Universidade do Minho
Escola de Engenharia

João da Cunha Coelho

Mecanismos de Raciocínio para Sistemas de Avaliação de Conhecimento

outubro de 2019

Universidade do Minho
Escola de Engenharia

João da Cunha Coelho

Mecanismos de Raciocínio para Sistemas de Avaliação de Conhecimento

Dissertação de Mestrado

Mestrado em Engenharia Informática

Trabalho efetuado sob a orientação do

Professor Doutor Orlando Manuel de Oliveira Belo

outubro de 2019

Agradecimentos

Em primeiro lugar, agradeço ao meu orientador, o professor Orlando Belo, a confiança e, sobretudo, motivação que me transmitiu ao longo da realização desta dissertação, para a qual muito contribuíram os seus ensinamentos e experiência, acompanhados de uma dedicação, disponibilidade e profissionalismo que realço.

À Mariana, minha namorada, pelo apoio, compreensão e carinho que demonstrou ao longo destes meses, o meu mais sincero obrigado.

Para os meus pais, que me ampararam, educaram e tornaram possível o culminar desta etapa da minha vida académica e pessoal, fica o meu agradecimento especial.

Uma palavra de agradecimento ainda para o Luís, meu amigo e companheiro de investigação, pelo esforço, paciência e partilha de muitas horas de entreajuda e boa disposição.

Por fim, resta-me agradecer aos meus amigos e familiares em geral, que me apoiaram durante o processo de realização desta dissertação e com os quais me orgulho de poder contar.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

Sendo certo que o recurso à tecnologia no ensino é cada vez mais notório, a utilização de sistemas informáticos de tutoria continua aquém do seu potencial, ainda que seja um tema abordado há já algumas décadas. Assim, surgiu a iniciativa *Leonardo* e o respetivo desenvolvimento de uma ferramenta computacional para sistemas de avaliação de conhecimento, com vista a ser aplicada, pelo menos, no suporte de processos de avaliação de alunos, na Universidade do Minho. De entre os módulos que caracterizam estes agentes de *software*, no contexto desta dissertação, destacam-se a base de conhecimento, o mecanismo de raciocínio e o modelo do estudante. Dado que o esforço maior recai em habilitar os tutores artificiais à adaptação, em tempo real, da avaliação ao nível de conhecimento atual dos alunos, surge a necessidade de desenvolvimento de um mecanismo de raciocínio, que seja capaz de determinar, criteriosamente, o que deve ser apresentado de seguida num dado momento avaliativo. O trabalho desta dissertação focou-se na conceção e implementação de um sistema de avaliação baseado em conhecimento para o sistema *Leonardo,* com a capacidade de ajustar de forma dinâmica, à medida da perícia e conhecimento dos estudantes alvos do processo de avaliação, o seu comportamento, acompanhando de perto a evolução do processo de aprendizagem dos estudantes. Essencialmente, neste trabalho implementou-se a "máquina" de raciocínio para o sistema *Leonardo* poder sustentar de forma efetiva a avaliação de estudantes ao longo do tempo, numa ou mais áreas do conhecimento.

Palavras-chave: Sistemas de Avaliação de Conhecimento, Motor de Inferência, Tutor Artificial, Processo de Avaliação, Mecanismo de Raciocínio, Regra de Produção, Base de Conhecimento.

Abstract

Despite the fact that the use of technology in education is progressively more noticeable, the use of computer tutoring systems stays below its potential, even though this is a subject that has been addressed for decades. Thus emerged the *Leonardo* initiative and the respective development of a computational tool for knowledge evaluation systems, aiming to be applied, at least, to support students' assessment processes at the University of Minho. Among the models that usually constitute these *software* agents, in the context of this dissertation, we focus the knowledge base, the reasoning mechanism and the student model. As the major effort is to enable artificial tutors to adapt, in real time, the assessment to the students' current level of knowledge, the need arises for the development of a reasoning mechanism capable of carefully decide what should be presented next at a certain evaluation moment. The work of this dissertation focused on the conception and implementation of a knowledge-based assessment system for the *Leonardo* system, capable of dynamically adjusting its behavior, according to the expertise and knowledge of the target students of the assessment process, keeping track of the students' learning process. Mainly, in this work the reasoning "machine" was implemented for the *Leonardo* system to effectively support student assessment over time in one or more areas of knowledge.

Keywords: Knowledge Evaluation Systems, Inference Engine, Artificial Tutor, Assessment Process, Reasoning Mechanism, Production Rule, Knowledge Base

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Os avanços tecnológicos que conduziram à Sociedade do Conhecimento imprimiram uma dinâmica de transformação no domínio da educação. Nesta sociedade em que a riqueza é produto do conhecimento, a crescente valorização da formação das pessoas leva a um aumento da procura de ferramentas informáticas – as TIC (Tecnologias da Informação e Comunicação) – capazes de educar não só na escola, como fora dela. A capacidade inteligente e adaptativa nos sistemas de *software* de apoio ao ensino, um conceito que vem sendo explorado há já algumas décadas, torna-se assim necessária e conveniente.

O processo evolutivo dos sistemas de *software* teve início na década de 50, quando o aparecimento do conceito de instrução programada, elaborado por B. F. Skinner e mais tarde denominado por CAI (Instrução Assistida por Computador), introduziu a informática na educação. Foi na Universidade de Pittsburgh, que o psicólogo norte-americano demonstrou a utilização de uma máquina de ensino, que, para além de ensinar, reforçava conhecimento de ortografia e de aritmética – "O utilizador

pode aceder a material auditivo, ouvir uma passagem tantas vezes quantas as necessárias e transcrevê-la. A máquina revela depois o texto correto. Ele pode voltar a ouvir a passagem e descobrir as origens de qualquer erro" (Skinner, 1958).

No início da década de 60, o PLATO surgiu como o primeiro sistema generalizado baseado em instruções assistidas por computador a ser desenvolvido. Estes foram anos de grande propagação e reconhecimento do conceito de instrução programada. Na altura, Skinner afirmava, numa entrevista de 1967, "não ter de todo dúvidas de que a instrução assistida (...) tomaria conta da educação" (Rutherford, 2012). Então, a Inteligência Artificial também vivia anos dourados, recebendo grandes investimentos de organizações governamentais. Inúmeros projetos foram construídos no âmbito destas tecnologias emergentes. No entanto, persistiam dúvidas sobre a eficácia das instruções programadas no ensino. Inclusive, em 1972, em nova entrevista, Skinner mostrava o seu desagrado e preocupação com o estado da educação – "Estou preocupado com a melhoria da educação. A instrução programada podia fazer uma grande diferença. A indústria, que aprecia coisas boas, usa-a extensivamente. No entanto, o seu uso em escolas primárias e secundárias, numa escala razoável, ainda está apenas no início" (Rutherford, 2012).

Para o descontentamento de Skinner certamente não contribuiu Jaime Carbonell, que em 1970 apresentou o primeiro tutor artificial – o sistema SCHOLAR (Carbonell, 1970a) – que introduziu "um novo tipo de Instrução Assistida por Computador, em muitos aspetos mais poderoso do que os existentes", procurando provar que os computadores poderiam atuar como um professor e não apenas como uma ferramenta. De facto, este foi o princípio da transição da CAI, termo que, com o decorrer das investigações sobre a representação de conhecimento num sistema inteligente, acabaria mesmo por declinar para ICAI (Instrução Assistida por Computador Inteligente), estruturalmente diferente da antecessora dada a utilização de técnicas de Inteligência Artificial e Psicologia Cognitiva para guiar o processo educativo.

O caminho desde a instrução programada ao ITS ficou completo quando Sleeman e Brown (1982) reviram o "estado da arte" dos sistemas CAI. Daí, surgiu o termo ITS (Intelligent Tutoring Systems), que podemos traduzir para Sistemas Inteligentes de Tutoria (ou simplesmente tutores inteligentes), que acentuou a nova perspetiva emergente, focada no uso inteligente dos computadores na educação. Desde então, em várias vertentes, muitos foram os estudos realizados neste âmbito. A componente lúdica dos sistemas está no centro da maioria da investigação realizada, com a pesquisa sobre o momento e a forma como é fornecido *feedback* e recomendações ao aluno, em função das debilidades reveladas durante a resolução de exercícios – veja-se, por exemplo os trabalhos (Záiane, 2002), (Anohina, 2007) e (Klašnja-Milićević *et al.*, 2011). A componente avaliativa, na qual se enquadram os diferentes métodos e critérios usados pelos tutores artificiais na avaliação do estudante (Baneres *et al.*, 2016), e o desenvolvimento de interfaces de utilizador mais amigáveis e sofisticadas, com processamento de linguagem natural, interação por voz ou adaptados à Web, são também alvos de constante investigação (Simmons, 1970), (Brusilovsky et al., 1999). Ainda que sejam muitos os esforços académicos no sentido de desenvolver, por vezes apenas conceptualmente, sistemas de tutores inteligentes, atualmente, em termos práticos, estes carecem de afirmação, continuando a ser pouco utilizados por estudantes dentro e fora do contexto de sala de aula.

1.2 Sistemas Inteligentes de Tutoria

Um ITS pode ser definido como um sistema composto por vários componentes de *software* que procuram imitar comportamentos de um tutor humano, visando fornecer *feedback* personalizado aos alunos, durante a execução das tarefas, sem a intervenção de seres humanos. Para cumprir a sua missão de forma inteligente, este tipo de ferramenta computacional necessita de uma base de conhecimento que garanta ao sistema de suporte à decisão a capacidade de armazenar, organizar e manipular a sua informação de forma ágil e funcional. Esta informação, que alimenta a base de conhecimento do tutor artificial, provém do próprio sistema, recolhida no momento da interação

com os utilizadores, e também resultante de processos de inserção de utilizadores qualificados – comparáveis aos docentes de uma instituição de ensino – que fornecem ao *software* o conhecimento do domínio. Esta é uma das várias definições possíveis para o conceito de ITS. É "um termo amplo, abrangendo qualquer programa de computador que contém alguma inteligência e pode ser usado em aprendizagem" (Freedman, Ali and McRoy, 2000), programas estes que "oferecem uma vantagem sobre os anteriores *software* CAI, que reside no facto de simularem o processo de pensamento humano num dado domínio e auxiliarem na resolução de problemas." (Fowler, 1991). Entre a comunidade de investigadores na área, é consensual considerar-se que os ITS são usualmente constituídos por quatro componentes básicos, nos quais o (1) modelo do domínio, o (2) modelo do estudante e o (3) modelo do tutor se podem considerar as três partes mais específicas e características desde os primórdios do ITS (Shute and Psotka, 1994), tendo o (4) modelo da interface de utilizador crescido em importância com o decorrer da investigação (Self, 1990) e o aparecimento da Web (Figura 1).

O primeiro destes componentes, o modelo do domínio, do inglês *Domain* ou *Expert model*, é também designado por modelo cognitivo e engloba a descrição do conhecimento e comportamentos do tutor no domínio que ensina – os conceitos, as regras, as estratégias de resolução de problemas e a deteção de erros, entre outros – a que recorre para exercer o seu acompanhamento ao aluno. Dependendo do objetivo e implementação, o modelo pode estar organizado sob módulos de conhecimento, interligados de acordo com sequências pedagógicas e dinamizados por estruturas como hierarquias, redes semânticas, ontologias ou regras de produção.

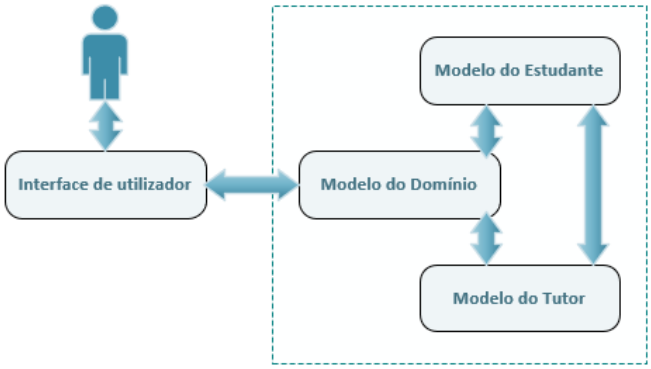


Figura 1: Os componentes principais de um ITS.

O segundo componente, o modelo do estudante, refere-se à construção do perfil de um aluno, através da definição de um modelo que contém descrições do conhecimento do indivíduo que representa, sendo inferido com base nos seus equívocos e lacunas de conhecimento na resolução dos problemas. É o componente principal no que diz respeito à propriedade de um tutor inteligente de prestar atenção ao que o estudante sabe e à forma como este evolui ao longo da interação com o sistema (Wenger, 1987). Porém, Self (1988) defende que este modelo se prende com seis características chave. São elas o ser diagnóstico e corretivo, para ajudar a identificar e erradicar equívocos do conhecimento do aluno, elaborativo e estratégico, para conseguir completar o conhecimento parcial do aluno e instruí-lo com estratégias de resolução de problemas, preditivo, para antecipar a provável resposta do aluno, e avaliativo, por uma questão de avaliar o aluno e até a si próprio.

Por sua vez, o terceiro componente, o modelo do tutor, recorre à informação mantida pelos dois modelos anteriores e seleciona qual a melhor ação e estratégia de ensino a aplicar no dado momento. É a discrepância entre o conhecimento, ou comportamento, do aluno e os previstos pelo domínio do problema que fornece um sinal ao tutor, para que possa fornecer atempadamente *feedback* ao aluno, resultando numa aprendizagem mais rápida.

O último dos modelos, o quarto componente, o modelo da interface do utilizador ou de comunicação, é o responsável por fornecer os meios para que o aluno possa interagir com o ITS e vice-versa. É caracterizado por uma interface gráfica, que necessita de compreender o aluno, seja qual for o meio de *input* de informação definido, e simultaneamente possuir forma de atuar, interagindo com o aluno. É, portanto, o canal de comunicação entre o utilizador e o sistema, estando o conteúdo dessa comunicação assente nos conhecimentos no domínio do problema – é o motor da interação entre o estudante e o tutor.

1.3 Motivação

O uso das novas tecnologias tem-se revelado bastante efetivo ao nível de qualquer tipo de processo de aprendizagem, quer em termos de evolução do próprio aluno, quer no suporte aos professores e aos seus processos de ensino, contribuindo para a inovação de novos modelos e processos. A tecnologia não só torna o ensino mais atrativo, permitindo o acesso a inúmeros recursos, como também promove a flexibilização e a autonomia no estudo do aluno. Tendo em conta o seu potencial, esta possibilita a transição do modelo tradicional de ensino, baseado na transmissão de informação entre professor e o aluno, para um modelo no qual o estudante despende mais tempo em momentos de autoaprendizagem e o docente na inovação da componente pedagógica.

Porém, ao avançar do pensamento das novas tecnologias na educação, enquanto ferramenta para a sua associação a um tutor, constatou-se que os sistemas atuais se têm mostrado pouco eficazes a operar num esquema de abordagem individual, comprometendo, não só, na parte de adaptar os seus métodos e conteúdos ao aluno e ao seu conhecimento, mas também na adaptação a múltiplos domínios. Foi neste contexto que se deu início ao desenvolvimento do sistema *Leonardo*[1] (Belo, Coelho and Fernandes, 2019). Os ideais por detrás deste projeto, que serviu como caso de estudo

[1] O 'Leonardo' é uma plataforma de ensino *online*, especialmente orientada para suportar processos de aferição de conhecimento em vários domínios de estudo. O sistema foi idealizado pelo Professor Orlando Belo, do Departamento de Informática da Universidade do Minho, que desde 2017 tem promovido e suportado diversas ações de investigação e desenvolvimento relacionadas com os vários componentes do sistema que tem vindo a arquitetar.

para esta dissertação, surgiram com a necessidade de colmatar a ausência de um sistema capaz de complementar a formação e avaliação no contexto do ensino académico, procurando a melhoria dos serviços de atendimento e esclarecimento de dúvidas dos alunos de uma determinada unidade curricular.

Em particular, o capítulo da avaliação, num ITS, foi o principal motivador do estudo realizado. Recorde-se que, a generalidade dos alunos aprende ao seu próprio ritmo, não àquele que é imposto em sala de aula. Neste sentido, sistemas que permitam ao estudante avaliar o seu conhecimento, de forma autónoma, as vezes que este necessitar, até que se sinta seguro numa dada temática, são uma contribuição importante para o processo de aprendizagem de um estudante e para a própria educação. Ademais, os tempos modernos requerem plataformas flexíveis, às quais o estudante possa aceder a qualquer momento e em qualquer lugar, aproveitando assim qualquer ímpeto para estudar, algo que não é conseguido em muitos dos ITS existentes.

1.4 Objetivos

Dado o papel do mecanismo de raciocínio na componente de avaliação de um sistema de tutoria inteligente, o principal objetivo do trabalho desta dissertação passou por tornar o *software* capaz de suportar a disponibilização de questões, criteriosamente selecionadas para o aluno, em cada momento de formação ou avaliação, de acordo com as fragilidades que revelou ao longo do processo de interação homem-máquina. Para tal, foi necessário construir um conjunto de estratégias e capacidades de inferência, indispensáveis para tornar o sistema um ITS, com sofisticação suficiente para permitir ao tutor artificial "ludibriar" o estudante, a fim de o testar e avaliar de forma exigente.

Por outro lado, a implementação deste sistema de avaliação seguiu uma abordagem modular, ou seja, a funcionalidade de avaliação está separada das restantes aptidões do tutor, o que vai de

encontro ao objetivo de facilmente se poderem associar ao mesmo outras funcionalidades, resultantes da sua integração com outros módulos, abrindo ainda espaço a futuros acoplamentos. Outro ponto importante prendeu-se com a disponibilização 24h do sistema, através da Web, para que o período de aprendizagem do aluno não tenha de estar remetido ao tempo letivo. A isto acrescentou-se o objetivo de tornar os instrumentos de autoavaliação acessíveis a partir de qualquer dispositivo computacional (*cross-platform*). Esta propriedade contribui para a flexibilidade e adaptação do tutor aos comportamentos dos alunos, neste caso no que diz respeito ao aparelho que estes utilizam na interação.

1.5 Contribuições

Como já referido, o caso de estudo escolhido para esta dissertação foi o sistema *Leonardo*, projeto orientado para o desenvolvimento de um sistema de tutoria inteligente, sob a forma de uma plataforma *online*, para suporte a processos de avaliação de alunos, na Universidade do Minho. À data da elaboração deste documento, os dados relativos à estruturação do conhecimento no sistema mostravam-se sólidos, mas abrindo espaço à exploração de componentes do sistema, como era o caso do mecanismo de raciocínio para o sistema de avaliação. A investigação em torno deste trabalho contribuiu, em termos concetuais e práticos, para o progresso do sistema *Leonardo* e do estudo em torno dos seus componentes, nomeadamente naquilo que é:

- a **representação da informação** no sistema, tendo em conta as várias funcionalidades que se pretendem para um tutor inteligente – suporte à educação em sala de aula, construção do perfil do aluno, independência do domínio, estudo autónomo e avaliação do aluno – e a versatilidade que isto implica. O foco centrou-se nos atributos mais relevantes para o sistema quando este desempenha o papel de avaliador;
- a **elaboração de estratégias para avaliação de conhecimento**, que guiam o processo de avaliação implementado e espelham o carácter adaptativo de um tutor

inteligente. Estas são partes intrinsecamente dependentes de uma base de conhecimento com dados suficientemente diversificados, pelo que o povoamento da base de dados foi necessário para testar as metodologias criadas;

- a **integração do mecanismo de raciocínio**, que coordena a componente cognitiva do ITS, com o restante sistema, com destaque para a sua associação com outros módulos, particularmente o responsável pelo *profiling* do aluno;
- a **interface de utilizador** construída, com vista a pôr em prática o sistema de avaliação, sem descurar a usabilidade e a experiência do estudante na interação com o *software*.

1.6 Estrutura da Dissertação

Para além do presente capítulo, este documento estende-se por mais quatro capítulos, nos quais se apresenta o trabalho desenvolvido nesta dissertação. Esta organização permite que, em primeiro lugar, sejam expostos os conceitos teóricos e cientificamente estudados por terceiros, surgindo depois os esclarecimentos quanto à abordagem pessoal, teórico-prática, sobre o tema. De forma concisa, os restantes capítulos resumem-se em:

- **Capítulo 2 – Trabalho Relacionado –**, que reúne um leque muito diversificado de informação acerca de trabalhos de investigação e casos de aplicação já existentes no domínio dos ITS, dando maior destaque, naturalmente, aos seus componentes de avaliação e à sua orgânica de funcionamento. São abordados conceitos em torno dos mecanismos de raciocínio - regras de produção, resolução de conflitos, algoritmo de inferência, entre outros.
- **Capítulo 3 – Um Modelo de Avaliação –**, em que se faz a apresentação genérica do modelo de avaliação e se define a sua área de aplicação, a sua motivação e os seus objetivos. Aqui, são caracterizados os elementos base do modelo e é descrita a sua influência num processo de avaliação de conhecimento. Além disso exemplifica-se,

também, o funcionamento de algumas das regras definidas para o funcionamento do processo e, graficamente, se esquematizar as diversas etapas da avaliação.

- **Capítulo 4 – Um Caso de Aplicação –**, neste capítulo aborda-se um pouco mais o caso de aplicação do tema desta dissertação – o sistema *Leonardo.* Nesta parte, surge a explicação de como este módulo de avaliação é integrado neste ITS em desenvolvimento, abordando-se também a sua integração com os restantes módulos do sistema, nomeadamente o módulo de *profiling*. Termina com a validação prática do mecanismo de raciocínio abordado no capítulo anterior, com recurso a uma demonstração da sua aplicação num processo de avaliação específico.
- **Capítulo 5 – Conclusões e Trabalho Futuro –**, no qual se apresentam as conclusões deste trabalho. Ao resumo dos temas chave do documento, segue-se uma avaliação crítica do mesmo, realçando as contribuições acarretadas com este trabalho. Numa segunda fase, salienta-se o trabalho futuro a realizar nesta dissertação, com o intuito de melhorar e refinar o módulo de avaliação desenvolvido até ao momento.

Capítulo 2

Trabalho Relacionado

2.1 O Passado e Presente dos ITS

Um ITS, pela sua vertente de procurar imitar comportamentos de um tutor humano, adaptando as suas ações ao aluno, necessita de uma série de componentes de *software* que, desde os primórdios destes sistemas, vem sendo recorrentemente mencionada nas arquiteturas base. A comunidade de investigadores reconhece quatro modelos fundamentais: o modelo do domínio, o modelo do estudante, o modelo do tutor e o modelo da interface de utilizador. Neste capítulo, explora-se a evolução destes elementos, com especial foco nas partes constituintes do sistema de avaliação. A forma de representação do conhecimento, a gestão do processo de avaliação e a interação entre os modelos são tópicos que se procurou investigar, não só em sistemas que surgiram numa fase embrionária, como o SCHOLAR, mas também em sistemas inteligentes mais atuais, casos do TEx-Sys e do ASPIRE, a propósito dos quais se faz uma breve passagem pela noção de sistema *Shell-based*. A relação dos ITS com a Web está, também, presente no seguimento deste capítulo de trabalho relacionado, pelo que o ELM-ART, enquanto um dos primeiros sistemas destas características a ser desenvolvido para a Web, marca presença nesta análise comparativa.

2.2 O Sistema SCHOLAR

Um mecanismo de inferência pode ser definido como o componente lógico que confere a um sistema informático a capacidade de deduzir, através da informação que possui no momento, novos factos e comportamentos, não especificados na sua ontologia inicial. Para se perceber o contexto em que este tipo de mecanismos surgiu, recupera-se neste documento o nome de Carbonell para discutir o artigo em que assinala a inovação do primeiro tutor artificial - o sistema SCHOLAR (Carbonell, 1970b), tema da sua dissertação de Ph.D.

O propósito da publicação é auto-descrito como sendo o de "introduzir um novo tipo de Instrução Assistida por Computador (CAI - *Computer Assisted Instruction*), em muitos aspetos mais poderoso do que os existentes, para provar que é viável, e demonstrar e exemplificar as suas principais capacidades.". Aplicado à temática da geografia da América do Sul, o SCHOLAR demonstrou que os computadores poderiam ter um papel mais influente na educação do que realmente tinham na época. Em termos de interface, o sistema seguia uma abordagem simples, interagindo, em inglês, de forma alternada com o utilizador, consoante o *input/output* de um asterisco, mas sofisticada, explorando os avanços e estudos que iam ocorrendo na área do processamento de linguagem natural – (Simmons, 1970); (Kellogg, 1968); (Bobrow, 1964). Por outro lado, o *software* marcou a diferença pela forma inteligente como, mediante a maior ou menor correção das respostas do aluno às perguntas lançadas, executava ações condicionais, que tomavam em consideração uma série de critérios e procedimentos associados ao domínio de conhecimento, no sentido de adaptar o seu conteúdo ao momento.

Como Carbonell refere, os sistemas CAI até então estavam dependentes de "blocos de material *ad hoc*, habitualmente chamados *frames*, inseridos anteriormente pelo professor", o que constituía uma grande limitação tanto para os alunos, que não podiam recorrer a linguagem natural nas suas

respostas, como para os professores, confrontados com o "fardo considerável de preparação de questões, respostas, palavras-chave, e derivados". A estes sistemas, denominados de *ad hoc-frame-oriented* (AFO) no artigo em causa, apontava-se a falta de poder de decisão, iniciativa e conhecimento no real sentido da palavra. Neste documento, a vertente da linguagem natural não será alvo de grande debate, pois as atenções estarão concentradas na transição dos sistemas AFO CAI, mencionados no parágrafo anterior, para um tipo de abordagem apelidado de *information-structure-oriented* (ISO), que visa a abstração entre o *software* e o domínio de conhecimento em que se aplica. No caso do SCHOLAR, o que Carbonell desenvolveu foi uma estrutura de informação bem definida sob a forma de rede "semântica", onde se reúne a informação alusiva ao domínio - factos, regras, conceitos - sem restringir blocos de texto, questões e outras informações a este. Ao invés, o programa mostra-se capaz de utilizar os dados e respetivas articulações, presentes na rede, para gerar, mostrar e até adaptar as suas questões ao aluno e às suas respostas. O sistema SCHOLAR até vai mais longe, sendo capaz de responder a questões colocadas pelo aluno, se a semântica da sua rede de conhecimento lhe permitir.

Atingiu-se, assim, a primeira implementação de um tutor inteligente. A convenção ISO foi, claramente, um passo vanguardista na implementação deste tipo de sistemas. “Ainda que pareça pouco provável que o SCHOLAR enganasse alguém, o programa estava indubitavelmente à frente do seu tempo num importante aspeto” (Morrison and Rus, 2013). Já era possível identificar-lhe uma base de conhecimento, onde estavam concentradas as informações do domínio numa rede semântica, e um mecanismo de raciocínio, que manipulava esta informação, inferindo e selecionando o conhecimento necessário em cada momento, combinando-o com conhecimento independente de contexto – linguagem natural – e produzindo, assim, conteúdo lúdico. Assim se processava internamente o fluxo de interação homem-máquina - aluno-tutor, nesta situação. Ora, este era o aspeto que o distinguia: o armazenamento independente do seu conhecimento do domínio e do alusivo à sua missão de tutor.

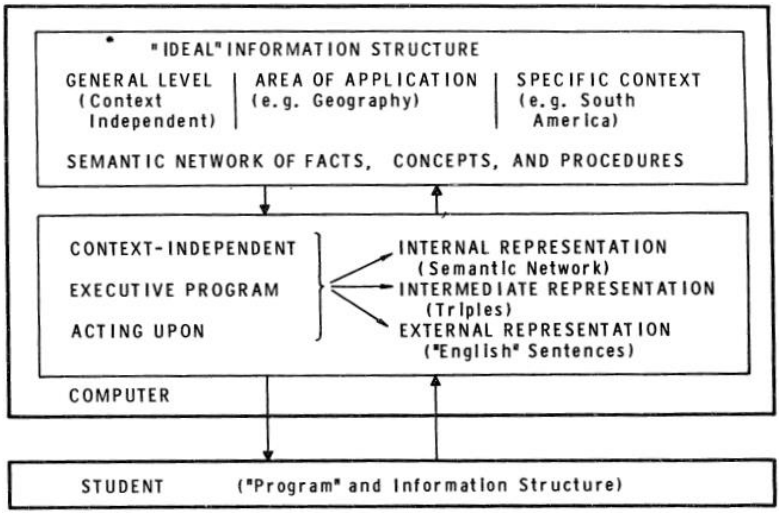


Figura 2: Estrutura geral do sistema SCHOLAR - imagem extraída de (Carbonell, 1970b).

A representação de conhecimento sob a forma de rede semântica favoreceu claramente o desenvolvimento de sistemas de tutoria inteligentes, dado o modo eficiente como estas permitem armazenar e obter informação. Sendo uma rede semântica um conjunto de nodos, que representam os conceitos do domínio, e arestas, representativas das relações entre os nodos, importa destacar a vertente hereditária destas redes. Cada nodo herda as propriedades dos nodos pai, aqueles com quem têm uma relação "é-um", e isto abriu caminho para que os sistemas inteligentes pudessem inferir sobre esta rede de conhecimento. Na Figura 3 podemos ver a representação simplificada de uma rede semântica à imagem da rede utilizada pelo SCHOLAR.

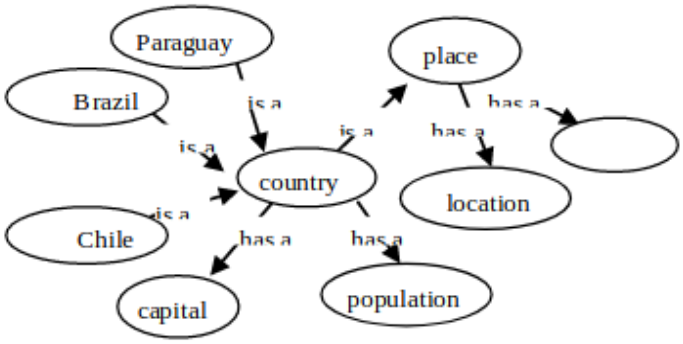


Figura 3: Rede semântica do SCHOLAR simplificada (imagem extraída de Morrison and Rus, 2013).

Utilizando a rede da Figura 3 como exemplo, como o motor de inferência conhece que Brasil é um país - *Brazil is-a country* - então sabe que o Brasil tem uma população, porque também sabe que um país tem uma população – *country has-a population*. Assim, através da herança de propriedades na rede, se o conhecimento estiver organizado em sequências profundas de nodos, é possível a um sistema inteligente raciocinar sobre a estrutura de uma forma flexível, enquanto simultaneamente evita armazenar informação redundante na base de dados.

Em termos da avaliação do conhecimento do estudante, este relacionamento entre os conceitos espelhados nos nodos da rede permite inferir questões como "Qual é a capital do Brasil?". Isto é possível porque o SCHOLAR consegue, a partir de um antecedente comum, formar questões que envolvam múltiplos conceitos, o que lhe permite, também, medir a correção da resposta do estudante, em função do quão próxima estiver do esperado. Por exemplo, para a questão anterior, se a resposta fosse uma cidade, mesmo que incorreta, estaria mais correta do que se fosse um país.

A sua capacidade de processamento de linguagem natural torna possível colocar questões, como a anterior, de várias formas: resposta aberta, escolha múltipla, preenchimento de espaços vazios ou verdadeiro-falso (Rickel, 1989). Focando atenções no modo de escolha múltipla, o mais importante no contexto desta dissertação, constatou-se que o SCHOLAR apresenta sempre quatro alternativas, sendo uma delas a correta. Uma estratégia que este tutor artificial implementa, numa tentativa de se aproximar do humano, é a geração de opções de resposta desajustadas com a questão, ainda que com espaço, segundo Carbonell, para reduzir a percentagem de aparecimento destas opções. A seleção do modo da questão dá-se com base em probabilidades, assim como a dos conceitos base da pergunta, quando não são fornecidos. Para manter controlo sobre o que já foi perguntado ao aluno, são usadas etiquetas temporárias, que influenciam inversamente a probabilidade de ser lançada uma questão sobre o tema.

Naturalmente, o mecanismo de inferência tem mais facilidade em operar sobre um conjunto fechado de valores, como acontece nas questões de escolha múltipla e verdadeiro-falso, uma vez que as respostas abertas implicam que o sistema consiga raciocinar sobre a própria extensão do seu conhecimento, para que possa avaliar quer a correção da resposta, quer o grau de certeza que possui sobre a mesma. Para isto são utilizadas meta-regras previamente definidas para o sistema, que conferem um carácter natural à inferência do SCHOLAR, isto é, que reflete o comportamento humano perante situações de conhecimento incompleto (Barr and Feigenbaum, 2014).

2.3 *Shell-based*

O aparecimento e crescimento da Web direcionaram para este ambiente uma grande parte da investigação que se realizava no campo dos sistemas de tutores inteligentes. A sua capacidade de reunir, à distância, mais pessoas, quer para ensinar, quer para aprender, potencia o desenvolvimento deste tipo de sistemas para uso *online*, porém, a maioria revela défice de flexibilidade, por serem baseados na representação estática do conteúdo lúdico, e elevados custos de implementação. As premissas anteriores estão na base do aparecimento de *authoring shells - frameworks* que promovem a interoperabilidade e reutilização entre sistemas de tutores inteligentes e a redução dos custos de desenvolvimento, através da abstração do domínio de conhecimento. Um dos objetivos é aumentar a influência dos professores na criação de tutores artificiais, fornecendo *know-how* na definição da base de conhecimento e da vertente pedagógica do *software* (Murray, Blessing and Ainsworth, 2003). Como exemplos de *authoring shells*, tome-se em consideração os sistemas TEx-Sys e ASPIRE. Veja-se um pouco em que consiste cada um deles.

2.3.1 TEx-Sys

À semelhança do sistema SCHOLAR, este sistema seguiu a abordagem das redes semânticas para representação do conhecimento. No entanto, difere desse sistema no sentido em que a respetiva rede não se encontra previamente definida e enquadrada num certo domínio. Ao invés, é

configurável, permitindo que diferentes ITS possam ser criados a partir da extensão deste *framework*, por indivíduos que não necessitam de conhecimentos de desenvolvimento de *software*. Os *domain knowledge experts* ficam, assim, habilitados a modelar o conhecimento, estratégias de resolução de problemas e técnicas associados ao domínio.

No TEx-Sys, designação que deriva de *Tutor-Expert System*, o modelo da rede semântica é baseado na memória associativa humana, onde os nodos representam os objetos do domínio de conhecimento, unidos por diferentes relacionamentos. Além de um nome e ligações a outros objetos, um nodo pode incluir também uma descrição textual, conteúdo multimédia (imagem, vídeo, aúdio, etc) ou endereços URL, material atualmente essencial para cativar os estudantes. A Figura 4 mostra os tipos distintos de associações que se estabelecem entre nodos. Desta especificação ontológica destaca-se a formação de conjuntos atributo-valor (*Slot-Filler*) relacionados com um dado conceito – *frames* – que melhoram a organização do conhecimento, sobretudo graças à sua capacidade de pesquisa (Stankov *et al.*, 2008).

No que diz respeito a testar o conhecimento do estudante, o TEx-Sys recorre a dois métodos: 1) o método da sobreposição e 2) testes no formato *quiz.* O processo começa com o professor a definir os conteúdos educacionais que devem ser avaliados no teste. Esta seleção reflete-se na base de conhecimento, pois só os nodos relacionados com as temáticas em causa serão envolvidos, e em ambos os métodos.

No método 1), é apresentada ao aluno uma rede semântica com conceitos errados ou em falta ou com ligações removidas ou incorretas, para testar a sua capacidade de representação do conhecimento. Ao avaliado compete, então, adicionar, remover ou corrigir nodos e respetivos relacionamentos, de modo a corrigir a semântica da rede na representação do domínio em avaliação. O diagnóstico da solução do aluno e respetiva classificação são, posteriormente, obtidos através do sistema próprio MARK, baseado em regras de produção, que oferece uma avaliação quantitativa e qualitativa e recomendações de estudo para o aluno.

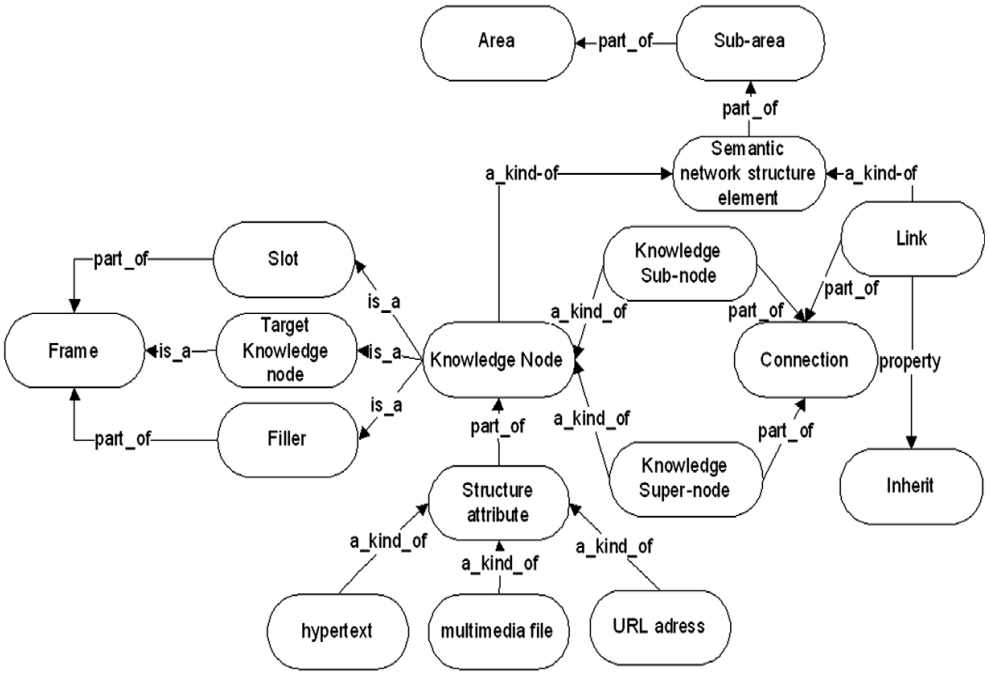


Figura 4: Ontologia da representação de conhecimento no Tex-Sys (imagem extraída de Stankov et al, 2008).

Quanto ao método 2), o *quiz* pode ser estático, ou seja, composto por questões previamente definidas, ou dinâmico, onde as questões vão sendo geradas, para que haja adaptação ao indivíduo em avaliação à medida que se vão obtendo resultados parciais. A cada questão é associado um conjunto de respostas, que podem estar corretas ou incorretas. Centrando atenções no *quiz* dinâmico, importa abordar a evolução do grau de complexidade das questões à medida que o aluno vai percorrendo as etapas do teste. No TEx-Sys, as questões são agrupadas em três categorias de peso, cada uma com quatro formatos de questões predefinidos, relacionados com a rede semântica. Exemplificando, para a primeira categoria existem os seguintes formatos (e respetiva descrição):

1. Selecionar o tipo de conexão (direta ou indireta) entre dois nodos:

   Questão: "Os nodos X e Y estão conectados?"

   Resposta: "Sim, diretamente/Sim, indiretamente/Não"

2. Confirmar se dois nodos estão conectados

   Questão: "O <Nodo X> e o <Nodo Y> estão conectados através da <Ligação Z>?"

   Resposta: "Sim/Não"

3. Selecionar o nodo que tem um determinado atributo estrutural (descrição textual, imagem, vídeo, etc):

   Questão: "O que está no <Atributo Estrutural X>?"

   Resposta: O aluno escolhe o(s) nodo(s), ou seja, os conceitos relacionados, de uma lista de respostas possíveis (uma correta e três incorretas)

4. Avaliar se um nodo possui um certo *frame:*

   Questão: "O <Atributo X> possui <Valor Y> para o <Nodo Z>?"

   Resposta: "Sim/Não"

Naturalmente, o grau de complexidade dos quatro tipos de questões cresce com a progressão no peso da categoria. A lógica por detrás da colocação do conhecimento do aluno numa dada categoria começa com as duas primeiras questões a serem arbitrariamente selecionadas da segunda categoria de peso. A cada duas respostas, mediante o desempenho do aluno, a sua categoria pode subir, manter-se ou descer, consoante acerte duas, uma ou nenhuma das questões, com exceções nas categorias das extremidades: o teste termina após duas respostas erradas consecutivas a perguntas da categoria mais fácil (primeira), prejudicando a nota final; o aluno não pode subir de categoria quando atingiu já o nível de exigência mais elevado (terceira). A nota final é calculada em função do rácio entre pontos obtidos e pontos possíveis, culminando o processo de avaliação, em conjunto com recomendações de material de estudo.

2.3.2 ASPIRE

O segundo sistema *shell-based* que se aborda neste trabalho define como objetivo a simplificação do processo de desenvolvimento e instalação de tutores artificiais baseados em restrições (*constraint-based*), partilhando com outros sistemas deste tipo a ambição de tornar possível aos professores – entenda-se, aos possuídores do conhecimento e capacidade de instrução – criarem um ITS, apenas definindo o conteúdo teórico e lúdico do domínio de ensino. O processo de criação, ou *authoring*, começa pela especificação das características do domínio, tais como os sub-domínios que representam áreas específicas, complementada pelo desenvolvimento da ontologia do domínio. Esta ontologia é modelada de forma semelhante a uma rede semântica, com conceitos representados em nodos, com as suas respetivas propriedades, e relacionamentos *é-um(a)* entre eles, numa perspetiva de incentivar o *expert* a refletir sobre o domínio e, consequentemente, ajudá-lo na posterior composição da base de restrições (Mitrovic *et al.*, 2009). A esta fase acrescenta-se, ainda, a criação de conteúdo pedagógico, como, por exemplo, a identificação dos procedimentos necessários para resolver um dado problema, na perspetiva do autor.

Após esta configuração inicial, espera-se que sejam adicionados ao *software* formatos de problemas, com o respetivo cabeçalho do problema, os requisitos da tarefa, eventuais sub-componentes - como gráficos ou descrições adicionais - e a estrutura da solução. Para a solução, o autor deve recorrer aos conceitos especificados na ontologia para listar o que deve ser abordado nas respostas à questão. Para tarefas procedimentais, cada passo deverá ter a sua lista de conceitos-chave, resultando numa solução final composta por uma lista de listas de conceitos provenientes da ontologia. Neste momento, o sistema fica em condições de povoar a base de restrições, traduzindo a informação transposta na ontologia e na estrutura dos problemas. Analisando os relacionamentos entre os conceitos e as propriedades que podem ter sido atribuídas aos mesmos, condições, como tipos ou intervalos de valor, aplicadas às propriedades dos conceitos são extrapoladas para restrições sintáticas. Os problemas procedimentais originam também este tipo de restrições, neste caso para garantir que as várias etapas do processo são cumpridas na ordem correta, o que na

prática se reflete no bloqueio da transição para o passo seguinte sem o cumprimento do atual. Em seguida, são originadas restrições semânticas, responsáveis por verificar se o aluno responde ao que a questão pretende. Estas restrições podem ser mais ou menos genéricas, consoante haja soluções alternativas ou não, já que, no primeiro caso, teriam de verificar formas equivalentes de resolver o problema, enquanto, no segundo, bastaria a comparação entre a solução apresentada e a definida para o problema.

Estando, previamente, definidos os formatos, segue-se a adição de problemas e das suas soluções. Para tal, deve ser selecionado o formato de questão, preenchida a informação que este necessita de receber, adicionado um nível de dificuldade (entre 1 e 9, com complexidade crescente) e, se necessário, definido um nome para o problema, que acompanhará o número identificador gerado automaticamente pelo sistema. Ao problema adicionado, devem ser acrescentadas uma ou mais soluções, também respeitando a estrutura previamente explicitada, ou seja, para uma tarefa procedimental são necessárias soluções para cada passo, havendo ainda espaço para notas adicionais ou simplesmente um nome para cada solução. Nota para o facto de que todas as soluções alternativas devem ser registadas, pelo que o *software*, no momento de adicionar a alternativa, automaticamente copia a solução anterior, já que o grau de similaridade é normalmente elevado. A Figura 5 mostra a interface de edição de problemas do ASPIRE.

Num ITS desenvolvido a partir do ASPIRE, o processo de avaliação é conduzido pelo módulo pedagógico, isto é, é o componente que interage com outros módulos, com uma dada funcionalidade, e, no fim de obter as múltiplas respostas, envia o resultado a ser apresentado. A seleção de questões, por exemplo, é resultante da interação com o módulo que constrói o perfil de conhecimento do estudante, que fornece *feedback* sobre o nível apropriado de complexidade das questões, e com o módulo que gere os domínios do sistema, para restringir a seleção ao domínio de conhecimento do ITS em causa. Já a avaliação da resposta do aluno é levada a cabo pela interação com o módulo de diagnóstico, que necessita da informação do gestor de domínios para identificar os erros que o aluno pode ter cometido. Esta análise do desempenho na resposta a um problema é

utilizada para atualizar o perfil do estudante, no módulo responsável, que por sua vez auxilia o módulo pedagógico no fornecimento de *feedback* ao aluno.

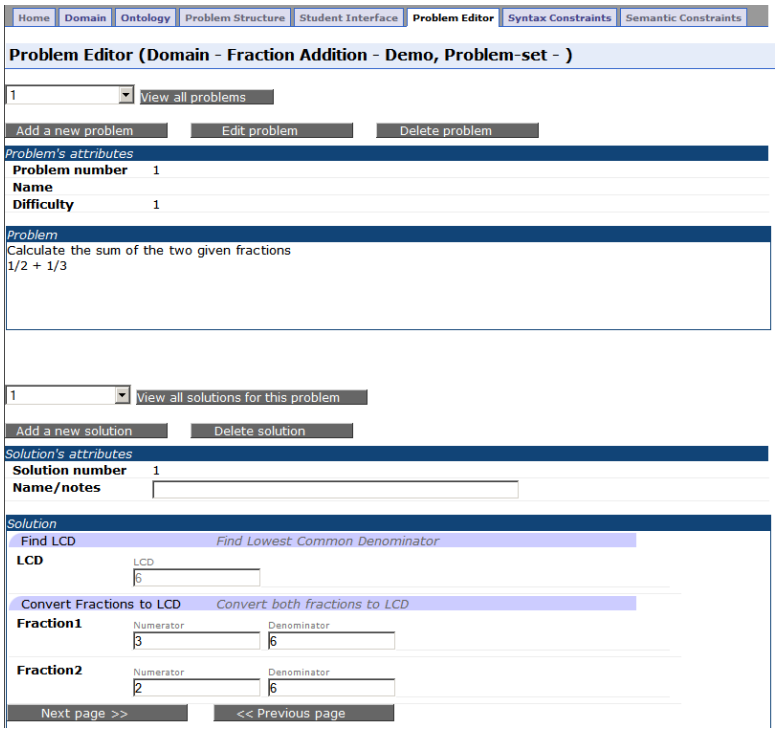


Figura 5: Edição de um problema no ASPIRE (imagem extraída de Mitrovic et al, 2009).

2.4 ELM-ART

O próximo sistema que abordaremos é o ELM-ART (*ELM Adaptive Remote Tutor*). Este é um tutor artificial para suporte à aprendizagem da linguagem de programação LISP. Este foi um dos primeiros sistemas destas características a ser desenvolvido especialmente para a Web, fornecendo material pedagógico em formato hipermédia. Trata-se de um manual de programação inteligente e interativo, organizado por capítulos, que se vão subdividindo em unidades pedagógicas mais pequenas. Tem a capacidade de conhecer, de antemão, o que é apresentado em cada página, isto é, que conceitos são introduzidos, o que demonstram os exemplos e que conhecimentos são

necessários para resolver um dado problema, pelo que, por um lado, consegue providenciar referências para o material de estudo mais adequado a certo aluno e, por outro lado, providenciar uma experiência interativa aos estudantes, permitindo-lhes navegar através dos exemplos e resolver os problemas.

Citando Brusilovsky, Schwarz e Weber (1996), "a chave para o comportamento inteligente do ELM-ART é o conhecimento sobre o domínio do tema e sobre o aluno, que é representado no sistema de várias formas. A maior parte da base de conhecimento do ELM-ART consiste no conhecimento acerca de resolução de problemas em LISP, que é representado como uma rede de conceitos, planos e regras". Ora, estes conceitos são elementos da linguagem de programação (funções, objetos de dados, tipos), estando cada um representado numa página do livro digital, e estão relacionados entre si maioritariamente através de ligações *é-um(a)* e *parte-de*, um pouco à imagem do que acontece noutros sistemas abordados nesta secção. Também as páginas de conhecimento pedagógico, isto é, o material de ensino, como explicações, exemplos e problemas, surgem indexadas nesta rede de conceitos, para que se associem ao material de estudo os vários assuntos relacionados com uma unidade pedagógica.

Outro componente importante deste ITS é o responsável pela modelação do estudante, que surge da combinação de abordagens *case-based* e *rule-based* como estratégia de inferência. Esta funcionalidade assenta num modelo de sobreposição multicamada, que suporta a adaptabilidade do tutor ao aluno, refletida nas anotações que os conteúdos do manual recebem: "já aprendido", "inferido", "declarado como sabido pelo utilizador", "preparado e sugerido para ser visitado" e "Não preparado para ser visitado". Ao leitor interessado em aprofundar este assunto recomenda-se (Weber, 1999). A modelação automática, por parte do sistema, foi complementada por uma interface para o aluno inspecionar o seu próprio modelo de conhecimento, podendo modificá-lo se assim o entender.

Pensando na avaliação das competências adquiridas por um aluno no ELM-ART, (Weber and Specht, 1997) introduziu os testes na segunda versão do sistema, o ELM-ART II, com o objetivo de avaliar o estado de conhecimento do aluno de forma mais competente. Um teste pode assumir o papel de exercício de validação da aprendizagem de um novo conceito abordado. Isto foi uma clara melhoria relativamente à versão original do ELM-ART, na qual o conhecimento do estudante num tema era inferido a partir das visitas às páginas correspondentes. Outro papel no sistema é o de teste introdutório, que testa conhecimentos nos conteúdos a serem lecionados de seguida. Por último, existem os testes finais, que põem à prova o aluno em situações de fim de uma unidade pedagógica.

Segundo (Weber and Brusilovsky, 2001), a estrutura criada para os elementos de avaliação tem como chave o designado *test item*. Um teste é um aglomerado destas questões, sendo cada uma constituída por:

- **Cabeçalho**, isto é, o texto da questão;
- **Respostas corretas**, ou seja, informação acerca das respostas à questão.
- **Nome da função de correção**, que verifica se o aluno respondeu corretamente, comparando a sua resposta com as corretas.
- **Dificuldade da questão**, parâmetro que determina em quanto deve ser incrementado o valor de confiança de que o aluno aprendeu o conteúdo avaliado na questão, quando este responde corretamente à questão;
- ***Feedback*** para o aluno com a explicação da resposta correta;
- **Lista de conceitos teóricos** relacionados com a questão.

Em termos de formato, estas podem assumir uma das seguintes vertentes:

- **Sim-não**, comparáveis às questões habitualmente designadas de “verdadeiro-falso”.

- **Escolha múltipla**, com distinção entre questões em que o aluno escolhe apenas uma resposta correta (*forced-choice*) e questões em que tem de selecionar todas as opções corretas (*multiple-choice*).
- **Resposta livre**, em que o estudante responde textual e livremente à questão.
- **Espaços em branco**, formato em que é requerido o preenchimento de espaços vazios com números ou caracteres, completando uma frase.

Consoante os conceitos que abordam, as questões são agrupadas em coleções, tratadas como *test groups*. Desta forma, se mais do que um conceito for testado numa questão, esta pode estar presente em várias coleções. A um *test group* associam-se vários parâmetros configuráveis, casos do *group-length*, que define quantas questões são exibidas em simultâneo, do *min-problems-solved*, que define o número de mínimo de questões do grupo que devem ser resolvidas corretamente, e do *max-errors*, isto é, o número máximo de erros permitido nas questões apresentadas na página (Weber and Specht, 1997).

O processo de avaliação no ELM-ART consiste, então, no lançamento de um enunciado digital, com tantas questões quantas o *test-group d*o conteúdo pedagógico em avaliação exige. Após o aluno responder a todas elas, o sistema verifica se o número de erros não excede o máximo permitido e se o número mínimo de questões acertadas foi atingido. Quando este objetivo é atingido, os conceitos são dados como aprendidos. Até lá, novas páginas com problemas diferentes são exibidas.

Por fim, o conhecimento do estudante numa unidade pedagógica é espelhado num "valor de confiança". O algoritmo de cálculo deste valor tem em conta a dificuldade de cada questão do conjunto, pela qual se multiplica um peso, que simboliza a importância do *test item* no determinado *test group*. Este produtório é somado ao valor de confiança nas situações em que o aluno acerta a questão. Já quando a resposta é errada, é multiplicado por um fator de erro e posteriormente subtraído ao valor de confiança. Nas situações em que um problema avalia conceitos que

pertencem a diferentes unidades de ensino, uma resposta certa ou errada por parte do estudante influencia os valores de confiança dos vários conceitos (Weber and Brusilovsky, 2001).

2.5 Análise Comparativa

Para concluir a apresentação de trabalho relacionado, na Tabela 1 apresenta-se um resumo das propriedades mais relevantes, no seio desta dissertação, dos sistemas de tutoria inteligente enumerados acima.

	SCHOLAR	TEx-Sys	ASPIRE	ELM-ART
Web-based	Não	Sim (2ª versão)	Sim	Sim
Geração dinâmica de questões	Sim	Sim	Não	Não
Seleção adaptativa	Conceitos e formato da questão	Conceitos e nível de dificuldade	Conceitos e nível de dificuldade	Conceitos
Base de conhecimento	Rede semântica	Rede semântica	Rede concetual	Rede concetual
Domínio	Geografia	Multi-domínio	Multi-domínio	LISP
Motor de Inferência	Baseado em regras	Baseado em regras	Baseado em restrições	Baseado em casos e regras

Tabela 1: Análise comparativa dos sistemas inteligentes de tutoria estudados.

Capítulo 3

Um Modelo de Avaliação

3.1 Caracterização base

Atualmente, a realização de testes continua a ser o método que oferece maior segurança a um professor quando chega o momento de concluir acerca da aprendizagem de um aluno. Os motivos por detrás da classificação atribuída por um professor devem ser objetivos, para que não restem dúvidas acerca do mérito do estudante. É esta segurança que os sistemas de avaliação automáticos oferecem e que leva a acreditar que os testes, e meios semelhantes, continuarão a liderar a avaliação no ramo da Educação.

Usualmente, um processo de avaliação resume-se à realização de um ou mais testes durante o período de avaliação, desenvolvidos por um ou mais professores, sobre os quais possivelmente recai a responsabilidade de corrigir e classificar as provas dos alunos. Esta carga administrativa dos professores, associada à classificação dos testes, fica reduzida com auxílio dos tutores artificiais, que automaticamente conseguem avaliar o aluno, contribuindo também para a aceleração do processo avaliativo. Os alunos recebem, no imediato, *feedback* dos seus resultados e aos

professores resta, então, mais tempo para se focarem na componente lúdica e no apoio aos estudantes com dificuldades.

O modelo de avaliação que aqui se apresenta visa, precisamente, suportar a componente avaliativa na área da Educação. O objetivo do modelo não passa por remover o professor do processo de análise do conhecimento adquirido por um aluno, mas sim por atuar como uma ferramenta complementar e de ajuda ao professor. A sua integração num sistema de tutoria inteligente *online*, acessível a qualquer momento, traz ao aluno a possibilidade de pôr em prática o conhecimento adquirido nas aulas ou em casa, no *timing* que lhe for mais conveniente. Se, por um lado, isto beneficia a aprendizagem autónoma do aluno, por outro, fornece evidências ao professor, o avaliador, sobre o empenho e estado de conhecimento do estudante, através dos dados da classificação automática do sistema.

Quanto à área de aplicação deste componente de *software*, destaca-se ainda a independência do ciclo de estudos em causa. Em diferentes países é comum encontrarem-se tabelas classificativas distintas. Em Portugal, por exemplo, as notas atribuídas aos alunos variam ao longo do trajeto escolar. Se, nos primeiros ciclos de escolaridade, a avaliação se faz a 6 níveis – de 0 a 5 –, a partir do ensino secundário o cenário muda e a classificação de um aluno enquadra-se numa escala de 0-20. Ora, o mecanismo de avaliação proposto abstrai-se e uniformiza as classificações dos diferentes ciclos. Internamente, a única classificação pela qual o modelo se rege é atribuída ao aluno pelo ITS em que se enquadra, nomeadamente pelo seu componente responsável pelo modelo do estudante. Como é expectável, essa classificação está intrinsecamente relacionada com o desempenho do aluno nos testes que realiza. Visto que o módulo do avaliador não prevê guardar histórico, o perfil do estudante evolui nos registos de outro componente, responsável por uma classificação ponderada que considera todos os testes realizados – um módulo de *profiling* específico.

3.1.1 O Formato *Quiz*

A objetividade dos testes em forma de *quiz* torna-os parte importante dos sistemas educacionais *online*, sendo dos formatos mais usados e desenvolvidos. Este carácter simples e objetivo motivou a escolha deste formato para as questões que alimentam o modelo de avaliação, abordado nesta dissertação. Cada questão assume várias opções de resposta, geralmente com texto breve, mas podendo também assumir formato multimédia. Ainda que em certos *quizzes* existam questões com resposta múltipla, o mecanismo desenvolvido, por agora, trabalha com o formato MC/SA – *multiple-choice/single-answer*, logo, apenas uma das opções de resposta pode ser selecionada.

Num ITS, usualmente uma questão passa por várias etapas. Brusilovsky e Miller (2001) definiram que o ciclo de vida de uma questão é composto por três etapas: preparação, lançamento e avaliação (Tabela 2). A primeira etapa consiste na criação, isto é, no momento em que o conteúdo e dados associados à questão são definidos e armazenados no sistema. Por norma, esta tarefa é desempenhada por professores ou peritos, capazes de gerar conteúdo, através de uma interface independente do sistema de avaliação, por vezes a cargo de uma ferramenta alheia ao ITS.

Preparação	Lançamento	Avaliação
Criação do conteúdo por um perito	Seleção da questão num *quiz*	Classificação da resposta
Armazenamento	Interação com o aluno	*Feedback* ao aluno (se aplicável)

Tabela 2: Etapas do ciclo de vida de uma questão num sistema de tutoria inteligente.

Porém, a vida ativa de uma questão tem início quando esta é selecionada para ser apresentada ao aluno num teste. Nesta categorização, a seleção surge como um passo da fase de preparação, uma vez que o professor pode, estaticamente, predefinir que questões compõem um *quiz.* Contudo, aqui a seleção ocorre em execução (*runtime*), dinâmica e especificamente para o aluno, com base em probabilidades ou no perfil do estudante, pelo que se enquadra sobretudo na fase seguinte. A fase do lançamento abrange o período em que o sistema entrega a responsabilidade à interface que

interage com o aluno. Quando apresentada ao aluno, espera-se que uma questão possibilite a interação com o mesmo. É importante que a interface providencie todos os meios para que o sujeito em avaliação consiga responder ao que se pergunta – mostrar todo o conteúdo, adaptando-se ao ecrã do dispositivo, ou garantir que as imagens têm dimensão suficiente, por exemplo –, resposta esta que desencadeia a etapa seguinte: a avaliação. O último passo do ciclo de vida da questão é a avaliação. Aqui, o sistema analisa a resposta dada pelo estudante e decide se está correta (total ou parcialmente) ou incorreta, podendo dar *feedback* ao aluno, se o tipo de *quiz* assim o permitir. Segue-se a classificação da resposta, face ao seu grau de correção, ação que pode desencadear a atualização de modelos do sistema, os quais podem ou não ter influência na seleção da próxima questão.

3.1.2 Preparação de Questões

Ainda que, o momento da preparação de questões não esteja diretamente a cargo do mecanismo de avaliação, importa mencionar a estrutura de informação que o componente desenvolvido em prol desta dissertação espera receber numa questão. Visto que o *software* desenvolvido perspetiva a integração do modelo de avaliação no projeto *Leonardo*, a abordar na próxima secção, as questões manipuladas são documentos JSON, persistidos numa base de dados NoSQL, orientada a documentos, neste caso com recurso à tecnologia MongoDB. Esta escolha oferece flexibilidade à estrutura. No entanto, existe um conjunto de campos que devem, necessariamente, ser especificados numa questão, para que o motor de inferência consiga gerir o processo de avaliação.
Inevitavelmente, o autor de uma questão precisa de fornecer um cabeçalho e opções de resposta. O campo `header´ de uma questão prevê receber a descrição textual da pergunta, ou seja, o seu conteúdo. Sendo necessário suporte visual, pode ser feito o *upload* de imagens para a ferramenta de edição, que contempla o seu armazenamento no campo `images´. Por sua vez, as opções de resposta constituem o corpo da questão – `body´ –, que se trata de uma lista de documentos com a sua própria estrutura.

Numa questão, cada resposta possível requer a definição de três campos: `answer´, `correction´ e `mandatory´. O primeiro está para a resposta como o campo `header´ está para a pergunta, isto é, descreve textualmente o seu conteúdo. O campo seguinte trata-se do fator de decisão acerca da correção da resposta. Opções erradas são assinaladas com valor 0, enquanto que a resposta correta recebe valor 1. Por último, o terceiro campo é booleano e representa a obrigatoriedade da resposta no conjunto final de opções apresentado ao estudante. A relevância deste campo surge da possível adição de mais opções de resposta do que o número máximo exibido na interface. Assim, as opções com valor *verdadeiro* neste campo são necessariamente selecionadas pela interface em todas as formas da questão, devendo ser limitadas ao número máximo de respostas permitido.

Com um papel fundamental no processo de avaliação, o nível de dificuldade - `difficulty_level´ - e o tempo de resposta - `answering time´ - são, tal como os nomes indicam, os campos a cargo de registar o grau de dificuldade e o tempo de resposta máximo da questão. O nível de dificuldade expressa-se na escala de '1' a '5', onde o último reflete a complexidade máxima. Por sua vez, o tempo de resposta contempla quantos segundos, no máximo, o aluno pode demorar a responder.

Outros campos importantes na estrutura das perguntas de um teste são o identificador - `id´ - e a solução - `solution´. O primeiro identifica a questão, papel semelhante ao do identificador único que o motor MongoDB define nos seus documentos, sendo este controlado pelo sistema e não pela base de dados. Já a solução pode assumir vários papéis, como explicitar textualmente o *feedback* a dar ao aluno quando este solicita esclarecimento acerca do caminho para a resposta correta, ou enumerar os apontadores para o material de estudo sobre os conceitos abordados na questão. A sua simbologia é, portanto, comparável a um campo existente no *test item* do ELM-ART, ITS abordado no capítulo anterior, cuja referência se recupera neste documento. No ELM-ART, um *test group* reúne questões associadas à mesma temática. Ora, o sistema de avaliação desenvolvido também considera que cada questão se enquadra num certo domínio. O campo `domain´, de presença obrigatória na estrutura, é particularizado por três valores textuais: `description´ contém a designação do domínio, `study_cycle´ especifica o ciclo de estudos ("Ensino Superior", por

exemplo) e `scholarity´ complementa o valor anterior (com referência ao curso e ano de formação em concreto, seguindo o exemplo anterior). Também obrigatório é o campo `subdomain´, que aprofunda o contexto teórico da questão, ou seja, subdivide o domínio em conceitos mais específicos. Existe ainda o campo opcional `subsubdomain´, que permite ao autor da questão introduzir um nível adicional de particularidade ao domínio da questão. Naturalmente, esta representação está relacionada com a forma como a base de conhecimento do *Leonardo* se organiza, como se verá no próximo capítulo. Estes campos eram, à data de início deste trabalho de dissertação, parte constituinte da estrutura planeada para as questões na iniciativa *Leonardo*. Para melhorar a capacidade e funcionalidade do mecanismo de raciocínio, foram definidos campos adicionais, essenciais na construção de regras e estratégias de avaliação. São eles:

- "repetitions": o número de vezes que a questão pode ser repetida, caso seja lançada num *quiz*, se o valor deste campo for '1', isto significa que, uma vez lançada, a questão pode sair uma segunda vez;
- "precedence": lista de *id*'s das questões que devem ser lançadas antes desta questão;
- "display_mode": a forma como as possíveis respostas aparecerão ao utilizador do sistema: todas de uma vez ou uma de cada vez; o valor do campo é representado por uma letra, "G" ou "I", que simbolizam o aparecimento simultâneo e iterativo das várias opções de resposta, respetivamente; neste segundo caso, todas as opções teriam uma resposta do género *verdadeiro* ou *falso*, o que justifica o aparecimento gradual, pois cada opção teria uma resposta independente.

Com estas adições, o professor fica com um maior leque de soluções ao seu dispor para poder aumentar a sofisticação dos seus métodos para o teste do conhecimento do aluno, sobretudo através dos atributos das repetições e do modo de aparecimento das respostas. Quanto à listagem das precedências, se um dos objetivos deste sistema de avaliação é tornar os testes personalizados, logo, sem fixação prévia de questões, este campo contribuiu para "manter a ordem" nas sequências

de questões. Em suma, desta forma é possível garantir um maior controlo por parte do docente, sem prejudicar a flexibilidade e adaptabilidade.

3.1.3 Seleção Adaptativa

Como tem sido referido, os *quizzes* tidos em conta pelo trabalho desenvolvido nesta dissertação pretendem ser, tanto quanto possível, dinâmicos ou, como são habitualmente apelidados no ramo dos tutores artificiais, adaptativos. Naturalmente, um sistema de avaliação completo inclui também uma vertente que permite ao professor criar, por motivos de equidade, por exemplo, um teste estático, que não varia de aluno para aluno, com um conjunto fechado de questões à partida. Todavia, este não é o ponto fulcral neste trabalho de dissertação, nem é necessário um sistema de tutoria inteligente para realizar este tipo de avaliação. Aqui, o foco centra-se numa análise automática do conhecimento do estudante, na qual é o sistema a selecionar adaptativamente todas as questões, em função do que conhece do aluno com quem interage.

As tarefas de uma equipa docente, auxiliada pelo mecanismo de avaliação descrito, ocorrem, essencialmente, numa fase embrionária do processo avaliativo. Anteriormente, apresentou-se e explicou-se uma das atividades do sistema - a preparação de questões -, que deve ser desempenhada por professores ou por quaisquer outros indivíduos com conhecimento e capacidades pedagógicas no domínio em questão, antes de um aluno poder ser avaliado pelo sistema.

Por si só, a preparação de questões não é suficiente. O motor de inferência deve ser, também ele, preparado para corresponder àquilo que são os critérios de avaliação da disciplina. Deste modo, consideram-se configuráveis ao nível do domínio alguns parâmetros, com aplicação nas regras de produção definidas para a máquina de raciocínio. Para que se entenda o significado de todos os parâmetros que serão apresentados de seguida, importa recordar que o sistema de avaliação se orienta pela classificação atribuída ao conhecimento do aluno pelo modelo do estudante. O cálculo

deste nível tem em conta o desempenho do aluno na resposta às questões dos sucessivos testes e a aptidão que revela a responder, isto é, a rapidez com que o faz. A relação entre respostas corretas e total de respostas reflete o desempenho. Um melhor desempenho reflete-se, claro está, no reconhecimento de um nível de aprendizagem superior. Já a aptidão relaciona-se com o tempo que o estudante consome para dar uma resposta às questões, dentro daquilo que é o máximo definido pelo campo `answering_time´ da questão. Nesta situação, um maior valor da aptidão representa uma menor fração do tempo máximo requerida, em média.

De seguida, enumeram-se, então, os parâmetros do domínio que são configuráveis e relevantes para o motor de inferência:

- Nível de conhecimento do aluno por omissão: a classificação predefinida do nível de conhecimento do aluno; até o tutor recolher informação suficiente acerca do estudante e conseguir estimar o seu nível de conhecimento, as questões lançadas serão de nível de dificuldade correspondente a este valor;
- Fator de elevado desempenho: o valor de desempenho mínimo para a subida do nível do aluno; juntamente com outros fatores, este fator contribui para a decisão de aumentar a dificuldade das questões para o dado estudante;
- Fator de *backlog*: o valor que intervém, conjuntamente com o nível de conhecimento do aluno, no cálculo do número mínimo de respostas certas ou erradas consecutivas que o estudante tem de dar para conseguir subir ou descer o seu nível no sistema, sem depender dos restantes fatores.

Então, como é realizada a seleção adaptativa das questões lançadas a um estudante num *quiz*? Para dar resposta a esta questão, entrar-se-á em detalhe acerca do algoritmo de inferência utilizado pelo mecanismo para inferir a questão mais apropriada para um dado momento de avaliação, enumerando as regras estabelecidas.

O primeiro filtro assenta, como esperado, em selecionar as questões cujo domínio coincide com o que o estudante pretende praticar. Resume-se à procura apenas de questões cujo ciclo de estudos, escolaridade e descrição, as três propriedades que caracterizam um domínio, coincidem com os estipulados pela temática em estudo. Pensando num exemplo mais concreto, um estudante universitário, que frequente o 3º ano do Mestrado integrado em Engenharia Informática (MiEI), ao utilizar este modelo para avaliar o seu nível de aprendizagem numa certa unidade curricular (UC), estaria a despoletar no sistema um filtro por domínio no qual o nome da UC seria a descrição, a informação do curso e respetivo ano seriam a escolaridade e, claro está, o ciclo de estudos encaixaria na categoria equivalente ao Ensino Superior.

Segue-se a especificação do nível de dificuldade da questão. Aqui, o motor de raciocínio decide entre dois caminhos possíveis:

1) seleciona o nível de dificuldade coincidente com o nível de conhecimento do aluno, que lhe é transmitido pelo componente responsável por esta análise;
2) utiliza o valor por omissão configurado para o domínio em avaliação, caso a interação aluno-tutor esteja ainda nos seus primórdios (Figura 6).

Na prática, quando o aluno acede ao *software* e seleciona um domínio para estudar, o motor de avaliação questiona o módulo responsável por traçar o perfil do estudante:

"Existem dados sobre o conhecimento no domínio X para o estudante Y?"

Se este não possuir informação prévia sobre o conhecimento do aluno no domínio em causa, o componente de avaliação lançará questões com um nível de dificuldade predefinido. Importa frisar que, após a primeira questão respondida pelo aluno no domínio, considera-se que o nível de conhecimento está incluído no perfil do estudante, juntamente com outras informações abordadas adiante.

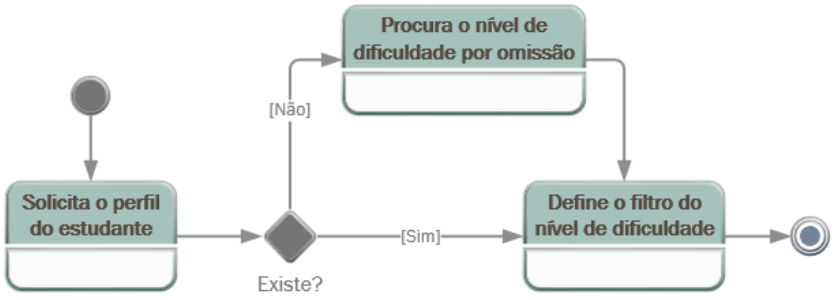


Figura 6: Máquina de estados representativa da aplicação do filtro do nível de dificuldade.

Embora a classificação do conhecimento do estudante não esteja a cargo do mecanismo de avaliação, pressupõe-se que o módulo com esta tarefa tem em conta sequências de respostas corretas e incorretas (*backlog*) para acelerar a variação do nível de conhecimento do aluno. Concetualmente, isto permite que um estudante atinja mais rapidamente a classificação do sistema para o nível de conhecimento em que se enquadra. Assim, no motor de inferência existe uma regra de *backlog,* algo que foi criado para dificultar subidas de nível fortuitas e descidas azaradas. Quando o utilizador em teste se encontra a uma resposta de completar a sequência mínima de *backlog*, o motor de raciocínio altera o nível que lhe é transmitido pelo modelo do estudante, incrementando-o ou decrementando-o em uma unidade, consoante falte uma resposta para o módulo do estudante subir ou descer o nível do aluno. Para tal, o parâmetro `backlog_factor´ da configuração do domínio, em conjunto com o nível de conhecimento do aluno, é utilizado para determinar em que questão esta regra é ativada. Logicamente, nos níveis da extremidade da escala – limite mínimo 0 e limite máximo 5 -, a regra deixa de ser, parcialmente, válida. Veja-se o seguinte exemplo:

- Supondo que são necessárias 5 respostas consecutivas para se ativar a regra de *backlog*, a um aluno com nível de aprendizagem classificado em 3 podem ocorrer dois cenários:
  - O aluno, após 4 respostas certas consecutivas, recebe uma questão de nível de dificuldade 4;
  - O aluno, após 4 respostas erradas consecutivas, recebe uma questão de nível de dificuldade 2.

A partir da primeira questão lançada num *quiz*, outro aspeto começa a ser considerado no momento de selecionar uma questão. A cada iteração torna-se mais importante controlar a lista de questões lançadas anteriormente no teste. Este controlo, que pode ser descrito como a "regra das repetições", surge da análise de uma lista de `id´'s que chega ao módulo de avaliação, disponibilizada por outro componente do tutor artificial, que está encarregue deste registo. Esta lista deverá conter tantos identificadores quantas as questões já lançadas, podendo conter valores repetidos, portanto. O resultado da sua análise será uma lista com os `id´'s das perguntas que não deverão voltar a ser colocadas ao aluno. Nesta decisão intervém o campo `repetitions´ da estrutura de uma pergunta, do seguinte modo:

- ao ser percorrida a lista de questões lançadas, é calculada a cardinalidade das ocorrências de um dado identificador;
- para cada identificador, verifica-se se já surgiu o número máximo de vezes, com base no campo `repetitions´ da questão que identifica;
- o identificador é adicionado à lista de `id´'s excluídos do restante teste, caso a resposta no passo anterior seja afirmativa.

Excluindo qualquer restrição causada por outros parâmetros, analise-se um exemplo concreto:

- suponha-se um *quiz* para o qual existe um conjunto de três perguntas possíveis, com identificadores Q1, Q2 e Q3;
- Q1 e Q2 representam questões irrepetíveis (`repetitions´ = 0). Já Q3 identifica uma questão que pode ser repetida uma vez (`repetitions´ = 1);
- para a primeira questão do *quiz,* aleatoriamente é selecionada a questão Q2;
- fica, assim, desde logo excluída a possibilidade de surgir a questão Q2 novamente, ficando disponíveis as questões Q1 e Q3;
- admitindo-se que, na segunda iteração, é selecionada a questão Q3, nenhum `id´ é adicionado à lista de excluídos, em função da regra das repetições;

- na iteração seguinte do teste, poderá novamente ser lançada uma de duas questões possíveis: Q1 ou Q3.

Além de auxiliar o mecanismo de avaliação a lidar com a repetição de questões, a lista dos identificadores das questões já lançadas permite também aplicar a "regra das precedências". Recorde-se, uma questão pode requerer que outras a antecedam num teste, através do seu campo `precedences´. Em termos concretos, esta restrição traduz-se em questões que não podem ser lançadas até que todos os identificadores presentes na sua lista de precedências constem na lista de `id´'s das questões avaliadas até ao momento. É evidente que, na primeira iteração do *quiz*, são apenas válidas questões sem precedências. Para uma melhor demonstração da regra, recupere-se o exemplo anterior, alterando-o a propósito:

- assumindo que a questão Q1 tem como precedências as questões Q2 e Q3, a primeira seleção contemplaria apenas duas hipóteses: Q2 ou Q3;
- mantendo a seleção aleatória da questão Q2 para primeira questão do *quiz*, a junção das regras das repetições e das precedências torna determinístico que a questão lançada na segunda iteração seria a Q3.

Este controlo da precedência e repetição de questões, por parte do motor de inferência, é de extrema importância para o correto funcionamento de um teste, no sentido de respeitar a pedagogia transmitida pelos autores aquando da criação das questões.

A dificuldade incremental dos *quizzes* ao longo do processo de aprendizagem (em crescendo) de um aluno é assegurada, não só, pelo aumento no parâmetro do nível de dificuldade, como também por regras que se aplicam apenas a estudantes com um certo nível de conhecimento reconhecido pelo *software.* Em primeira instância surge a seleção do subdomínio, quando o aluno ascende ao nível de aprendizagem intermédio, e posteriormente a seleção do subsubdomínio, assim que o aluno sobe ao nível seguinte. Neste processo intervém novamente o módulo responsável por traçar o perfil do estudante, fornecendo informações acerca do seu desempenho nos vários conteúdos

pertencentes ao domínio testado. Este registo, que é fortalecido à medida que se acumulam as questões respondidas, permite ao avaliador inferir o subdomínio e respetivo subsubdomínio nos quais o estudante revela maiores dificuldades. Por sua vez, refletindo esta informação na filtragem de uma questão, é possível complicar a tarefa do aluno, pondo ainda mais à prova o seu conhecimento. Veja-se um exemplo:

- Cenário: domínio de Bases de Dados, com dois subdomínios possíveis – 'SQL' e 'MongoDB' –, cada um com dois subsubdomínios, neste caso comuns – 'Projeção' e 'Agregação'.
- Se o estudante revelar um pior desempenho no subdomínio de 'SQL', ou seja, se a porção de respostas corretas em questões desta temática é inferior comparativamente ao que ocorre com questões sobre 'MongoDB', este será o conceito avaliado na próxima pergunta.
- Num nível de particularização mais avançado, se a responder a questões sobre 'Agregação' o desempenho do aluno é inferior, este será o subsubdomínio selecionado para a seguinte iteração do teste.

Dado que o subsubdomínio não é um campo obrigatório na estrutura da questão, a aplicação da regra de filtragem por esta propriedade pode não acontecer, mesmo que o nível de conhecimento do aluno já o permita.

Embora se admita que um ITS, antes da sua utilização em contextos pedagógicos reais, vê construída para si uma base de conhecimento sólida, com um número de questões diversificado e vasto, num qualquer domínio de conhecimento passível de ser estudado por um aluno, é dever do mecanismo de avaliação precaver-se para situações em que não existem questões que respeitam todos os filtros. Em termos tecnológicos, a ausência de prevenção desta situação poderia fazer o sistema entrar em *deadlock*, aguardando infinitamente por uma questão inexistente. Assim, o que acontece neste sistema é que alguns dos filtros aplicados pelas regras anteriores são removidos, na tentativa de obter resultados na seleção da próxima pergunta. Em primeira instância, o motor de

inferência retira o filtro de seleção por subsubdomínio, uma vez que se trata da maior particularização concetual da questão. Se mesmo sem este filtro o motor não conseguir lançar uma questão, o subdomínio é o próximo campo a abandonar o filtro de questões. Passa-se, então, a procurar perguntas de um qualquer conceito incluído no domínio avaliado, porém, isto pode ainda não ser suficiente.

A última medida do mecanismo, no âmbito de prevenir filtragens inconclusivas, consiste em alterar o nível de dificuldade da questão procurada, podendo ocorrer comportamentos distintos face ao desempenho do estudante. O relevo do "fator de elevado desempenho" para o módulo de avaliação surge neste momento, já que é este valor que determinará se a dificuldade das questões filtradas será aumentada ou reduzida. Havendo margem para incrementar o nível, isto é, se este não for já o nível máximo, e um desempenho acima do fator configurado para o domínio, a alteração que ocorre no momento de seleção é a subida do nível de dificuldade esperado para a questão. Por sua vez, a descida de nível ocorrerá caso o nível anteriormente filtrado não seja o mínimo e o desempenho seja igual ou inferior ao fator. Uma terceira hipótese é a remoção do nível de dificuldade, que se aplica quando as condições anteriores não se verificam, medida esta que roça o limiar do erro pedagógico, pois questões de declarada facilidade poderão ser colocadas a alunos de topo, e vice-versa, descontrolando o nível de exigência do *quiz* do dado aluno.

A partir da remoção do filtro de nível de dificuldade, o comportamento do sistema de avaliação resume-se ao lançamento de qualquer questão do domínio que respeite repetições e precedências, até que não restem mais questões. Sintetizando a informação transmitida entre os dois componentes do tutor artificial mais destacados – o mecanismo de raciocínio da avaliação e o modelo do estudante –, a figura seguinte (Figura 7) simboliza também a utilização da configuração do domínio de conhecimento em avaliação, no processo de seleção adaptativa da próxima questão.

Figura 7: Comunicação entre módulos no processo de seleção.

3.1.4 Lançamento e Avaliação

A seleção adaptativa da questão que constará na iteração seguinte de um teste é o processo complexo que antecede o lançamento da questão ao utilizador-alvo – o aluno. Se a seleção é importante para personalizar a avaliação e a aproximar do estado atual de aprendizagem do aluno, o momento de lançar marca o início do contacto entre sistema e estudante, quando a pergunta escolhida faz o seu percurso até à interface, a qual tem de assegurar certos requisitos.

Sabe-se que uma questão possui um tempo máximo de resposta – `answering_time´ – estipulado em segundos. Deste modo, uma funcionalidade que deve ser proporcionada pela interface do *quiz* é a presença de um temporizador, para a gestão do tempo de resposta, da parte dos estudantes. Terminado o tempo máximo para responder, o aluno deve ser impossibilitado de responder ou alterar a sua resposta, sendo-lhe lançada a questão seguinte. É importante que este controlo não esteja apenas ao nível da interface, sendo da competência do motor de raciocínio verificar se o tempo entre o lançamento da questão e a recolha da resposta não excede o máximo permitido.

Quando o sistema recolhe uma resposta válida, o passo seguinte é a sua avaliação. Aqui, o papel do avaliador consiste na análise da correção da resposta, ou seja, se a opção selecionada corresponde à resposta correta. Juntamente com o tempo de resposta, a conclusão obtida é enviada ao componente do tutor artificial responsável por acompanhar o progresso do estudante. Com estes

dados, o modelo do estudante é atualizado, nomeadamente o seu desempenho e a sua aptidão na temática em avaliação, o que porventura se reflete na classificação atribuída ao nível de conhecimento do aluno. A partir deste momento, o mecanismo de avaliação fica em condições de iniciar um novo processo de seleção adaptativa, para lançar a próxima questão mais apropriada para o estudante em avaliação.

3.2 Esquematização do Processo de Avaliação

Agora que o processo iterativo gerido pelo avaliador foi descrito na totalidade, apresenta-se um esquema representativo dos diversos momentos do processo de avaliação (Figura 8). Deste diagrama de atividades, excluiu-se a etapa de preparação de questões, uma vez que esta operação não intervém diretamente na interação do estudante com o tutor, nem tem de ocorrer necessariamente antes de cada *quiz.* Assim, a representação começa na ação de iniciar um *quiz,* por parte de um aluno. Como se pode comprovar pelas conexões entre atividades, na avaliação não existe interação direta entre o estudante e o modelo que o representa no sistema de tutoria inteligente. A comunicação é feita através do sistema avaliador e do seu motor de raciocínio, que coordena o processo.

O comportamento do sistema em situações de simultaneidade de processos de avaliação pode, eventualmente, levantar questões. Todavia, o mecanismo encontra-se preparado para lidar com mais do que um utilizador simultaneamente, algo que, em situação contrária, seria bastante reprovável, tratando-se de um componente de *software* desenvolvido no âmbito de um projeto com vista à construção de um ITS acessível na Web. Então, como se faz a gestão de vários processos de avaliação em curso?

Nos dias de hoje, qualquer *framework* básico de desenvolvimento Web fornece um conjunto de funcionalidades chave que auxiliam a equipa por detrás do desenvolvimento, contribuindo para a

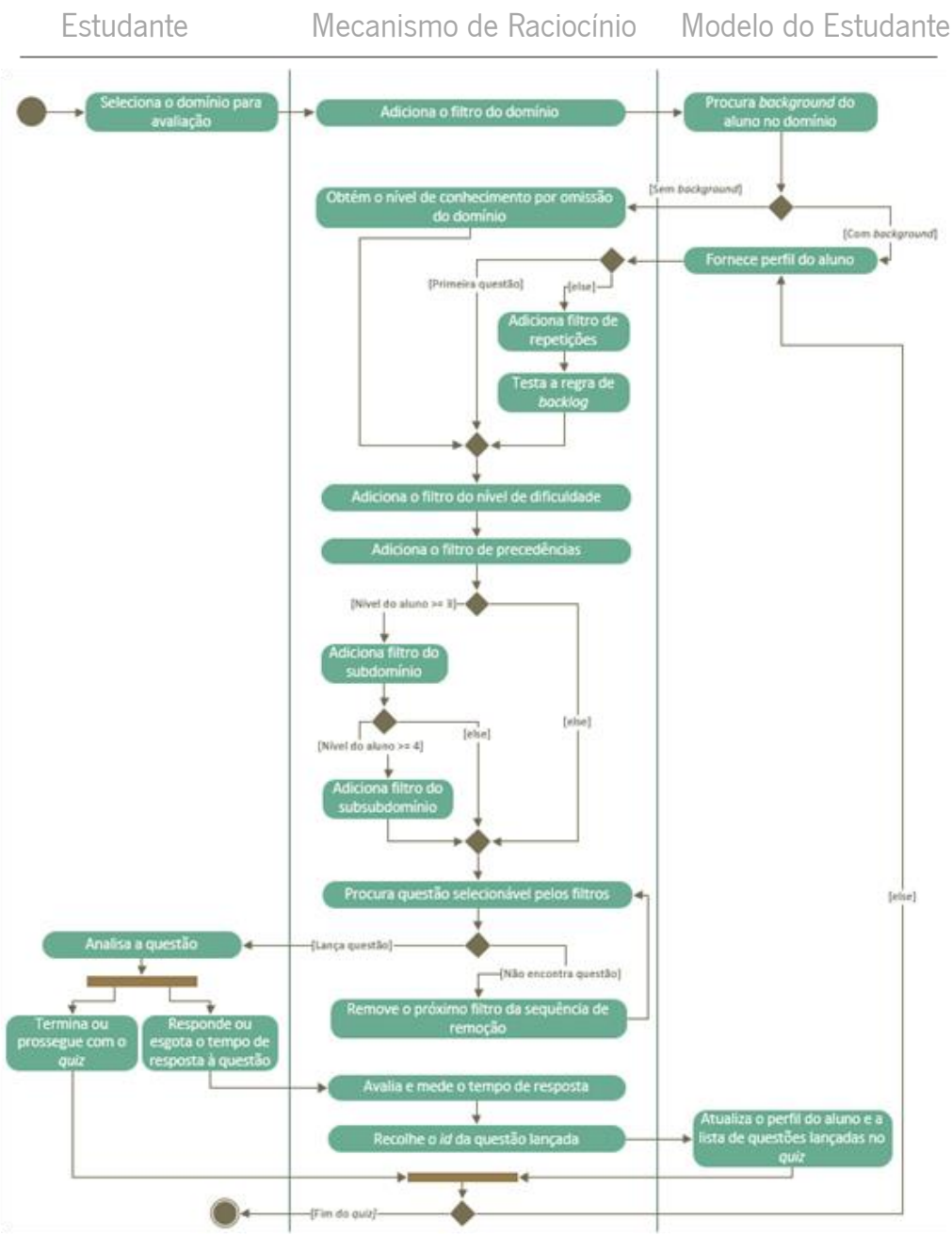


Figura 8: Diagrama de atividades do fluxo de interação na avaliação.

agilização do período de implementação. Uma dessas funcionalidades trata-se, precisamente, do controlo de sessões. Embora o pensamento mais correto seja o de acessos diferentes através de um *browser*, sessões simultâneas podem simbolizar estudantes distintos, neste contexto. Cada estudante (sessão) recebe um código identificador – um *token* – que o representará enquanto perdurar no sistema, acompanhando os seus pedidos HTTP ao servidor. Dado que o protocolo HTTP é *stateless* por natureza, esta identificação por *token* da sessão é o mecanismo por norma utilizado para que o servidor Web consiga fornecer conteúdo para um cliente (aluno) específico. No seio desta sessão, não existem processos de avaliação simultâneos. Logo, ocorrem da forma que foi esquematizado no diagrama de atividades apresentado na Figura 8.

A concorrência entre processos de avaliação de alunos distintos dá-se na resposta aos pedidos HTTP e ao nível da base de dados - na leitura de questões, por exemplo. Contudo, esta gestão está incluída no ambiente proporcionado pelo *framework.* Para culminar este capítulo, sumaria-se o seu conteúdo com um esquema dos processos de multiutilização do sistema de avaliação (Figura 9).

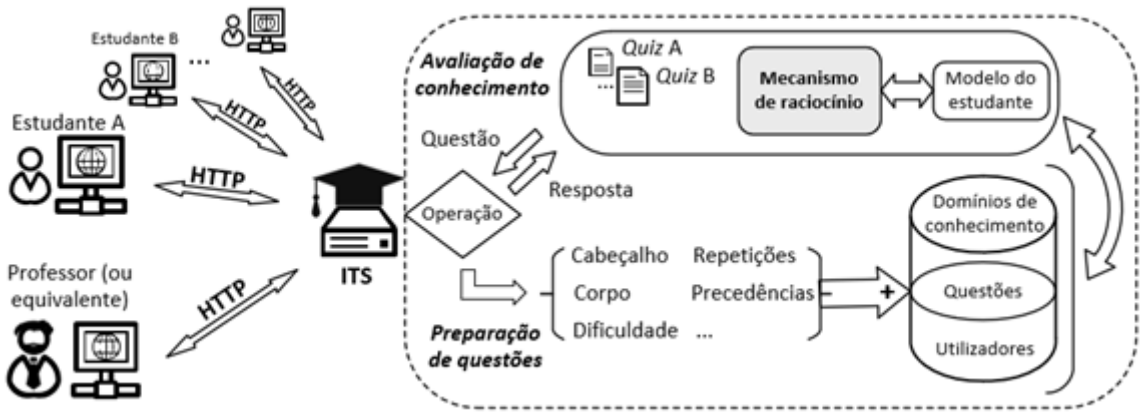


Figura 9: Esquema de processos com a multiutilização do ITS no sistema de avaliação.

Capítulo 4

Um Caso de Aplicação

4.1 O Sistema 'Leonardo'

O sistema 'Leonardo' surgiu de uma iniciativa de I&D interna com o objetivo de fornecer uma plataforma educacional *online*, especialmente orientada ao suporte de processos de avaliação em cursos de Licenciaturas da Universidade do Minho. O suporte assenta numa plataforma *web-based*, que assegura os serviços principais – registo de utilizadores, manutenção da base de conhecimento, armazenamento de estatísticas, entre outros. Além desta plataforma, planeia-se a adição de outra, especialmente adaptada a dispositivos móveis, cujo objetivo passa por melhorar a interface e a usabilidade dos processos de avaliação, neste tipo de dispositivos.

Os objetivos da iniciativa em desenvolvimento passam, então, pela criação de um tutor inteligente com capacidade de adaptação aos perfis de conhecimento revelados pelos alunos num dado conteúdo. Através da recolha de informação em diferentes sessões de teste, o sistema pretende delinear progressivamente o estado de aprendizagem do estudante, numa seleção de conceitos, a fim de decidir os tópicos mais importantes em avaliações futuras. Este processo é designado de *profiling* e procura que o sistema seja capaz de implementar uma avaliação apoiada deste

conhecimento dos alunos, recomendando elementos de estudo de acordo com as fragilidades que os mesmos revelam ao longo do processo de interação homem-máquina. Além dos objetivos concetuais anteriores, outro objetivo mais técnico, e que se prende com a necessidade da mencionada plataforma adicional, consiste em tornar o tutor um sistema de estudo com disponibilidade 24h, acessível a partir de qualquer plataforma computacional (*cross-platform).* Deste modo, o sistema incentiva o aluno interessado a pôr à prova o seu conhecimento, independentemente do momento e do dispositivo que tenciona utilizar na interação. Já estando decorridas as primeiras fases do projeto, importa abordar os aspetos técnicos que foram previamente definidos acerca do sistema, como acontecerá nas próximas secções deste capítulo. No final, será descrita a integração no sistema *Leonardo* do mecanismo de raciocínio desenvolvido nesta dissertação.

4.2 Arquitetura do sistema

A arquitetura do sistema foi delineada segundo o que é comum num ITS, com algumas nuances no que à sua representação diz respeito. Na Figura 10 pode-se observar essa arquitetura, na qual se identificam desde logo três grandes módulos: módulo de edição, módulo de raciocínio e, ao centro, o módulo da base de conhecimento. Este módulo central foi o ponto fulcral do desenvolvimento inicial do *Leonardo,* constituindo a base do sistema, onde será guardada toda a sua informação. A base de conhecimento é alimentada por dois tipos de agente – aluno e utilizador qualificado – descritos como 'Estudante e 'Perito', respetivamente, na Figura 10.

Continuando a analisar a arquitetura apresentada, em torno do módulo central vê-se o módulo de edição, que permite a passagem direta de informação, por parte dos utilizadores qualificados, à base de conhecimento, o que inclui exercícios para o aluno, devidamente acompanhados do material de estudo apropriado ou de referências para o mesmo, mas também algumas configurações, utilizadas no momento de inferir qual a próxima ação do sistema, durante a interação

com o aluno. Do lado oposto, encontra-se o módulo de raciocínio, o componente que foi alvo desta dissertação. O sistema utiliza um motor de inferência baseado em regras de produção (*rule-based*), definidas com base num diversificado conjunto de heurísticas adquiridas ao longo de anos de prática do ensino em diferentes cursos (Figura 11).

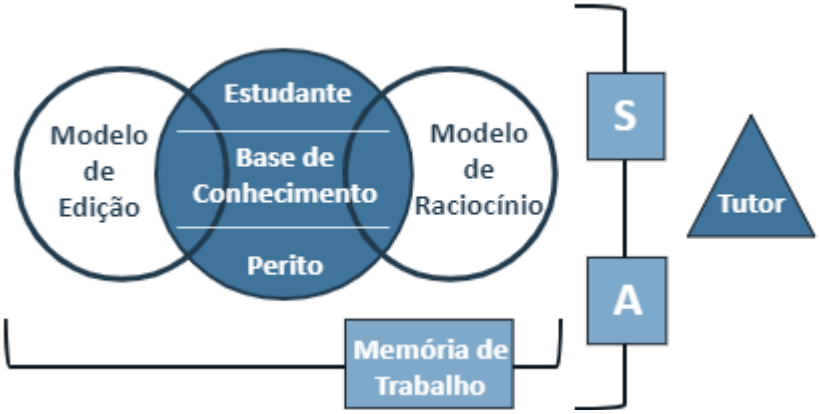


Figura 10: A arquitetura do sistema *Leonardo*.

Este motor trabalha sobre a base de conhecimento que contém as estruturas de suporte ao funcionamento do sistema – metodologia de avaliação, dados de caracterização do estudante (*profiling*), configurações de domínios de conhecimento –,que lhe conferem a capacidade de definir, em tempo real, a melhor estratégia de avaliação para determinar o estado atual do conhecimento de um aluno.

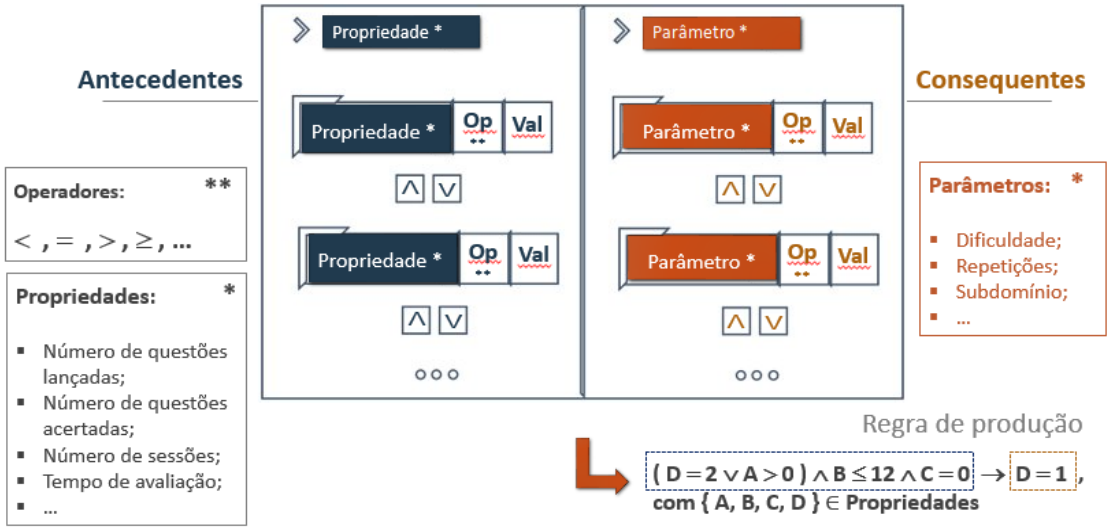


Figura 11: A lógica presente nas regras de produção do *Leonardo*.

Na interligação entre a base de conhecimento e o mecanismo de raciocínio opera o módulo do Profiler, componente que representa o modelo do estudante neste sistema. Este componente é responsável pelo *profiling* – construção do perfil – de um aluno, através da definição de um modelo que parametriza o conhecimento do aluno, com base nas suas prestações na resolução de problemas. Tem um papel ativo nos processos de avaliação.

Na Figura 10, é possível visualizar, também, a presença de uma memória de trabalho, que simboliza o armazenamento temporário de informação associada ao desempenho de tarefas cognitivas, como leitura, resolução de problemas ou aprendizagem (Baddeley, 1992). O seu objetivo nesta arquitetura é sustentar o comportamento do tutor em atividade, na medida em que, ao ser carregado para memória o perfil do aluno com quem o sistema está a interagir, este consegue, assim, acesso a informação relevante sobre o conhecimento do estudante, o que lhe permite adaptar a sua ação ao mesmo. Por sua vez, as formas quadrangulares com um S e um A representam, respetivamente, os Sensores e Atuadores do ITS na interação com o utilizador. Estão, portanto, intrinsecamente relacionados com o módulo da interface do utilizador, característico da estrutura de um ITS. Neste caso particular, enquadram-se no ambiente gráfico proporcionado pela aplicação Web, com o qual o aluno interage por meio de cliques e *input* textual, ainda que esteja nos planos futuros do *Leonardo* o acionamento de comandos de voz.

4.3 Ferramentas utilizadas

Modelado o sistema, o passo seguinte consistiu na escolha das ferramentas a utilizar, para que se pudesse dar início, de facto, à implementação do tutor. As decisões foram alusivas às ferramentas a utilizar para criar a base de conhecimento do tutor e também a base da sua aplicação e respetiva interface gráfica. Na escolha do motor de base de dados a utilizar, pesou a necessidade de o sistema de informação ser capaz de suportar grandes quantidades de dados, processando-os de

forma rápida, uma vez que são estes dados o conhecimento do tutor e é a sua manipulação que o torna inteligente. Assim, esta necessidade de elevado desempenho e escalabilidade no armazenamento e utilização da informação na base de conhecimento levou à escolha de um SGBD - *Sistema de Gestão de Bases de Dados* - não relacional (NoSQL). De entre as possibilidades (MongoDB, Cassandra, Neo4j, etc), a opção final foi MongoDB (MongoDB, 2019), um mecanismo orientado a documentos. A necessidade de acesso rápido a dados, para uma resposta atempada do tutor, torna as bases de dados orientadas a documentos a escolha ideal, porque, em troca de alguma redundância na informação armazenada, oferecem a vantagem de um só documento poder conter informação diversa, evitando o que seria a junção com dados de outras tabelas num modelo relacional, por exemplo.

No que à aplicação diz respeito, com o intuito de concretizar o objetivo de disponibilizar um sistema multi-plataforma e disponível 24h, a opção recaiu sobre uma *webapp*. Com a implementação de uma interface *responsive*, capaz de se adaptar a ecrãs de diferentes tamanhos, o *software* permite que o aluno possa aceder ao tutor em praticamente qualquer dispositivo com acesso à Internet, desde *smartphones* a computadores d*esktop*. A escolha do *framework* Web a utilizar na implementação foi encurtada pela decisão de se utilizar a linguagem de programação Python (Python.org, 2019) no servidor da aplicação, muito devido às diversas bibliotecas de análise de dados disponíveis nesta linguagem. Neste âmbito, o facto de o *framework Flask* (Pallets, 2019) permitir maior controlo sobre os componentes a usar no desenvolvimento, como é o caso da base de dados e da forma como a aplicação interage com esta, motivou a sua escolha.

4.4 Representação do Conhecimento

Para pôr à prova e testar assertivamente as competências do estudante, o elemento manipulado nas estratégias de formação e avaliação do mecanismo de raciocínio é a questão. De facto, este é o elemento primário da componente lúdica e avaliativa do sistema. Na adaptação de conteúdos ao

estudante por meio de técnicas de *profiling*, os dados em estudo são, precisamente, as questões lançadas ao aluno e as suas respostas, aliadas a dados do sistema sobre o momento em que a questão foi colocada. Posto isto, necessitar-se-á do conhecimento da estrutura atribuída a uma questão, ainda que parte da sua constituição tenha sido abordada no capítulo anterior. De seguida, apresenta-se os vários campos que formam o documento JSON:

- **id**: Código único identificador da questão;
- **domain**: domínio de conhecimento em que se enquadra a questão; este é definido por um documento com três campos, nomeadamente:
  - **study_cycle**: ciclo de estudos em que se insere o domínio, segundo a classificação portuguesa (ex: Ensino Superior);
  - **scholarity**: nível de escolaridade dentro do ciclo de estudos a que pertence o domínio, segundo a classificação portuguesa (ex: Engenharia Informática);
  - **description**: designação do domínio (ex: Bases de Dados);
- **subdomain**: designação do subdomínio, isto é, uma particularização do domínio. Trata-se da especificação do tema da pergunta (ex: SQL);
- **subsubdomain**: designação do subsubdomínio, um campo opcional que representa um nível de especificação adicional ao subdomínio (ex: Projeções);
- **header**: cabeçalho da questão, ou seja, o texto que a verbaliza;
- **body**: lista de respostas possíveis para a questão. Este campo trata-se, assim, de uma lista de documentos embebidos compostos por:
  - **answer**: texto que verbaliza a opção de resposta;
  - **correction**: fator de correção da opção de resposta (0 -> opção incorreta | 1 -> opção correta);

  - **mandatory**: valor booleano que indica se esta opção de resposta deve constar sempre na lista de opções exibidas;

- **difficulty_level**: grau de dificuldade da questão;
- **answering_time**: tempo máximo, em segundos, que o aluno possui para responder à questão;
- **type**: tipo de ação que a questão requer, isto é, se é um exercício, uma definição, uma regra, entre outros;
- **solution**: exposição da resposta correta à questão e/ou do procedimento para a sua resolução;
- **source**: referência à fonte de onde a questão foi retirada. É um campo opcional, para cobrir situações em que a questão inserida pelo docente não é original;
- **notes**: campo onde, opcionalmente, são colocadas notas gerais relativas à questão;
- **language**: idioma em que está escrita a questão e toda a informação relacionada;
- **inserted_at**: data de inserção da pergunta na base de dados. Importante na análise de utilização do *software*;
- **inserted_by**: identifica o utilizador que introduziu a questão, através de dois campos:
  - **id**: identificador único do utilizador;
  - **name**: nome do utilizador;
- **validated_at**: data de validação da questão, por parte de um utilizador com permissões para tal;
- **validated_by**: identifica o utilizador que validou a questão, através dos campos:
  - **id**: identificador único do utilizador;
  - **name**: nome do utilizador;

- **status**: campo que representa o estado da pergunta, ou seja, se está em condições de ser lançada ou se, por exemplo, aguarda validação;
- **repetitions**: indica o número de vezes que a questão pode ser repetida (ver no Capítulo 3);
- **precedence**: lista de questões que devem ser lançadas antes desta questão (ver no Capítulo 3);
- **display_mode:** a forma como as possíveis respostas aparecerão ao utilizador do sistema (ver no Capítulo 3).

Outra estrutura relevante para o sistema e para esta dissertação é a de representação dos domínios de conhecimento no sistema. Também esta estrutura foi previamente alvo de atenção neste documento, novamente de forma parcial. Os campos `default_user_level´, `high_performance_factor´ e `backlog_factor´ foram introduzidos no Capítulo 3 de forma genérica como "nível de conhecimento do aluno por omissão", "fator de elevado desempenho" e "fator de *backlog*", respetivamente. Estes são parte constituinte do lote de parâmetros que o campo `config´ reúne. A composição total de um domínio consiste em:

- **id**: identificador único atribuído ao domínio pelo sistema;
- **study_cycle**: especifica o ciclo de estudos em que se enquadra o domínio -Ensino Superior, Ensino Secundário, entre outros exemplos;
- **scholarity**: referência adicional para particularizar o domínio inserido. Pode-se referir ao ano do ciclo de estudos ou ao curso específico, por exemplo;
- **description**: designação atribuída ao domínio;
- **subdomains**: lista dos subdomínios associados ao domínio. Cada subdomínio é composto por:
  - **id**: este é derivado do identificador do domínio a que pertence, ou seja, se o *id* do domínio é "1", o seu primeiro subdomínio terá *id* "1.1";

- **description**: descrição do subdomínio;

- **subsubdomains**: lista dos subsubdomínios associados ao domínio. Assim como num subdomínio, também um subsubdomínio é composto por:
  - **id:** derivado do identificador do subdomínio, logo o primeiro subsubdomínio seria identificado por "1.1.1", seguindo o exemplo anterior;
  - **description**: descrição do subsubdomínio;

- **config**:
  - **default_user_level**: a classificação predefinida do nível de conhecimento do aluno (ver no Capítulo 3);
  - **high_performance_factor**: o valor de desempenho mínimo para a subida do nível do aluno (ver no Capítulo 3);
  - **low_performance_factor**: valor de desempenho máximo para a descida do nível do aluno. Idêntico ao `high_performance_factor´, porém com intervenção na redução da dificuldade das questões;
  - **high_skill_factor**: valor de aptidão mínimo para a subida do nível do aluno. À semelhança do `high_performance_factor´, sem o aluno revelar uma aptidão suficientemente satisfatória, isto é, maior ou igual a este valor, o nível de dificuldade das questões que recebe, em condições normais, não aumentará;
  - **low_skill_factor**: valor de aptidão máximo para a descida do nível do aluno. Idêntico ao fator anterior, porém com intervenção na redução da dificuldade das questões;
  - **min_questions_number**: número mínimo de questões predefinido para qualquer nível de conhecimento; em termos práticos, este valor define a quantidade de questões a que um aluno tem de responder, atingido um dado nível, para ficar habilitado a subir ou descer para o nível seguinte;

- **questions_factor**: intervém no número mínimo de questões a responder num nível, incrementando-o à medida que o nível sobe. Se este fator for 0, a quantidade mínima de questões será igual a `min_questions_number´;
- **backlog factor**: o valor de respostas certas ou erradas consecutivas que o estudante tem de dar para conseguir subir ou descer o seu nível no sistema (ver no Capítulo 3).

A estes campos dos documentos de domínios, subdomínios e subsubdomínios acrescentam-se, ainda, os identificadores dos utilizadores que os inseriram e validaram, bem como a data em que estas operações ocorreram, como acontece para a estrutura da questão. Outras informações, como utilizadores, dados de *profiling* e registos das ações no sistema, também possuem uma coleção de documentos própria. Não obstante o seu detalhe não ser necessário, importa mencionar a distinção realizada entre os diferentes tipos de utilizador do *Leonardo*. São considerados 4 tipos distintos – administrador, responsável, operador e estudante – enumerados por ordem decrescente de privilégios. Entre as diferenças estão, por exemplo, a permissão para adicionar um novo domínio de conhecimento ao sistema (apenas administradores) ou para inserir uma nova questão de um domínio existente (todos à exceção do estudante). Quando neste documento se associa o termo professor ao sistema, em termos práticos, refere-se um utilizador com estatuto de responsável sobre um domínio, que pode, assim, editar os seus parâmetros configuráveis e adicionar-lhe subdomínios e subsubdomínios, além de questões.

4.5 Processo de Avaliação

Nesta altura, está-se em condições de fazer a descrição do funcionamento do componente de avaliação no seio do sistema, realçando a sua integração com os restantes módulos. Antes de uma avaliação poder ocorrer, cabe a um administrador registar o domínio de conhecimento no sistema e inserir um utilizador responsável pelo mesmo (assumindo que o administrador não acumula esta função). Por sua vez, o responsável fica a cargo de configurar o domínio, mediante o que pretende

para a avaliação, e de especificar o conhecimento sob a forma de subdomínios, subsubdomínios e questões, além de introduzir os seus alunos no sistema. Se assim o entender, pode também adicionar um operador, para o auxiliar em parte destas tarefas.

Na inserção de questões intervém o módulo de edição, no qual é solicitado ao utilizador qualificado o preenchimento do conjunto de parâmetros, que foram abordados na secção de representação de conhecimento, dos quais uns têm carácter obrigatório e outros têm carácter facultativo, dando assim conteúdo à questão. As questões são, depois, armazenadas na base de conhecimento, que deve ser tão abrangente quanto possível, cobrindo os vários níveis de dificuldade existentes, pois só assim o sistema será capaz de avaliar criteriosamente as diferenças no estado de aprendizagem dos estudantes.

Passando à análise do *quiz* propriamente dito, destaca-se a interação entre o motor de inferência e o Profiler, que ocorre por intermédio da troca de objetos JSON, formato escolhido para protocolo de comunicação entre os módulos do *Leonardo*. Numa fase inicial, isto é, quando o aluno é novo no sistema e o seu perfil ainda está a ser delineado, o motor recorre à configuração do domínio para lançar questões, nomeadamente ao campo `default_user_level´. Contudo, desde logo o Profiler assume o seu papel indispensável, registando as questões que vão sendo respondidas no *quiz.* Esta informação é, como se mostrou, necessária para o controlo da repetição de questões e gestão de precedências, mas não só. Internamente, o Profiler acumula dados das várias sessões de avaliação do aluno e, como tal, consegue determinar quantas questões de um dado nível de dificuldade este já respondeu, desde o seu registo até à sessão atual.

O histórico de questões permite ao módulo de *profiling* controlar as transições de nível de conhecimento do aluno, que podem ocorrer quando o aluno 1) responde ao número mínimo de questões ou 2) completa uma sequência mínima de respostas com o mesmo grau de correção (*backlog*), para o seu nível de conhecimento atual. Nesta gestão intervém a configuração do domínio, da seguinte forma:

- quanto mais avançado na sua aprendizagem um estudante se encontra, maior é a quantidade de respostas que terá de acertar ou falhar para contrariar o normal rumo do nível de dificuldade das questões;
- em condições normais, ou seja, sem surgir *backlog* que permita acelerar a variação da dificuldade, um aluno tem de responder a um mínimo de questões calculado com recurso ao seu nível de conhecimento atual e aos parâmetros `questions_factor´ e `min_questions_number´ do domínio de conhecimento em avaliação; ao atingir esta quantidade mínima de questões, a mudança de nível depende ainda do desempenho e da aptidão do aluno no momento, além dos fatores `high_performance_factor´, `low_performance_factor´, `high_skill_factor´ e `low_skill_factor´;
- nas situações de *backlog*, o nível de conhecimento do aluno associa-se ao `backlog_factor´ para determinar o número de respostas certas ou erradas consecutivas que este necessitará de dar para poder transitar de nível sem depender da quantidade de questões, nem dos seus valores de desempenho e aptidão.

Para garantir a atualização do nível conforme os princípios acima, o modelador do estudante contabiliza, não só, o número de questões de um certo nível respondidas, como também a quantidade de respostas certas ou erradas consecutivas. Com estes cálculos, aliados à listagem de questões lançadas no *quiz*, o Profiler fica em condições de transmitir ao mecanismo de raciocínio a informação que este necessita para selecionar a próxima questão.

O processo de seleção de questões foi analisado no Capítulo 3, com a pormenorização das regras utilizadas pelo motor de inferência do sistema, portanto, no que resta deste capítulo far-se-á ênfase à ligação entre o módulo de avaliação desenvolvido e, numa primeira fase, a interface de utilizador do sistema e, depois, o Profiler. Através de evidências gráficas – *screenshots* da aplicação Web –, mostra-se como o sistema de avaliação de conhecimento interage com o utilizador, e vice-versa. Para tal, considere-se o seguinte excerto do domínio de conhecimento para a demonstração:

```
{
    study_cycle: 'Ensino Superior',
    scholarity: 'Engenharia Informática',
    description: 'Sistemas de Bases de Dados',
    config: {
        default_user_level: 2,
        high_performance_factor: 0.7,
        low_performance_factor: 0.3,
        high_skill_factor: 0.6,
        low_skill_factor: 0.2,
        backlog_factor: 4,
        questions_factor: 2,
        min_questions_number: 3
    },
    ...
```

Como é suposto, um perito nesta área deve alimentar a base de conhecimento com questões. Para tal, o sistema dispõe de uma página, cujo aspeto parcial se exibe na Figura 12, onde o utilizador qualificado preenche o cabeçalho da questão, insere as suas várias opções de resposta, seleciona um nível de dificuldade, específica o domínio e as suas variantes, entre outras informações.

Já no que diz respeito à resolução dos problemas, numa perspetiva de aluno, o processo de avaliação começa com a escolha do domínio de conhecimento (Figura 13). Após essa seleção, dá-se início ao *quiz,* onde o cenário se assemelha ao que consta na Figura 14. Destaca-se o temporizador para controlo do tempo de resposta, que influencia os dados alusivos à aptidão do aluno no domínio, e a presença de imagens, permitindo a ilustração das questões. Na Figura 15, mostra-se o carácter *responsive* da interface, que, por sua vez, contribui para o objetivo do suporte multiplataforma.

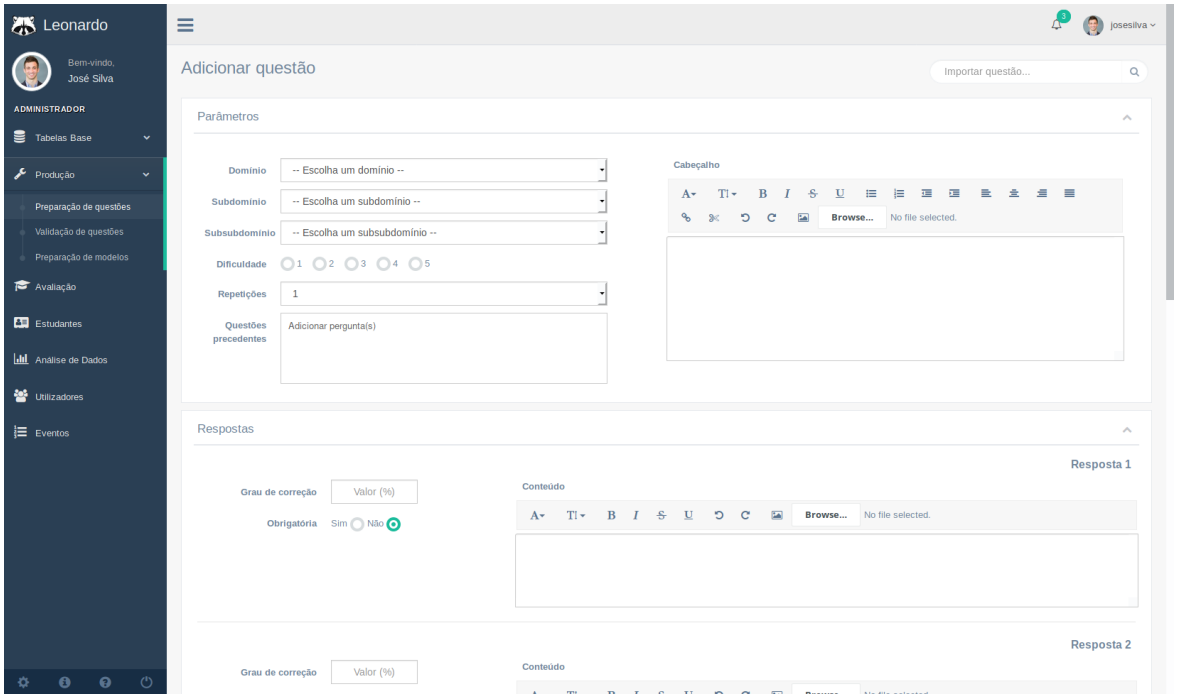


Figura 13: Interface do módulo de edição de questões do *Leonardo*.

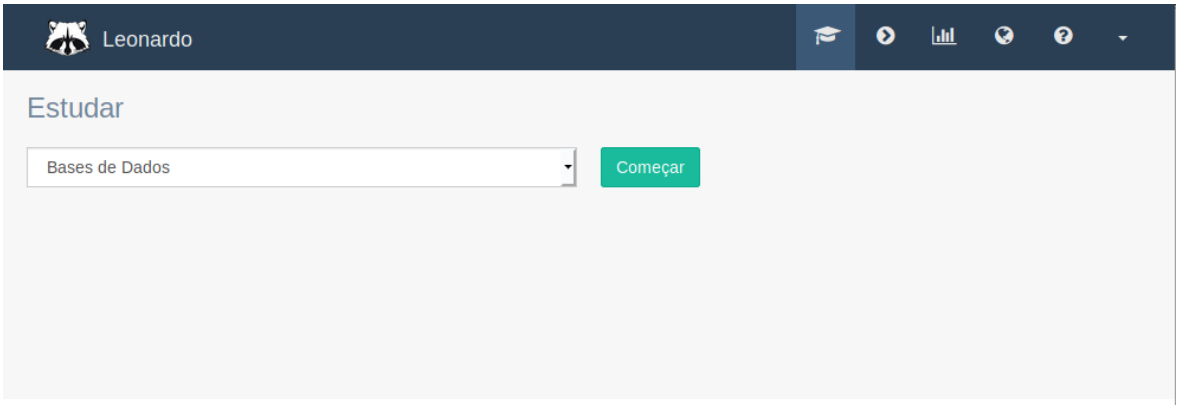


Figura 12: Escolha do domínio de conhecimento no início do processo de avaliação.

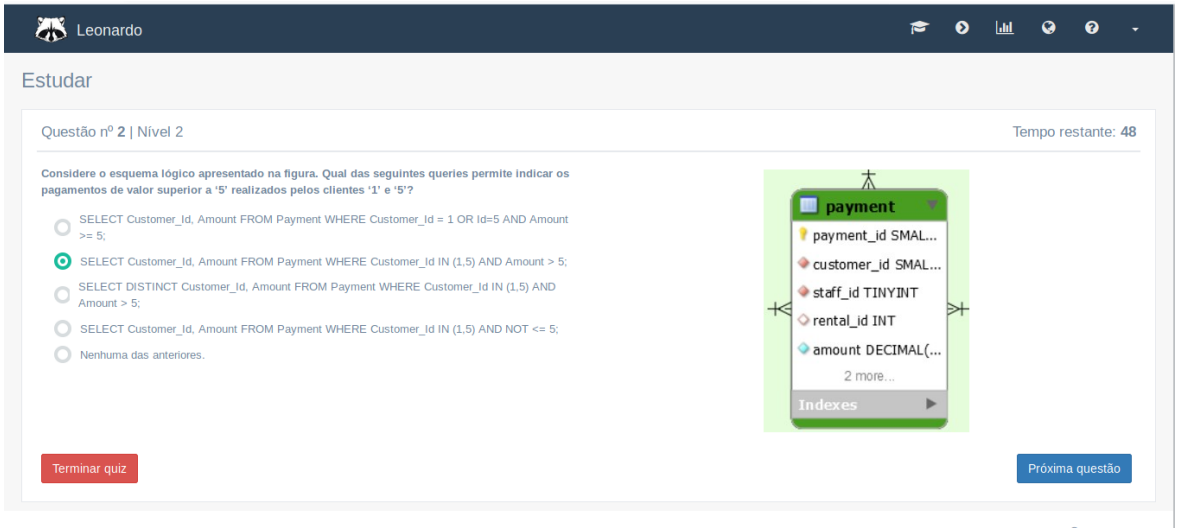


Figura 14: Exemplo de uma questão com imagem de um *quiz*.

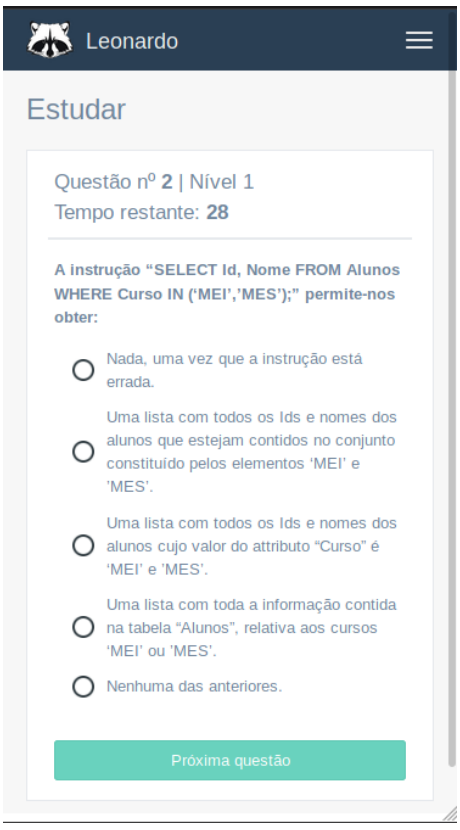


Figura 15: Modo *responsive* da interface da questão, na vista de um telemóvel.

Apresentada a interface, concentram-se agora as atenções sobre a intervenção da configuração do domínio no desenrolar da avaliação. Ora, recuperando o que foi mencionado acima acerca das transições de nível de conhecimento do aluno, perante a configuração apresentada podem ocorrer dois cenários.

O primeiro cenário está relacionado com o acumular de questões respondidas para o domínio em circunstância. Ao longo de vários *quizzes*, ou até num só, o estudante tem a hipótese de responder a 7 questões, valor mínimo que resulta da conjugação de três dos fatores configuráveis do domínio: `min_questions_number´, `questions_factor´ e o nível atual de conhecimento do aluno, que assume o valor do parâmetro `default_user_level´, quando o estudante inicia a sua avaliação num domínio. Obviamente, esta é uma quantidade reduzida de questões, pois o objetivo é conseguir-se, em poucas iterações, demonstrar neste documento os resultados do processo avaliativo. Porém, para que se perceba o potencial adaptativo desta configuração a diferentes critérios de avaliação, veja-se que o incremento do `questions_factor´ para 5, por exemplo, colocaria o patamar mínimo em 13 questões, o que resultaria também num crescimento mais acelerado do valor mínimo aquando da subida do nível do aluno (9 questões, com `questions_factor´ a 2, contra 18 questões, com o fator a 5, quando a classificação do conhecimento está em nível 3). Ainda, esta subida incremental pode ser anulada colocando o `questions_factor´ a 0, ficando a quantidade mínima de questões constante entre níveis de conhecimento e apenas dependente do `min_questions_factor´.

Com isto, a partir da sétima questão, inclusive, a cada nova resposta o sistema testa se deve alterar o seu registo do nível do aluno, com recurso aos dados do desempenho e aptidão do estudante, mantidos pelo módulo de *profiling*, e aos fatores da configuração do domínio alusivos aos mesmos dois parâmetros. Assim, o incremento está relacionado quer com o desempenho, quer com a aptidão do estudante, no domínio e nível de conhecimento atuais, quando estes superarem os valores calculados em função do `high_performance_factor´ e do `high_skill_factor´, respetivamente. Após responder à questão marginal do nível - a sétima, neste caso -, o estudante subirá de nível se, no(s) *quiz(zes)* que realizou, acertou em, pelo menos, 6 questões e não

necessitou, em média, de mais de 34% do tempo disponível para as respostas. Estes valores surgem de, aos parâmetros configuráveis referidos, o sistema acrescentar uma parcela proporcional ao nível do aluno atual, o que coloca o desempenho mínimo para subida em 78%, que se traduz nas ditas 6 questões corretas, e a aptidão mínima em 66%, valor inversamente proporcional ao tempo de resposta, calculando-se então os 34% do tempo máximo.

Naturalmente, a redução do nível segue o processo inverso e está dependente dos valores `low_performance_factor´ e `low_skill_factor´ da configuração. Daqui resulta que, perante o erro na resposta a 3 ou mais questões e um tempo médio de resposta superior a 64%, no total das 7 questões, o sistema atualiza a sua avaliação do conhecimento do aluno para o nível imediatamente abaixo - 1, neste caso. Nota para o facto de, caso não se cumpram as várias condições de subida ou descida de nível, o modelo do estudante do *Leonardo* opta por conservar a presente classificação - o nível 2.

Caso ocorra uma situação de *backlog* antes do estudante responder ao número mínimo de questões, a variação de nível é mais célere. É este o segundo cenário de evolução do estudante no processo de avaliação. Neste caso, com a configuração do domínio que se apresentou, a sequência para o segundo nível de conhecimento é de 8 respostas consecutivas, o que significa que o sistema incrementa/decrementa o nível do estudante à sétima pergunta certa/errada consecutiva e, por conseguinte, que o motor de inferência do módulo de avaliação ativa a regra de *backlog* na pergunta anterior. No capítulo da ativação de regras, é de referir que as regras relacionadas com a adição dos filtros de subdomínio e subsubdomínio só seriam primeiramente acionadas nos dois níveis de aprendizagem posteriores: 3 e 4, respetivamente.

Sintetizando, o nível de conhecimento de um estudante recém-chegado ao *software*, no domínio de Bases de Dados, parte da segunda classificação mais baixa e a sua transição está diante de 4

situações possíveis, com base na configuração assumida para a avaliação neste domínio em concreto:

1. o aluno acerta em 8 questões consecutivas, ocorrendo a subida do seu nível de conhecimento para 3;
2. o aluno erra em 8 questões consecutivas, ocorrendo a descida do seu nível de conhecimento para 1;
3. o aluno responde a, pelo menos, 7 questões, conseguindo um desempenho igual ou superior a 78% e um tempo de resposta igual ou inferior a 34%, na globalidade das questões respondidas, ocorrendo a subida do seu nível de conhecimento para 3;
4. o aluno responde a, pelo menos, 7 questões, conseguindo um desempenho igual ou inferior a 38% e um tempo de resposta igual ou superior a 64%, na globalidade das questões respondidas, ocorrendo a descida do seu nível de conhecimento para 1.

Para comprovar a análise efetuada anteriormente, realizou-se um *quiz* experimental nas condições enumeradas. Foi necessário, então, realizar as seguintes ações:

- introduzir o domínio abordado acima, precisamente com a configuração apresentada;
- criar um novo utilizador no sistema, com estatuto de *Estudante*, identificável pelo nome de utilizador "joaocoelho", para garantir que não existem registos anteriores do seu perfil;
- associar o novo domínio à lista de domínios elegíveis, tanto do aluno introduzido, como de um utilizador com estatuto de *Responsável*, sendo irrelevante qual, neste caso
- inserir um conjunto de questões do domínio, através do utilizador perito;
- simular a realização de um *quiz* de estudo no âmbito do novo domínio, por parte do utilizador estudante, interrompido após se responder a 7 questões.

Naturalmente, a interrupção após 7 questões deve-se a essa ser a quantidade mínima de questões que possibilita a mudança de nível do aluno, no domínio introduzido, tal como foi explicitado

anteriormente. As questões, e respetivas respostas, a este *quiz* encontram-se entre os anexos 1 e 7, apresentados por ordem de saída no *quiz*, como aliás se pode confirmar pela numeração na imagem da própria questão. Na sétima questão, ao invés de se clicar em "Próxima questão", deu-se por encerrada a sequência de questões, clicando em "Terminar quiz", ação que desencadeou o aparecimento da página de resultados que se mostra na Figura 16.

Figura 16: Resultados do *quiz* realizado, fornecidos pelo sistema após o seu final.

Nos resultados, destacam-se as métricas que permitem perceber se o nível de conhecimento do aluno no sistema poderá variar. Ora, atentando no gráfico mais à direta na imagem, pode-se concluir que o nível não pode ter variado por meio de uma sequência de *backlog*. Desde logo, não foram realizadas as 8 questões mínimas para um `backlog_factor´ de 4 e um nível de conhecimento classificado em 2. Ademais, como apenas 6 questões tiveram o mesmo grau de correção de forma consecutiva, respostas corretas nesta situação, nem houve lugar à ativação da regra específica do módulo de avaliação. Todavia, já que foram respondidas questões suficientes, avaliam-se os valores do desempenho e da aptidão. Recupera-se que, para poder ocorrer uma subida de nível, os valores mínimos para a taxa de acerto e tempo médio de resposta requerem 6 respostas corretas e uma percentagem de tempo não superior a 34% do total, respetivamente. Olhando para os valores da imagem, 6 foi precisamente o número de questões com resposta correta, o que equivale a uma taxa de acerto de 86%, e o tempo médio de resposta ficou em 25%.

Conclui-se, então, que o sistema deverá ter incrementado o nível de conhecimento do aluno, algo que pode ser comprovado na Figura 17, onde se realça o valor deste nível, registo obtido do módulo

de *profiling* após o *quiz*. De facto, o sistema, desconhecendo o aluno e o seu conhecimento no domínio avaliado, começou por assumir o nível 2 para o conhecimento do avaliando. Porém, após a primeira avaliação, concluiu que esta classificação não se adequa ao estado de aprendizagem do estudante, tendo enquadrado o seu conhecimento no nível 3.

Figura 17: Registo do módulo de *profiling* para nível de conhecimento do aluno, após o *quiz*.

Para terminar, referem-se ainda os restantes da Figura 16, que, além da exposição do período entre o qual foram lançadas as questões e do domínio que esteve a ser avaliado, indicam que nenhuma questão ficou por responder ("questões não respondidas") e que, no total das 7 questões, o estudante precisou de 107.80s para as suas respostas, isto é, o *quiz* durou cerca de 2 minutos ("Tempo gasto a responder").

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Síntese e Apreciação do Trabalho Realizado

No desenrolar do trabalho desta dissertação, procurou-se associar o conceito de ITS com o protótipo de sistema de apoio à avaliação de conhecimento em desenvolvimento na Universidade do Minho. As vantagens dos tutores artificiais e a sua estrutura típica são temáticas desenvolvidas, com ênfase para o papel preponderante que o ITS pode ter no apoio ao ensino, dentro e fora da sala de aula, e para a sua composição modular clássica, à imagem da qual foi arquitetado o *software* da iniciativa *Leonardo*.

Atualmente a informação está, a qualquer instante, à distância de um bolso onde conste um *smartphone* com acesso à Internet. Esta acessibilidade pode ser explorada na Educação, através dos sistemas de tutoria inteligentes. Qual professor que acompanha os alunos para todo o lado, um tutor artificial disponibilizado na Web 24h por dia consegue, potencialmente, suportar o processo de aprendizagem, permitindo aos estudantes testar os seus conhecimentos, durante ou após as aulas,

na plataforma que mais lhes convém. A avaliação gradual fica, então, facilitada e melhorada, até, face ao carácter adaptativo do sistema.

A análise do conhecimento do estudante é a questão central no contexto desta dissertação. O mecanismo de raciocínio responsável por gerir este processo foi alvo de estudo, tendo sido desenvolvidas regras de produção com base no que se consideram ser heurísticas de avaliação equilibradas, que procuram fomentar as capacidades de cada aluno. Para tal, muito contribuiu a integração com o módulo do sistema *Leonardo* responsável por traçar o perfil individual dos estudantes – o Profiler –, cuja prototipagem acompanhou em simultâneo o desenrolar desta investigação.

Destacam-se como aspetos positivos desta implementação teórico-prática o trabalho realizado sobre a estrutura da questão e a competência do motor de inferência ao variar os seus parâmetros de seleção de questões, em função do estado atual do *quiz,* acompanhando o nível de conhecimento do aluno, que lhe é comunicado pelo módulo de *profiling* do sistema, e aplicando as regras concebidas. Aliado a este aspeto, reconhece-se ainda a sua capacidade de lançar uma questão sem recorrer à aleatoriedade, quando não existe na base de conhecimento um registo que cobre todos os filtros da seleção adaptativa.

Por fim, em termos positivos, a integração conseguida com o módulo de *profiling* e a interface de utilizador atingiu um estado avançado, o que oferece garantias de futuro para o protótipo concebido. O desenvolvimento da interface W*eb* foi também muito importante, já que a disponibilização *online* do sistema se trata de um dos pontos relevantes do projeto. Contudo, foi um aspeto que consumiu uma parte muito significativa do período limitado de implementação do trabalho desta dissertação. Pode, portanto, ser considerado um aspeto menos positivo, uma vez que o tempo poderia ser investido no aperfeiçoamento do motor de inferência, através da introdução de novas estratégias de avaliação no *software*. Uma segunda limitação deste sistema, comparativamente a outros trabalhos, é a falta de provas e testemunhos reais, até à data. Para que a eficácia do mecanismo de avaliação

de conhecimento possa ser apreciada verdadeiramente, este deve ser testado por estudantes em ambiente real, algo que não ocorreu face à (ainda) precocidade atual da iniciativa *Leonardo.*

5.2 Trabalho Futuro

Para uma eventual reedição ou segunda etapa deste trabalho, existem alguns aspetos a ter em atenção futuramente. O estudo da aplicação de meta-regras tornar-se-ia progressivamente mais relevante à medida que se fossem adicionando regras de produção, intervenientes no processo de seleção das questões, no sistema. Em sistemas inteligentes baseados em regras, o meta-conhecimento – conhecimento acima de qualquer domínio de conhecimento – é representado sobre a forma de meta-regras. Portanto, uma meta-regra pode ser definida como uma estratégia para o uso de regras específicas de uma tarefa, num tutor inteligente. Um exemplo concreto poderia ser: "Regras criadas por peritos têm maior prioridade do que aquelas criadas por utilizadores menos qualificados".

Mais prioritário, no entanto, seria a introdução de técnicas de aprendizagem no sistema. Naturalmente, isto permitiria ao motor de inferência do tutor induzir as suas próprias regras de produção, ajustando-as ao longo da interação com o estudante. Seria, então, uma aprendizagem conjunta – o sistema aprenderia o comportamento do aluno e este o domínio de conhecimento – que poderia avançar para um cenário em que o próprio mecanismo de raciocínio definiria dinamicamente o tempo máximo de resposta à questão ou faria subir ou descer drasticamente o nível de dificuldade, testando o lado emocional do estudante. Com o modelo do comportamento emocional, o sistema fica potencialmente capaz de antecipar os momentos em que o aluno pode vacilar, o que constitui o verdadeiro teste às suas competências no domínio avaliado.

Outra melhoria do mecanismo passaria pela utilização de modelos de raciocínio *fuzzy*. A lógica *fuzzy*, ou multi-valor, assenta na ideia de que toda a informação admite um nível de verdade ou

pertença a uma certa classe. Em vez de se considerar apenas o verdadeiro e falso, ou seja, os dois valores booleanos, encoraja a que se aceite o parcialmente verdadeiro e falso ao mesmo tempo, o que se reflete na divisão da escala booleana em intervalos intermédios. Ora, dado que em ambiente de estudo informal e sem local ou plataforma específica não há garantia da total concentração do aluno, esta forma de raciocínio cobriria a consequente incerteza associada a alguns dados.

Concluindo, agora numa perspetiva de reformulação da implementação realizada, uma mudança vantajosa para o sistema consiste na sua modelação com agentes. Os agentes têm a capacidade de agir e tomar decisões de forma autónoma, com o intuito de alcançar os seus objetivos, sendo capazes de interagir com outros agentes, usando protocolos de interação inspirados no ser humano, tais como: coordenação, cooperação, competição e negociação. Face a estas características, os ITS têm ganhos significativos a nível da comunicação e interação que o sistema é capaz de fornecer.

Bibliografia

Anohina, A. (2007) 'Advances in intelligent tutoring systems: problem-solving modes and model of hints', *International Journal of Computers Communications & Control*, 2(1), pp. 48–55.

Baddeley, A. (1992) 'Working memory', *Science*. American Association for the Advancement of Science, 255(5044), pp. 556–559.

Baneres, D. *et al.* (2016) 'Towards an adaptive e-assessment system based on trustworthiness', in *formative assessment, learning data analytics and gamification*. Elsevier, pp. 25–47.

Barr, A. and Feigenbaum, E. A. (2014) *The handbook of artificial intelligence*. Butterworth-Heinemann.

Belo, O., Coelho, J. and Fernandes, L. (2019) 'An Evolutionary Software Tool For Evaluating Students On Undergraduate Courses', in *ICERI2019 Proceedings*.

Bobrow, D. G. (1964) *Natural language input for a computer problem solving system*. Massachusetts Institute of Technology.

Brusilovsky, P. and Miller, P. (2001) 'Course delivery systems for the virtual university', *Access to knowledge: New information technologies and the emergence of the virtual university*, pp. 167–206.

Brusilovsky, P. and others (1999) 'Adaptive and intelligent technologies for web-based eduction', *Ki.* Citeseer, 13(4), pp. 19–25.

Brusilovsky, P., Schwarz, E. and Weber, G. (1996) 'ELM-ART: An intelligent tutoring system on World Wide Web', in *International conference on intelligent tutoring systems*, pp. 261–269.

Carbonell, J. R. (1970a) '{AI} in {CAI}: An Artificial-Intelligence Approach to Computer-Assisted Instruction', *IEEE*.

Carbonell, J. R. (1970b) *Mixed-initiative man-computer instructional dialogues*. Massachusetts Institute of Technology.

Fowler, D. G. (1991) 'A model for designing intelligent tutoring systems', *Journal of medical systems*. Springer, 15(1), pp. 47–63.

Freedman, R., Ali, S. S. and McRoy, S. (2000) 'Links: what is an intelligent tutoring system?', *intelligence*. ACM, 11(3), pp. 15–16.

Kellogg, C. H. (1968) 'A natural language compiler for on-line data management', *Fall Joint Computer Conference*, pp. 473–492.

Klašnja-Milićević, A. *et al.* (2011) 'E-Learning personalization based on hybrid recommendation strategy and learning style identification', *Computers & Education*. Elsevier, 56(3), pp. 885–899.

Mitrovic, A. *et al.* (2009) 'ASPIRE: an authoring system and deployment environment for constraint-based tutors', *International Journal of Artificial Intelligence in Education*. IOS Press, 19(2), pp. 155–188.

Morrison, D. M. and Rus, V. (2013) 'The SCHOLAR legacy: A new look at the affordances of semantic networks for conversational agents in intelligent tutoring systems', in *CEUR Workshop Proceedings*, pp. 128–136.

Murray, T., Blessing, S. and Ainsworth, S. (2003) *Authoring tools for advanced technology learning environments: Toward cost-effective adaptive, interactive and intelligent educational software*. Springer Science & Business Media.

Rickel, J. W. (1989) 'Intelligent computer-aided instruction: A survey organized around system components', *IEEE Transactions on Systems, Man, and Cybernetics*. IEEE, 19(1), pp. 40–57.

Rutherford, A. (2012) 'BF Skinner: Scientist, celebrity, social visionary', *APS Observer*, 25(3).

Self, J. (1988) 'Student models: what use are they', *Artificial intelligence tools in education*. Elsevier Science Publishers BV (North-Holland), pp. 73–86.

Self, J. (1990) 'Theoretical foundations for intelligent tutoring systems', *Journal of Artificial Intelligence in Education*. Association for the Advancement of Computing in Education, 1(4), pp. 3–14.

Shute, V. J. and Psotka, J. (1994) *Intelligent Tutoring Systems: Past, Present, and Future.*

Simmons, R. F. (1970) 'Natural language question-answering systems: 1969', *Communications of the ACM*, 13, pp. 15–30.

Skinner, B. F. (1958) 'Teaching machines', *Science*. JSTOR, 128(3330), pp. 969–977.

Sleeman, D. and Brown, J. S. (1982) *Intelligent Tutoring Systems*. London : Academic Press. Available at: https://hal.archives-ouvertes.fr/hal-00702997.

Stankov, S. *et al.* (2008) 'TEx-Sys model for building intelligent tutoring systems', *Computers & Education*. Elsevier, 51(3), pp. 1017–1036.

Weber, G. (1999) 'Adaptive learning systems in the World Wide Web', in *UM99 User Modeling*. Springer, pp. 371–377.

Weber, G. and Brusilovsky, P. (2001) 'ELM-ART: An adaptive versatile system for Web-based instruction'.

Weber, G. and Specht, M. (1997) 'User modeling and adaptive navigation support in WWW-based tutoring systems', in *User Modeling*, pp. 289–300.

Wenger, E. (1987) 'Artificial intelligence and tutoring system: Computational and Cognitive Approaches to the Communication of Knowledge', *Califórnia: Morgan Kaufmann Publishers. Texto publicado na: Pátio-revista pedagógica Editora Artes Médicas Sul Ano*, 1, pp. 19–21.

Zaíane, O. R. (2002) 'Building a recommender agent for e-learning systems', in *International Conference on Computers in Education, 2002. Proceedings.*, pp. 55–59.

(...).

Referências WWW

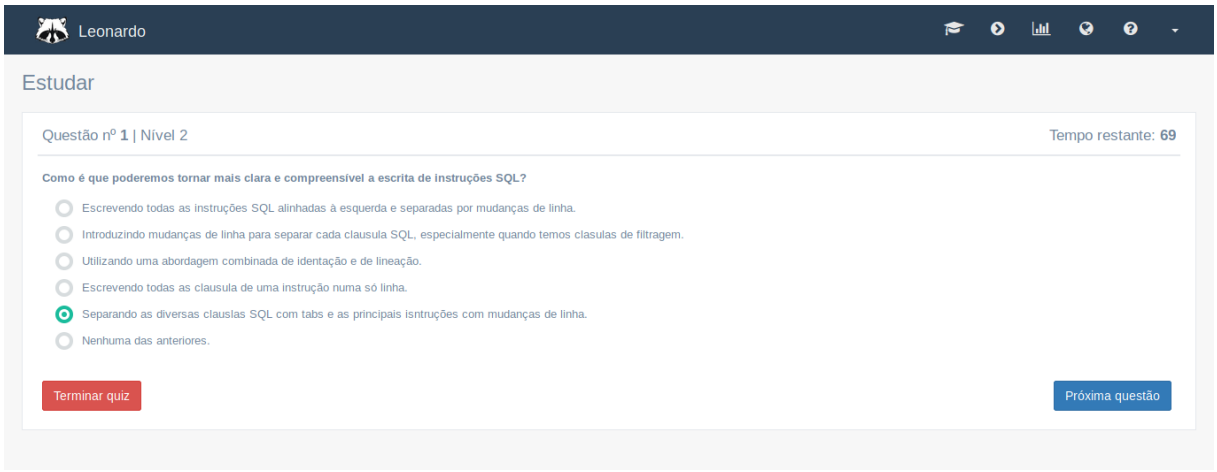
MongoDB. (2019). *The most popular database for modern apps*. [online] Available at: https://mongodb.com [Accessed 20 Oct. 2019].

Python.org. (2019). *Welcome to Python.org*. [online] Available at: https://www.python.org/ [Accessed 20 Oct. 2019].

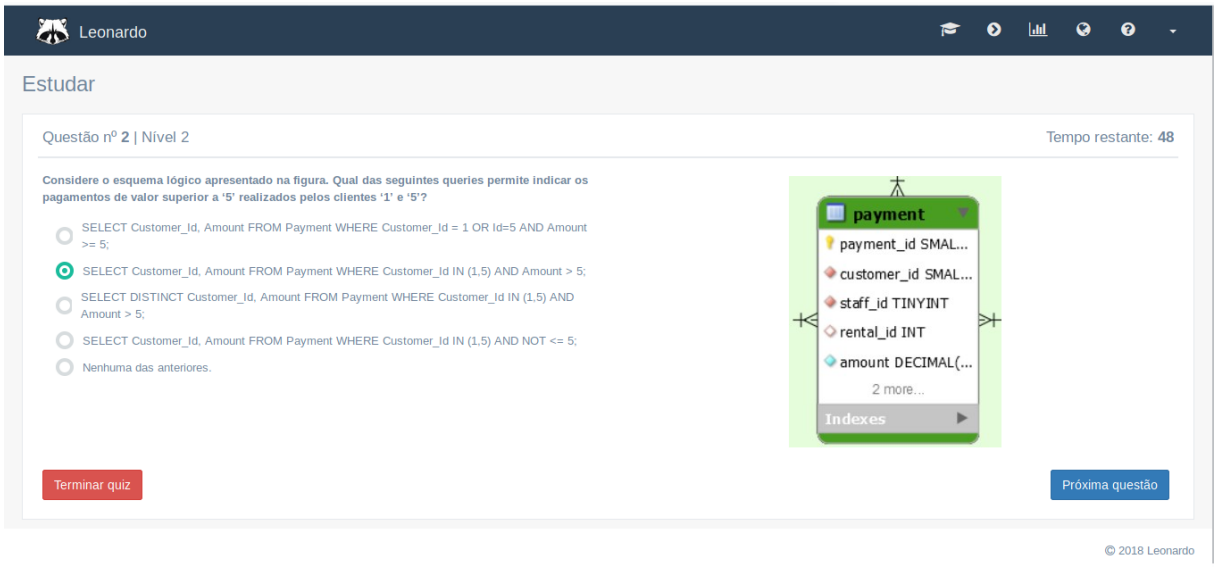
Pallets. (2019). *Flask*. [online] Available at: https://palletsprojects.com/p/flask/ [Accessed 20 Oct. 2019].

Anexos

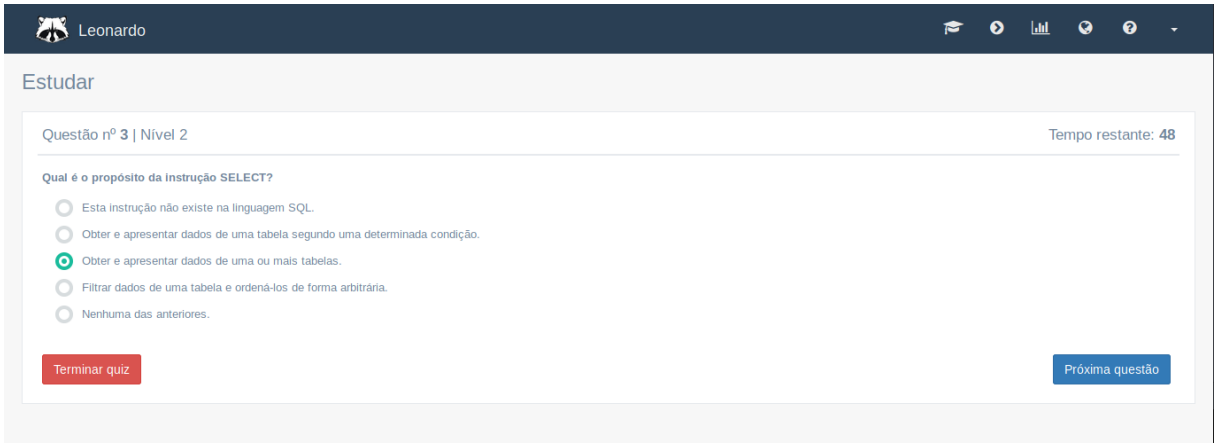
A.1 Primeira questão do quiz experimental



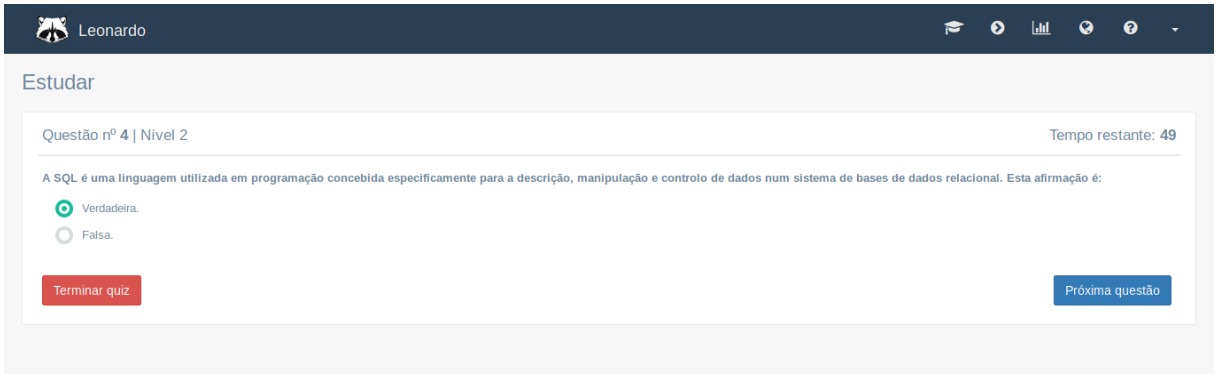
A.2 Segunda questão do *quiz* experimental



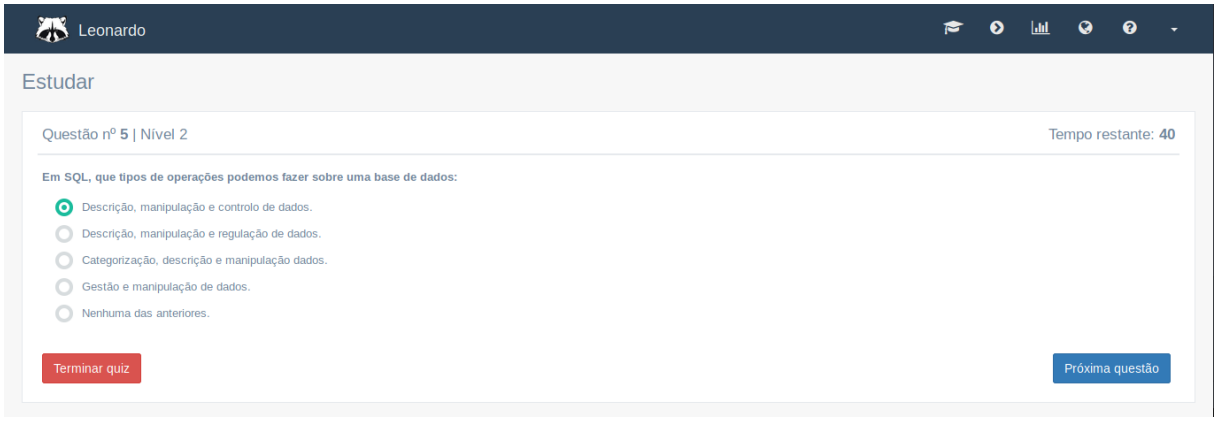
A.3 Terceira questão do *quiz* experimental



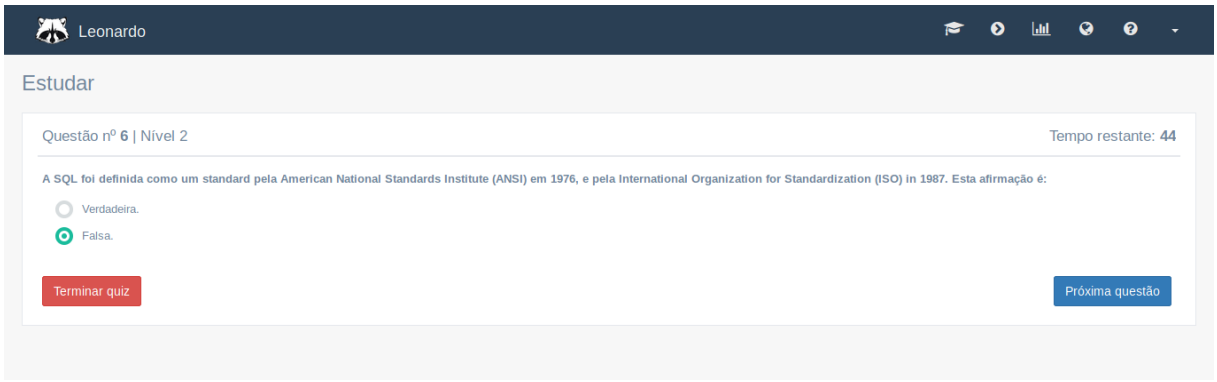
A.4 Quarta questão do *quiz* experimental



A.5 Quinta questão do *quiz* experimental



A.6 Sexta questão do *quiz* experimental



A.7 Sétima questão do *quiz* experimental