Sistema Tutor Inteligente baseado em Aprendizado de Máquina para Ensino-aprendizagem de Manutenção de Software

Rodrigo Elias Francisco

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2023

Rodrigo Elias Francisco

Sistema Tutor Inteligente baseado em Aprendizado de Máquina para Ensino-aprendizagem de Manutenção de Software

Tese de doutorado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Flávio de Oliveira Silva, Ph.D.

Uberlândia
2023

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

F819s
2023

Francisco, Rodrigo Elias, 1986-
Sistema tutor inteligente baseado em aprendizado de máquina para ensino-aprendizagem de manutenção de software [recurso eletrônico] / Rodrigo Elias Francisco. - 2023.

Orientador: Flávio de Oliveira Silva.
Tese (Doutorado) - Universidade Federal de Uberlândia, Programa de Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.te.2024.5007
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Silva, Flávio de Oliveira, 1970-, (Orient.). II. Universidade Federal de Uberlândia. Programa de Programa de Pós-graduação em Ciência da Computação. III. Título.

CDU: 681.3

André Carlos Francisco
Bibliotecário Documentalista - CRB-6/3408

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Tese, 29/2023, PPGCO				
Data:	21 de dezembro de 2023	Hora de início:	15:01	Hora de encerramento:	19:38
Matrícula do Discente:	11723CCP005				
Nome do Discente:	Rodrigo Elias Francisco				
Título do Trabalho:	Sistema Tutor Inteligente baseado em Aprendizado de Máquina para Ensino-aprendizagem de Manutenção de Software				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de Computação				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Pedro Frosi Rosa - FACOM/UFU, João Henrique de Souza Pereira - FACOM/UFU, Rodrigo Silva Duran - IFMS, Ronaldo Celso Messias Correia - FCT/UNESP e Flávio de Oliveira Silva - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Rodrigo Silva Duran - Nova Andradina/MS, Ronaldo Celso Messias Correia - Presidente Prudente/SP e Flávio de Oliveira Silva - Braga, Portugal. Os outros membros da banca participaram da cidade de Uberlândia e o aluno da cidade Morrinhos/GO.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Flávio de Oliveira Silva, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Doutor.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação

interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Rodrigo Silva Duran**, **Usuário Externo**, em 02/01/2024, às 15:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Ronaldo Celso Messias Correia**, **Usuário Externo**, em 03/01/2024, às 10:32, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **João Henrique de Souza Pereira**, **Professor(a) do Magistério Superior**, em 03/01/2024, às 13:42, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Flávio de Oliveira Silva**, **Professor(a) do Magistério Superior**, em 05/01/2024, às 12:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Pedro Frosi Rosa**, **Professor(a) do Magistério Superior**, em 07/01/2024, às 14:57, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5074833** e o código CRC **8E95A677**.

Referência: Processo nº 23117.089258/2023-30 SEI nº 5074833

Dedico esta Tese aos meus pais, Nelza e Luiz.

Agradecimentos

Agradeço a Deus por ter concedido minha vida com saúde e capacidade de lidar com as dificuldades.

Agradeço a toda minha família. Mãe e Pai, obrigado por nos ter criado com fé, carinho e honestidade e por nos ter proporcionado o acesso à educação. Laiz, minha irmã, obrigado por sermos sempre unidos e por ter revisado o texto desta tese.

Agradeço a minha Avó Benedita (em memória), pela força e pelas orações. Aos meus Avôs Durval e Valdim e à minha Avó Cleonice. Aos meus tios João e Valclênio, pelo contato que tivemos com computadores na minha infância. Às minhas tias professoras (Dicléia, Maria José e Adriana), por terem acompanhado toda minha vida escolar. Às minhas tias Maria Antônia, Elza, Hilda e Raquel e meus primos, por serem sempre presentes.

Agradeço a todos os colegas e professores da Facom-UFU. Em especial ao meu orientador Flávio, que me trouxe uma grande motivação e apoio em uma fase difícil através de um novo olhar para a pesquisa. Trata-se de um olhar mais crítico, contextualizado, pragmático e inovador. Agradeço também por tudo que aprendi sobre a concepção de sistemas e escrita científica. Agradeço ao professor Pedro Frosi Rosa por sua forma dinâmica e ativa de conduzir o estudo quando fui seu aluno. Ao meu amigo Eduardo Castilho, por nossas conversas sobre estratégias para termos êxito nos estudos.

Agradeço a todos os colegas docentes do IF Goiano - Morrinhos. Em especial aos professores da Computação (Norton, Felipe, Ana Maria, Pereira, Odilon, Hiury, Reidner, Alexandre, Marcel, Antônio Neco, Fernando, Jesmmer e Leila), que me ajudaram a conciliar meu trabalho com o doutorado.

Agradeço ao IF Goiano pelo afastamento e licença-capacitação que me foram concedidos e contribuíram para a construção desta Tese.

Agradeço à professora Ana Paula Ambrósio (em memória), da UFG, que me proporcionou experiências de pesquisa no mestrado e me motivou para com o doutorado.

Agradeço a todas as pessoas que eu convivi durante o doutorado e me ajudaram com suas experiências.

"Acredite, pense e faça,
use sua intuição,
transforme sonho em suor,
pensamento em ação.
Enfrente cada batalha
sabendo que a gente falha
e que isso é natural,
cair pra se levantar,
aprender pra ensinar
que o bem é maior que o mal"
(Bráulio Bessa / Poesia que transforma)

Resumo

Há uma demanda de mercado sobre profissionais capacitados para trabalhar com *Software Maintenance* (SM). A formação desses profissionais é bastante complexa, pois é necessário que eles estejam aptos a realizar certas atividades, como por exemplo, compreender o código-fonte e diagramas e manipular técnicas e ferramentas visando SM. O professor de SM se depara com dificuldades de oferecer um suporte adequado a turmas em um tempo viável, o que torna *Intelligent Tutoring System* (ITS) para SM uma categoria de sistemas bastante promissora. A literatura aponta desafios sobre o tema e indicam baixo uso de *Machine Learning* (ML) em ITS para SM. Esta tese propõe uma arquitetura para de ITS com foco em SM e aborda o uso de ML nos Módulos Tutor e do Estudante e a integração entre eles, contribuindo com os desafios de pesquisa. O Módulo do Estudante trabalha com a identificação de tipos de estudantes de SM utilizando Clusterização. O Módulo Tutor trabalha com a recomendação de *Didactic Materials* (DM) de SM utilizando *Reinforcement Learning* (RL) através do algoritmo *Q-Learning*. A tese também apresenta a modelagem do *Expert Knowledge Module* (EKM) a partir do conteúdo de SM, que contribuiu para a avaliação dos Módulos Tutor e do Estudante a partir da dimensão de conhecimento de SM. Os resultados indicam que o algoritmo *K-Means* é adequado para o Módulo do Estudante e que a sua integração com o Módulo Tutor, em certas determinadas condições, traz altos ganhos na eficiência da recomendação de DM. As avaliações foram realizadas a partir de um conjunto de dados de capacidades de SM de estudantes baseado em um conjunto de dados real de desempenho de estudantes e simulação computacional e mostraram que a proposta de ITS trouxe resultados significativos quanto a eficiência da recomendação de DM de SM.

Palavras-chave: Sistema Tutor Inteligente. Manutenção de Software. Aprendizado de Máquina.

Abstract

There is a market demand for qualified professionals to work with SM. The training of these professionals is quite complex, as they need to be able to carry out certain activities, such as understanding the source code and diagrams and manipulating techniques and tools aimed at SM. The SM teacher faces difficulties in offering adequate support to classes at a viable time, which makes ITS for SM an up-and-coming category of systems. The literature points out challenges on the topic and indicates low use of ML in ITS for SM. This thesis proposes an architecture for ITS focusing on SM. It addresses the use of ML in the Tutor and Student Modules and their integration, contributing to research challenges. The Student Module works with identifying types of SM students using Clustering. The Tutor Module works with the DM recommendation of SM using RL through the *Q-Learning* algorithm. The thesis also presents the modeling of EKM based on SM content, which contributed to evaluating the Tutor and Student Modules based on the SM knowledge dimension. The results indicate that the *K-Means* algorithm is suitable for the Student Module and that its integration with the Tutor Module, under certain conditions, brings high gains in the efficiency of DM recommendation. The evaluations were conducted using a data set of student SM capabilities based on a real data set of student performance and computer simulation. They showed that the ITS proposal brought significant results regarding the efficiency of DM recommendations for SM.

Keywords: Intelligent Tutoring System. Software Maintenance. Machine Learning.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ACM *Association for Computing Machinery*

AI *Artificial intelligence*

BIRCH *Balanced Iterative Reducing and Clustering Using Hierarchies*

BKT *Bayesian Knowledge Tracing*

CBR *Case-Based Reasoning*

CPU *Central Process Unit*

CG *Computing Grade*

DM *Didactic Materials*

EKM *Expert Knowledge Module*

ENADE *Exame Nacional de Desempenho dos Estudantes*

EWPI *Experiment Weighted Performance Indicator*

GPT-3 *Generative Pre-Training Transformer 3*

GG *Generic Grade*

HTML *Hypertext Markup Language*

HIT *Number of Hits*

IT *Information Technology*

ITS *Intelligent Tutoring System*

ITSB *Intelligent Tutoring System Builder*

JDI *Java Debugger Interface*

LMS *Learning Management System*

ML *Machine Learning*

RQ *Research Question*

RL *Reinforcement Learning*

REC *Number of Recommendations*

RAM *Random Access Memory*

SM *Software Maintenance*

SBC *Brazilian Computer Society*

UML *Unified Modeling Language*

UC *Use Case*

UI *User Interface*

Sumário

Capítulo 1

Introdução

Software Maintenance (SM) é uma área de atuação profissional cuja importância se relaciona com dinâmica de mudanças nos ambientes organizacional e tecnológico relacionados ao *software*. Organizações modificam sua maneira de operar por razões diversas, como para atender a mudanças nas legislações ou mesmo para melhorar sua produtividade, e isso precisa refletir nos seus sistemas de aplicação.

O documento *Guide to the Software Engineering Body of Knowledge* (SWEBOOK 3.0) (BOURQUE; FAIRLEY, 2014) define que a SM possui atividades realizadas nas etapas de pré-entrega e pós-entrega do *software*, que tem como objetivo oferecer suporte econômico ao *software*. As atividades pré-entrega enfatizam o planejamento para operações pós-entrega, manutenibilidade e determinação logística para a transição, enquanto as atividades pós-entrega envolvem a modificação do *software* propriamente dito, treinamento, operação e, se necessário, suporte técnico.

A Educação em Computação precisa dar suporte os estudantes a se tornarem profissionais capazes de trabalhar com SM (HECKMAN; STOLEE; PARNIN, 2018). Essa necessidade se justifica pelo cenário profissional de SM, cujas porções consumidas de tempo e esforço são altas (FERNÁNDEZ-SÁEZ; CHAUDRON; GENERO, 2018). O ensino-aprendizagem de SM precisa proporcionar ao estudante o desenvolvimento da capacidade de realizar a SM em projetos de *software* existentes considerando os níveis de dificuldade adequados. Para isso, o estudante precisa lidar com a prática de SM.

Realizar SM implica na necessidade de lidar com uma variedade de atividades. A dinâmica do ensino-aprendizagem de SM em uma perspectiva que inclua a prática de SM necessita de que o estudante realize atividades como: decidir se usará modelos para compreensão do *software* (FERNÁNDEZ-SÁEZ; CHAUDRON; GENERO, 2018); caso resolva usar modelos, como *Unified Modeling Language* (UML), encontrar as informações importantes (FERNÁNDEZ-SÁEZ; CHAUDRON; GENERO, 2018), como regras de negócio e justificativas de decisões; lidar com documentação inadequada, nomes de variáveis sem significados e falta de modularidade no design (ALLISON; JOO, 2015); aprender habilidades diferentes das quais foram educados em desenvolvimento de *software* (ALLI-

SON; JOO, 2015); direcionar sua atenção ao processo de SM (AHRENS; SCHNEIDER; BUSCH, 2019); e lidar com suas dúvidas sobre o processo de SM (WOOD et al., 2018). Essa variedade de tarefas de SM, associada à complexidade da SM, dificultam o trabalho do instrutor de SM, que pode ser o professor na educação formal na área de Computação ou outro profissional na educação corporativa em organizações de *Information Technology* (IT).

O contexto de SM em organizações de IT se beneficia de metodologias que envolvem aspectos como o a gestão do conhecimento em SM e o o conhecimento da equipe de SM (L'ERARIO; THOMAZINHO; FABRI, 2020). Prover esse conhecimento necessário envolve treinamento e pode estar contextualizado nos produtos de *software* aos quais a organização trabalha.

Como o ensino-aprendizagem de SM necessita de que o estudante resolva uma grande quantidade de tarefas de SM e estas tarefas possuem variados níveis de complexidade (FERNÁNDEZ-SÁEZ; CHAUDRON; GENERO, 2018; ALLISON; JOO, 2015; WOOD et al., 2018; AHRENS; SCHNEIDER; BUSCH, 2019). Isto dificulta para professores e instrutores de SM conseguirem oferecer um suporte adequado aos estudantes no ensino-aprendizagem de SM considerando o momento certo e as capacidades e dificuldades que os estudantes possuem em relação à SM neste dado instante temporal.

Diante disso, investir na construção de *Intelligent Tutoring System* (ITS) para o ensino-aprendizagem de SM torna-se bastante promissor. Os ITS podem trabalhar com tarefas no âmbito do ensino-aprendizagem em SM, oferecendo automação e adaptação durante o uso nos contextos educacional e organizacional em SM.

ITS são sistemas capazes de melhorar, adaptar e automatizar elementos do ensino (ALKHATLAN; KALITA,). Essa categoria de *software* oferece ao estudante a interação com o conteúdo de maneira adaptada ao seu perfil, buscando melhorar a experiência de aprendizagem. Eles são criados a partir de uma arquitetura padrão, explicada na seção 2.1.2, e de técnicas de *Artificial intelligence* (AI) e *Machine Learning* (ML) (ALSHAIKH; HEWAHI, 2021). Um ITS pode, por exemplo, recomendar conteúdos e apoiar o estudante na resolução de tarefas.

Esta Tese se insere na linha de pesquisa Tecnologias para o Ensino de Computação, descrita por (JR et al., 2020), e tem como objetivo melhorar o ensino-aprendizagem de SM. Ela contribui com o desenvolvimento de um ITS para apoiar o ensino-aprendizagem de SM.

1.1 Motivação

É importante refletir sobre os altos custos da SM e suas implicações na Educação em Computação. Na indústria de Computação, a SM pode ser responsável por mais de 60% de todos os custos (RUSSELL; VINSEL, 2017) e consequentemente gerar um alto

consumo de tempo e esforço (FERNÁNDEZ-SÁEZ; CHAUDRON; GENERO, 2018) dos profissionais. No entanto, diversos setores tendem a enfatizar a inovação e deixar para segundo plano a manutenção (RUSSELL; VINSEL, 2017), e existe um estigma de que SM é uma tarefa menos favorecida no ciclo de vida do *software* (ALLISON; JOO, 2015). Por isso, a comunidade de Educação em Computação precisa trazer a reflexão sobre a importância da SM e melhorar o ensino-aprendizagem nesta mesma área.

Para lidar com pressão sobre os programadores em realizar SM em código-fonte legado de maneira econômica, (OBERHAUSER, 2016) indicou a necessidade de apoiar esses programadores com Tutores e Recomendadores de trechos de código-fonte para compreensão de *software*. Essa indicação mostra a importância da pesquisa contribuir com a construção de ITS para SM.

O ensino-aprendizagem de SM em uma perspectiva que inclui a prática de SM envolve a necessidade do estudante desenvolver habilidades de SM. Essas habilidades são relacionadas às tarefas em SM e possuem as características variedade, quantidade e complexidade (FERNÁNDEZ-SÁEZ; CHAUDRON; GENERO, 2018; ALLISON; JOO, 2015; WOOD et al., 2018; AHRENS; SCHNEIDER; BUSCH, 2019). Por isso, o trabalho do professor, que precisa oferecer suporte adequado a turmas de SM torna-se difícil.

A construção de um ITS envolve projetar e treinar modelos de ML e isso envolve muito custo humano relacionado à geração de dados para treinar e avaliar modelos de ML (ALSHAIKH; HEWAHI, 2021). Para o contexto de SM, é necessário que os estudantes resolvam problemas de SM para gerar esses dados. Isso faz com que a investigação científica procure outras maneiras para acelerar a pesquisa sobre construção de ITS para SM.

1.2 Hipótese e Questões de Pesquisa

Diante da problemática discutida, esta Tese considera as seguintes hipóteses:

- ❑ *Hipótese Nula*: Técnicas de *Machine Learning* não contribuem para projetar um *Intelligent Tutoring System* capaz de recomendar *Didactic Material* de *Software Maintenance* para estudantes

- ❑ *Hipótese Alternativa*: Técnicas de *Machine Learning* contribuem para projetar um *Intelligent Tutoring System* capaz de recomendar *Didactic Material* de *Software Maintenance* para estudantes

Diante da Hipótese apresentada, percebeu-se a possibilidade de utilizar ML nos Módulos do ITS para SM. Este uso precisa estar de acordo com a definição conceitual da arquitetura geral de um ITS (ALKHATLAN; KALITA,), de modo que o conteúdo de SM seja representado no *Expert Knowledge Module* (EKM), o Módulo Tutor realize ações

pedagógicas para ajudar o estudante a avançar no estudo de SM e o Módulo do Estudante gerencie a representação dos estudantes para que esta seja usada pelo Módulo Tutor no seu trabalho. O Módulo Tutor precisa de um mecanismo que tome decisões sobre como apoiar o estudante, e.g. recomendando DM de SM e o Módulo do Estudante precisa de um mecanismo que sintetize dados dos estudantes para fornecer a representação do estudante. Isso trouxe essa Hipótese da possibilidade de apoio da ML no ITS para SM e possibilitou a definição das *Research Question* (RQ) abaixo para apoiar sua avaliação.

- ❏ **RQ 1:** Como projetar a arquitetura de um ITS para atender o ensino-aprendizagem de SM?
- ❏ **RQ 2:** Como ML pode contribuir com a identificação de tipos de estudantes de ML no Módulo do Estudante?
- ❏ **RQ 3:** Como ML pode ser utilizado no Módulo Tutor para recomendar DM de SM?
- ❏ **RQ 4:** Como ML pode ser utilizado no Módulo Tutor para recomendar DM de SM em um contexto que considere as variações iniciais no conhecimento de SM dos estudantes?
- ❏ **RQ 5:** A identificação dos tipos de estudantes de SM contribui para a recomendação de DM de SM?

1.3 Objetivos

Apesar de que ML e ITS podem contribuir com uma educação de alta qualidade (ALSHAIKH; HEWAHI, 2021), a literatura apresenta dificuldades na construção de ITS para SM.

Os ITS para SM concebidos até o momento não abordaram o tema de SM de uma maneira ampla, conforme a Tabela 4. Além disso, não há um consenso sobre como projetar os Módulos Tutor e do Estudante considerando a modelagem e o uso de algoritmos de ML nas suas funcionalidades internas, e sobre como projetar a integração entre esses módulos.

Para contribuir com esses problemas, o objetivo geral deste trabalho é conceber e projetar um ITS para o ensino-aprendizagem de SM utilizando técnicas de ML. Para atingir o objetivo geral, os seguintes objetivos específicos foram propostos:

1. Propor a Arquitetura do ITS para SM.
2. Propor um EKM para ITS para SM baseado no conteúdo de SM.
3. Propor e construir um Módulo Tutor capaz de recomendar DM de SM com ML.
4. Propor e construir um Módulo do Estudante para o ITS para SM com ML que contribua com a recomendação de DM (Módulo Tutor).
5. Propor e construir um Método baseado em Simulação para avaliar a recomendação de DM de SM.

6. Avaliar os Módulos do Estudante e Tutor do ITS para SM.

1.4 Contribuições

Esta Tese, através dos experimentos realizadas na busca de responder às RQ e da reflexão acerca dos conceitos e da literatura sobre ITS para SM, trouxe as seguintes contribuições.

1. Um método para a identificação de tipos de estudantes de SM baseado em Clusterização e na dimensão de conhecimento de SM para o Módulo do Estudante de ITS para SM.

2. Um método para recomendação de DM para um ITS para SM (Módulo Tutor) baseado em *Reinforcement Learning* (RL) e integrado à identificação de tipos de estudantes de SM realizada pelo Módulo do Estudante.

3. A melhoria na eficiência da recomendação de DM em ITS para SM através da integração com o método de identificação de tipos de estudantes de SM.

4. Um modelo para representar DM de SM para um EKM de ITS construído a partir de um amplo estudo do currículo de SM.

5. Um método para avaliação da recomendação de DM baseado dados referentes à dimensão do conhecimento de estudantes de SM e simulação.

6. Uma arquitetura de ITS para SM proposta a partir de um estudo fundamentado no conteúdo de SM, na dimensão do conhecimento de SM e em experimentos com técnicas de ML para as funcionalidades do ITS.

1.5 Metodologia

Esta pesquisa foi realizada a partir de uma metodologia experimental, que conforme (WAZLAWICK, 2009) ocorre a intervenção do pesquisador no ambiente na busca de resultados satisfatórios. Essa intervenção ocorreu com a implementação de modelos e métodos para o ITS para SM, enfatizando os Módulos do Estudante e Tutor, e a avaliação da eficiência desses métodos.

As avaliações realizadas utilizaram um *Dataset* de estudantes de SM construído a partir de dados reais (INEP, 2022) e simulação. Os experimentos realizados e os estudos envolvendo conceitos e literatura sobre ITS para SM contribuíram para a proposta de uma arquitetura para ITS para SM.

As seguintes etapas foram percorridas nesta tese:

1. Definir o problema e revisar a literatura sobre o tema.
2. Projetar e refinar sucessivamente a Arquitetura do ITS para SM.
3. Projetar um EKM para o domínio de SM para o ITS.
4. Construir um Módulo Tutor para o ITS capaz de recomendar DM de SM.

5. Construir um Método para avaliar o Módulo Tutor do ITS e avaliá-lo.

6. Construir um Módulo do Estudante para o ITS capaz de identificar os tipos de estudantes de SM.

7. Avaliar o Módulo do Estudante do ITS para SM.

8. Avaliar a contribuição da identificação de tipos de estudantes de SM (Módulo do Estudante) para a recomendação de DM de SM (Módulo Tutor).

9. Comparar os resultados obtidos com a literatura

A Figura 1 apresenta o detalhamento das etapas de desenvolvimento desta tese e seus respectivos resultados. Todos esses elementos são apresentados detalhadamente no texto desta tese. O levantamento do Estado da Arte de ITS para SM envolveu o levantamento dos trabalhos e a síntese do estado da arte, o que inclui documentar os desafios de pesquisa, e tem como saída a revisão da literatura para o tema. A proposta da arquitetura de ITS para SM envolveu o levantamento de funcionalidades, o desenho da arquitetura e a modelagem do EKM para o conteúdo de SM, e tem como saída o projeto de *software* para o ITS para SM. A implementação do Módulo Tutor envolveu a modelagem do algoritmo de *Reinforcement Learning* (RL) *Q-Learning* considerando a arquitetura proposta, a implementação do *Q-Learning* para recomendação de DM e a avaliação dele com somente simulação e simulação com o uso de um *Dataset* que representa as capacidades iniciais de SM dos estudantes, e gerou como saída os resultados da avaliação do Módulo Tutor. A implementação do Módulo do Estudante envolveu a utilização de um *Dataset* com as capacidades de SM dos estudantes, a implementação da Clusterização para este *Dataset*, a avaliação do mesmo, e gerou como saída a avaliação do Módulo do Estudante. A integração entre os Módulos Tutor e do Estudante envolveu a implementação da proposta de integração, a avaliação da integração usando simulação e o *Dataset* de estudantes de SM e a comparação dos resultados com o uso e sem o uso da integração e gerou como saída os resultados da avaliação da integração. A identificação das contribuições comparou os resultados obtidos com os desafios presentes na literatura e possibilitou levantar as contribuições desta tese.

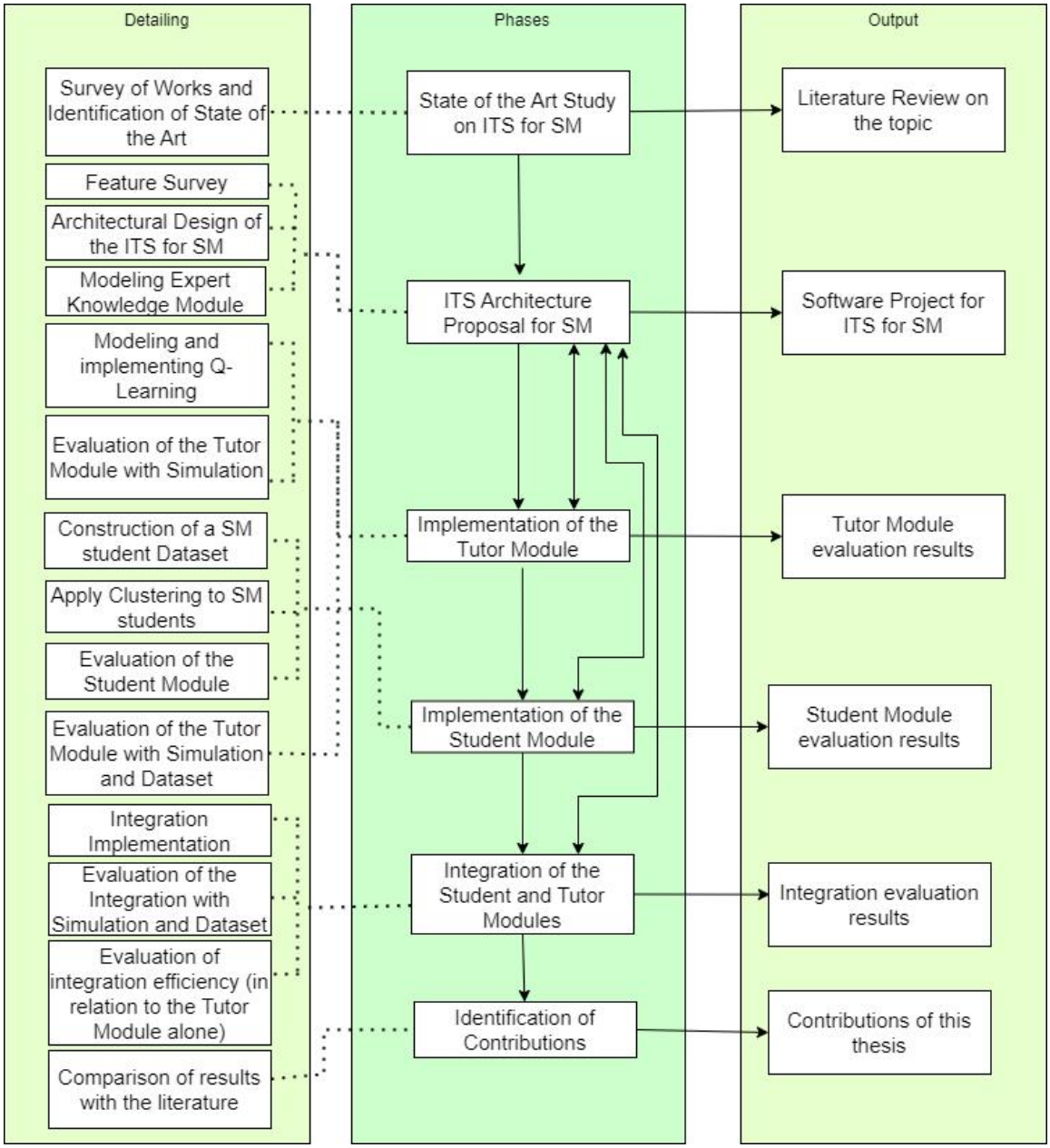

Figura 1 – Visão sobre as etapas do desenvolvimento da Tese

1.6 Organização da Tese

Esta possui uma sequência de capítulos que reflete às atividades realizadas para contribuir com o problema da construção de um ITS para SM em uma abordagem que propõe uma Arquitetura para o ITS e a avaliação de ML em módulos do mesmo. Essa sequência é descrita a seguir.

- ❑ O Capítulo 1 trata-se da introdução e apresenta o contexto do problema sobre a construção de ITS para SM, a motivação, os objetivos e desafios de pesquisa, as RQ

e hipóteses, as contribuições e a metodologia. O objetivo deste capítulo é apresentar o contexto do problema e a estrutura da estratégia de resolução adotada;

- O Capítulo 2 apresenta os Conceitos Fundamentais e Trabalhos Correlatos relacionados à construção de ITS para SM. Os Conceitos Fundamentais envolvem o ensino-aprendizagem de SM, ITS, a taxonomia de Bloom revisada, AI e ML. Os Trabalhos Correlatos estão organizados em um conjunto mais amplo sobre ITS para Educação em Ciência da Computação e ITS para SM;

- O Capítulo 3 apresenta a proposta de ITS para SM. A descrição da Arquitetura do ITS, da modelagem referente à estrutura de seus módulos e de como eles se relacionam com o conteúdo de SM são apresentadas. O Módulo EKM operacionaliza o conteúdo de SM. O Módulo do Estudante foi proposto a partir de Clusterização e contribui diretamente para o funcionamento do Módulo Tutor. O Módulo Tutor foi proposto com RL e trabalha com a recomendação de *Didactic Materials* (DM) de SM;

- O Capítulo 4 apresenta os Experimentos e Discussão. O Módulo do Estudante é avaliado utilizando um *Dataset* de capacidades de SM de estudantes e com condições identificadas para a escolha do algoritmo de Clusterização e parâmetros mais adequados. O Módulo Tutor e sua integração com o Módulo do Estudante são avaliados com uma estratégia que utiliza simulação da resolução de DM de SM por estudantes e *Dataset* de capacidades de SM de estudantes. Este capítulo destaca as contribuições da pesquisa realizada nesta Tese. Este capítulo também apresenta os limites da pesquisa a partir das ameaças à validade.

- O Capítulo 5 apresenta a Conclusão. O texto da Conclusão discute como a pesquisa realizada por esta tese contribui para os problemas encontrados no tema, as produções bibliográficas realizadas e trabalhos futuros identificados.

O conteúdo desta tese está organizado de modo a responder às RQ, conforme apresentado na Figura 2.

Como pode ser percebido na Figura 2, a arquitetura do ITS para SM é apresentada no Capítulo 3 e busca responder à RQ2. O Capítulo 4 resolve as demais RQ através de avaliações dos Módulos Tutor e do Estudante e da integração entre esses utilizando um *Dataset* de estudantes de SM e simulação.

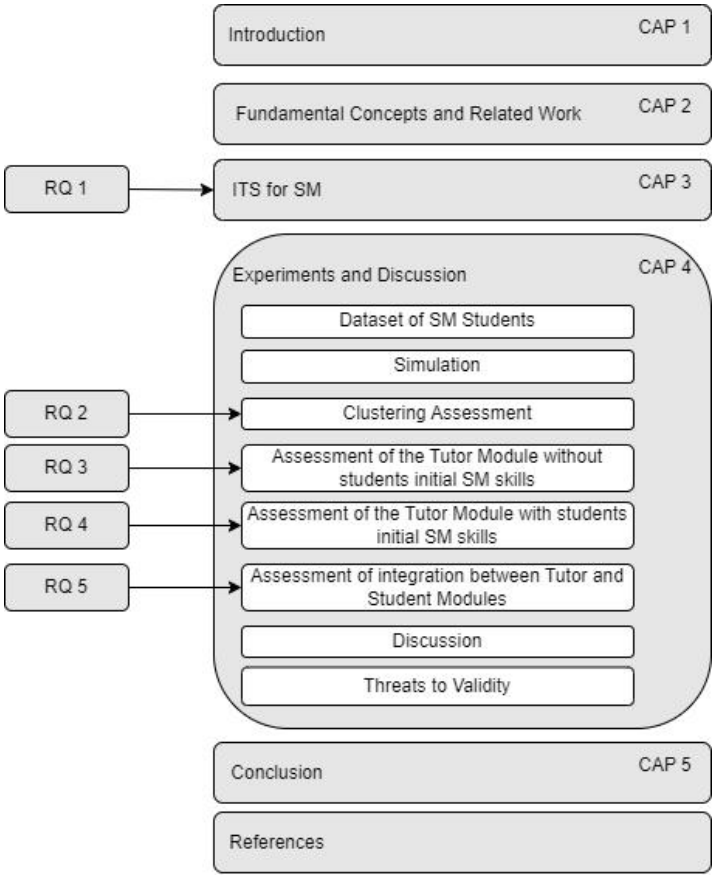

Figura 2 – Estrutura de capítulos com relação às RQ

Capítulo 2

Conceitos Fundamentais e Trabalhos Correlatos

Este capítulo apresenta os Conceitos Fundamentais, na seção 2.1, e Trabalhos Correlatos, na seção 2.2, para esta tese. Os Conceitos Fundamentais incluem Ensino-Aprendizagem em SM, ITS, Taxonomia de Bloom Revisada, AI e Técnicas de ML. Os Trabalhos Correlatos incluem ITS para o ensino-aprendizagem de Computação e ITS para SM.

2.1 Conceitos Fundamentais

Esta seção aborda os Conceitos Fundamentais desta Tese. Esses Conceitos são importantes para a compreensão do trabalho realizado e para a obtenção das contribuições desta pesquisa.

2.1.1 Ensino-Aprendizagem em Manutenção de *software* (SM)

A SM é uma atividade profissional que consome muito tempo e esforço dos profissionais de IT (FERNÁNDEZ-SÁEZ; CHAUDRON; GENERO, 2018). A Educação em Computação precisa ajudar os estudantes a se tornarem aptos para praticar SM nas organizações (HECKMAN; STOLEE; PARNIN, 2018). Pensar na complexidade da SM e ao mesmo tempo no fato dela ter sido considerada uma tarefa menos favorecida no ciclo de vida do *software* (ALLISON; JOO, 2015) traz à luz o desafio para a Educação em Computação de evoluir este tema de pesquisa. Apoiar a formação em SM é importante para trazer qualidade e profissionalismo para esta atividade profissional e melhorar a forma como ela é vista pelos profissionais nas organizações.

A definição de SM proposta pelo documento *Guide to the software Engineering Body of Knowledge* (SWEBOOK 3.0) (BOURQUE; FAIRLEY, 2014), aborda atividades de pré e pós-entrega do *software*, inclui tarefas como a logística para a transição e implantação,

o treinamento, suporte técnico e a modificação do *software* propriamente dito. A ideia da dinâmica sobre como essas tarefas de SM são executadas, relacionada ao processo de *software* utilizado em cada organização, precisa estar refletida nas pesquisas sobre educação em SM.

As recomendações da *Association for Computing Machinery* (ACM) para o currículo de Ciência da Computação (ACM Computing Curricula Task Force, 2020) apresentam como competências para *Software Development Fundamentals* as competências: Demonstrar erros comuns de codificação, construindo e depurando programas usando bibliotecas e linguagem de programação; Refatorar um programa da indústria e aplicar abstração processual. As recomendações da *Brazilian Computer Society* (SBC) para o currículo de Ciência da Computação (ZORZO et al., 2017) apresentam o eixo de formação "Gerenciamento e Processo de Software"com a competência derivada "Aplicar Técnicas e Procedimentos de Manutenção e Evolução de Software"e os temas Refatoração, Engenharia reversa, Reengenharia, Análise de impacto, Manutenção e Depuração. As Tabelas 5 e 6, apresentadas no Capítulo 3, detalham os temas trabalhados em SM considerando o currículo da disciplina.

Essas duas referências curriculares apresentadas diferenciam-se porquê a SBC adiciona os temas Engenharia reversa, Reengenharia e Análise de impacto, propondo um ensino-aprendizagem de SM com atividades mais complexas e que podem tornar sua execução prática mais trabalhosa. No entanto, a ideia de que o ensino-aprendizagem precisa possibilitar aos estudantes evoluírem nos níveis de complexidade e dificuldade de SM sem que haja a ocorrência de lacunas com partes conceituais e práticas não compreendidas implica em planejar o ensino-aprendizagem de SM considerando um equilíbrio com relação à complexidade e dificuldade.

Algumas questões nesse contexto foram estudadas pelas pesquisas. A proposta de melhorar a aprendizagem de compreensão de *software* com o uso de exemplos e auto-explicações envolvendo o código-fonte percebeu a importância de adequar o tempo para o estudante processar cognitivamente o conteúdo (ZHI et al.,). Por outro lado, a análise de como a SM foi praticada por estudantes da disciplina de Engenharia de *software* por 25 semestres na *North Carolina State University*, identificou dificuldades envolvendo a gestão do tempo, de adequação tecnológica e de compreensão do design arquitetural (HECKMAN; STOLEE; PARNIN, 2018).

Esse cenário de dificuldades relacionadas à SM, que envolvem o âmbito educacional e empresarial, traz a urgência do desenvolvimento tema e possibilitar aos estudantes o desenvolvimento das habilidades de SM, considerando as características diversidade, quantidade e complexidade (FERNÁNDEZ-SÁEZ; CHAUDRON; GENERO, 2018; ALLISON; JOO, 2015; WOOD et al., 2018; AHRENS; SCHNEIDER; BUSCH, 2019) importantes para o ensino-aprendizagem em SM, dificultando o trabalho do professor quanto a oferecer suporte adequado a turmas de SM. A complexidade operacional nessa área, torna

importante o design de ferramentas para apoiar os professores e estudantes neste processo ensino-aprendizagem.

O estudo de (KRANCHER; DIBBERN, 2012) investigou entre aprendizes SM indianos, suíços e alemães questões que afetam a aprendizagem de SM e verificou a existência de uma relação entre a aprendizagem de SM e a carga cognitiva da tarefa de SM e também entre essa aprendizagem de SM e a distância com o instrutor. Esse trabalho percebeu também que as dificuldades dos estudantes possuem uma relação com características semânticas e culturais dos mesmos e que existe um impacto positivo forte sobre a aprendizagem de SM quando esse estudante realiza tarefas cuja carga cognitiva é considerada média em relação à sua capacidade. Abordar a SM de forma mais controlada do ponto de vista pedagógico, conforme indica (KRANCHER; DIBBERN, 2012), trouxe resultados positivos e indicações sobre como realizar esse controle. No entanto, outras tentativas que visam uma maior aproximação com o mercado de SM trazem informações importantes para a continuidade da pesquisa sobre o tema.

Uma discussão sobre a relação entre o currículo acadêmico de SM e as necessidades de mercado foi realizada por (BORDIN; BENITTI, 2018) e apontou que os tópicos relacionados ao processo de manutenção, medição, reengenharia e engenharia reversa são pouco utilizados na indústria; os tópicos relacionados a sistemas legados e análise de impacto são realidade nas empresas e pouco explorados na academia; o tópico refatoração foi encontrado com equilíbrio entre academia e indústria; e que os maiores problemas de manutenção são as características do software e a gestão da manutenção, o que pode estar relacionado a qualidade da documentação, complexidade, estimativas, cronogramas e falta de planejamento.

A busca de projetos *open-source* para disciplina relacionada SM foi investigada por (SMITH et al., 2014). Esta busca precisa estar alinhada ao contexto de ensino-aprendizagem, o que traz preocupações quanto à variabilidade em tamanho, complexidade e qualidade dos projetos. A pesquisa aponta dificuldades, incluindo o excesso de tempo para operacionalizar a seleção e questões ligadas aos repositórios, como diferenças sintáticas nas buscas, inexistência de dados importantes, dificuldade de avaliar a modularidade e muito trabalho manual. A pesquisa traz critérios importantes para esta seleção, como tamanho do código, linguagem de programação, domínio da aplicação, design modular, atividade recente, qualidade da documentação, e *buildability*. A experiência com estudantes aponta que muitos deles se envolveram com a arquitetura e a representação de dados nos projetos, e muitas das dificuldades iniciais depois foram superadas com relação a compreensão do código.

Uma experiência de SM realizada por estudantes com software de tamanho médio com problemas reais mostrou que os estudantes que obtiveram notas entre 75 e 100 utilizaram estratégias específicas para compreensão do código, descritas como macro, sistemática, *top-down* e *botton-up*, e apontou dificuldades quanto à realização de testes

automatizados e ao uso de ferramentas estabelecidas (SZABO, 2015). Outra experiência de ensino-aprendizagem direcionou o conteúdo conceitual acompanhado da prática de SM com softwares *open-source* ativos contendo longos históricos de revisão, *bugs*, solicitações de melhorias, colaboradores e moderadores, trabalhando com equipes de quatro a cinco membros (GALLAGHER; FIORAVANTI; KOZAITIS, 2019), apresentou resultados positivos por atingir os objetivos do curso e aumentar o engajamento dos estudantes devido à sua conexão com o mundo real, e dificuldades, como lidar com o processo, ferramenta de controle de versão e documentar as atividades realizadas.

A proposta de atividades online para o ensino-aprendizagem de refatoração de código, que envolveu aspectos como a remoção de *bad smells*, a análise e síntese do uso de refatorações em SM, aprendizagem baseada em casos, seminário virtual e taxonomia de Bloom (LÓPEZ et al., 2014) tornou o trabalho do professor mais dinâmico e trouxe desempenhos favoráveis aos estudantes na resolução de atividades.

2.1.2 Sistema Tutor Inteligente (ITS)

ITS são softwares capazes de melhorar, adaptar e automatizar o ensino (ALKHATLAN; KALITA,). Esses sistemas oferecem ao estudante a interação com o conteúdo de maneira adaptada ao seu perfil, buscando melhorar a experiência de aprendizagem. Um ITS pode, por exemplo, recomendar conteúdos e apoiar o estudante na resolução de tarefas. A construção de um ITS envolve atividades diversas, como a definição da maneira de abordar o conteúdo de forma adaptada ao estudante e o planejamento e avaliação do uso de ML no âmbito dos seus módulos.

Preferências dos estudantes, estilos cognitivos, desempenho anterior, habilidades e crenças compõem maneiras de ITS realizarem a adaptação do conteúdo ao estudante (ERüMIT; ÇETIN, 2020). Existem variações nos nomes dados aos componentes dos ITS pelos trabalhos, apesar dos componentes responsáveis pela adaptação serem os mais importantes. Eles podem utilizar técnicas computacionais diversas, como teoria dos grafos para modelar relações entre conceitos e objetos de conhecimento e processamento de linguagem natural para compreender a entrada do usuário.

ITS foram propostos para diversas disciplinas, como Matemática, Medicina, Economia e IT (ERüMIT; ÇETIN, 2020). Isso foi possível devido a haver reúso de ideias e da arquitetura geral do ITS a partir de uma visão abstrata da mesma.

Considerando essas variações, um termo comum é *model* em vez de *module*. O EKM as vezes é chamado de modelo de domínio. Houve, nesta tese, a escolha de uma nomenclatura comum aos componentes do ITS para melhorar a compreensão e a comparação. Os nomes de Módulos escolhidos foram: Tutor, Estudante, EKM e *User Interface* (UI). A Figura apresenta a Arquitetura Geral de um ITS.

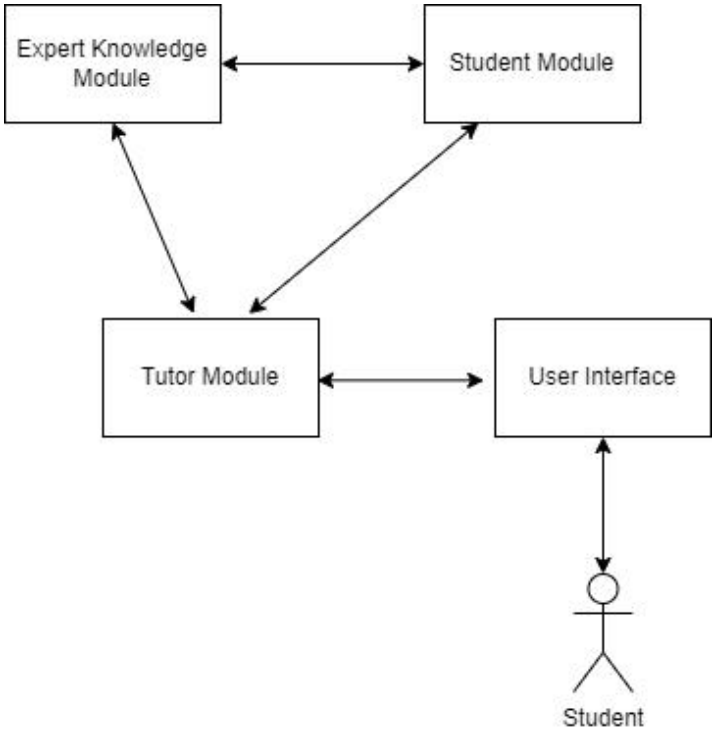

Figura 3 – Arquitetura Geral de um ITS. Adaptado de (ALKHATLAN; KALITA,).

O Módulo Tutor é responsável por realizar ações pedagógicas. Este módulo necessita de algoritmos para definir o limite de intervenção do ITS no processo de estudo, operacionalizar esta intervenção e recomendar conteúdo. O Módulo do Estudante precisa capturar, atualizar e recuperar a representação do perfil do estudante para facilitar a adaptação. O EKM precisa representar e manipular o conteúdo de aprendizagem. O EKM e o Módulo do Estudante fornecem informações para que o Módulo do Tutor execute suas tarefas, e.g., a recomendação de conteúdo adaptada ao estudante. O Módulo de UI permite que os estudantes interajam com o conteúdo e com o comportamento de um ITS.

2.1.2.1 Expert Knowledge Module

O EKM tem o objetivo de representar aspectos como fatos, conceitos, estratégias de resolução de problemas e regras relacionadas ao domínio do conteúdo a ser ensinado e também disponibiliza ao ITS o conteúdo para que este seja entregue ao estudante (ALKHATLAN; KALITA,). O conteúdo a ser adicionado no EKM, que é capturado do domínio, é construído a partir de explicações, perguntas e respostas e formas de avaliar o desempenho do estudante. O EKM pode conter um modelo especialista capaz de gerar caminhos de resolução de problemas para orientar o estudante na resolução de problemas. Para representar o conhecimento no EKM, são utilizados as linguagens de representação de conhecimento, esquemas de representação e raciocínio de conhecimento, que incluem *symbolic rules, fuzzy logic, Bayesian networks, case-based reasoning*, representações híbridas como as abordagens *neuro-symbolic* e *neuro-fuzzy*, e modelos de conhecimento de

domínio, como o modelo cognitivo e modelo baseado em restrições. Por exemplo, O ITS Heraclito (GALAFASSI et al., 2019) possui a representação do conhecimento de Dedução Natural em Lógica Proposicional e possibilita ao estudante uma aprendizagem que envolve exercícios contendo fórmulas, tabelas verdade e provas.

2.1.2.2 Módulo do Estudante

O Módulo do Estudante precisa representar dinamicamente o conhecimento e habilidades dos estudantes de modo que esta representação possa ser explicitamente comunicada no âmbito dos módulos do ITS para que o ITS possa inferir sobre o desempenho e sobre as habilidades do estudante. A representação do estudante contribui para o ITS inferir aspectos não observáveis do comportamento do estudante para a reconstrução de conceitos errados no conhecimento do estudante. Projetar a representação do estudante não é fácil e deve se basear na busca de responder duas perguntas, que são: O que o estudante sabe? E quais conhecimentos ele precisa para resolver um problema do referido domínio? A representação do conteúdo gerenciada pelo EKM geralmente é usada como base para a representação do conhecimento no Módulo do Estudante (ALKHATLAN; KALITA,).

A representação do estudante inclui informações estáticas e dinâmicas e pode ser realizada a partir de várias abordagens a partir de dados capturados e inferidos pelo sistema (ALKHATLAN; KALITA,). As informações estáticas são capturadas no início do uso do ITS, como idade, língua materna e e-mail, e as informações dinâmicas envolvem características dinâmicas dos estudantes, como conhecimentos, habilidades, erros, preferências de aprendizagem e fatores afetivos e cognitivos. Existem várias possibilidades de abordagens para construir a representação do estudante, como: Modelo de Sobreposição, Modelo de Perturbação, Modelo de Estereótipos e Modelagem *Fuzzy* do Estudante. O Modelo de Sobreposição, por exemplo, pressupõe que o conhecimento do estudante é um subconjunto do conhecimento do domínio representado no EKM. O Modelo de Perturbação estende o Modelo de Sobreposição incluindo possíveis erros dos estudantes. O modelo de estereótipos considera atributos em comum entre estudantes para o ITS fazer inferências usando um número menor de observações. Caso as caraterísticas de um novo estudante forem similares a um estereótipo, o sistema atribui o estudante a determinado estereótipo. Como pôde ser percebido, as abordagens são muito distintas. A Modelagem *Fuzzy* do Estudante, por sua vez, é usada quando certas características não podem ser diretamente observadas e medidas pelo ITS.

2.1.2.3 Módulo Tutor

O Módulo Tutor decide o que apresentar ao estudante e fornece suporte adaptado ao estudante na resolução de exercícios (ALKHATLAN; KALITA,). Esse suporte pode incluir *feedbacks*, dicas, explicações com relação a regras não compreendidas que são importantes na resolução de problemas e recomendações de conteúdos para reforçar o estudo.

A característica da adaptação ao estudante, exercida pelo Módulo Tutor, é possível graças a sua colaboração com o Módulo do Estudante. O funcionamento do Módulo Tutor precisa incluir mecanismos que tenham a capacidade de inferir questões pedagógicas do processo ensino-aprendizagem para realizar ações pedagógicas que sejam percebidas de maneira natural, como identificar o momento mais adequado para apresentar um conceito ou avaliar o desempenho do estudante.

Certas técnicas são usadas na construção do Módulo Tutor, como *Decision Making in Cognitive Tutor and Constraint Based Systems, Tutorial Dialog in Natural Language* e *Spoken Dialogue* (ALKHATLAN; KALITA,). *Decision Making in Cognitive Tutor and Constraint Based Systems* trabalham com a ideia de Tutores de Rastreamento de Modelos, ou Tutores Cognitivos, e fornecem *feedbacks* do tipo, *feedbacks* de sinalização, mensagens com erros e cadeia de dicas. Esses ITS recomendam sobre áreas nas quais um estudante precisam concentrar seu estudo. *Tutorial Dialog in Natural Language* e *Spoken Dialogue* apresentam a funcionalidade do Módulo Tutor a partir de diálogos construídos em linguagem natural e diálogo falado.

O ITS ViSiTR (OBERHAUSER, 2017), que trabalha com uma abordagem de visualização 3D e recomendação de código-fonte para apoiar a compreensão de software, possui um Módulo Tutor baseado em modelos de aprendizagem cognitiva e em uma teoria de compreensão de programas. Por outro lado, o trabalho de (ALLISON; JOO, 2015) descreve um Módulo Tutor centrado no estudante e baseada no conceito de zona de desenvolvimento proximal de Vygotsky e nas categorias corretiva e desafiadora para Engenharia de Software. Nesse sentido, as pesquisas possuem como desafios encontrar maneiras de operacionalizar computacionalmente essas funcionalidades.

2.1.3 Taxonomia de Bloom Revisada

A Taxonomia de Bloom (AMER, 2006) representa a estrutura das dimensões do processo cognitivo. Existem seis processos cognitivos principais *(Remember, Understand, Apply, Analyze, Evaluate, Create)* que se subdividem em 19 outros processos cognitivos. Esses processos cognitivos descritos na taxonomia podem ser utilizados para categorizar DM em ITS. A hierarquia de complexidade existente entre esses processos pode auxiliar os ITS a diagnosticar a capacidade cognitiva do estudante perante a determinado conteúdo e consequentemente a tomar decisões pedagógicas adequadas. A Figura 4 apresenta a estrutura da Taxonomia de Bloom Revisada e a relação entre a hierarquia dos seus processos cognitivos e as habilidades de pensamento de ordem superior.

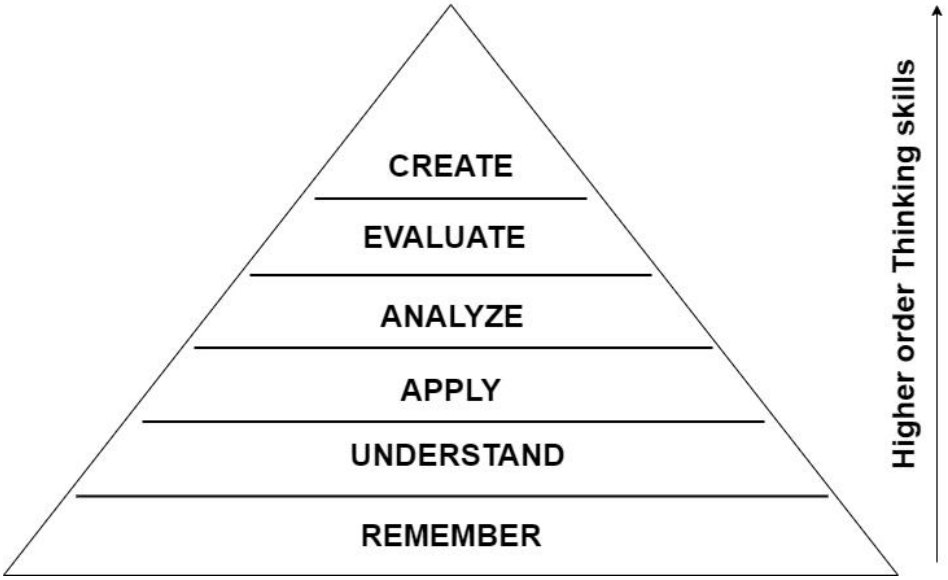

Figura 4 – Estrutura da Taxonomia de Bloom Revisada. Adaptado de (RAZMJOO; KAZEMPOURFARD, 2012)

A estrutura hierárquica da taxonomia de Bloom revisada indica que um exercício disponível para um estudante resolver atinge o desenvolvimento de um determinado processo cognitivo e de todos os demais processos cognitivos inferiores (MASAPANTA-CARRIÓN; VELÁZQUEZ-ITURBIDE, 2019). Essa compreensão é importante para a construção de tecnologias educacionais, que precisam gerenciar a relação entre níveis de dificuldades que são disponibilizados para os estudantes e o seu desenvolvimento cognitivo com relação ao conteúdo que ele se propõe a estudar com o apoio tecnológico.

Quando um professor realiza a classificação de um exercício, segundo a taxonomia de Bloom Revisada, ocorre a identificação da sequência de processos cognitivos esperados que o estudante típico realize para resolver o exercício (MASAPANTA-CARRIÓN; VELÁZQUEZ-ITURBIDE, 2019). Essa classificação por parte do professor indica a possibilidade de construção de tecnologias educacionais para que professores criem exercícios e conteúdos e realizem essa classificação para que a tecnologia seja mais assertiva em considerar a capacidade cognitiva do estudante.

2.1.4 Inteligência Artificial

A definição do conceito de AI, de acordo com (WANG, 2019), passa por uma discussão sobre as suas definições deste com relação a aspectos como o comportamento adaptativo de sistemas às demandas do ambiente, capacidade de resolução de problemas, métodos para atender a objetivos em situações em que as informações disponíveis são complexas e métodos que simulam a um humano resolvendo problemas.

A comunidade de Ciência da Computação identifica a AI por meio de suas técnicas desenvolvidas, que variaram em diferentes períodos (WANG, 2019). Essas técnicas incluem,

entre outros, prova de teoremas, pesquisa heurística, sistemas especialistas, redes neurais, redes bayesianas, agentes e aprendizado profundo. Elas possuem diferenças relacionadas aos fundamentos teóricos utilizados e aos tipos de problemas alvo. Representação do conhecimento, raciocínio, planejamento, ML e processamento de linguagem natural são exemplos de subdomínios que foram formados dentro de AI.

A matemática trouxe três grandes contribuições para a AI, que foram a Lógica, a Ciência da Computação e a Probabilidade (RUSSEL; NORVIG, 2013). A Lógica trouxe conceitos como a lógica de primeira ordem e teoria de referência, que trabalham com objetos do mundo real e relações, a Ciência da Computação trouxe conceitos como o algoritmo, teorema da incompletude, computabilidade e tratabilidade e a Probabilidade, juntamente com o desenvolvimento de novos métodos estatísticos, trouxe possibilidades de atualizar as probabilidades à luz de novas evidências e lidar com o raciocínio incerto.

Outras áreas, como a Economia, a Pesquisa Operacional, a Neurociência, a Psicologia e a Linguística também trouxeram grandes contribuições para a AI (RUSSEL; NORVIG, 2013). Relacionado à Economia e à Pesquisa Operacional, a teoria da decisão, que combina teoria da probabilidade com teoria de utilidade e fornece uma estrutura formal para decisões tomadas sob incerteza utilizando dados capturados do ambiente, a teoria dos jogos, que traz a ideia de que um agente racional precisa também adotar políticas de decisão aleatórias, o problema de realizar ações em sequência em situações em que a recompensa não é imediata e a classe de problemas de decisão sequencial chamada processos de decisão de Markov foram conceitos utilizados nas pesquisas sobre AI. A Neurociência contribuiu com ideias relacionadas ao mapeamento, modelos matemáticos e funcionamento dos cérebros e sistema nervoso. A psicologia trouxe contribuições do behaviorismo e psicologia cognitiva, como os três passos fundamentais para um agente baseado em conhecimento. A relação entre AI e linguística foi beneficiada a partir do trabalho do linguista Noam Chomsky, possibilitando o desenvolvimento do campo processamento de linguagem natural, ou linguística computacional. Ideias como a formas de representar o contexto e o conhecimento vieram da união dessas duas áreas distintas de pesquisa.

Historicamente, o desenvolvimento da AI teve que lidar com certos problemas que surgiram, como a dificuldade de incluir conhecimento de assuntos específicos nos programas, a dificuldade de lidar com a explosão combinatória e limitações que as estruturas que estavam sendo usadas para gerar comportamento inteligente possuíam (RUSSEL; NORVIG, 2013). As redes neurais, por exemplo, tiveram seu desenvolvimento retomado em 1986, com inovações relacionadas ao algoritmo de aprendizado por retroprogramação e modelos conexionistas.

De 2001 até a atualidade, deferente da na Ciência da Computação que enfatizava o algoritmo, a área de AI percebeu que havia uma necessidade de direcionar suas preocupações com os dados, dada a disponibilidade de trabalhar com grandes fontes de dados, e sugeriu que existe um gargalo do conhecimento e AI (RUSSEL; NORVIG, 2013). Esse

gargalo envolve expressar todo o conhecimento que um sistema precisa, que pode, ao invés de ser construído com engenharia de conhecimento, ser obtido através de métodos de aprendizagem disponíveis em ML.

As seções 2.1.4.1, 2.1.4.2 e **??** apresentam subáreas da AI que contribuem para a compreensão da dimensão da AI. A subárea denominada ML, cujas técnicas foram usadas nesta tese, é apresentada na seção 2.1.5.

2.1.4.1 Agentes Inteligentes

Um Agente Inteligente é uma estrutura capaz de perceber seu ambiente por meio de sensores e agir nesse ambiente por meio de atuadores (RUSSEL; NORVIG, 2013). Alguns agentes se comportam melhor do que outros, ou seja agem de forma racional. No entanto, aferir a performance de um agente depende da qualidade da percepção do estado do ambiente.

A Figura 5 ilustra a relação entre agente e ambiente. A sequência de percepções de estados do ambiente influenciará a ação que o agente irá executar. A função do agente artificial é uma descrição matemática que mapeia uma sequência de percepções em uma ação e é implementada pelo programa do agente (RUSSEL; NORVIG, 2013).

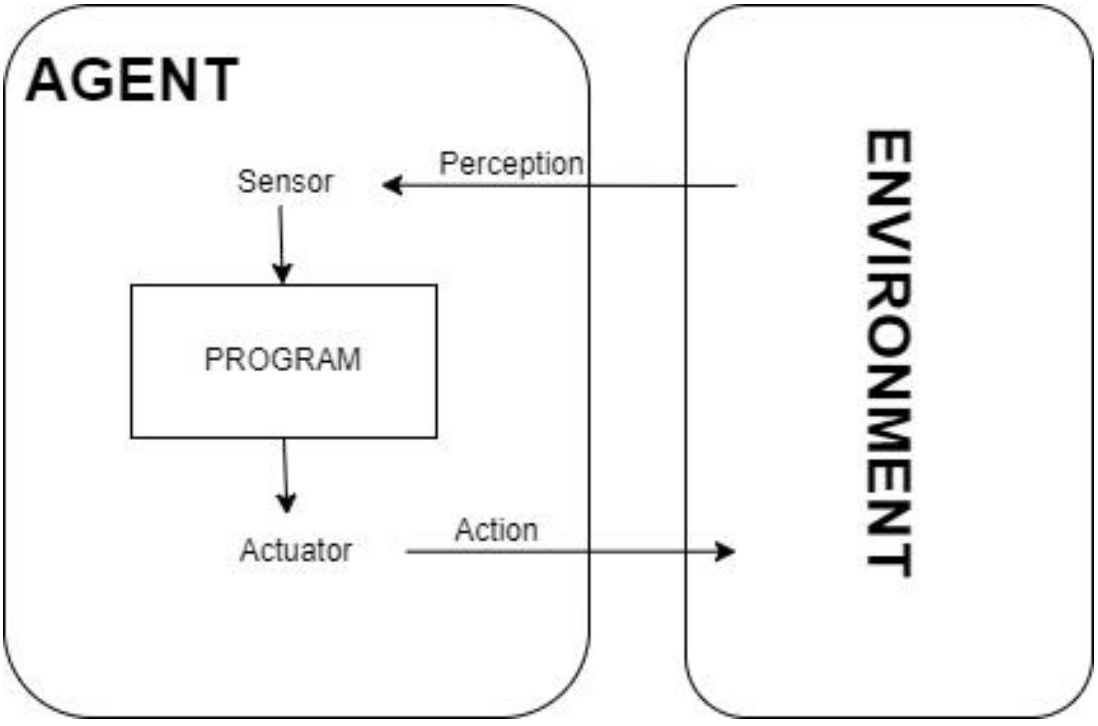

Figura 5 – Relação entre Agente e Ambiente. Adaptado de (RUSSEL; NORVIG, 2013)

Um Agente Racional faz tudo corretamente. Se as ações executadas pelo agente no ambiente fizerem com que a sequência de estados do ambiente for desejável, de acordo com uma medida de desempenho, o agente teve um bom desempenho (RUSSEL; NORVIG, 2013). A racionalidade depende dos quatro fatores, que são: a medida de desempenho,

conhecimento prévio do ambiente, as ações possíveis ao agente, e a sequência de percepções do ambiente até o momento.

Os conceitos de onisciência, aprendizado e autonomia são importantes para a compreensão dos agentes e ambientes (RUSSEL; NORVIG, 2013). A onisciência indica que um agente conhece com exatidão o resultado real de suas ações, porém é inatingível. A aprendizagem indica que um agente aprende com suas percepções do ambiente. O agente pode ter um conhecimento prévio do ambiente e durante sua execução ocorre a ampliação desse conhecimento. Um agente autônomo deve aprender para compensar o conhecimento prévio e possivelmente incorreto.

A especificação do ambiente, através da ideia de ambiente de tarefa, envolve descrever os quatro itens: medida de desempenho, ambiente, atuadores e sensores (RUSSEL; NORVIG, 2013). Um ambiente de tarefas para um táxi automatizado, por exemplo, possui a descrição (Tipo de Agente: Motorista de Táxi; Media de Desempenho: Viagem segura, rápida, confortável, maximizar lucros; Ambiente: Estradas, pedestres e clientes; Atuadores: Direção, acelerador, freio, sinal buzina, visor; Sensores: Câmeras, velocímetros e sensor do motor.

Os ambientes de tarefas podem divididos em categorias, como completamente observável, parcialmente observável, agente único, multiagente, determinístico, estocástico, episódico, sequencial, entre outros (RUSSEL; NORVIG, 2013). Se os sensores conseguem perceber tudo que é importante para uma ação, o ambiente é completamente observável, e se não, ele é parcialmente observável. Pode ocorrer a existência de um agente único ou de muitos agentes em um ambiente, chamado de multiagente e envolver aspectos como cooperação, competição, comunicação. Se o estado atual do ambiente e a ação realizada determinam com exatidão o novo estado do ambiente, ele é determinístico, se não, ele seria estocástico. Em um ambiente episódico, a experiência do agente é dividida em episódios atômicos e para cada episódio o agente percebe o estado do ambiente e realiza uma ação. O ambiente sequencial indica que a ação realizada no momento atual tem consequências a longo prazo, o que ocorre no jogo de xadrez e dirigir um táxi.

2.1.4.2 Representação do Conhecimento

A Representação do Conhecimento envolve descrever ontologias para domínios complexos, que exigem representações mais gerais e flexíveis (RUSSEL; NORVIG, 2013). Existem conceitos gerais que são comuns a vários domínios e situações, que são: eventos, tempo, objetos físicos e crenças. Descrever esses conceitos abstratos é comumente chamado de engenharia ontológica.

Uma ontologia é um acordo social sobre um domínio e possui componentes organizados em uma estrutura de categorias (RUSSEL; NORVIG, 2013). Essas categorias permitem raciocínios e deduções com o auxílio de ferramentas como a lógica de primeira ordem. A representação de categorias em lógica de primeira ordem ocorre pelas duas opções que são

predicados e objetos, através de definições como pertinência, subconjunto, subcategoria e subclasse. As categorias simplificam o conhecimento por herança e são organizadas em uma hierarquia taxonômica.

As diferenças com relação aos tipos de informação existentes nos diferentes domínios de aplicação são representadas e tratadas utilizando diferentes abordagens (RUSSEL; NORVIG, 2013). Diferente da ideia de eventos discretos, existem os processos ou categorias de eventos líquidos, em que há a necessidade de realizar cálculos envolvendo intervalos de tempo variados, como no caso de um evento como um voo. Esses cálculos são manipulados por funções que representam, por exemplo, início, fim, instante e duração e o conjunto de relações de intervalo completo proposto por (ALLEN,). A seguir são apresentados exemplos de relações de intervalo completo nas Equações 1, 2, 3, 4 e 5.

$$Encontram(i,j) <=> Fim(i) = Início(j) \tag{1}$$

$$Antes(i,j) <=> Fim(i) < Início(j) \tag{2}$$

$$Depois(j,i) <=> Antes(i,j) \tag{3}$$

$$Durante(i,j) <=> Início(j) < Início(i) < Fim(i) < Fim(j) \tag{4}$$

$$Sobrepõe(i,j) <=> Início(i) < Início(j) < Fim(i) < Fim(j) \tag{5}$$

A Equação 1 indica um encontro temporal entre o fim do evento i e início do evento j. A Equação 2 indica que o fim do evento i ocorre antes do início do evento j. A Equação 3 indica o oposto da Equação 2. A Equação 4 indica que o evento i ocorreu durante o evento j. A Equação 5 indica que ocorre uma sobreposição do evento i em relação ao evento j.

Objetos são representados no espaço-tempo. Eventos mentais e objetos mentais são tratados com a ideia de crença, sobre algo que está na cabeça de alguém ou em alguma base de conhecimentos e sobre os processos mentais que manipulam essa informação. As atitudes proposicionais de um agente sobre objetos mentais (acredita, sabe, quer, pretende e informa) e a lógica modal, que visa lidar com mundos possíveis e não mundos verdadeiros, ajudam a lidar com esses objetos e eventos mentais (RUSSEL; NORVIG, 2013).

Esses conceitos são aplicados na Web através da Web-Semântica. A Web-Semântica significa propor uma Web que compartilhe dados e fatos ao invés de texto. Para isso, existe uma pilha de tecnologias, que envolvem formatos para intercâmbio de dados, ontologia, linguagens para consultar dados considerando sua distribuição. As ontologias permitem a organização e a integração de dados de várias fontes da Web, comunicação eficiente quanto

aos significados dos dados da Web e raciocínio sobre os dados (PATEL; JAIN, 2021). Nesse sentido, apesar da complexidade de trabalhar com a Representação de Conhecimento, ela permite a construção de sistemas de *software* capazes de oferecer funcionalidades complexas, e.g. através da descoberta de novos conhecimentos através da inferência e do reuso de dados disponíveis na Web.

2.1.5 Aprendizado de Máquina

ML é uma subárea da AI que tem como base algoritmos que se propõem a aprender com os dados de modo que não exista dependência de programação baseada em regras, o que permite a cientistas de dados treinarem automaticamente modelos que são abstrações dos dados (PYLE; JOSÉ, 2015). ML é formada por algoritmos com estratégias de aprendizado para a construção de modelos. Aprendizado Supervisionado, Aprendizado Não-Supervisionado e RL fazem parte dessas estratégias (MISILMANI; NAOUS, 2019).

A execução de ML na prática utiliza dados como entrada e divide-os em três partes (SHALEV-SHWARTZ; BEN-DAVID, 2014), que são: dados para treinamento de modelo através de algoritmo de ML, dados de validação cruzada usado para seleção de modelo e dados de teste para analisar o desempenho do algoritmo. Em contextos que vão de bancos, pequenas empresas a *games* a ideia da aprendizagem automática que a ML trouxe permitiu a criação de modelos capazes de realizar, e.g. predições e classificações, relacionadas aos dados, e.g. com relação ao comportamento de jogadores, vendas e comportamento de clientes sobre pagamento (PYLE; JOSÉ, 2015). Isso permite às organizações modificarem seus processos de modo a torná-los mais automatizados e eficientes.

ML é considerada uma abordagem para alcançar aplicações de AI (MISILMANI; NAOUS, 2019). ML e ITS são capazes de contribuir com uma educação de alta qualidade. Raciocínio, inferência e aprendizado são possibilidades que ML traz e que são aplicados na construção de ITS. Essas técnicas permitem aos ITS, por exemplo, modelar o estado de conhecimento do estudante, separar estudantes em grupos e oferecer dicas de acordo com o perfil e estado de conhecimento do estudante (ALSHAIKH; HEWAHI, 2021).

Esta seção aborda técnicas de ML, que são: Aprendizado Supervisionado, Aprendizado Não-Supervisionado e RL. RL e Aprendizado Não-Supervisionado são os tipos de ML utilizados nesta tese. O algoritmo *Q-Learning*, no contexto de RL, e a Clusterização, no contexto de Aprendizado Não-Supervisionado, foram utilizados nesta tese.

2.1.5.1 Aprendizado Supervisionado

O Aprendizado Supervisionado permite a construção de modelos abstraídos a partir de treinamento de dados rotulados (MISILMANI; NAOUS, 2019). A generalização contida em um modelo criado a partir de um conjunto de dados fornece previsões precisas para um conjunto grande de entradas possíveis (MARSLAND, 2015). *Linear regression, logistic*

regression, artificial neural networks, e *support vector machines* são algoritmos que fazem parte do conjunto de técnicas de ML do tipo Aprendizado Supervisionado (MISILMANI; NAOUS, 2019). O treinamento, a partir dos dados rotulados e a computação fornecida por esses algoritmos, possibilita a construção de modelos matemáticos que permite a realização de testes e verificação de acurácia, o que torna sua aplicação bastante útil.

Neural Network é um exemplo de algoritmo de Aprendizado Supervisionado. Trata-se de um modelo de computação que simula o funcionamento de neurônios humanos representados por nós conectados (WU; FENG, 2018). O seu funcionamento, e.g. para tarefas de classificação, pode ser compreendido com as fases de treinamento e teste. No treinamento, ocorre o ajuste de um modelo matemático, que é dependente da topologia da rede, dos dados de entrada e dos cálculos realizados na configuração escolhida do algoritmo, por meio de pesos que ajustam funções de ativação de unidades neurônios, realizando transformações que não podem ser compreendidas fora do sistema (WU; FENG, 2018). No teste, esse modelo é aplicado a novas entradas, classifica-se a informação utilizando o modelo construído e compara-se esse resultado com o rótulo.

2.1.5.2 Aprendizado Não-Supervisionado

O Aprendizado Não-Supervisionado trabalha com dados que não são rotulados e possibilita derivar uma estrutura a partir dos dados e suas semelhanças (MARSLAND, 2015). *K-Means Clustering* e *Dimentionality Reduction Algorithms* são exemplos desse tipo de algoritmos de ML (MISILMANI; NAOUS, 2019).

Clusterização é uma técnica de Aprendizado Não-Supervisionado bastante usada em Ciência de Dados para separar dados por *clusters*, que são sub-grupos de dados em um conjuntos de dados, de modo que a estrutura de *clusters* gerada representam maior similaridades entre os pontos dos *clusters* e maior disimilaridade entre os diferentes *clusters* (SINAGA; YANG, 2020). As seções a seguir, descrevem algoritmos de Clusterização.

K-Means

O Algoritmo *K-Means* tem como objetivo dividir n instâncias em K *clusters* (JAHWAR; ABDULAZEEZ, 2020). Sua estratégia trabalha com a ideia de centroide do *cluster*, distância entre os objetos do *cluster* e o respectivo centroide e a diminuição da quantidade de erros quadráticos. Para executar o processo, o algoritmo possui uma matriz de distância e a distância euclidiana.

O algoritmo 1 define os passos percorridos pelo *K-means*. Primeiro, é realizada a seleção de K pontos como centroides iniciais. Feito isso, é realizado um *loop* até os centroides se estabilizarem. Neste *loop*, os objetos são associados aos centroides mais próximos, formando os *clusters*, e os centroides são recalculados considerando os *clusters* recém criados.

Algoritmo 1 Algoritmo K-Means (Adaptado de (JAHWAR; ABDULAZEEZ, 2020))

```
c ← data.Centroids(k)
while old.Centroids <> c do
    data.group(c)
    old.Centroids ← c
    c ← data.CentroidsRecalc(k)
end while
```

Spectral Clustering

O algoritmo *Spectral Clustering* é baseado na ideia de grafo não direcionado ponderado (JANANI; VIJAYARANI, 2019). Este algoritmo representa os objetos no formato de grafo não direcionado a partir de vértices, arestas e a matriz de afinidade do grafo. Ele possui três etapas, que são: similaridade do grafo, matriz laplaciana do grafo, vetores próprios. A similaridade do grafo é calculada a partir de ideias referentes ao grafo de vizinhança, grafo de vizinho mais próximo e grafo totalmente conectado. A matriz laplaciana do grafo trabalha com os seguintes conceitos: grafo laplaciano não normalizado, grafo laplaciano normalizado e passeios aleatórios. Vetores próprios expressam que agrupamentos adequados foram realizados.

Agglomerative Clustering

Agglomerative clustering é uma abordagem de clusterização que trata cada objeto de dados inicialmente como um *cluster* e que a cada iteração mescla os *clusters* por similaridade de maneira hierárquica através de comparações utilizando métricas de distância (SHARMA; BATRA et al., 2019). Este processo é executado até que um grande *cluster* é formado. Este método de clusterização traz qualidade nos resultados oferecidos, porém é executado com um alto custo computacional quanto a tempo e armazenamento. A hierarquia dos *clusters* gerados como resultado é apresentada em uma estrutura de dendrogramas.

Balanced Iterative Reducing and Clustering Using Hierarchies (BIRCH)

Balanced Iterative Reducing and Clustering Using Hierarchies (BIRCH) é um algoritmo de Clusterização desenvolvido para grandes bases de dados com benefícios para quando os dados não podem ser carregados na memória (FONTANINI; ABREU, 2018). Este método é dividido em quatro fases: carregar na memória os dados representados no formato de árvore de recursos de *clustering*; condensar os dados; *clustering* global; refinar *cluster*. A segunda e quarta fase são opcionais.

Bisecting K-Means

O Algoritmo *Bisecting K-Means* utiliza ideias do agrupamento hierárquico e do algoritmo *K-Means*. Ele inicia considerando o conjunto de dados como sendo um único *cluster* e a cada iteração vai dividindo cada *cluster* em dois *clusters* (MAHMUD et al., 2018). O algoritmo possui as quatro etapas: a) colocar um *cluster* para ser separado; b) descobrir dois intragrupos utilizando o *K-Means*; c) realize ensaios da etapa (b) e pegue a divisão que traz um agrupamento com maior similaridade; d) repetir as etapas (a, b e c) até atingir o número pretendido de *clusters*.

2.1.5.3 Aprendizado por Reforço

No RL as informações são coletadas após a interação com o ambiente, onde a máquina captura o que se chama de reforço (MISILMANI; NAOUS, 2019). Esse reforço traz uma recompensa que qualifica a ação realizada, que pode ser e.g. positiva ou negativa.

A ideia de RL se relaciona com o conceito de Agentes Inteligentes, pois no RL ocorre também a interação entre agente e ambiente que precisam sempre ajustar seu modelo para que seja capaz de maximizar as recompensas obtidas através do reforço do ambiente em resposta às suas ações. Em domínios complexos e situações que não existem dados rotulados para uso de outros tipos de aprendizagem, a RL pode ser o único caminho possível para construir um modelo de ML de alto nível (RUSSEL; NORVIG, 2013). A seguir é apresentado um algoritmo de RL.

Q-Learning

Q-Learning (SHAWKY; BADAWI, 2018) é um algoritmo de RL que constrói e aprimora em tempo de execução um modelo que decide a melhor ação para um dado estado do ambiente. Uma matriz de ações e estados que contém probabilidades é atualizada a cada reforço que o ambiente gera, o que permite o aprimoramento do modelo para que melhores ações sejam executadas no futuro.

As notações e etapas do *Q-Learning* foram adaptadas de (GUPTA; ROY; DUTT, 2021) para facilitar a compreensão de seu funcionamento.

Notações: (Adaptado de (GUPTA; ROY; DUTT, 2021))

A(s): Ações disponíveis para o estado s

R(s): Reforço obtido no estado s

Q(s, a): Atualizar valor para a tupla ação-estado

Episode: Uma execução completa do algoritmo

α: Taxa de aprendizagem

γ: Fator de desconto

Inicialmente a matriz Q, composta de tuplas de ações e estados, é construída randomicamente. Depois disso, o algoritmo executa seus processos de acordo com a quantidade

Algoritmo 2 Algoritmo *Q-Learning* (Adaptado de (GUPTA; ROY; DUTT, 2021))

```
Q(s, a).startRandom()
while ep <= kitEpsode do
    s.start()
    while s <= ep.kitStates() do
        a ← A(s, a).selectAction()
        s.execute(a)
        s' ← reinforcement.stateUpdated()
        r ← reinforcement.r()
        Q(s, a) ← Q(s, a) + α[r + γ * (maxQ(s', a')) − Q(s, a)]
        s ← s'
    end while
end while
```

de episódios previamente definida. Para cada um desses episódios, o estado atual é inicializado e um outro *loop* é executado para cada estado "s" do episódio conforme ocorram as modificações entre os estados. Feito isso, o algoritmo escolhe a melhor ação de acordo com o grau de exploração definido na política, executa esta ação no estado atual, atualiza o estado e obtém o reforço para atualizar seu modelo de decisão representado na matriz Q.

A equação abaixo representa a forma do algoritmo atualizar a matriz Q.

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma * (maxQ(s',a')) - Q(s,a)] \tag{6}$$

Esta atualização da matriz Q ocorre sempre que o algoritmo realiza uma recomendação de ação e esta é executada. Q(s, a) representa o benefício de se executar a ação a no estado s. α e γ são os parâmetros taxa de aprendizagem e fator de desconto. A função maxQ(s', a') retorna o *score* da melhor ação (a') possível no novo estado (s').

2.2 Trabalhos Correlatos

Esta seção apresenta os Trabalhos Correlatos. Esses trabalhos são descritos nas seções 2.2.1 e 2.2.2 conforme as duas categorias: ITS para Computação e ITS para SM. O tema ITS para SM foi pouco explorado de forma enfática pelas pesquisas. Isso trouxe a percepção de que analisar os ITS para Computação auxilia na identificação de um subconjunto desses trabalhos para o contexto de SM e de que as ideias, conceitos e técnicas no contexto dos ITS para Computação podem ajudar a avançar a pesquisa sobre ITS para SM.

2.2.1 ITS para Educação em Ciência da Computação

Esta seção tem como objetivo revisar a literatura sobre os ITS para a Educação em Ciência da Computação. O escopo da Educação em Ciência da Computação considerado nesta pesquisa é o ensino superior. O trabalho (FRANCISCO; SILVA, 2022b) apresenta

detalhes sobre esta revisão da literatura, que foi realizada em 2022, trouxe trabalhos de 2012 a 2021, e foi atualizada para esta tese.

As Tabelas 1 e 2 objetivam ajudar na compreensão do cenário de uso e desenvolvimento dos ITS na Educação em Ciência da Computação. Ela lista os trabalhos encontrados que se enquadram em Sistemas Tutores com domínios de aplicação alinhados ao Currículo da ACM (ACM Computing Curricula Task Force, 2013). Ela apresenta também se foram apresentados os três módulos de um ITS tradicional e detalha quais destes módulos foram abordados.

Apesar de (ACM Computing Curricula Task Force, 2020) ser a última publicação da ACM das referências curriculares para Ciência da Computação, esta seção usou a referência (ACM Computing Curricula Task Force, 2013) devido a ela apresentar uma organização de disciplinas e assuntos baseada em *Body of Knowledge* e tópicos, que puderam ser vinculados diretamente às áreas de aplicação dos ITS encontrados. O trabalho (ACM Computing Curricula Task Force, 2020) apresenta uma organização do conteúdo de Ciência da Computação utilizando a ideia de *Draft Competencies*, que indica para cada disciplina os objetivos planejados para estudantes atingirem sem uma definição precisa, o que dificulta na classificação precisa desses trabalhos encontrados.

De acordo com os trabalhos encontrados, foi possível perceber percebemos a existência de duas categorias de trabalhos. A primeira categoria deixa evidente que o trabalho apresenta um ITS conforme a sua definição conceitual. No entanto, a segunda categoria não deixa evidente que se trata de um ITS, permitindo afirmar que se trata de Sistemas Tutores.

Dentre os trabalhos, alguns deles apresentaram um ITS para Ensino de Ciência da Computação (RAHMAN; ABDULLAH; ALIAS, 2016; ALSHAIKH; TAMANG; RUS, 2021; CAO, 2023; CARTER; BLANK, 2013; CHRYSAFIADI et al., 2023; FIGUEIREDO; GARCÍA-PEÑALVO, 2020; FRANCISCO; SILVA, 2022a; GALAFASSI et al., 2020; HARSLEY et al., 2016; HASANEIN; NASER, 2017; HOOSHYAR et al., 2018; HUANG et al., 2023; JEREMIC; JOVANOVIC; GASEVIC, 2009; JEREMIC; JOVANOVIĆ; GAŠEVIĆ, 2012; LUBURIć et al., 2022b; LUBURIć et al., 2022a; NKAMBOU et al., ; PRICE; DONG; LIPOVAC, 2017; SHARMA; HARKISHAN, 2022; TROUSSAS et al., 2023; VERDÚ et al.,). Os módulos encontrados nestes trabalhos estão alinhados com a definição conceitual dos módulos ITS. Apresentam dinâmicas baseadas em algoritmos e representações de dados que possibilitam a execução das funcionalidades do ITS.

Conforme apresentado nas Tabelas 1 e 2, diversos conteúdos de Ciência da Computação foram abordados nos ITS. Entre esses conteúdos, é possível exemplificar: Design de Redes de Computadores (VERDÚ et al.,), Programação de Computadores (TROUSSAS et al., 2023), e Dedução Natural na Lógica Proposicional (GALAFASSI et al., 2020).

Foram encontrados dez (10) trabalhos (ALSHAIKH; TAMANG; RUS, 2021; FIGUEIREDO; GARCÍA-PEÑALVO, 2020; HOOSHYAR et al., 2018; PRICE; DONG; LIPO-

Tabela 1 – Trabalhos sobre ITS para Educação em Ciência da Computação - Parte 1

Reference	**ACM Body of Knowledge**	**ACM Discipline**	**All ITS Mod.**	**ITS Module**		
				T	**S**	**EKM**
(NAKHAL; BASHHAR, 2017)	Algorithms and Complexity	Basic Automata, Computability and Complexity; Advanced Automata Theory and Computability	No	Yes	No	No
(HARSLEY et al., 2016)	Algorithms and Complexity	Fundamental Data Structures and Algorithms	No	No	Yes	No
(ALBATISH; MOSA; ABU-NASER, 2018)	Architecture and Organization	Digital Logic and Digital Systems	No	Yes	No	No
(GALAFASSI et al., 2020)	Discrete Struct.	Proof Techniques	No	No	Yes	No
(ABUEL-REESH; ABU-NASER, 2018)	Information Ass. and Security	Cryptography	No	Yes	No	No
(ELREESH; ABU-NASER, 2019)	Information Ass. and Security	Cryptography	No	Yes	No	No
(MAHDI; ALHABBASH; NASER, 2016)	Information Ass. and Security	Threats and Attacks; Cryptography	No	No	No	No
(AL-HANJORI; SHAATH; ABU-NASER, 2017)	Networking and Communication	Introduction; Local Area Networks	No	No	No	No
(ALSHAWWA; AL-SHAWWA; ABU-NASER, 2019)	Networking and Communication	Introduction; Local Area Networks	No	Yes	No	No
(VERDÚ et al.,)	Networking and Communication	Introduction; Local Area Networks	Yes	Yes	Yes	Yes
(MAROUF; ABU-NASER, 2019)	Operating Syst.	Overview of Operating Systems	No	Yes	No	No
(HASANEIN; NASER, 2017)	Parallel and Dist. Computing	Cloud Computing	No	Yes	No	No
(OBERHAUSER, 2017)	Software Dev. Fundamentals	Develop. Methods	No	No	No	No
(HADDAD; NASER, 2017)	Software Dev. Fundamentals	Develop. Methods	No	Yes	No	No
(CARTER; BLANK, 2013)	Software Dev. Fundamentals	Develop. Methods	Yes	Yes	Yes	Yes
(PAASSEN; JENSEN; HAMMER, 2016)	Software Dev. Fundamentals	Fund. Programming Concepts	No	No	No	No
(AL-BASTAMI; NASER, 2017)	Software Dev. Fundamentals	Fund. Programming Concepts	No	Yes	No	No
(PRICE; DONG; LIPOVAC, 2017)	Software Dev. Fundamentals	Fund. Programming Concepts	No	No	No	No
(MOSA; ALBATISH; ABU-NASER, 2018)	Software Dev. Fundamentals	Fund. Programming Concepts	No	Yes	No	No

VAC, 2017; CARTER; BLANK, 2013; SHARMA; HARKISHAN, 2022; TROUSSAS et al., 2023; CHRYSAFIADI et al., 2023; CAO, 2023; HUANG et al., 2023) cujos domínios de aplicação dos ITS estão alinhados ao *ACM Body of Knowledge* descrito como *Software Development Fundamentals*.

Destes dez (10) trabalhos enquadrados no domínio de *Software Development Fundamentals*, oito trabalhos (FIGUEIREDO; GARCÍA-PEÑALVO, 2020; HOOSHYAR et al., 2018; PRICE; DONG; LIPOVAC, 2017; SHARMA; HARKISHAN, 2022; TROUSSAS et al., 2023; CHRYSAFIADI et al., 2023; CAO, 2023; HUANG et al., 2023) estão abordando o domínio de programação de computadores. O ITS apresentado pelo estudo de (FIGUEIREDO; GARCÍA-PEÑALVO, 2020), que foi construído como uma ferramenta de apoio ao

Tabela 2 – Trabalhos sobre ITS para Educação em Ciência da Computação - Parte 2

Reference	**ACM Body of Knowledge**	**ACM Discipline**	**All ITS Mod.**	**ITS Module**		
				T	**S**	**EKM**
(HOOSHYAR et al., 2018)	Software Dev. Fundamentals	Fund. Programming Concepts	No	Yes	No	Yes
(AL-SHAWWA; ALSHAWWA; ABU-NASER, 2019)	Software Dev. Fundamentals	Fund. Programming Concepts	No	Yes	No	No
(FIGUEIREDO; GARCÍA-PEÑALVO, 2020)	Software Dev. Fundamentals	Fund. Programming Concepts	No	No	Yes	No
(ALSHAIKH; TAMANG; RUS, 2021)	Software Dev. Fundamentals	Fund. Programming Concepts	No	Yes	No	Yes
(JEREMIC; JOVANOVIC; GASEVIC, 2009)	Software Eng.	Software Design	Yes	Yes	Yes	Yes
(JEREMIC; JOVANOVIĆ; GAŠEVIĆ, 2012)	Software Eng.	Software Design	Yes	Yes	Yes	Yes
(RAHMAN; ABDULLAH; ALIAS, 2016)	Software Eng.	Software Project Manag.	Yes	Yes	Yes	Yes
(FRANCISCO; SILVA, 2022a)	Software Eng.	Software Evolution	No	Yes	No	Yes
(LUBURIć et al., 2022a)	Software Eng.	Software Design	No	No	No	Yes
(LUBURIć et al., 2022b)	Software Eng.	Software Design	No	Yes	No	No
(SHARMA; HARKISHAN, 2022)	Software Dev. Fundamentals	Fund. Programming Concepts	No	Yes	No	Yes
(TROUSSAS et al., 2023)	Software Dev. Fundamentals	Fund. Programming Concepts	No	Yes	No	Yes
(CHRYSAFIADI et al., 2023)	Software Dev. Fundamentals	Fund. Programming Concepts	No	Yes	No	Yes
(CAO, 2023)	Software Dev. Fundamentals	Fund. Programming Concepts	No	Yes	No	No
(HUANG et al., 2023)	Software Dev. Fundamentals	Fund. Programming Concepts	No	No	No	Yes
(NKAMBOU et al.,)	Discrete Structures	Basic Logic	No	No	Yes	Yes

professor, é capaz de detectar a situação, necessidades e habilidades do estudante a partir de um modelo preditivo para contribuir com o trabalho do professor que precisa decidir as próximas etapas da instrução. Para isso, o estudo apresenta um modelo de estudante que mapeia informações como a curiosidade, perfeccionismo, iniciativa, o seu conhecimento quanto a pensamento computacional e programação, resultados de avaliações e a sua experiência com a gamificação. O ITS baseado em soluções proposto por (HOOSHYAR et al., 2018) busca melhorar estratégia de resolução de problemas de programação dos estudantes. Para isso, ele oferece soluções que auxiliam os estudantes com geração automática de fluxogramas que representam os algoritmos referentes aos exercícios disponibilizados pela ferramenta e o apoio à navegação dos tópicos e exercícios. O estudo sobre o ambiente de programação para novatos *iSnap* (PRICE; DONG; LIPOVAC, 2017) apresentou a incorporação da ideia de oferecer dicas durante a resolução de tarefas, oriunda dos ITS, em sua proposta. Este ambiente, que inclui uma interface baseada em *Scratch*, utilizou representações dos programas e suas edições para propor um mecanismo de geração de dicas sobre trechos de instruções em tempo de programação. O ITS *Trace Table Tutor* foi proposto para ensino-aprendizagem de programação utilizando rastreamento de código-fonte *Python* a partir da ideia de competências integrativas (HUANG et al., 2023). O trabalho apresenta que o EKM foi construído a partir das competências integrativas e

representadas como componentes de competências e suas combinações. A interface do ITS possui programa, caixa de dicas, tabela de rastreamento e texto para instrução. Um ITS Gamificado foi proposto para o ensino-aprendizagem de programação a partir de uma estratégia baseada em *Generative Pre-Training Transformer 3* (GPT-3) e agentes (CAO, 2023). O ITS Gamificado é baseado em histórias, oferece *feedback* adaptativo e personalizado, utiliza o GPT-3 para fornecer perguntas, respostas e explicar código-fonte ao sistema. Agentes inteligentes foram projetados para interagir, fornecer orientação e se responsabilizarem pelo progresso e *feedback* ao estudante e *chatbots* são utilizados para apoiar a interação do ITS com o estudante.

Os outros estudos encontrados para *software Development Fundamentals* abordam a compreensão de *software* (ALSHAIKH; TAMANG; RUS, 2021) e depuração de *software* (CARTER; BLANK, 2013). O estudo de (ALSHAIKH; TAMANG; RUS, 2021) apresentou um ITS socrático e sua ferramenta de autoria de conteúdo para o domínio de compreensão de *software*. Esse ITS socrático trabalha com a geração e compreensão automática de diálogos com estudantes sobre artefatos de código-fonte. O ITS para o ensino-aprendizagem de depuração de *software* (CARTER; BLANK, 2013) oferece suporte ao estudante na resolução no contexto da resolução de uma tarefa em específico neste domínio. Nessa proposta, os erros de sintaxe, tempo de execução e lógica formam os casos para o módulo de domínio e as tentativas de resolução de problemas formam o modelo do estudante.

Para o domínio de *Software Engineering* foram encontrados os ITS (RAHMAN; ABDULLAH; ALIAS, 2016; JEREMIC; JOVANOVIĆ; GAŠEVIĆ, 2012; JEREMIC; JOVANOVIC; GASEVIC, 2009; LUBURIć et al., 2022a; LUBURIć et al., 2022b; FRANCISCO; SILVA, 2022a). O ITS ABITS-FPM (RAHMAN; ABDULLAH; ALIAS, 2016) foi proposto para ensino-aprendizagem de Métricas para Pontos de Função. Este ITS, que fornece visualização, *feedback* imediato, recomendação, ajuda interativa e ajuda orientada, apresentou um modelo do estudante composto por fatos pessoais do estudante, níveis de aprendizagem e estilos de apresentação e resultados de avaliação, um modelo de domínio alinhado ao conhecimento sobre Métricas para Pontos de Função, que inclui o conteúdo teórico, questões práticas, dicas para as questões e informações de suporte relacionada ao desempenho prático do estudante. Um ITS foi proposto para o domínio de manutenibilidade de código-fonte a partir de uma descrição de EKM construída com os conceitos *knowledge components, instructional, and assessment items* disponibilizados pelo *KLI framework* (LUBURIć et al., 2022a). O trabalho (FRANCISCO; SILVA, 2022a) apresenta a aplicação do algoritmo *Q-Learning* para recomendação de DM de SM para o Módulo Tutor de um ITS para SM. Essa abordagem foi possível graças a um extenso estudo do currículo de SM e adaptação do mesmo no EKM do ITS para SM.

O ITS DEPTHS (JEREMIC; JOVANOVIĆ; GAŠEVIĆ, 2012; JEREMIC; JOVANOVIC; GASEVIC, 2009) foi proposto para o tópico de padrões de projeto de software. O

estudo de (JEREMIC; JOVANOVIC; GASEVIC, 2009) apresentou o uso de um grafo de dependências para modelar o domínio de padrões de projeto de software, propôs para o módulo tutor uma estratégia que envolve fatos, regras, *queries* e a produção de planos de conceitos e planos de aula e a realização de testes para avaliar o estudante que coletam dados de dificuldade, tempo e resultado. Entretanto, o estudo (JEREMIC; JOVANOVIĆ; GAŠEVIĆ, 2012) enfatizou o modelo do estudante no ITS DEPTHS com o uso de dados pessoais, dados de desempenho estáticos e dinâmicos e histórias de ensino e sua atualização.

No entanto, foi encontrado somente um trabalho para cada um dos demais domínios, i.e. *Algorithms and Complexity* (HARSLEY et al., 2016), *Discrete Structures* (GALAFASSI et al., 2020; NKAMBOU et al.,), *Networking and Communication* (VERDÚ et al.,) e *Parallel and Distributed Computing* (HASANEIN; NASER, 2017).

O ITS Colaborativo Collab-ChiQat Tutor foi proposto para o ensino-aprendizagem de algoritmos e estruturas de dados básicas (HARSLEY et al., 2016). Este ITS, além de incluir uma UI bastante intuitiva que integrou programação e visualização de estruturas de dados, apresentou um modelo do estudante que considera o comportamento individual e colaborativo do estudante durante o uso do sistema. Determinadas informações, como o histórico e o tempo de execução de ações, *feedback*, desfazer e refazer, tentativas, problemas resolvidos, erros de compilação e bônus dos pares de estudantes são utilizadas para o modelo do estudante.

O estudo sobre o ITS EvoLogic (GALAFASSI et al., 2020) abordou o ensino-aprendizagem da Dedução Natural na Lógica Proposicional. Eles apresentaram o *model tracing* como um recurso capaz de acompanhar os passos individuais de cada estudante visando oferecer *feedback* em tempo real e o modelo do estudante, que representa esses passos realizados pelo estudantes, a categorização da qualidade de seus passos e a sua linha de raciocínio durante a resolução de tarefas.

Um ITS para ensino-aprendizagem de lógica foi proposto com um Módulo do Estudante bayesiano (NKAMBOU et al.,). Uma rede bayesiana é utilizada no processo *Bayesian Knowledge Tracing* (BKT) para inferir as habilidades do estudante e uma rede neural foi treinada para o processo de rastreamento de conhecimento profundo. Essa proposta permite a generalização do conhecimento com poucas amostras e traz ótimos resultados na predição do conhecimento do estudante e contribuem com a inicialização do Módulo do Estudante.

A abordagem INTUITEL (VERDÚ et al.,), que inclui um ITS adaptável a *Learning Management System* (LMS) e realiza a recomendação de conteúdo, foi aplicada a um curso de Design de Redes de Computadores a partir de uma integração com o Moodle. Esta abordagem oferece recomendações e *feedback* não intrusivos sobre o melhor caminho de aprendizagem considerando perfil, progresso, contexto, estratégias pedagógicas e influências ambientais. Os professores precisam modelar o processo de aprendizagem no

INTUITEL com relação aos seus materiais didáticos e estratégias de aprendizagem e esta modelagem é manipulada pelo INTUITEL em uma abordagem baseada em ontologias. Apesar da abordagem ter sido aplicada no domínio de Design de Redes de Computadores e integrada ao LMS Moodle, ela pode ser aplicada a outros cursos e integrada a outros LMS.

A literatura de ITS para a Educação em Ciência da Computação apresenta trabalhos que descrevem Sistemas Tutores que, apesar de afirmarem que se são ITS, abordam as características dos ITS de maneira reducionista, e.g. sem apresentar técnicas computacionais como algoritmos de AI ou estruturas para representação dos modelos. Os trabalhos (MAROUF; ABU-NASER, 2019; MOSA; ALBATISH; ABU-NASER, 2018; ABUEL-REESH; ABU-NASER, 2018; ALBATISH; MOSA; ABU-NASER, 2018; HADDAD; NASER, 2017; AL-HANJORI; SHAATH; ABU-NASER, 2017; NAKHAL; BASHHAR, 2017; MAHDI; ALHABBASH; NASER, 2016), que basearam-se na ferramenta de autoria *Intelligent Tutoring System Builder* (ITSB), e os trabalhos (ALSHAWWA; AL-SHAWWA; ABU-NASER, 2019; AL-SHAWWA; ALSHAWWA; ABU-NASER, 2019; PAASSEN; JENSEN; HAMMER, 2016; OBERHAUSER, 2017) enquadram-se nesta perspectiva.

Nesse sentido, para esta categoria de Sistemas Tutores serão apresentados somente os trabalhos de (OBERHAUSER, 2017; PAASSEN; JENSEN; HAMMER, 2016) por apresentarem possibilidades futuras de contribuir com a construção de ITS para a Educação em Ciência da Computação.

Uma abordagem de recomendação, navegação e visualização 3D de código-fonte (OBERHAUSER, 2017) foi proposta para contribuir com a compreensão de *software* em uma perspectiva de processos cognitivos exploratórios, analíticos e descritivos. Este Sistema Tutor possui um serviço de recomendação baseado em um modelo teórico de compreensão de programas e em informações relacionadas a métrica *MethodRank* para código-fonte, processamento de filtragem, cálculos de distâncias e pontos de interesse no código-fonte.

A estratégia apresentada pelo estudo de (PAASSEN; JENSEN; HAMMER, 2016), que abordou o rastreamento da execução de programas, pode ser utilizada em ITS para o ensino-aprendizagem de programação de computadores. Nessa estratégia, os traços de execução dos programas são capturados e comparados com o algoritmo Distância de Edição para a obtenção da informação sobre a funcionalidade que o programa implementa independente de diferenças sintáticas. Esta estratégia pode possibilitar a um ITS na tarefa de identificação da estratégia de solução, identificação de soluções erradas e localização de erros em programas para oferecer um *feedback*.

2.2.1.1 Desafios e Oportunidades

Esta seção apresenta alguns desafios e oportunidades de pesquisa identificados nesta área. Os desafios e oportunidades de pesquisa encontrados foram categorizados em: baixo

uso de técnicas de AI; ITS não disponíveis publicamente; falta de *datasets* públicos; dificuldade de reprodução de *datasets* e baixa cobertura de tópicos da área. Apesar desta tese aplicar técnicas de ML na construção de ITS, a apresentação desses trabalhos correlatos sobre ITS para Educação em Ciência da Computação foi descrita utilizando o termo AI por ter sido utilizado ao longo das pesquisas conforme verificado na literatura.

A Tabela 3 apresenta os trabalhos que abordaram AI em ITS para Educação em Ciência da Computação. Há baixo uso de técnicas de AI em ITS para Educação em Ciência da Computação. Entre os artigos incluídos nesta revisão, apenas 14 abordaram algumas técnicas de AI, o equivalente a 40%.

Tabela 3 – Trabalhos que abordaram AI em ITS para Educação em Ciência da Computação

Reference	**AI Based technique**	**Algorithm**	**Dataset Available**	**Dataset Reproducible**	**ITS Available**
(HASANEIN; NASER, 2017)	Agent-based Architecture	No	No	No	No
(RAHMAN; ABDULLAH; ALIAS, 2016)	Agent-based Architecture	No	No	No	No
(HOOSHYAR et al., 2018)	Bayesian networks; Multi-agent system	Superficially presented	No	No	No
(CARTER; BLANK, 2013)	Case-based reasoning	No	No	Yes	No
(JEREMIC; JOVANOVIC; GASEVIC, 2009)	Fuzzy sets	No	No	No	No
(JEREMIC; JOVANOVIĆ; GAŠEVIĆ, 2012)	Fuzzy rules	Yes	No	Yes	No
(GALAFASSI et al., 2020)	Genetic Algorithm	Yes	No	Yes	No
(FIGUEIREDO; GARCÍA-PEÑALVO, 2020)	Neural network	Yes	No	Yes	No
(VERDÚ et al.,)	Ontology-based reasoning	No	No	No	Yes
(FRANCISCO; SILVA, 2022a)	Q-Learning	Yes	No	No	No
(SHARMA; HARKISHAN, 2022)	Case-based reasoning; k-nearest neighbor's	Yes	No	No	No
(CHRYSAFIADI et al., 2023)	Fuzzy	Yes	Yes	No	No
(NKAMBOU et al.,)	Bayesian networks Neural networks	Yes	No	Yes	Yes
(CAO, 2023)	Gen. Pre-Training Transf. 3 (GPT-3)	No	No	No	No

Além dos poucos artigos que abordavam técnicas de AI, notamos que 35,7% dos artigos que utilizavam AI nem sequer apresentavam detalhes dos algoritmos. Os trabalhos (HASANEIN; NASER, 2017; RAHMAN; ABDULLAH; ALIAS, 2016), que utilizavam arquitetura baseada em Agentes, nem sequer mencionaram quais algoritmos foram utilizados em seus agentes. As técnicas de AI estão impactando diversas áreas e pode-se afirmar claramente que existem muitas oportunidades diferentes para explorar seu uso em ITS para o Ensino de Ciência da Computação, melhorando e fortalecendo então esta área.

O Ensino de Ciência da Computação possui particularidades geográficas, dada a existência conteúdos relevantes considerando o mercado de IT para cada localidade, o que exige investigar os efeitos que esses ITS e suas funcionalidades baseadas em AI trazem

para o ensino em diferentes contextos. Apenas dois dos ITS baseados em AI estavam disponíveis publicamente para uso (VERDÚ et al., ; CAO, 2023). Este fato dificulta a popularização dos ITS em cenários educacionais reais, parcerias entre pesquisadores e avanços nas pesquisas. Neste sentido, é positivo que os ITS possam estar à disposição de outros investigadores e docentes, o que poderá melhorar a investigação e o desenvolvimento desta área. Pelo menos, a comunidade científica poderia ter pelo menos esse objetivo para novos sistemas e técnicas.

Apenas o trabalho (CHRYSAFIADI et al., 2023), desses estudos com os ITS que abordaram AI, apresentou os conjuntos de dados utilizados nos algoritmos de AI. Portanto, os pesquisadores precisam reconstruir cenários para coletar dados que possibilitem analisar o resultado de novas técnicas de AI ou mesmo melhorias em métodos existentes. Esta falta de dados públicos dificulta a comparação entre diferentes técnicas de AI. Uma oportunidade bem-vinda aqui é capturar e tornar públicos conjuntos de dados obtidos de cenários reais de uso de ITS para Educação em Ciência da Computação.

Outro desafio é reproduzir conjuntos de dados utilizando ITS apresentados na literatura. No conjunto de estudos sobre ITS para este domínio utilizando AI, apenas 35,7% destes trabalhos descreveram dados ou textos que poderiam auxiliar nesta reprodução. Além disso, existe a dificuldade de acessar cenários educacionais e modificar o processo de ensino-aprendizagem para realizar essa coleta de dados a partir da utilização dos ITS pelos estudantes. A reprodução de cenários poderia proporcionar uma oportunidade sistemática para comparar diferentes ITS, indicando a direcção certa para melhorias e encontrando lições aprendidas que a investigação seguinte deveria evitar.

Os ITS pesquisados cobrem apenas 55,5% (10 de 18) dos diversos tópicos do *ACM Body of Knowledge*. Além disso, o grupo de disciplinas *ACM Body of Knowledge* denominado *Software Development Fundamentals* ocupa 45,7% dessas pesquisas. Este fato traz a oportunidade de expandir o design, desenvolvimento e implantação em outros grupos de disciplinas da área de Ciência da Computação.

2.2.2 ITS para SM

Esta revisão da literatura sobre ITS para SM. O resultado dessa revisão foi resumido na Tabela 4. A revisão analisou se o trabalho abordou os Módulos do Estudante e Tutor, as técnicas de ML detalhadas nestes módulos e também se o trabalho apresenta como ocorre a integração entre esses módulos.

O ITS para o ensino-aprendizagem de depuração de *software* (CARTER; BLANK, 2013) utilizou *Case-Based Reasoning* (CBR). Eles basearam-se na ideia de que a depuração realizada pelo programador é inerentemente baseada em casos. Nessa proposta, o módulo de domínio é composto de casos que representam erros de sintaxe, tempo de

Tabela 4 – Trabalhos sobre ITS para SM

WORK	**STM**[a]	**ML in STM**	**TM**[b]	**ML in TM**	**I**[c]	**ML technique**	**SM topics** [d]
(LUBURIć et al., 2022a)	YES	NO	YES	NO	NO	-	REF; MAN.
(LUBURIć et al., 2022b)	NO	NO	YES	NO	NO	-	REF; MAN.
(CARTER; BLANK, 2013)	YES	NO	YES	YES	NO	CBR	DEP.
(ALSHAIKH; TAMANG; RUS, 2021)	NO	NO	YES	NO	NO	-	COMP.
(FRANCISCO; SILVA, 2022a)	NO	NO	YES	YES	NO	*Q-Learning*	COMP; PRAT; TEST; CONS.

[a] STM: Student Module

[b] TM: Tutor Module

[c] I: Integration between Student Module and Tutor Module

[d] SM topics: Refactoring (RE), Source Code Maintainability (MA), Debugging (DE), *software* Understanding (COMP), SM Practice (PRA), Tests for MS (TST) and Comprehension of Concepts in SM (CON)

execução e lógica. O Módulo Tutor recomenda exercícios e sugestões de correção utilizando o CBR. Essas sugestões representam a correção mais adequada para um estudante considerando o erro e o número de tentativas em uma perspectiva verbal e visual. O modelo do estudante é construído quando o estudante tenta resolver problemas e o módulo tutor busca casos. O artigo não relata a contribuição do Módulo do Estudante para o Módulo Tutor e também não menciona se alguma técnica de ML foi utilizada no Módulo do Estudante.

A ferramenta de autoria para ITS socrático aplicado a compreensão de *software* (ALSHAIKH; TAMANG; RUS, 2021) foi construída para a geração de diálogos sobre código-fonte. Essa ferramenta de autoria permite ao Módulo Tutor dialogar com o estudante, por exemplo sobre conceitos de programação e compreensão da dinâmica de processamento, no contexto do código-fonte a partir de uma abordagem que envolveu templates de diálogo e ferramentas da linguagem Java, como análise estática, simulador dinâmico de código, *abstract syntax tree*, *Java Debugger Interface* (JDI). A similaridade semântica foi usada para analisar as respostas dos estudantes. Esse trabalho não descreve o Módulo do Estudante.

Um ITS para manutenibilidade de código-fonte (LUBURIć et al., 2022a) foi proposto com uma estratégia que utilizou análise e refatoração de código limpo. O trabalho apresentou um Módulo de Domínio baseado em modelos de conhecimento e habilidade para manutenção de código e um Módulo de Estudante. O Módulo do Estudante se trata de um modelo de sobreposição de domínio do estudante. O artigo menciona que o sistema captura o progresso e as lacunas de habilidades do estudante por meio da sua interação com componente de conhecimento e permite realizar a adaptação da instrução. No entanto, o trabalho não mencionou como essa adaptação se relaciona com os componentes de conhecimento existentes para o domínio, o que seria esperado caso o trabalho descrevesse o Módulo Tutor. O trabalho também não detalha a implementação do Analisador de Manutenibilidade.

O artigo sobre Clean CaDET ITS (LUBURIć et al., 2022b), proposto para manuteni-

bilidade de código-fonte em uma perspectiva de refatoração de código-limpo, apresenta a existência de um modelo de conteúdo, estudante, progresso e instrução. O Módulo Tutor é baseado na ideia de modelo de progresso e instrução. O estudante resolve um exercício de refatoração, o sistema o corrige com base na manutenibilidade do código-fonte, e ocorre a recomendação da adaptação ao estudante a partir da confrontação do modelo do estudante ao modelo de instrução. O artigo não apresenta uso de algoritmos de ML e também não detalha o Modelo do Estudante.

A pesquisa (FRANCISCO; SILVA, 2022a) aplica o RL, por meio do algoritmo *Q-Learning*, na funcionalidade de recomendação de conteúdo de um Sistema Tutor Inteligente para SM. Esse trabalho apresenta uma modelagem do domínio de SM, descrita como EKM e construída a partir do currículo de SM.

Os ITS trazem certos desafios (ALSHAIKH; HEWAHI, 2021) relacionados à representação do estado do conhecimento do estudante, à coleta de dados para sistemas inteligentes e ao seu gerenciamento considerando os respectivos modelos de ML. A coleta de dados envolve a dificuldade de trabalhar com cenários de uso real. O gerenciamento do ITS necessita de monitoramento na a execução das técnicas de ML, o que implica em treinar os profissionais que farão essa gestão no ambiente educacional, além de lidar com os custos com a infraestrutura tecnológica (ROLL; WYLIE, 2016).

A revisão da literatura sobre ITS para Educação em Ciência da Computação (FRANCISCO; SILVA, 2022b) contribui para a compreensão dessas dificuldades. Essa revisão apresentou trabalhos que usaram ML no desenvolvimento dos ITS e apontou que 55.5% deses trabalhos não detalharam como os algoritmos de ML foram modelados para o problema, nenhum dos trabalhos disponibilizaram os respectivos *Datasets* e somente 44.4% dos trabalhos apresentaram como reproduzir os *Datasets* em outros cenários.

Conforme a Tabela 4, nenhum trabalho abordou ML no Módulo do Estudante, somente dois (2) dos cinco (5) trabalhos abordou ML no Módulo Tutor e nenhum deles abordou a integração entre esses dois módulos dos ITS. Além disso, somente o trabalho (FRANCISCO; SILVA, 2022a), que foi publicado como um dos resultados deste processo de doutoramento, abordou o domínio educacional de SM de maneira ampla. Essas lacunas são abordadas nesta pesquisa sobre ITS para SM.

Capítulo 3

ITS para SM

Este capítulo propõe um ITS para SM. Conforme apresentado por (ALKHATLAN; KALITA,), esta pesquisa realizou um planejamento envolvendo o conteúdo, as funcionalidades e a arquitetura do ITS para a construção do ITS para SM. O ITS para SM precisa possibilitar ao estudante estudar os conteúdos de SM, e.g. resolvendo problemas. Para isso, o projetista precisa decidir quais as atividades o ITS abordará, planejar quais as funcionalidades que serão executadas e que vão interferir no comportamento da UI, e planejar uma arquitetura que suporte essa execução.

Em um ITS é importante que o Módulo Tutor ajude o estudante a se concentrar em apenas uma questão por vez e para isso o ITS pode acompanhar cada passo do estudante na resolução de uma tarefa para descobrir quais os aspectos do conhecimento precisam ser mais reforçados (HU, 2019). Segundo este mesmo autor, o Módulo Tutor percebe a interação do estudante com a UI, se comunica com o EKM para receber o *feedback* sobre a resolução de um exercício e o devolver à UI, que pode mostrar informações para orientar o estudante no processo de estudo ou simplesmente mostrar a resposta correta de um exercício, e também se comunica com o Módulo do Estudante para obter uma informação que represente o estado de conhecimento do estudante no domínio abordado pelo ITS de modo que essa informação possa ser usada no planejamento de ações pedagógicas.

Esta tese aborda os Módulos do Estudante, Tutor e EKM, enfatizando o uso de ML no ITS. O EKM, que foi proposto a partir de um estudo do conteúdo de SM e de como esse conteúdo se relaciona com DM de SM, traz a possibilidade de ser explorado com relação à UI na perspectiva de requisitos do ambiente educacional. A Figura 6 apresenta a estratégia para construção do ITS para SM adotada nesta Tese, enfatizando ML nos Módulos Tutor e do Estudante, o conteúdo de SM no EKM e como esses módulos colaboram entre si para prover o funcionamento do ITS. Como esta tese enfatiza os modelos de ML no contexto do ITS para SM, o tema de UI não foi abordado.

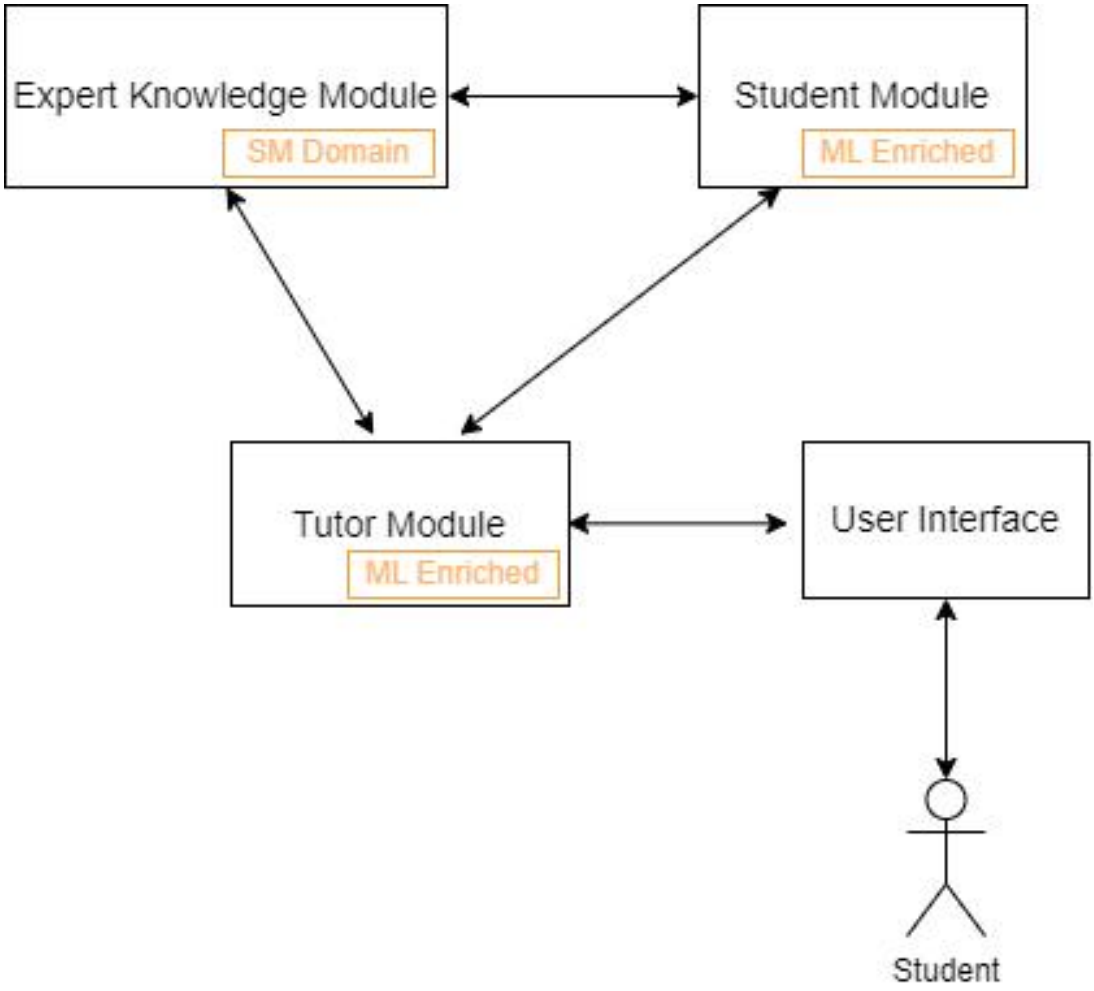

Figura 6 – Estratégia para construção do ITS para SM

A construção do ITS para SM depende de seja realizada uma definição adequada do conteúdo de SM que será abordado. Considerar o uso desse ITS proposto em uma sala de aula no contexto do ensino-aprendizagem de SM possibilitou apresentar alguns direcionamentos relacionados aos estudantes e o conhecimento relacionado que serão apresentados e ajudaram a direcionar a maneira sobre como o estudo envolvendo o conteúdo de SM foi realizado.

- ❏ A necessidade de formar profissionais para trabalharem com SM em uma perspectiva prática (HECKMAN; STOLEE; PARNIN, 2018) traz a necessidade de planejar como o conteúdo de SM pode ser abordado em um ITS de forma que possa ser trabalhado com estudantes, no ambiente acadêmico, e com profissionais de IT, no ambiente organizacional através de treinamento técnico em SM.

- ❏ Considerando a aplicação de um ITS para SM no ambiente acadêmico, estudantes que foram aprovados nas disciplinas de Programação Orientada a Objetos e Modelagem de *software* Orientada a Objetos podem começar a resolver problemas de SM no ITS que estejam relacionados a esses conteúdos. Níveis mais avançados vão

exigindo mais competências, que são adquiridas com o tempo. SM pode abordar projetos de *software* de diferentes níveis de dificuldade.

- ❏ Abordar um ensino-aprendizagem de SM que enfatize a prática é importante para a demanda de profissionais de SM (HECKMAN; STOLEE; PARNIN, 2018), o que pode ser verificado através dos interesses de pesquisas atuais (RAABE; GOMES, 2018; TONHÃO; SOUZA; PRATES, 2021). Por exemplo, metodologias ativas de educação foram propostas para domínios educacionais relacionados, como a cultura *Maker* na Educação (RAABE; GOMES, 2018) e a abordagem prática baseada em projetos e gamificação de ensino de Engenharia de *Software* (TONHÃO; SOUZA; PRATES, 2021). Um ITS contribui com a operacionalização desses cenários educacionais complexos devido á sua capacidade de adaptação e automatização no processo ensino-aprendizagem (ALKHATLAN; KALITA,).

Este capítulo está organizado da seguinte forma. A seção 3.1 apresenta o Conteúdo de SM, a seção 3.2 apresenta um conjunto de funcionalidades identificadas como importantes para este ITS, e a seção 3.3 apresenta a arquitetura e os componentes propostos do ITS para SM.

3.1 Conteúdo de SM

A definição do conteúdo abordado pelo ITS é essencial para a construção do EKM, que por sua vez contribui para a representação do estudante no Módulo do Estudante (ALKHATLAN; KALITA,). Nesse sentido, essa definição do conteúdo é importante para a construção de um ITS. Por isso, foi realizado levantamento dos temas abordados no ensino de SM que contou com a análise de disciplinas que envolvem SM em Universidades, referências da SBC (ZORZO et al., 2017) e ACM (ACM Computing Curricula Task Force, 2020).

A busca de Fichas de Disciplinas que contenham o conteúdo de SM em cursos de graduação da área de Computação das Universidades Federais Brasileiras foi realizada visando analisar como o ensino-aprendizagem de SM ocorre em grande parte do ensino público superior brasileiro. Os seguintes cursos apresentam este conteúdo em suas Fichas de Disciplinas: Bacharelado em Ciência da Computação, Bacharelado em Sistemas de Informação, Bacharelado em Engenharia da Computação, Bacharelado em Tecnologia da Informação, Bacharelado em Ciência de Dados e Inteligência Artificial, Bacharelado em Engenharia de *Software*, Bacharelado em Engenharia de Computação e Licenciatura em Computação. Foram encontradas 85 Fichas de Disciplinas.

Visando melhorar a compreensão do conteúdo de SM, foram construídos uma lista dos temas identificados no levantamento do currículo de SM e um detalhamento desses temas

a partir de atividades propostas possíveis de serem abordadas em um ITS para SM. Esta lista detalhada de temas e atividades propostas em SM está disponível nas Tabelas 5 e 6.

Conforme as Tabelas 5 e 6, trata-se de 26 temas de SM e 63 atividades propostas. Isso mostra que SM é um assunto bastante amplo, com uma diversidade de habilidades e conteúdo, o que o torna um domínio desafiador para a construção de um ITS.

A Tabela 7 apresenta as categorias abstraídas a partir dos temas em SM apresentados nas Tabelas 5 e 6. Essas categorias foram identificadas para facilitar a organização do conteúdo de SM e contribuir com o planejamento do ITS para SM, considerando a influência do conteúdo na construção do EKM em um ITS (ALKHATLAN; KALITA,).

Conforme pode ser verificado, os temas em SM foram abstraídos para quatro categorias, que são: A - Compreensão de *Software*; B - Prática em SM; C - Testes em SM; e D - Compreensão de Conceitos. A categoria D se subdivide em: D1 - Processo de SM; e D2 - Manutenibilidade de *Software*.

A categoria "A - Compreensão de *Software*" foi composta do conteúdo que indica ao estudante a necessidade de ler e compreender artefatos de *software*. A categoria "B - Prática em SM" foi composta pelo conteúdo que implica na necessidade do estudante realizar tarefas práticas no contexto de SM, podendo utilizar ferramentas, e.g. Ambiente de Desenvolvimento de *Software* e Sistemas de Controle de Versão, e executar tarefas, e.g. estimar o esforço em SM. A categoria "C - Testes em SM" ocupa-se dos testes em SM. A categoria "D1 - Processo de SM" envolve o conteúdo conceitual sobre processo de SM e a categoria "D2 - Manutenibilidade de *Software*" envolve também o conteúdo conceitual sobre Manutenibilidade de *Software*.

A Tabela 8 apresenta como as atividades propostas para os temas em SM podem ser abordadas como DM em um ITS. Esta tabela descreve para cada categoria, quais os tipos de DM são possíveis e também as atividades propostas relacionadas a cada DM de acordo com as Tabelas 5 e 6. Percebe-se uma grande quantidade de tipos de DM para SM que envolvem a prática em SM e também um grande número de atividades propostas no âmbito conceitual em SM. Os tipos de DM (A1, A2 e A3_1) foram propostos a partir da estruturação de atividades de compreensão de programas (IZU et al., 2019), que apresenta uma subdivisão da compreensão em *read*, *trace* e *summarize*. O tipo de DM A2 é apresentado na pesquisa de (ERICSON et al., 2022).

Os DM com atividades práticas em SM são mais diversas, e.g. podendo necessitar do uso de ferramentas, enquanto as atividades conceituais em SM podem ser tratadas com uma estrutura comum, e.g. a sequência em que o conceito é apresentado e uma pergunta objetiva com quatro alternativas afere a compreensão do mesmo.

Essa estruturação do Conteúdo de SM foi modelada a partir das classes apresentadas na Figura 7. As quatro categorias são representadas representam os atributos da classe *DimensionKnowledgeSM*, que representa quantitativamente a dimensão de conhecimento de SM necessária como capacidade (classe *Capacity*) para que um estudante seja capaz de

Tabela 5 – Temas em SM e Atividades Propostas - Parte 1

COD	Theme / Possible Activities
1	**Software Maintenance Process**
1.1	Present concepts to the student about SM
1.2	Conceptually question the student about the SM process
2	**Reverse Engineering**
2.1	Introduce the concept of Reverse Engineering
2.2	Introduce Reverse Engineering tools
2.3	Practical activities involving Reverse Engineering and SM
3	Reengineering
3.1	Present the concept of Reengineering
3.2	Present Reengineering tools
3.3	Practical activities involving Reengineering and SM
4	**Migration**
4.1	Present the relationship between Migration and SM
4.2	Practical activity involving Migration and SM
5	**Configuration and Change Management**
5.1	Introduce the concept of Configuration and Change Management
5.2	Present tools for Configuration and Change Management
5.3	Practical activities involving tools for Configuration and Change Management
6	**Maintenance Cost**
6.1	Present a reflection on the Cost of Maintenance
6.2	Present conceptual issues involving the Cost of Maintenance

COD	Theme / Possible Activities
7	**Maintainability and Metrics for Maintainability**
7.1	Introduce the concept of *software* maintainability
7.2	Present metrics that help analyze *software* maintainability
7.3	Question conceptually about Maintainability and Metrics for Maintainability
7.4	Questions related to practical cases involving the analysis of maintainability metrics
8	**Effort estimation**
8.1	Introduce the concept of Effort Estimation in SM
8.2	Question about the concept of Effort Estimation in SM
8.3	Practical activities on effort estimation in SM
9	**Software (Re)Documentation**
9.1	Present a reflection on the importance of *software* (Re)Documentation for SM
9.2	Present conceptual questions about *software* (Re)Documentation and SM
10	**Types of *software* Maintenance**
10.1	Present the types of SM (defect correction, environmental adaptation, addition of functionality, refactoring)
10.2	Present conceptual questions about the types of SM
11	**Relationship between SM Types**
11.1	Present conceptual questions about the relationship between types of SM
12	**Maintenance Problems**
12.1	Present conceptually about the Problems in SM
12.2	Question conceptually about Problems in SM

Tabela 6 – Temas em SM e Atividades Propostas - Parte 2

COD	Theme / Possible Activities
13	**Software Understanding**
13.1	Understanding the execution result (source code)
13.2	Order the program (parson puzzles) (DURAN et al., 2020)
13.3	Answer summary (source code)
13.4	Answer summary (UML models)
13.5	Answer about processing details of a given software
13.6	Understand the software before and/or after modifications
14	**Reuse**
14.1	Conceptually present the relationship between Reuse and SM
14.2	Question conceptually about Reuse and SM
14.3	Practical activities on Reuse and SM
15	**Refactoring**
15.1	Conceptually present refactoring
15.2	Practical refactoring activities
16	**Mining Software Repositories**
16.1	Present the concept of Software Repository Mining and its importance for SM
16.2	Present Software Repository Mining tools for use in SM
16.3	Present practical activities involving Mining Software Repositories and SM
17	**Defect prediction**
17.1	Present the relationship between defect prediction and SM
17.2	Practical activities on Defect Prediction
18	**Impact Analysis**
18.1	Present conceptually about Impact Analysis of changes in SM
18.2	Practical activities on Impact Analysis of changes in SM

COD	Theme / Possible Activities
19	**Debugging**
19.1	Introduce about the use of debugging in SM
19.2	Practical activities on debugging in SM
20	**Software Evolution**
20.1	Present the concept of Software Evolution and its relationship with SM
20.2	Conceptual activities on Software Evolution and SM
21	**Integration and Continuous Delivery**
21.1	Conceptually present Integration and Continuous Delivery and its relationship with SM
21.2	Question conceptually about Integration and Continuous Delivery and its relationship with SM
22	**Laws of evolution**
22.1	Present the Software Evolution Laws and their relationship with SM
22.2	Conceptual questions about Software Evolution Laws and their relationship with SM
23	**Tests for SM**
23.1	Present software that has gone through SM and ask the student to test it to ensure that a change request was correctly implemented
24	**Addresses the practice of SM**
24.1	Practical SM activity of the Defect Correction type
24.2	Practical SM activity of the Environmental Adaptation type
24.3	Practical SM activity of the Functionality Addition type
24.4	Practical SM activity of the Refactoring type
25	**Addresses SM in a summarized way**
25.1	Conceptual questions that present discussions involving SM
26	**Legacy Systems**
26.1	Present conceptually about Legacy Systems
26.2	Conceptual questions about Legacy Systems

Tabela 7 – Categorias identificadas a partir dos Temas em SM

Category	**SM Themes**
A - Software Understanding	Software Understanding (13)
B - SM Practice	Reverse Engineering (2), Reengineering (3), Migration (4), Configuration and Change Management (5), Maintainability and Maintainability Metrics (7), Effort Estimation (8), Reuse (14), Refactoring (15), Software Repository Mining (16), Defect Prediction (17), Impact Analysis (18), Debugging (19), Address SM practice (24)
C - Tests in SM	Tests for SM (23)
D - Concept Understanding (D1 - SM Process)	Software Maintenance Process (1), Reverse Engineering (2), Reengineering (3), Migration (4), Configuration and Change Management (5), Maintenance Cost (6), Effort Estimation (8), Types of Software Maintenance (10), Relationship between SM Types (11), Refactoring (15), Software Repository Mining (16), Defect Prediction (17), Debugging (19), Software Evolution (20), Integration and Continuous Delivery (21), Address SM briefly (25)
D - Concept Understanding (D2 - Software Maintainability)	Maintainability and Maintainability Metrics (7), Effort Estimation (8), Software (Re)Documentation (9), Maintenance Problems (12), Reuse (14), Impact Analysis (18), Laws of evolution (22), Legacy Systems (26)

resolver um DM de SM. A classe DM representa o DM de SM, incluindo seu tipo de DM, a capacidade demandada, a pontuação que ele disponibiliza, o conteúdo em *Hypertext Markup Language* (HTML) e a resposta correta. A classe *DMType* representa os tipos de DM para SM conforme a Tabela 8. A classe *DMCategory* relaciona cada tipo de DM a uma categoria de DM, o que permite a um DM de SM estar relacionado a um tipo e a uma categoria de DM de SM.

Essas informações apresentadas referentes ao conteúdo de SM são subsídios para a proposta de um ITS para SM, cuja arquitetura e seus módulos são apresentados nas próximas seções.

3.2 Requisitos Funcionais

Esta seção apresenta os Requisitos Funcionais para o ITS para SM. Existem requisitos não-funcionais importantes para o ITS, como a Experiência do Usuário, desempenho, portabilidade e manutenibilidade, que foram identificados como importantes mas não são tratados nesta tese.

A Experiência do Usuário possui uma forte relação com a UI, o desempenho é importante devido à questão do custo computacional relacionado à ML em ITS (ROLL; WYLIE, 2016), e a portabilidade é importante devido necessidade de implantação desses sistemas em ambientes reais e a manutenibilidade contribui com a continuidade de pesquisa e desenvolvimento relacionado à construção de ITS.

Esta abordagem, adotada neste trabalho, decorre do fato que o foco da arquitetura está

Tabela 8 – Tipos de DM relacionados às categorias e Atividades Propostas em SM

Category	**Type of Didactic Material for Software Maintenance**	**Associated Proposed Activities**
A - Software Understanding	A1 - Understand the result of execution (source code)	13.1, 13.5, 13.6
	A2 - Order the program (parson puzzles) (DURAN et al., 2020)	13.2
	A3_1 - Reply to abstract (source code)	13.3
	A3_2 - Reply to abstract (UML models)	13.4
B - SM Practice	B1 - Defect Correction SM type	24.1
	B2 - Environmental Adaptation SM type	24.2
	B3 - Add Functionality SM type	24.3
	B4 - Refactoring	15.2, 24.4
	B5 - Practical Activity on Reuse and SM	14.3
	B6 - Practical Activity on Reverse Engineering and SM	2.3
	B7 - Practical Activity on Reengineering and SM	3.3
	B8 - Practical Activity on Migration and SM	4.2
	B9 - Practical Activity on Tools for Configuration and Change Management	5.3
	B10 - Questions Related to Practical Cases involving the Analysis of Maintainability Metrics	7.4
	B11 - Practical Activity on Effort Estimation in SM	8.3
	B12 - Practical Activity on Mining Software Repositories and SM	16.3
	B13 - Practical Activity on Defect Prediction	17.2
	B14 - Practical Activity on Impact Analysis of changes in SM	18.2
	B15 - Practical Activity on debugging in SM	19.2
C - Tests in SM	C1 - Tests	23.1
D - Concept Understanding	D1 - SM Process	1.1, 1.2, 2.1, 2.2, 3.1, 3.2, 4.1, 5.1, 5.2, 6.1, 6.2, 8.2, 10.1, 10.2, 11.1, 15.1, 16.1, 16.2, 17.1, 19.1, 20.1, 20.2, 21.1, 21.2, 25.1
D - Concept Understanding	D2 - Software Maintainability	7.1, 7.2, 7.3, 8.1, 9.1, 9.2, 12.1, 12.2, 14.1, 14.2, 18.1, 22.1, 22.2, 26.1, 26.2

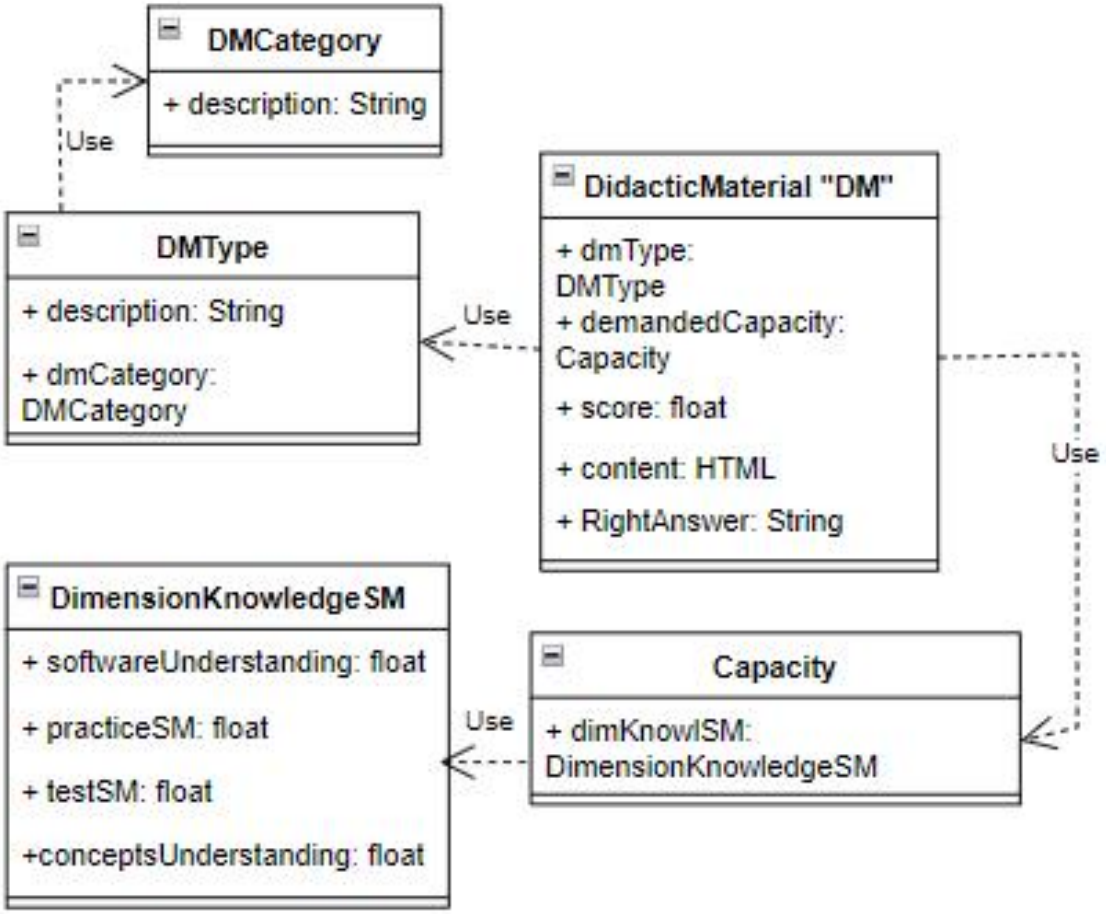

Figura 7 – Classes para acomodar o conteúdo de SM

nos requisitos funcionais e na hipótese de que ML pode contribuir com a funcionalidade do ITS.

A Figura 8 apresenta os Requisitos Funcionais importantes para o ITS para SM através de *Use Case* (UC). O estudante precisa se cadastrar no sistema, resolver um teste da dimensão de conhecimento de SM caso o ITS não tenha informações sobre sua capacidade em SM, resolver DM de SM recomendado de forma adaptativa pelo ITS, obter suporte na sua interação com DM de SM e obter *feedback* na sua interação com o processo de estudo de SM. O professor (ou o instrutor caso o ITS seja usado em uma organização) precisa gerenciar turmas de estudantes de SM e objetivos de aprendizagem, gerenciar o conjunto de DM de SM, criar um teste da dimensão de conhecimento de SM e analisar o processo ensino-aprendizagem de SM.

Esta seção apresenta as Histórias de Usuário, de acordo com o formato proposto pelo *Scrum* (AMNA; POELS, 2022), para cada UC apresentado na Figura 8. As Tabelas 9, 10, 11, 12 e 13 apresentam as Histórias de Usuário do estudante de SM e as Tabelas 14, 15, 16 e 17 descrevem as Histórias de Usuário do professor ou instrutor.

Esta Tese aborda o Requisito Funcional referente ao UC-C, que visa permitir ao es-

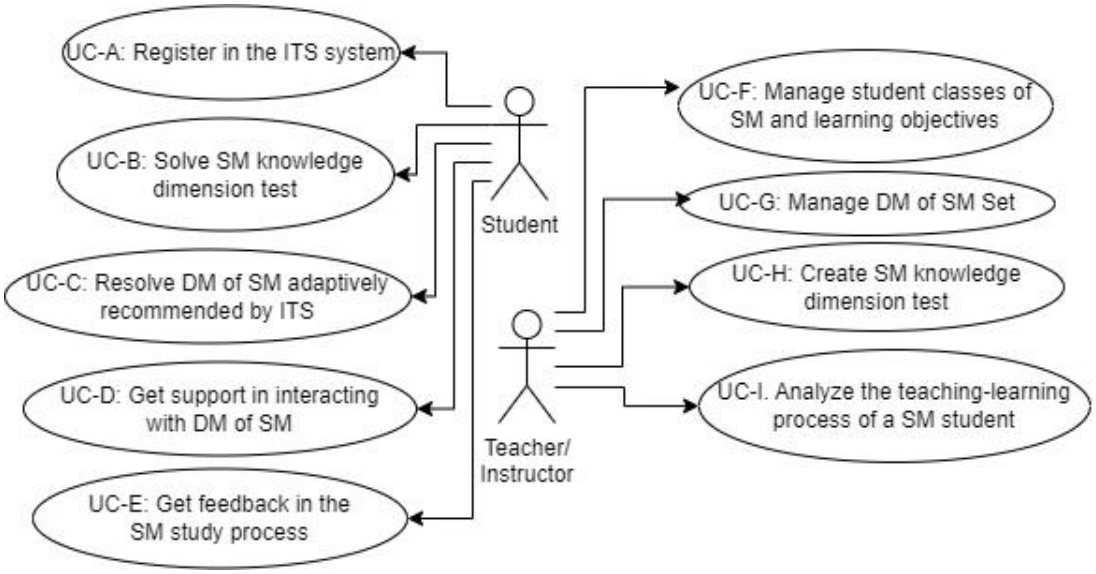

Figura 8 – Diagrama de UC do ITS para SM

Use Case	UC-A
As a	student
I want	register in the ITS system
So that	use ITS to solve DM of SM and improve my knowledge about SM
Acceptance criteria:	be able to access the ITS

Tabela 9 – História de Usuário (UC-A) - Registrar o estudante no ITS para SM

Use Case	UC-B
As a	student
I want	solve SM knowledge dimension test
So that	for ITS to have information about my SM capabilities and work adaptively to my SM knowledge
Acceptance criteria:	generate information about my SM capabilities

Tabela 10 – História de Usuário (UC-B) - Resolver teste da dimensão de conhecimento de SM

Use Case	UC-C
As a	student
I want	resolve DM of SM adaptively recommended by ITS
So that	improve my SM capabilities
Acceptance criteria:	improve hits in solving existing tasks in DM of SM

Tabela 11 – História de Usuário (UC-C) - Resolver DM de SM recomendado adaptativamente pelo ITS

Use Case	UC-D
As a	student
I want	get support in interacting with DM of SM
So that	have my doubts resolved while solving a task in a DM of SM
Acceptance criteria:	resolve a specific DM of SM correctly and more quickly

Tabela 12 – História de Usuário (UC-D) - Obter suporte na interação com o DM de SM

Use Case	UC-E
As a	student
I want	get feedback in the SM study process
So that	understand which part of the SM content I need to improve
Acceptance criteria:	obtain information about success/error and how to learn if I made a mistake

Tabela 13 – História de Usuário (UC-E) - Obter *feedback* na interação com o processo de estudo de SM

Use Case	UC-F
As a	teacher / instructor
I want	manage student classes of SM and learning objectives
So that	relate SM learning objectives to the DM of SM for specific student classes
Acceptance criteria:	adequate pedagogical planning involving groups of students and DM of SM

Tabela 14 – História de Usuário (UC-F) - Gerenciar turmas de estudantes de SM e os objetivos de aprendizagem

Use Case	UC-G
As a	teacher / instructor
I want	manage DM of SM Set
So that	register, edit and delete DM of SM
Acceptance criteria:	enable an adequate DM of SM set in ITS

Tabela 15 – História de Usuário (UC-G) - Gerenciar o conjunto de DM de SM

Use Case	UC-H
As a	teacher / instructor
I want	create SM knowledge dimension test
So that	enable ITS to have information on students capacity of SM and work adaptively
Acceptance criteria:	a test that generates information about SM knowledge dimension of students

Tabela 16 – História de Usuário (UC-H) - Criar um teste da dimensão de conhecimento de SM

Use Case	UC-I
As a	teacher / instructor
I want	analyze the teaching-learning process of a SM student
So that	know how the student's SM knowledge is at the current moment and a history of its evolution
Acceptance criteria:	a report on student knowledge in the context of using ITS to SM

Tabela 17 – História de Usuário (UC-I) - Analisar o processo ensino-aprendizagem de SM

tudante resolver DM de SM através da recomendação adaptativa realizada pelo ITS para SM. A hipótese é que ML pode contribuir com a construção desse ITS.

A recomendação adaptativa de DM de SM, que é realizada pelo Módulo Tutor e se beneficia de uma colaboração com o Módulo do Estudante do ITS (HU, 2019; ALMASRI et al., 2019), pode ser implementado utilizando RL, onde um sistema determina as ações pedagógicas e melhora a eficiência desse mecanismo a partir das recompensas relacionadas ao resultado da tentativa de resolução de tarefas pelo estudante (ALSHAIKH; HEWAHI, 2021; GEORGILA et al., 2019; WANG, 2018). Nesse sentido, a recomendação adaptativa explorada nesta pesquisa é abordada através da colaboração ente os Módulos Tutor e do Estudante do ITS para SM, utilizando técnicas de ML, DM que representam o domínio de SM e são gerenciados pelo EKM do ITS para SM.

A Clusterização aplicada a dados de estudantes em um ITS permite o agrupamento desses estudantes com características semelhantes, possibilitando ao ITS trabalhar com a adaptação do conteúdo em relação a cada cluster criado (MOUSAVINASAB et al., 2021). Por outro lado, a RL é capaz de recomendar ações no âmbito desses ITS que vão refinando no decorrer da sua atuação (ALSHAIKH; HEWAHI, 2021; GEORGILA et al., 2019; WANG, 2018), o que traz a hipótese de que associar essas duas técnicas de ML é promissor para a recomendação de DM de SM devido à possibilidade de ter modelos de RL específicos para cada conjunto de estudantes semelhantes possibilitados pela Clusterização, tornando o ambiente trabalhado pela RL mais específico e facilitando seu trabalho. Essas considerações foram importantes para esta proposta de ITS para SM descrita nas próximas seções.

3.3 Arquitetura

A Arquitetura de *Software* pode ser compreendida como uma abstração, construída a partir de decisões estruturadas para atender a determinados requisitos, que gerencia a complexidade do sistema e proporciona a comunicação e coordenação entre seus componentes (JAISWAL, 2019). Esta conceituação permite compreender a ideia de que a Arquitetura de *software* sintetiza decisões de projeto previamente realizados.

Esta tese propõe uma arquitetura de ITS para SM. Essa arquitetura foi proposta

a partir de refinamentos sucessivos através de estudos envolvendo experimento com os Módulos do Estudante e Tutor, sua integração, e de uma reflexão acerca do levantamento do conteúdo didático de SM gerenciado pelo EKM. Ela sintetiza as decisões tomadas a partir da pesquisa realizada.

É importante ressaltar que uma arquitetura de *software* pode ser vista por vários ângulos, como é o caso da proposta do modelo 4+1 para visualização de arquitetura (KRUCHTEN, 1995). Para este projeto, verificou-se a importância de apresentar visões sobre o processo e sobre os dados, que serão exibidas a seguir na forma dos diagramas de atividades e de classes da UML.

O ITS foi proposto com os Módulos: Tutor, Estudante, EKM e UI. A Figura 9 possibilita a compreensão desta arquitetura e de como seus componentes se comunicam. Quando estudante resolve tarefas de SM através da IU, o Módulo Tutor recomenda DM com o apoio do Módulo do Estudante. Esses DM são caracterizados conforme o EKM. A adaptação do conteúdo é possível graças à inteligência dos Módulos do Estudante e Tutor. O processo envolve também a atualização do desempenho do estudante. O professor utiliza o ITS como ferramenta no seu trabalho docente, podendo registrar turmas de estudo, tarefas como DM e realizar configurações no ITS para SM.

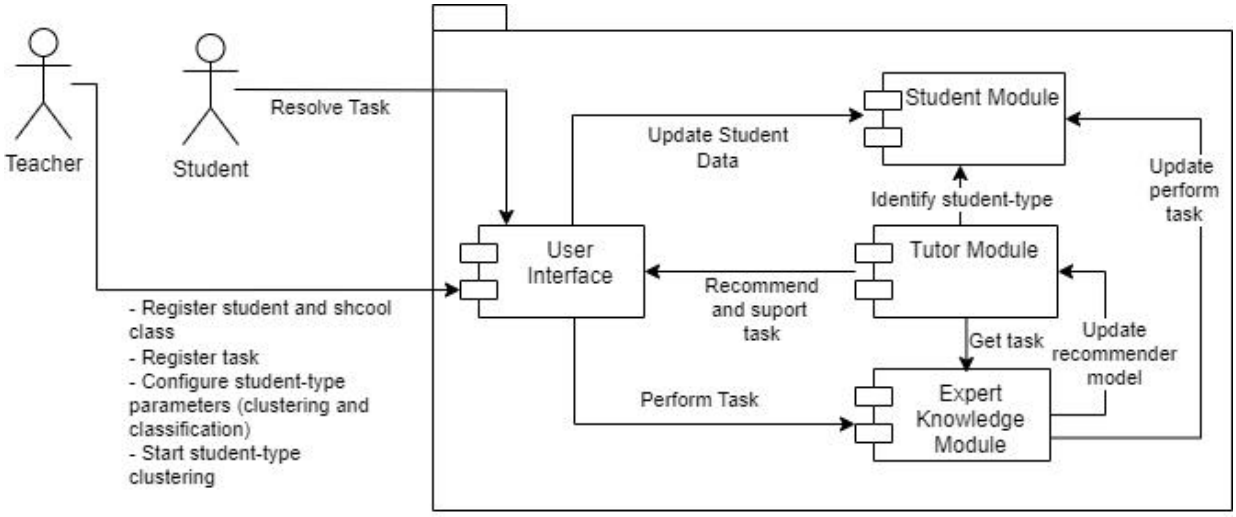

Figura 9 – Arquitetura Geral do ITS para SM. Construído a partir de (ALKHATLAN; KALITA,).

Considerando os requisitos apresentados, o planejamento da Arquitetura apresentado na Figura 9 relaciona-se com as seguintes funcionalidades. O Módulo Tutor é responsável por recomendar DM e oferecer suporte e *feedback* ao estudante na resolução dos mesmos. O Módulo do Estudante tem como objetivo representar e gerir a representação do estudante para informá-la ao Módulo Tutor, ajudando-o a trabalhar de forma adaptativa, e gerar informações sobre o desempenho dos estudantes. O Módulo EKM é responsável por representar o domínio de SM através dos DM e gerenciar esses DM. A UI é responsável

por permitir a interação entre os estudantes e o ITS de SM, disponibilizando os DM recomendados, disponibilizando o suporte necessário na resolução, o *feedback* e informações sobre o desempenho dos estudantes no estudo sobre SM.

O professor utiliza o ITS como ferramenta no seu trabalho, podendo registrar turmas de estudo, tarefas como DM e realizar configurações envolvendo os algoritmos de ML utilizados. A Figura 10 apresenta o diagrama de atividades com a sequência de atividades do ITS. O diagrama é dividido em professor, estudante e sistema. O professor inicia o uso do sistema com a possibilidade de atualizar os modelos de ML responsáveis pela identificação do tipo do estudante, o que é executado pelo Módulo do Estudante. Em seguida, o professor pode registrar turmas de estudo, criar DM e associar DM às turmas de estudo.

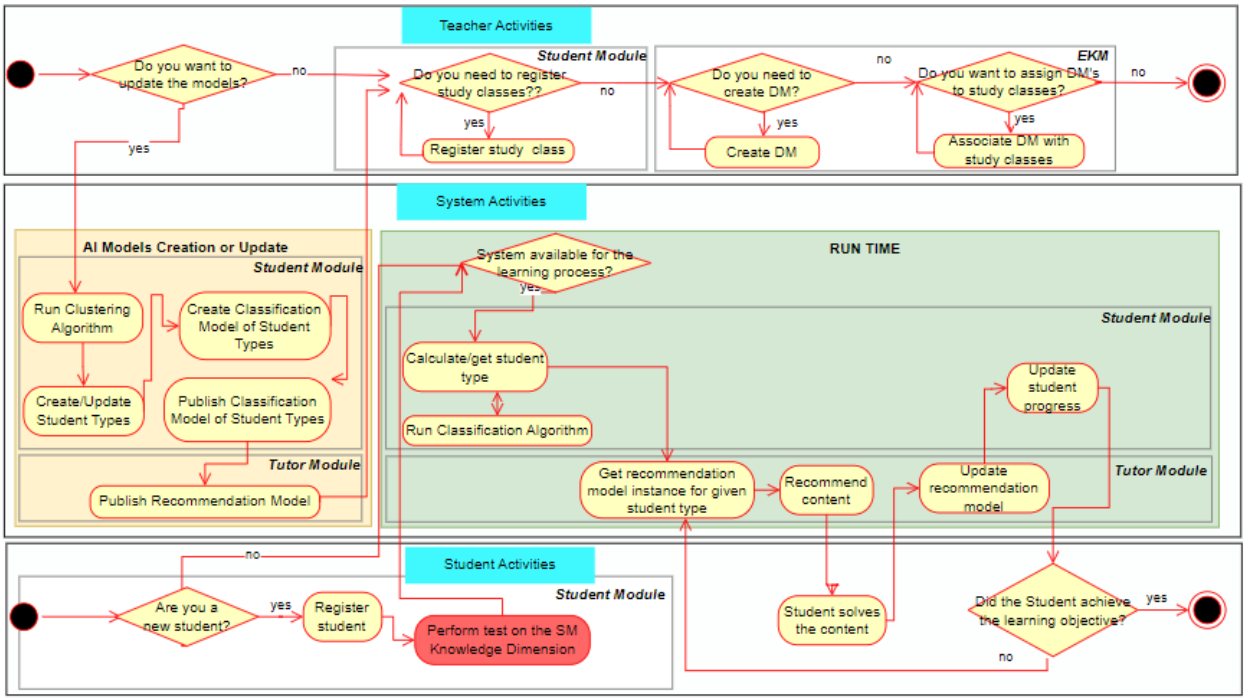

Figura 10 – Diagrama de Atividades do ITS para SM

O sistema possui as atividades que são mais relacionadas com a manipulação dos modelos de ML e as atividades *RUN TIME*, que se tratam do fluxo da recomendação de DM. As atividades relacionadas com a manipulação dos modelos de ML possibilitam executar o algoritmo de Clusterização para agrupar os estudantes por tipo, criar e publicar modelo de classificação de tipos de estudantes alinhados aos tipos identificados pelo algoritmo de Clusterização e publicar o modelo de recomendação e seus devidos parâmetros. A ideia é que o professor possa configurar e gerenciar a execução do sistema e do algoritmo de ML. As atividades *RUN TIME*, que são executadas quando o sistema estiver habilitado para o processo ensino-aprendizagem de SM, possibilitam a identificação do tipo de estudante de SM através do algoritmo de classificação, recomendar DM de acordo com o modelo

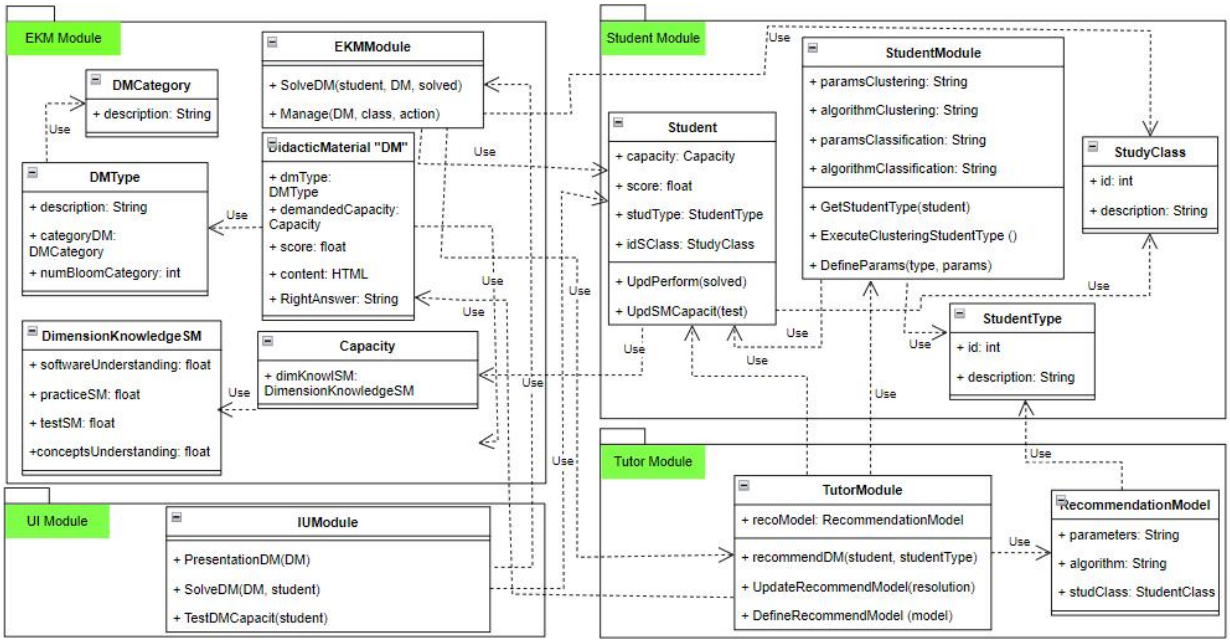

Figura 11 – Diagrama de Classes do ITS para SM

de recomendação adequado para o referido tipo de estudante, atualizar o modelo de recomendação e o progresso do estudante de acordo com a performance do estudante. O estudante, caso seja um novo usuário do sistema, cadastra-se e resolve um teste da dimensão de conhecimento de SM, o que é necessário para o sistema classificar seu tipo. O fluxo de recomendação de DM ocorre até que o estudante atinja o objetivo de aprendizagem.

A Figura 11 apresenta um diagrama de classes para o ITS para SM. O Módulo EKM gerencia e representa DM de SM através da relação entre DM, *DM Type*, *DM Category*, *Capacity* e *Dimension of Knowledge of SM*. Essas classes representam o domínio de SM e são detalhadas na seção 3.3.1. O Módulo do Estudante, que é apresentado com mais detalhes na seção 3.3.2 possui uma classe para lidar com o Estudante, que gerencia sua capacidade, seu desempenho, possibilita a resolução de um teste de aptidão em SM, o seu tipo de estudante de SM e a sua turma. Ele também possui uma classe para lidar com o seu funcionamento, gerenciando os algoritmos de Clusterização e Classificação, uma classe para lidar com o tipo de estudante, e outra para lidar com a turma do estudante. O Módulo Tutor, que é apresentado com mais detalhes na seção 3.3.3, possui classes para lidar com a recomendação de DM, o que permite atualizar o modelo de recomendação e seus parâmetros relacionando com o respectivo tipo de estudante de SM. O Módulo UI possui uma classe que possibilita ao estudante responder um problema em um DM e realizar o teste de capacidade de SM.

3.3.1 Expert Knowledge Module

O EKM representa o conteúdo do domínio a ser ensinado e o disponibiliza ao ITS para que ele o entregue ao estudante (ALKHATLAN; KALITA,). Para isso, o EKM deverá contribuir para a operacionalização do conteúdo do SM através da utilização de algoritmos baseados em ML, nos demais módulos, a partir de uma semântica relacionada ao domínio do SM.

O EKM foi construído a partir do estudo sobre o conteúdo de SM apresentado na seção 3.1. A proposta de (ALKHATLAN; KALITA,) para a construção do EKM, que propõe o levantamento do domínio com especialistas, contribuiu com esta abordagem de modelagem do EKM para o ITS para SM, que trabalhou com o levantamento do domínio de SM utilizando documentos oficiais relacionados ao currículo de SM construídos por especialistas. O EKM oferece uma representação do domínio de SM através do conjunto de DM de SM que contribui para a modelagem do algoritmo *Q-Learning* realizada pelo Módulo Tutor.

O EKM possui uma abstração do conteúdo que suporta os exercícios de SM que o ITS disponibiliza aos estudantes. Foram necessários realizar decisões pedagógicas e técnicas em torno da construção do ITS para SM que impactam o EKM.

As decisões pedagógicas consideram a possibilidade de ensinar SM de forma que o aprendizado comece com experiências práticas, o que contribui gradativamente para aumentar o conhecimento conceitual. O ITS para SM deve ser capaz de ser utilizado por estudantes e profissionais, abordando problemas em diferentes níveis de complexidade. Além disso, o ITS abrange amplamente os temas abordados no SM. Por fim, o ITS deve permitir uma experiência de aprendizado que considere o avanço gradual na obtenção de capacidades em SM pelos estudantes.

As decisões técnicas consideraram um EKM para o domínio SM que pode ser reutilizado, expandido e alinhado com o projeto arquitetural de um ITS. O EKM deve contribuir para a operacionalização do conteúdo SM usando algoritmos baseados em ML. A modelagem de conteúdo deve definir uma semântica que facilite a criação e modificação de atividades no ITS. As decisões de projeto consideram que o conteúdo possa ser gerenciado pelo EKM e recomendado pelo *Módulo Tutor*, facilitando o processo de avaliação de algoritmos de ML em ITS para SM.

Os temas em SM propostos para este EKM foram organizados em quatro categorias: *A - Compreensão de Software*, *B - Prática de SM*, *C - Teste de SM* e *D - Compreensão de Conceitos*. Essa categorização, que foi realizada a partir do estudo apresentado na seção 3.1, permitiu o planejamento da representação do conteúdo de SM com os DM de SM para serem abordados por este EKM. A Tabela 8 representa o conteúdo de SM abordado pelo EKM.

Este EKM foi proposto a partir de adaptações do currículo de SM para o objetivo desta pesquisa, considerando as decisões pedagógicas e técnicas. As categorias de conteúdo re-

sultantes desta análise são apresentadas na Figura 12. Entretanto, esta tese aborda o contexto educacional de SM com perspectiva prática, trazendo a ideia de que é possível expandir a proposta para o contexto organizacional de SM, o que requer adaptações com relação ao conceito de Gestão de Configuração e Mudança. Outra questão é a ideia de *Conteúdo no Contexto de um Software*, que se refere à possibilidade de que as tarefas de SM, representadas em DM, estejam contextualizadas em um produto de *software* em específico, o que indica a possibilidade de haver conjuntos de DM que levam à aprendizagem da SM em determinado *software* e atende à educação em SM nos âmbitos acadêmico e organizacional.

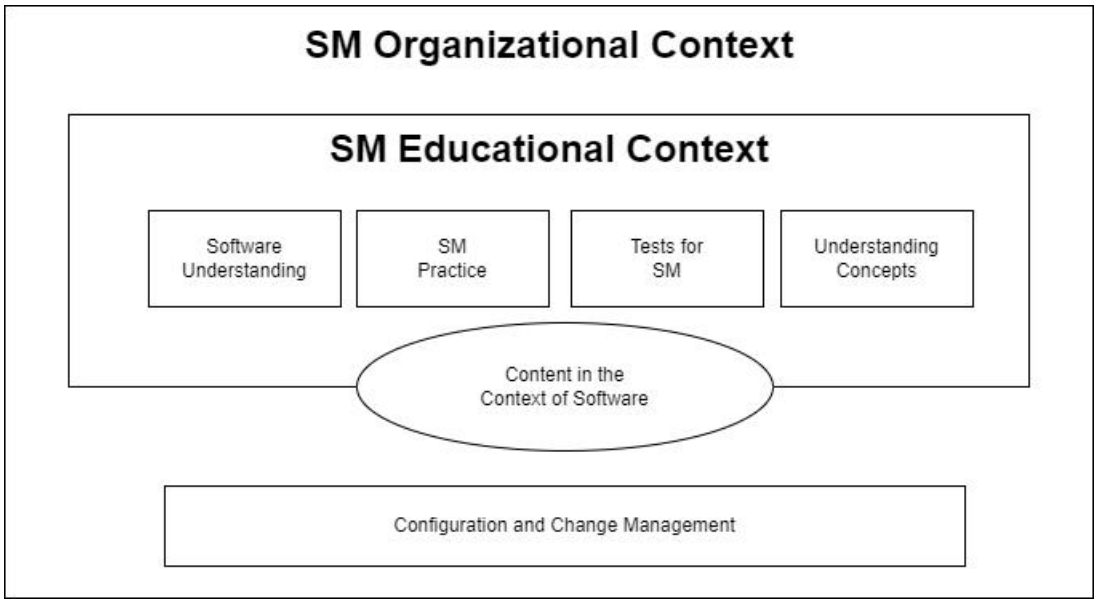

Figura 12 – Categorias de conteúdo para SM

As decisões de projeto em relação ao domínio de SM permitiram a proposição deste EKM. A Figura 13 apresenta o diagrama de classes parcial das classes deste EKM, que traz uma visão sobre a estruturação do conteúdo de SM por meio de DM e sobre as funcionalidades que o EKM oferece.

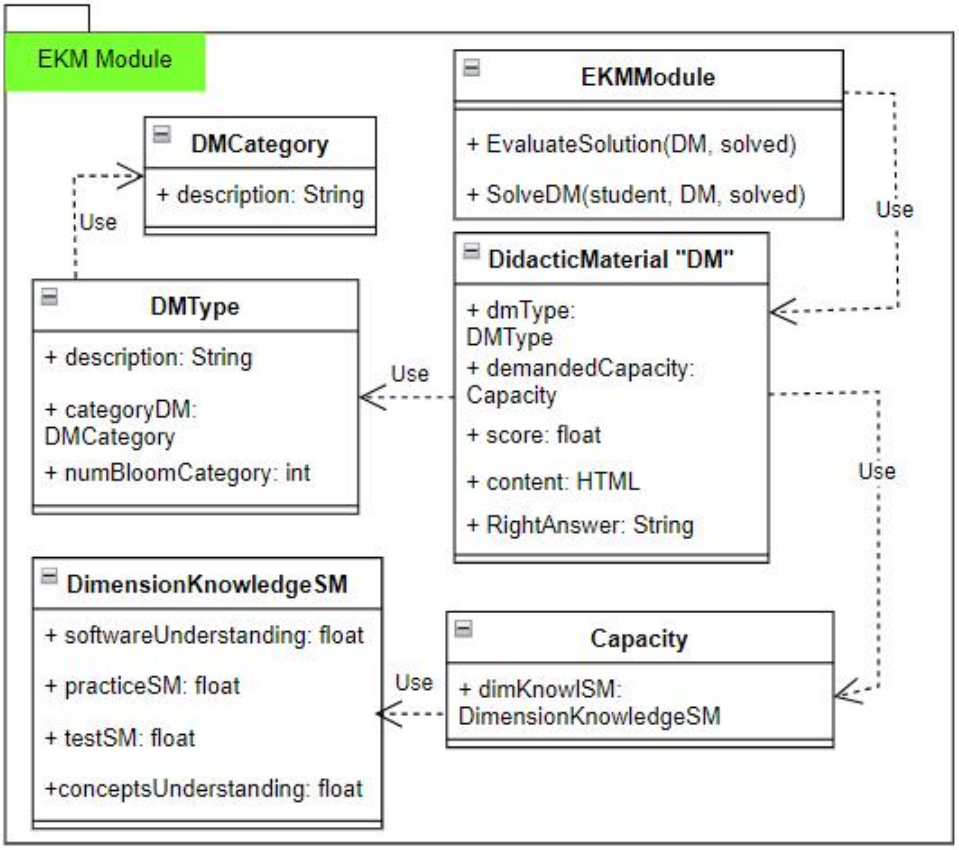

Figura 13 – Diagrama de Classes Parcial para o EKM

O diagrama apresentado na Figura 13 possui seis classes. A classe *EKMModule* permite ao estudante resolver um DM e realiza a avaliação do mesmo. Os DM são generalizados por *DM Type* e esses *DM Types* são generalizados por *DM Category*. *DM Category* e *DM Type* são equivalentes à Tabela 18. Um DM possui um conteúdo apresentado no formato de HTML, uma resposta correta para ser utilizada no momento de avaliação, uma pontuação e o atributo capacidade demandada para realizar auxiliar no processo de simulação de resolução do DM. O número de dimensões da taxonomia de Bloom foi utilizado no *DM Type* para estimar a complexidade cognitiva que os DM assumem. A classe *Capacity* oferece um suporte ao Módulo do Estudante ao representar o conhecimento do estudante sobre a dimensão de conhecimento de SM relacionado a determinado DM.

3.3.2 Módulo do Estudante

O Módulo do Estudante tem a finalidade de contribuir com o Módulo Tutor através do fornecimento de informações que o auxiliem na adaptação do conteúdo ao estudante, conforme proposto por (HU, 2019; ALMASRI et al., 2019). Por isso, o Módulo Tutor trabalha com modelos de recomendação específicos para cada tipo de estudante, conforme apresenta a seção 3.3.3.

O Módulo do Estudante proposto para o ITS para SM possui três funcionalidades, que são: a identificação do tipo de estudante de SM, a gestão dos dados do estudante e a possibilidade do estudante resolver um teste de aptidão de SM. Os Tipos de Estudantes de SM representam conjuntos com capacidades de SM semelhantes. A decisão de projeto de utilizar Tipos de Estudantes foi a possibilidade de construir modelos de recomendação de DM para o Módulo Tutor específicos para cada tipo de estudante de SM, o que se relaciona com a definição proposta por (HU, 2019; ALMASRI et al., 2019).

O Módulo do Estudante precisa identificar o tipo de estudante de SM. Essa identificação de tipos de estudantes de SM, quando o ITS possui uma base de dados da dimensão de conhecimento de estudantes de SM, pode utilizar a Clusterização, pois ela é utilizada para agrupar estudantes a partir de características similares nos Módulos do Estudantes de ITS (MOUSAVINASAB et al., 2021). A Classificação pode ser usada em situações em que os tipos de estudantes de SM já estão devidamente armazenados e um novo estudante iria utilizar o ITS, que classifica seu tipo de estudante com base no modelo de tipos de estudante gerado inicialmente pela Clusterização. O teste de aptidão de SM tem como finalidade disponibilizar informações sobre a dimensão de conhecimento de SM de estudantes, alimentando a base de dados para que essa identificação de tipos de estudantes de SM seja possível. Esta tese enfatiza a Clusterização para a identificação inicial de tipos de estudantes de SM.

A Figura 10 apresenta o fluxo de atividades relacionadas a essas funcionalidades do Módulo do Estudante. As Atividades do Sistema incluem a execução do algoritmo de Clusterização e a criação/atualização dos tipos de estudantes de SM, que caracterizam a identificação inicial. As Atividades do Sistema também incluem a criação e a publicação de um modelo de Classificação dos tipos de estudantes. Isso permite, em Run Time, a obtenção do tipo do estudante, considerando a recuperação dos tipos de estudantes de SM armazenados após a Clusterização ou a Classificação. No caso da Classificação dos tipos de estudantes de SM, o processo ocorre após a realização de um teste de aptidão de SM.

A representação do conteúdo de SM disponibilizada pelo EKM contribui com a representação do estudante (ALKHATLAN; KALITA,). Nesse sentido, os atributos da dimensão de conhecimento de SM são utilizados para identificar os tipos de estudantes de SM através da similaridade dos estudantes nos grupos de estudantes gerados pela Clusterização.

A Figura 14 possui informações sobre a capacidade, o score, o tipo do estudante de SM e a turma. A capacidade possui a dimensão de conhecimento de SM, que visa quantificar o conhecimento do estudante sobre o conteúdo de SM e foi apresentada no EKM. O score quantifica a nota geral, que se refere a quantos DM de SM o estudante resolveu.

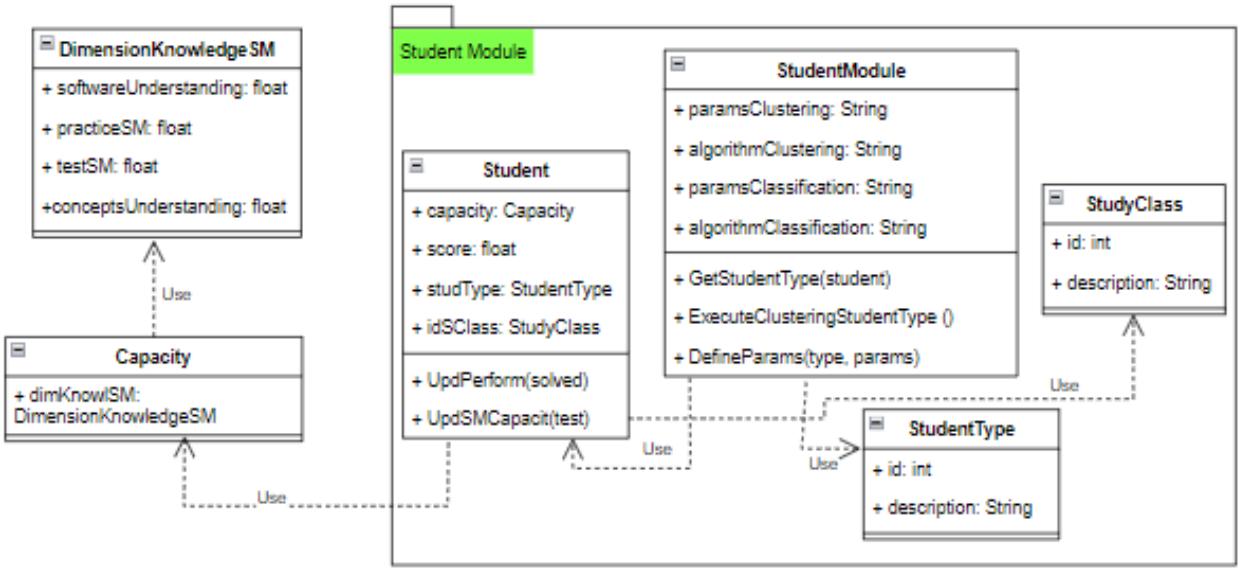

Figura 14 – Diagrama de Classes para o Módulo do Estudante

A classe *Student* inclui também os métodos para atualizar a performance do estudante e a sua capacidade de SM. A classe *StudentModule* inclui as configurações dos algoritmos de clusterização e classificação, além de métodos para sua execução e manipulação. Existem também classes para lidar com a turma e o tipo de estudante de SM.

A identificação inicial de tipos de estudantes através da Clusterização foi proposta para o ITS para SM com dois passos. O primeiro passo é executar, para a base de estudantes e dimensões de conhecimento de SM, os algoritmos de Clusterização e suas variações de parâmetros que permitem definir previamente o número de *clusters*. A definição prévia do número de *clusters* é importante para não haver variações do número de *clusters* em diferentes execuções da mesma configuração algoritmo/parâmetros. O segundo passo é verificar qual resultado da Clusterização melhor atende às duas condições de avaliação e usá-lo de modo que os *clusters* representem os tipos de estudantes de SM, realizando as devidas atualizações na base de dados.

As duas condições, capazes de avaliar a Clusterização e indicar o algoritmo e parâmetros com os melhores resultados, são: i) O resultado precisa gerar de 3 a 5 *clusters* de estudantes. A ideia dessa condição foi propor um número de *clusters* viável para a realização deste experimento e viabilizar a integração dos Módulos do Estudante e Tutor utilizando esses resultados. A estimativa de 3 a 5 *clusters* traz a ideia desse número estar equilibrado, de modo que se essa quantidade fosse menor que 3 haveriam *clusters* muito heterogêneos e se fosse acima de 5 haveriam *clusters* muito homogêneos. Essa condição permite variar da quantidade de *clusters* para cada algoritmo de Clusterização executado. ii) Melhor atenda ao coeficiente de *Silhouette* com Distância Euclidiana.

A métrica coeficiente de *Silhouette* com Distância Euclideana foi utilizada para avaliar

a qualidade dos *clusters* gerados e possibilitar a escolha do melhor resultado que representa os tipos de estudantes de SM. A justificativa do uso desse coeficiente foi porquê, entre um intervalo de (-1 e 1), ela indica que quando o seu valor é mais alto existe uma melhor atribuição dos objetos nos *clusters* (ŘEZANKOVÁ, 2018; SHUTAYWI; KACHOUIE, 2021).

3.3.3 Módulo Tutor

O Módulo Tutor pode ser comparado às estratégias de ensino executadas em um ambiente de ensino-aprendizagem, porém sem a intervenção do professor (ALSHAIKH; HEWAHI, 2021). A colaboração entre o Módulo Tutor e o Módulo do Estudante proporciona ao ITS a possibilidade de trabalhar de forma adaptativa, pois o Módulo do Estudante fornece ao Módulo Tutor a representação do estudante para que sejam realizadas ações adaptadas a essa representação (HU, 2019; ALMASRI et al., 2019).

O Módulo Tutor decide qual conteúdo deve ser entregue ao estudante (ALKHATLAN; KALITA,) em um ITS. Nesse sentido, esta pesquisa propõe um método para recomendação de DM para um ITS para SM (Módulo Tutor) baseado em RL e integrado à identificação de tipos de estudantes de SM realizada pelo Módulo do Estudante.

O Funcionamento Geral do Módulo Tutor pode ser compreendido com as Figuras 15 e 16. O Diagrama de Sequência para Recomendar DM de SM, apresentado na Figura 15, foi projetado de modo que a UI, para exibir um DM de SM para o usuário, enviará uma mensagem ao método de recomendação de DM do Módulo Tutor. O Módulo Tutor precisa identificar o tipo do estudante de SM e executar um algoritmo de recomendação de DM de SM que leva em consideração essa informação sobre o tipo de estudante de SM e recuperar o DM para exibi-lo na UI.

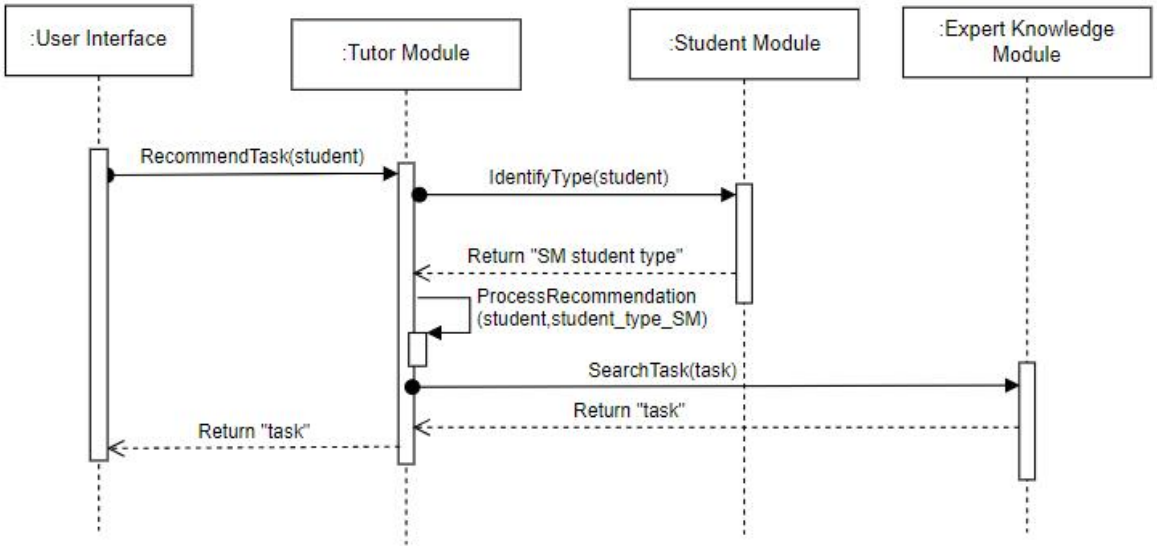

Figura 15 – Diagrama de Sequência para Recomendação de DM de SM

O Diagrama de Sequência para a resolução do DM de SM, apresentado na Figura 16, considera que quando o estudante resolve um DM de SM a UI envia para o Módulo Tutor a identificação do estudante e os dados da resolução do DM visando atualizar a representação do estudante no Módulo do Estudante quanto ao DM resolvido. Em seguida, o Módulo Tutor afere o resultado através do EKM para que o Módulo do Estudante realize as devidas atualizações.

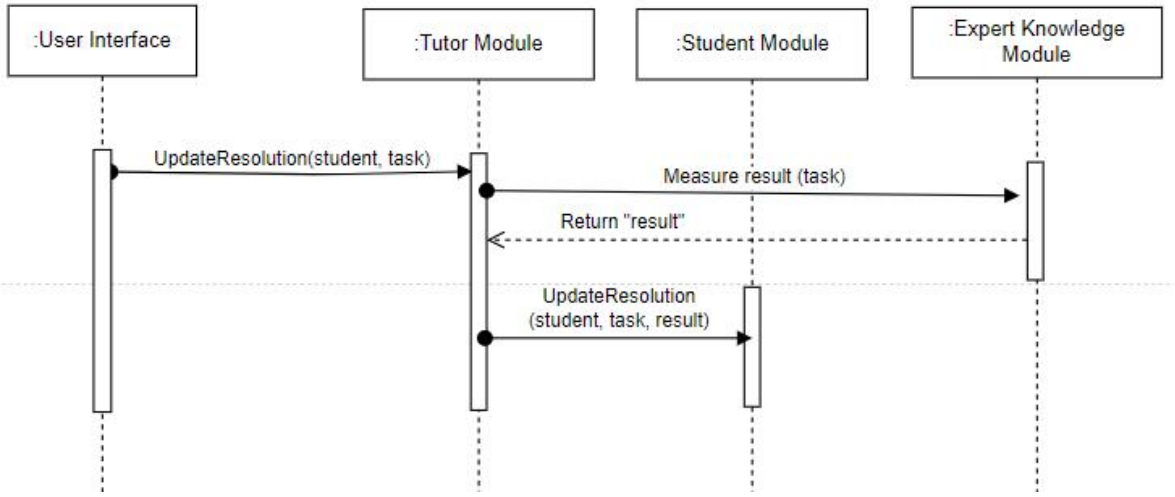

Figura 16 – Diagrama de Sequência para a resolução de DM de SM

O Diagrama de Sequência do ITS para SM, apresentado na Figura 10, aborda o funcionamento do Módulo Tutor com mais detalhes em relação aos modelos de ML. *System Activities* indica que os modelos que geram os tipos de estudantes de SM, que envolvem a Clusterização e a Classificação e são construídos no âmbito do Módulo do Estudante, trazem resultados (tipos de estudantes de SM) para a construção do modelo de recomendação de DM de SM. As Figuras 15 e 16 ilustram detalhadamente a relação entre a funcionalidade de recomendação de DM de SM, a identificação de tipos de estudantes de SM e o uso do sistema pelo estudante de forma alinhada com a Figura 10.

Considerando o apoio do Módulo do Estudante ao Módulo Tutor em relação à representação do estudante (HU, 2019; ALMASRI et al., 2019), esta tese propõe que a recomendação de DM de SM seja construída com o algoritmo de RL *Q-Learning* a partir de uma estratégia onde o *Q-Learning* seja instanciado para cada tipo de estudante de SM gerado pelo Módulo do Estudante. Com isso, haverão modelos de recomendação específicos para cada tipo de estudante de SM, que serão construídos com RL durante o uso do ITS e representarão estratégias de recomendação de DM de SM alinhadas com o conhecimento em SM de estudantes dos *clusters* representados pelos tipos de estudantes de SM.

Conforme apresentado na Figura 10, no início da utilização do ITS, logo após os tipos de estudantes de SM terem sido gerados pelo Módulo do Estudante, o Módulo Tutor inicializa tabelas Q das instâncias do *Q-Learning* com dados aleatórios, representando

que os modelos de decisão ainda não possuem experiência. Após algum tempo de uso do ITS, cada tabela Q relacionada a uma determinada instância do *Q-Learning* terá um modelo de decisão que representa o conhecimento obtido pelo algoritmo *Q-Learning* até aquele momento. Com isso, o Módulo Tutor, através das instâncias do algoritmo *Q-Learning* com as tabelas Q construídas com RL, terá o modelo de decisão específico para cada tipo de estudante para a recomendação de DM de SM.

3.3.3.1 *Q-Learning* para Recomendação de DM de SM

O algoritmo *Q-Learning* (SHAWKY; BADAWI, 2018) foi usado no Módulo Tutor do ITS para SM porque ele constrói e melhora este modelo baseado em RL em tempo de execução. Este algoritmo modela as melhores ações para determinados estados usando probabilidades e políticas de exploração, conforme funcionamento explicado nas seções 2.1.4.1 e 2.1.5.3, através dos conceitos de Agentes Inteligentes e RL e do próprio algoritmo *Q-Learning*.

O modelo *Q-Learning* possui estados e ações. O feedback obtido do ambiente após a execução das ações, denominado reforço, é utilizado para atualizar seu modelo, i.e. atualizar como o modelo lida com cada estado de conhecimento dos estudantes. Para construir o conjunto de Ações utilizou-se o conjunto de DM de SM categorizados com a Taxonomia de Bloom e com o EKM. A construção do conjunto de estados visou representar o progresso do conhecimento para os DM escolhidos. O estudante começa com o estado zero e termina com o número total de DM configurados no ITS. Supondo que o professor queira trabalhar com 200 DM de SM, haverão 200 ações, pois cada ação representa uma recomendação de DM de SM, e 201 estados, pois existe o estado zero que indica que o ITS não capturou nenhum conhecimento de SM do estudante.

O algoritmo *Q-Learning* (SHAWKY; BADAWI, 2018) modela as melhores ações para determinados estados e armazena este modelo na matriz Q. Conforme apresentado na seção 2.1.5.3, o algoritmo possui uma etapa de atualização da matriz Q, que é realizada conforme a equação 6. Essa atualização depende do reforço obtido e de como os parâmetros do algoritmo *Q-Learning* foram calibrados.

A Figura 17 ilustra uma matriz Q para um contexto com quatro DM e consequentemente cinco estados e quatro ações. A equação 6 apresentada na seção 2.1.5.3 representada por Q(s, a), sendo s um estado e a uma ação, gera os valores para esta matriz, conforme os exemplos Q(0,3), Q(1,2) e Q(2,1) ilustrados.

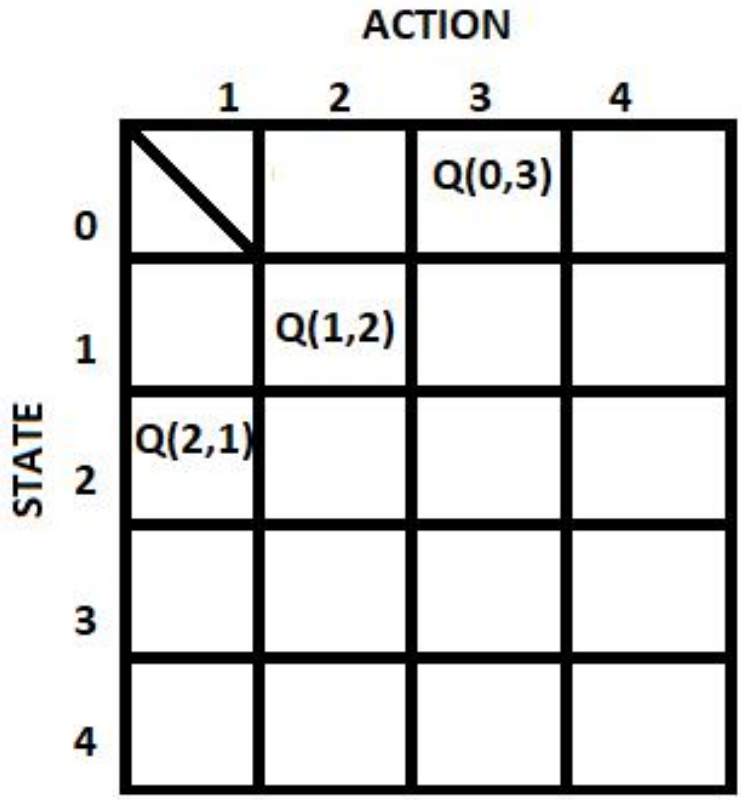

Figura 17 – Exemplo de Matriz Q

Percentual de Exploração (%), Reforço Positivo, Reforço Negativo, Taxa de Aprendizado (α) e Fator de Desconto (γ) são os parâmetros do *Q-Learning* que precisam ser calibrados. Percentual de Exploração (%) indica o percentual de exploração do ambiente em relação ao uso de políticas de exploração aleatória. Reforço Positivo e Negativo indicam um número inteiro utilizado para representar o acerto e outro para representar o erro com relação ao estudante resolver um DM de SM. Taxa de Aprendizado (α) e Fator de Desconto (γ) são variáveis que vão alterar matematicamente a atualização da matriz Q e precisam ser calibradas.

Capítulo 4

Experimentos e Discussão

Este capítulo aborda os experimentos que contribuem para fornecer evidências às RQ apresentadas nesta Tese. As funcionalidades de identificação do tipo de estudante de SM (Módulo do Estudante do ITS para SM) e recomendação de DM de SM (Módulo Tutor do ITS para SM) foram avaliadas, bem como a colaboração desses dois módulos pode melhorar a eficiência da recomendação de DM de SM. Os experimentos utilizaram um *Dataset* de estudantes de SM e a simulação computacional de estudantes resolvendo DM de SM.

A seção 4.1 apresenta o Dataset de capacidades de SM de estudantes construído para a realização desta pesquisa. A seção 4.2 descreve a simulação realizada para a avaliação do Módulo Tutor e da sua integração com o Módulo do Estudante. A seção 4.3 apresenta o experimento que avalia a Clusterização no Módulo do Estudante. A Seção 4.4 avalia RL no Módulo Tutor sem considerar as capacidades iniciais de SM dos estudantes. A seção 4.5 avalia RL no Módulo Tutor considerando as capacidades iniciais de SM dos estudantes. A seção 4.6 avalia a integração entre o Módulo do Estudante e o Módulo Tutor. A seção 4.7 discute os resultados dos experimentos e realiza uma comparação com a literatura do tema. A seção 4.8 apresenta as ameaças à validade que refletem à limitação da pesquisa realizada.

4.1 Dataset de Estudantes de SM

Durante o desenvolvimento dessa pesquisa foi necessário construir um *Dataset* com capacidades de estudantes referentes à dimensão de conhecimento de SM para a avaliar os Módulos do Estudante e Tutor, bem como para a construção do modelo de recomendação de DM para SM. Este *Dataset* foi construído para o contexto de SM derivando informações do *Dataset* do *Exame Nacional de Desempenho dos Estudantes* (ENADE) do ano de 2017 para Ciência da Computação (INEP, 2022).

Para gerar um contexto de dados que represente as capacidades de SM de estudantes, a estratégia realizada utilizou as notas desses estudantes do ENADE (INEP, 2022), bem

como notas aleatórias e pesos para complementar o contexto de SM. Os quatro atributos da dimensão de conhecimento de SM *(softwareUnderstanding, practiceSM, testSM e conceptsUnderstanding)* foram gerados para cada um dos 8489 estudantes de Ciência da Computação que realizaram o ENADE de 2017.

O ENADE possui a nota geral e nota específica. Nota geral é referente a conhecimentos gerais e nota específica é referente ao curso, que é Ciência da Computação. Essas duas notas variam de zero (0) a 100 e foram utilizadas para compor esta estratégia. A nota geral está descrita nesta seção como *Generic Grade* (GG) e a nota específica para Ciência da Computação está descrita como *Computing Grade* (CG).

Considera-se que a prática de SM envolve conhecimento específico da área e conhecimento prático de SM (ANQUETIL et al., 2007), o que motivou a criação da nota complementar DEV_PR_SM, e que testes em SM envolvem conhecimento específico da área e conhecimento prático de testes em SM (WANG; KING; WICKBURG, 1999), o que motivou a criação da nota complementar DEV_TE_SM. A nota DEV_PR_SM representa o grau de desenvolvimento em prática de SM e a nota DEV_TE_SM representa o grau de desenvolvimento em testes para SM. Essas duas notas são aleatórias e foram criadas para complementar a geração de dados de estudantes de SM, variando de zero (0) a 100. Ao calcular cada atributo da dimensão de conhecimento de SM, os pesos um (1) e meio (1/2) multiplicam as notas, no contexto das duas categorias, para que cada nota utilizada represente parte do referido atributo da dimensão de conhecimento de SM.

Os cálculo das quatro categorias da dimensão de conhecimento de SM para cada um dos estudantes do *Dataset* estão apresentados nas Equações 7, 8, 9 e 10.

$$softwareUnderstanding = CG * 1 \tag{7}$$

$$practiceSM = CG * 1/2 + DEV\_PR\_SM * 1/2 \tag{8}$$

$$testSM = CG * 1/2 + DEV\_TE\_SM * 1/2 \tag{9}$$

$$conceptsUnderstanding = CG * 1/2 + GG * 1/2 \tag{10}$$

4.2 Simulação

Esta tese propõe um processo de simulação para avaliar a eficiência da recomendação de DM de SM. O modelo de RL utilizado no ITS, relacionado ao algoritmo *Q-Learning*, precisa de um grande número de interações com o ambiente para ser construído.

A Figura 18 apresenta o processo que representa a interação entre o ITS para SM e o ambiente educacional. Neste processo, o papel do estudante é executado através da

simulação. A sequência de atividades ocorre envolvendo os seguintes atores: o professor, o ITS e o estudante simulado. Inicialmente, o professor seleciona um conjunto de estudantes e um conjunto de DM e configura a sessão de aprendizagem com os parâmetros para as instâncias do *Q-Learning* apropriados. O simulador executará um *loop* que representa um conjunto de estudantes resolvendo DM de SM. Durante este processo, o simulador primeiro verifica se há um DM disponível. Se sim, o ITS usa o algoritmo *Q-Learning* e o estado de conhecimento de SM atual do estudante para *recomendar um DM*. O estudante *resolve o DM*. No processo de simulação, o simulador infere sucesso ou falha para a resolução do DM e atualiza o estado de conhecimento de SM do estudante. Quando o sistema identificar que já não existem mais DM para serem recomendados para os estudantes, o simulador para e exibe os resultados.

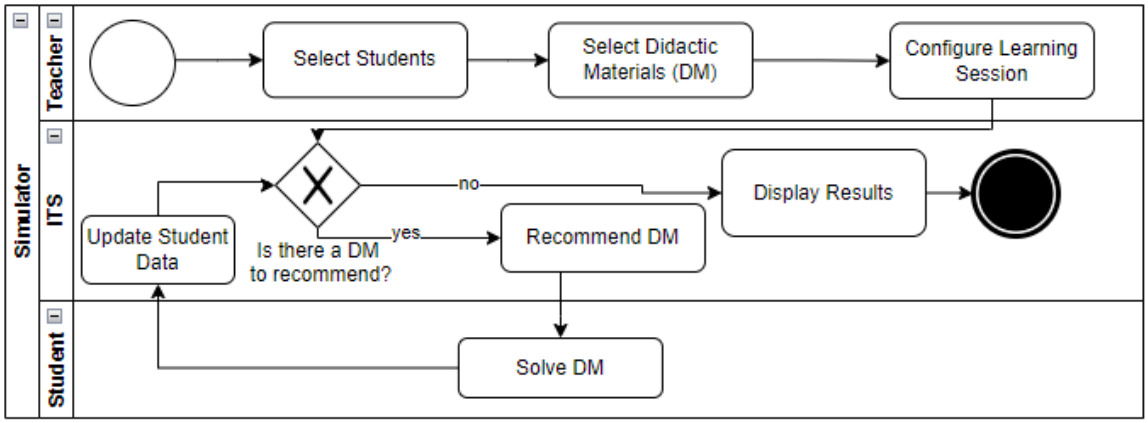

Figura 18 – Processo de Interação do ITS para SM com o ambiente educacional

O simulador infere sucesso ou insucesso verificando se a capacidade do estudante (atributo *Student.capacity*) é suficiente para resolver o DM considerando sua *DM.demandedCapacity*. Se o estudante tiver capacidade suficiente, o simulador atualiza o *capacity* do estudante atual, sua *score* e a lista de DM resolvidos.

O sistema permite criar um conjunto de estudantes usando um construtor padrão ou carregar um *Dataset* de estudantes, de acordo com o *Dataset* apresentado na seção 4.1.

Caso a criação do conjunto de estudantes seja selecionada, o parâmetro inicial para o simulador é o tamanho do conjunto. Estes estudantes criados possuem todos os atributos da dimensão de conhecimento de SM (*softwareUnderstanding*, *practiceSM*, *testSM* e *understandingConcepts*) inicializados com valor zero (0). O *score* do estudante representa o número de atividades resolvidas corretamente e inicialmente também é zero.

Os estudantes que são lidos do *Dataset* possuem a dimensão de conhecimento de SM conforme o arquivo carregado e o *score* também inicialmente é zero (0).

O simulador cria um conjunto de DM que é semelhante à seleção de DM de SM pelo professor. O simulador recebe o tamanho do conjunto DM e cria atividades de todos os tipos (*DMType*) para o tamanho determinado. O simulador calcula a capacidade

necessária para cada atividade. O atributo *categoryDM* define a categoria do DM e indica os atributos da dimensão do conhecimento associados ao atributo *demandedCapacity*. O Algoritmo 3 descreve o cálculo da capacidade de conhecimento demandada e o *score* para um determinado DM de SM.

Algoritmo 3 Calcular a Capacidade Demandada e Score do DM

```
i ← 1
while i <= n do
    MaterialDidatico[i].demandedCapacity[j] ← i * (numBloomCategory/n)
    MaterialDidatico[i].score ← numBloomCategory/n
    i++
end while
```

Para configurar as sessões de aprendizagem, o professor descreve número médio vezes em que um estudante tenta resolver um DM e os parâmetros das instâncias do algoritmo *Q-Learning*: reforço positivo, reforço negativo, taxa de aprendizagem, fator de desconto e indicador do percentual de exploração.

A recomendação de DM obedece a sequência de estudantes gerados ou pode ser aleatória. Inicialmente a matriz Q é gerada aleatoriamente. Para cada estudante, enquanto este não atingiu o *score* máximo ou o número máximo tentativas, ocorre a recomendação dos DM, a observação de se houve mudança de estado por parte do simulador de estudantes, e a atualização da matriz Q.

Na simulação da resolução do DM pelo estudante o sistema verifica a sua capacidade de SM em *Knowledge Dimension* para inferir o acerto ou o erro, através da equação DM.*demandedCapacity* - DM.*score*. Quando ocorre o acerto, a capacidade do estudante, seu *score* e a lista de DM resolvidos são atualizados. Após a realização destas etapas, o resultado da simulação é exibido.

Conforme a Figura 19 apresenta, o domínio da simulação conhece as classes de persistência referentes ao EKM e o estudante, porém o domínio de recomendação de DM do ITS conhece apenas as classes de persistência referentes ao estudante e ao DM. Essa configuração é possível graças ao Módulo Tutor depender apenas do tipo de estudante de SM gerado previamente pelo Módulo do Estudante, o que seria diferente caso o Módulo do Estudante fornecesse ao Módulo Tutor informações em tempo real.

A prototipação do ITS para SM utilizada nos experimentos baseados em simulação realizada nesta tese utilizou uma estruturação de tipos de DM apresentada na Tabela 18. A Tabela 18 possui um subconjunto de tipos de DM de SM, também dentro das quatro Categorias de SM, em relação à Tabela 8. Trata-se de uma simplificação realizada para a simulação que foi projetada e utilizada nesta pesquisa durante a avaliação do ITS proposto para SM. Essa simplificação foi necessária para lidar com a questão do alto custo computacional. Ter um número maior de DM implica em mais trabalho para o ITS e para os estudantes e consequentemente mais custo computacional em um ambiente simulado.

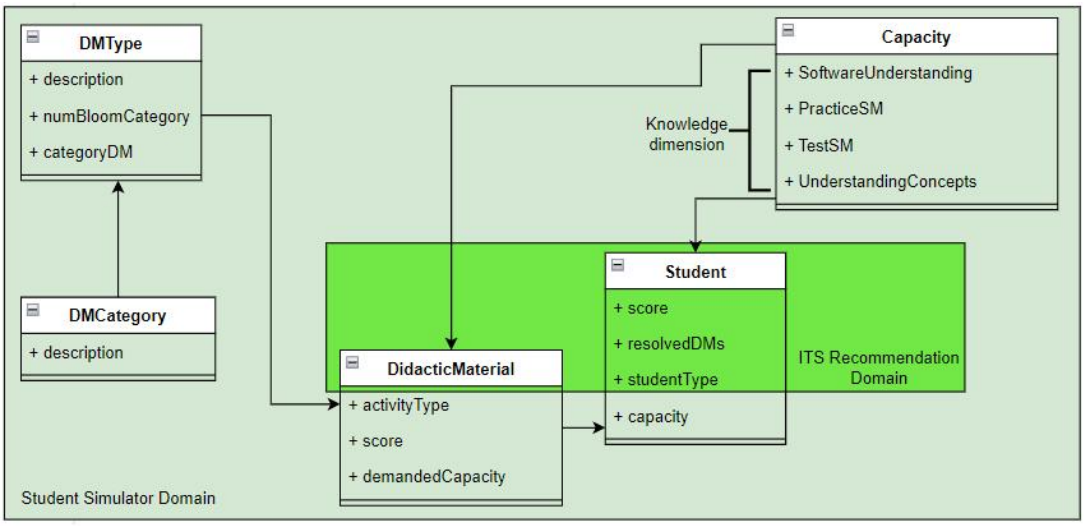

Figura 19 – Classes de persistência relacionadas à Simulação

Tabela 18 – DM's de SM categorizados com a Taxonomia de Bloom Revisada para Simulação do Ambiente do ITS

DM Type	**Remember**	**Understand**	**Apply**	**Analyze**	**Evaluate**	**Create**	**No. Dimens.**
A1 - Understand the result of execution (source code)	x	x					2
A2 - Order the program (parson puzzles) (DURAN et al., 2020)	x	x	x	x			4
A3_1 - Reply to abstract (source code)	x	x	x	x			4
A3_2 - Reply to abstract (UML models)	x	x	x	x			4
B1 - Defect Fix type SM	x	x	x	x	x	x	6
B2 - SM of the Environmental Adaptation type	x	x	x	x	x	x	6
B3 - SM of Add Functionality type	x	x	x	x	x	x	6
B4 - Refactoring	x	x	x	x			4
C1 - Tests	x	x	x	x	x		5
D1 - SM Process	x	x					2
D2 - Maintainability of Software	x	x					2

A Taxonomia Revisada de Bloom (AMER, 2006) foi utilizada para estimar as dimensões do processo cognitivo necessário pelos estudantes em cada tipo de atividade. A utilização da Taxonomia Revisada de Bloom permite inferir a complexidade dessas atividades a partir da última coluna da Tabela e permite avaliar o mecanismo de recomendação de DM de SM por meio deste processo de simulação. Apesar da Taxonomia Revisada de Bloom ter sido usada nesta pesquisa para estimar a complexidades dos DM de SM, este valor pode ser aferido de outras formas, e.g. levantado e avaliado por um conjunto de professores, para que o mesmo seja adicionado na base de dados do ITS.

4.3 Avaliação de Clusterização no Módulo do Estudante do ITS para SM

Um ITS precisa de que o Módulo do Estudante represente o estudante e forneça essa representação para que sejam tomadas decisões pedagógicas adaptadas ao estudante (ALMASRI et al., 2019). O conceito de tipo de estudante de SM representa padrões de similaridade no conhecimento de SM de estudantes e é usado nesta pesquisa para representar o estudante.

A clusterização é utilizada nos ITS para agrupar os estudantes a partir de características similares (MOUSAVINASAB et al., 2021). Ela é utilizada nesta tese para agrupar estudantes a partir da similaridade de conhecimentos em SM, representando os tipos de estudantes de SM.

Algoritmos de clusterização, disponibilizados pela biblioteca *scikit-learn* (SCIKIT-LEARN, 2023), foram executados com os 8489 registros do *Dataset* de Estudantes de SM obtido da seção 4.1. A relação de algoritmos e parâmetros, e os resultados estão disponíveis (FRANCISCO, 2023a).

A escolha de um algoritmo de Clustering e seus parâmetros possui dependência dos dados de entrada (EZUGWU et al., 2022). Foram exploradas variações dos parâmetros, como: *n_clusters*, que refere-se à definição prévia do número de *clusters*; *affinity*, que é uma métrica usada para calcular a distância entre instâncias; e *random_state*, que faz com que a geração de números aleatórios para inicialização do centróide seja determinística relacionada a um número inteiro. Os resultados dessas execuções estão disponíveis (FRANCISCO, 2023b).

Os algoritmos de Clusterização disponíveis na biblioteca *scikit-learn* (SCIKIT-LEARN, 2023) que possibilitam a definição prévia do número de *clusters* foram utilizados. A possibilidade de definição prévia do número de *clusters* é importante para não haver variações do número de *clusters* em diferentes execuções do mesmo algoritmo. Foram usadas com diferentes variações nos parâmetros dos algoritmos. *K-Means, Spectral Clustering, Agglomerative clustering,* BIRCH e *Bisecting K-Means* foram os algoritmos utilizados. Os parâmetros e indicador de resultado *(Silhouette)* referentes a essas execuções estão listados a seguir e os detalhes de processamento estão disponíveis publicamente (FRANCISCO, 2023a).

- *K-Means*: *n_clusters* variando de dois (2) a seis (6)
- *Spectral clustering*: *n_clusters* variando de dois (2) a seis (6), *assign_labels*='discretize', *random_state*=0, *affinity*='euclidean'
- *Agglomerative clustering*: *n_clusters* variando de dois (2) a seis (6) e *affinity* = 'euclidean'

- ❏ BIRCH: *n_clusters* variando de dois (2) a seis (6)
- ❏ *Bisecting K-Means*: *n_clusters* variando de dois (2) a seis (6)

A métrica coeficiente de *Silhouette* com Distância Euclideana foi utilizada para avaliar a qualidade dos *clusters* gerados e possibilitar a escolha do algoritmo e seus respectivos parâmetros. A justificativa do uso desse coeficiente foi porquê, entre um intervalo (-1 e 1), ela indica que quando o seu valor é mais alto existe uma melhor atribuição dos objetos nos *clusters* (ŘEZANKOVÁ, 2018; SHUTAYWI; KACHOUIE, 2021).

Foram criadas duas condições para realizar a escolha do algoritmo e parâmetros: i) O resultado precisa gerar de três (3) a cinco (5) *clusters* de estudantes. A ideia dessa condição foi propor um número de *clusters* viável para a realização deste experimento e viabilizar a integração dos Módulos do Estudante e Tutor utilizando esses resultados. Considera-se a estimativa de três (3) a cinco (5) *clusters* por considerar esse número equilibrado, de modo que se essa quantidade for menor que três (3) haveriam *clusters* muito heterogêneos e se for acima de cinco (5) haveriam *clusters* muito homogêneos. Essa condição permite analisar a variação da quantidade de *clusters* para cada algoritmo de Clusterização executado. ii) Melhor atenda ao coeficiente de *Silhouette* com Distância Euclidiana.

A Tabela 19 apresenta o melhor resultado da clusterização para as instâncias que atendem às condições definidas. O *K-Means* com quatro (4) *clusters* definidos previamente foi o algoritmo que melhor atendeu a essas condições, com o Índice de *Silhouette* 0.24033005280108247.

Tabela 19 – Melhores Resultados da Clusterização considerando as condições

Algorithm	**Parameters**	**Silhouette (euclidean)**	**Seconds**
Agglomerative clustering	n_clusters=4, affinity='euclidean'	0.16695543836240956	5
BIRCH	n_clusters=3	0.19868815581513474	7
Bisecting K-Means	n_clusters=3	0.22913192420912357	1
K-Means	n_clusters=4, random_state=0	0.24033005280108247	2

O computador utilizado para execução, disponibilizado Google Colab (GOOGLE, 2023), possui *Central Process Unit* (CPU) Intel(R) Xeon(R) com 2.2GHz, 13 GB de *Random Access Memory* (RAM) e 120 GB de disco rígido.

4.4 Avaliação do RL no Módulo Tutor do ITS para SM sem considerar as capacidades iniciais de SM dos estudantes

Esta seção apresenta uma avaliação da Recomendação de DM de SM sem considerar as capacidades iniciais em SM. Assim, as capacidades de SM dos estudantes gerados

iniciam-se com valor "zero". Este experimento avaliou a modelagem utilizada no algoritmo *Q-Learning* para a recomendação de DM de SM de acordo com o EKM representado pela Tabela 18.

Usando simulação, o método de recomendação de DM de SM baseado no algoritmo *Q-Learning* foi avaliado. Esta seção detalha a avaliação realizada baseada em simulação.

A simulação realizada nesta seção trabalhou com a ideia de criar um conjunto de estudantes virtuais com o construtor padrão.

Para avaliar o método de recomendação de DM de SM sem considerar as capacidades iniciais de SM, foram feitas várias simulações usando diferentes configurações de parâmetros para o algoritmo *Q-Learning*. Dez (10) configurações diferentes foram usadas para o mesmo cenário de simulação, conforme Tabela 20.

Tabela 20 – Execuções e Resultados da Simulação para a Recomendação de DM de SM sem considerar as capacidades iniciais de SM dos estudantes

Execution	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Max. Number of Attempts	330	330	440	660	770	550	550	550	550	550
Exploration Percentage (%)	50	33,3	25	20	20	16,6	16,6	16,66	12,5	12,5
Positive Reinforcement	4	4	4	10	8	9	15	15	13	13
Negative Reinforcement	-4	-4	-4	-9	-8	-9	-15	-15	-13	-13
Learning Rate (α)	0,12	0,12	0,12	0,16	0,16	0,4	0,4	0,5	0,5	0,7
Discount Factor (γ)	0,9	0,9	0,9	0,7	0,7	0,8	0,8	0,8	0,8	0,7
Number of Recommendations:	1473	1538	1569	1388	1408	1303	1276	1342	1326	1361

O cenário de simulação foi composto por dez (10) estudantes e um conjunto com 110 DM para cada estudante. Foram criados dez (10) DM para cada um dos onze (11) tipos de atividade apresentados na Tabela 18. O objetivo de cada execução foi utilizar o sistema de recomendação até que cada estudante atingisse a pontuação máxima, neste caso, 110 considerando um ponto para cada DM resolvidos corretamente.

A Tabela 20 apresenta as diferentes configurações para o algoritmo *Q-Learning* em cada execução. O *Máx. Número de tentativas* define o número máximo de tentativas que um estudante faz para resolver o conjunto completo de DM. A Porcentagem de Exploração define a política do algoritmo *Q-Learning* para explorar o ambiente. Reforço positivo, reforço negativo, taxa de aprendizado e fator de desconto são parâmetros utilizados pelo Q-*Aprendizagem* para atualizar seu modelo ao receber reforço. O *Número de Recomendações* indica o total de recomendações que foram necessárias para que todos os estudantes realizassem todos os DM corretamente. Esse número está relacionado à eficiência da execução, e quanto menor, melhor.

As execuções seis (6) e sete (7) foram as que tiveram os melhores resultados, pois necessitaram de menos recomendações para atingir o score máximo. Essas execuções possuem configurações bastante similares, onde a única diferença são os parâmetros re-

ferentes ao reforço positivo e negativo. Há indícios de que um percentual de exploração relativamente baixo, como o caso de 16,6%, contribui para a eficiência da execução, o que se relaciona com a ideia de privilegiar o modelo do algoritmo *Q-Learning* ao invés de trabalhar aleatoriamente. As execuções dois (2) e três (3) foram as que tiveram os piores resultados. Observa-se que o percentual de exploração deles estão entre os maiores, o que indica um comportamento mais aleatório.

A simulação contribuiu para avaliação da funcionalidade de recomendação de DM de SM do Módulo Tutor sem considerar as capacidades iniciais de SM dos estudantes em relação ao EKM em tempo de projeto. Além disso, os resultados mostraram que o EKM pode ser devidamente operacionalizado pelo processo de simulação e pela funcionalidade de recomendação de DM de SM do Módulo Tutor.

Para destacar os benefícios do método de recomendação baseado no algoritmo *Q-Learning*, um cenário hipotético sem recomendação foi descrito usando dois tipos diferentes de estudantes. Esses estudantes foram modelados usando dados médios de estudantes de graduação em Ciência da Computação (TOMKIN; WEST; HERMAN, 2018).

O cenário é o mesmo: um conjunto de dez (10) estudantes e um conjunto com 110 DM, dez (10) DM diferentes para cada um dos onze (11) tipos de atividade apresentados na Tabela 18. Em cada ciclo, cada tipo de estudante tenta resolver todos os DM. Assume-se uma determinada habilidade de acertar DM para cada ciclo e, entre os ciclos, considera-se um incremento dessa habilidade.

A Tabela 21 apresenta esses valores e detalha o cenário hipotético considerando os tipos de estudantes. O *Número de Tentativas (tamanho do conjunto)* é o tamanho do conjunto de DM que o estudante precisa resolver em cada ciclo. *Increment (Inc.)* mostra o aumento da capacidade de acerto em um ciclo específico para um determinado tipo de estudante. O *Ability to Hit* mostra a capacidade total para resolver cada DM corretamente após o incremento da capacidade. *Número de Acertos (Nº Acertos)* é o número de DM que o estudante resolveu corretamente no ciclo. A tabela resume o desempenho geral de cada tipo de estudante, apresentando o total *Número de tentativas* para um e dez estudantes. Para ambos os tipos de estudantes, o incremento da habilidade de acerto ocorre de forma não linear. O tipo de estudante hipotético #1 começa com uma leve habilidade de acerto de 20% com um incremento gradual (0%, 10%, 10%, 10%, 20%, 30%). Em contrapartida, o estudante hipotético do tipo #2 começa com uma capacidade de acerto de 50% com diferentes incrementos graduais (0%, 15%, 15%, 20%) em cada ciclo.

A Figura 20 apresenta uma comparação entre os dois tipos de estudantes hipotéticos. A linha vertical do gráfico mostra o número acumulado de acertos de DM, cujo máximo é 110. A linha horizontal do gráfico mostra os ciclos em que os estudantes tentam resolver o conjunto completo de DM, onde o Tipo de Estudante #1 usa seis ciclos, e o tipo de estudante #2 usa quatro ciclos. O Tipo de Estudante #2 tem uma capacidade de resolução de DM mais alta do que o Tipo de Estudante #1, o que permite resolver DM

Tabela 21 – Cenário Hipotético sobre resoluções de DM

	Student Type #1				**Student Type #2**			
Cycle	**No. Attempts (set size)**	**Inc.**	**Ability to Hit**	**No. Hits**	**No. Attempts (set size)**	**Inc.**	**Ability to Hit**	**No. Hits**
1	110	0%	20%	22	110	0%	50%	55
2	88	10%	30%	26,4	55	15%	65%	35,75
3	61,6	10%	40%	24,64	19,25	15%	80%	15,4
4	36,96	10%	50%	18,48	3,85	20%	100%	3,85
5	18,48	20%	70%	12,93	-	-	-	-
6	5,54	30%	100%	5,54	-	-	-	-

No. of Attempts (1 student)	320,584		188,1
No. of Attempts (10 students)	3205,84		1881

mais rapidamente.

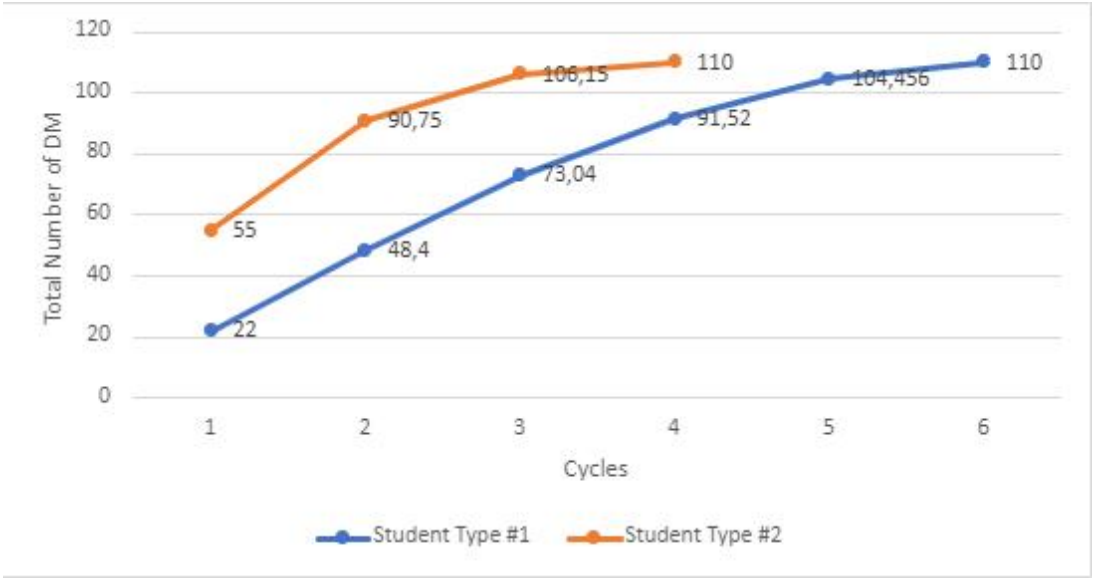

Figura 20 – Comparação entre os tipos de estudantes

O estudante do Tipo #1, que pode ser considerado um estudante com menor habilidade, precisaria de aproximadamente 320 tentativas para resolver os 110 DM, e o total para dez estudantes desse tipo exigiria cerca de 3206 tentativas para resolver esses 110 DM. O estudante do Tipo #2, que é considerado um estudante com habilidade média, precisaria de 188,1 tentativas para resolver esses 110 DM, e dez estudantes desse tipo precisariam de 1881 tentativas para resolver esses 110 DM.

A recomendação de DM e o EKM para ITS para SM apresentados, de acordo com os parâmetros de simulação descritos na Execução 7, permitem que dez estudantes virtuais resolvam os 110 DM com 1276 recomendações. Essas informações permitem estimar a melhoria que a proposta pode trazer para a eficiência do processo de ensino-aprendizagem SM ao considerar dois tipos de estudantes hipotéticos com diferentes capacidades em SM

de acordo com a tabela 21.

O método de recomendação e o EKM propostos neste trabalho reduzem em aproximadamente 60,2% o número de tentativas em relação aos estudantes do Tipo #1, então ao invés de 3206 tentativas, apenas 1276 são necessárias para resolver o DM completo definido por Todos os estudantes. Ao considerar os estudantes do Tipo #2, que possui uma habilidade maior desde o início, pôde-se observar uma redução de aproximadamente 32,2%, com 1276 tentativas de resolver todo o conjunto de DM contra 1881 tentativas.

Assim, os resultados deste trabalho superam o cenário hipotético descrito, cujos tipos de estudantes projetados atendem aos níveis de habilidade iniciante (Estudante do Tipo #1) e médio (Estudante do Tipo #2) em SM.

4.5 Avaliação do RL no Módulo Tutor do ITS para SM considerando as capacidades iniciais de SM dos estudantes

Esta tese propõe uma modelagem do algoritmo *Q-Learning* para recomendar DM de SM a partir do EKM do ITS. A avaliação realizada na seção 4.4 considerou somente estudantes sem nenhum conhecimentos sobre SM. No entanto, o fato de uma sala de aula real possuir estudantes com diferentes capacidades de SM motivou este experimento que avalia a recomendação de DM para estudantes com capacidades iniciais de SM variadas.

Este experimento refere-se à execução do algoritmo *Q-Learning*, responsável por recomendar DM de SM, para os 8489 registros de estudantes de SM descrito na seção 4.1. Este *Dataset* representa as variações da dimensão de conhecimento de SM dos estudantes.

A fila de estudantes para recomendações para este experimento é aleatória. Porém, como a biblioteca gera números pseudoaleatórios (PYTHON, 2023), e alguns estudantes terminam antes de outros, o progresso temporal segue em uma sequência próxima à lista de estudantes no *Dataset* de entrada. O modelo *Q-Learning*, representado pela tabela Q, é inicializado com dados aleatórios. Os DM de SM foram usados para ações e estados. Uma ação representa uma recomendação de DM de SM para um estudante dentre os disponíveis e um estado representa como o mesmo evoluiu neste conjunto de DM. Em Ações, foram utilizados os DM categorizados com a Taxonomia de Bloom e seus 11 tipos de atividades, conforme apresentado na seção 3.3.1 sobre o EKM através da Tabela 18. Para cada um destes tipos de atividades foram criados 10 DM, o que resultou em 110 DM. Como os estudantes começam com pontuação igual a zero (0), os 111 estados representam seu progresso na resolução de DM. São 110 ações e 111 estados.

Os parâmetros utilizados no Algoritmo *Q-Learning* foram: *Max. Number of Attempts*: 220; *Exploration Percentage* (%): 12,5; *Positive Reinforcement*: 13; *Negative Reinforcement*: -13; *Learning Rate* (α): 0,7; e *Discount Factor* (γ): 0,7. A escolha destes parâ-

metros foi realizada a partir de testes empíricos com execuções utilizando um conjunto menor de dados, com parâmetros adaptados do experimento apresentado na seção 4.4, visto que o *Dataset* usado no experimento da seção 4.4 não é o mesmo deste experimento. Esses testes foram realizados devido à necessidade de calibrar o sistema para encontrar os parâmetros adequados, pois esses parâmetros são definidos empiricamente para o contexto de aplicação do algoritmo *Q-Learning* (GUPTA; ROY; DUTT, 2021), que é representado pelo *Dataset* de entrada.

Foram realizadas normalizações nos atributos de Conhecimento da Dimensão SM do *Dataset* para transformar as pontuações da escala original (0 a 100) para a escala de DM para SM (0 a seis) e permitir explorar diferentes cenários de capacidades de SM dos estudantes. Essa medida de dificuldade máxima, apesar de ter sido estimada com a Taxonomia de Bloom para realizar os experimentos desta Tese, pode ser levantada de outras maneiras, pois o processo de simulação do ambiente do ITS para SM apenas precisa dessa informação atualizada para realizar seu trabalho. O *PERCENT* indica qual percentual da pontuação máxima do DM, de seis, o *Dataset* possui estudantes com capacidades mais significativas na Dimensão de Conhecimento de SM dos estudantes, conforme a Tabela 22.

Como existem nove normalizações e 30 execuções para cada normalização, este experimento tem um total de 270 execuções. Este número de execuções alto foi escolhido para trabalhar com a média da eficácia do método recomendado e evitar ruídos nesses resultados. O código fonte deste experimento está disponível (FRANCISCO, 2023c) e os resultados são mostrados na Tabela 23.

Tabela 22 – Normalizações

PERCENT	**SM Score Max para DM**	**SM Score Max NORM**
100%	6	6
80%	6	4,8
60%	6	3,6
50%	6	3
40%	6	2,4
20%	6	1,2
10%	6	0,6
5%	6	0,3
1%	6	0,06

A métrica *Experiment Weighted Performance Indicator* (EWPI) refere-se à *Accuracy* dos experimentos de recomendação de DM de SM realizados no âmbito desta Tese. Ela indica o percentual de recomendações bem sucedidas, i.e. aquelas que referem-se aos acertos dos estudantes, em relação ao número total de recomendações.

O objetivo deste experimento é a otimização do EWPI, apresentado na equação 11, a partir da execução do Módulo Tutor. A equação 11 possui os seguintes parâmetros:

- n - número de estudantes
- fDesemp - uma função que retorna o desempenho de um determinado estudante do conjunto E de estudantes, i.e. fDesemp(i) - ED[i]
- fNRecom - uma função que retorna o número de recomendações para um determinado estudante do conjunto E de estudantes, i.e. fNRecom(i) – ER[i]
- E = e1, e2, ..., en) - o conjunto de estudantes
- ED = d1, d2, ..., dn - o conjunto de desempenhos dos estudantes (número de tarefas realizadas corretamente)
- ER = r1, r2, ..., rn - o conjunto com os números de recomendações feitas aos estudantes

$$EWPI = \frac{\sum_{i=1}^{n} fDesemp(i)}{\sum_{i=1}^{n} nRecom(i)} \tag{11}$$

Tabela 23 – Resultados do Experimento de RL no Módulo Tutor do ITS

NORM %	**No. HIT**	**No. REC**	**Seconds**
1%	700427.53	1867210.7	184
5%	721666.08	1929515.8	184
10%	721207.8	1931488.7	184
20%	744476.46	1930809.69	187
40%	790078.61	1899425.12	235
50%	851696.05	1657925.27	245
60%	927857.10	1264244.40	199
80%	961122.43	1044407.21	115
100%	964719.78	1013193.04	95
EWPI	0.5078511620937646		

O computador usado para realizar os experimentos nesta seção e na seção 4.6 possui as seguintes características: CPU Intel Core i5-7200U 2.71 GHZ, 8 GB de RAM e 250 GB de disco rígido. Os tempos de processamento apresentados nas Tabelas 23 e 26 referem-se à média de tempo de execução para cada normalização.

As Figuras 21, 22, 23 e 24 mostram a eficiência do método de recomendação de DM de SM a partir de uma perspectiva que considera a evolução temporal do modelo de decisão do algoritmo *Q-Learning* e diferentes cenários iniciais de capacidades de SM de estudantes representados pela normalização. Conforme o experimento é executado e esse modelo é atualizado, sua performance é melhorada de modo que o número de acertos se aproxima do número de recomendações. A Figura 21 enfatiza todos os cenários iniciais de capacidades de SM e as Figuras 23 e 24 enfatizam os cenários iniciais de capacidades com variação de média a alta de SM.

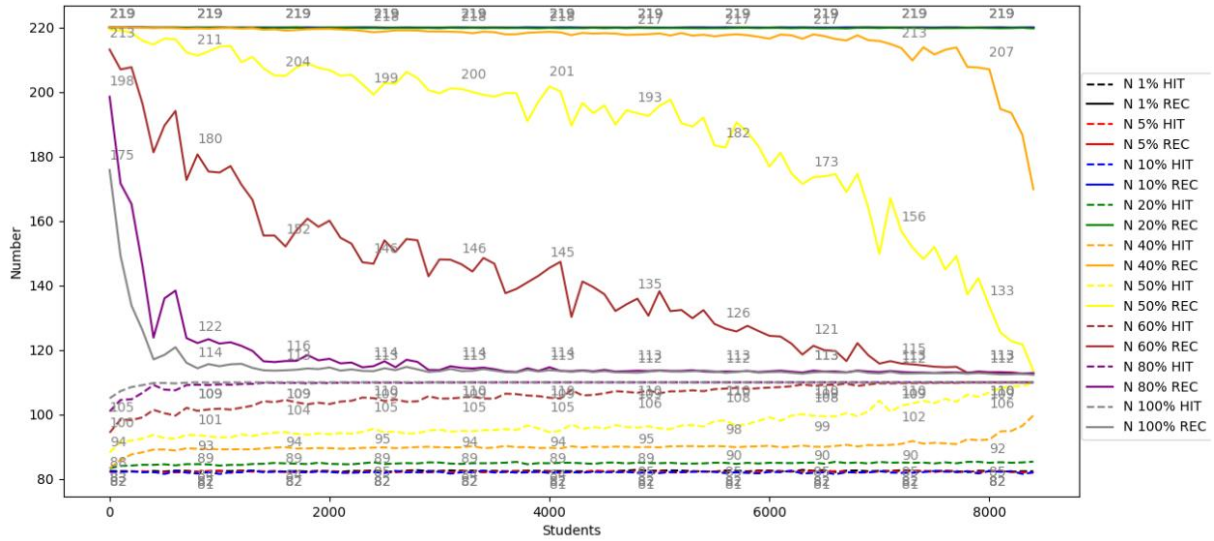

Figura 21 – Execução do experimento com todas as normalizações

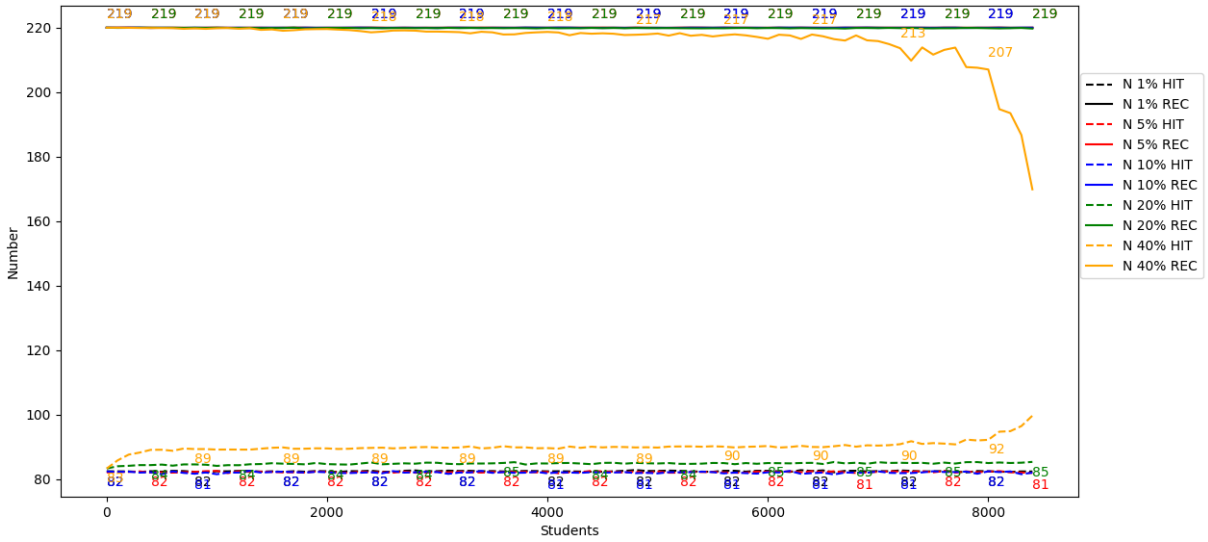

Figura 22 – Execução do experimento com as normalizações de 1% a 40%

A definição de normalização utilizada nesta pesquisa refere-se à capacidade máxima de resolução de DM de SM dos estudantes obtidos via *Dataset* considerando os quatro atributos da dimensão de conhecimento de SM. O valor 100% indica que os estudantes obtidos com a maior capacidade em SM terão suas capacidades normalizadas em relação à dificuldade máxima de um DM de SM, que é seis. À medida que a normalização diminui, os estudantes obtidos têm menores capacidades de resolução de problemas de SM, o que faz com que o gráfico da Figura 24 tenha melhor desempenho. Esta relação também pode ser verificada na Tabela 23.

As linhas sólidas traçadas nestes gráficos indicam o número médio de recomendações e as linhas tracejadas indicam o número médio de acertos. A medida que o experimento

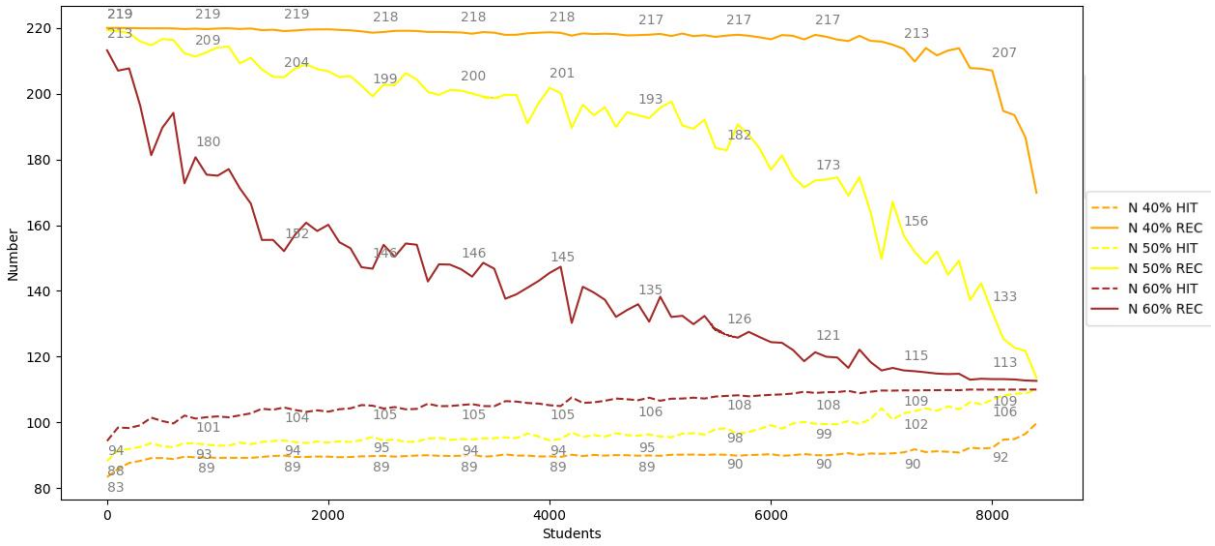

Figura 23 – Execução do experimento com as normalizações de 40% a 60%

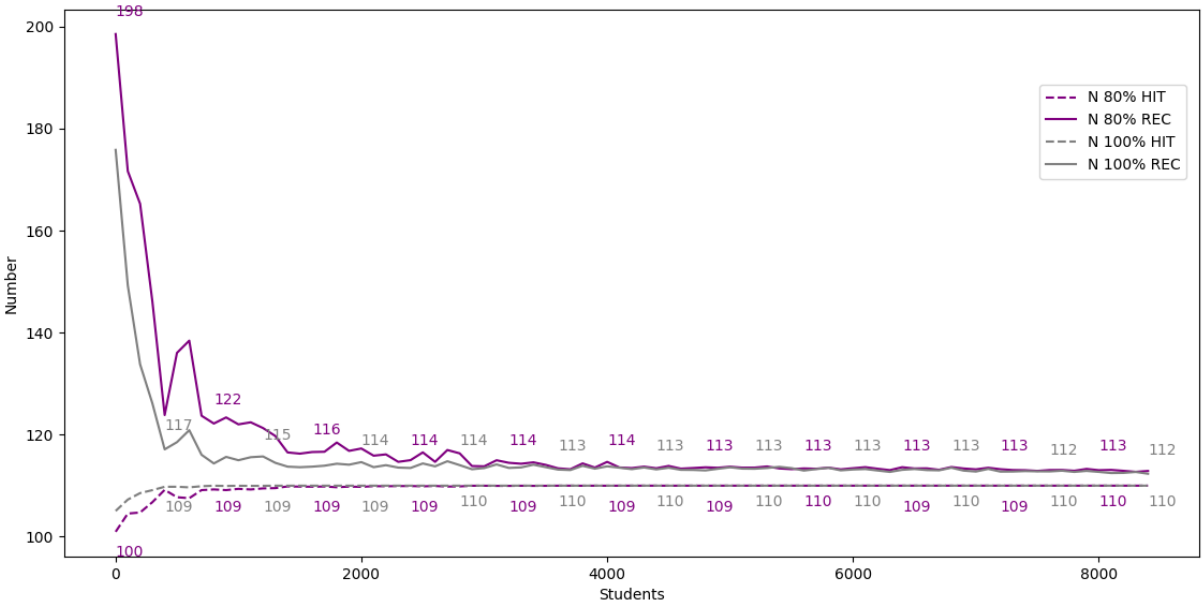

Figura 24 – Execução do experimento com as normalizações de 80% a 100%

avança em reação à execução, o número de recomendações diminui e o número de acertos aumenta, pois o algoritmo já está mais treinado, o que pode ser visto na Figura 21.

4.6 Avaliação da Integração entre o Módulo do Estudante e o Módulo Tutor

Considerando que a representação do estudante apoia o ITS em ações adaptadas ao estudante (ALMASRI et al., 2019), esta seção tem como objetivo responder à RQ "A identificação de tipos de estudantes de SM contribui para a recomendação de DM de SM?".

Cada turma pode ser composta de um ou mais tipos de estudantes. O Módulo Tutor cria uma instância de *Q-Learning* para cada tipo de estudante gerado pelo Módulo do Estudante. A mesma configuração de ações e estados das seções 4.5 foi utilizada nesta seção. A mesma configuração de ações e estados da seção 4.5 foi utilizada nesta seção.

Conforme apresentado na Figura 10, no início da utilização do ITS, logo após os tipos de estudantes terem sido gerados pelo Módulo do Estudante, o Módulo Tutor inicializa tabelas Q das instâncias do *Q-Learning* com dados aleatórios, representando que o modelo de decisão ainda não tem experiência.

Após algum tempo de uso do ITS, cada tabela Q relacionada a uma instância específica do *Q-Learning* terá um modelo de decisão representando o conhecimento obtido pelo algoritmo *Q-Learning* até aquele momento por meio de RL. Com isso, o Módulo Tutor, através das instâncias do algoritmo *Q-Learning* com as tabelas Q auxiliadas pelo RL, terá um modelo de decisão específico para cada tipo de estudante.

Este experimento usou as mesmas configurações, parâmetros e normalizações apresentadas na seção 4.5 devido à necessidade de comparar esses dois experimentos. O mesmo conjunto de dados do estudante também foi usado. Porém, foi feita uma alteração onde cada *cluster* (tipo de estudante de SM) possui uma tabela *Q-Learning*. Isso faz com que cada *cluster* tenha um modelo de recomendação de DM para seus respectivos estudantes.

A mesma normalização do experimento apresentado na Seção 4.5 foi usada. O número de execuções do algoritmo *Q-Learning* também foi o mesmo de antes, 270.

As Tabelas 24 e 25 apresentam a quantidade e o percentual de estudantes com capacidades em SM variando de baixa para média e de média para alta para cada normalização. Os estudantes possuem um score para cada um dos quatro atributos da dimensão de conhecimento de SM. As inequações envolvem o score e a limitação referente à metade da dificuldade máxima para o referido atributo da dimensão de conhecimento de SM.

Percebe-se que as normalizações de 40% e 50% tendem a criar cenários onde os estudantes possuem habilidades de SM que variam de baixas a médias. Isso pode ser observado ao analisar os percentuais de estudantes com pontuações de SM menores ou iguais à metade da pontuação máxima para um determinado atributo de *DimensionKnowledgeSM*.

Tabela 24 – Distribuição dos estudantes em relação ao Score médio para *softwareUnderstanding* e *understandingConcepts*

NORM	**softwareUnderstanding**		**understandingConcepts**	
	Score <= 2	**Score >2**	**Score <= 1**	**Score >1**
100%	3211 (37,83%)	5278 (62,17%)	111 (1,31%)	8378 (98,69%)
80%	4960 (58,43%)	3529 (41,57%)	234 (2,76%)	8255 (97,24%)
60%	7138 (84,09%)	1351 (15,91%)	680 (8,01%)	7809 (91,99%)
50%	8048 (94,81%)	441 (5,19%)	1390 (16,37%)	7099 (83,63%)
40%	8470 (99,78%)	19 (0,22%)	2988 (35,20%)	5501 (64,80%)
20%	8489 (100,00%)	0 (0%)	8473 (99,81%)	16 (0,19%)
10%	8489 (100,00%)	0 (0%)	8489 (100,00%)	0 (0%)
5%	8489 (100,00%)	0 (0%)	8489 (100,00%)	0 (0%)
1%	8489 (100,00%)	0 (0%)	8489 (100,00%)	0 (0%)

Tabela 25 – Distribuição dos estudantes em relação ao Score médio para *practiceSM* e *testSM*

NORM	**practiceSM**		**testSM**	
	Score <= 3	**Score >3**	**Score <= 2,5**	**Score >2,5**
100%	5096 (60,03%)	3393 (39,97%)	3826 (45,07%)	4663 (54,93%)
80%	7098 (83,61%)	1391 (16,39%)	5513 (64,94%)	2976 (35,06%)
60%	8457 (99,62%)	32 (0,38%)	7892 (92,97%)	597 (7,03%)
50%	8489 (100%)	0 (0%)	8459 (99,65%)	30 (0,35%)
40%	8489 (100%)	0 (0%)	8489 (100,00%)	0 (0%)
20%	8489 (100%)	0 (0%)	8489 (100,00%)	0 (0%)
10%	8489 (100%)	0 (0%)	8489 (100,00%)	0 (0%)
5%	8489 (100%)	0 (0%)	8489 (100,00%)	0 (0%)
1%	8489 (100%)	0 (0%)	8489 (100,00%)	0 (0%)

Os atributos *softwareUnderstanding, practiceSM* e *testSM*, com normalizações de 40% e 50%, demonstram claramente esta tendência, pois *softwareUnderstanding* tem com 40% o valor de 99,78% e com 50% o valor de 94,81%, *practiceSM* tem com 40% o valor de 100% e com 50% o valor de 100%, *testSM* tem com 40% o valor de 100% e com 50% o valor de 99,65% de estudantes que atendem essa tendência. O atributo *Understanding-Concepts*, por ter pontuação máxima para o DM com valor dois (2), possui valor "muito baixo" como metade desse valor, i.e. um (1), o que faz com que os scores de SM dos estudantes normalizados sejam muito superiores a esse valor e, consequentemente, não atenda diretamente a tendência, pois neste caso a normalização de 20% (um nível abaixo de 40%) vai ao encontro da tendência.

O código fonte deste experimento está disponível (FRANCISCO, 2023d) e os resultados são mostrados na Tabela 26. Comparações foram feitas entre o experimento apresentado na seção 4.5 e este experimento para avaliar as diferenças no EWPI, ou seja, o EWPI deste experimento - EWPI apresentado na seção 4.5. Como ambos os experimentos realizaram 30 execuções, consideramos a média das 30 execuções de cada experimento.

Tabela 26 – Resultados do Experimento de Integração entre os Módulos do Estudante e Tutor

NORM %	**No. HIT**	**No. REC**	**Seconds**
1 %	755764.76	1866636.2	218
5 %	779559.35	1928874.94	207
10 %	780601.84	1930819.96	189
20 %	812949.19	1927509.96	190
40 %	868730.006	1708796.63	189
50 %	909581.6	1462569.72	170
60 %	938525.25	1229930.72	159
80 %	960396.14	1062414.32	119
100 %	964456.93	1016296.77	97
EWPI	0.5497840658025246		

A Figura 25 e a Tabela 27 têm como objetivo avaliar qual dos dois experimentos otimiza o EWPI, possibilitando analisar os ganhos de desempenho na recomendação do DM de que a integração entre o Módulo do Estudante e o Módulo Tutor.

Tabela 27 – Diferenças representando ganhos de EWPI para diferentes normalizações

NORM %	**EWPIInteg**[a]	**EWPIMTutor**[b]	**EWPIInteg**[a] **- EWPIMTutor**[b]
1%	0,4048806185	0,3751197019	0,0297609166
5%	0,4041280685	0,3739772187	0,03015084982
10%	0,4042897394	0,3735242889	0,03076545048
20%	0,422282721	0,3859978286	0,03628489236
40%	0,5114909204	0,4170254557	0,09446546474
50%	0,6265648083	0,5176942186	0,1088705898
60%	0,7689304847	0,7441873577	0,02474312704
80%	0,9096550979	0,9280993627	-0,01844426475
100%	0,9506187728	0,9532955543	-0,002676781519

[a] EWPIInteg: EWPI of Integration between Student Module and Tutor Module (this section)

[b] EWPIMTutor: EWPI of the RL Experiment in the ITS Tutor Module (section 4.5)

A normalização de 80% a 100% representam estudantes com capacidade de resolução de DM muito alta. Nestes casos, o algoritmo *Q-Learning* não apresenta melhorias de desempenho quando integrado aos *clusters* gerados pelo Módulo do Estudante. Os estudantes que atingem naturalmente um maior volume de DM são menos dependentes de padrões sequenciais gerados pelas recomendações de DM. Assim, utilizar o conjunto completo de estudantes em detrimento dos *clusters* trouxe melhores resultados, pois o conjunto todo representa mais a heterogeneidade da habilidade de SM dos estudantes.

Uma curva pode ser vista no gráfico apresentado nas Figuras 26 e 27, onde a normalização com 40% e 50% traz os melhores desempenhos usando *clusters* para criar modelos de *Q-Learning*. São diferentes agrupamentos de estudantes com habilidades que variam de baixa a média, o que tende a auxiliar o algoritmo *Q-Learning* a aprender os padrões

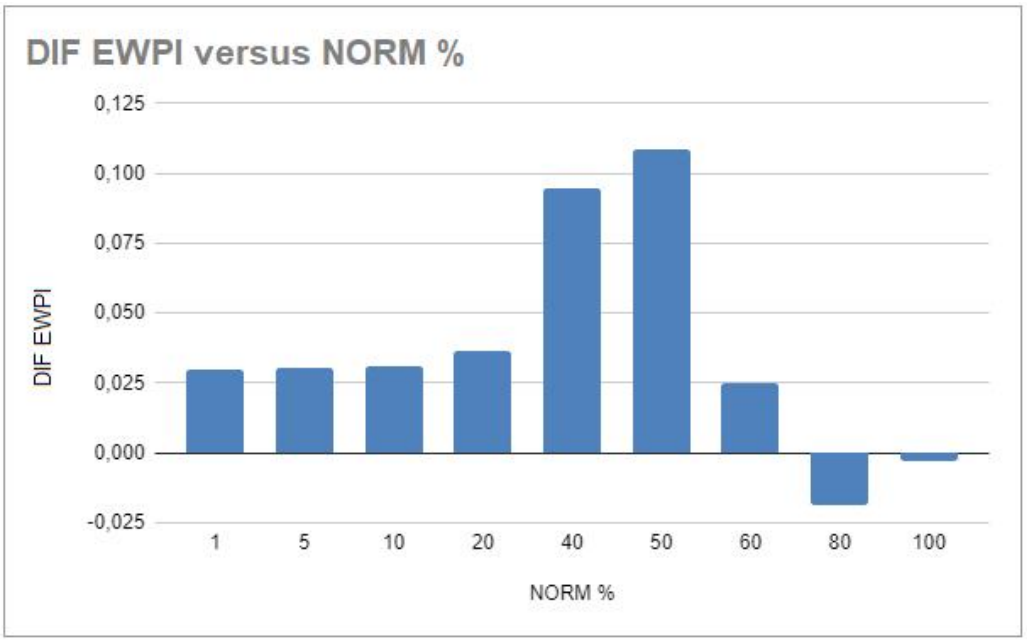

Figura 25 – Ganhos de EWPI com diferentes normalizações para o Experimento 4.6 em comparação com o Experimento 4.5

sequenciais por considerar a heterogeneidade de cada agrupamento associada à falta de estudantes altamente competentes, o que afeta a dificuldade do Algoritmo *Q-Learning* para encontrar os padrões. Este cenário se aproxima da ideia de uma sala de aula real.

A normalização de 50% representa o cenário de mais estudantes com melhor desempenho. Este cenário contribui para o funcionamento do algoritmo *Q-Learning*, que precisa construir um modelo capaz de recomendar os padrões sequenciais de DM. A construção deste modelo exige que os estudantes desenvolvam competências de SM, uma vez que a RL observa o comportamento dos estudantes em resposta às recomendações feitas pelo sistema. A normalização de 50% e o uso de *clusters* faz com que os estudantes representem o cenário adequado de capacidades de SM, e os padrões de recomendação sejam criados explicitamente para os *clusters*, o que melhora a eficiência do sistema de recomendação, conforme Figura 26. É importante considerar que para o contexto desta Tese, um sistema de recomendação de DM de SM eficiente gera recomendações que tendem a serem mais assertivas, i.e. pode ser percebido a partir de experimentos que aproximem o número de recomendações do número de acertos pelos estudantes.

Este cenário se assemelha a uma sala de aula real onde os estudantes desenvolveriam habilidades de SM. Isso envolve conjuntos de estudantes com habilidades de SM de baixas a médias. É fundamental considerar que salas de aula com capacidades muito baixas ou muito altas, nestes experimentos, não não contribuem para o treinamento do algoritmo de RL avaliado. No entanto, uma vez que o algoritmo de RL esteja treinado e seu modelo esteja gerado, ele pode ser aplicado a diferentes perfis iniciais de capacidades de SM de estudantes devido ao fato de que ele modelou os padrões de sequências adequadas de DM

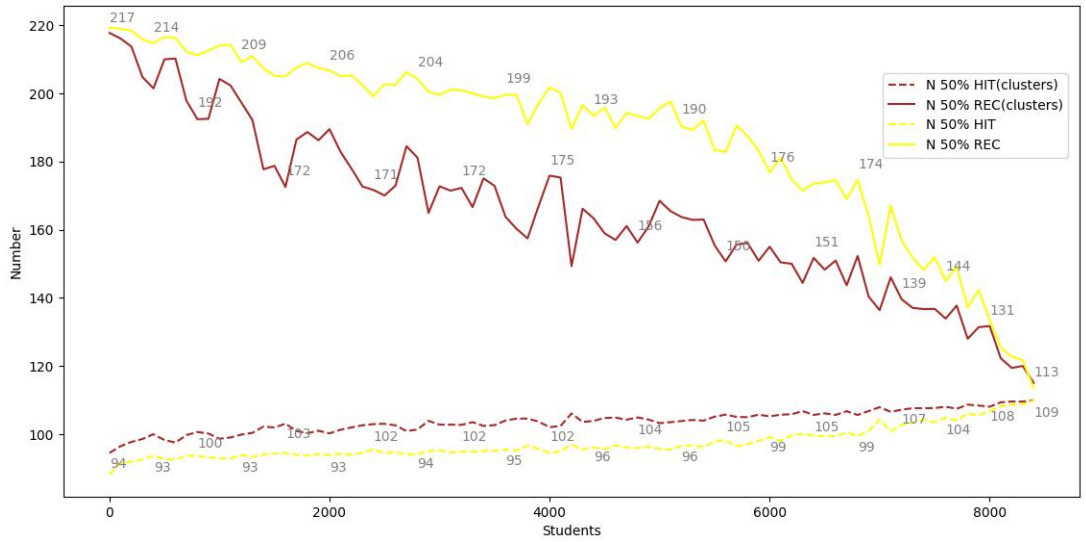

Figura 26 – Comparação do Experimento 4.6 em relação ao Experimento 4.5 com normalização de 50%

de SM para cada nível de capacidade de SM dos estudantes considerando inclusive os diferentes tipos de estudantes em relação a SM.

A Figura 26 mostra o benefício de usar *clusters* para criar e usar o modelo de recomendação *Q-Learning*. As linhas vermelhas, comparadas às amarelas, mostram uma redução gradual no *Number of Recommendations* (REC) e um aumento gradual no *Number of Hits* (HIT) com diferenças significativas, o que também é exibido na Figura 25 e Tabela 27.

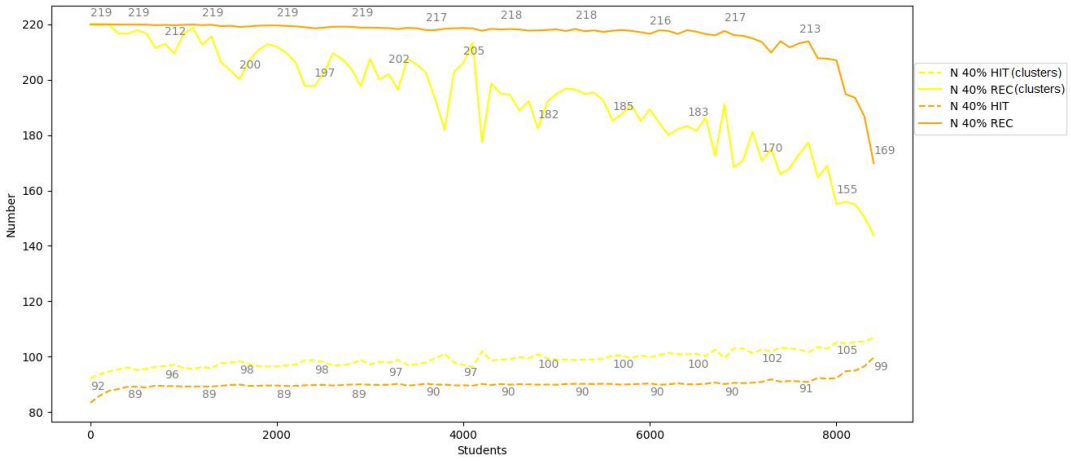

Figura 27 – Comparação do Experimento 4.6 em relação ao Experimento 4.5 com normalização de 40%

As Figuras 27 e 28 também mostram os ganhos com o uso de clusters; entretanto, como já mostrado na Tabela 27, esses ganhos não são mais significativos que a normalização de 50%.

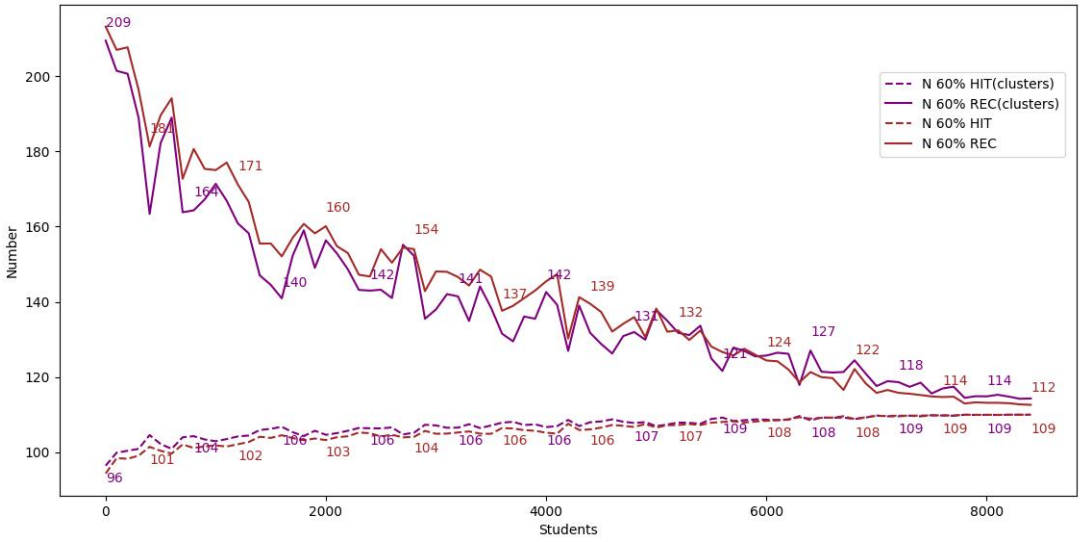

Figura 28 – Comparação do Experimento 4.6 em relação ao Experimento 4.5 com normalização de 60%

Conforme pode ser analisado na Figura 25, os resultados podem ser categorizados com relação à normalização em: *clusters* trazem ganhos pouco significativos, *clusters* trazem ganhos significativos, *clusters* não apresentam ganhos.

As normalizações de 1%, 5%, 10%, 20% e 60% geraram resultados que estão na categoria " *clusters* trazem ganhos pouco significativos". Os Gráficos apresentados nas Figuras 28, 29 e 30 representam esses resultados, porém essa representação gráfica é mais perceptível no gráficos das Figuras 28 e 30.

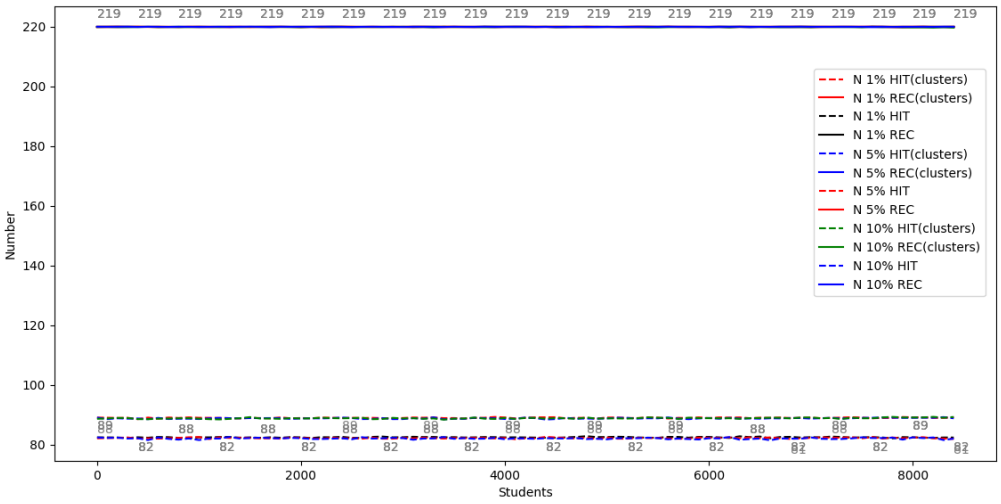

Figura 29 – Comparação do Experimento 4.6 em relação ao Experimento 4.5 com normalizações de 1% a 10%

A categoria de resultados "clusters trazem ganhos significativos" possui os resultados relacionados às normalizações de 40% e 50%, apresentadas nas Figuras 26 e 27. A categoria " *clusters* não apresentam ganhos" são representadas pelas normalizações de 80% e

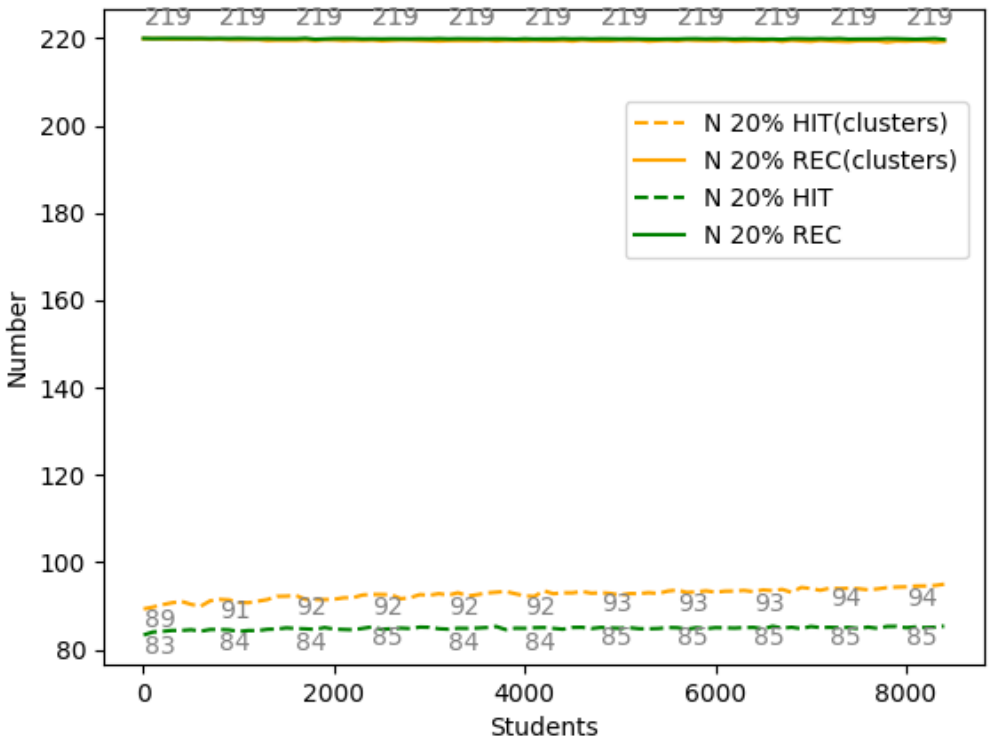

Figura 30 – Comparação do Experimento 4.6 em relação ao Experimento 4.5 com normalização de 20%

100% e pelas Figuras 31 e 32.

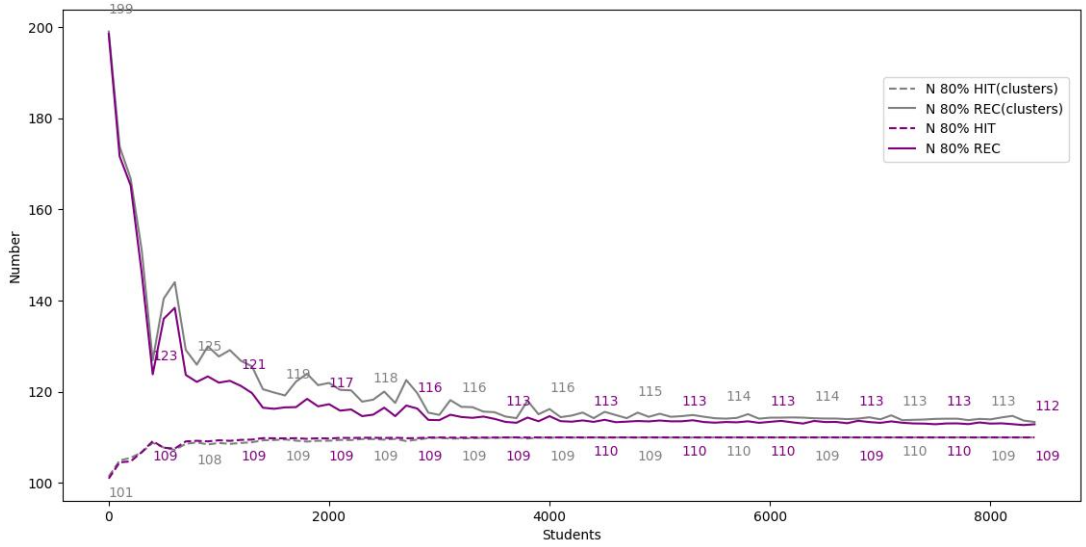

Figura 31 – Comparação do Experimento 4.6 em relação ao Experimento 4.5 com normalização de 80%

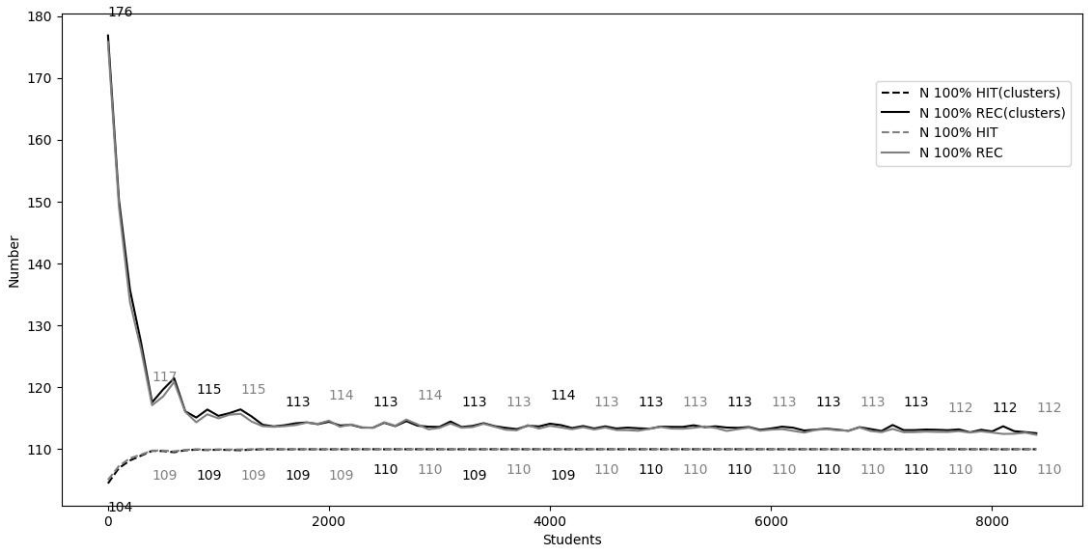

Figura 32 – Comparação do Experimento 4.6 em relação ao Experimento 4.5 com normalização de 100%

4.7 Discussão

Esta seção apresenta uma discussão acerca dos resultados desta pesquisa através de uma análise dos mesmos com relação às RQ e também com relação aos Trabalhos Correlatos.

4.7.1 Discussão dos Resultados

A discussão dos resultados segue a mesma sequência das RQ e é apresentada a seguir.

4.7.1.1 RQ 1: Como projetar a arquitetura de um ITS para atender o ensino-aprendizagem de SM?

Uma abordagem baseada em componentes padrão de um ITS (Módulos do Estudante, Módulo Tutor e EKM) (ALKHATLAN; KALITA,) com o apoio de ML, da representação do conteúdo de SM e das capacidades de SM de estudantes realizada nesta Tese permitiu propor uma arquitetura de ITS para SM.

4.7.1.2 RQ 2: Como ML pode contribuir com a identificação de tipos de estudantes de SM no Módulo do Estudante?

A Clusterização aplicada aos dados da dimensão de conhecimento de SM de estudantes em conjunto com condições de avaliação dos resultados produzidos possibilitam identificar os tipos de estudantes de SM.

O Experimento apresentado na seção 4.3 detalha a resposta dessa RQ. A estratégia proposta executa algoritmos de Clusterização que permitem a definição prévia do número

de *clusters* desejado em dados de estudantes com relação à dimensão de conhecimento de SM e utiliza condições de avaliação propostas a partir da relação entre heterogeneidade e número de *clusters* e do coeficiente de Silhouette com Distância Euclidiana.

O algoritmo K-Means com quatro (4) *clusters* foi a opção que melhor atendeu a essas condições.

4.7.1.3 RQ 3: Como ML pode ser utilizado no Módulo Tutor para recomendar DM de SM?

O EKM para o domínio de SM e o algoritmo *Q-Learning* contribuem para a criação de um método para recomendar DM para um ITS para SM, conforme o Experimento apresentado na seção 4.4.

A dimensão do conhecimento de SM, descrita no EKM, foi utilizada para a construção de DM virtuais utilizados na modelagem do algoritmo de RL *Q-Learning* para a recomendação de DM de SM. A avaliação dessa proposta foi realizada a partir de um método de simulação da resolução de DM de SM por estudantes, que foi construído também baseado no EKM do ITS para SM.

Os resultados desse experimento indicam que o Módulo Tutor realiza recomendações de DM de SM e aprimora seu modelo no decorrer do tempo. A execução mais eficiente realizou 1276 recomendações de DM de SM para um contexto em que 10 (dez) estudantes respondem a 110 DM cada. Como para esses 10 (dez) estudantes resolverem todos os DM seria necessário pelo menos 1100 tentativas, o resultado mostra que o método proposto é bastante eficiente em relação ao modelo de simulação proposto. É importante considerar que este foi o primeiro experimento realizado, com o objetivo de avaliar tecnicamente o algoritmo de RL *Q-Learning* para o contexto simplificado do ITS para SM, com um número baixo de estudantes virtuais, com desempenho e capacidades lineares, e de DM de SM, o que facilitou o treinamento do algoritmo *Q-Learning* e trouxe um desempenho alto.

Esta pesquisa apresenta também uma comparação com dois tipos de estudantes de SM hipotéticos com capacidades de dois tipos (iniciante e média). A comparação mostra que o método proposto para recomendação de DM de SM tem uma melhoria de 60,2% na capacidade de SM de seus estudantes virtuais simulados com relação a estudantes do tipo iniciante e tem também uma melhoria de 32,2% em relação a estudantes do tipo médio, corroborando para mostrar a eficiência do método proposto.

4.7.1.4 RQ 4: Como ML pode ser utilizado no Módulo Tutor para recomendar DM de SM em um contexto que considera as variações iniciais no conhecimento de SM dos estudantes?

O EKM para o domínio de SM e o algoritmo *Q-Learning* contribuem para a criação de um método para recomendar DM para um ITS para SM capaz de considerar as variações

nas capacidades em SM dos estudantes, conforme o Experimento apresentado na seção 4.5.

O Experimento apresentado na seção 4.5, além de utilizar a simulação computacional de estudantes resolvendo DM de SM, utilizou um *Dataset* que possui dados que representam capacidades de SM de 8489 estudantes de SM para trabalhar com as variações iniciais nas capacidades de SM dos estudantes. O experimento descreve como calibrou os parâmetros do sistema, propõe e utiliza normalizações para representarem diferentes cenários de capacidades de SM de estudantes, é executado 30 vezes para cada normalização e calcula o indicador de desempenho EWPI para o experimento. Como as normalizações aumentam gradativamente as capacidades dos estudantes (de 1% a 100%), os resultados sugerem, conforme Tabela 23, um aumento explícito no HIT e uma redução no REC que não é tão explícita mas também ocorre. O EWPI do experimento é 0.5078511620937646, o que indica que os estudantes acetam em cerca de 50% das tentativas em relação ao número de recomendações de DM de SM.

4.7.1.5 RQ 5: A identificação dos tipos de estudantes de SM contribui para a recomendação de DM de SM?

O Experimento apresentado na seção 4.6 explorou a ideia de ser possível melhorar o desempenho da recomendação de DM com a criação de instâncias do algoritmo *Q-Learning* específicas para cada tipo de estudante de SM obtida pela clusterização (Módulo do Estudante) e mostrou que essa estratégia de integração trouxe ganhos para a eficiência da recomendação de DM de SM. Isso pode ser verificado na comparação entre os Experimentos das seções 4.5 e 4.6. Os ganhos do Experimento da seção 4.6 em relação ao Experimento da seção 4.5 podem ser visualizados no gráfico apresentado na Figura 25, que indica melhorias no indicador EWPI em quase todas as estratégias de normalização exploradas, com exceção somente nas normalizações de 80 e 100%.

Refletir sobre os resultados desses experimentos possibilitaram construir um direcionamento sobre o uso desse ITS para SM. As normalizações de 40 e 50% apresentadas na Figura 25 simulam um contexto onde os melhores estudantes tem aproximadamente 40% e 50% de capacidade em relação à dificuldade máxima exigida pelos DM de SM. Ou seja, se assemelha a uma sala de aula que possui estudantes com capacidade que variam entre baixa e média. O ITS precisa desse contexto para que o processo de aprendizado do algoritmo de RL construa um modelo mais eficiente para cada tipo de estudante de SM. Esse cenário se assemelha a uma sala de aula real, que possui uma certa heterogeneidade de capacidade entre os estudantes.

Esse cenário de comparação dos dois Experimentos entre as seções 4.5 e 4.6 indica que a integração entre os Módulos do Estudante e Tutor possibilita uma melhoria de 9% e 10% na eficiência da recomendação de DM considerando as normalizações de 40% e 50%.

4.7.2 Comparação com a Literatura

Os Trabalhos Correlatos sobre "ITS para SM", descritos na seção 2.2.2, trazem desafios de pesquisa que foram abordados nesta Tese. Conforme a Tabela 28, nenhum dos trabalhos abordou até então havia abordado ML no Módulo do Estudante para ITS para SM, apenas dois (2) de cinco (5) abordaram ML no Módulo Tutor e nenhum deles abordou a integração entre esses dois módulos. Em comparação com esses trabalhos, esta tese aborda ML nos Módulos do Estudante e Tutor do ITS para SM e como pode ser feita a integração entre esses dois módulos por meio de experimentos detalhados, contribuindo para melhorar o detalhamento de como usar ML na construção desses sistemas (FRANCISCO; SILVA, 2022b). Além disso, esta tese aborda a área de SM de maneira ampla no EKM do ITS para SM.

Tabela 28 – Comparação deste trabalho com trabalhos sobre ITS para SM

WORK	**STM**[a]	**ML in STM**	**TM**[b]	**ML in TM**	**I**[c]	**ML technique**	**SM topics** [d]
(LUBURIć et al., 2022a)	YES	NO	YES	NO	NO	-	REF; MAN.
(LUBURIć et al., 2022b)	NO	NO	YES	NO	NO	-	REF; MAN.
(CARTER; BLANK, 2013)	YES	NO	YES	YES	NO	CBR	DEP.
(ALSHAIKH; TAMANG; RUS, 2021)	NO	NO	YES	NO	NO	-	COMP.
(FRANCISCO; SILVA, 2022a)	NO	NO	YES	YES	NO	*Q-Learning*	COMP; PRAT; TEST; CONS.
THIS WORK	YES	YES	YES	YES	YES	*Q-Learning*; K-Means	COMP; PRAT; TEST; CONS.

[a] STM: Student Module

[b] TM: Tutor Module

[c] I: Integration between Student Module and Tutor Module

[d] SM topics: Refactoring (RE), Source Code Maintainability (MA), Debugging (DE), Software Understanding (COMP), SM Practice (PRA), Tests for MS (TST) and Comprehension of Concepts in SM (CON)

Esta pesquisa contribui também para outros problemas desta linha de pesquisa. Dificuldades relacionadas à representação do estado do conhecimento do estudante, à coleta de dados para sistemas inteligentes e ao seu gerenciamento considerando os respectivos modelos de ML foram levantadas por (ALSHAIKH; HEWAHI, 2021) no âmbito de ML e Inteligência Artificial na construção de ITS. O gerenciamento do ITS envolvendo o monitoramento das técnicas de ML e sua relação com os profissionais que farão essa tarefa foi apresentado como desafio por (ROLL; WYLIE, 2016). Esta tese propõe uma abordagem baseada em simulação computacional e em um *Dataset* de estudantes com capacidades de SM que contribui com as dificuldades relacionada a coleta de dados para sistemas inteligentes. Ela também propõe uma modelagem baseada no conceito de ITS que possibilita aos seus módulos operacionalizarem o estudante com os algoritmos de Clusterização e RL de modo a representar o seu estado de conhecimento, permitindo ao ITS trabalhar de forma adaptativa, e avalia o comportamento de algoritmos de ML utilizados de forma a

contribuir com a gestão do ITS para SM visando melhorar o seu desempenho em tempo de execução.

4.8 Ameaças à Validade

Esta pesquisa desenvolve módulos do ITS para SM em tempo de projeto e lida com desafios apresentados na seção 2.2.2, como o baixo detalhamento de como os modelos de ML são construídos nos ITS (FRANCISCO; SILVA, 2022b), a baixa cobertura de temas de SM abordados e o baixo número de trabalhos que exploram ML com detalhes nos Módulos do Estudante e Tutor do ITS para SM apresentados na Tabela 4.

O custo computacional relacionado à ML nos ITS (ROLL; WYLIE, 2016) e sua relação com a indisponibilidade de *Datasets* para reúso em outras pesquisas (FRANCISCO; SILVA, 2022b) foram fatores que tornaram necessário criar um planejamento sobre como avaliar esta proposta, que enfatiza ML no ITS para SM.

Nesse sentido, foi utilizada uma estratégia de avaliação com dados estudantes de SM baseados no *Dataset* do Enade para Ciência da Computação do ano de 2017 (INEP, 2022) e em simulação. Para inferir informações sobre capacidades de SM de estudantes foram realizados o uso de notas dos estudantes no Enade e a complementação para o domínio de SM conforme identificado na literatura sobre a natureza de tarefas em SM (ANQUETIL et al., 2007; WANG; KING; WICKBURG, 1999) através de números aleatórios em um contexto muito bem definido, que é apresentado na seção 4.1. Essas simplificações foram importantes para avaliar se os métodos de ML utilizados no ITS para SM conseguem construir modelos eficientes através do aprendizado em relação ao contexto de dados trabalhado.

A possibilidade do ITS armazenar a capacidade inicial de estudantes de SM, que pode ser identificada inicialmente através da resolução de um teste de aptidão, e também a evolução do desempenho da dimensão de conhecimento de SM desses estudantes trouxe a possibilidade do processo de simulação proposto inferir se um estudante consegue ou não resolver determinado DM de SM.

Esse método de avaliação mostrou-se viável para lidar com a complexidade de trabalhar com modelos de ML no contexto educacional durante a construção do ITS para SM. Ele permitiu realizar diversos refinamentos nos algoritmos de ML utilizados, e.g. quanto aos parâmetros no âmbito da RL e da Clusterização e também quanto à escolha de um algoritmo de Clusterização adequado, através de um volume bastante alto de experimentos.

O número de execuções foi bastante alto para encontrar os melhores resultados, que são apresentados nesta tese. Esta estratégia de avaliação possibilitou também normalizar as capacidades iniciais de SM de estudantes para analisar diferentes contextos de capacidades iniciais de SM dos estudantes. Explorar todos esses cenários seria levaria muito tempo

em um ambiente educacional real.

Esta Tese mostra com detalhes que o método de recomendação de DM de SM manipulado pelo ITS com o apoio de ML possui um desempenho positivo e que foi melhorando sucessivamente, o que pode ser verificado através da integração dos Módulos Tutor e do Estudante. Esse nível de detalhes, que só foi possível graças ao grande número de execuções, possibilita compreender o comportamento dos algoritmos de ML e as maneiras adequadas de utilizá-los.

O ITS para SM conseguiu se adaptar ao contexto de dados através da aprendizagem obtida pela Clusterização e RL. Dessa mesma forma, ele pode se adaptar a outros contextos de dados no mesmo domínio que representam, e.g., variações de capacidades iniciais de SM dispersas geograficamente.

O *Dataset*, os artefatos de software e o método de simulação são disponibilizados por esta Tese no sentido de contribuir com o avanço desta linha de pesquisa através da possibilidade colaboração entre diferentes pesquisadores e da continuidade em refinar sucessivamente a arquitetura do ITS para SM e seus respectivos modelos de ML.

Esses resultados oferecem subsídios para a engenharia de ITS para SM, contribuindo também para a continuidade da pesquisa, a partir da aplicação dos módulos de ITS para SM propostos em um ambiente real. Isso requer a instanciação dos artefatos apresentados e consequentemente lidar com as dificuldades pedagógicas e tecnológicas em tempo de uso.

As dificuldades nesta linha de pesquisa trazem reflexões sobre a construção desses ITS para SM quanto à necessidade de diferentes metodologias de pesquisa. Dados reais, que podem ser também obtidos de outras fontes, e.g. envolvendo o uso do ITS para SM ou a realização de testes de aptidão de SM, e a construção de diferentes cenários de simulação a partir desses dados tendem a contribuir para lidar com a complexidade de construir e refinar a precisão modelos de ML em tempo de projeto.

Capítulo 5

Conclusão

Considerando a demanda de profissionais de SM no mercado de IT, as dificuldades de realizar uma educação adequada para esta competência e a falta de tecnologias educacionais que apoiam o ensino-aprendizagem de SM em uma perspectiva ampla, esta tese aborda a problemática do ensino-aprendizagem de SM através da proposição de um ITS para SM. Os Módulos do Estudante e Tutor do ITS para SM foram construídos com ML e a integração entre eles também foi proposta e construída, o que permitiu a realização de diversos experimentos que contribuem com desafios apontados pela pesquisa neste tema.

É importante ressaltar que a aprendizagem de SM não deve ocorrer somente em ambientes acadêmicos, pois pode ocorrer em empresas ou departamentos de IT com profissionais recém-contratados. Esta pesquisa considera essa possibilidade de uso do ITS, que se relaciona com a ideia de que os ambientes organizacionais de SM possuem a necessidade de profissionais capazes de lidar com a Gestão de Configuração e Mudança e com o fato do EKM ser bastante amplo e possibilitar trabalhar com DM nesta perspectiva. Outra questão importante, que se relaciona com a aprendizagem no ambiente educacional e no ambiente organizacional, é a possibilidade de incluir no ITS os DM que representam o conhecimento no âmbito de um software específico de modo a contextualizar o ensino-aprendizagem de SM.

A literatura aponta vários desafios sobre o tema de ITS para SM. Nenhum dos trabalhos analisados até o momento em que a revisão da literatura foi feita abordou ML no Módulo do Estudante para ITS para SM, apenas dois (2) de cinco (5) abordaram ML no Módulo Tutor, nenhum trabalho abordou a integração entre esses dois módulos e somente o artigo precursor desta tese (que é parte desta pesquisa) abordou o domínio de SM de maneira ampla. Como esta tese aborda ML nesses dois módulos e a integração entre eles, ela contribui com esses desafios da literatura trazendo recursos para a continuidade da pesquisa sobre este tema e para a construção dos ITS para SM.

Esta Tese traz contribuições conceituais e experimentais sobre o assunto. Do ponto de vista conceitual, é apresentado uma proposta abstrata de ITS derivada do conteúdo de SM, da definição conceitual de ITS e de reflexões que envolveram a revisão da literatura

sobre o tema, os problemas de pesquisa relacionados e os experimentos realizados.

Do ponto de vista experimental, esta Tese contribui com: um método para a identificação de tipos de estudantes de SM baseado em clusterização e na dimensão de conhecimento de SM dos estudantes (Módulo do Estudante do ITS para SM), um método para recomendação de DM de SM baseado em RL (Módulo Tutor do ITS para SM) integrado ao resultado do Módulo do Estudante, a melhoria da eficiência da recomendação de DM de SM através da integração entre os Módulos do Estudante e Tutor, um modelo para representar a área de SM no EKM do ITS para SM e um método para avaliar a recomendação de DM baseado em dados da dimensão de conhecimento de SM e em simulação.

O Módulo do Estudante proposto trabalha com a identificação inicial dos tipos de estudantes de SM a partir de um *Dataset* de estudantes de SM e com a classificação desses tipos após esta identificação inicial ter sido executada e um modelo que representa esses tipos de estudantes de SM ter sido criado. Esta tese enfatizou a Clusterização aplicada aos dados da dimensão de conhecimento de SM de estudantes neste Módulo. Os resultados da Clusterização foram avaliados segundo condições relacionadas aos requisitos do sistema e indicaram que o algoritmo *K-Means* com quatro (4) *clusters* definidos previamente possibilitou os melhores resultados para o *Dataset* processado.

O Módulo Tutor proposto, construído com o algoritmo *Q-Learning* e uma modelagem baseada nos DM de SM, é capaz de recomendar DM de SM aprimorando sua eficiência no decorrer do tempo. Os experimentos mostraram a evolução na construção deste Módulo e a avaliação dele em diversos contextos. Inicialmente, este módulo foi avaliado com simulação envolvendo estudantes de SM gerados artificialmente sem capacidades iniciais de SM e mostraram resultados muito satisfatórios sobre a quantidade de acertos dos estudantes com relação a quantidade de recomendações realizadas pelo sistema.

Foram realizados também experimentos utilizando um *Dataset* que representa a capacidade inicial de SM (dimensão de conhecimento) de estudantes construído a partir de dados reais (INEP, 2022). Em um primeiro momento, foi possível, a partir de uma estratégia baseada em normalizações que alteram as capacidades de SM dos estudantes de forma gradativa (de 1% a 100%), verificar que os resultados seguem a sequência de desempenho de acordo com as capacidades de SM de estudantes propostas pela normalização.

A integração dos Módulos Tutor e do Estudante, que propõe construir instâncias do algoritmo *Q-Learning* específicas para cada tipo de estudante de SM gerado pela Clusterização, também foi avaliada e comparada com o experimento que representa as capacidades iniciais de SM mas não possui essa integração. Essa integração trouxe ganhos de 9% e 10% com as normalizações de 40% e 50%. Isso indica que essa integração é vantajosa para a recomendação de DM de SM e que os cenários com turmas de estudantes com capacidade que variam de baixa a média (40% e 50% na normalização) representam os melhores desempenhos de eficiência quanto ao uso do algoritmo de RL *Q-Learning* em conjunto com a Clusterização. Essas informações contribuem para o gerenciamento desse

ITS em um ambiente real a partir do monitoramento da execução dos algoritmos de ML enfatizando o desempenho do mesmo.

Considerar esse cenário de normalizações de 40% e 50%, com estudantes com capacidades iniciais de SM que variam de baixa para média, o fato das instâncias do algoritmo de RL iniciarem com um modelo sem nenhum conhecimento prévio, mas que no decorrer do tempo a partir de sua interação com o ambiente consegue refinar esse modelo e melhorar a eficiência da recomendação de DM de SM, e os resultados de EWPI para cada uma dessas duas (2) normalizações apresentados na Tabela 27, que sintetizam o resultado de toda a história de execução para cada normalização e indicam para a normalização de 40% um percentual de acertos em relação às recomendações de aproximadamente 51% e para a normalização de 50% esse percentual de acertos de aproximadamente 62%, possibilita a percepção do desempenho bastante alto com relação ao uso da integração entre os Módulos Tutor e do Estudante. Essa significativa evolução da eficiência de recomendações de DM de SM possibilitou, ao fim da execução, a obtenção de modelos de RL bastante refinados. Esses resultados validam a Hipótese Alternativa proposta, pois as técnicas de ML contribuem para construir um ITS para SM capaz de recomendar DM de SM para estudantes.

Turmas com habilidades que são no geral muito baixas ou muito altas não contribuem para a construção dos modelos de ML trabalhados nesta tese. No entanto, isso não impede que as mesmas se beneficiem das recomendações geradas pelos modelos previamente criados.

O EKM proposto contribui para avançar a pesquisa sobre ITS para SM. Ele permite que outros pesquisadores possam usá-lo e refiná-lo com o objetivo de construir um ITS capaz de auxiliar o ensino-aprendizagem de SM. Esse módulo traz para o Módulo do Estudante informações que representam a capacidade de SM em relação aos diferentes tipos e categorias de DM de SM referentes à dimensão de conhecimento de SM que podem ser abordados pelo ITS.

Como propor um ITS para SM envolve construir e avaliar modelos de ML e requer dados que representam o ambiente de ensino-aprendizagem de SM, existe a dificuldade de ter acesso a esse ambiente utilizando um ITS para SM real por questões como o alto tempo de uso do sistema e outras questões organizacionais. Por isso, a estratégia utilizada nesta tese baseou-se em simulação computacional de estudantes resolvendo DM de SM e no *Dataset* que representa a dimensão de conhecimento de SM de estudantes. Essa estratégia contribuiu para esta pesquisa que enfatizou a avaliação deste ITS para SM, a partir do uso de algoritmos de ML, em tempo de projeto. O volume de dados e processamento executados necessitou de uma grande quantidade de tempo de processamento.

O tema desta pesquisa é bastante promissor, pois além do ITS contribuir para a aprendizagem acadêmica e organizacional sobre SM e haver possibilidade real de uso, a resolução dos problemas críticos sobre a construção do ITS envolvendo o uso de técnicas

de ML traz resultados positivos para contribuir com os desafios de pesquisa identificados na literatura e consequentemente evoluir o estado da arte. A evolução desta pesquisa tende a beneficiar a ciência a partir de um aprofundamento sobre como essas técnicas de ML podem ser usadas para de fato contribuir com o ensino-aprendizagem de SM e como esses sistemas podem ser concebidos e utilizados e também a toda a comunidade acadêmica e organizacional interessada em educação em SM.

Essa pesquisa mostrou a existência da relação entre o EKM do ITS para SM e o conteúdo de SM. Essa relação também precisa refletir na UI, porém esta tese enfatizou o uso de ML no âmbito dos Módulos do Estudante e Tutor do ITS e não abordou a UI deste ITS.

5.1 Contribuições em Produções Bibliográficas

Esta seção apresenta as publicações oriundas desta Tese. Estes trabalhos são resultados dos estudos e pesquisas que contaram com experimentos utilizando algoritmos de ML para a construção de um ITS para SM e análises da literatura e documentos relacionados ao ensino de SM.

Os trabalhos publicados são:

- Francisco, R. E., & de Oliveira Silva, F. (2022). ***A Recommendation Module based on Reinforcement Learning to an Intelligent Tutoring System for Software Maintenance***. In: 14th International Conference on Computer Supported Education - CSEDU, 2022, Online. (pp. 322-329). <https://www.scitepress.org/Link.aspx?doi=10.5220/0011083900003182>

- Francisco, R. E., & de Oliveira Silva, F. (2022). ***Intelligent Tutoring System for Computer Science Education and the Use of Artificial Intelligence: A Literature Review***. In: 14th International Conference on Computer Supported Education - CSEDU, 2022, Online. (pp. 338-345). <https://www.scitepress.org/PublicationsDetail.aspx?ID=6O6FxqCVy/8=&t=1>

Há um trabalho em fase final de escrita para submissão ao periódico *Journal: Computer Applications in Engineering Education*:

- Francisco, R. E., & de Oliveira Silva, F. ***Clustering and Reinforcement Learning in an Intelligent Tutoring System for Software Maintenance***. Computer Applications in Engineering Education. Online ISSN:1099-0542. Print ISSN:1061-3773.

5.2 Trabalhos Futuros

Esta tese propõe alguns trabalhos futuros. Foi identificada a necessidade de conceber um teste de aptidão de SM para estudantes que queiram utilizar o ITS. Esse teste contribui para que o ITS tenha informações iniciais que o apoiem na identificação do tipo de estudante de SM através do Módulo do Estudante. A ideia de monitorar os algoritmos de ML executados pelo ITS para SM precisa também ser explorada por pesquisas futuras para apoiar a proposição de métodos que facilitem a gestão do ITS para SM no ambiente de ensino-aprendizagem de SM. Os algoritmos utilizados nos Módulos do ITS para SM também podem ser avaliados para outros *Datasets* que representam o contexto do ensino-aprendizagem de SM. Avaliar se os modelos psicométricos *Teoria Clássica do Teste* e *Teoria de Resposta ao Item* podem trazer novas maneiras de modelar o Simulador e, se sim, como isso impacta nos resultados dos experimentos também é outra direção possível de trabalhos futuros.

Existem possibilidades de pesquisa relacionados a propor novas funcionalidades para o ITS para SM e também uma UI. Funcionalidades como gerar *feedback* e dicas tendem a auxiliar no avanço desta linha de pesquisa. Explorar a relação entre a UI e o EKM para ITS para SM proposto também contribuirá com o avanço desta pesquisa do ponto de vista científico e técnológico.

Outras possibilidades de continuidade desta pesquisa estão relacionados a implantar o ITS para SM em ambiente real. Essa implantação possibilita a produção de diversos *Datasets* a partir de dados reais, que representam a relação entre o uso do ITS e o ambiente ao qual ele foi implantado. Feito isso, preparar o ITS com DM relacionado a artefatos de sistemas de software reais é também bastante promissor. Com isso, é possível utilizá-lo para treinamento de SM para profissionais de mercado e realizar pesquisas científicas com o ITS para SM no contexto empresarial de IT.

Referências

ABUEL-REESH, J. Y.; ABU-NASER, S. S. An Intelligent Tutoring System for Learning Classical Cryptography Algorithms (CCAITS). v. 2, n. 2, p. 11, 2018.

ACM Computing Curricula Task Force (Ed.). **Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science**. ACM, Inc, 2013. ISBN 978-1-4503-2309-3. Disponível em: <http://dl.acm.org/citation.cfm?id=2534860>.

_____. **Computing Curricula 2020: Paradigms for Global Computing Education December 2020**. ACM, Inc, 2020. Disponível em: <https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>.

AHRENS, M.; SCHNEIDER, K.; BUSCH, M. Attention in software maintenance: an eye tracking study. In: IEEE. **2019 IEEE/ACM 6th International Workshop on Eye Movements in Programming (EMIP)**. [S.l.], 2019. p. 2–9.

AL-BASTAMI, B. G.; NASER, S. A. Design and Development of an Intelligent Tutoring System for C# Language. **undefined**, 2017. Disponível em: <https://www.semanticscholar.org/paper/Design-and-Development-of-an-Intelligent-Tutoring-Al-Bastami-Naser/72ac908ac6d6c7247cb9cc695eea74d7119a3be6>.

AL-HANJORI, M. M.; SHAATH, M. Z.; ABU-NASER, S. S. Learning computer networks using intelligent tutoring system. 2017.

AL-SHAWWA, M.; ALSHAWWA, I. A.; ABU-NASER, S. S. An Intelligent Tutoring System for Learning Java. v. 3, n. 1, p. 6, 2019.

ALBATISH, I.; MOSA, M. J.; ABU-NASER, S. S. ARDUINO Tutor: An Intelligent Tutoring System for Training on ARDUINO. v. 2, n. 1, p. 10, 2018.

ALKHATLAN, A.; KALITA, J. Intelligent tutoring systems: A comprehensive historical survey with recent developments. **arXiv preprint arXiv:1812.09628**.

ALLEN, J. F. Maintaining knowledge about temporal intervals. **Communications of the ACM**, v. 26, n. 11, p. 832–843.

ALLISON, M.; JOO, S. F. An adaptive delivery strategy for teaching software testing and maintenance. In: IEEE. **2015 10th International Conference on Computer Science & Education (ICCSE)**. [S.l.], 2015. p. 237–242.

ALMASRI, A. et al. Intelligent tutoring systems survey for the period 2000-2018. IJARW, 2019.

ALSHAIKH, F.; HEWAHI, N. AI and Machine Learning Techniques in the Development of Intelligent Tutoring System: A Review. In: **2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)**. [S.l.: s.n.], 2021. p. 403–410.

ALSHAIKH, Z.; TAMANG, L. J.; RUS, V. Experiments with auto-generated socratic dialogue for source code understanding. In: **CSEDU (2)**. [S.l.: s.n.], 2021. p. 35–44.

ALSHAWWA, I. A.; AL-SHAWWA, M.; ABU-NASER, S. S. An Intelligent Tutoring System for Learning Computer Network CCNA. v. 3, n. 2, p. 9, 2019.

AMER, A. Reflections on bloom's revised taxonomy. **Electronic Journal of Research in Educational Psychology**, Universidad de Almería, v. 4, n. 1, p. 213–230, 2006.

AMNA, A. R.; POELS, G. Ambiguity in user stories: A systematic literature review. **Information and Software Technology**, Elsevier, v. 145, p. 106824, 2022.

ANQUETIL, N. et al. Software maintenance seen as a knowledge management issue. **Information and Software Technology**, Elsevier, v. 49, n. 5, p. 515–529, 2007.

BORDIN, A. S.; BENITTI, F. B. V. Software maintenance: what do we teach and what does the industry practice? In: **Proceedings of the XXXII Brazilian Symposium on Software Engineering**. [S.l.: s.n.], 2018. p. 270–279.

BOURQUE, P.; FAIRLEY, R. E. **Guide to the Software Engineering Body of Knowledge. Version 3.0**. [S.l.]: IEEE, 2014.

CAO, C. Leveraging large language model and story-based gamification in intelligent tutoring system to scaffold introductory programming courses: A design-based research study. **arXiv preprint arXiv:2302.12834**, 2023.

CARTER, E.; BLANK, G. D. An intelligent tutoring system to teach debugging. In: SPRINGER. **International Conference on Artificial Intelligence in Education**. [S.l.], 2013. p. 872–875.

CHRYSAFIADI, K. et al. Evaluating the user's experience, adaptivity and learning outcomes of a fuzzy-based intelligent tutoring system for computer programming for academic students in greece. **Education and Information Technologies**, Springer, v. 28, n. 6, p. 6453–6483, 2023.

DURAN, R. et al. Cognitive complexity of comprehending computer programs. Aalto University, 2020.

ELREESH, J. Y. A.; ABU-NASER, S. S. Cloud Network Security Based on Biometrics Cryptography Intelligent Tutoring System. v. 3, n. 3, p. 34, 2019.

ERICSON, B. J. et al. Parsons problems and beyond: Systematic literature review and empirical study designs. **Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education**, p. 191–234, 2022.

ERüMIT, A. K.; ÇETIN, Design framework of adaptive intelligent tutoring systems. **Education and Information Technologies**, v. 25, n. 5, p. 4477–4500, set. 2020. ISSN 1573-7608. Disponível em: <https://doi.org/10.1007/s10639-020-10182-8>.

EZUGWU, A. E. et al. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. **Engineering Applications of Artificial Intelligence**, v. 110, p. 104743, abr. 2022. ISSN 0952-1976. Publisher: Pergamon. Disponível em: <https://www-sciencedirect.ez34.periodicos.capes.gov.br/science/article/pii/S095219762200046X>.

FERNÁNDEZ-SÁEZ, A. M.; CHAUDRON, M. R.; GENERO, M. An industrial case study on the use of uml in software maintenance and its perceived benefits and hurdles. **Empirical Software Engineering**, Springer, v. 23, n. 6, p. 3281–3345, 2018.

FIGUEIREDO, J.; GARCÍA-PEÑALVO, F. J. Intelligent tutoring systems approach to introductory programming courses. In: **Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality**. [S.l.: s.n.], 2020. p. 34–39.

FONTANINI, A. D.; ABREU, J. A data-driven birch clustering method for extracting typical load profiles for big data. In: IEEE. **2018 IEEE Power & energy society general meeting (PESGM)**. [S.l.], 2018. p. 1–5.

FRANCISCO, R. **Details of Clustering runs for SM students, 2023. Available in <https://github.com/rodrigoefr/researchdr/blob/main/Compare_Cluster _GIT.xlsx>. Accessed in September 14, 2023.** [S.l.], 2023.

_____. **Results of Clustering runs for SM students, 2023. Available in <https://github.com/rodrigoefr/researchdr/blob/main/dataset_m _student___input _output.rar>. Accessed in September 14, 2023.** [S.l.], 2023.

_____. **SM DM's Recommender Version 2. 2023. Available in <https://github.com/rodrigoefr/pesquisadr/tree/main/sewlearn_v2>. Accessed in September 15, 2023.** [S.l.], 2023.

_____. **SM DM's Recommender Version 3. 2023. Available in <https://github.com/rodrigoefr/pesquisadr/tree/main/sewlearn_v3>. Accessed in September 15, 2023.** [S.l.], 2023.

FRANCISCO, R.; SILVA, F. A Recommendation Module based on Reinforcement Learning to an Intelligent Tutoring System for Software Maintenance. In: . [s.n.], 2022. p. 322–329. ISBN 978-989-758-562-3. Disponível em: <https://www.scitepress.org/Link.aspx?doi=10.5220/0011083900003182>.

FRANCISCO, R. E.; SILVA, F. de O. Intelligent Tutoring System for Computer Science Education and the Use of Artificial Intelligence: A Literature Review. abr. 2022. Accepted: 2022-04-06T16:10:54Z. Disponível em: <https://www.scitepress.org/PublicationsDetail.aspx?ID=VGBKvt7K1xY=&t=1>.

GALAFASSI, C. et al. Evologic: Sistema tutor inteligente para ensino de lógica. In: SBC. **Anais do XLVII Seminário Integrado de Software e Hardware**. [S.l.], 2020. p. 222–233.

GALAFASSI, F. F. P. et al. Heráclito: Intelligent tutoring system for logic. In: SPRINGER. **International Conference on Practical Applications of Agents and Multi-Agent Systems**. [S.l.], 2019. p. 251–254.

GALLAGHER, K.; FIORAVANTI, M.; KOZAITIS, S. Teaching software maintenance. In: IEEE. **2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)**. [S.l.], 2019. p. 353–362.

GEORGILA, K. et al. Using reinforcement learning to optimize the policies of an intelligent tutoring system for interpersonal skills training. In: **Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems**. [S.l.: s.n.], 2019. p. 737–745.

GOOGLE. **Google Colab - Colaboratory, 2023. Available in <https://colab.research.google.com/>. Accessed in November 1, 2023.** [S.l.], 2023.

GUPTA, A.; ROY, P. P.; DUTT, V. Evaluation of Instance-Based Learning and Q-Learning Algorithms in Dynamic Environments. **IEEE Access**, v. 9, p. 138775–138790, 2021. ISSN 2169-3536. Conference Name: IEEE Access.

HADDAD, I. A. E.; NASER, S. A. ADO-Tutor: Intelligent Tutoring System for leaning ADO.NET. **undefined**, 2017. Disponível em: <https://www.semanticscholar.org/paper/ADO-Tutor%3A-Intelligent-Tutoring-System-for-leaning-Haddad-Naser/e0f76d52476a5838e8a841be260247ad872d1570>.

HARSLEY, R. et al. Integrating support for collaboration in a computer science intelligent tutoring system. In: SPRINGER. **International conference on intelligent tutoring systems**. [S.l.], 2016. p. 227–233.

HASANEIN, H. A. A.; NASER, S. S. A. An intelligent tutoring system for cloud computing. p. 5, 2017.

HECKMAN, S.; STOLEE, K.; PARNIN, C. 10+ years of teaching software engineering with itrust: the good, the bad, and the ugly. In: IEEE. **2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)**. [S.l.], 2018. p. 1–4.

HOOSHYAR, D. et al. Sits: A solution-based intelligent tutoring system for students' acquisition of problem-solving skills in computer programming. **Innovations in Education and Teaching International**, Taylor & Francis, v. 55, n. 3, p. 325–335, 2018.

HU, Y. **Teaching Effectiveness of Intelligent Tutoring Systems**. [S.l.]: Washington State University, 2019.

HUANG, Y. et al. Supporting skill integration in an intelligent tutoring system for code tracing. **Journal of Computer Assisted Learning**, Wiley Online Library, v. 39, n. 2, p. 477–500, 2023.

INEP. **Microdados Enade 2017. Brasília: Inep, 2022. Available in <https://www.gov.br/inep/pt-br/acesso-a-informacao/dados-abertos/microdados/enade>. Accessed in June 30, 2022.** [S.l.], 2022.

IZU, C. et al. Fostering program comprehension in novice programmers-learning activities and learning trajectories. In: **Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education**. [S.l.: s.n.], 2019. p. 27–52.

JAHWAR, A. F.; ABDULAZEEZ, A. M. Meta-heuristic algorithms for k-means clustering: A review. **PalArch's Journal of Archaeology of Egypt/Egyptology**, v. 17, n. 7, p. 12002–12020, 2020.

JAISWAL, M. Software architecture and software design. **International Research Journal of Engineering and Technology (IRJET) e-ISSN**, p. 2395–0056, 2019.

JANANI, R.; VIJAYARANI, S. Text document clustering using spectral clustering algorithm with particle swarm optimization. **Expert Systems with Applications**, Elsevier, v. 134, p. 192–200, 2019.

JEREMIC, Z.; JOVANOVIC, J.; GASEVIC, D. Evaluating an intelligent tutoring system for design patterns: The depths experience. **Journal of Educational Technology & Society**, JSTOR, v. 12, n. 2, p. 111, 2009.

JEREMIC, Z.; JOVANOVIĆ, J.; GAŠEVIĆ, D. Student modeling and assessment in intelligent tutoring of software patterns. **Expert Systems with Applications**, v. 39, n. 1, p. 210–222, 2012.

JR, E. L. B. et al. Tecnologias na educação em computação: Primeiros referenciais. **Revista Brasileira de Informática na Educação**, v. 28, p. 509, 2020.

KRANCHER, O. J.; DIBBERN, J. Learning software-maintenance tasks in offshoring projects: A cognitive-load perspective. 2012.

KRUCHTEN, P. B. The 4+ 1 view model of architecture. **IEEE software**, IEEE, v. 12, n. 6, p. 42–50, 1995.

LÓPEZ, C. et al. Design of e-activities for the learning of code refactoring tasks. In: IEEE. **2014 International Symposium on Computers in Education (SIIE)**. [S.l.], 2014. p. 35–40.

LUBURIć, N. et al. SSRN Scholarly Paper, **An Intelligent Tutoring System to Support Code Maintainability Skill Development**. Rochester, NY: [s.n.], 2022. Disponível em: <https://papers.ssrn.com/abstract=4168647>.

_____. Clean Code Tutoring: Makings of a Foundation:. In: **Proceedings of the 14th International Conference on Computer Supported Education**. Online Streaming, — Select a Country —: SCITEPRESS - Science and Technology Publications, 2022. p. 137–148. ISBN 978-989-758-562-3. Disponível em: <https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0010800900003182>.

L'ERARIO, A.; THOMAZINHO, H. C. S.; FABRI, J. A. An approach to software maintenance: A case study in small and medium-sized businesses it organizations. **International Journal of Software Engineering and Knowledge Engineering**, World Scientific, v. 30, n. 05, p. 603–630, 2020.

MAHDI, A. O.; ALHABBASH, M. I.; NASER, S. S. A. An intelligent tutoring system for teaching advanced topics in information security. p. 9, 2016.

MAHMUD, M. et al. Comparative analysis of k-means and bisecting k-means algorithms for brain tumor detection. In: IEEE. **2018 International conference on computer, communication, chemical, material and electronic engineering (IC4ME2)**. [S.l.], 2018. p. 1–4.

MAROUF, A. M.; ABU-NASER, S. S. Intelligent tutoring system for teaching computer science i in al-azhar university, gaza. **International Journal of Academic and Applied Research (IJAAR)**, v. 3, n. 3, p. 31–53, 2019.

MARSLAND, S. **Machine learning: an algorithmic perspective**. [S.l.]: CRC press, 2015.

MASAPANTA-CARRIÓN, S.; VELÁZQUEZ-ITURBIDE, J. Á. Evaluating instructors' classification of programming exercises using the revised bloom's taxonomy. In: **Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education**. [S.l.: s.n.], 2019. p. 541–547.

MISILMANI, H. M. E.; NAOUS, T. Machine learning in antenna design: An overview on machine learning concept and algorithms. In: IEEE. **2019 International Conference on High Performance Computing & Simulation (HPCS)**. [S.l.], 2019. p. 600–607.

MOSA, M. J.; ALBATISH, I.; ABU-NASER, S. S. ASP.NET-Tutor: Intelligent Tutoring System for leaning ASP.NET. v. 2, n. 2, p. 8, 2018.

MOUSAVINASAB, E. et al. Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods. **Interactive Learning Environments**, v. 29, n. 1, p. 142–163, jan. 2021. ISSN 1049-4820, 1744-5191. Disponível em: <https://www.tandfonline.com/doi/full/10.1080/10494820.2018.1558257>.

NAKHAL, M.; BASHHAR, B. Adaptive ITS for Learning Computer Theory. **European Academic Research**, v. 4, n. 10, p. 8770–8782, 2017. Disponível em: <https://philarchive.org/rec/NAKAIF-2>.

NKAMBOU, R. et al. Learning logical reasoning using an intelligent tutoring system: a hybrid approach to student modeling. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.]. v. 37, n. 13, p. 15930–15937.

OBERHAUSER, R. Rescu: A trail recommender approach to support program code understanding. In: **Proc. 8th Int. Conf. on Information, Process, and Knowledge Manage**. [S.l.: s.n.], 2016. p. 112–118.

_____. Visitr: 3d visualization for code visitation trail recommendations. **International Journal on Advances in Software Volume 10, Number 1 & 2, 2017**, 2017.

PAASSEN, B.; JENSEN, J.; HAMMER, B. Execution traces as a powerful data representation for intelligent tutoring systems for programming. **International Educational Data Mining Society**, ERIC, 2016.

PATEL, A.; JAIN, S. Present and future of semantic web technologies: a research statement. **International Journal of Computers and Applications**, Taylor & Francis, v. 43, n. 5, p. 413–422, 2021.

PRICE, T. W.; DONG, Y.; LIPOVAC, D. isnap: towards intelligent tutoring in novice programming environments. In: **Proceedings of the 2017 ACM SIGCSE Technical Symposium on computer science education**. [S.l.: s.n.], 2017. p. 483–488.

PYLE, D.; JOSÉ, C. S. An executive's guide to machine learning. **McKinsey Quarterly**, v. 3, p. 44–53, 2015.

PYTHON. **Random — Gera números pseudoaleatórios. 2023. Available in <https://docs.python.org/pt-br/3.7/library/random.html>. Accessed in September 28, 2023.** [S.l.], 2023.

RAABE, A.; GOMES, E. B. Maker: uma nova abordagem para tecnologia na educação. **Revista Tecnologias na Educação**, v. 26, n. 26, p. 6–20, 2018.

RAHMAN, A. A.; ABDULLAH, M.; ALIAS, S. H. The architecture of agent-based intelligent tutoring system for the learning of software engineering function point metrics. In: IEEE. **2016 2nd International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR)**. [S.l.], 2016. p. 139–144.

RAZMJOO, S. A.; KAZEMPOURFARD, E. On the representation of bloom's revised taxonomy in interchange course books. JOURNAL OF TEACHING LANGUAGE SKILLS (JTLS)(JOURNAL OF SOCIAL SCIENCES AND ..., 2012.

ŘEZANKOVÁ, H. Different approaches to the silhouette coefficient calculation in cluster evaluation. In: **21st International Scientific Conference AMSE Applications of Mathematics and Statistics in Economics**. [S.l.: s.n.], 2018. p. 1–10.

ROLL, I.; WYLIE, R. Evolution and revolution in artificial intelligence in education. **International Journal of Artificial Intelligence in Education**, Springer, v. 26, p. 582–599, 2016.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial - Tradução da Terceira Edição**. [S.l.]: Elsevier, 2013.

RUSSELL, A.; VINSEL, L. **Andrew Russell and Lee Vinsel. Let's Get Excited About Maintenance! Disponível em https://www.nytimes.com/2017/07/22/opinion/sunday/lets-get-excited-about-maintenance.html. Acessado em 13 de novembro de 2020.** [S.l.], 2017.

SCIKIT-LEARN. **Scikit-Learn - Machine Learning in Python, 2023. Available in <https://scikit-learn.org/stable/>. Accessed in September 14, 2023.** [S.l.], 2023.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: From theory to algorithms**. [S.l.]: Cambridge university press, 2014.

SHARMA, P.; HARKISHAN, M. Designing an intelligent tutoring system for computer programing in the pacific. **Education and Information Technologies**, Springer, v. 27, n. 5, p. 6197–6209, 2022.

SHARMA, S.; BATRA, N. et al. Comparative study of single linkage, complete linkage, and ward method of agglomerative clustering. In: IEEE. **2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)**. [S.l.], 2019. p. 568–573.

SHAWKY, D.; BADAWI, A. A reinforcement learning-based adaptive learning system. In: SPRINGER. **International Conference on Advanced Machine Learning Technologies and Applications**. [S.l.], 2018. p. 221–231.

SHUTAYWI, M.; KACHOUIE, N. N. Silhouette Analysis for Performance Evaluation in Machine Learning with Applications to Clustering. **Entropy**, v. 23, n. 6, p. 759, jun. 2021. ISSN 1099-4300. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute. Disponível em: <https://www.mdpi.com/1099-4300/23/6/759>.

SINAGA, K. P.; YANG, M.-S. Unsupervised k-means clustering algorithm. **IEEE access**, IEEE, v. 8, p. 80716–80727, 2020.

SMITH, T. M. et al. Selecting open source software projects to teach software engineering. In: **Proceedings of the 45th ACM technical symposium on Computer science education**. [S.l.: s.n.], 2014. p. 397–402.

SZABO, C. Novice code understanding strategies during a software maintenance assignment. In: IEEE. **2015 IEEE/ACM 37th IEEE International Conference on Software Engineering**. [S.l.], 2015. v. 2, p. 276–284.

TOMKIN, J. H.; WEST, M.; HERMAN, G. L. An Improved Grade Point Average, With Applications to CS Undergraduate Education Analytics. **ACM Transactions on Computing Education**, v. 18, n. 4, p. 17:1–17:16, set. 2018. Disponível em: <https://doi.org/10.1145/3157086>.

TONHÃO, S. de F.; SOUZA, S. M. Andressa de; PRATES, J. M. Uma abordagem prática apoiada pela aprendizagem baseada em projetos e gamificação para o ensino de engenharia de software. In: SBC. **Anais do Simpósio Brasileiro de Educação em Computação**. [S.l.], 2021. p. 143–151.

TROUSSAS, C. et al. Enhancing personalized educational content recommendation through cosine similarity-based knowledge graphs and contextual signals. **Information**, MDPI, v. 14, n. 9, p. 505, 2023.

VERDÚ, E. et al. Integration of an intelligent tutoring system in a course of computer network design. **Educational Technology Research and Development**, v. 65, n. 3, p. 653–677.

WANG, F. Reinforcement learning in a pomdp based intelligent tutoring system for optimizing teaching strategies. **International Journal of Information and Education Technology**, v. 8, n. 8, p. 553–558, 2018.

WANG, P. On defining artificial intelligence. **Journal of Artificial General Intelligence**, De Gruyter Poland, v. 10, n. 2, p. 1–37, 2019.

WANG, Y.; KING, G.; WICKBURG, H. A method for built-in tests in component-based software maintenance. In: IEEE. **Proceedings of the Third European Conference on Software Maintenance and Reengineering (Cat. No. PR00090)**. [S.l.], 1999. p. 186–189.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. [S.l.]: Elsevier, 2009. v. 2.

WOOD, A. et al. Detecting speech act types in developer question/answer conversations during bug repair. In: **Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering**. [S.l.: s.n.], 2018. p. 491–502.

WU, Y.-c.; FENG, J.-w. Development and application of artificial neural network. **Wireless Personal Communications**, Springer, v. 102, p. 1645–1656, 2018.

ZHI, R. et al. Exploring the impact of worked examples in a novice programming environment. In: **Proceedings of the 50th ACM Technical Symposium on Computer Science Education**. [S.l.: s.n.]. p. 98–104.

ZORZO, A. F. et al. Referenciais de Formação para os Cursos de Graduação em Computação 2017. 2017.