Generació automàtica de test, per a l'avaluació d'exercicis de programació

Ricard Bernal Petit

Resum—Donada la problemàtica de la correcció d'exercicis de programació, causada per la gran quantitat d'alumnes i per tant d'exercicis entregats per part d'aquests, els docents encarregats de mantenir l'avaluació continua al dia han de realitzar grans esforços, ja que han de crear des de zero els test que s'aplicaran als exercicis per obtenir la nota de l'exercici de cada alumne, això sumat al temps invertit a crear nous exercicis per poder dinamitzar les classes i adaptar-se als avenços en el camp de les noves metodologies i procediments de desenvolupament. Per tal d'agilitzar aquest procés, s'ha desenvolupat una aplicació Java que permet generar automàticament aquestes classes de test, a través d'introduir un fitxer de text pla en l'estil acordat, el qual és tractat i utilitzat per obtenir la informació necessària per generar un fitxer de sortida en el format acordat, en aquest cas c++, que ja permet testejar els exercicis demanats pel docent.

Paraules clau—POO, concepte classe, software, BDD, TDD, testing automàtic, test extrem a extrem, GUI, document de text pla, C++, Java, UML, Inteligencia artificial.

Abstract— Due to the problem of the correction of programming exercises, caused by the large number of students and therefore of exercises delivered by them, the teachers responsible for keeping the evaluation up-to-date must realize large efforts, since they have to create from scratch the tests that will be applied to the exercises to obtain the note of the exercise of each student, this added to the time invested in creating new exercises to be able to dynamize the classes and adapt to the advances in the field of new methodologies and development procedures. In order to streamline this process, a Java application has been developed that allows you to automatically generate these test classes, by entering a plain text file in the agreed style, which is treated and used to obtain the necessary information to generate an output file in the agreed format, in this case c++, which already allows to test the exercises requested by the teacher.

Index Terms— POO, class concept, software, BDD, TDD, automatic testing, end to end test, GUI, plain text document, C++, Java, UML, Artificial intelligence.

1 Introducció

DURANT els últims anys el testing s'ha tornat imprescindible en les empreses de desenvolupament de software que volen oferir un producte de qualitat, precís i adaptat a les necessitats dels clients. Mantenir una àmplia cobertura del software implica obtenir uns resultats de gran qualitat en diferents nivells del producte final, principalment s'aconsegueix que el software sigui funcional en tot moment de tal manera que els desenvolupadors sempre treballen sobre un software sòlid i per altra banda s'obté un producte sense errors de funcionalitat, d'integració, de components i fins i tot d'acceptació. Però tots aquests beneficis impliquen una inversió de capital i de temps important, és per això que amb aquest projecte es pretén agilitzar la creació de test, mitjançant l'automatització d'aquest. Per poder entendre aquesta relació cal explicar com es relaciona el testing convencional i les classes de test que verifiquen els exercicis de programació, indicant una puntuació per a cada apartat de l'exercici. D'una forma molt sintetitzada es pot dir que la creació d'una classe de test per cada exercici implica la creació de test automàtic el qual a de reproduir un escenari específic i verificar el codi desenvolupat en aquella situació, per tant si s'aplica al projecte actual, l'aplicació cal que genèric un escenari específic (mètodes a verificar) i ha de verificar que el codi compleixi la funció desitjada assignant una puntuació a cada exercici. Per tal de definir els límits amb més claredat, en l'apartat del desenvolupament s'aprofundirà més en aquest concepte.

En aquest article es realitzarà un anàlisis de la problematica, de la qual es presentarà la solució que dona lloc a la realització d'aquest projecte i per tant es donarà una explicació del projecte. L'article està estructurat de la següent manera, primer es definiran els objectius, s'esmentaran els projectes que utilitzen la mateixa idea de base, es parlarà de la metodologia que s'ha seguit per desenvolupar el software i finalment es mostraran els resultats obtinguts així com trobarem un apartat final amb les conclusions a les que s'han arribat.

- *E-mail de contacte: ricard.bernal@e-campus.uab.cat*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Ernest Valveny Llobet (Ciències de la Computació)*
- *Curs 2018/19*

2 OBJECTIUS

Per entendre l'objectiu d'aquest projecte cal com a mínim tenir en compte la problemàtica per la qual sorgeix la necessitat de crear un projecte com aquest, per tant primer s'explicarà el problema i després l'abast dels objectius.

2.1 Problema inicial

Si ens parem a pensar la quantitat d'alumnes matriculats al grau d'enginyeria informàtica i les dobles titulacions[1] i per tant la quantitat d'alumnes [2] que han de realitzar l'assignatura de Laboratori de Programació, s'entendrà que si el 80% d'aquests entrega els exercicis, el nombre de correccions és molt elevada sobretot si el docent, amb l'objectiu de dinamitzar l'assignatura, vol renovar els exercicis cada cert temps, creant-ne de nous o adaptant-ne algun d'existent, per tal d'aplicar noves tecnologies i metodologies, cal que també generi o modifiqui el seu codi de test per validar que l'exercici.

2.2 Objectius i abast

L'objectiu principal d'aquest projecte és la generació automàtica de classes en llenguatge C++ que actuïn com a classes de tests sobre el codi presentat per part dels alumnes per tal d'avaluar aquesta gran quantitat d'exercicis entregats, però la proposta té un abast més extens, ja que es vol arribar a donar el màxim de flexibilitat a la generació d'aquests codis de test per tal que el docent pugui focalitzar-se en la creació de nous exercicis sense preocupar-se de la generació del seu codi de test.

3 ESTAT DE L'ART

Actualment existeixen moltes solucions al mercat, que apliquen les tècniques de testing convencionals, principalment utilitzant l'enginyeria de software BDD (behavior driven develpoment) [3], que pretén acostar el desenvolupament a departaments o companys amb menys coneixements tècnics, però igualment responsables d'obtenir un software de qualitat i ho aconsegueix mitjançant les històries d'usuari. Existeixen diferents nivells de testing, a la figura 1 es mostra la situació ideal del testing de software, la piràmide esta formada pels següents nivells:

- Els **test unitaris** els quals són responsabilitat dels desenvolupadors.
- Els **tests d'integració**, dins dels quals trobem els tests de components i els test de sistema, els quals esta sota la responsabilitat dels testers.
- Finalment els **test *end-to-end*** que fan referència als test automàtics de la GUI o interfície gràfica i també son responsabilitat dels testers.

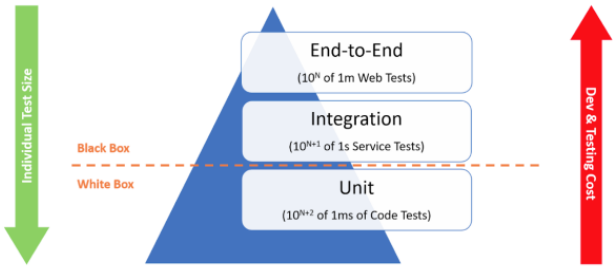

Figura 1: Piràmide del *testing*

Amb el naixement d'aquesta metodologia, empreses com cucumber amb la sintaxi gherkin [4], selenium [5] que permet fer testing automàtic, que no generar test de forma automàtica, a partir d'un ventall de llenguatges de programació web molt ampli o llenguatges de programació com JUnit que permeten desenvolupar test unitari. El fet és que per molt que hi hagi solucions i productes al mercat que treballen sobre aquest camp, no hi ha productes que generin el codi automàticament i és per això que té cabuda un projecte com aquest. La principal diferencia entre el test automàtic i el que s'ha desenvolupat amb aquest projecte és que el testing automàtic permet executar test *end to end* sobre el codi, en canvi la finalitat d'aquest projecte és la de crear el test pròpiament dit.

Per altra banda durant molts anys els científics estan buscant algoritmes i tècniques per dotar les màquines d'una capacitat d'aprenentatge i independència que els permeti assemblar-se als éssers humans, aquestes característiques fan que les seves aplicacions englobin un ventall molt ampli de camps i en aquest cas tenen una implicació en el cicle de desenvolupament [6] que tractarem com a concepte de generació de codi.

Aquest nou concepte de generació de codi es pot trobar per exemple en els compiladors de codi els quals donen recomanacions mentre el desenvolupador programa, per tant serveix com a suport i ajuda en el procés de desenvolupament, però es requereix un programador que gestioni aquestes noves línies de codi i la seva concordança amb el projecte i per altra banda, avui dia ja hi ha intel·ligències artificials i tecnologies d'aprenentatge automàtic que poden arribar a gestionar el codi de tal manera que es poden integrar en el cicle de desenvolupament de software, a continuació es descriuen algun dels exemples més recents:

- **Google buspot tool w3c**: utilitza algoritmes d'aprenentatge automàtic i anàlisi estadístic profund, per tal de trobar línies de codi poc fiables, a partir de mètriques, dependències i fins i tot altres línies de codi. Molt útil per tal de gestionar la gran quantitat de línies de codi que és modifiquen i s'incorporen al software diàriament.
- **Stack overflow autocomplet**: L'eina de Stack Exchange permet autocompletar línies de codi plenament funcionals a partir de les dades de codi existents i les intencions del desenvolupador.
- **Deep code**: Es tracta d'un assistent per desenvolupadors, que mitjançant l'aprenentatge de línies de

codi de repositoris públics com github, suggereix com arreglar el codi per solucionar errors i optimitzar codi.

Analitzant la llista podem observar que els avenços més recents [7], no estan molt estesos i la seva aplicació encara és molt dèbil, però serveixen per entendre l'interes que hi ha en aquesta generació automàtica de codi que pot ser podria arribar a convertir-se en una nova forma de desenvolupar, on el desnevolupador passa a ser un espectador, que s'encarrega de supervisar la generació de codi.

En conclusió, es tracta d'un concepte nou, pel fet que avui dia no hi ha cap codi efectiu que generi codi de test o productiu des de zero i es per això que el seu desenvolupament ha estat costos i difícil de portar a terme.

4 METODOLOGIA

Després d'una breu recerca dels models de desenvolupament de programari [8] i a causa d'una convicció personal, s'ha escollit el model de desenvolupament àgil, del qual en deriven diverses metodologies avui en dia molt esteses i aplicades a moltes empreses. Com que estan enfocades a la col·laboració entre equips i en aquest cas no serà possible realitzar un desenvolupament col·laboratiu, caldrà redactar uns principis bàsics recolzats per alguna o varies de les metodologies analitzades, per tal d'obtenir la metodologia que se seguirà en aquest projecte. Per tal de generar aquests principis, s'han analitzat tres de les metodologies més rellevants, ja sigui pel fet que van canviar la manera de pensar i fer les coses, com pel fet de ser les més utilitzades, aquestes són: SCRUM [9], Canvas [10][11] i DevOps [12][13]; Com que la majoria de les metodologies estan enfocades a negoci, ens centrarem en els punts que tractin la manera d'actuar a l'hora de desenvolupar programari i la manera com planificar les activitats a realitzar, de tal manera que els principis de la metodologia que se seguirà en aquest projecte queden de la següent manera:

- Cicles temporals de treball, curts i de duració fixa. Amb sincronització diària.
- Cada iteració aporta un resultat complet, que incrementa el producte final. Així com es tornen a valorar els requisits, per tal de prioritzar objectius.
- Proves periòdiques i versions freqüents en comptes de planificació fixa.
- Estret contacte amb el client (en aquest cas el docent) el qual prioritza els objectius.

Aquesta metodologia de treball àgil comptarà amb el suport de modelatge UML, amb els corresponents diagrames, per tal de treballar sobre dissenys sorgits d'anàlisis previs al desenvolupament del programari. Per altra banda i després d'una anàlisi intern dels coneixements de tots els llenguatges de programació coneguts i algun cop utilitzats, s'ha arribat a la conclusió que el llenguatge més adient per desenvolupar el software, serà Java.

5 DESENVOLUPAMENT

5.1 Disseny

Per tal de portar aquest projecte endavant i poder arribar a aconseguir l'objectiu acordat, la solució ha passat per diversos estats, per començar es va realitzar un anàlisi previ, que amb el suport d'una recerca amplia i continua, ha permès treballar sobre una base consistent suportada per diagrames i decisions precises, per altra banda a causa de la metodologia àgil utilitzada, aquest anàlisis ha sigut continuo i ha permès dinamitzar el desenvolupament del software, a través de modelar l'estructura segons les necessitats i noves idees sorgides durant el desenvolupament. Cal esmentar que per realitzar els diagrames s'ha fet servir el concepte UML, diagram a que s'adjunta a l'apendix A1, com a figura 2. Les recerques més rellevants estan reflectides a la bibliografia.

Donat un estudi i posterior anàlisi del disseny estructural més adequat per a la realització d'aquest projecte, finalment l'estructura que s'ha seguit és la que es mostra a la Figura 3, que es troba a l'apèndix A3.

En la Figura 3 trobem representada l'estructura de paquets i classes del projecte Java, estructura de la qual és detallaran certes parts rellevants d'aquesta distribució en el següent apartat:

Per tal de poder treballar amb coherència i estabilitat es va arribar a un acord, segons el qual el document de text pla que ha d'omplir el docent, ha d'estar escrit en un format específic i per tant es van redactar unes normes per estandarditzar aquest document, les normes acordades són les següents:

Plantilla estàndard del fitxer d'entrada:

1. El fitxer d'entrada ha de ser del tipus text pla, amb l'extensió txt.
2. A cada línia del fitxer es defineix un mètode a verificar.
3. Per definir un mètode s'ha de seguir la plantilla següent:
   3.1. <tipus de retorn>_<nom metode>
   #param:<tipu>_<nom parametre>
   #atr:<tipus>_<nomatribut>;<tipus>_<nom>
   #ret:<nom>=<valor>

   3.2. Els tags:
      3.2.1. #-> separador entre apartats.
      3.2.2. _-> separador parella tipus, nom.
      3.2.3. param: -> apartat dels paràmetres d'entrada.
      3.2.4. atr: -> apartat de les variables del mètode.
      3.2.5. ret: -> apartat del 'return', on s'indica el nom de la variable i el seu valor inicial.
4. Implicacions del format descrit en el punt anterior:
   4.1. No tots els mètodes han de tenir paràmetres d'entrada, atributs o valor de retorn, la plantilla

serveix de guia.

4.2. Els mètodes poden tenir un o més paràmetres d'entrada igual que els atributs, per afegir-los cal utilitzar el ';' com a separador.

4.3. El primer tipus que s'introdueix fa referència al tipus del mètode i per tant al tipus de retorn que ha de generar, de manera que el valor a l'apartat ret, ha de coincidir amb aquest tipus.

5. L'aplicació elimina els espais en blanc entre dades i les línies en blanc, per això l'únic que cal assegurar és que la definició d'un mètode estigui contingut en la mateixa línia, obviant la visualització que es fa d'un fitxer de text pla, l'important és no afegir 'ENTER' al mig de la declaració d'un mètode.

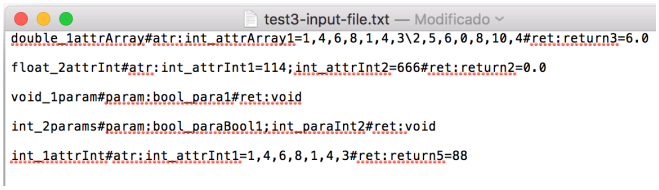

Figura 4: Fitxer de proves utilitzat durant el desenvolupament.

A la Figura 4 podem veure un exemple de quin és el format que ha de seguir el fitxer per tal de poder ser interpretat per l'aplicació, la idea principal és poder simplificar aquesta lectura, ja que un cop s'han de processar les dades, si no se segueix un estàndard, és molt fàcil no obtenir el resultat esperat.

5.2 Eines

Des del punt de vista tècnic, s'ha fet servir IntelliJ IDEA de JetBrains com entorn de desenvolupament (IDE), principalment perquè és una eina fàcil d'utilitzar, intuïtiva i amb una funcionalitat d'ajuda que permet explotar totes les funcionalitats de l'entorn.
Visual Studio Code per visualitzar els exemples proporcionats pel tutor del TFG.
Per tal de reflectir els anàlisis obtinguts de les recerques, s'han modelat els diagrames fent servir ArgoUML.
Pel desenvolupament del software que forma l'aplicació s'ha utilitzat Java com a llenguatge de programació.
Pel que fa a la documentació, s'ha utilitzat el repositori digital de documents de la UAB (DDD) [14] que ofereix la universitat, per tal d'obtenir una referència i fins i tot per un enriquiment personal.

5.3 Implementació

Un cop analitzada i dissenyada l'estructura que farà de pilars del desenvolupament, es va procedir a la implementació, la qual s'explica a continuació. Com ja s'ha esmentat en l'apartat anterior, per desenvolupar l'aplicació s'ha fet servir Java i per tant s'ha obtingut suport de la documentació de la API de Java[15]. Per tal de poder començar a treballar es va implementar l'estructura dissenyada en UML i sorgida de l'anàlisi previ, de tal manera que a poc a poc, a mesura que s'anava desenvolupant el codi, s'anava refactoritzant l'estructura de classes inicial fins que finalment l'estructura definitiva es veu reflectida a la figura 4 a l'apèndix 3 [A3] i s'explicarà a continuació junt amb el procés de desenvolupament.

Una de les primeres decisions que es va prendre va ser la capacitat de l'aplicació de buscar el fitxer mitjançant comandes per terminal i a partir d'una petita finestra dissenyada amb Swing Components, propis de la màquina virtual de Java [16], un cop obtingut el path i nom del fitxer d'entrada, el següent pas implicava un parell de funcionalitats i decisions importants, la primera d'elles l'obertura i processament d'aquest fitxer, que es realitza des de la classe Input, per portar-ho a terme es va triar fer servir la funcionalitat de bufferedreader [17], que era la més polivalent i la que permetia treballar més dinàmicament.
Per tal de poder treballar en els diferents sistemes operatius s'ha fet servir una funcionalitat que permet obtenir el separador de fitxers del sistema, ja que en MS-DOS va en una direcció (contrabarra) i en Unix va en la direcció contrària (barra)[18].

Un cop llegit i emmagatzemat el fitxer, calia processar la informació obtinguda del fitxer, per això es va dissenyar la classe Splitter, encarregada de realitzar aquest pas, dins la qual trobem un sol mètode que s'encarrega de separar el fitxer a partir dels separadors [19] que es van acordar, fent que la lectura fos un estàndard per facilitar així aquest tractament, aquestes dades es guarden com a objectes de Java, la qual es va triar com la més adequada per em-magatzemar aquesta informació, a causa del fet que es s'ha d'anar construint aquesta estructura conforme es va llegint la línia del fitxer que descriu el mètode, es per això que la distribució final d'objectes presenta 4 classes que juntes conformen el resultat final esperat, aquestes classes són les següents: Method, Parameter, Attribute i Result. Per entendre aquesta distribució cal tenir en compte que el que busca aquest projecte és testejar mètodes/funcions, per això la base es l'objecte Method, el qual està format per un nom com a mínim i pot contenir, paràmetres d'entrada, representats per la classe Parameter, en la qual trobem la definició de paràmetre mitjançant un nom i el seu tipus de dada associat, per altra banda un mètode pot contenir atributs propis, en aquest cas representats per la classe Attribute que està definida amb un nom, tipus de dada associat i una llista per als valors, ja que aquest atribut pot ser un array que esta format per un conjunt de valors i finalment la classe Result que fa referència al retorn que emet la funció (mètode) si és que en te.

A causa d'aquesta complexitat estructural s'ha triat l'objecte com a base de construcció del codi, el que ha facilitat la creació de la mateixa estructura, però alhora a afegit un punt de complexitat a l'hora de tractar les dades, de tal manera que l'objectiu inicial a anat quedant reduït al mínim. Per altra banda la complexitat de la classe Splitter ha anat augmentant considerablement i a anat reduint en-

cara més la possibilitat d'assolir l'objectiu inicial per complet, ja que per establir un tractament de fitxer cal acotar molt el format d'entrada, evitant al màxim la desviació de l'estàndard, ja que si no es comencen a obtenir resultats no esperats.
El fet és que per tal de generar codi, es requereix alguna cosa més que un fitxer d'entrada amb certa informació més bàsica o més complexa, ja que si requerim un fitxer amb moltes dades, al final el que s'està produint és un desenvolupament mitjançant la redacció que al final desemboca un software que es basa en el tractament d'un fitxer que s'acaba convertint en una estructura bàsica del que s'espera i si fem servir un fitxer amb menys dades o les mínimes dades, al final s'obtenen un conjunt de capçaleres referents als mètodes. Per aconseguir generar codi es requereix d'unes eines, ja esmentades anteriorment, molt més avançades que un fitxer i un software que tracti aquest fitxer, eines com IA o aprenentatge computacional les quals són capaces d'aprendre a partir d'uns bancs de dades i posteriorment són capaces de crear, en aquest cas línies de codi. El fet és que per tal de treballar amb aquestes eines es requeria una preparació prèvia basada en l'anàlisi i la viabilitat de l'ús d'aquestes, que no s'ha pogut assolir i per tant a causa de la falta de temps, ja que el desenvolupament d'aquest projecte té uns recursos limitats de temps, no s'han pogut aplicar i s'ha hagut de reformular l'objectiu inicial, de manera que el software permet generar a partir del fitxer d'entrada, una plantilla que serveix com a base per desenvolupar manualment les funcions que permeten puntuar els exercicis, que són les classes a testejar.

És per això que els resultats obtinguts no concorden en la seva totalitat amb l'objectiu plantejat a l'inici, tot i això s'ha adquirit una experiència tant en l'àmbit tècnic, ja que s'ha treballat a fons amb la capacitat dels objectes (POO, programació orientada a objectes), com en anàlisis de situacions i problemes sorgits en un projecte d'enginyeria. Per altra banda l'objectiu inicial s'ha anat reformulat fins a arribar a obtenir uns resultats d'acord amb aquestes modificacions.

Per continuar amb la implementació, quan les dades que conté el fitxer estan tractades, ja tenim informació que hem de convertir en dades de sortida, per fer això s'ha creat una classe dedicada exclusivament a aquesta funcionalitat, la qual és la que ha portat més feina i probablement es pot optimitzar en una futura implementació, però aquesta qüestió ja es tracta en un apartat posterior. La funcionalitat consta principalment de dos punts a tractar, la més important és com gestionar les dades obtingudes del fitxer, principalment com identificar-les per poder ordenar-les de manera coherent, per tal d'obtenir la sortida desitjada, aquesta gestió s'ha convertit en el coll d'ampolla d'aquest projecte per culpa de la complexitat per determinar on comença i on acaba cada mètode a testejar, el fet és que finalment es fa servir un caràcter separador que permet identificar fàcilment aquests límits. I el segon punt a tractar és el tipus de dades que l'aplicació permet gestionar, ja que la quantitat de tipus de dades que permet tractar un llenguatge com C++ és elevat i per tant si llegim del fitxer, hem d'establir quins tipus de dades pot tractar i això ha de ser el mes dinàmic possible, ja que el codi s'ha de poder ampliar en un futur, es va acordar que no seria estàtic, per aquesta raó es va procurar trobar l'opció més dinàmica i finalment es va decidir treballar amb una classe que actues d'objecte a testejar, en aquest cas mètodes o funcions i per tant adquirís les característiques d'aquests, nom, atributs, retorn, tipus de dades. I finalment només falta el punt final on es desa tota aquesta informació en un fitxer de sortida amb extensió cpp, automàticament el fitxer de sortida s'enmagatzema a l'arrel del projecte per qüestions pràctiques. Es va acordar que aquesta extensió es podria ampliar fins a desar aquestes dades en diferents tipus de llenguatge com per exemple Python.

Tot això tractat amb les funcions de try-catch[20], que permeten capturar els errors en els punts on succeeixen, punts que nosaltres hem d'implementar, identificant el possible error i envoltant-l'ho amb el try-catch, de tal manera que si la funció falla, el try-catch recull l'error i mostra per pantalla el missatge per defecte o bé el que tenim creat.

6 RESULTATS

Tal com s'ha esmentat a l'apartat de la implementació, durant el desenvolupament s'ha anat refactoritzant el codi i al seu temps s'han anat reformulant els objectius inicials de tal manera que els resultats obtinguts avui dia es basen en la generació automàtica de la base de les classes de test. Aquesta base generada automàticament permet establir un punt de partida sòlid i necessari per continuar desenvolupant les classes de test.

Per un fitxer d'entrada com el següent:

test2-input-file.txt

```
float_attrInt#atr:int_attrInt1=114;int_attrInt2=666#ret:return2=20.0
```

Figura 5: Exemple fitxer d'entrada.

El resultat per terminal té el següent aspecte:

```
------------------ TestGenerator ------------------
------------------------- by Ricard Bernal Petit

float attrInt() {
    int attrInt1 = 114;
    int attrInt2 = 666;
    float return2 = 0.0;
    return return2;
}
```

Figura 6: Resultat per terminal.

Si observem la figura 6 veiem com es representa un format C++ característic.

I el resultat en format cpp es el següent:

```cpp
#include <iostream>

float attrInt()
{
    int attrInt1 = 114;
    int attrInt2 = 666;
    float return2 = 0.0;
    return return2;
}

```

Figura 7: Fitxer de sortida en format C++.

Tal i com s'observa a la figura 7, el resultat final es fidel a una estructura C++ estandard. La captura ha estat extreta del Visual Studio Code i el compilador no mostra cap error.

8 Futures Implementacions

Pel fet que només s'ha pogut desenvolupar un sol pas de l'aplicació amb entorn gràfic, una futura implementació estaria lligada a aquesta qüestió, de manera que es podria desenvolupar l'aplicació íntegrament amb el seu entorn gràfic propi, de tal manera que es podria fins i tot distribuir de forma comercial, a més es pretenia empaquetar en un executable, però finalment s'ha entregat el projecte java, per tant en futures implementacions es pretén empaquetar-ho.
I per altra banda, només s'ha pogut implementar perquè retorni un fitxer de sortida amb extensió cpp, però l'abast del projecte incloïa la possibilitat d'ampliar fins a poder retornar el fitxer preparat per algun altre tipus de format, com per exemple Python, a causa de la falta de temps aquest apartat es va quedar fora i per això s'inclou com a futura implementació.

9 Conclusions

Pel fet que es tracta d'un projecte ambiciós i nou des de zero, ha costat ajustar-se al resultat esperat, ja que la dificultat per generar codi automàticament és elevada i el temps per desenvolupar el software és limitat, amb tot i això s'ha pogut assolir un objectiu bàsic de funcionament demanat a l'aplicació, el qual a sorgit d'anar reformulant la idea incial plantejada, ja que la dificultat que anava adquirint el projecte durant el desenvolupament comprometia l'objectiu incial.
El fet que la dificultat hagi està elevada a forçat un aprenentatge dinàmic i àgil, que ha augmentat l'interès a voler acabar el projecte amb tota la interfície gràfica i millorar la precisió de creació de tests.
Per altra banda el projecte a permès obtenir una visió realista de les dificultats de portar a terme el desenvolupament d'un software que té com a objectiu satisfer un client i ha demostrat les habilitats que cal que tingui un enginyer alhora de resoldre problemes i situacions difícils.

Agraïments

Aquest projecte no seria possible sense el suport incansable de la meva mare i el meu pare, és per això que els hi vull dedicar el projecte.
Gràcies a Ernest Valveny Llobet del departament de computació i tutor encarregat d'aquest projecte, per la documentació aportada, així com la seva atenció quan ha sigut requerida.
Finalment donar les gràcies als companys de Roche Diagnostics que m'han donat suport i m'han brindat ajuda sempre que l'he necessitat.

Bibliografia

[1] Universitat Autònoma de Barcelona, Grau en Enginyeria Informàtica. El grau en xifres, 2019. Recuperat de: https://www.uab.cat/web/estudiar/llistat-de-graus/qualitat/el-grau-en-xifres/enginyeria-informatica-1297754209672.html?param1=1263367146646

[2] Universitat Autònoma de Barcelona, Grau en Enginyeria Informàtica. Repositori ampliat d'indicadors, 2019. Recuperat de: http://siq.uab.cat/siq_public/titulacio/2502441/

[3] Get Started With Behavior Driven Development. TechMagic. (20 de setembre 2017). Recuperat de: https://medium.com/@TechMagic/get-started-with-behavior-driven-development-ecdca40e827b

[4] Gherkin Reference. Recuperat de: https://cucumber.io/docs/gherkin/reference/

[5] SeleniumHQ Browser Automation.What is Selenium? Recuperat de: https://www.seleniumhq.org/

[6] Impacto de la Inteligencia Artificial en el Desarrollo de Software. Tecnologia. Inteligencia artificial (24 abril 2019). Recuperat de: https://sg.com.mx/revista/56/inteligencia-artificial-desarrollo-software

[7] ¿Cómo está revolucionando la inteligencia artificial el desarrollo de software?. Tecnologia. Inteligencia artificial (24 abril 2019). Recuperat de: https://www.bbvaopenmind.com/tecnologia/inteligencia-artificial/como-esta-revolucionando-la-inteligencia-artificial-el-desarrollo-de-software/

[8] Procés de desenvolupament de programari. (s.f). A Wikipedia. Recuperat de: https://ca.wikipedia.org/wiki/Proc%C3%A9s_de_desenvolupament_de_programari#Models_de_desenvolupament_de_programari

[9] Proyectos Agiles. Qué es SCRUM. Recuperat de: https://proyectosagiles.org/que-es- scrum/

[10] Celia Aguirregabiria García. Modelo de Negocio: Metodología Canvas. [Blog] Generation. Recuperat de: https://www.generation.org/modelo-de-negocio-metodologia- canvas/

[11] Softeng. Metodología Scrum para desarrollo de software - aplicaciones compleixes. Recuperat de: https://www.softeng.es/es-es/empresa/metodologias-de- trabajo/metodologia-scrum.html

[12] Jose Ruiz Cristina. (26 de novembre del 2015). [Blog] Paradigmadigital. Qué es DevOps (y sobre todo qué no es DevOps). Recuperat de: https://www.paradigmadigital.com/techbiz/que-es-devops-y-sobre-todo-que-no-es-devops/ [7] Devops. Where the world meets DevOps. Recuperat de https://devops.com/

[13] Devops. Where the world meets DevOps. Recuperat de https://devops.com/

[14] Repositori digital de documents de la UAB (DDD). Recuperat de: https://ddd.uab.cat/collection/escengtfg?ln=ca

[15] JavaTM Platform, Standard Edition 7. API Specification. Recuperat de: https://docs.oracle.com/javase/7/docs/api/

[16] Oracle Java Documentation. Lesson: Using Swing Components. How to use FileChoosers. Recuperat de: https://docs.oracle.com/javase/tutorial/uiswing/components/

[17] Mkyong. (1 de desembre 2008). [Blog] How to read file in Java – BufferedReader. Recuperat de: https://www.mkyong.com/java/how-to-read-file-from-java-bufferedreader- example/

[18] Java2. Java I/O How to - Get file system separator. Recuperat de: http://www.java2s.com/Tutorials/Java/IO_How_to/File_System/Get_file_system_separator.htm

[19] Tutorials Point. Simply easy learning. Java.lang.String.contains() Method. Recuperat de: https://www.tutorialspoint.com/java/lang/string_contains.htm

[20] The world's largest web developer site. Java Exceptions - Try...Catch. Recuperat de: https://www.w3schools.com/java/java_try_catch.asp

APÈNDIX

A1. DIAGRAMES UML

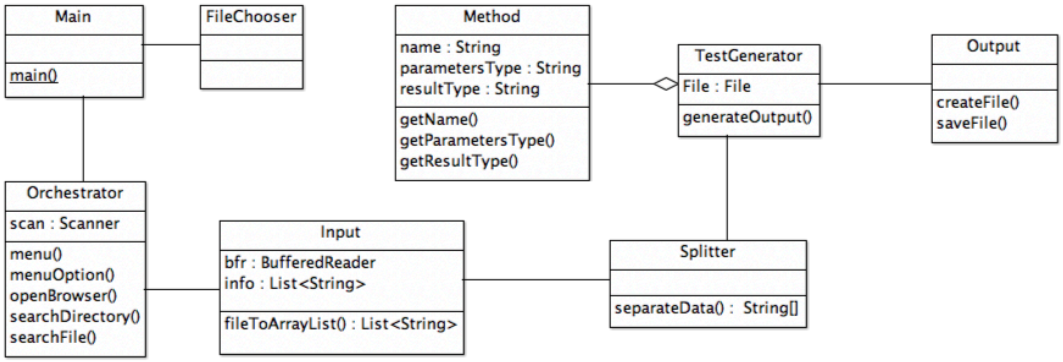

Figura 2: Diagrama de classes UML inicial

A2. FASES I ACTIVITATS

Fase	Descripció	Tasques / Activitats
Planificació inicial del projecte	Planificació inicial, el·licitació de requisits, estudi de la viabilitat dels objectius marcats.	• Redactar requisits • Anàlisis funcional / no funcional • Estudi de viabilitat • Planificar activitats
Obtenció d'informació	Recerca massiva d'informació útil i necessari per al desenvolupament del *software*.	• Recerca durant un període de temps mitjà • Recerca continua i activa
Anàlisis	Definir i comparar diferents possibles models.	• Presa de dedicions estructurals
Disseny	Modelatge de l'estructura inicial del *software*.	• Diagrama de classes • Diagrames complementaris
Implementació / Verificació del *software*	Desenvolupament del programari paral·lelament amb la verificació i validació del mateix.	• Desenvolupament • Verificació • Validació

Taula 1: Fases i activitats

A3. ESTRUCTURA DEL PROJECTE

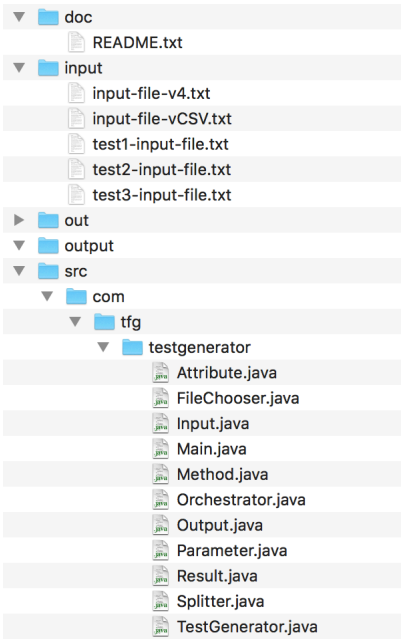

Figura 3: Estructura del projecte