PROJECTE HORARIS

MEMÒRIA

Alumne: Pere Sánchez Miranda
Director: Joan Vancells Flotats
Projecte Final de Carrera 25-6-2002
Enginyeria Tècnica Informàtica de Gestió
Universitat Oberta de Catalunya

ÍNDEX

1. INTRODUCCIÓ 4
1.1. Justificació del projecte 4
1.2. Justificació de la complexitat del problema 5
1.3. Oportunitat del projecte 5
1.4. Objectius 6
1.5. Agraïments 6

2. PLA DE TREBALL 7
2.1. Calendari previst 7
2.2. Descripció de les tasques 8
2.2.1. Anàlisi de requeriments 8
2.2.1.1. Entrevistes amb caps d'estudi de diverses escoles d'EP i ESO 8
2.2.1.2. Determinar requisits obligatoris i opcionals 8
2.2.2. Creació del prototip 8
2.2.2.1. Anàlisi, disseny i documentació 8
2.2.2.2. Implementació i proves 9
2.2.2.3. Manual bàsic d'usuari 9
2.2.2.4. Lliurament del prototipus a les escoles i recollida d'opinions 9
2.2.2.5. Modificacions i correccions 9
2.2.3. Mètodes de generació dels horaris 9
2.2.3.1. Algorisme voraç per posar sessions ràpidament 9
2.2.3.2. Algorisme per forçar la col·locació de les sessions pendents 9
2.2.4. Completar manual d'usuari 10
2.2.5. Lliurament del programa a les escoles i recollida d'opinions 10
2.2.6. Modificacions i correccions 10

3. REQUERIMENTS 11
3.1. Objectius del programa 11
3.2. Requeriments 11
3.2.1. Requeriments mínims demanats pels usuaris 11
3.2.2. Requeriments opcionals 12
3.2.3. Possibles millores per futures versions del programa 12
3.2.4. Requeriments de la interfície d'usuari 12
3.2.4.1. Perfil dels usuaris 12
3.2.4.2. Descripció de les tasques 13
3.2.5. Requeriments no funcionals 14
3.2.5.1. Plataforma requerida 14
3.2.5.2. Mètode, llenguatge i entorn de desenvolupament 14
3.2.5.3. Cicle de vida 15
3.3. Diagrama general de casos d'ús 15

4. CREACIÓ DEL PROTOTIP 17
4.1. Anàlisi i disseny 17
4.1.1. Classes d'entitats 17
4.1.1.1. Persistència 17
4.1.1.2. Jerarquia de les classes d'entitats 18
4.1.2. Classes de control 21
4.1.2.1. Col·leccions d'elements 21
4.1.2.2. Control general 22
4.1.3. Classes de frontera 24
4.1.3.1. Disseny visual de les classes frontera 26

4.2. Tractament de les excepcions 27
4.2.1. Guardar i recuperar dades 27
4.2.2. Crear nous elements 27
4.2.3. Aturar un procés 27
5. GENERACIÓ D'HORARIS 28
5.1. Estudi del problema 28
5.2. Estudi dels possibles mètodes 29
5.3. Algorisme voraç 29
5.3.1. Anàlisi i disseny 29
5.3.2. Estructures de dades 32
5.3.3. Implementació 35
5.3.4. Test 36
5.4. Algorisme combinatori 36
5.4.1. Anàlisi i disseny 37
5.4.2. Millores del mètode bàsic 37
5.4.2.1. Ordenar les diferents possibilitats 37
5.4.2.2. Descartar les possibilitats menys prometedores 37
5.4.3. Implementació i proves 38
5.4.4. Càlcul del cost 40
5.4.4.1. Consideracions prèvies 40
5.4.4.2. Simplificació de les funcions 42
5.4.4.3. Càlcul 43
5.5. Contrastació de la teoria amb la pràctica 46
6. CONCLUSIONS 48

RESUM 49
BIBLIOGRAFIA 50
GLOSSARI I ABREVIACIONS 51
ANNEX 53

1. INTRODUCCIÓ

NOTACIÓ, TERMES I ABREVIACIONS UTILITZADES

Abans de començar a llegir aquest treball, és molt convenient mirar el capítol GLOSSARI I ABREVIACIONS degut a què alguns termes utilitzats a les escoles d'educació primària i secundària tenen significat diferent al que tenen en l'àmbit universitari, en concret els termes *crèdit* i *assignatura*.

El text corresponent a pseudo-codi del programa està escrit en el tipus de lletra `Courier New`. Les paraules en negreta corresponen a les instruccions elementals.

1.1. JUSTIFICACIÓ DEL PROJECTE

Després de la reforma educativa, els caps d'estudi de les escoles d'educació primària i secundària s'han trobat amb el problema d'haver de generar uns horaris molt més complicats que fins aleshores.

Alguns del problemes amb què s'han trobat són:

- Haver de fer grups barrejant alumnes de diverses aules (grups flexibles i crèdits variables) cosa que obliga a què alguns crèdits s'hagin de fer simultàniament.
- Tenir en compte les aules addicionals que es necessitaran pels grups flexibles i crèdits variables, i les aules específiques, com ara la d'informàtica, laboratoris, biblioteca, gimnàs..., que també poden ser utilitzades per altres assignatures.
- Assignatures que convé fer en hores consecutives perquè requereixen una preparació prèvia (tecnologia) o perquè cal desplaçar els alumnes a un altre recinte que es pot trobar allunyat de l'escola (educació física).
- Programar les hores lliures i de dedicació dels professor per aconseguir que sempre hi hagi algun professor disponible, i per fer reunions de cicles i etapes.
- Tenir en compte les hores no disponibles d'alguns professors amb dedicació parcial.

Haver de tenir en compte tots aquests factors fa que sigui molt complicat generar l'horari manualment, es perd una gran quantitat d'hores i a més hi ha la possibilitat de cometre errors. Per això és molt recomanable, si no imprescindible, recórrer a un programa que s'encarregui de fer-ho.

Els motius que finalment m'han fet decidir a triar aquest tipus de projecte han estat:

- És un cas pràctic que pot ser molt útil
- És un projecte prou complicat i interessant
- Intervenen diverses especialitats de la carrera: Teoria d'autòmats, Estructura de la informació, Interacció humana amb els computadors i Tècniques de desenvolupament de programari principalment

1.2. JUSTIFICACIÓ DE LA COMPLEXITAT DEL PROBLEMA

El problema al què ens enfrontem és del tipus complex.

En el pitjor dels casos és un problema exponencial donat que si tenim per posar n sessions en una setmana de 30 hores, el nombre de possibles combinacions és de 30^n. Donat el cas d'una escola de secundària mitjana que tingui 8 grups d'alumnes ens podem trobar amb unes 240 sessions per col·locar, el nombre de possibles combinacions és astronòmic: superior a $3 \cdot 10^{354}$.

1.3. OPORTUNITAT DEL PROJECTE

Un cop decidit quin projecte volia fer, el primer pas era comprovar si realment calia fer un programa o si ja hi havia en el mercat algun programa que satisfés totes les necessitats.

Per això vam decidir buscar programes generadors d'horaris a Internet i provar-los, encara que fossin versions "demo", per veure si es podien adaptar a les necessitats [1].

Després de provar alguns d'ells s'ha pogut comprovar que els què són capaços de complir tots els requisits esmentats són massa cars i complicats d'utilitzar, i els més senzills no satisfan totes les necessitats o són incapaços de generar un horari en un temps raonable quan s'imposen moltes limitacions.

També existeixen altres estudis fets [2] però fan referència a horaris universitaris i, tal com s'indica en un d'ells, amb característiques diferents. A més, el mètode estudiat és el Backtracking, que en aquest cas no es considera el més adequat donat que, de totes les combinacions possibles hi ha més que no són solució que de les altres i per tant és més difícil trobar una solució buscant les possibles combinacions, fins i tot amb podes abundants i triant correctament les primeres branques que cal estudiar.

Així, aquest projecte pot complementar aquests altres, buscant un enfocament nou i més adaptat al cas específic de les escoles de primària i secundària. A més, no es limitarà a generar l'horari, sinó que també permetrà la modificació posterior.

---

[1] A l'annex trobareu informació sobre alguns dels programes examinats, amb els inconvenients i avantatges generals.

[2] Bibliografia: projectes finals de carrera de la Universitat de Vic: nº 32 i 224

1.4. OBJECTIUS

L'objectiu és crear un programa que permeti tenir en compte tots els condicionants esmentats, que generi un horari en un temps raonable, que permeti modificar l'horari un cop generat i que sigui fàcil d'utilitzar.

Per aconseguir-ho caldrà dedicar especial atenció als següents punts:

- Interfície d'usuari gràfica per facilitar la introducció i visualització de les dades de forma ràpida i intuïtiva.
- Estructura de dades per poder introduir totes les restriccions necessàries i aconseguir la màxima eficiència a l'hora de generar l'horari.
- Mètode de generació que sigui capaç de trobar un horari en un temps acceptable.

El resultat del projecte serà el programa generador d'horaris amb exemples d'horaris reals i un informe que inclourà:

- Descripció del sistema a implementar
- Anàlisi de requeriments
- Disseny del sistema
- Consideracions d'implementació (estudi, comparació i conclusió sobre els diferents mètodes que s'hagin provat per generar els horaris)
- Manual d'usuari
- Exemples d'horaris generats partint de dades reals

Tot i que amb el projecte es pretén obtenir un programa totalment acabat, degut a la càrrega de treball que pot suposar implementar tots els detalls necessaris, es preveu que algunes de les opcions, com ara les sortides per impressora o la possibilitat de desfer les últimes accions realitzades, no es podran implementar en el temps previst pel projecte (un quatrimestre) i es deixaran indicades per implementar-les més endavant.

1.5. AGRAÏMENTS

En primer lloc el meu agraïment pel meu director de projecte, en Joan Vancells, pels seus consells i ajuda, sense els quals difícilment hauria portat a bon terme aquest projecte.

També he d'agrair la col·laboració de les escoles Col·legi Santa Maria de Blanes, Col·legi Immaculada Concepció de Lloret de Mar, Escola Freta de Calella i l'IES Blanes 3, especialment als caps d'estudi, que s'han encarregat de provar a fons el programa i han aportat valuoses idees per millorar-lo.

No em vull oblidar dels companys i consultors que m'han ajudat durant tota la carrera i m'han fet gaudir d'aquesta retrobada època d'estudiant, especialment la meva tutora, Carme Martín, que amb el seu recolzament constant m'ha animat en els moments difícils.

2. PLA DE TREBALL

2.1. CALENDARI PREVIST [3]

TASQUES	D[4]	INICI	FINAL
Anàlisi de requeriments			
Entrevistes amb caps d'estudi de diverses escoles d'EP i ESO	5	4-3-02	8-3-02
Determinar requisits obligatoris i opcionals	2	9-3-02	10-3-02
Creació del prototip			
Anàlisi, disseny i documentació			
Estructures de dades	6	11-3-02	16-3-02
Interfície d'usuari	6	17-3-02	22-3-02
Implementació i proves			
Estructures de dades	12	23-3-02	3-4-02
Interfície d'usuari	14	4-4-02	17-4-02
Test global	2	18-4-02	19-4-02
Manual bàsic d'usuari	2	20-4-02	21-4-02
Lliurar el prototip a les escoles i recollida d'opinions	5	22-4-02	26-4-02
Modificacions i correccions	2	27-4-02	28-4-02
Mètodes de generació dels horaris			
Algorisme voraç per posar sessions ràpidament			
Anàlisi, disseny i documentació	6	29-4-02	4-5-02
Implementació i proves	11	5-5-02	15-5-02
Algorisme combinatori per forçar les sessions pendents			
Anàlisi, disseny i documentació	8	16-5-02	23-5-02
Implementació i proves	13	24-5-02	5-6-02
Test global	2	6-6-02	7-6-02
Completar manual d'usuari	2	8-6-02	9-6-02
Lliurar el programa a les escoles i recollida d'opinions	5	10-6-02	14-6-02
Modificacions i correccions	2	15-6-02	16-6-02
Revisar i completar documentació. Lliurar memòria i resum	5	17-6-02	21-6-02

[3] Donat que l'únic component de l'equip de treball soc jo, no m'ha semblat necessari presentar un diagrama doncs les tasques es duran a terme seqüencialment. Pel mateix motiu, no he descomptat els dies festius perquè precisament serà quan podré dedicar més temps al projecte.

[4] Durada aproximada de cada tasca en dies

2.2. DESCRIPCIÓ DE LES TASQUES [5]

2.2.1. Anàlisi de requeriments

Primer de tot cal determinar de forma detallada les necessitats dels usuaris.

2.2.1.1. Entrevistes amb caps d'estudi de diverses escoles d'EP i ESO

S'han realitzat entrevistes amb els caps d'estudis de diverses escoles de la zona amb un guió preparat prèviament que inclou una enquesta sobre la situació actual respecte al sistema que utilitzen per fer els horaris.

2.2.1.2. Determinar requisits obligatoris i opcionals

A partir de les entrevistes esmentades al punt anterior s'han determinat els objectius del projecte i s'han extret els requisits que haurà de complir l'aplicació però donada la càrrega de crèdits del TFC s'ha hagut de triar quins s'han d'implementar i quins es poden deixar per més endavant.

També s'han incorporat a la llista les opcions que haurà d'incorporar el programa encara que no s'hagin demanat de forma explícita i les possibles millores que es preveu realitzar en futures versions del programa.

Abans de continuar amb les següents fases cal dir que, a partir dels requeriments vistos fins ara, s'ha decidit que el mètode de desenvolupament més recomanable per un projecte d'aquestes característiques és el mètode en W que es realitza en dues fases: primer es crea un prototip (bàsicament tot el relacionat amb la interfície d'usuari) per tal que es pugui comprovar si el programa s'adaptarà a les necessitats dels usuaris, i després s'implementen els processos interns del programa que ja serà definitiu.

2.2.2. Creació del prototip

Per la creació, tant del prototip com de la resta del programa, s'ha decidit utilitzar la programació orientada a objectes amb l'entorn de treball Visual C++.

En aquesta fase es definiran i programaran les estructures de dades bàsiques per gestionar la informació i la interfície d'usuari.

2.2.2.1. Anàlisi, disseny i documentació

A partir dels requisits del punt 1 es dissenyaran les classes i el sistema per guardar les dades intentant aprofitar tot el que incorpora l'entorn de

[5] Abans de decidir fer aquest projecte es va fer un estudi d'oportunitat basat en les opinions dels caps d'estudi de l'escola Santa Maria de Blanes que havien provat diversos programes sense que cap d'ells satisfés les seves necessitats.

desenvolupament triat. També s'haurà de dissenyar la interfície d'usuari: visualització de les llistes d'elements i horaris, i els diàlegs per introduir les dades i visualitzar la informació necessària, per exemple en els processos llargs com el de generació de l'horari.

A mida que es realitzi l'anàlisi i el disseny, s'anirà generant la documentació necessària.

2.2.2.2. Implementació i proves

S'implementaran les classes i la interfície d'usuari a partir del disseny del punt anterior. També s'aniran fent proves unitàries a mida que es vagi fent la implementació. Al final es farà la prova de conjunt.

2.2.2.3. Manual bàsic d'usuari

Es prepararà un manual amb les opcions implementades fins el moment tot i que en el moment de lliurar el prototip als usuaris es farà una explicació del funcionament.

2.2.2.4. Lliurament del prototipus a les escoles i recollida d'opinions

Es lliurarà als usuaris el prototip del programa i el manual resumit. Es farà una explicació de les possibilitats i es deixarà un qüestionari per omplir un cop hagin l'hagin provat.

2.2.2.5. Modificacions i correccions

Es realitzaran les modificacions i correccions oportunes a partir de les opinions recollides a l'apartat anterior.

2.2.3. Mètodes de generació dels horaris

En aquesta fase s'estudiaran i provaran diferents mètodes per generar els horaris, i es modificaran les classes creades anteriorment per afegir els atributs i mètodes necessaris. També s'afegiran al programa les opcions i diàlegs relacionats amb la generació automàtica i modificació dels horaris.

2.2.3.1. Algorisme voraç per posar sessions ràpidament

El primer mètode que s'estudiarà és un algorisme voraç per intentar col·locar de forma ràpida el màxim nombre possible de sessions. Aquest mètode difícilment aconseguirà posar totes les sessions però amb les poques que no s'hagin pogut posar s'utilitzarà el mètode que veurem a continuació.

2.2.3.2. Algorisme per forçar la col·locació de les sessions pendents

El segon mètode que s'estudiarà, permetrà posar una sessió en una hora determinada desplaçant si cal altres sessions que siguin incompatibles amb

aquesta (per exemple, que tinguin el mateix professor). Aquest mètode permetrà posar manualment i de forma automàtica, sessions que encara no estiguin posades, i moure sessions d'un lloc a un altre de l'horari.

2.2.4. Completar manual d'usuari

Abans de lliurar el programa definitiu es completarà el manual afegint explicacions sobre la generació i modificació dels horaris i alguns exemples.

2.2.5. Lliurament del programa a les escoles i recollida d'opinions

Es lliurarà als usuaris el programa i el manual definitiu. Es farà una explicació de les noves opcions i es deixarà un qüestionari per omplir un cop hagin l'hagin provat.

2.2.6. Modificacions i correccions

Es realitzaran les modificacions i correccions oportunes a partir de les opinions recollides a l'apartat anterior.

3. REQUERIMENTS

3.1. OBJECTIUS DEL PROGRAMA

De les entrevistes realitzades amb els caps d'estudi de diverses escoles de primària i secundària es desprèn que el què necessiten és un programa que els faciliti la feina de generar l'horari de cada curs o de cada trimestre i evitar els possibles errors.

Per això el programa haurà de complir:

- Que sigui fàcil d'utilitzar
- Que permeti introduir i modificar totes les dades i restriccions necessàries
  - Grups d'alumnes, aules, professors i crèdits (assignatures)
  - Grups flexibles, crèdits variables i sessions en hores consecutives
  - Reunions de professors dins l'horari escolar
  - Hores prohibides (impossibles) per grups, aules, professors i crèdits
- Que sigui capaç de generar els horaris en un temps raonable
- Que permeti modificar manualment els horaris generats

3.2. REQUERIMENTS

Els requeriments s'han extret principalment dels objectius anteriors i de les entrevistes amb els caps d'estudi, però també s'han agafat algunes idees dels programes provats en la fase prèvia d'estudi d'oportunitat del projecte.

Degut a la càrrega de treball del projecte s'ha decidit implementar només una part de tots els requeriments. A continuació es concreten quins es consideren requeriments mínims i s'hauran d'implementar obligatòriament, quins són opcionals i s'implementaran si dona temps, i quins es consideren millores a implementar en futures versions del programa.

També s'han especificat els requeriments relacionats amb la interfície d'usuari i, per últim, els requeriments no funcionals, relacionats amb el maquinari, programari i desenvolupament del projecte.

3.2.1. Requeriments mínims demanats pels usuaris

- S'han de poder indicar les hores impossibles dels elements i crèdits
- S'han de poder posar, treure, fixar i moure sessions manualment
- S'ha de poder indicar quins crèdits s'han de fer simultàniament
- S'ha de poder indicar si les sessions d'un crèdit s'han de fer en hores consecutives

- S'ha de poder programar reunions dins l'horari escolar
- No ha de posar més d'una sessió el mateix dia al generar l'horari automàticament
- Però s'ha de poder posar més d'una sessió el mateix dia si es fa manualment
- Evitar que es posi una sessió en hores ocupades per crèdits amb elements comuns
- Visualitzar els elements ordenats alfabèticament (cursos, aules i professors)
- Poder inserir un crèdit a qualsevol lloc de la llista de crèdits
- Número de dies i hores variable per poder programar activitats extra-escolars

3.2.2. Requeriments opcionals

- Poder canviar l'ordre dels crèdits a la llista
- Separar el màxim possible les sessions d'un mateix crèdit
- No posar el mateix professor dues sessions seguides al mateix curs
- Poder modificar l'horari un cop generat
- Imprimir les llistes i els horaris
- Poder veure només els crèdits d'un element determinat
- Avisar si a un element se li assignen més hores de les que té disponibles
- Avisar si a un crèdit se li assignen més sessions que dies disponibles
- Llista d'àrees per assignar cada crèdit a una àrea i saber quants crèdits de cada àrea fa cada grup d'alumnes
- Llistats de crèdits per professor i per grup d'alumnes

3.2.3. Possibles millores per futures versions del programa

- Poder canviar l'amplada de les columnes de llistes i horaris
- Opcions per desfer i refer les últimes accions realitzades
- Presentació preliminar abans d'imprimir
- Seleccionar avisos visuals o sonors
- Editar elements directament sobre la vista en lloc de fer sortir un diàleg
- Menú Ajuda, ajuda amb tecla F1 i ajuda als diàlegs (?)
- Configuració del programa (colors, tipus i mida de les lletres, canvi d'idioma...)
- Guardar cada X minuts i recuperar si el programa no s'ha tancat correctament
- Protecció anticòpia

3.2.4. Requeriments de la interfície d'usuari

3.2.4.1. Perfil dels usuaris

Els usuaris del programa seran els caps d'estudi de diverses escoles d'educació primària i secundària.

Tots estan acostumats a utilitzar aplicacions ofimàtiques (processadors de text i fulls de càlcul), i altres aplicacions relacionades amb la gestió del centre (gestió dels alumnes, personal docent i notes).

Coneixen bé l'entorn Windows, no els costa aprendre a utilitzar programes relacionats amb la seva feina i estan ben predisposats a utilitzar qualsevol programa que els faciliti la seva tasca i els estalviï temps.

Han de ser capaços d'utilitzar el programa amb una explicació d'una hora i resoldre qualsevol dubte consultant el manual.

3.2.4.2. Descripció de les tasques

A més de les tasques que es descriuen a continuació, també hi haurà les tasques habituals de crear un nou horari, obrir un horari existent i guardar l'horari amb el què s'està treballant. Aquestes opcions no es descriuen donat que el seu funcionament és prou conegut pels usuaris del sistema operatiu Windows.

- **Configurar setmana:** cal introduir quants dies té la setmana lectiva, quantes hores tindrà cada dia i entre quines hores hi ha un descans. També es poden modificar els noms dels dies i les hores que es vol que surtin quan s'imprimeixin els horaris.
- **Afegir un element (professor, aula o grup d'alumnes):** cal indicar que es vol afegir un element i introduir el nom de l'element desitjat en un diàleg. El programa haurà d'avisar si l'element ja existeix. Es podrà cancel·lar la operació.
- **Modificar un element:** s'haurà de seleccionar l'element, indicar que es vol modificar i canviar el nom que tenia. Igual que quan s'introdueix un de nou, avisarà si l'element ja existeix i es podrà cancel·lar la operació.
- **Esborrar un element:** primer s'ha de seleccionar l'element i indicar que es vol esborrar. No es podrà esborrar un element que tingui sessions posades. Si no té sessions posades però pertany a algun crèdit, s'avisarà abans d'esborrar-lo per donar la possibilitat de cancel·lar.
- **Indicar hores prohibides:** es podrà indicar que una hora és prohibida sempre que no hi hagi cap sessió posada en aquesta hora. També es podrà anular la prohibició d'una hora. Primer cal indicar que volem canviar la prohibició de les hores i després seleccionar les hores que volem canviar el seu estat.
- **Indicar sessions fixades:** es podrà indicar que no volem que es mogui una sessió i també anular la fixació seleccionant quines sessions volem. Primer cal indicar que volem canviar la fixació de les hores i després seleccionar les hores que volem canviar el seu estat.
- **Afegir crèdits:** s'haurà d'introduir el nom del crèdit, el professors que el fa i l' aula. També s'haurà de seleccionar els grups d'alumnes que el fan, el nombre de sessions que té el crèdit i quantes hores consecutives durarà cada sessió. Els grups d'alumnes i les aules no serà obligatori. D'aquesta manera es podran programar reunions de professors sense haver de crear diàlegs diferents.

- **Modificar crèdit:** es podrà modificar un crèdit sempre que no tingui sessions ja col·locades. Si en tingués, abans s'haurien de treure.
- **Esborrar crèdit:** no haurà de tenir cap sessió posada per poder-lo esborrar.
- **Sol·licitar la generació de l'horari:** un cop seleccionada la opció, el programa informarà de l'evolució (nombre de sessions que falten per posar). També hi haurà la possibilitat d'aturar el procés en qualsevol moment.
- **Modificar l'horari:** l'usuari podrà indicar quina sessió vol moure i a quin lloc. El programa informarà de quines sessions s'hauran de moure i donarà les possibilitats "Acceptar", "Continuar buscant altres solucions" o "Cancel·lar".
- **Llistar:** l'usuari haurà de seleccionar què vol imprimir: quina (o quines) llistes (grups d'alumnes, aules, professors o crèdits), quins horaris (tots els de les llistes que se seleccionin) o l'horari de l'element que estigui seleccionat.

3.2.5. Requeriments no funcionals

3.2.5.1. Plataforma requerida

Per aconseguir que el programa es pugui utilitzar a la major part de les escoles, es desenvoluparà per una plataforma de tipus PC-compatible amb sistema operatiu Windows 95. Això també farà que el programa resulti semblant a altres aplicacions conegudes i per tant més amigable i fàcil d'aprendre a utilitzar.

Donades les característiques del programa, no es necessitaran grans requeriments quant a maquinari: el programa podrà funcionar en qualsevol ordinador que suporti Windows 95, per tant hi haurà prou amb un 486-100 MHz i 8 Mb de memòria RAM. Evidentment, quanta més velocitat, més ràpida serà la generació dels horaris i el moviment de sessions.

3.2.5.2. Mètode, llenguatge i entorn de desenvolupament

S'ha decidit desenvolupar el projecte amb la metodologia orientada a objectes donat que facilita la creació, modificació i reutilització de codi. Això ha portat a triar entre JAVA o C++ que són els llenguatges més coneguts. Donat que el programa haurà de ser el màxim d'eficient s'ha optat per triar el llenguatge C++ en lloc de JAVA doncs, encara que aquest permet transportar fàcilment el programa a altres plataformes, és més lent.

L'entorn triat per desenvolupar el programa serà el Visual C++ 5.0. Aquest entorn porta incorporades moltes classes específiques per treballar amb el sistema Windows i facilita la creació de la interfície d'usuari gràcies al disseny visual dels diàlegs i altres elements de la interfície gràfica (GUI).

3.2.5.3. Cicle de vida

El cicle de vida més convenient en aquest cas és el mètode en W que, després de la recollida d'informació i anàlisi de requeriments, es realitza en dues etapes: en la primera es crearà un prototip amb el què es podrà comprovar si el programa s'adapta a les necessitats dels usuaris, i en la segona s'implementaran els processos interns del programa per generar els horaris automàticament.

En cada etapa se seguiran les fases del cicle de vida clàssic: anàlisi, disseny, implementació i proves. Donades les característiques i la durada màxima del projecte s'ha decidit unir les fases d'anàlisi i disseny, i les fases d'implementació i proves. D'aquesta manera se simplifica i accelera el procés de creació.

3.3. DIAGRAMA GENERAL DE CASOS D'ÚS

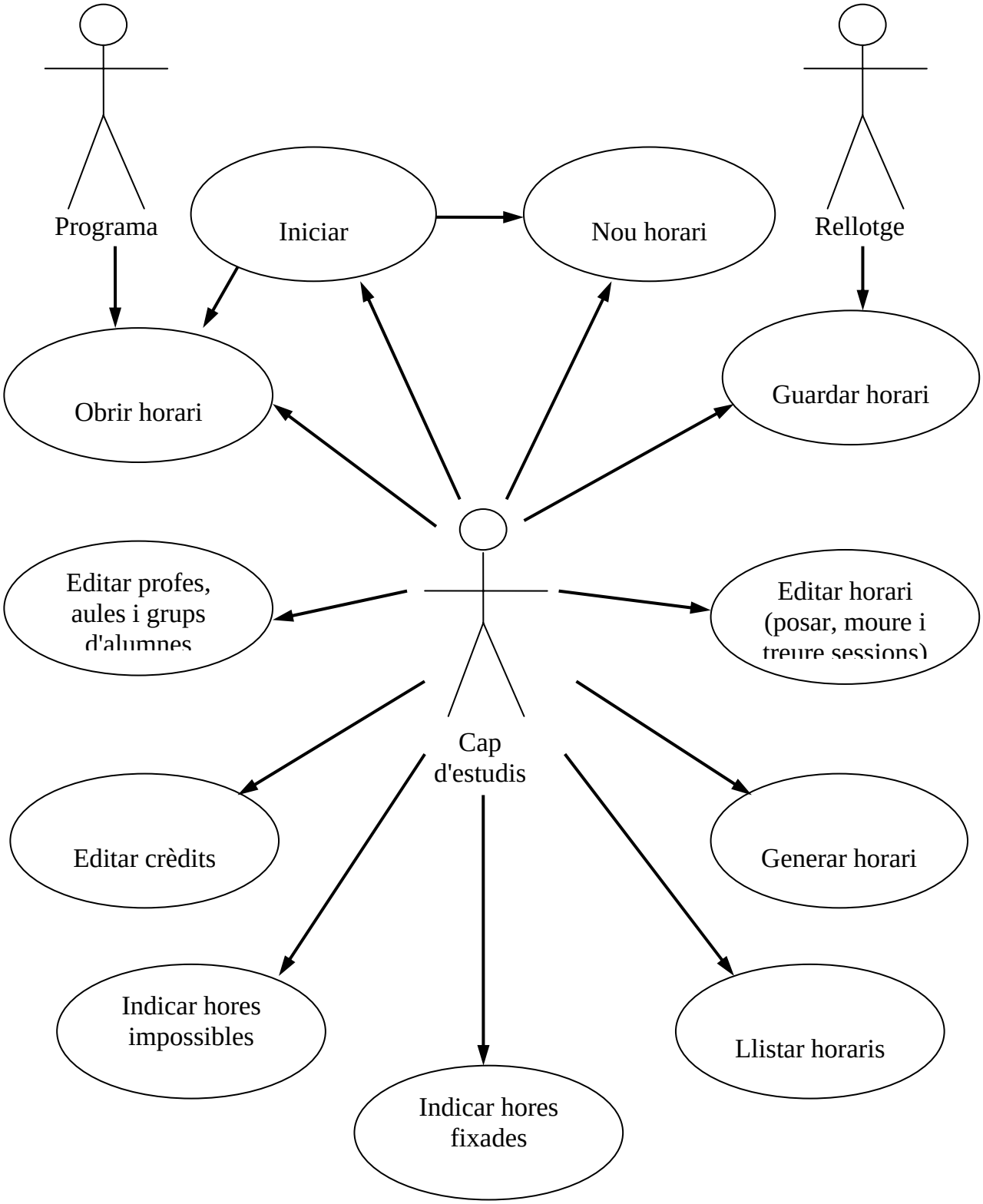
L'únic actor real que treballarà amb el programari és el cap d'estudis però s'han afegit els actors *Rellotge* i *Programa* perquè hi ha dues tasques que es realitzen automàticament sense intervenció de l'usuari:

- **Gravar automàticament** l'horari cada cert temps en un arxiu temporal per poder recuperar el treball fet en cas que es produeixi un error que impedeixi la sortida normal del programa (d'això s'encarrega el *Rellotge*).

- **Carregar automàticament** la còpia temporal de l'horari amb què s'estava treballant en cas que es detecti que la última vegada no es va tancar correctament el programa (d'això s'encarrega el *Programa* en el moment en què es posa en marxa).

Al següent esquema es poden veure els casos d'ús generals.

Diagrama de casos d'ús generals



4. CREACIÓ DEL PROTOTIP

4.1. ANÀLISI I DISSENY

Per facilitar els canvis posteriors s'ha decidit distingir entre classes d'entitats, classes de control i classes de frontera. D'aquesta manera, els canvis en les classes d'un tipus (per exemple si es decideix canviar la forma de visualitzar o introduir les dades) afectaran les mínimament les classes dels altres tipus.

4.1.1. Classes d'entitats

Corresponen als objectes que contenen la informació amb què es treballarà. Aquests objectes han de ser persistents i per tant haurem d'implementar mètodes per guardar-los i recuperar-los. A partir de l'anàlisi de requeriments es pot veure que es necessitaran les següents entitats:

- **Setmana:** ens servirà principalment per poder configurar el nombre de dies que té la setmana i el nombre d'hores dels dies. Això permetrà programar activitats extraescolars, per exemple els dissabtes o després de l'horari escolar.
- **Professor, Aula i Grup:** totes aquestes entitats tenen les mateixes necessitats i per tant només haurem de crear una única classe. Cal que se li pugui donar un nom, tenir en compte en quines hores no s'han de posar sessions i saber quin crèdit es fa en cada hora.
- **Crèdit:** contindrà el nom de l'assignatura, el professor que la imparteix, l'aula on es fa, el grup d'alumnes a què afecta, el nombre de sessions setmanals que s'han de fer i quantes hores consecutives s'han de fer a les sessions. També haurà de tenir en compte en quines hores s'ha posat cada sessió i en quines hores no s'han de posar sessions. Per poder complir els requisits de què s'ha de poder indicar quins crèdits s'han de fer simultàniament i tenir en compte els grups flexibles es crearà una classe que permetrà contenir més d'un crèdit.

4.1.1.1. Persistència

Per poder guardar i recuperar les entitats que conformen l'horari s'ha decidit utilitzar la "serialització" (fitxers clàssics) en lloc d'utilitzar una base de dades. Els motius pels què s'ha triat aquest sistema són els següents:

- No val la pena la complicació d'accedir a una base de dades tenint en compte les poques taules que hi ha i les senzilles relacions que hi ha entre elles. A més, tampoc no cal fer consultes complexes.

- Implementar en una base de dades el sistema necessari per guardar els crèdits simultanis i els crèdits pels grups flexibles és molt més complicat que crear una classe específica per aquesta finalitat.
- El sistema de serialitació és molt fàcil d'implementar si derivem les nostres classes de la classe CObject de les MFC.

4.1.1.2. Jerarquia de les classes d'entitats [6]

A més de les classes especificades al punt anterior, s'hauran d'afegir altres classes necessàries per complir els requeriment. Per exemple, haurem de tenir elements ordenats i per tant ens cal un tipus d'objecte que permeti fer comparacions.

També necessitarem algunes classes auxiliars per facilitar algunes tasques, com per exemple la classe Hora i les seves derivades, que serviran per contenir informació sobre cada hora de l'horari: si està prohibida, ocupada, fixada, quin crèdit hi ha, etc.

El cas dels crèdits es mereix una anàlisi més detallada. En principi sembla fàcil, doncs la definició de crèdit ens diu que ha de contenir el nom de l'assignatura, el professor que la imparteix, l'aula i el grup d'alumnes que l'ha de fer. Però també hem de tenir en compte els crèdits simultanis, crèdits variables, grups flexibles i reunions de professors.

El primer problema és que en els crèdits variables i en els grups flexibles hi poden participar alumnes de diferents grups. Per exemple, al crèdit variable de "Matemàtiques de reforç" hi ha alumnes dels grups 3rA i 3rB d'ESO. Per tant, a cada crèdit li hem de poder assignar més d'un grup d'alumnes.

El següent problema és que hem de poder agrupar crèdits que s'han de fer al mateix temps, els crèdits simultanis, que normalment són grups de crèdits variables.

Per últim cal considerar les reunions de professors, però aquest cas és fàcil de resoldre si considerem les reunions com crèdits en què no cal assignar cap grup d'alumnes ni cap aula.

Per aconseguir posar aquestes restriccions hi ha la possibilitat de crear una classe que, a més dels atributs esmentats anteriorment, inclogui un atribut indicant un grup de crèdits, de forma que els crèdits que s'hagin de fer simultàniament tinguin el mateix número de grup. El problema és que s'han d'introduir moltes dades repetides i alguns dels camps haurien de contenir els mateixos valors en tots els registres. Veiem un exemple: volem que es facin simultàniament els crèdits "Mates de reforç", "Mates d'ampliació" i "Mates empresarials" que ha de tenir 3 sessions d'una hora cada una i els poden fer els alumnes de 4rtA i 4rtB d'ESO.

---

[6] Les classes que pertanyen a les MFC's s'han dibuixat en color gris.
La majoria són derivades de *CObject* per poder implementar la serialització

Haurem d'introduir els següents registres i camps, posant els mateixos valors al nombre de sessions i d'hores i al número d'agrupació de crèdits.

NºGrup	Assignatura	Professor	Aula	Grup Alum.	Sessions	Hores / S.
1	Mates Ref.	Anna	4A	4rtA ESO	3	1
1	Mates Ref.	Anna	4A	4rtB ESO	3	1
1	Mates Ampl.	Bernat	4B	4rtA ESO	3	1
1	Mates Ampl.	Bernat	4B	4rtB ESO	3	1
1	Mates Empr.	Carme	Variable1	4rtA ESO	3	1
1	Mates Empr.	Carme	Variable1	4rtB ESO	3	1

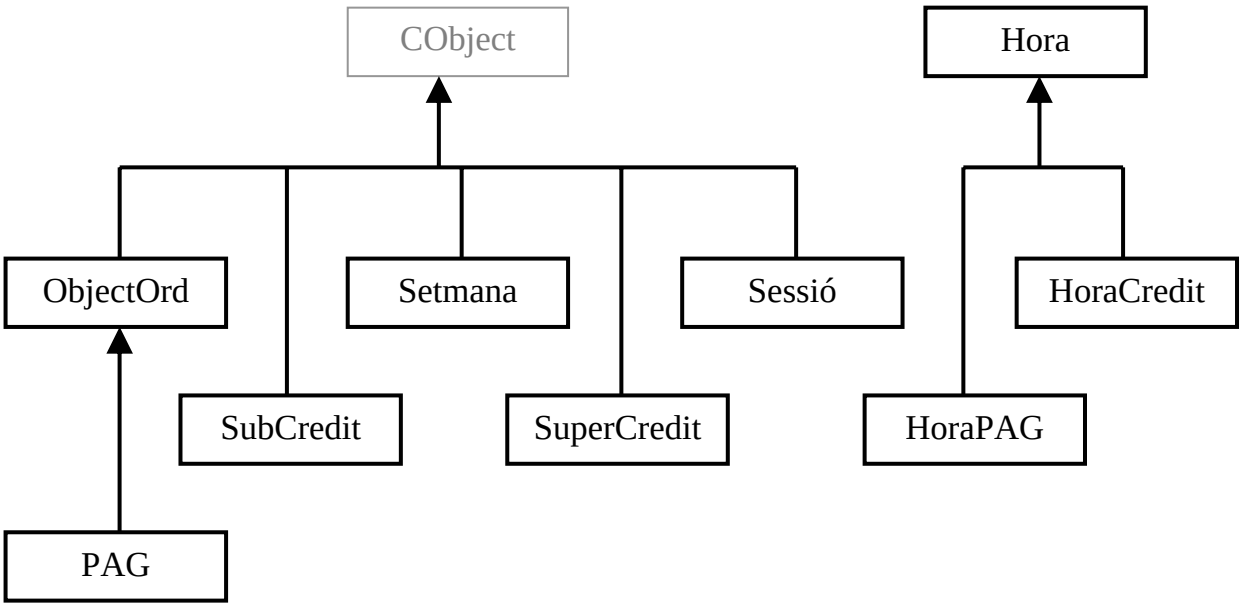
La solució que s'ha trobat és crear la classe *Supercredit* que conté el nombre de sessions, el nombre d'hores per sessió, una llista amb els cursos que el poden fer i una llista de *Subcredit*s que serà una altra classe que contindrà el nom de l'assignatura, el professor i l'aula.

El què quedarà, representat en forma de graella, serà això:

Assignatura	Professor	Aula	Grup Alum.	Sessions	Hores / S.
Mates Ref.	Anna	4A	4rtA ESO 4rtB ESO	3	1
Mates Ampl.	Bernat	4B			
Mates Empr.	Carme	Variable1			

També es pot donar el cas que es vulgui posar diferent nombre d'hores a cada sessió. Per exemple, un crèdit que tingui 3 hores setmanals i es vulgui fer una sessió d'una hora i una sessió de dues hores consecutives. S'ha decidit no tractar aquest cas especial però el programa permetrà que l'usuari posi de forma manual més d'una sessió el mateix dia.

Diagrama de la jerarquia de les classes d'entitats [7]



[7] PAG: Profes, Aules i Grups d'alumnes

4.1.2. Classes de control

Són aquelles que modelen objectes interns. Aquests objectes són temporals però per comoditat a l'hora de guardar i recuperar els objectes que controlen, en alguns casos s'ha decidit implementar també mètodes de serialització.

4.1.2.1. Col·leccions d'elements

Per mantenir en memòria les col·leccions dels diferents elements s'ha decidit aprofitar les classes de les MFC [8] per la versatilitat que ofereixen i, sobre tot, per la característica de què no cal dimensionar-les al principi sinó que poden créixer de forma dinàmica, cosa molt útil quan no se sap quina capacitat es necessita. De totes formes s'han hagut de derivar classes específiques per a implementar algunes funcionalitats necessàries.

El programa no necessita molta velocitat a l'hora d'afegir, modificar o eliminar elements, donat que es realitza de forma interactiva amb l'usuari, però sí que requerirà una gran velocitat per accedir als elements en el moment de generar l'horari. Per tant s'implementaran classes derivades de *CObArray* per crear les llistes dels elements. Aquesta classe proporciona l'accés més ràpid als elements encara que pot ser força lenta quan afegim i eliminem elements al principi o al mig, si tenim molts elements, cosa que no és el cas (com a màxim es preveu tenir uns 50 elements de tipus Professor, Aula o Grup d'alumnes i uns 500 de tipus Crèdit).

- **Vector:** afegeix la possibilitat d'esborrar automàticament els elements. *CObArray* guarda punters als objectes i només esborra els punters però no els objectes en sí.
- **Llista en vector:** respecte de l'anterior, fa que els punters als objectes es mantinguin consecutius quan s'esborra un objecte, sense deixar espais buits.
- **Llista en vector ordenada:** permet mantenir ordenats els objectes que conté. Per aconseguir-ho haurem de tenir una classe d'objectes amb mètodes per determinar quan un element és major, menor o igual que un altre.

En alguns casos però, sí que convé que les operacions d'afegir o treure elements siguin ràpides. Per exemple quan cal afegir i treure d'una llista els moviments que es van fent i desfent per moure sessions i aconseguir deixar espai per posar un sessió en un lloc determinat. Aquesta llista ha de funcionar com una pila, posant l'últim moviment realitzat a dalt de tot i traient el moviment de dalt per desfer els moviments. En aquest cas, el sistema més ràpid és utilitzar una llista encadenada amb un punter a l'últim element que hem afegit.

[8] Microsoft Foundation Classes incorporades al Visual C++

Dintre de les classes MFC hi ha la classe *CObList* que permet fer totes aquestes operacions però, igual que amb el cas dels vectors, s'ha decidit crear una classe que hereta d'aquesta i afegeix algunes possibilitats més.

- **Llista:** afegeix la possibilitat d'esborrar automàticament els elements. *CObList* guarda punters als objectes i només esborra els punters però no els objectes en sí.

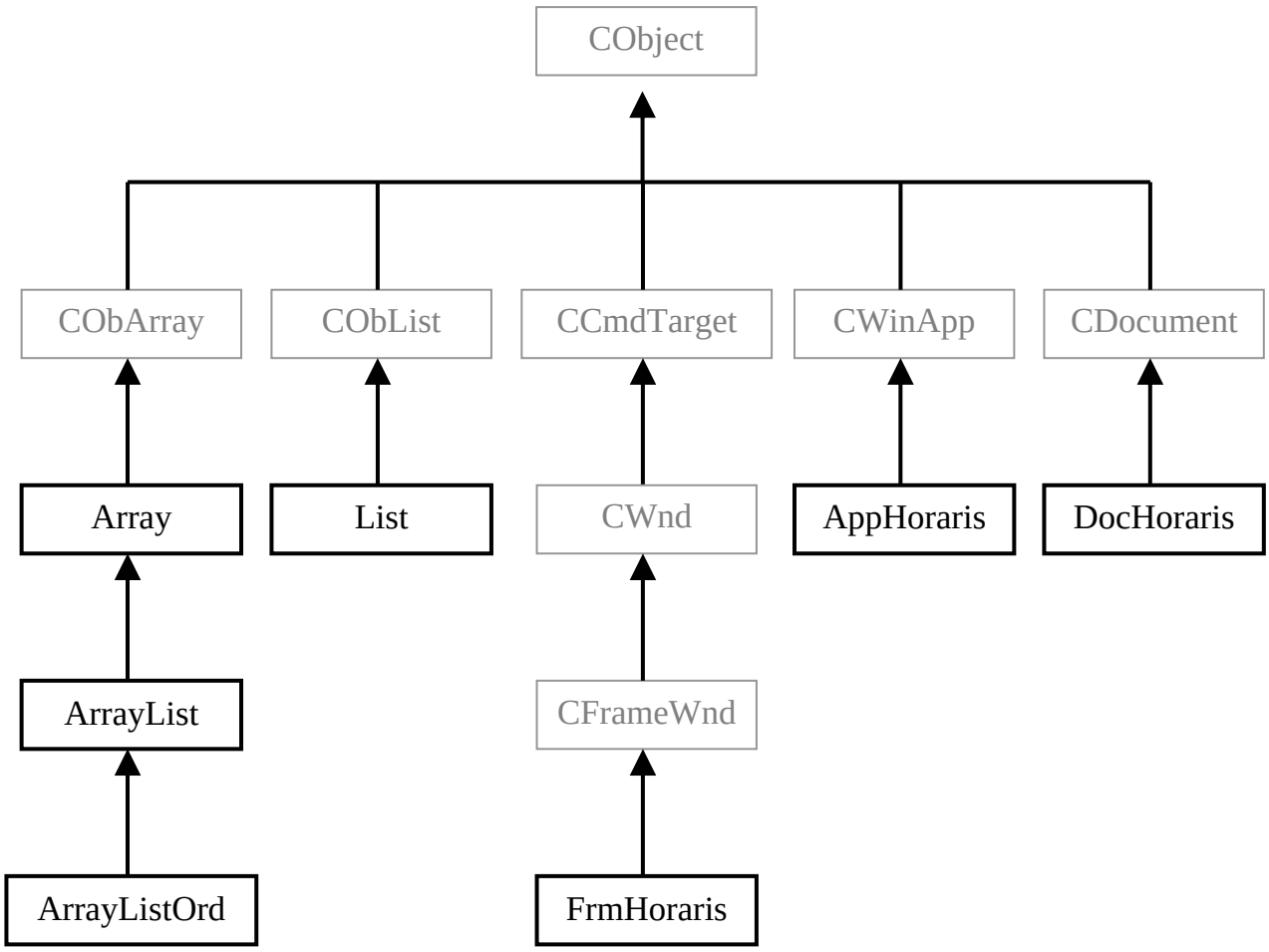
4.1.2.2. Control general

Les entitats que es descriuen en aquest apartat s'encarreguen de controlar el què volem visualitzar i la forma d'actuar sobre un horari. També s'encarreguen de modificar, guardar i recuperar les dades.

S'ha decidit treballar amb el model Document-Vista que implementa Microsoft a les seves MFC. En aquest model, es considera que podem tenir un o més documents i diferents vistes de cada document (diferents formes de veure'l). Això coincideix amb el què necessitem: tenim un document (el projecte d'horari) i diferents formes de veure la informació que conté: llistes i horaris dels elements.

- **Aplicació:** deriva de *CWinApp* i representa l'objecte aplicació que s'encarrega de posar en marxa el programa i inicialitzar tot el necessari.
- **Document de l'horari:** deriva de *CDocument* i agrupa dins seu les entitats esmentades anteriorment i els mètodes per obrir, guardar, crear i modificar un document d'horaris i tots els seus elements.
- **Finestra principal:** deriva de CFrameWnd i agrupa dins seu el menú, barres d'estat i barra d'eines. S'encarrega del control del menú, dels botons de la barra d'eines i la visualització de les diferents vistes. Encara que a primera vista pot semblar més una classe de frontera, donat les tasques de control que ha de realitzar, és clarament una classe de control.

Diagrama de la jerarquia de les classes de control



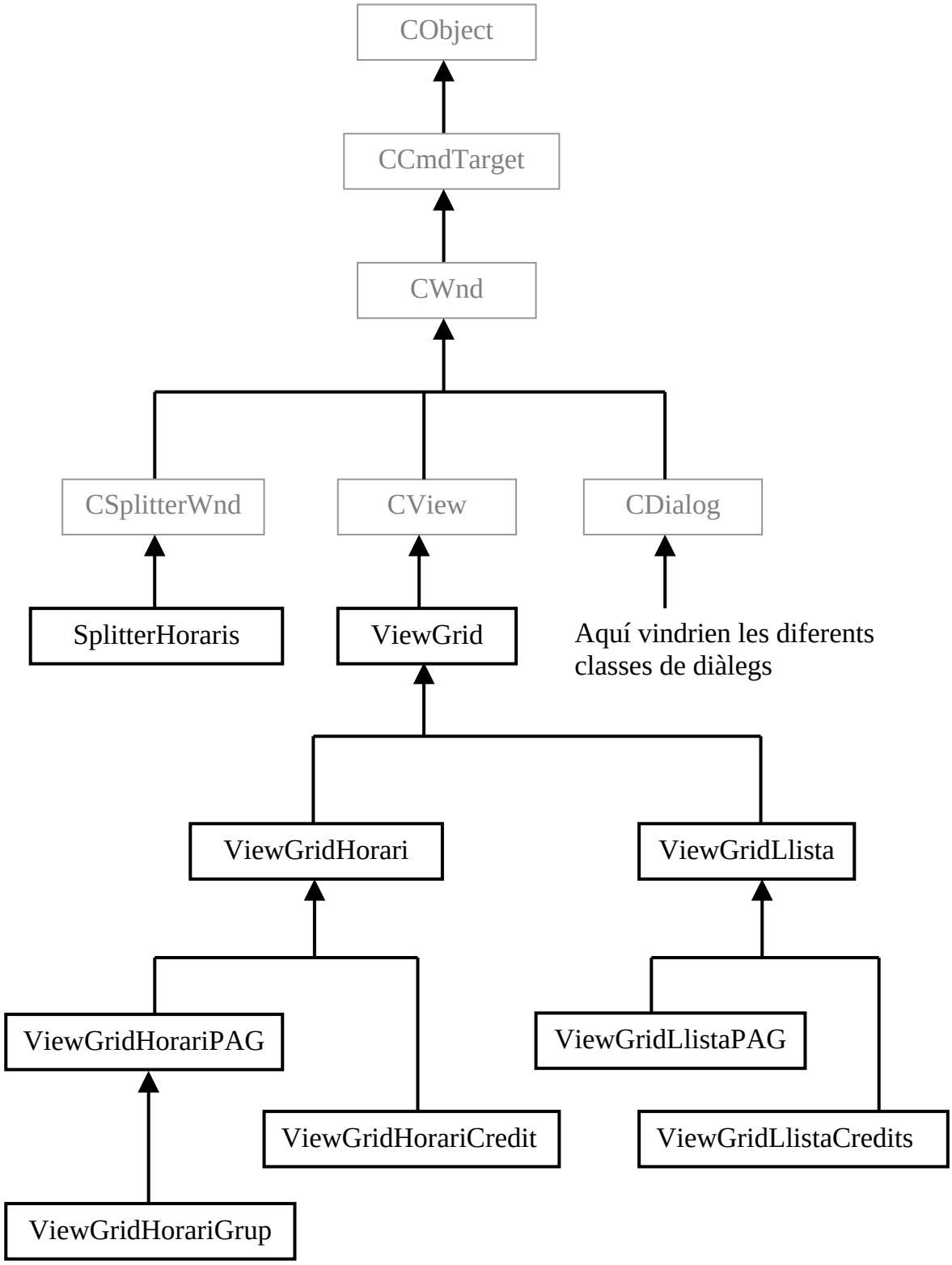
4.1.3. Classes de frontera

Aquestes classes representen els objectes de la interfície d'usuari.

S'ha de pensar en una interfície d'usuari que permeti aprendre a utilitzar-lo el més ràpidament possible, que sigui fàcil d'utilitzar i que no faci perdre el temps als usuaris. Per això s'ha pensat en una interfície semblant a altres programes de Windows, posant a disposició dels usuaris el màxim d'informació possible en pantalla per evitar que hagin d'anar canviant de finestres i amb diverses formes d'actuar segons les seves preferències i el seu nivell de coneixement de Windows (barra d'eines, menús, tecles acceleradores...).

- **Finestra dividida:** deriva de *CSplitterWnd*. Consisteix en una finestra dividida en dos: una part per veure i editar les llistes i l'altra per veure i editar l'horari de l'element seleccionat a la llista.
- **Vistes de graella:** deriven de *CView* i serveixen per visualitzar informació en forma de graella. D'aquesta se'n deriven dues més:
  - Vista de graella per horaris: permet seleccionar arrossegant el ratolí. D'aquesta s'en derivaran altres per elements específics.
  - Vista de graella per llista: permet tenir seleccionada una de les files (un dels elements). D'aquesta se'n derivaran altres per elements específics.
- **Diàlegs:** deriven de *CDialog* i serveixen per introduir i modificar les dades.

Diagrama de la jerarquia de les classes de frontera



4.1.3.1. Disseny visual de les classes frontera

El disseny de les diferents finestres i diàlegs, s'ha fet directament amb l'editor de recursos de Visual C++.

A la figura següent es pot veure la finestra principal del programa [9]. A la part esquerra es on es poden veure les llistes i a la part dreta l'horari de l'element seleccionat a la llista.

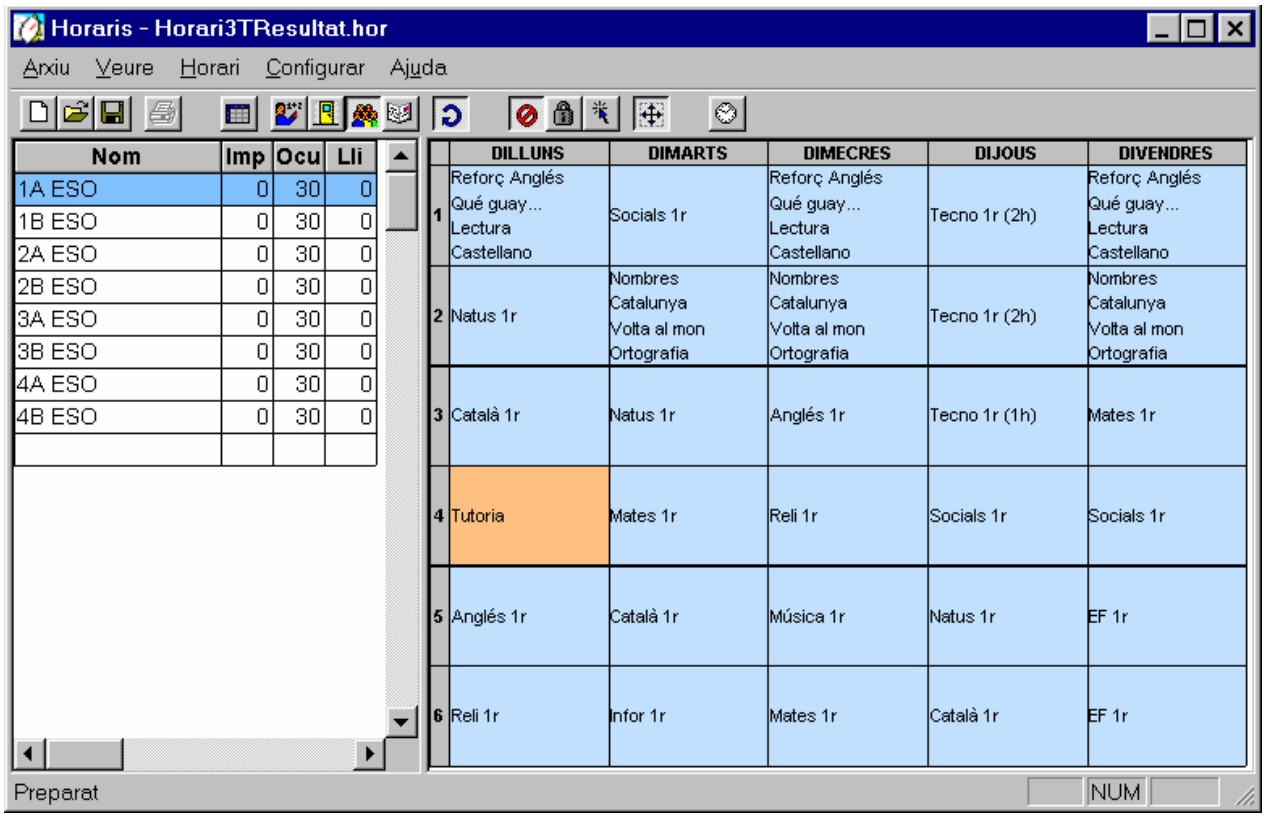


Figura 1

A la barra d'eines s'han posat les opcions que més s'utilitzaran i en l'ordre que se seguirà normalment en la creació d'un horari.

S'ha intentat crear icones el més significatives possible i s'han triat colors ben diferenciats, amb un significat clar per la majoria de la gent (per exemple, vermell per indicar prohibit) i suaus perquè resultin més relaxants i no amaguin el text.

[9] La resta de finestres es poden veure al manual d'usuari. No s'han posat en aquest document per evitar que tingués una mida excessiva.

4.2. TRACTAMENT DE LES EXCEPCIONS

Les principals excepcions es poden produir al guardar i recuperar les dades, i quan es creen nous elements, però a part d'aquestes s'ha creat una excepció per quan l'usuari vulgui aturar un procés llarg com la generació de l'horari o el moviment d'una sessió a un altre lloc.

4.2.1. Guardar i recuperar dades

Aquestes excepcions es tractaran dins el mètode de serialització de l'entitat document, que és l'encarregada de guardar i recuperar les dades.

Les excepcions que es poden produir són:

- **CMemoryException** per falta de memòria.
- **CArchiveException** representa diferents condicions d'error en la serialització (intentar carregar un tipus d'objecte quan s'espera un altre, fi de fitxer...).
- **CFileException** representa els diferents tipus d'error que es poden produir amb els fitxers (no s'ha pogut obrir el fitxer, error de disc...).

S'ha decidit deixar que l'excepció sigui tractada per la classe *CDocument* de les MFC. El què fan és mostrar un missatge d'error indicant el motiu i inicialitzar el document.

4.2.2. Crear nous elements

Aquestes excepcions es tractaran en el moment de crear nous elements, o sigui, en els mètodes que permeten afegir nous elements.

La única excepció que es pot produir és:

- **CMemoryException:** la causa és la falta de memòria i l'únic que es pot fer és avisar a l'usuari sobre aquest fet.

4.2.3. Aturar un procés

S'ha creat una nova classe anomenada **Excepció** que servirà aturar els processos de llarga durada. Aquesta excepció la generarà el programa quan detecti que l'usuari ha decidit aturar un procés d'aquest tipus.

L'usuari pot aturar aquests processos amb el botó adequat dels diàlegs que apareixen quan sol·licita les opcions per generar horari i per moure una sessió a un altre lloc.

5. GENERACIÓ D'HORARIS

5.1. ESTUDI DEL PROBLEMA

Aquesta és, sens dubte, la part més crítica del projecte.

Tothom pot veure que la feina de posar totes les sessions evitant les coincidències entre els elements que hi intervenen és d'una gran dificultat, però per fer-nos una idea més exacte, fem uns quant números:

- Només a l'ESO, que són 4 cursos, una escola acostuma a tenir 2 línies, això vol dir 8 grups d'alumnes (1r A, 1r B, 2n A ... 4t B).
- El nombre d'aules, comptant les pròpies de cada grup d'alumnes més les aules específiques com la d'informàtica o gimnàs, i aules per crèdits variables i desdoblaments, venen a ser unes 16.
- El nombre de professors necessaris per a impartir tots els crèdits està al voltant dels 16 i cal tenir en compte que no tenen totes les hores disponibles.
- Hi ha un centenar de crèdits, que tenen entre 1 i 4 sessions setmanals que cal repartir el millor possible, i algunes sessions s'han de fer en 2 hores consecutives.
- El total de sessions que cal combinar és d'unes 240 (8 grups per 30 hores setmanals). Això significa que, en el pitjor dels casos, si totes les sessions es poden posar a qualsevol hora de la setmana, tindrem 30^{240} combinacions possibles (el resultat és un número de més de 350 dígits: $3 \cdot 10^{354}$!!!).

I cal fer un horari nou cada trimestre!

És fàcil trobar un algorisme que generi totes les combinacions possibles i per tant es tracta d'un problema decidible (es pot resoldre amb l'ordinador).

També es pot veure que el nombre de combinacions possibles és exponencial doncs tenim un nombre d'hores setmanals constant (h) i el nombre de combinacions possibles varia exponencialment ($h^{\boldsymbol{n}}$) depenent del nombre de sessions ($\boldsymbol{n}$). Aquesta $\boldsymbol{n}$ és un valor molt elevat (nombre de grups d'alumnes multiplicat per 30), per tant es tracta d'un problema clarament intractable (difícilment es pot resoldre en un temps acceptable).

Ara bé, si tinguéssim una màquina no determinista i trobéssim un algorisme no determinista que ens permetés triar per cada sessió quin és el millor lloc possible, el cost seria polinòmic en lloc d'exponencial.

5.2. ESTUDI DELS POSSIBLES MÈTODES

Ara que s'ha vist que és un problema de tipus NP, falta trobar un mètode que permeti generar un horari en un temps raonable.

Entre tots els mètodes estudiats, sembla que els més adequats són el Backtracking i el Branch & Bound perquè garanteixen una resposta: o ens donen la solució o ens diran que no n'hi ha.

Aquest seria el camí més fàcil, però pel mateix motiu també és un camí força estudiat fins i tot en altres projectes de final de carrera. En aquests casos[10] però, el què s'ha fet és crear un programa per generar els horaris d'una universitat, que té uns requeriments més senzills que una escola de primària i secundària.

Amb els requeriments d'aquestes escoles[11], sembla més difícil que es pugui arribar a trobar una solució en un temps raonable per més ordenacions i podes que es facin amb els mètodes anteriors, per tant provarem altres mètodes.

Com que la programació dinàmica no sembla adequada per aquest problema, es pot provar a trobar un algorisme voraç.

5.3. ALGORISME VORAÇ

En principi, aquest mètode no sembla adequat, perquè una funció que permeti decidir on s'ha posar una sessió per garantir que es podran posar totes, tindria igualment cost exponencial. Però podem utilitzar aquest mètode per posar ràpidament la major part de les sessions i utilitzar un mètode basat en combinacions per posar les poques que quedin sense posar. Aquest altre mètode s'estudiarà més endavant.

5.3.1. Anàlisi i disseny

En aquest cas, el què ha de fer aquest algorisme és triar una sessió i decidir quin és el millor lloc per posar-la, fins que no quedi cap sessió pendent o fins que no es pugui posar cap més.

El principal problema consisteix en decidir quina sessió s'ha de posar i en quin lloc per aconseguir posar el màxim possible.

S'han estudiat diferents mètodes per poder decidir quin és el que dona millors resultats.

Els mètodes per triar la millor sessió (el millor crèdit) han estat els quatre següents, i dintre de cada un s'han provat dues versions per triar el millor lloc per posar la sessió.

---

[10] Projectes de final de carrera de la Universitat de Vic esmentats a la bibliografia

[11] Veure Capítol 3 - Anàlisi de requeriments

Exceptuant els mètodes 1.x, el sistema triat és dinàmic: les condicions van canviant a mida que anem posant sessions. Això farà que s'aconsegueixin millors resultats que si triéssim de bon principi l'ordre en què s'han de posar les sessions.

Evidentment, també serà menys eficient, però com que no s'han de reconsiderar les decisions preses, s'ha considerat més important posar el màxim d'hores que fer-ho el més ràpidament possible, sempre dintre d'uns límits. Amb unes estructures de dades adequades i tenint en compte la velocitat dels ordinadors actuals no es preveu que aquest mètode sigui massa lent.

Mètodes 1 i 2

Aquests dos mètodes trien un únic crèdit i decideixen el millor lloc per posar una sessió d'aquest crèdit.

El mètode 1 agafa els crèdits en el mateix ordre en què estan a la llista. Com es veurà a les proves, aquest mètode és el més ràpid però no dona gaire bons resultats. El cost és O(n), n = nombre de sessions, tenint en compte que les operacions per triar el dia i la hora són de cost màxim constant.

El mètode 2 recorre tota la llista i agafa el crèdit que té menys dies lliures[12] (poden haver-hi diversos crèdits que tinguin el mínim nombre de dies lliures, però agafa el primer que troba). Aquest és força millor i encara que ha de recórrer tota la llista cada vegada que s'ha de posar una sessió, continua essent molt ràpid. Per calcular el cost cal tenir en compte que les operacions per triar el dia i la hora són de cost màxim constant, per posar cada sessió hem de recórrer tota la llista de crèdits (m = crèdits) i tenim n sessions per posar. Tenint en compte que el nombre de sessions és aproximadament $c \cdot m$ (c és una constant que està al voltant de 2.5), el cost és $O(n^2)$.

Per decidir en quin lloc s'ha de posar la sessió, la primera variant (mètodes 1.1 i 2.1) agafa la primera hora lliure del primer dia que té hores lliures. Aquest sistema és molt senzill i ràpid però el resultat és que les sessions d'un mateix crèdit acostumen a quedar posades en dies consecutius, com es pot comprovar amb els resultats de les proves fetes, i un dels requeriments (encara que opcional) era intentar separar al màxim les sessions d'un mateix crèdit.

La segona variant busca quina seria la millor combinació possible a partir de les sessions d'aquest crèdit que ja estan posades i dels dies on no es pot posar cap sessió. Un cop ha trobat la millor combinació, agafa la primera hora del primer dia disponible d'aquesta combinació.

[12] El nombre de dies lliures és el nombre de dies disponibles (on es pot posar una sessió) menys el nombre de sessions que té el crèdit (en principi, cada sessió ha d'anar en un dia diferent)

Exemple:

Setmana de 5 dies
Crèdit amb 3 sessions
I = Dia Impossible
O = Dia Ocupat
D = Dia Disponible

Estat actual del crèdit:

Dilluns	Dimarts	Dimecres	Dijous	Divendres
I	O	D	D	D

Possibles combinacions de 3 sessions en 5 dies, ordenades de millor a pitjor:

	L	**M**	**X**	**J**	**V**
1	O		O		O
2	O	O			O
3	O			O	O
4		O	O		O
5	O		O	O	
6	O	O		O	
7		O		O	O
8		O	O	O	
9	O	O	O		
10			O	O	O

En aquest cas, la primera combinació que s'adapta a la situació actual del crèdit és la 4 i el primer dia disponible és el tercer dia (Dimecres).

Totes les possible combinacions es creen i es valoren quan es configura la setmana.

Mètodes 3 i 4

Aquests dos mètodes poden triar més d'un crèdit, i un cop triats decideixen quin és el crèdit que cal posar buscant entre ells l'hora on es pot posar una sessió de forma que afecti el mínim possible a tots els altres crèdits. Aquests mètodes continuen essent de cost polinòmic però més elevat: $O(n^3)$ pel cas pitjor.

El mètode 3 tria tots els crèdits que tenen el mateix nombre mínim de dies lliures. Els resultats són semblants al mètode 2, al menys amb els horaris amb que s'ha comprovat fins ara. És possible que amb horaris amb més hores prohibides, els resultats siguin millors.

El mètode 4 realitza les mateixes operacions que l'anterior però amb tots els crèdits, sense importar el nombre de dies lliures que tingui cada un. El temps emprat és molt superior al del mètode 3 i els resultats també han estat pitjor.

També amb aquests mètodes s'han provat dues versions per triar el millor dia i hora: provar totes les hores de tots els dies lliures (mètodes 3.1 i 4.1) o totes les hores dels dies que coincideixen amb la millor combinació possible (mètodes 3.2 i 4.2).

Tenint en compte els resultats i sobre tot el major cost (tot i que el temps final tampoc no és elevat perquè el nombre de crèdits no és excessiu), sembla que el millor mètode de tots quatre és el 2.2 i és el què finalment s'ha deixat implementat en la versió final del programa.

5.3.2. Estructures de dades

A continuació es descriuen els principals atributs de la classe *Credit* que serveixen per la generació d'horaris, especialment per la Fase 1 (algorisme voraç) i també els de les altres classes d'elements que conté *Credit*.

Credit

int sessions_setmana	Nombre de sessions que cal posar d'aquest crèdit
int dies_disponibles	Nombre de dies on encara es poden posar sessions
List sessions	Llista de sessions posades: indica a quina hora està posada cada sessió, si està fixada i permet obtenir el nombre de sessions posades consultant el nombre d'elements que conté
HoraCredit* horari	Taula on es guarda la informació de cada hora d'aquest crèdit (veure informació de l'estructura *HoraCredit* més avall) Conté *dies_setmana * hores_dia* elements
DiaCredit* dies	Taula on es guarda la informació de cada dia d'aquest crèdit (veure informació de l'estructura *DiaCredit* més avall) Conté *dies_setmana* elements

HoraCredit

bool impossible	Indica que aquesta hora està prohibida per posar sessions
int impossibles	Indica el nombre d'elements d'aquest crèdit que tenen aquesta hora prohibida (no es poden posar sessions en aquesta hora)
int fixades	Indica el nombre de crèdits relacionats amb aquest que tenen una sessió fixada en aquesta hora (per tant tampoc no es poden posar sessions d'aquest crèdit en aquesta hora)
int ocupades	Indica el nombre de crèdits relacionats amb aquest que tenen una sessió posada en aquesta hora (no es poden

	posar sessions d'aquest crèdit en aquesta hora a no ser que es treguin abans les sessions dels crèdits relacionats)
List credits	Conté els crèdits relacionats que tenen sessions posades en aquesta hora. Serveix per millorar el cost quan s'han de treure sessions coincidents per posar-ne una altra quan s'utilitza l'algorisme combinatori que s'estudia més endavant.

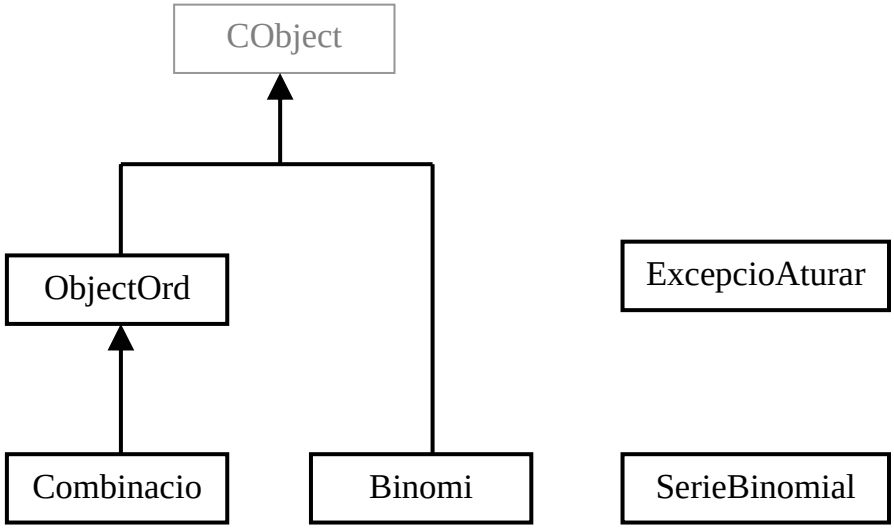
DiaCredit

int sessions_disponibles	És el nombre de llocs on es pot posar una sessió. Si el crèdit només té una hora per sessió, és el nombre d'hores lliures que té aquest dia
int sessions_posades	Quantes sessions hi ha posades en aquest dia. Encara que el programa només posarà una sessió a cada dia, manualment es poden posar més d'una.

Les classes necessàries per la generació d'horaris són les que es descriuen a continuació.

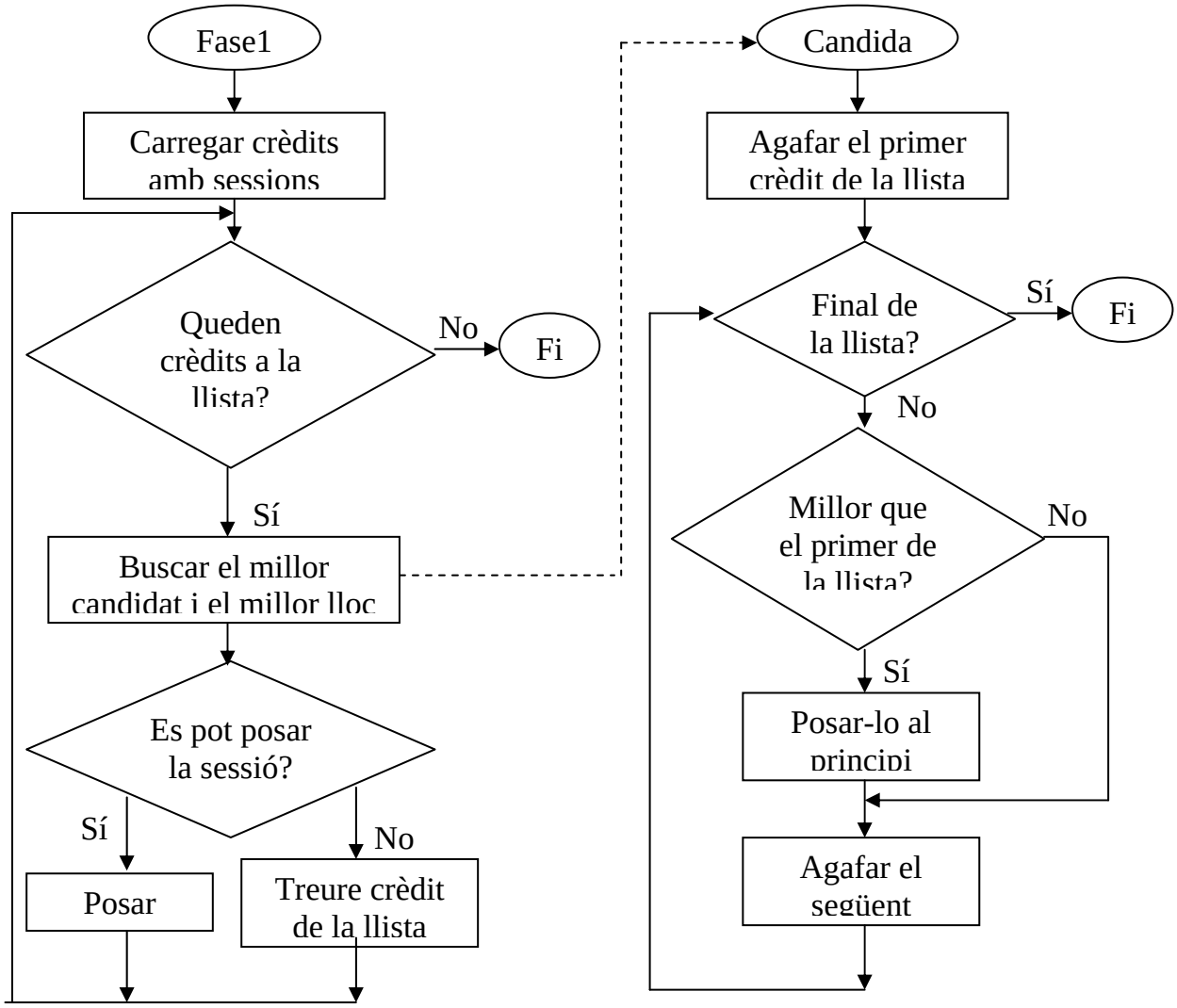
- **ExcepcioAturar:** Els processos de llarga durada llancen una excepció d'aquest tipus quan l'usuari decideix aturar el procés,
- **Combinacio:** Guarda una combinació amb els dies ocupats de la setmana. Per exemple, si la setmana té 5 dies i volem posar 3 sessions, una combinació podria ser O.L.O.L.O (O = Ocupada, L = Lliure). Cada combinació es valora per saber quines són millors, per exemple, O.L.O.L.O és molt millor que L.O.O.O.L,
- **Binomi:** Conté totes les combinacions possibles de **m** sobre **n** elements. En aquest cas, **m** és el nombre de dies que té la setmana i **n** el nombre de sessions setmanals. Les combinacions estan ordenades per poder triar fàcilment quina és la millor. Si **m** = 5 i **n** = 3 tindrem 10 possibles combinacions.
- **SerieBinomial:** Conté tots els possibles binomis (combinacions) donat el nombre de dies de la setmana. Si la setmana té 5 dies, tindrem 5 sobre 1 (5 dies, 1 sessió), 5 sobre 2, 5 sobre 3, 5 sobre 4 i 5 sobre 5.

Diagrama de la jerarquia de les classes per generar horaris



5.3.3. Implementació

Segueix l'esquema general dels algorismes voraços.
L'esquema representat correspon al mètode 2.2.
La resta de mètodes són semblants.



5.3.4. Test

A les següents taules es mostren els resultats de generar horaris amb les diverses variants de l'algorisme voraç.

Es poden veure el tant per cent de sessions que no s'han pogut posar i el nombre de crèdits que s'han posat però que tenen totes les seves sessions en dies consecutius, el què dona una idea de la qualitat de cada mètode.

Horari: Curs0203 (Primària)

Mètode	% Pendents	2 Sess. Consec.	3 Sess. Consec.	4 Sess. Consec.
1.1	12 %	36	20	10
1.2	8 %	-	-	-
2.1	5 %	20	13	26
2.2	5 %	6	3	-
3.1	7 %	13	23	26
3.2	7 %	5	-	-
4.1	7 %	33	18	10
4.2	7 %	7	-	-

Horari: Horari3T (Secundària)

Mètode	% Pendents	2 Sess. Consec.	3 Sess. Consec.	4 Sess. Consec.
1.1	12 %	9	22	-
1.2	9 %	1	1	-
2.1	4 %	8	13	-
2.2	4 %	1	3	-
3.1	4 %	2	11	-
3.2	3 %	3	4	-
4.1	7 %	7	14	-
4.2	11 %	1	6	-

5.4. ALGORISME COMBINATORI

El què es pretén amb aquest algorisme és posar sessions en una hora determinada desplaçant si cal altres sessions.

Aquest algorisme servirà per posar les sessions que no s'han pogut posar amb l'algorisme voraç. Com ara quedaran poques sessions, serà més fàcil que es puguin posar amb un mètode d'aquest tipus.

A més, la rutina encarregada de posar una sessió en una hora determinada servirà també per moure sessions manualment d'un lloc a un altre.

5.4.1. Anàlisi i disseny

La rutina principal provarà de posar totes les sessions que falten cridant la rutina encarregada de buscar un lloc on es pugui posar. Primer ho provarà amb un nivell de recursió 1, després amb el 2 i així fins que aconsegueixi posar totes les sessions pendents o fins que l'usuari aturi el procés. Això és el què es coneix com exploració en amplada de l'arbre de possibilitats i és útil en casos com aquest en què l'arbre de possibilitats és infinit o massa extens.

5.4.2. Millores del mètode bàsic

S'han provat dues variants d'aquest mètode per intentar reduir el temps que triga en trobar una solució.

5.4.2.1. Ordenar les diferents possibilitats

Aquest sistema consisteix en ordenar prèviament les hores on intentarà posar una sessió de forma que provi primer els llocs que semblen més prometedors.

Per triar quines hores semblen millors es mira el nombre de crèdits que cal desplaçar i la dificultat que comportaria recolocar-los.

La dificultat es calcula en el moment de crear el crèdit i ve donada pel nombre de crèdits simultanis i el nombre d'hores consecutives que té cada sessió. Després de varies proves, amb diferents factors, aquests han estat el que sembla que més afecten a la dificultat per moure un crèdit.

Amb aquesta millora s'han reduït els temps de l'ordre d'un 40% perquè, encara que es triga una mica més per haver d'ordenar les hores, en la majoria dels casos prova abans les hores que tenen més possibilitats de portar més ràpidament cap a una solució.

5.4.2.2. Descartar les possibilitats menys prometedores

En aquest cas s'ha decidit descartar les hores en què caldria desplaçar crèdits simultanis i crèdits amb més d'una hora per sessió.

Això ha portat, en la majoria dels casos, a una millora espectacular, però en algun cas concret el resultat ha estat pitjor, segurament degut a què les hores descartades eren les que portaven a una solució més ràpida encara que s'hagués de moure crèdits més complicats.

Per aquest motiu, en la versió definitiva del programa, s'ha decidit deixar implementada la primera millora en lloc d'aquesta última.

5.4.3. Implementació i proves

Aquesta és la fase més difícil degut principalment a les rutines recursives.

Per detectar fàcilment els possibles errors, s'ha inclòs codi que verifica si es compleixen els invariants. Aquest codi només s'executa quan es compila el programa en mode de depuració. La diferència entre executar el programa amb codi de depuració o sense és que es triga el triple de temps, però val la pena perquè la majoria d'errors s'han detectat fàcilment.

Algorisme PosarSessions2 (i auxiliars)

```
funcio PosarSessions2()
  var
    credit: element  // Crèdit del què s'intenta posar una sessió
    nivell: int      // Nivell màxim de recursivitat
    pendents: llista // Conté els crèdits amb sessions pendents
    solucio: llista  // On es guarden els moviments que cal fer
  fvar

  nivell = 0;
  // Posa a la llista els crèdits que tenen sessions pendents
  CarregarCredits(pendents)
  // Mentre quedin crèdits amb sessions pendents
  mentre no pendents.buit() fer
    // Prova amb el següent nivell de recursivitat
    nivell = nivell + 1
    // Agafa el primer crèdit de la llista de crèdits pendents
    credit = pendents.primer()
    // Prova tots els crèdits de la llista
    mentre credit != NULL fer
      // Si es pot posar, el posa
      si EsPotPosar(credit, nivell, solucio)
        PosarSolucio(solucio)
        // Si no té més sessions pendents, l'elimina de la llista
        si credit.sessionsPendents = 0 llavors
          credit = pendents.eliminar(credit)
        fsi
      // Si no es pot posar, prova el següent crèdit
      else
        credit = pendents.seguent(credit)
      fsi
    fmentre
  fmentre
ffuncio


// Busca una hora on es pugui posar una sessió del crèdit
// Si la troba, guarda els moviments que cal fer a 'solucio'.
// 'nivell' indica el nivell màxim de recursivitat
funcio EsPotPosar(credit, nivell, solucio) : boolea
  var
```

```
    tHora: int
  fvar

  // Prova a totes les hores de la setmana
  per tHora = 1 fins hores_setmana fer
    // Si ha trobat una solucio, retorna cert
    si ObligarSessio(credit, tHora, nivell, solucio) = true
      return true
    fsi
  fper
  // Retorna sense haver trobat cap solució
  return false
ffuncio


// Posa la sessió en una hora desplaçant si cal altres sessions
funcio ObligarSessio(credit, hora, nivell, solucio) : boolea
  var
    trets: llista de crèdits que s'han tret per posar la sessio
  fvar

  // Si es pot posar directament sense treure cap sessió...
  si HoraLliure(credit, hora) llavors
    // Posa la sessió i afegeix el moviment realitzat
    PosarSessio(credit, hora, solucio)
    return true
  sino
    // Si s'ha arribat al límit de recursivitat retorna fals
    si nivell == 0 return false
    // Si no s'ha arribat al límit de recursivitat...
    // Treu les sessions necessàries per posar aquesta sessió
    TreureSessionsCoincidents(credit, hora, trets)
    // Posa la sessio i afegeix el moviment realitzat
    PosarSessio(credit, hora, solucio)
    // Si aconsegueix recolocar tots els credits trets retorna
cert
    si Recolocar(trets, trets.primer(), hora, nivell -1, solucio)
      return true
    fsi
    // No ha pogut recolocar totes les sessions,
    // per tant cal desfer els moviments
    DesferMoviments(solucio)
    return false
  fsi
ffuncio


// Intenta recolocar les sessions que havia tret
funcio Recolocar(trets, credit, hora, nivell, solucio) : boolea
  var
    hPossibles: int[horesSetmana] // Hores possibles
    nhp: int     // Nombre total d'hores on es pot posar
    tHora: int
```

```
  fvar

  // Si s'ha arriba a l'últim crèdit de la llista, retorna cert
  si credit = NULL return true

  // Busca les hores on es pot posar la sessió,
  // encara que hagi de desplaçar altres sessions
  nhp = HoresPossibles(hPossibles)

  // Prova a cada hora fins que els pot posar
  per tHora = 1 fins horesSetmana
    // Si el pot posar, intenta posar el següent credit
    si ObligarSessio(credit, tHora, nivell, solucio) llavors
      // Si els ha pogut posar tots, retorna cert
      si   Recolocar(trets,   credit.seguent(),   tHora,   nivell,
solucio)
        return true
      fsi
      // No ha pogut recolocar tots els crèdits,
      // per tant cal desfer els moviments
      DesferMoviments(solucio)
    fsi
  fper
  return false
ffuncio
```

5.4.4. Càlcul del cost

5.4.4.1. Consideracions prèvies

Es farà el càlcul del cost assimptòtic de la rutina recursiva *ObligarSessio* per comprovar si es manté el cost exponencial o si empitjora gaire respecte els mètodes habituals que consisteixen provar totes les combinacions possibles.

En aquests casos, com s'havia vist a la introducció, el cost per un cas típic d'un horari de 30 hores setmanals era de 30^n, essent *n* el nombre de sessions que calia posar.

Degut a què el cost de les rutines varia segons els crèdits que intervenen, les hores impossibles i altres molts factors que a més canvien dinàmicament, es parteix de l'estudi d'un cas típic amb els següents valors comprovats estadísticament:

- Escola de secundària amb 2 línies (8 grups d'alumnes), 100 crèdits, 240 sessions
- Cada crèdit està relacionat, de mitjana, amb altres 25 crèdits
- El nombre d'hores setmanals és de 30
- La mitjana de sessions que cal treure per poder posar una sessió en un lloc determinat és 2.

Aquí no s'estudiarà amb detall el cost de les rutines bàsiques necessàries pel càlcul del cost de la rutina recursiva però a la següent taula es proporciona l'explicació de perquè són de cost constant [13].

Rutina	Cost
`HoraLliure()` Serveix per saber si es pot posar una sessió en una hora determinada d'un crèdit. Només cal consultar un valor que es troba en un vector i per tant el cost és unitari.	c_1
`HoresPossibles()` Busca les hores on es pot posar una sessió d'un crèdit encara que sigui desplaçant sessions d'altres crèdits. En el pitjor dels casos, que és quan no hi ha hores impossibles ni sessions fixades, es podrà posar la sessió en qualsevol hora de la setmana, per tant el cost serà el de consultar si l'hora està disponible i guardar-la en un vector, multiplicat per 30 hores que té la setmana..	c_2
`PosarSessio()` Quan es posa una sessió, a més d'indicar que l'hora del crèdit està ocupada, també cal informar als crèdits relacionats per a evitar que cada cop que vulguin saber si una hora està disponible, hagin de consultar a tots els crèdits relacionats. Cada crèdit conté una llista de crèdits relacionats i per tant només cal seguir la llista i actualitzar l'hora de cada un dels crèdits de la llista. També s'afegeix el crèdit a una llista que utilitzarà la funció `TreureSessio()` i `TreureSessionsCoincidents()` per aconseguir cost constant.	c_3
`TreureSessio()` Igual que al cas anterior.	c_3
`TreureSessionsCoincidents()` Com de mitjana cal treure dues sessions per posar-ne una, el cost serà el doble que per treure una sessió.	$c_4 = 2 * c_3$
`DesferMoviments()` Els moviments que cal desfer són les sessions coincidents que s'han hagut de treure (normalment 2) i la sessió que s'ha posat en el seu lloc, per tant el cost serà el de treure una sessió i posar-ne dos.	$c_5 = 3 * c_3$

[13] S'han considerat operacions elementals de cost unitari les operacions matemàtiques, l'accés a les variables i als elements dels vectors, l'avaluació dels condicionals, el control dels bucles i els salts.

5.4.4.2. Simplificació de les funcions

En el cas pitjor que és quan no es pot trobar una solució les rutines *ObligarSessio* i *Recolocar* es poden simplificar i quedaran de la següent manera [14]:

```
funcio ObligarSessio(credit, hora, nivell, solucio) : boolea
  si HoraLliure(credit, hora) llavors
    // Mai entrarà aquí
  sino
    si nivell == 0 return false
    TreureSessionsCoincidents(credit, hora, trets)
    PosarSessio(credit, hora, solucio)
    si Recolocar(trets, trets.primer(), hora, nivell-1, solucio)
      // Mai entrarà aquí
    fsi
    DesferMoviments(solucio)
    return false
  fsi
ffuncio

funcio Recolocar(trets, credit, hora, nivell, solucio) : boolea
  si credit = NULL ... // Mai es compleix això
  nhp = HoresPossibles(hPossibles)
  per tHora = 1 fins horesSetmana
    si ObligarSessio(credit, tHora, nivell, solucio)
      si  Recolocar(trets,  credit.seguent(),  tHora,  nivell,
solucio)
        // Mai entrarà aquí
      fsi
      DesferMoviments(solucio)
    fsi
  fper
  return false
ffuncio
```

La rutina *Recolocar* es pot reconvertir de forma que no es cridi a sí mateixa, cosa que també simplificarà l'anàlisi posterior.

Tampoc cal desfer els moviments aquí, doncs com sempre retornarà fals, els desfarà la rutina *ObligarSessio*.

```
funcio Recolocar(trets, credit, hora, nivell, solucio) : boolea
  si credit = NULL // Mai es compleix això
  mentre credit != NULL fer
    nhp = HoresPossibles(hPossibles)
    per tHora = 1 fins horesSetmana
      si ObligarSessio(credit, tHora, nivell, solucio)
```

[14] s'han tret tots els comentaris, la definició de les variables locals i s'ha indicat els lloc per on no passarà mai en el cas pitjor.

```
        // Mai entrarà aquí
      fsi
    fper
    credit = trets.seguent(credit)
  fmentre
  return false
ffuncio
```

Per últim, posarem la rutina `Recolocar` dintre de `ObligarSessio`. D'aquesta manera el càlcul quedarà només en funció de `ObligarSessio`.

```
funcio ObligarSessio(credit, hora, nivell, solucio) : boolea
  si HoraLliure(credit, hora) llavors
    // Mai entrarà aquí
  sino
    si nivell == 0 return false
    TreureSessionsCoincidents(credit, hora, trets)
    PosarSessio(credit, hora, solucio)
    credit = trets.primer()
    mentre credit != NULL fer
      nhp = HoresPossibles(hPossibles)
      per tHora = 1 fins horesSetmana
        si ObligarSessio(credit, tHora, nivell-1, solucio)
          // Mai entrarà aquí
        fsi
      fper
      credit = trets.seguent(credit)
    fmentre
    DesferMoviments(solucio)
    return false
  fsi
ffuncio
```

5.4.4.3. Càlcul

A partir dels valors calculats prèviament i la funció simplificada, es pot trobar fàcilment una fórmula per calcular el cost.

Un mètode que es pot utilitzar en casos senzills com aquest és intentar deduir la fórmula a partir del càlcul per n = 0, 1, 2...

Primer es calcula el cost pel cas n = 0 que és quan s'arriba a l'últim nivell de recursivitat. En aquest cas només es passa per les primeres línies i per tant el cost serà:

```
(c1)          si HoraLliure(credit, hora) llavors
                // Mai entrarà aquí
              sino
(c1)            si nivell == 0 return false
---------------------------------
(2 * c1)
```

Per tant, el cost és constant.

Després es calcula el cost per la resta de casos, però es farà per parts, començant pels bucles més interns.

El cost de cada línia s'ha posat al principi, entre parèntesis.

X serà el cost de la funció *ObligarSessio*.

Bucle **per**:

```
Aquest bucle es realitza 30 vegades (30 hores setmanals)

(c1)      per tHora = 1 fins horesSetmana
(c1 + X)    si ObligarSessio(...)
              // Mai entrarà aquí
            fsi
(c1)      fper
---------------------------------
(3 * c1 + X)
```

Com el bucle s'executa 30 vegades, el cost serà:

$$30 * (3 * c_1 + X) = 90 * c_1 + 30X$$

Per simplificar:

$$c_6 = 90 * c_1 \quad \text{-->} \quad TOTAL = c_6 + 30X$$

Bucle **mentre**:

```
Aquest bucle es realitza 2 vegades (es treuen 2 sessions de
mitjana per poder posar una sessió en un lloc ocupat) i conté el
bucle per
```

```
(c1)          mentre credit != NULL fer
(c2)            nhp = HoresPossibles(hPossibles)
(c6 + 30X)      // cost del bucle per
(c1)            credit = trets.seguent(credit)
(c1)          fmentre
---------------------------------
(3 * c1 + c6 + 30X))
```

Com el bucle s'executa 2 vegades, el cost serà:

$$2 * (3 * c_1 + c_6 + 30X)) = 6 * c_1 + 2 * c_6 + 2 * 30X$$

Per simplificar:

$$c_7 = 6 * c_1 + 2 * c_6 \quad \text{-->} \quad TOTAL = c_7 + 60X$$

Per últim es calcula el cost de la rutina que es vol analitzar, substituint els valors calculats dels bucles.

Funció **`ObligarSessio:`**

```
(c1)          si HoraLliure(credit, hora) llavors
                // Mai entrarà aquí
              sino
(c1)            si nivell == 0 return false
(c4)            TreureSessionsCoincidents(credit, hora, trets)
(c3)            PosarSessio(credit, hora, solucio)
(c1)            credit = trets.primer()
(c7 + 60X)      // cost del bucle mentre
(c5)            DesferMoviments(solucio)
(c1)            return false
              fsi
---------------------------------
(3 * c1 + c3 + c4 + c5 + c7 + 60X)
```

Per simplificar:

$$c_n = 3 * c_1 + c_3 + c_4 + c_5 + c_7 \quad \text{-->} \quad TOTAL = c_n + 60X$$

Per tant, tenim la següent funció de cost:

- per n = 0 $T(n) = 2 * c_1$ // cost constant
- per n > 0 $T(n) = c_n + 60 * T(n-1)$

Ara serà fàcil deduir la fórmula per calcular el cost:

n	cost
0	$2 * c_1$
1	$c_n + 60 * (2 * c_1)$
2	$c_n + 60 * (c_n + 60 * (2 * c_1))$
3	$c_n + 60 * (c_n + 60 * (c_n + 60 * (2 * c_1)))$

$T(n) = 60^0 * c_n + 60^1 * c_n + ... + 60^{n-1} * c_n + 60^n * 2 * c_1$

Per tant, el cost assimptòtic està en l'ordre de $O(60^n)$, o sigui, cost exponencial [15].

5.5. CONTRASTACIÓ DE LA TEORIA AMB LA PRÀCTICA

Comparem ara amb el mètode habitual que consisteix a provar totes les combinacions possibles, per veure si realment s'aconseguirà una millora pràctica.

En el mètode habitual, si considerem que les rutines per posar i treure sessions són de cost constant, el càlcul és ben fàcil: si tenim una setmana de 30 hores i s'han de posar *n* sessions, el cost és 30^n [16].

A més, la constant oculta serà inferior degut a que no cal treure i recolocar sessions, perquè si una sessió no es pot posar, simplement farà marxa enrera, i tampoc cal guardar una llista de les sessions que estan posades a cada hora perquè és suficient saber que l'hora està ocupada per un altre crèdit.

Per tant està clar que el cost real del mètode habitual és inferior al del mètode que s'ha estudiat, encara que tots dos siguin de cost exponencial.

---

[15] Aquest cost només és per posar una sessió, per tant encara s'hauria de multiplicar pel nombre de sessions que s'han de posar, encara que això no afecta al cost assimptòtic.

[16] Deixem de banda les possibles millores com *Branch & Bound* perquè estem parlant de cost assimptòtic i en el pitjor cas tenen el mateix cost.

Però no s'ha de perdre de vista l'objectiu d'un mètode i de l'altre.

El mètode habitual ha de posar TOTES les sessions, que poden ser unes 240 en un cas mitjà.

El mètode per obligar una sessió en un lloc determinat només s'utilitzarà després que l'algorisme voraç hagi posat la immensa majoria de les sessions [17], i no es preveu que s'hagi d'arribar a un nivell de recursivitat gaire elevat, si bé és veritat que en el pitjor dels casos també s'hauria d'arribar a un nivell tan elevat com amb el mètode habitual.

De fet, amb les proves realitzades s'ha comprovat que, amb horaris de dificultat normal, el nivell màxim de recursió a què s'ha arribat ha estat el nivell 4 i ha trigat uns 90 segons. Fins i tot amb horaris especialment complicats, amb molts condicionants [18], el nivell màxim ha estat el 6 i el temps, 10 minuts.

[17] Amb horaris reals, fins i tot amb molts condicionants, l'algorisme voraç ha aconseguit posar al voltant del 95% de les sessions.

[18] Moltes hores impossibles, hores per reunions i hores reservades per tardes lliures dels professors.

6. CONCLUSIONS

A la vista dels resultats, queda demostrada l'eficiència del sistema utilitzat per generar els horaris. De totes formes, amb una mica més de temps, s'haurien pogut estudiar altres millores i fer proves més exhaustives amb horaris més complicats.

Dels comentaris fets pels caps d'estudi què han provat el programa sembla que també s'ha aconseguit l'objectiu de que fos senzill d'utilitzar i que permetés treballar amb tots els condicionants que es donen a les escoles de primària i secundària.

Malgrat la dificultat de realitzar un projecte d'aquesta magnitud en el poc temps que dura un curs, el resultat ha valgut la pena.

RESUM

El projecte consisteix en la creació d'un programa per generar els horaris de les escoles de primària i secundària.

Aquest programa ha de ser fàcil d'utilitzar i capaç de generar l'horari en un temps raonable.

Degut als requeriments d'aquestes escoles cal cercar mètodes alternatius als habituals que consisteixen en provar totes les combinacions (*Backtracking* i similars).

Per aconseguir-ho s'ha optat per dividir el procés en dues fases:

1) A la primera fase s'intenta posar el màxim de sessions amb un algorisme voraç. Amb això s'aconsegueix posar un gran nombre de sessions en molt poc temps donat que no es proven combinacions ni es reconsideren les opcions preses.

2) A la segona fase s'intenta posar les sessions que no s'han pogut posar abans, desplaçant altres sessions que s'havien posat prèviament. Donat el petit nombre de sessions que quedaran per posar ara sí que es podrà utilitzar un algorisme combinatori.

Els resultats han estat molt satisfactoris doncs, fins i tot amb horaris de prova amb molts més condicionants del què és normal, ha aconseguit trobar una solució en un temps màxim de 10 minuts. Amb horaris normals, amb dades proporcionades per les escoles que l'han provat, el temps ha estat entre 30 i 90 segons.

BIBLIOGRAFIA

Llibres

Brassart, Bratley (1997)
Fundamentos de algoritmia
Prentice Hall

Angel Franco García (1995)
Desarrollo avanzado de aplicaciones Windows
McGraw-Hill

Richard C. Leinecker i Tom Archer (1999)
La Biblia de Visual C++ 6
AnayaMultimedia

Apunts de la Universitat Oberta de Catalunya de les següents assignatures:

Teoria d'autòmats i llenguatges formals II
Estructura de la Informació
Interacció Humana amb els Computadors
Tècniques de Desenvolupament de Programari

Projectes de final de carrera de la Universitat de Vic

Josep Grifell Mauri *Treball Final de Carrera nº 32*
Frederic Grangé Borràs *Treball Final de Carrera nº 224*

Adreces Web

http://www.lenguaje-c.es.vg/
http://www.codeguru.com/index.shtml
http://devcentral.iftech.com/articles/C++/default.php
http://www.lavariable.com/

GLOSSARI I ABREVIACIONS

Assignatura: conjunt de coneixements relacionats amb un tema i que s'imparteixen de manera fraccionada en forma de crèdits al llarg dels diferents cursos escolars. Per exemple, Matemàtiques, Ciències Socials...

Crèdit: assignatura impartida per un professor concret a un grup d'alumnes. Per exemple, el crèdit *Fun with words* és un crèdit variable d'anglès, impartit per la professora M. Riera als alumnes de 4t A i 4t B d'ESO.

Crèdits comuns: (veure *Crèdits obligatoris*)

Crèdits obligatoris: són crèdits que han de fer tots els alumnes.

Crèdits optatius: són crèdits que els alumnes poden triar. Poden ser realitzats per alumnes de diversos grups.

Crèdits simultanis: són els crèdits optatius que es realitzen al mateix temps.

Curs: cada una de les etapes educatives. Per exemple, 1r d'EP, 3r d'ESO, 2n de Batxillerat...

EP: Educació Primària

ESO: Educació Secundària Obligatòria

Grup d'alumnes: alumnes que estan junts a la mateixa aula quan fan els crèdits obligatoris d'un curs.

Grups flexibles: són grups d'alumnes que es formen amb alumnes d'un o de diferents grups per realitzar crèdits adaptats a les seves necessitats.

Hora disponible: hora en la què es pot posar una sessió

Hora possible: hora en la què es pot posar una sessió si es treuen abans les sessions d'altres crèdits relacionats.

Hora impossible: (veure *Hora prohibida*)

Hora lliure: (veure *Hora disponible*)

Hora ocupada: hora en la què hi ha una sessió posada

Hora prohibida: hora en la què no es poden posar sessions.

Línies: nombre de grups d'alumnes que té l'escola per cada curs. Una escola que faci els 4 cursos de l'ESO i tingui dues línies, tindrà 8 grups d'alumnes: 1rA, 1rB, 2nA, 2nB, 3rA, 3rB, 4tA i 4tB

Sessió: conjunt d'hores consecutives en què s'imparteix un crèdit. Per exemple, un crèdit pot tenir dues hores per sessió. Si aquest crèdit té dues sessions setmanals (dos dies per setmana), en total té quatre hores setmanals.

Sessió fixada: sessió que no es pot canviar de lloc.

ANNEX

Sense intenció de fer una anàlisi exhaustiva dels programes que hi ha actualment al mercat donat que això suposaria feina per un projecte sencer, a continuació hi ha una llista amb alguns dels programes que s'han provat i els avantatges i inconvenients que s'han trobat en general.

Empresa	Nom del programa	Adreça Web
Sinectics	Horario	http://webs.sinectis.com.ar/alejand/
Peñalara Software	GCH 6.0	http://www.penalara.com/
Knosis	KronoWin Mil·lenium 2	http://www.adossis.es

Avantatges

Tenen moltes possibilitats de configuració i moltes opcions, especialment a l'hora de treure les llistes i els horaris per impressora.

Permeten veure les dades combinades de moltes formes diferents.

Si es configuren correctament, generen horaris òptims.

Permeten exportar i importar dades d'altres programes.

Inconvenients

És excessivament complicat configurar totes les dades correctament.

La introducció de dades és complicada i gens intuïtiva.

No donen gaires facilitats per modificar els horaris generats. Parteixen de què si els configures correctament, l'horari que generen ja és correcte i no cal fer cap modificació.

No permeten barrejar alumnes de diferents grups.

Tenen massa coloraines, que en lloc d'ajudar a distingir les coses importants, distreuen.

Són molt lents.

Són molt cars.

Les "demos" són molt limitades. No permeten comprovar si el programa ens serà útil.