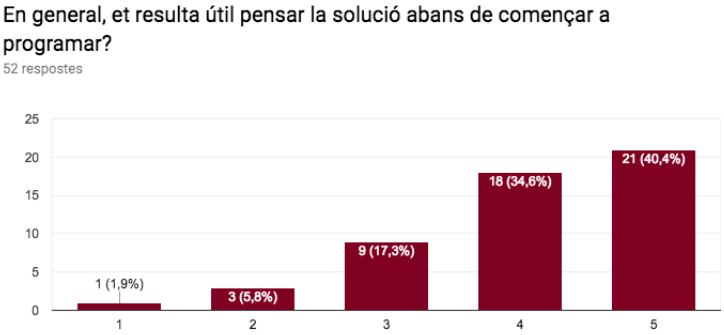
Resulta interessant comprovar que, tot i que hi ha un gran nombre d'alumnes als quals els agrada la programació, hi ha un percentatge important que es queden al punt intermedi (25%) o menys (13%).



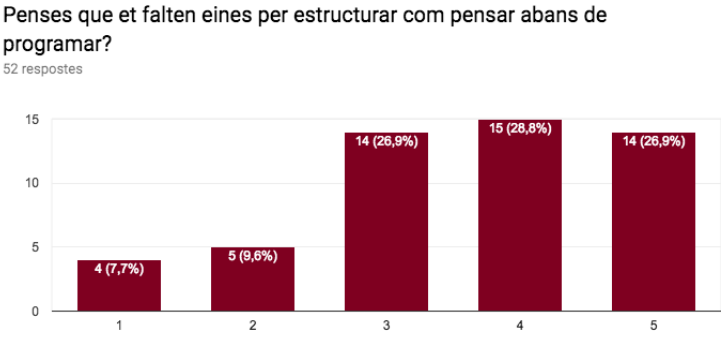
Els alumnes ja han fet exàmens de programació en aquest moment, pel que es tractaria d'una resposta no intuïtiva, sinó basada en els resultats.



El desenvolupament de jocs provoca un interès superior a la programació general (la primera pregunta), tot i que hi ha alumnes als quals no els agrada.



Els alumnes els sembla que pensar una solució té utilitat. Molt probablement, aquesta mateixa pregunta sobre un grup de programadors experimentats hauria resultat en una major predominança del resultat "poc", la que molts programadors experimentats escriuen el codi mentre dissenyen.



Aquesta pregunta estava orientada a identificar mancances en el pensament computacional dels alumnes, que es van confirmar amb les respostes.

Finalment, se'ls va fer una pregunta de resposta lliure: "**Explica el teu mètode per pensar la solució abans de programar**".

Aquestes són algunes estratègies interessants que comenten a les respostes: estar segur que has entès l'enunciat, representar el problema mitjançant esquemes, diagrames, dibuixos, dividir-lo en parts, cercar una seqüència de passos, plantejar alternatives, buscar al calaix de solucions anteriors, dirigir-se primer a les parts senzilles, començar per la resposta i anar enrere, imaginar-se la solució funcionant, etc.

- *Escric el desenvolupament del problema en un full, una vegada sé el que haig de fer, escric o intento escriure la solució en codi.*
- *Intentar imaginar el problema al cap.*
- *Cuando no me sale un programa suelo dibujarlo.*
- *Escric el mateix problema en un full i intento solucionar el problema en petites parts, encara que després no em surti res.*
- *Visualitzo el problema en parts, però no sé administrar bé el temps ni el ordre correcte de procedir.*

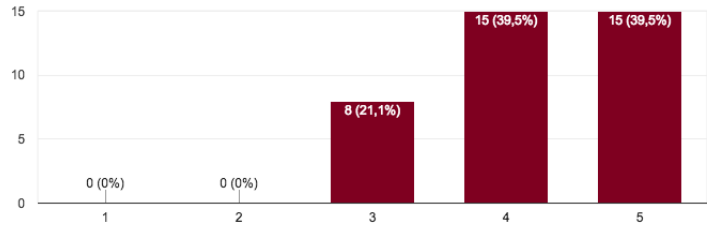
- *Escric en un paper varies formes de plantejar el problema/programa.*
- *Intento plantejar els problemes a una llibreta de una manera més visual per intentar entendre-ho, tot i que no sempre ho faig. Penso que a vegades em centro massa en el codi i no en resoldre el problema.*
- *Normalment escric la idea en un paper, faig dibuixos o penso en pseudocodi.*
- *Intento ver la solución paso a paso y luego como seria en código.*
- *Intentar imaginar com té que funcionar el programa e intentar saber que farà el programa en cada moment.*
- *Primer penso el codi com si el dibuixes amb fletxes, després penso la traducció a codi i el reviso.*
- *Leer, comprobar si tengo algo parecido hecho anteriormente, y si no, hacerlo.*
- *Intentar resoldre el problema a paper i recrear els mateixos passos que realitza el meu cap amb llenguatge de programació.*
- *Llegir l'enunciat mes d'una vegada.*
- *Intento desgranar el programa, dividint-lo en els diferents processos que ha de fer, i penso el codi que utilitzaré en cada procés per tal de poder seguir un guió i no saltar-me ni oblidar cap part.*
- *Pienso cómo se haría de forma mecánica, cómo funciona la herramienta y luego lo traduzco a código, lo que ocasiona que de la primera idea hasta el final el código va cambiando.*
- *En funció de la dificultat del problema segueixo diferents mètodes. Si el trobo fàcil començo a programar directament la idea general que el problema sembla que suggereix i treballo a partir d'aquí. Si no és tan immediat, un dels mètodes que intento fer servir és buscar una part del problema que sí que sembli més accessible i itero sobre ella, tot i que habitualment em trobo bastant perdut a l'hora de plantejar el problema de manera més general si no trobo cap "punt d'accés".*
- *Esquematitzar.*
- *Hacer un borrador en papel con los pasos a seguir para intentar solucionar el problema.*
- *Escribirla en un papel, pensarla y después probar.*
- *Pensar en el problema, pensar en la formula i aplicarla.*
- *Intento visualizar cómo funciona el programa antes de empezar.*
- *Croquis mental bàsic i anar provant.*
- *Paper i boli.*
- *Començar a escriure lo mes lògic i que vagin sorgint les idees.*
- *Posar la idea primer en paper.*
- *Faig el problema en paper per tal d'entendre-ho millor.*
- *Començo per la resposta, penso com ho faria jo a ma i ho passo a programació/pseudocodi.*
- *Avaluar els possibles casos que es poden donar, les excepcions i les estructures.*
- *Intentando comprender que es lo que pide el anunciado, y estructurarlo por pasos.*
- *No poder fer res perquè el professor planteja problemes que no sabem com resoldre ja que en ningun moment ha explicat tals conceptes.*
- *Doncs intento visualitzar el resultat i què ha de fer el programa per arribar fins al resultat.*
- *Hoja y papel, pensar paso por paso, hacer el programa de lo mas sencillo a lo más complejo.*
- *Por norma general cojo una libreta que llevo siempre junto al portátil y planteo allí el problema que me proponen.*
- *No tinc un mètode fixe.*
- *Normalment em concentro al problema i les possibles solucions amb el que ja sé, i si no trobo res d'utilitat amb el coneixement propi faig recerca per internet.*
- *Normalment sempre acudeixo a la recursivitat o "if" per plantejar una alternativa possible, després provo els codis que podria utilitzar de forma mes òptima afegint-los un a un per detectar errors.*
- *Pensar en com ho faria en paper i llapis.*

7.3.2 Enquesta posterior

A continuació es comenten les respostes a les cinc preguntes d'escala de Likert de cinc nivells (1: poc, 5: molt). Van contestar 38 alumnes.

T'ha semblat interessant el taller?

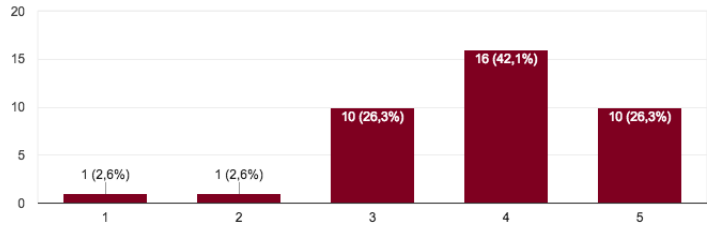
38 respostes



Als alumnes els va semblar interessant el taller.

T'ha resultat útil com a programador?

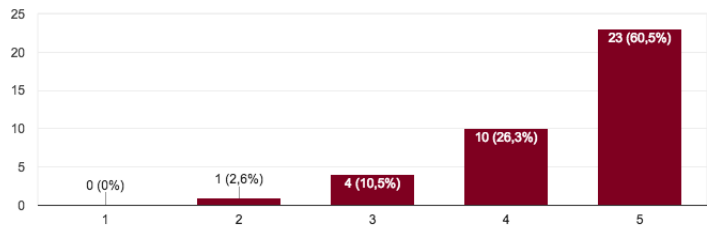
38 respostes



També els va semblar que era útil no només pels conceptes del PC o la temàtica dels jocs, sinó com a programadors.

Recomanaries el taller a algú interessat en programar jocs?

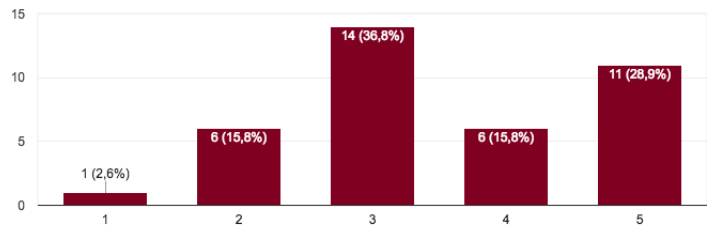
38 respostes



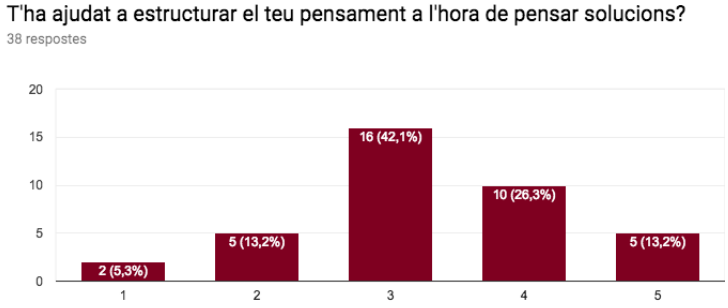
El taller està ben orientat cap a programar jocs. Tot i que els resultats d'aprenentatge pels quals estava dissenyat eren uns altres: introduir-los als conceptes del PC.

Recomanaries el taller a algú que comença amb la programació, com a eina per a pensar solucions?

38 respostes



Aquí hi ha opinions dividides, tot i que més positives que negatives. El taller sembla que pot ser interessant per introduir eines de programació per a novells.



La pregunta es refereix al PC com a mitjà per a estructurar el pensament. El resultat, molt dividit, tot i que amb una tendència positiva. Potser era massa optimista pensar que el taller deixaria una empremta significativa en una sessió de dues hores.

Aquestes són les respostes més interessants a la pregunta de resposta lliure: "**Explica quines coses t'ha fet veure aquest taller**". Aquesta pregunta tenia la intenció de fer-los reflexionar sobre el seu procés cognitiu. Respecte de l'actitud, diuen que cal constància, que va bé tenir curiositat, que cal treballar fort, com a la vida. Respecte de la dificultat del procés d'aprenentatge, hi ha opinions diverses: que semblava més fàcil, o que semblava més complicat. Respecte del procés de disseny previ a programar: que pot ser una eina artística, que cal fer un bon disseny abans, estructurar les idees, etc. Els falta un vocabulari més extens per poder expressar millor el seu procés d'aprenentatge.

- *Estructurar las idees.*
- *Que programar jocs pot ser molt mes divertit del que pensava.*
- *La complexitat dels elements mòbils als videojocs.*
- *No m'agrada la programació per a jocs.*
- *La forma de manejar la velocitat d'un objecte.*
- *Programar jocs no es tan fàcil com es pensa.*
- *Que dissenyar un joc senzill no es fàcil, però amb constància es pot treure un problema i resoldre'l.*
- *Que programar un joc no es gens fàcil, abans de posar-te a picar codi, necessites dissenyar bé el problema i totes les opcions que hi han, valorar-ho tot.*
- *Programar con objetos y pensar cómo tienes que hacer para que un objeto te haga caso.*
- *Encara que no ha estat el punt focal del taller, m'ha fet pensar que la visió de la programació com a eina d'expressió artística és interessant. No coneixia la programació en parelles. M'ha semblat molt bona idea si es compleixen certes condicions (sobretot la compatibilitat amb el company), almenys a nivell conceptual.*
- *M'ha donat ganes de començar a programar jocs.*
- *Estructurar més el pensament.*
- *El codi del joc sembla complex inicialment però un cop l'analitzes tot té lògica i sentit.*
- *Una forma de programació diferent relacionada amb elements gràfics i no processament de dades.*
- *El extens món de la programació, com estructurar bé els problemes i dividir-los en petites parts.*
- *Que a vegades les coses no són tan complicades con sembla a l'hora de programar.*
- *Que la programación en videojuegos consiste sobre todo en esfuerzo y dedicación además de tener la curiosidad por avanzar y descubrir nuevos métodos.*
- *Pensar lógicamente.*
- *Para crear un programa hay que hacerlo paso a paso y con tiempo y hacerlo los mas sencillo posible.*
- *El detall de com es fa un joc bàsic.*
- *Aprendre a crear videojocs en Java.*
- *M'ha fet veure que en la programació, com a tot en aquesta vida, s'ha de treballar de valent per a poder realitzar els teus somnis.*
- *La visión de los conceptos de la programación de objetos. Salir del cuerpo del main string [] args.*

També hi havia una pregunta lliure, "**Opcionalment, afegeix un comentari**". Les respostes són de satisfacció pel taller, alhora que es comenta que faltava temps per a la part pràctica.

- *Es més entretingut que tractar problemes de programació estàndard.*
- *Muy corto para todo lo que se quería ver.*
- *Ha estado bien como primeras nociones.*
- *Trobo que aquest tipus de tallers funcionen molt millor amb assistència voluntària, sobretot si cal programar en parelles/grups. Calia més temps per la part pràctica, almenys ~1h extra.*
- *Ha sido interesante.*
- *Molt bon taller, felicitats.*
- *Penso que es necessari més temps per aquest taller, ja que es força interesant y es podrien haver polit més detalls.*
- *Crec que ha sigut una experiència molt positiva en el nostre aprenentatge i ens servirà per a poder reflexionar i pensar més alhora de voler resoldre un problema mitjançant la programació d'objectes, gràcies per aquest taller tant interessant.*

7.3.3 Conclusions del taller

Els resultats per a les dues sessions van ser similars, tot i que els alumnes eren de diferents procedències: la sessió 1 eren alumnes del torn de tarda, als que jo donava classe, i els de la sessió 2 eren alumnes del torn de matí que no coneixia.

En termes generals, els alumnes han assolit nous coneixements de programació associats a les tasques que han completat en un context de diversió, motivació i recerca de solucions. Aquestes serien les principals conclusions:

- Es van preparar cinc activitats al taller, però només es va poder acabar la primera, i començar la segona. Per tant, no es va avaluar correctament el temps i el nivell dels alumnes.

- L'objectiu de millorar la capacitat d'estructurar el pensament, mitjançant els conceptes de PC, i amb el temps reduït i no totes les pràctiques de taller acabades, va ser massa optimista.

- L'impacte sobre els alumnes va ser-hi, però més en el sentit d'obrir una porta inicial que els permeti afegir unes primeres estratègies a la motxilla.

- A alguns alumnes no els agrada programar jocs, però les respostes a l'enquesta posterior fan pensar que una gran part d'ells van tenir una motivació extra gràcies a la temàtica.

- Les enquestes i, especialment les preguntes durant les activitats, van resultar molt interessants per a millorar l'orientació de la meva docència. Al marge de les dificultats per trobar solucions a les tasques plantejades, es van identificar alguns problemes associats als conceptes fonamentals de la construcció del codi, que suggereixen l'ús del PC com a estratègia.

- Ha estat important tot el material de bastiment que es va preparar per a anar directament a treballar al codi, sense necessitat d'entendre els detalls tècnics de com s'implementa el motor de videojoc en Java.

- El treball per parelles (*pair programming*) no va acabar de funcionar, perquè les aules tenien grups de tres taules que no es podien moure pel cablejat, i les parelles no eren fàcils de fer. A més, una de les sessions es va fer barrejant dos grups, i els companys de taula no coincidien amb els habituals, per la qual cosa hi havia menys afinitat.

- La part més positiva és la bona acceptació del taller per part dels alumnes, que en general van gaudir a partir de l'interès pel tema dels videojocs, aprenent aspectes de programació com a resultat de l'esforç que van demostrar. És a dir, en la línia del treball de Gestwicki i Sun (2008).

El principal problema del taller va ser la incorrecta adaptació al nivell dels alumnes, que va impedir acabar les activitats. Tot i això, les activitats finalitzades i encetades pels alumnes van funcionar bé a l'aula, i una proporció important dels participants van engrescar-se. Això, i els comentaris positius de l'enquesta posterior, fan pensar que el mòdul de programació o almenys

algunes UFs podrien estructurar-se al voltant d'un projecte motivant de desenvolupament d'un joc, sigui individual o col·lectiu.

8 Conclusions

Aquest treball volia comprovar si la incorporació del Pensament Computacional (PC) al mòdul de programació del CFGS d'informàtica podia ajudar a resoldre les dificultats que vaig identificar durant les classes que vaig impartir. Quines competències o estratègies els faltaven?

Per començar, es va voler comprovar si els problemes dels alumnes tenien un reflex als resultats acadèmics del centre 1. Es va observar que el percentatge de repeticions d'Unitats Formatives (UF) al mòdul de programació s'incrementava entre un 60% i un 70% respecte del total del cicle. Per tant, els resultats al mòdul de programació eren significativament pitjors.

El següent pas era estudiar els motius de les dificultats d'aprenentatge de la programació. Es van revisar els estudis que fan referència als problemes dels alumnes que s'enfronten per primera vegada amb la programació. Molts d'ells es fonamenten en el xoc que es produeix entre les nostres formes de pensar i expressar-nos i l'aspecte formal i abstracte de la programació, que a més a més té una alta càrrega cognitiva.

Calia comprovar si el PC era una estratègia vàlida per modificar el currículum i els seus mètodes, amb l'objectiu de millorar l'aprenentatge de la programació. Primer, s'havia de definir el concepte de PC per tal de conèixer les aproximacions que permeten integrar-lo al currículum. Després es van identificar uns marcs pedagògics que definien com s'havia d'integrar el PC als ensenyaments mitjançant dos aspectes: conceptes computacionals i pràctiques computacionals.

Combinant els marcs analitzats, es va crear el marc pedagògic de referència, que havia de permetre estudiar la incorporació al mòdul de programació. Amb aquest marc:

- Es va comprovar que adreça les dificultats trobades a l'aula.
- Es va analitzar detalladament el currículum del mòdul de programació per comprovar si cobria o no els conceptes i pràctiques del PC, i així decidir si tenia sentit incorporar-lo. Aquesta anàlisi va concloure que:
    - El mòdul de CF està massa orientat a la programació.
    - Falta reflectir conceptes computacionals essencials, com per exemple, el pensament lògic.
    - S'han identificat poques pràctiques computacionals, i per tant falten estratègies sòlides per fomentar la metacognició dels alumnes.

L'objectiu final ha estat el disseny d'una nova UF que integra el marc pedagògic de referència. Els professors dels mòduls de programació poden utilitzar-la per incloure el PC a les seves aules i avaluar la seva efectivitat en la millora de l'aprenentatge dels alumnes.

Finalment, l'experiència en forma de taller va resultar molt interessant com a primer contacte dels participants amb el PC. Vam parlar dels conceptes i pràctiques computacionals mitjançant un tema que els resulta proper i engrescador: els jocs i les seves estratègies de disseny. Necessitem la motivació com a catalitzador de l'aprenentatge, i aquesta activitat, com altres de tipus creatiu o d'expressió personal, té el potencial de generar-la.

Quant a treball futur, caldria programar una UF amb el disseny que s'ha fet, i avaluar la seva efectivitat en l'aprenentatge de la programació. Quant a l'avaluació d'aquesta UF, seria important dirigir-la per avaluar els progressos metacognitius dels alumnes, tal com explica el treball. El següent pas, molt més ambiciós, seria redissenyar tot el mòdul de programació dins del marc pedagògic de referència que s'ha proposat. Aquesta aproximació faria del PC el fil conductor de tot el mòdul, i permetria tenir un currículum més coherent i amb uns fonaments clars.

Hi ha moltes publicacions sobre les dificultats amb la programació que són perfectament vàlides avui en dia i que cal seguir revisant. És fàcil deixar-se portar pel moment d'apogeu del PC i ignorar-les. El PC no és, segurament, una solució per a tothom, però ens pot ajudar a canviar on posem el focus de l'ensenyament de la programació.

Quina direcció ha de prendre l'ensenyament de la programació i dels CFGS d'informàtica en general?

Pel que fa a l'ensenyament de la programació, hem de preparar als alumnes perquè siguin els millors pensadors computacionals, amb les millors habilitats metacognitives, i que els llenguatges de programació siguin l'eina principal per a aconseguir-ho, no la finalitat. Els nous temps ens porten a espais virtuals i d'autoaprenentatge, i ens cal identificar quin valor afegit es pot oferir des de les aules per a fer-les valer.

Pel que fa als CFGS d'informàtica, l'ensenyament ha de sortir de l'aula i obrir-se a la societat en què viu. Els alumnes han de poder expressar-se, ser creatius i col·laborar en la transformació del món que els envolta.

9 Bibliografia

ACM, & IEEE (Eds.). (2013). «Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science». ACM, Inc. Disponible a: https://doi.org/10.1145/2534860

Aho, A. V. (2011). «What is computation?». *Ubiquity*, *2011*(January), 1. Disponible a: https://doi.org/10.1145/1922681.1922682

Akker, J. van den, Boer, W. de, Folmer, E., Kuiper, W., Letschert, J., Nieveen, N., & Thijs, A. (2009). «Curriculum in Development». *Netherlands Institute for Curriculum Development*. Disponible a: https://doi.org/10.4135/9781412958806.n116

Barendset, E., & Schulte, C. (2018). «Perspectives on Computing Curricula». In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer Science Education. Perspectives on Teaching and Learning in School* (pp. 77–90).

Blackwell, A. F. (2002). «What is Programming?». *In Proceedings of Ppig 2002*, 204--218. Recuperat de http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.1345

Blakey, E., & Spence, S. (1990). «Developing Metacognition. ERIC Digest.». ERIC Clearinghouse on Information Resources, 030 Huntington Hall, Syracuse, NY 13244-2340 (free while supply lasts). Recuperat de https://eric.ed.gov/?id=ED327218

Bransford, J., National Research Council (U.S.). Committee on Developments in the Science of Learning., & National Research Council (U.S.). Committee on Learning Research and Educational Practice. (2000). «How people learn : brain, mind, experience, and school». National Academy Press.

College Board. (2017). «AP Computer Science Principles Curriculum Framework». Recuperat de https://apcentral.collegeboard.org/courses/ap-computer-science-principles?course=ap-computer-science-principles

Curzon, P., & McOwan, P. W. (2016). «The Power of Computational Thinking: Games, magic and puzzles to help you become a computational thinker».

Denning, P. J. (2017). «Remaining trouble spots with computational thinking». *Communications of the ACM*, *60*(6), 33–39. Disponible a: https://doi.org/10.1145/2998438

Dijkstra, E. W. (1988). «On the Cruelty of Really Teaching Computing Science». Austin: E.W. Dijkstra Archive. Center for American History, University of Texas. Recuperat de http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1036.PDF

Du Boulay, B. (1986). «Some Difficulties of Learning to Program». *Journal of Educational Computing Research*, *2*(1), 57–73. Disponible a: https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9

Du Boulay, B., O'Shea, T., & Monk, J. (1989). «Black box inside the glass box: Presenting computing concepts to novices». In *Studying the Novice Programmer* (pp. 431–446). Academic Press. Disponible a: https://doi.org/10.1006/ijhc.1981.0309

Dwyer, C. P., Hogan, M. J., & Stewart, I. (2014). «An integrated critical thinking framework for the 21st century». *Thinking Skills and Creativity*. Disponible a: https://doi.org/10.1016/j.tsc.2013.12.004

Ensenyament. (2007). «Decret 143/2007. Ordenació dels ensenyaments de l'educació secundària obligatòria».

Ensenyament. (2016). «Orientacions als centres per a organitzar el CFGS Desenvolupament d'aplicacions multiplataforma». Recuperat de http://xtec.gencat.cat/ca/curriculum/professionals/fp/titolsloe/infcomunicacions/

Ensenyament. (2018). «FP. Títols LOE. Informàtica i comunicacions». Recuperat de http://xtec.gencat.cat/ca/curriculum/professionals/fp/titolsloe/infcomunicacions/

Ensenyament. (2013). «Decret 260/2013. Currículum CFGS Desenvolupament d'aplicacions multiplataforma». Recuperat de http://xtec.gencat.cat/ca/curriculum/professionals/fp/titolsloe/infcomunicacions/

Gestwicki, P., & Sun, F.-S. (2008). «Teaching Design Patterns Through Computer Game Development». *Journal on Educational Resources in Computing*, *8*(1), 1–22. Disponible a: https://doi.org/10.1145/1348713.1348715

Grandell, L., Peltomäki, M., Back, R. J., & Salakoski, T. (2006). «Why complicate things? Introducing programming in high school using Python». *Conferences in Research and Practice in Information Technology Series*.

Grover, S., & Pea, R. D. (2018). «Computational Thinking: A Competency Whose Time Has Com». In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer Science Education. Perspectives on Teaching and Learning in School* (pp. 19–38).

Guo, P. (2014). «Python Is Now the Most Popular Introductory Teaching Language at Top U.s. Universities | blog@CACM | Communications of the ACM». Recuperat el 06/10/2018, de https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext

Harel, D., & Feldman, Y. A. (2004). «Algorithmics : the spirit of computing». Addison Wesley.

Hidalgo-Céspedes, J., Marín, G., & Lara-Villagrán, V. (2016). «Understanding Notional Machines through Traditional Teaching with Conceptual Contraposition and Program Memory Tracing». *CLEI Electronic Journal*, *19*(2), 2:1-2:17. Disponible a: https://doi.org/10.19153/cleiej.19.2.2

Holvikivi, J. (2010). «Conditions for successful learning of programming skills». In *Key Competencies In The Knowledge Societyn* (pp. 155–164). Disponible a: https://doi.org/10.1007/978-3-642-15378-5_15

Jenkins, T. (2002). «On the Difficulty of Learning to Program». *Language*. Disponible a: https://doi.org/10.1109/ISIT.2013.6620675

Kallia, M., & Sentance, S. (2017). «Computing Teachers' Perspectives on Threshold Concepts». In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education - WiPSCE '17* (pp. 15–24). New York, New York, USA: ACM Press. Disponible a: https://doi.org/10.1145/3137065.3137085

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). «Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching». *Educational Psychologist*, *41*(2), 75–86. Disponible a: https://doi.org/10.1207/s15326985ep4102_1

Krauss, J., & Prottsman, K. (2017). «Computational thinking and coding for every student : the teacher's getting-started guide».

Lahtinen, E., Ala-Mutka, K., Järvinen, H.-M., Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). «A study of the difficulties of novice programmers». In *Proceedings of the 10th*

annual SIGCSE conference on Innovation and technology in computer science education - ITiCSE '05 (Vol. 37, p. 14). New York, New York, USA: ACM Press. Disponible a: https://doi.org/10.1145/1067445.1067453

Lockwood, J., & Mooney, A. (2017). «Computational Thinking in Education: Where does it fit?», 1–58. Disponible a: https://doi.org/10.21585/ijcses.v2i1.26

Lye, S. Y., & Koh, J. H. L. (2014). «Review on teaching and learning of computational thinking through programming: What is next for K-12?». *Computers in Human Behavior*, *41*, 51–61. Disponible a: https://doi.org/10.1016/J.CHB.2014.09.012

Mayer, R. E. (1998). «Cognitive, metacognitive, and motivational aspects of problem solving». *Instructional Science*, *26*(1/2), 49–63. Disponible a: https://doi.org/10.1023/A:1003088013286

Meyer, J. H. F., & Land, R. (2003). «Threshold concepts and troublesome knowledge: Linkages to ways of thinking and practising within the disciplines». *Improving Student Learning – Ten Years On.*, *4*(1), 1–16. Disponible a: https://doi.org/10.1007/978-3-8348-9837-1

Muratet, M., Torguet, P., Jessel, J.-P., & Viallet, F. (2009). «Towards a Serious Game to Help Students Learn Computer Programming». *International Journal of Computer Games Technology*, *2009*, 1–12. Disponible a: https://doi.org/10.1155/2009/470590

Nystrom, R. (2014). «Game programming patterns».

Pane, J. F., Ratanamahatana, C. A., & Myers, B. A. (2001). «Studying the language and structure in non-programmers' solutions to programming problems». *International Journal of Human Computer Studies*, *54*(2), 237–264. Disponible a: https://doi.org/10.1006/ijhc.2000.0410

Papert, S., & Seymour. (1980). «Mindstorms : children, computers, and powerful ideas». Basic Books. Recuperat de https://dl.acm.org/citation.cfm?id=1095592

Pea, R. D., & Kurland, D. M. (1984). «On the cognitive effects of learning computer programming». *New Ideas in Psychology*. Disponible a: https://doi.org/10.1016/0732-118X(84)90018-7

Perkins, D. N., & Salomon, G. (1988). «Teaching for Transfer.». *Educational Leadership*, *46*(1), 22–32. Recuperat de https://eric.ed.gov/?id=EJ376242

Pintrich, P. R. (2002). «The Role of Metacognitive Knowledge in Learning, Teaching, and Assessing». *Theory Into Practice*. Taylor & Francis, Ltd. Disponible a: https://doi.org/10.2307/1477406

Pretz, J. E., Naples, A. J., & Sternberg, R. J. (2003). «Recognizing, defining, and representing problems». In *The Psychology of Problem Solving*. Disponible a: https://doi.org/10.1017/CBO9780511615771.002

Robins, A., Rountree, J., & Rountree, N. (2003). «Learning and Teaching Programming: A Review and Discussion». *Computer Science Education*, *13*(2), 137–172. Disponible a: https://doi.org/10.1076/csed.13.2.137.14200

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). «Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test». *Computers in Human Behavior*, *72*, 678–691. Disponible a: https://doi.org/10.1016/J.CHB.2016.08.047

Schulte, C., & Carsten. (2013). «Reflections on the role of programming in primary and secondary computing education». In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education on - WiPSE '13* (pp. 17–24). New York, New York, USA: ACM Press. Disponible a: https://doi.org/10.1145/2532748.2532754

Selby, C. (2013). «Computational Thinking : The Developing Definition». *ITiCSE Conference 2013*.

Servei d'Ordenació de la FPI. (2011). «Guia de programació de mòduls professionals (LOE)». Recuperat de http://xtec.gencat.cat/ca/curriculum/professionals/fp/modelcurricular/

Sorva, J., & Juha. (2013). «Notional machines and introductory programming education». *ACM Transactions on Computing Education*, *13*(2), 1–31. Disponible a: https://doi.org/10.1145/2483710.2483713

Stanovich, K. E. (2003). «The fundamental computational biases of human cognition: Heuristics that (sometimes) impair decision making and problem solving». In *The Psychology of Problem Solving*. Disponible a: https://doi.org/10.1017/CBO9780511615771.011

Tedre, M., & Denning, P. J. (2016). «The long quest for computational thinking». In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research - Koli Calling '16* (pp. 120–129). New York, New York, USA: ACM Press. Disponible a: https://doi.org/10.1145/2999541.2999542

Van Eck, R. (2006). «Digital Game-Based Learning: It's Not Just the Digital Natives Who Are Restless». *Educause Review*. Disponible a: https://doi.org/10.1145/950566.950596

Whitten, S., & Graesser, A. C. (2003). «Comprehension of text in problem solving». In J. E. Davidson & R. J. Sternberg (Eds.), *The Psychology of Problem Solving* (pp. 207–230). Cambridge: Cambridge University Press. Disponible a: https://doi.org/10.1017/CBO9780511615771.008

Wing, J. M. (2006). «Computational thinking». *Communications of the ACM*. Disponible a: https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2010). «Computational Thinking: What and Why?». Recuperat el 05/24/2018, de https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why