Projeto (SelRA) de GCS - Grupo 10

Diogo Paulo da Costa Pereira - A84092
Universidade do Minho

Gonçalo Rodrigues Pinto - A83732
Universidade do Minho

Luís Francisco Mendes Lopes - A85367
Universidade do Minho

Luís Mário Macedo Ribeiro - A85954
Universidade do Minho

(4 de Janeiro de 2021)

Resumo

O presente relatório descreve o processo de implementação de um sistema cujo objectivo foi sensibilizar e motivar os alunos para a criação e desenvolvimento deste, de forma a dar apoio ao professor que lhe permita escolher um recurso de aprendizagem adequado a um dado aluno para ensino de um determinado conceito de programação, de modo a tentar maximizar a eficácia do processo de ensino/aprendizagem num curso de Introdução à Programação. O presente trabalho teve como tema *Domain-Specific Language - DSL* que controla, gere e regula o comportamento do sistema desenvolvido. Este assume diversos comportamentos de acordo com dados fornecidos e necessários. Deste modo, desenvolveu-se um sistema baseado numa *DSL*, tendo como base algumas pesquisas científicas de forma a tornar o sistema mais real possível. De forma reconhecer esta linguagem de domínio específico criada utilizou-se o poderoso gerador de análise *ANTLR* para efectuar a análise léxica e sintáctica.

1 Introdução

No 1º semestre do 4º ano do Curso de Engenharia Informática da Universidade do Minho, a Unidade Curricular, enquadrada no perfil de Processamento de Linguagens e Conhecimento, denominada por Gramáticas na Compreensão de Software tem como objectivo desenvolver a capacidade dos estudantes para desenvolver especificações da sintaxe/semântica de linguagens e problemas em geral com gramáticas, aumentar as competências de geração de programas usando fer-

ramentas automáticas baseadas em gramáticas, fortalecer habilidades de construção de *front-ends* poderosos para a análise de linguagens de programação; Ampliar a aptidão dos alunos concepção e implementação de estruturas de dados complexas para representação intermédia da informação extraída da análise do código, expandir os conhecimentos ao nível da criação de representações visuais adequadas à compreensão clara do conhecimento complexo detido, amplificar o desenvolvimento de software como uma tarefa de transformação de programas e/ou especificações em implementações eficientes, também têm como objectivo desenvolver a capacidade para utilizar técnicas de transformação de programas para optimizar programas, efectuar *debugging* de programas e melhorar a estrutura dos programas e ainda fortalecer a aptidão de desenvolver ferramentas para ajuda à compreensão de código.

De forma a atingir esses objectivos foi nos proposto o desenvolvimento de uma linguagem de domínio especifico projectada de forma a implementar um sistema de apoio ao professor que lhe permita escolher um recurso de aprendizagem adequado a um dado aluno para ensino de um determinado conceito de programação de modo a tentar maximizar a eficácia do processo de ensino/aprendizagem num curso de Introdução à Programação. Esta linguagem foi uma maneira de expressar a solução para problema proposto de maneira mais simples e directa que a utilização de uma linguagem de programação específica.

Uma das muitas motivações foi fornecer uma forma rápida e fácil de obter um determinado recurso para o professor ensinar um tema da nossa área de trabalho, Programação, efectuando uma determinada questão ao nosso sistema pois partilhamos da mesma opinião e valores da Associação para o Ensino da Computação para tal partilhamos a seguinte mensagem ...

> "O pensamento humano, sempre em evolução, alterou-se radicalmente com a escrita e ainda mais com a computação. Computação diz respeito a métodos de resolução de problemas e de construção de sistemas que permitem compreender e actuar sobre o mundo natural e social. Jovens capazes de "pensar computacionalmente", estarão, assim, mais bem equipados e preparados para conceber, compreender e usar tecnologias baseadas em computadores, hoje e no futuro."

2 Contextualização

Tal como foi referido previamente foi desenvolvido uma linguagem, sendo necessário descrever a estrutura das frases e palavras em linguagem natural. Para tal foi utilizado uma **Gramática de Atributos** pois permite a definição local (sem a utilização) do significado de cada símbolo num estilo declarativo além disso a atribuição do valor de cada símbolo é local.

Esta gramática é definida por símbolos terminais e não terminais. Os símbolos terminais têm atributos intrínsecos, que descrevem o valor léxico associado a cada um deles. Por outro lado, os símbolos não-terminais são associados com atributos genéricos, através dos quais se pode sintetizar informação semântica,

ou herdar informação semântica.

Esta gramática utilizada gera uma árvore de sintaxe abstracta em memória quando processadas, para interligar produções e os conjuntos de não terminais e terminais, isto revela-se uma grande vantagem no sentido em que apenas é representado os símbolos com carga semântica, e são descartados os nodos que correspondem a palavras reservadas.

Deste modo temos à nossa disposição, através desta gramática, várias vantagens e ferramentas para implementar o nosso de sistema de recomendação de recursos ao professor mediante o conceito de ensino que pretende ensinar a um determinado aluno.

3 Conceitos básicos em SelRA

Como foi apresentado anteriormente desenvolveu-se uma *DSL*, designada *SelRA*, assente numa **Gramática de Atributos** de forma a dar apoio ao professor. Para isto, foi necessário identificar alguns conceitos importantes.

3.1 Perfis

Um dos conceitos básicos nesta linguagem foi uma representação de todas as características que podemos encontrar num determinado aluno, contudo as mesmas características não descrevem propriamente um determinado recurso. Por isso, o nosso grupo focou a nossa pesquisa no sentido de encontrar os diversos perfis que possam existir numa sala de aula e em cada um deles descrever características de uma pessoa que se enquadra nesse perfil. Contudo, é lógico que uma pessoa possa partilhar características de diferentes perfis tal como os recursos serem focados para mais que um determinado perfil.

3.2 Recursos

Como foi referido esta linguagem pretende recomendar recursos de ensino logo é essencial descrever esses mesmos. Para tal descrição decidiu-se caracterizar um recurso algo que possua um identificador único, um tipo, uma breve descrição, um intervalo de idade qual este recurso é ideal de ensinar e um conjunto de características emocionais e de aprendizagem dos praticantes que mais explora.

3.3 Alunos

Tal como o prévio conceito, este presente conceito também é essencial para o sistema, pois são o público alvo do professor a quem se pretende apoiar e para tal foi necessário descrever um aluno possuindo um identificador único, um nome, uma idade e características emocionais/sociais que interferem na aprendizagem.

3.4 Conceitos

Seguidamente existe outra ideia importante de ser descrita que é o conceito de programação que o professor pretende ensinar. Neste sentido, representou-se como algo com um determinado identificador e uma breve descrição.

3.5 Ensinamentos

Posteriormente, existe outra noção importante que foi descrita na linguagem construída que foi a ligação entre os conceitos de Recursos e Conceitos, de modo a descrever quais os conceitos de programação que transmite um determinado recurso.

3.6 Questões

Por fim, outro conceito básico fulcral para esta linguagem e consequentemente para o sistema de apoio é a ligação entre o determinado conceito de programação ao qual se pretende ensinar a um aluno em específico.

4 A Proposta

Com base nos conceitos básicos na linguagem desenvolvida *SelRA* apresentados o sistema deve propor os recursos mais adequados para tal propomos a seguinte resolução:

```
selRA → perfis HIFEN
        recursos HIFEN
        alunos HIFEN
        conceitos HIFEN
        ensinamentos HIFEN
        questoes PONTO ;

perfis → perfil
       ( PONTOVIRGULA perfil )* ;
perfil → desc DOISPONTOS LPAR carateristicas RPAR ;

desc → STR ;

carateristicas → carateristica ( VIRGULA carateristica )* ;
caracteristica → desc ;

recursos → recurso ( PONTOVIRGULA recurso )* ;
recurso → idrecurso tipo desc intervaloidade LPAR carateristicas RPAR ;
idrecurso → IDR ;
tipo → STR;
intervaloidade → LPAR idademin HIFEN idademax RPAR ;
idademin → idade ;
```

```
idademax → idade ;

idade → NUM ;

alunos → aluno ( PONTOVIRGULA aluno )* ;
aluno → idaluno nome idade LPAR carateristicas RPAR ;
idaluno → IDA ;
nome → STR ;

conceitos → conceito ( PONTOVIRGULA conceito )* ;
conceito → idconceito desc ;
idconceito → IDC ;

ensinamentos → ensinamentos ( PONTOVIRGULA ensinamentos )* ;
ensinamento → idrecurso ENSINA idconceito ;

questoes → questao ( questao )* ;
questao → ENSINAR idconceito AO idaluno PONTOINTERROGACAO ;
```

É de notar que as regras "`desc →STR`" , "`idrecurso →IDR`" , "`tipo →STR`" , "`idade →NUM`" , "`idaluno →IDA`" , "`nome →STR`" e "`idconceito →IDC`" derivam em símbolos terminais.

É de salientar também que as palavras que se encontram em letras maiúsculas tais como "HIFEN", "PONTO", "PONTOVIRGULA", "LPAR", "RPAR", "STR", "VIRGULA", "DOISPONTOS", "IDR", "NUM", "IDA", "IDC"e "PONTOINTERROGACAO"são símbolos terminais, conceito apresentado anteriormente, ainda existem algumas palavras reservadas como por exemplo as palavras "ENSINA", "ENSINAR"e "AO"são utilizadas de forma a tornar a linguagem o mais natural possível.

Em relação ao léxico desta proposta, as palavras reservadas possibilitamos de o utilizador escrever alternadamente maiúsculas e minúsculas, ao nível dos sinais de pontuação o nome dos símbolos terminais representa exactamente esse sinal, a nível dos identificadores mediante do símbolo terminal que seja começa pela última letra desse mesmo combinando com a possibilidade do número ("`IDR : 'R'[0-9]+ `" apenas nos alunos temos dupla possibilidade "`IDA : ('A'|'PG')[0-9]+ `"), em relação ao símbolo "NUM"origina um conjunto de números e o símbolo "STR"foi construído de forma a possibilitar mais que uma palavra através do recurso das aspas.

Nesta linguagem tentamos ser o mais concisos, claro e práticos possíveis por isso tentamos seguir a mesma lógica de representação dos conceitos básicos acima apresentados, onde representamos um determinado conceito no seu plural. Desta forma podemos descrever vários conceitos (no mínimo tem que existir um) separados por pontos e vírgulas reforçando a ideia de clareza, posteriormente cada conceito básico possui os atributos identificados referidos acima, onde cada um destes derivam de alguma forma num símbolo terminal inclusive se for atributo mais composto, por exemplo as características emocionais/sociais de um determinado aluno é efectuado o mesmo processo de raciocínio.

5 A Implementação

Nesta secção iremos discutir a implementação do sistema de apoio como respectivos testes efectuados utilizando as particularidades que uma **Gramática de Atributos** fornece. O sistema foi desenvolvido utilizando o gerador *ANTLR* para processar a linguagem acima apresentada, porque geralmente é usado para construir ferramentas e estruturas. Este gerador forneceu-nos acesso a primitivas de processamento de linguagem como *lexers*, gramáticas e analisadores, e de forma a ter acesso a estas efectuou-se uma ligação à linguagem de programação, Java, utilizada para implementar o sistema. Processo de desenvolvimento foi divido em diferentes etapas que seguidamente irão ser apresentadas.

5.1 1º Etapa - Construção de classes

Como foi referido efectuou-se uma programação orientada aos objectos, nesse sentido criou-se objectos que representam a informação que se pretende armazenar e processar. As classes construídas basicamente são os conceitos básicos apresentados com os respectivos atributos.

```
class Perfil{
        String nome;
        ArrayList<String> carateristicas;
}
class Recurso{
        String id;
        String tipo;
        String descricao;
        int idademin;
        int idademax;
        ArrayList<String> perfis;
}
class Aluno{
        String id;
        String nome;
        int idade;
        ArrayList<String> carateristicas;
}
class Conceito{
        String id;
        String descricao;
}
class Ensinamento{
        String idRecurso;
        String idConceito;
}
```

Desta forma em cada classe conseguimos guardar a informação fornecida sobre os mais diversos conceitos.

5.2 2º Etapa - Construção de estruturas

Partindo das vantagens que a linguagem utilizada fornece utilizou-se as estruturas/colecções fornecidas de forma armazenar mais que um determinado conceito, isto revelou-se uma enorme vantagem pois temos várias possibilidades de escolha mediante o objectivo que queremos armazenar, por exemplo se quisermos uma lista de um determinado objecto utilizou-se uma colecção *ArrayList<Objecto>* se quisermos mapear um determinado objecto através dos identificados utilizou-se um *HashMap<Key,Objecto>* , entre outras possibilidades. Desta forma para armazenar a informação criou-se as seguintes estruturas

```
HashMap<String,Perfil> mapPerfis ;
HashMap<String,Recurso> mapRecursos ;
HashMap<String,Aluno> mapAlunos ;
HashMap<String,Conceito> mapConceitos ;
ArrayList<Ensinamento> listEnsinamentos ;
```

Assim, juntando a primeira etapa da implementação do sistema com as estruturas conseguiu-se armazenar a informação.

5.3 3º Etapa - Povoação das estruturas e Detecção de Erros

Nesta etapa foi necessário recorrer às particulares que a **Gramática de Atributos** fornece, mais propriamente ao factor de sintetizar informação semântica ou herdar informação semântica através dos símbolos não terminais através da árvore de sintaxe abstracta em memória que é construída, ou seja, em cada símbolo não terminal que representa um conjunto de um determinado conceito básico sintetiza-se a estrutura respectiva apresentada previamente após o conceito básico herdar essa estrutura efectuar a povoação ou detecção dos erros mediante os atributos que fazem parte e após este processo sintetiza uma cópia da estrutura herdada para o conjunto desses conceitos. De seguida apresentamos apenas o exemplo de povoação e detecção de erros para os conceitos de programação com uma imagem ilustrativa, os restantes conceitos básicos possuem a mesma lógica de sintetização e hereditariedade.

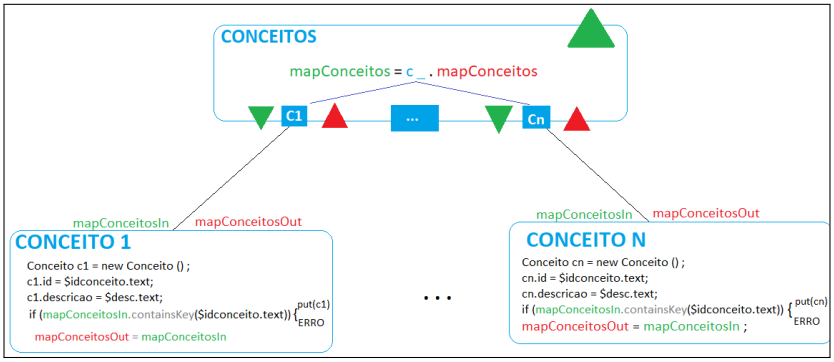

Figura 1: Sintetização e Hereditariedade de forma a povoar as estruturas com as classes criadas como também detecção de erros.

```
conceitos
    returns[HashMap<String,Conceito> mapConceitos]
    @init { $mapConceitos = new HashMap<String,Conceito>(); } :

    c1=conceito[$mapConceitos]  {$mapConceitos= $c1.mapConceitosOut;}
    ( PONTOVIRGULA c2=conceito[$mapConceitos]
                                {$mapConceitos = $c2.mapConceitosOut;} )*;

conceito[HashMap<String,Conceito> mapConceitosIn]
    returns[HashMap<String,Conceito> mapConceitosOut]
    @init { $mapConceitosOut = new HashMap<String,Conceito>(); } :
        idconceito desc
            {Conceito c = new Conceito();
            c.id = $idconceito.text;
            c.descricao = $desc.text;
            if(!($mapConceitosIn.containsKey($idconceito.text))){
                $mapConceitosIn.put($idconceito.text,c);
            }
            else{
                System.out.println("ERRO: ... ");
            }
            $mapConceitosOut = $mapConceitosIn; };
```

Como pode ser observado por este processo conseguimos armazenar as classes criadas nas estruturas respectivas tudo isto é possível através da passagem através dos vários nós da árvore criada.

5.4 4º Etapa - Implementação do algoritmo de recomendação

De forma a que o professor tenha acesso aos recursos de aprendizagem para um determinado aluno é necessário que efectua uma questão na linguagem apresentada. Nesta questão é executado o nosso algoritmo criado para oferecer os

recursos agrupados por percentagem de correspondência. De seguida iremos apresentar os vários passos do mesmo, tal como a etapa anteriormente. O facto da gramática escolhida ter criado uma árvore foi essencial pois todas as estruturas previamente apresentadas e povoadas são herdadas por uma questão.

Passo 1 : Confirmou-se que o identificador do conceito de programação e do aluno introduzidos na questão existem ;

Passo 2 : Acedeu-se ao mapAlunos herdado de forma a obter o Aluno ao qual se pretende ensinar (através do identificador do aluno na questão), consequentemente obteve-se as características deste Aluno;

Passo 3 : Percorreu-se a listEnsinamentos de forma a filtrar os recursos, ou seja, comparou-se o identificador do conceito a ensinar com o respectivo atributo da classe Ensinamento, caso exista acedeu-se ao mapRecursos de forma a obter o Recurso através do outro atributo de Ensinamento, verificou-se ainda se a idade do aluno encontrado no passo anterior encontra-se no intervalo de idade ideal do recurso ;

Passo 4 : Identificou-se os perfis em que o Aluno enquadra-se, para tal acedeu-se ao mapCaracteristicas verificando se uma determinada característica está na lista de características de um determinado Perfil ;

Passo 5 : Associou-se a cada Recurso filtrado (passo 3) a maior percentagem encontrada de correspondência, para tal fim foi obtido as características de um Recurso. Por uma questão de maior rigorosidade efectuou-se o passo 4 mas em relação ao Recurso. Posteriormente, com recurso a expressões *lambda* criou-se uma lista de perfis comuns ao Recurso iterado naquele momento e o Aluno cujo objectivo é ser ensinado. De seguida, caso a lista possua perfis é calculado a percentagem de *matching* (o tamanho da lista encontrada sobre lista de características normalizadas do Aluno), caso esta percentagem seja maior que uma outra previamente encontrada o Recurso em causa corresponde a esta nova;

Passo 6 : Por fim agrupou-se os recursos por percentagem de cada Recurso de forma a facilitar a apresentação dos resultados;

5.5 5º Etapa - Apresentação dos resultados

Nesta última etapa, após todo o processo de construção, povoação e implementação do sistema foi necessário apresentar os resultados obtidos conforme as nossas convicções : breve, claro e sucinto. Para tal recorreu-se aos conhecimentos da outra única curricular do perfil onde esta cadeira se enquadra e foi construído um servidor em **Node.js** que fornece páginas HTML mediante pedidos efectuado no *browser*. É de salientar que as estruturas construídas originam um ficheiro *JSON* pois com o auxílio do *JSON-Server* facilitou a apresentação da informação.

6 Conclusão

O presente relatório descreveu, de forma sucinta, um sistema de apoio ao professor que lhe permita escolher um recurso de aprendizagem adequado a um dado aluno para ensino de um determinado conceito de programação de modo a tentar maximizar a eficácia do processo de ensino/aprendizagem num curso de Introdução à Programação, com base numa linguagem de domínio específico analisada e processada com *ANTLR*, ligada à linguagem *Java*.

Em suma, após implementado o sistema descrito previamente foi possível obter os seguintes resultados:

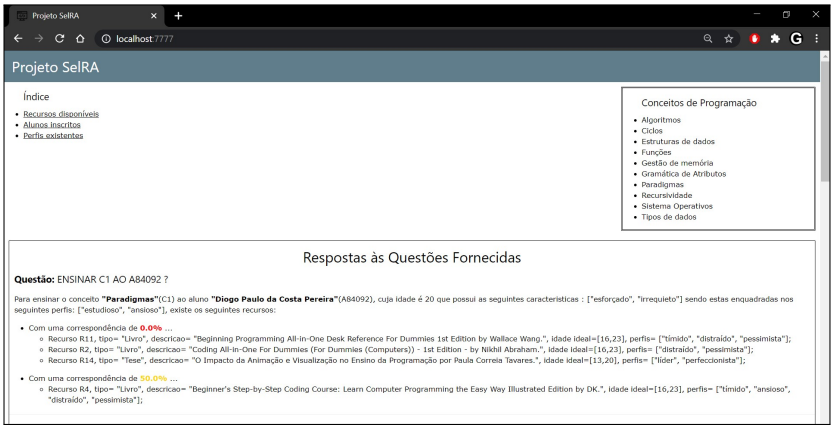

Figura 2: Resultados obtidos

Como podemos observar na figura 2, o servidor criado corre na porta 7777 onde na página inicial apresenta os conceitos de programação que introduzidos, ligações para a visualização dos recursos, perfis e alunos introduzidos, onde na página dos alunos foi construído um gráfico de barras de forma a demonstrar as características mais frequentes. Em relação às respostas apresentou-se a descrição do aluno e os recursos agrupados por percentagem juntamente ainda descreveu-se a descrição do conceito a ensinar e quais os perfis onde as características emocionais/sociais deste aluno fazem parte.

Após a realização deste trabalho, ficamos consciente das potencialidades da construção de uma linguagem de domínio específico através de uma **Gramática de Atributos** traz ao ser combinada com uma bom ferramenta como é o *ANTLR* de forma reconhecer o que se construiu aliado a uma excelente linguagem de programação como é Java.

Numa perspectiva futura poderia-se tornar o sistema mais iterativo, ou seja, o utilizador efectuava um pedido com a questão que pretendia resolver e o servidor encarregava-se de mostrar o resultado.

Consideramos que os principais objectivos foram cumpridos e sentimos também que a realização deste trabalho prático consolidou os nossos conhecimentos.

Referências

[1] Ensico. ENSINO. Acedido em 17 de Dezembro de 2020, em: https://ensico.pt/

[2] Educador360. Você sabe quais habilidades desenvolver para cada perfil de aluno? Acedido em 17 de Dezembro de 2020, em: https://educador360.com/gestao/tipos-de-alunos-na-escola/

[3] Direção-Geral da Educação. Perfil dos alunos - Direção-Geral da Educação. Acedido em 17 de Dezembro de 2020, em: https://www.dge.mec.pt/sites/default/files/Noticias_Imagens/perfil_do_aluno.pdf

[4] ANTLR. Documentação do ANTLR. Acedido em 17 de Dezembro de 2020, em: https://github.com/antlr/antlr4/blob/master/doc/index.md

[5] Tese do professor José Carlos Ramalho. Anotação Estrutural de Documentos e sua Semântica - Capítulo 2. Notações e Formalismos utilizados - 2.1. Gramáticas de Atributos. Acedido em 17 de Dezembro de 2020, em: http://www4.di.uminho.pt/ jcr/XML/publicacoes/teses/phd-jcr/src/c105.htm

[6] Opensource. What developers need to know about domain-specific languages. Acedido em 17 de Dezembro de 2020, em: https://opensource.com/article/20/2/domain-specific-languages

[7] Jetbrains. Domain-Specific Languages. Acedido em 17 de Dezembro de 2020, em: https://www.jetbrains.com/mps/concepts/domain-specific-languages/