

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Vandré Leal Cândido

**Revisão Sistemática da Literatura como
ferramenta de apoio na definição de
arquiteturas de software para
Sistemas Tutores Inteligentes**

Uberlândia, Brasil

2016

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Vandré Leal Cândido

**Revisão Sistemática da Literatura como ferramenta de
apoio na definição de arquiteturas de software para
Sistemas Tutores Inteligentes**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito parcial exigido à obtenção do grau
de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Aleksandro Santos Soares

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2016

Vandré Leal Cândido

Revisão Sistemática da Literatura como ferramenta de apoio na definição de arquiteturas de software para Sistemas Tutores Inteligentes

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito parcial exigido à obtenção do grau de Bacharel em Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 28 de Novembro de 2016:

Prof. Dr. Alexsandro Santos Soares
Orientador

Prof. Dr. Fabiano Azevedo Dorça

Prof. Dr. Carlos Roberto Lopes

Uberlândia, Brasil
2016

Dedico este trabalho aos meus pais, que com muito incentivo e apoio, não mediram esforços para que eu alcançasse este momento de minha vida. Obrigado pelo carinho e dedicação constantes durante esta jornada.

Agradecimentos

Agradeço a todos os professores que me acompanharam durante a graduação pela atenção e sabedoria compartilhadas, em especial o professor Dr. Alexsandro Santos Soares, pelas contribuições dadas à este trabalho. Agradeço também a todos os amigos presentes no decorrer de cada semestre.

Resumo

Este trabalho verifica a possibilidade de emprego da metodologia de revisão sistemática da literatura como instrumento de identificação de uma arquitetura de *software* apropriada à um sistema tutor educacional. Os sistemas tutores inteligentes direcionados ao ensino de programação são utilizados como um estudo de caso a fim de reproduzir a aplicabilidade da técnica. Através do processo de revisão conduzido foi possível sugerir uma arquitetura composta por seis módulos: módulo do domínio, módulo do estudante, módulo pedagógico, módulo especialista, módulo de geração de problemas e módulo da comunicação.

Palavras-chave: revisão sistemática da literatura, arquitetura de software, sistemas tutores inteligentes, tutoria em programação.

Lista de ilustrações

Figura 1 – Visão geral do modelo 4+1 descrito por (KRUCHTEN, 1995)	16
Figura 2 – Arquitetura clássica de um Sistema Tutor Inteligente	24
Figura 3 – Ciclo do Raciocínio Baseado em Casos	28
Figura 4 – Processo de recompensa do modelo de decisão sequencial	29
Figura 5 – Exemplo de um processo de decisão adaptado de (SAWICKI et al., 2015) . .	30
Figura 6 – Exemplo de rede Neuro-Fuzzy	31
Figura 7 – Exemplo de raciocínio de uma rede Bayesiana	33
Figura 8 – Representação das publicações categorizadas	40
Figura 9 – Quantidade de publicações por categoria	41
Figura 10 – Análise quantitativa da questão I do formulário de avaliação	43
Figura 11 – Análise quantitativa da questão II do formulário de avaliação	44
Figura 12 – Análise quantitativa da questão III do formulário de avaliação	45
Figura 13 – Análise quantitativa da questão IV do formulário de avaliação	45
Figura 14 – Análise quantitativa da questão V do formulário de avaliação	46
Figura 15 – Análise quantitativa da questão VI do formulário de avaliação	47
Figura 16 – Análise quantitativa da questão VII do formulário de avaliação	48
Figura 17 – Análise quantitativa da questão VIII do formulário de avaliação	48
Figura 18 – Análise quantitativa da questão IX do formulário de avaliação	49
Figura 19 – Análise quantitativa da questão X do formulário de avaliação	50
Figura 20 – Arquitetura do C-Tutor apresentada em (SONG et al., 1997)	51
Figura 21 – Arquitetura do E-TCL apresentada em (EL-KHOULY; FAR; KOONO, 2000) .	53
Figura 22 – Arquitetura de geração de problemas apresentada em (KUMAR, 2002) . . .	53
Figura 23 – Arquitetura do Haskell-Tutor apresentada em (XU; SARRAFZADEH, 2004) .	55
Figura 24 – Arquitetura do C++ STL ITS apresentada em (LEE; BABA, 2005)	56
Figura 25 – Arquitetura de três camadas do C++ STL ITS apresentada em (LEE; BABA, 2005)	56
Figura 26 – Arquitetura do BITS apresentada em (BUTZ; HUA; MAGUIRE, 2006)	57
Figura 27 – Fluxograma do processo de tutoria apresentado em (NASER, 2008)	59
Figura 28 – Arquitetura do J-LATTE apresentada em (HOLLAND, 2009)	60
Figura 29 – Arquitetura do JO-Tutor apresentada em (ABU-NASER et al., 2011)	61
Figura 30 – Arquitetura do ASK-ELLE apresentada em (GERDES, 2012)	62
Figura 31 – Arquitetura do Protus 2.0 apresentada em (VESIN et al., 2013)	63
Figura 32 – Arquitetura do PHP ITS apresentada em (WERAGAMA, 2013)	64
Figura 33 – Arquitetura web do PHP ITS apresentada em (WERAGAMA, 2013)	65
Figura 34 – Arquitetura do Java Sensei apresentada em (ZATARAIN-CABADA et al., 2015)	66
Figura 35 – Arquitetura do SITS apresentada em (HOOSHYAR et al., 2016)	66

Figura 36 – Interface com o curso de ações do tutor extraída de (SHAH; KUMAR, 2002)	67
Figura 37 – Interface principal do ProPl extraída de (LANE; VANLEHN, 2004)	68
Figura 38 – Interface principal do Haskell-Tutor extraída de (XU; SARRAFZADEH, 2004)	69
Figura 39 – Interface de ensino do C++ STL ITS extraída de (LEE; BABA, 2005)	70
Figura 40 – Interface de ensino do BITS extraída de (BUTZ; HUA; MAGUIRE, 2006)	71
Figura 41 – Interface principal do tutor extraída de (EMURIAN, 2006)	72
Figura 42 – Interface principal do CPP-Tutor extraída de (NASER, 2009)	73
Figura 43 – Interface principal do J-LATTE extraída de (HOLLAND, 2009)	74
Figura 44 – Interface principal do JITS extraída de (SYKES, 2010)	75
Figura 45 – Interface principal do JO-Tutor extraída de (ABU-NASER et al., 2011)	76
Figura 46 – Interface principal do ASK-ELLE extraída de (GERDES, 2012)	76
Figura 47 – Interface de solução de problemas do PHP ITS extraída de (WERAGAMA, 2013)	77
Figura 48 – Protótipo do Java Sensei apresentado em (ZATARAIN-CABADA et al., 2015)	77
Figura 49 – Interface de ensino de conceitos do SITS extraída de (HOOSHYAR et al., 2016)	78
Figura 50 – Arquitetura sugerida para um sistema tutor educacional	81

Lista de abreviaturas e siglas

AC	Arquitetura Clássica
AS	Arquitetura Singular
IA	Inteligência Artificial
NF	Neuro-Fuzzy
PDM	Processo de Decisão de Markov
RB	Redes Bayesianas
RBC	Raciocínio Baseado em Casos
RSL	Revisão Sistemática da Literatura
STI	Sistema Tutor Inteligente

Sumário

1	INTRODUÇÃO	12
1.1	Problema	12
1.2	Hipótese	12
1.3	Objetivos	13
1.4	Resultados Esperados	13
1.5	Organização do Trabalho	13
2	ARQUITETURA DE SOFTWARE	14
2.1	Importância	15
2.2	Modelagem Arquitetural	15
2.3	Desafios	16
2.4	Indicadores de Qualidade	17
3	REVISÃO SISTEMÁTICA DA LITERATURA	18
3.1	Processo de Revisão	19
3.1.1	Planejamento	20
3.1.2	Execução	21
3.1.3	Conclusão	22
3.2	Estudo de Caso	22
4	SISTEMAS TUTORES INTELIGENTES	23
4.1	Módulo Pedagógico	25
4.1.1	Ressalvas	26
4.2	Módulo Especialista	26
4.2.1	Técnicas de Representação do Conhecimento do Especialista	27
4.2.1.1	Raciocínio Baseado em Casos	27
4.2.1.2	Processo de Decisão de Markov	29
4.2.1.3	Neuro-Fuzzy	31
4.2.1.4	Redes Bayesianas	32
5	ESTUDO DE CASO	34
5.1	Planejamento	34
5.1.1	Justificativa da Revisão	34
5.1.2	Protocolo de Revisão	34
5.1.3	Coleta de Material	34
5.1.4	Estratégias de Busca	35

5.1.5	Resultados e Documentação	36
5.1.6	Validação da Estratégia de Pesquisa	36
5.1.7	CrITÉrios de Seleção	37
5.1.7.1	Inclusão de Material	37
5.1.7.2	Exclusão de Material	37
5.1.7.3	Processo de Seleção	37
5.1.7.4	Avaliação da Relevância da Publicação	38
5.1.8	Extração de Dados	39
5.1.8.1	Processo de Coleta	39
5.1.8.2	SÍntese dos Dados	40
5.1.9	Limitações do Estudo	40
5.2	Execução	40
5.2.1	Coleta das Publicações	40
5.2.2	Avaliação da Relevância dos Estudos	41
5.2.2.1	Quadro de Pontuação Geral	42
5.2.2.2	Arquitetura	43
5.2.2.3	Inferência dos dados	43
5.2.2.4	Retorno de Informações	44
5.2.2.5	Protótipo	45
5.2.2.6	Experimento	47
5.2.2.7	Resultados	49
5.2.3	Análise dos dados	50
5.2.3.1	Arquitetura	50
5.2.3.1.1	C-Tutor	51
5.2.3.1.2	E-TCL	52
5.2.3.1.3	Sistema Tutor em C++	52
5.2.3.1.4	ProPI	53
5.2.3.1.5	Haskell-Tutor	54
5.2.3.1.6	C++ STL ITS	55
5.2.3.1.7	BITS	56
5.2.3.1.8	Sistema Tutor em applets Java	57
5.2.3.1.9	CPP-Tutor	58
5.2.3.1.10	J-LATTE	58
5.2.3.1.11	JITS	60
5.2.3.1.12	JO-Tutor	61
5.2.3.1.13	ASK-ELLE	62
5.2.3.1.14	Protus 2.0	62
5.2.3.1.15	PHP ITS	64
5.2.3.1.16	Java Sensei	65

5.2.3.1.17	SITS	66
5.2.3.2	Protótipo	67
5.2.4	Conclusões do processo de revisão	79
6	CONCLUSÃO	82
6.1	Trabalhos Futuros	83
	REFERÊNCIAS	84
	APÊNDICES	90
	APÊNDICE A – FORMULÁRIO DE AVALIAÇÃO DA RELEVÂNCIA DA PUBLICAÇÃO	91

1 Introdução

A Revisão Sistemática da Literatura é uma metodologia de estudo que permite a investigação de material relevante à uma determinada questão ou tópico de interesse de modo confiável e minucioso (MAJOR; KYRIACOU; BRERETON, 2012). Por meio deste método é possível resumir o estado da arte de uma área de conhecimento, responder questões práticas de pesquisa e até mesmo identificar eventuais lacunas passíveis de investigação por pesquisadores de determinada área.

Este trabalho propõe a utilização da metodologia de Revisão Sistemática da Literatura como instrumento de apoio no processo de definição da arquitetura de sistemas educacionais. Para isso, um estudo de caso na área de sistemas tutores inteligentes orientados ao ensino de programação é selecionado como modo de validação desta hipótese. Por meio do estudo de caso selecionado, é possível observar a condução de uma revisão sistemática em sua integridade, incluindo as etapas de planejamento, execução e conclusão.

A arquitetura inicial para um sistema tutor inteligente orientado ao ensino de programação é alcançada através do processo de revisão, e uma investigação das dissimilaridades entre a arquitetura clássica de um sistema tutor e as arquiteturas estudadas é conduzida após a execução do processo de revisão.

O protocolo de revisão formulado na fase de planejamento desta pesquisa é enxuto e considera as características de um sistema computacional, o que permite sua aplicação em outros estudos da área com uma quantidade baixa de adequações. Ou seja, outros pesquisadores poderão se beneficiar do protocolo desenvolvido neste estudo através de sua utilização como modelo inicial em suas revisões.

1.1 Problema

“É possível utilizar a metodologia de revisão sistemática da literatura como ferramenta de apoio na definição de uma arquitetura para um sistema tutor inteligente?”

1.2 Hipótese

A definição de uma arquitetura de software adequada a um sistema pode ser alcançada por meio da condução de uma revisão sistemática da literatura correlata à área de conhecimento em estudo.

1.3 Objetivos

O objetivo geral deste trabalho consiste em mostrar que a execução de uma revisão sistemática da literatura pode ser um instrumento de identificação de uma arquitetura de *software* apropriada a um determinado sistema de informação. Os sistemas tutores inteligentes direcionados ao ensino de programação são utilizados como um estudo de caso a fim de reproduzir a aplicabilidade do protocolo de revisão. Os objetivos específicos são enumerados a seguir:

- Identificar as fases e atividades essenciais para a condução adequada do processo de revisão sistemática da literatura.
- Formular um protocolo apropriado para a execução de uma revisão sistemática no domínio de arquitetura de software por meio do estudo de caso selecionado.

1.4 Resultados Esperados

O trabalho tem como propósito guiar indivíduos tanto da área acadêmica quanto da indústria na execução de uma revisão sistemática como instrumento de apoio na definição de uma arquitetura de software para um sistema tutor, uma vez que a definição de uma arquitetura para um sistema pode se tornar uma tarefa árdua conforme o cenário e número de variáveis envolvidas. A condução de uma revisão sistemática permite, então, poupar tempo e recursos preciosos durante o processo de definição de uma arquitetura, especialmente em cenários complexos.

1.5 Organização do Trabalho

- O capítulo 2 introduz conceitos de Arquitetura de Software, com foco na importância, modelagem arquitetural, desafios e qualidade.
- O capítulo 3 contém os fundamentos básicos de uma metodologia de revisão sistemática e descreve as fases de planejamento, execução e conclusão de um processo de revisão bem estruturado.
- O capítulo 4 introduz os conceitos primordiais de Sistemas Tutores Inteligentes, com foco nos aspectos de arquitetura e inferência de dados.
- O capítulo 5 descreve o estudo de caso selecionado para o estudo, assim como o processo de revisão sistemática executado.
- O capítulo 6 contém as conclusões obtidas por meio da revisão executada e discute trabalhos futuros a serem realizados.

2 Arquitetura de Software

Uma arquitetura de software descreve a estrutura em que um sistema é organizado. Esta estrutura permite a atribuição de tarefas e responsabilidades distintas à cada componentes de um sistema (GORTON, 2006) durante a sua subdivisão. Dessa forma, a arquitetura auxilia na tomada de decisões de projeto de alto nível incluindo caminhos de interação de um sistema e propriedades do sistema como um todo, atuando como uma ponte entre os requisitos e a implementação (GARLAN, 2014). Sua utilização pode atuar em seis aspectos distintos do ciclo de desenvolvimento de software:

1. *Entendimento*: uma arquitetura fornece um novo nível de abstração à um sistema, o que simplifica a sua compreensão geral e a tomada de novas decisões arquiteturais.
2. *Reuso*: uma arquitetura suporta reuso de componentes e quadros de trabalhos nos quais os componentes podem ser integrados, reduzindo a complexidade e tempo de desenvolvimento de um sistema.
3. *Construção*: uma arquitetura fornece um escopo básico de desenvolvimento, uma vez que indica os principais componentes e dependências de um sistema. Além disso, fornece uma visão clara da interface interna de um software por meio da identificação das atribuições de cada parte integrante do sistema.
4. *Evolução*: uma arquitetura expõe as dimensões e limitações nas quais um sistema pode evoluir. Através da arquitetura, os custos de modificação e/ou extensão de um sistema são apurados de forma mais precisa.
5. *Análise*: a descrição de uma arquitetura permite uma análise mais rigorosa da consistência de um sistema e a verificação da conformidade de um software com os requisitos estipulados.
6. *Gerenciamento*: a avaliação crítica de um software oferece uma visão clara dos requisitos e estratégias de implementação, como também dos potenciais riscos associados ao desenvolvimento de certo software. Além disso, reduz consideravelmente a quantidade de retrabalho exigida durante o ciclo de vida de um sistema.

Dessa forma, a arquitetura de software é uma ótima ferramenta para administrar a complexidade de um projeto e pode atuar como um plano de negociação de requisitos de sistema, como também um método de estruturar discussões com clientes, desenvolvedores e gerentes, uma vez que abstrai os detalhes de implementação de baixo nível e proporciona uma visão de alto nível de fácil discussão (SOMMERVILLE, 2010).

A arquitetura também lida com requisitos não-funcionais de um projeto. Os requisitos não-funcionais são os requisitos relacionados ao método no qual uma aplicação oferece determinada funcionalidade. Há três tipos de requisitos não-funcionais que uma arquitetura deve cuidar ([GORTON, 2006](#)):

- *Restrições técnicas*: são as restrições responsáveis pela especificação das tecnologias que devem ser utilizadas para o desenvolvimento de uma aplicação.
- *Restrições de negócios*: são as restrições de projeto ligadas aos custos de desenvolvimento, como exemplo a utilização de certa plataforma ou ferramenta devido aos menores custos de obtenção e licenciamento.
- *Atributos de qualidade*: definem os requisitos que uma aplicação deve atender como nível de escalabilidade, disponibilidade, manutenibilidade, portabilidade, usabilidade, desempenho, entre outros. Estes atributos se relacionam com as preocupações que um usuário da aplicação normalmente manifesta.

2.1 Importância

A definição de uma arquitetura de software é importante porque é geralmente o primeiro artefato produzido durante o projeto de um sistema e pode ser utilizado como forma de alcançar qualidades específicas desejadas em um software. Por meio de seu uso é possível ainda analisar os cenários de implementação e utilização de um projeto e tomar decisões críticas ainda no início do ciclo de desenvolvimento, mitigando custos adicionais com modificações ao longo do ciclo do projeto ([NORTHROP, 2003](#)).

Dessa forma, é possível melhorar os índices de qualidade dos atributos de qualidade não-funcionais do software como desempenho, confiança, segurança, manutenibilidade e reusabilidade. Além disso, é um importante veículo de comunicação entre as partes interessadas de um projeto, pois permite que profissionais de vários perfis distintos sejam capazes de compreender a consistência de um software.

2.2 Modelagem Arquitetural

A dificuldade em se descrever a arquitetura de um software e lidar com todas as suas particularidades pode ser difícil e exigir bastante tempo e atenção. Modelos de divisão arquitetural têm sido propostos por arquitetos nos últimos anos a fim de se reduzir este empecilho. Um dos mais famosos é o modelo 4+1 apresentado por ([KRUCHTEN, 1995](#)). Este modelo propõe o entendimento e descrição de uma arquitetura de software em quatro visões. Esta divisão permite que a complexidade de uma arquitetura seja melhor analisada e compreendida. Uma ilustração do modelo é apresentada na figura 1.

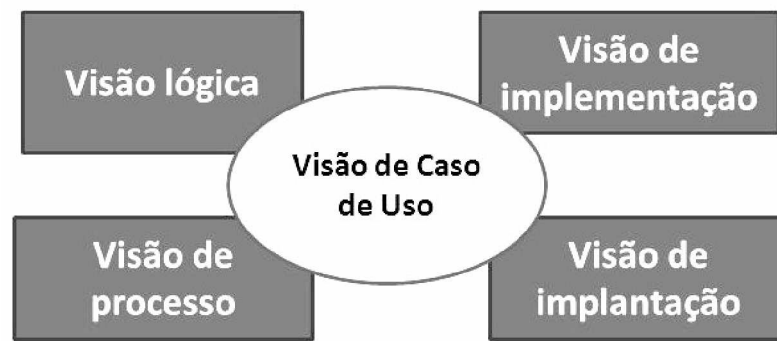


Figura 1 – Visão geral do modelo 4+1 descrito por (KRUCHTEN, 1995)

- *Visão lógica*: descreve os elementos arquiteturais da arquitetura e os relacionamentos entre eles. A descrição acontece por meio de diagramas de classes ou equivalentes.
- *Visão de implementação*: retrata a organização interna dos componentes da arquitetura no ambiente de desenvolvimento. Esta visão normalmente apresenta a hierarquia de pacotes e classes utilizada em uma aplicação.
- *Visão de processo*: descreve a comunicação e concorrência entre os elementos da arquitetura, como exemplo a descrição de componentes multi-tarefas ou mecanismos de comunicação síncronos ou assíncronos utilizados.
- *Visão de implantação*: apresenta o modelo de mapeamento de processos e componentes nos elementos físicos. Pode descrever, por exemplo, como uma base de dados de uma aplicação é acessada e distribuída nos servidores.

2.3 Desafios

O uso de uma arquitetura de software possui diversas vantagens, como já apresentado ao longo deste capítulo. Entretanto, uma série de desafios também pode ser observada. Alguns dos desafios identificados por (BOSCH, 2004) são apresentados abaixo:

- *Falta de representação de primeira classe*: a medida que um conjunto de decisões arquiteturais é tomada, fica difícil identificar o efeito implícito de cada uma delas individualmente na arquitetura resultante. Consequentemente, as razões e conhecimento sobre as particularidades de uma arquitetura são rapidamente perdidas.
- *Decisões de projeto entrelaçadas*: muitas vezes as decisões arquiteturais afetam múltiplos componentes da arquitetura e se tornam interligadas com outras decisões.
- *Alto custo de mudança*: um problema bastante comum é a dificuldade em se modificar uma arquitetura após o término da implementação de um sistema. A mudança de uma arquitetura geralmente apresenta alto custo pois afeta diversos componentes interligados de um sistema.

- *Regras de projeto e restrições violadas*: arquitetos e desenvolvedores podem facilmente violar as regras de projeto e restrições impostas à arquitetura durante o ciclo de vida do sistema.
- *Decisões arquiteturais obsoletas não removidas*: a remoção de funcionalidades implementadas é normalmente evitada ou esquecida durante o ciclo de desenvolvimento. Como consequência, o software se degrada rapidamente e o custo de manutenção do sistema aumenta.

2.4 Indicadores de Qualidade

A qualidade de uma arquitetura pode ser questionada por meio da formulação de um questionário de avaliação do software produzido. É possível medir cada atributo de qualidade de forma relativamente simples. Alguns exemplos são apresentados abaixo:

Desempenho

- O software roda dentro do intervalo de tempo desejado?
- O software apresenta alguma lentidão em determinada seção do sistema?

Segurança

- Há alguma falha que permite o acesso ao sistema por alguém que não possua credenciais autorizadas?
- O envio e recebimento de dados e mensagens é realizado de forma criptografada?

Funcionalidade

- O software atende as expectativas de funcionamento do usuário final?
- As funcionalidades desenvolvidas atuam como o esperado?

Facilidade de utilização

- A interface do software é amigável ao usuário?
- O usuário relata alguma dificuldade em navegar pelo sistema?

Manutenibilidade

- O código fonte do sistema pode ser facilmente revisado, alterado ou incrementado?
- A alteração de uma funcionalidade do sistema prejudicará o funcionamento de outras?

3 Revisão Sistemática da Literatura

A Revisão Sistemática da Literatura (RSL) é uma metodologia de estudo que permite identificar, avaliar e interpretar material de pesquisa relevante à uma determinada questão de pesquisa ou tópico de interesse de modo confiável e rigoroso. Abaixo, é apresentado uma lista de possíveis razões identificadas por (KITCHENHAM; CHARTERS, 2007) e (OKOLI; SCHABRAM, 2010) para a realização de uma revisão sistemática:

- Resumir o estado da arte de uma área de conhecimento ou tópico de interesse por meio de evidências empíricas.
- Fornecer um quadro de pesquisa para posicionamento apropriado e direcionado de novas atividades de pesquisa.
- Identificar eventuais lacunas em pesquisas atuais de um área de conhecimento a fim de sugerir novas tópicos de investigação.
- Responder questões práticas por meio da compreensão do estado da arte em determinado tópico.
- Auxiliar no processo de geração ou validação de novas hipóteses de pesquisa.

As principais vantagens apontadas por (KITCHENHAM; CHARTERS, 2007) para a condução de uma revisão sistemática são:

- Permite identificar evidências consistentes em um amplo conjunto de publicações, bem como resultados inconsistentes ou questionáveis.
- A metodologia de revisão quando bem definida reduz a probabilidade dos resultados da literatura serem tendenciosos, apesar de não prevenir a possibilidade de existência de vieses de pesquisa nos estudos primários considerados.
- É possível combinar dados quantitativos por meio de técnicas de meta-análise, o que aumenta a possibilidade de detecção de efeitos reais de um fenômeno, aspecto difícil de ser detectado em trabalhos menores.

Em contrapartida, a principal desvantagem se refere ao esforço necessário para a execução de uma revisão sistemática, que é consideravelmente maior do que em métodos de revisão de literatura tradicionais devido ao rigor do método.

3.1 Processo de Revisão

Segundo (KITCHENHAM; CHARTERS, 2007), as seguintes fases são detectadas como primordiais durante a execução de uma revisão sistemática da literatura no âmbito de arquitetura de software: planejamento, execução e conclusão da revisão. A seguir, é apresentada uma enumeração das principais atividades associadas a cada uma das fases. É importante destacar que, as atividades não são obrigatoriamente sequenciais e podem demandar refinamento constante durante a execução da revisão sistemática, como exemplo as atividades de seleção de estudos primários e extração do dados.

Planejamento

- Identificação da necessidade da revisão.
- Comissionamento da revisão.
- Definição das questões de pesquisa.
- Desenvolvimento de um protocolo de execução da revisão.
- Avaliação do protocolo de revisão definido.

Execução

- Identificação de pesquisa correlata.
- Seleção dos estudos primários.
- Avaliação da qualidade do estudos.
- Extração de dados.
- Síntese dos dados.

Conclusão

- Especificação dos mecanismos de disseminação.
- Formatação do relatório contendo as conclusões obtidas.
- Avaliação do relatório final contendo as conclusões da revisão sistemática.

O comissionamento de uma revisão é um estágio opcional que é realizado comumente em revisões realizadas com fins comerciais. A avaliação do protocolo de revisão e relatório final também são estágios considerados opcionais e dependem dos critérios de qualidade definidos pelo(s) autor(es) da revisão.

3.1.1 Planejamento

A fase de planejamento tem o intuito de confirmar a necessidade da execução de uma revisão sistemática no domínio pretendido, o que inclui a definição das questões de pesquisa e desenvolvimento de um protocolo contendo os procedimentos de execução.

A formulação das questões de pesquisa é uma etapa essencial que deve ser realizada de forma cuidadosa, uma vez que as questões a serem respondidas ao longo do trabalho moldam o formato da metodologia de revisão.

Após a definição das questões de pesquisa é necessário definir o protocolo de revisão, responsável por guiar a execução do processo de revisão sistemática e reduzir a possibilidade de vieses de pesquisa. Entre os componentes inclusos em um protocolo estão:

- *Justificativa da revisão*: detecta a necessidade da revisão e objetivos a serem alcançados através de sua realização.
- *Questões de pesquisa*: identifica as questões de pesquisa que o processo de revisão deve ser capaz de responder.
- *Estratégia de pesquisa dos estudos primários*: define a estratégia de busca dos estudos primários incluindo termos de consulta e plataformas de busca a serem utilizadas.
- *Crterios de seleção dos estudos*: determinam os critérios para a inclusão e exclusão de estudos relevantes ao processo de revisão.
- *Procedimentos de seleção dos estudos*: descreve como os critérios de seleção serão aplicados durante o processo de seleção de estudos relevantes.
- *Formulário de avaliação da qualidade dos estudos*: tem o propósito de avaliar o grau de relevância individual de cada um dos estudos coletados e permite identificar o nível de relevância de cada um para o alcance dos propósitos da revisão.
- *Estratégias de extração de dados*: define quais informações serão obtidas de cada estudo e os métodos a serem utilizados para a extração dessas informações.
- *Síntese dos dados extraídos*: sumariza os dados relevantes extraídos de cada estudo.
- *Estratégia de disseminação da revisão*: apresenta os meios de disseminação dos resultados da revisão, como exemplos: publicação de artigo em periódico, disponibilização online de relatório final.
- *Cronograma de execução*: define o cronograma de execução previsto para a execução do processo de revisão.

Assim que o protocolo de revisão estiver definido, sua consistência deve ser avaliada a fim de confirmar que os procedimentos de pesquisa, análise e extração dos dados propostos são apropriados para responder as questões de pesquisa formuladas.

3.1.2 Execução

A fase de execução é iniciada assim que o protocolo de revisão é avaliado como satisfatório e não segue obrigatoriamente um fluxo iterativo de ações, uma vez que as atividades podem ser intercaladas e constantemente refinadas ao longo do processo de revisão. As atividades principais dessa fase são: identificação de pesquisa correlata, seleção dos estudos primários para análise, avaliação da qualidade dos estudos selecionados, extração dos dados e síntese dos dados extraídos. A seguir, uma descrição geral das atividades desempenhadas ao longo da fase de execução.

A identificação de pesquisa correlata é a primeira atividade a ser desempenhada e tem como intuito a coleta da maior quantidade possível de estudos primários relacionados às questões de pesquisa propostas. A atividade deve ser conduzida de forma metódica, seguindo estratégias de busca rigorosas que garantam a relevância do material coletado. Técnicas como a execução de buscas preliminares e a combinação de termos de pesquisa são comumente empregadas como instrumentos de validação e aprimoramento da qualidade das estratégias de busca.

O processo de busca deve ser documentado em detalhes, incluindo informações como as plataformas acessadas para consulta de material, a data em que as consultas foram realizadas, o período de publicação considerado e eventuais critérios de busca adicionais. O intuito principal dessa documentação é comprovar a transparência do estudo e permitir que outros pesquisadores sejam capazes de reproduzir o processo de busca executado. Além disso, é fundamental que justificativas sejam apresentadas caso critérios de busca definidos no protocolo sejam modificados ou descartados.

À medida que estudos potencialmente relevantes são coletados, é preciso dar início ao processo de seleção e avaliação da relevância de cada estudo em relação às questões de pesquisa definidas. Para isso, são definidos critérios de seleção que estabelecem princípios para a inclusão e/ou exclusão de material. Em caráter complementar, é conduzida a avaliação da qualidade dos estudos para a inclusão e/ou exclusão de critérios de análise e seleção ainda mais detalhados, o que é importante também para a estipulação de graus de importância para os estudos selecionados.

O formulário de extração de dados, definido durante o desenvolvimento do protocolo de revisão, deve garantir a extração de todos os dados necessários para responder as questões de pesquisa. Além de todas as questões imprescindíveis, é fundamental a inclusão de dados básicos como o nome do revisor, a data de extração dos dados, a referência bibliográfica completa da publicação e um espaço em branco para anotações adicionais. É importante destacar que, caso haja múltiplas publicações de um mesmo estudo, apenas o estudo mais completo deve ser levado em consideração para evitar a adição de duplicatas que prejudiquem a qualidade da revisão.

A síntese dos dados envolve sumarizar os resultados encontrados por meio da extração de dados dos estudos primários. É uma atividade que pode mesclar tanto dados quantitativos quanto dados qualitativos, como exemplo a escrita de uma narrativa descritiva das conclusões obtidas acrescida de tabelas de dados quantitativos. O único ponto de atenção a ser observado é que os dados quantitativos devem sempre incluir o tamanho da amostra levada em consideração e a unidade de medida utilizada.

3.1.3 Conclusão

A fase final da revisão diz respeito a documentação das conclusões obtidas e disseminação do relatório final da revisão. A formatação do relatório final é definida pelo teor do estudo e plataforma de publicação selecionada pelo(s) autor(es), com possibilidade de restrições quanto ao tamanho do documento e formato de apresentação do conteúdo. A avaliação posterior é, comumente, realizada por pesquisadores e especialistas da área de conhecimento a que o estudo pertence.

3.2 Estudo de Caso

Um exemplo de revisão sistemática é apresentado na íntegra no capítulo 5, por meio de um estudo de caso no âmbito de sistemas tutores inteligentes orientados ao ensino de programação. As fases de planejamento, execução e conclusão da revisão executadas são descritas em detalhes e transmitem uma visão precisa das atividades desempenhadas em cada uma das fases do processo de revisão.

4 Sistemas Tutores Inteligentes

Um Sistema Tutor Inteligente (STI) consiste em um sistema educacional computadorizado que contém um componente de Inteligência Artificial. Um sistema com tais características objetiva simular o comportamento humano perante um processo de aprendizagem qualquer através da aplicação de técnicas inerentes à ciência cognitiva, termo que comumente se refere à intersecção das áreas de ciência computacional, psicologia cognitiva e pesquisa educacional (NWANA, 1990). Algumas das características ideais em um sistema tutor são apontadas por (SOTTILARE, 2015):

- **Auto-regulação:** suporte ao aprendizado tanto de indivíduos quanto de times.
- **Adaptatividade:** modelagem das instruções de acordo com as necessidades e preferências do(s) usuário(s).
- **Exatidão:** utilização de métodos de instruções precisos.
- **Usabilidade:** acessibilidade à diferentes tipos de usuários.
- **Disponibilidade:** disponível ao usuário sempre que for de seu anseio.

(SOTTILARE, 2015) discorre ainda que o uso de sistemas tutores é recomendado para atividades de ensino nos seguintes domínios:

- **Cognitivo:** tomada de decisões, pensamento estratégico e solução de problemas.
- **Afetivo:** habilidades interpessoais e conduta ética.
- **Psicomotor:** operação de equipamentos e/ou plataformas sofisticadas.
- **Social:** colaboração e atividades em equipes.

Há uma ampla diversidade de benefícios associados à utilização de sistemas tutores inteligentes. Dentre os principais estão: o sequenciamento de conteúdo personalizado, o suporte à solução interativa de problemas e a análise inteligente das soluções de um usuário. Esses fatores são responsáveis pela diferenciação dos STIs dos sistemas convencionais de instrução assistida, uma vez que as características citadas são destinadas à simulação do comportamento humano em um ambiente de ensino tradicional, cenário que não é suportado por sistemas de tutoria tradicionais (BRUSILOVSKY; SCHWARZ; WEBER, 1996).

O emprego de sistemas de tutoria inteligente representa, então, uma estratégia fundamental em direção à introdução de adaptatividade em metodologias de ensino eletrônicas, uma vez que o modelo independe de intervenção humana durante o processo de personalização do conteúdo apresentado ao usuário.

A arquitetura clássica de um sistema tutor inteligente, representada na figura 2, é composta por quatro componentes que se interrelacionam: o módulo do domínio, o módulo do estudante, o módulo pedagógico e o módulo da comunicação (WOOLF, 2009). Entretanto, é importante destacar que esta arquitetura pode ser modificada de acordo com o domínio e as necessidades específicas observadas em uma aplicação, o que pode conferir maior flexibilidade e operabilidade à um sistema.

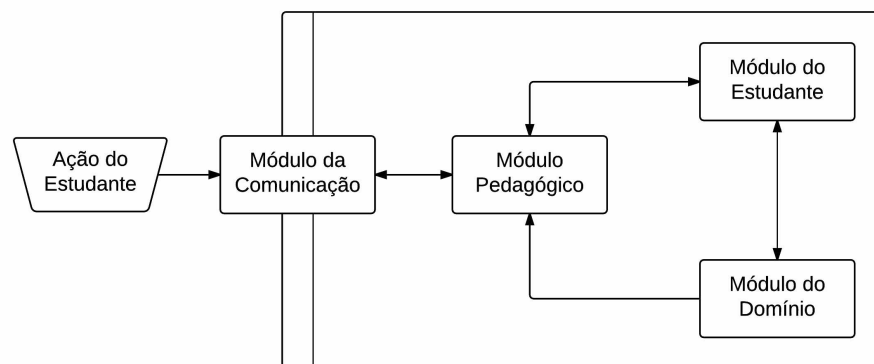


Figura 2 – Arquitetura clássica de um Sistema Tutor Inteligente

O **Módulo do Domínio** representa o conhecimento do especialista em um determinado domínio do conhecimento e contém o material didático a ser apresentado ao estudante no decorrer do aprendizado de determinado tópico ou programa de estudos (BUTZ; HUA; MAGUIRE, 2006). É fundamental representar o conhecimento de forma organizada de modo a viabilizar o crescimento incremental do domínio e uso eficiente dos recursos durante o processo de ensino (DORÇA, 2004).

O **Módulo do Estudante** representa o conhecimento do estudante no domínio do sistema e descreve os métodos de raciocínio sobre o seu conhecimento (WOOLF, 2009). O módulo armazena as informações individuais do estudante tais como as suas preferências, o nível de conhecimento, os objetivos, o histórico de navegação, o desempenho e quaisquer dados adicionais considerados relevantes ao processo de inferência do sistema. O histórico de aprendizagem do estudante é acessado pelo módulo pedagógico durante o processo de seleção dos objetos de aprendizagem mais adequados para o sequenciamento de conteúdo de um estudante (DORÇA, 2004).

O **Módulo Pedagógico** determina as estratégias de ensino mais adequadas ao estudante com base nas informações contidas no sistema, no módulo do domínio e no módulo do estudante (DORÇA, 2004). O módulo também é encarregado da seleção das atividades e

exemplos para exibição, correção dos exercícios resolvidos pelo estudante, retorno de informações sobre o seu desempenho e avaliação do progresso no domínio do conhecimento do sistema (CRUZ, 2007).

O **Módulo da Comunicação** contém a interface de interação entre estudantes e computadores, como interfaces gráficas, agentes animados ou mecanismos de diálogo (WOOLF, 2009). Não há um grau de complexidade definido para a sua implementação, entretanto é importante considerar que uma interface de usuário bem desenhada pode impactar significativamente na clareza com a qual o estudante absorve as instruções e informações retornadas pelo sistema (BUTZ; HUA; MAGUIRE, 2006).

Considere o seguinte cenário prático que exemplifica o papel de cada um desses módulos durante a apresentação de um objeto de aprendizagem. O módulo da comunicação é responsável pela exibição do objeto e captura das interações realizadas pelo estudante, tanto na leitura quanto na resolução de um exercício associado. Assim que o exercício é solucionado, o módulo pedagógico avalia a solução do estudante com o apoio do módulo do domínio e do módulo do estudante. O módulo do domínio contém a resposta correta para o exercício, enquanto que o módulo do estudante contém as características individuais do aluno. O módulo pedagógico avalia, então, a corretude da solução enviada pelo estudante e escolhe o nível de *feedback* a ser apresentado através do módulo da comunicação.

Há casos em que as atribuições do módulo pedagógico são fragmentadas por meio da criação de um novo módulo chamado *módulo especialista*. As responsabilidades de cada um desses módulos serão detalhadas nas próximas seções a fim de diferenciá-los nos casos em que são descritos isoladamente.

4.1 Módulo Pedagógico

O Módulo Pedagógico é composto por três componentes funcionais básicos: o planejador de instruções, o mecanismo de retorno de informações e o componente de avaliação (JEREMIĆ; JOVANOVIĆ; GAŠEVIĆ, 2009). O planejador de instruções é responsável pelo sequenciamento dos objetos de aprendizagem e tipos de atividades apropriados ao usuário. O mecanismo decide os critérios de apresentação de conselho e recomendações ao usuário durante o processo de aprendizagem. O componente de avaliação é responsável pela coleta das informações durante o processo de aprendizagem e avaliação do progresso demonstrado pelo usuário com a utilização do sistema.

O planejador de instruções determina as estratégias pedagógicas mais pertinentes ao processo de aprendizagem. As estratégias pedagógicas são planos de ações que possuem como objetivo estabelecer a ordem de apresentação de conteúdo e atividades mais adequadas ao nível de conhecimento do usuário. Entretanto, definir os mecanismos de geração automática de estratégias pedagógicas é uma tarefa complexa e trabalhosa devido ao grande

número de variáveis envolvidas no processo e requer que um sistema tutor possua uma base de conhecimento extensa sobre o domínio do conhecimento. Caso contrário, recomendações inadequadas podem ser apresentadas ao usuário, o que pode impactar negativamente em seu aprendizado (PIÑERES; JOSYULA; BUILES, 2015).

O mecanismo de retorno de informações é responsável pela apresentação de dicas e conselhos ao usuário durante seu processo de aprendizado. O feedback pode ser negativo, positivo ou ambos dependendo da implementação do sistema e domínio suportado.

Definir o mecanismo de feedback é uma das principais dificuldades no domínio de programação. Muitos sistemas são incapazes de apresentar feedback instantâneo ao estudante em razão do grande número de resoluções válidas que exercícios de programação geralmente apresentam. Além disso, a apresentação em momentos inapropriados ou em quantidade exagerada pode resultar em prejuízo no aprendizado (WERAGAMA, 2013).

4.1.1 Ressalvas

A maioria dos sistemas são interessados na adaptação apenas sob uma perspectiva de apresentação de conteúdo e geralmente ignoram o aspecto pedagógico inerente ao processo de aprendizagem. Isto ocorre devido à muitas das informações estarem codificadas no módulo do domínio e serem limitadas à semântica de relacionamentos pedagógicos (pré-requisito, faz-parte-de, etc.) (HADJI; CHOI; JEMNI, 2012).

Os usuários geralmente não possuem a opção de navegar através do material de aprendizagem no momento em que são concedidas recomendações pelo sistema (HOOSHYAR et al., 2015).

4.2 Módulo Especialista

A principal atribuição do Módulo Especialista consiste em auxiliar o Módulo Pedagógico na tomada de decisões durante o sequenciamento dos objetos de aprendizagem e apresentação adaptativa do material de ensino. Esta tarefa é comumente desempenhada pelo Módulo Pedagógico em diversos sistemas tutores tradicionais, contudo, como constatado por (JEREMIĆ; JOVANOVIĆ; GAŠEVIĆ, 2009) e (DORÇA, 2004), a separação em dois módulos distintos aumenta a flexibilidade de um sistema, uma vez que permitiria uma fácil manutenção. Isto é, mudanças em um módulo poderiam ser realizadas sem impacto direto na estrutura de outro módulo. Além disso, há ainda um potencial poder de reuso de módulos bem projetados. (HADJI; CHOI; JEMNI, 2012).

A separação não impacta na composição e nos papéis do Módulo Pedagógico. A diferença é que todo o papel de decisão é delegado exclusivamente para o Módulo Especialista.

As principais técnicas de representação do conhecimento do especialista são exploradas na seção seguinte, o que permite uma compreensão mais detalhada das várias abordagens existentes de inferência sobre as informações de um estudante para a geração das estratégias de ensino mais apropriadas.

4.2.1 Técnicas de Representação do Conhecimento do Especialista

De acordo com (CRUZ, 2007) existem várias técnicas em IA comumente utilizadas para representação do conhecimento do especialista, sendo as principais: Raciocínio Baseado em Casos, Processo de Decisão de Markov, Neuro-Fuzzy e Redes Bayesianas. As subseções seguintes descrevem as características gerais de cada uma das abordagens.

4.2.1.1 Raciocínio Baseado em Casos

O modelo de Raciocínio Baseado em Casos (RBC) é um paradigma de inferência de soluções para novas situações que se baseia em experiências similares previamente registradas (casos) e armazenadas em uma biblioteca para consulta (HUANG et al., 2012). Um caso pode ser interpretado como uma unidade de conhecimento especialista de um determinado domínio. Este método de descoberta indutivo considera a similaridade entre casos para a descoberta de uma solução válida por analogia.

O raciocínio é composto por um ciclo de quatro etapas: *recuperação*, *reutilização*, *revisão* e *retenção*. Na etapa de *recuperação*, o sistema identifica as características mais relevantes do problema atual e analisa a base de casos em busca de um subconjunto de casos similares. Na etapa seguinte de *reutilização*, o sistema busca uma solução para o problema atual por meio da adaptação da solução dos casos selecionados. Na etapa de *revisão*, o sistema valida a correção da solução proposta. Na etapa final de *retenção*, o sistema decide incrementar ou não a base com o novo caso solucionado (PRENTZAS; HATZILYGEROUDIS, 2011), (ONTAÑÓN; RAM, 2011). O ciclo completo é retratado na figura 3.

Entre as principais vantagens do modelo do Raciocínio Baseado em Casos apontadas por (PRENTZAS; HATZILYGEROUDIS, 2007) estão:

- Capacidade de expressar conhecimento especializado através de unidades de representação de conhecimento simples e bastante compreensíveis (casos).
- Aumento da modularidade de um sistema, dado que cada caso é uma unidade que pode ser facilmente inserida ou removida da base de casos do sistema.
- Capacidade em lidar com casos inesperados que ainda não foram registrados na base de casos do sistema através da avaliação por similaridade de relevância.

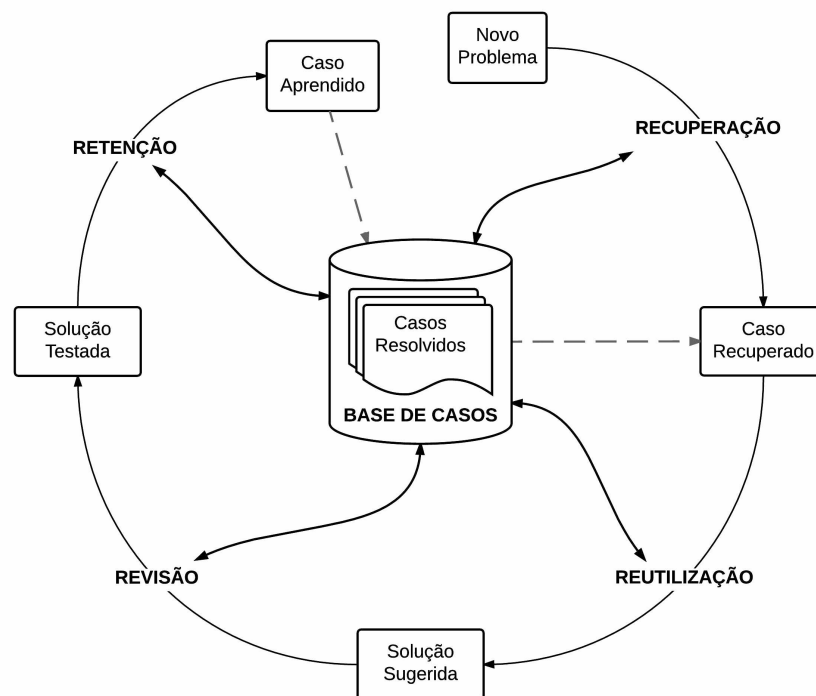


Figura 3 – Ciclo do Raciocínio Baseado em Casos

- Atualização automática da base de casos por meio da incorporação de novos casos durante o seu funcionamento, característica que facilita também a manutenção do sistema e contribui para a manipulação de casos inesperados.
- Eficiência na inferência de soluções, uma vez que adaptar casos pré-existentes é geralmente mais eficiente que resolver um problema por completo.

Há também algumas desvantagens para a sua utilização, tais como:

- Incapacidade de expressar conhecimento generalista, visto que um caso pode representar apenas conhecimento especializado dado a sua natureza.
- Problemas de aquisição de conhecimento em domínios onde os casos são inexistentes ou estão disponíveis em uma quantidade bastante limitada.
- Ineficiência da inferência quando o custo de recuperação de um caso aumenta devido ao crescimento da dimensão da base de casos do sistema e o tempo de processamento torna-se progressivamente maior por essa razão.
- Dificuldade em fornecer explicações sobre os passos executados durante o raciocínio de um problema, o que impacta na capacidade em oferecer *feedback* ao usuário.

Um exemplo de caso é apresentado na tabela 1. O exemplo consiste na avaliação da resposta enviada pelo estudante de *ID 001* em uma questão de nível *básico* pertencente ao tópico *Objeto* na linguagem de programação *Java*.

Tabela 1 – Exemplo de Caso

Descrição do Caso	
<i>Estudante (ID)</i>	001
<i>Nível</i>	Básico
<i>Linguagem</i>	Java
<i>Tópico</i>	Objeto
<i>Enunciado</i>	Qual o modificador de acesso torna os membros da classe acessíveis apenas as classes do mesmo pacote?
<i>Resposta Enviada</i>	Private
<i>Resposta Correta</i>	Protected
<i>Avaliação</i>	Resposta Incorreta

4.2.1.2 Processo de Decisão de Markov

O Processo de Decisão de Markov (PDM) é um modelo de tomada de decisão sequencial ideal para cenários onde o resultado é incerto. Neste modelo há um conjunto de tempos de decisão, estados, ações, recompensas e probabilidades de transições que dependem somente do estado e ação atuais, sem influência dos estados e ações executadas no passado (PUTERMAN, 2005). Alguns exemplos de aplicações do método em cenários reais incluem: roteamento de dados em redes, controle de manutenção e reparo de componentes de um sistema, controle de custos de inventários, controle ideal de filas, agendamento de tarefas (FEINBERG; SHWARTZ, 2002).

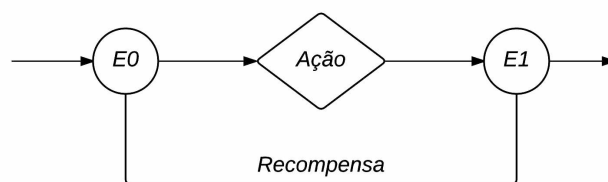


Figura 4 – Processo de recompensa do modelo de decisão sequencial

Em um processo de decisão sequencial, há um controlador que observa o estado atual de um sistema e seleciona uma ação a ser realizada com base no estado observado. Esta ação produz dois resultados: uma recompensa imediata e a evolução do sistema para um novo tempo de decisão de acordo com a distribuição probabilística determinada pela ação escolhida. Para cada estado há uma política de escolha associada contendo as regras de decisão que definem a próxima ação a ser executada. O processo de recompensa do modelo de decisão sequencial é ilustrado na figura 4.

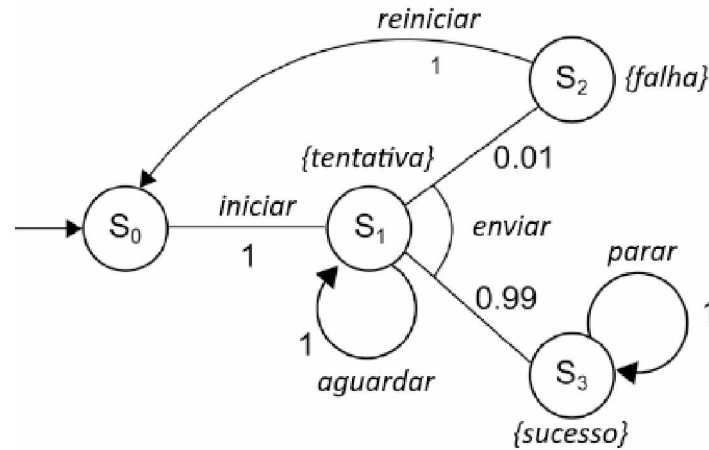


Figura 5 – Exemplo de um processo de decisão adaptado de (SAWICKI et al., 2015)

A figura 5 apresenta um exemplo de um processo de decisão de Markov para um protocolo de comunicação. As ações são iniciar, aguardar, enviar, reiniciar ou parar. Os estados são representados pelos círculos $S[n]$. Os estados são: estado inicial $S[0]$, tentativa $S[1]$, sucesso $S[2]$ e falha $S[3]$. A possibilidade de transição é um valor entre 0 e 1.

A ação *iniciar* representa o início da tentativa de envio de uma mensagem, acompanhado de uma escolha não-determinística entre *aguardar*, nos casos em que o canal de envio está ocupado, ou *enviar*, quando o canal estiver disponível. Caso a ação de envio ocorra, a probabilidade de sucesso na entrega da mensagem é de 0.99, enquanto que a possibilidade de falha é de 0.01. Em caso de sucesso, a ação *parar* é executada. Em caso de falha, a ação *reiniciar* é executada.

Em planejadores de instruções que utilizam esta técnica, o tutor deve determinar uma ação a ser executada em cada estado que corrobore em uma transição de estado benéfica ao estudante. Um estado representa o conhecimento do tutor sobre uma determinada situação. Este conhecimento é obtido através da combinação do conhecimento do estudante, uma resposta fornecida, ou qualquer outra informação derivada das ações registradas no sistema. Uma ação corresponde a qualquer interação realizada pelo estudante que pode conduzir a diferentes estados. Um evento é uma ação que também pode causar transições de estado (CRUZ, 2007).

As grandes vantagens deste método consistem na facilidade de análise, compreensão e implementação do mecanismo de transição de estados e o baixo nível de recursos computacionais exigidos para o seu desempenho. Entretanto, a difícil modelagem de sistemas complexos nos quais não há informações sobre o estado em que o sistema se encontra e a necessidade de conhecimento de todos os elementos que desencadeiam transições de estados durante a etapa de modelagem são desvantagens que dificultam a aplicação do método em certos cenários.

4.2.1.3 Neuro-Fuzzy

A técnica Neuro-Fuzzy (NF) é um modelo híbrido que combina os algoritmos de aprendizagem em redes neurais com o raciocínio da lógica Fuzzy. As redes neurais são estruturas de baixo nível capazes de lidar com informações não processadas. A lógica Fuzzy armazena conhecimento especialista em sua base de regras para o raciocínio posterior sobre argumentos de entrada e inferência de um valor médio como resultado. Através da combinação das duas técnicas, é possível explorar a capacidade de aprendizado de redes neurais com a capacidade de representação do conhecimento de forma linguística da lógica Fuzzy, tornando um sistema mais transparente (ARORA; SAINI, 2014).

Um sistema Neuro-Fuzzy é capaz de lidar com incerteza de modo eficaz e funciona do seguinte modo: a rede neural determina os valores dos parâmetros, enquanto que a lógica Fuzzy lida com as regras condicionais. Para tanto, é necessário seguir um conjunto de etapas que inclui a definição de valores de entrada e saída, a definição de conjuntos fuzzy para a classificação dos valores de entrada, a definição de regras fuzzy e, por último, a criação e treinamento da rede neural (SEVARAC, 2006). Uma rede Neuro-Fuzzy básica é ilustrada na figura 6. Na rede apresentada, os valores de entrada são representados por $E1$ e $E2$, enquanto que os valores de saída são representados por $S1$, $S2$ e $S3$.

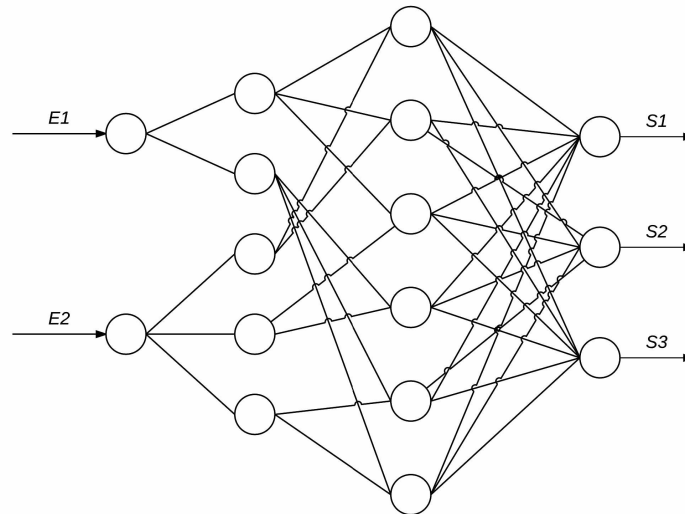


Figura 6 – Exemplo de rede Neuro-Fuzzy

É importante citar que a complexidade de interpretação de uma rede neural e a necessidade de definição prévia de um conjunto extenso de parâmetros são algumas das principais desvantagens apontadas por (CRUZ, 2007) para a utilização do método.

4.2.1.4 Redes Bayesianas

As Redes Bayesianas (RB) são projetadas para lidar com as relações de dependência e independência entre todas as variáveis pertencentes a um determinado domínio, permitindo prever o comportamento de variáveis desconhecidas com base nos valores das variáveis conhecidas. Portanto, é um modelo de inferência probabilístico que pressupõe que qualquer variável pode se comportar tanto como incógnita quanto como evidência de acordo com o caso em análise (CRUZ, 2007). Dessa forma, é possível tomar decisões até mesmo em cenários onde não há informações suficientes que corroborem a validade de determinada ação ou escolha. Este raciocínio realizado sobre incertezas é a principal vantagem do raciocínio probabilístico sobre o raciocínio lógico e tem sido bastante utilizado em STIs para o raciocínio dos estados cognitivos de um aluno.

As RBs constituem um modelo gráfico que representa, de forma simples, as relações de causalidade das variáveis de um sistema. Para a construção de uma RB é necessário que haja, primeiramente, a definição de um conjunto de variáveis e arcos ligando as variáveis. Em seguida, cada variável deve conter um conjunto limitado de estados mutuamente exclusivos para que as variáveis e os arcos formem um grafo dirigido sem ciclos. Finalmente, para cada variável existirá uma tabela com a probabilidade da variável em relação ao seu conjunto de relações. Este modelo de raciocínio desempenhado pelas RBs podem torná-las mais atrativas em relação as técnicas anteriormente apontadas devido as seguintes características observadas por (LUNA, 2004):

- *Aquisição de conhecimento*: possibilidade de juntar várias categorias de conhecimentos diferentes em um mesmo modelo.
- *Representação de conhecimentos*: permite a representação gráfica de forma intuitiva e compreensível para não especialistas, facilitando a validação e evolução do modelo.
- *Utilização de conhecimentos*: é possível utilizar o mesmo modelo na avaliação, predição e aperfeiçoamento de decisões, diminuindo os esforços na construção e atualização da rede de raciocínio.

Os seguintes passos são necessários para a construção de uma rede Bayesiana:

- Definir uma ordenação para as variáveis, enquanto houver variáveis no conjunto.
- Selecionar uma variável X e adicionar um nó para ela na rede.
- Definir o nó raiz de X de tal modo que a propriedade de independência condicional seja satisfeita.
- Criar a tabela de probabilidades condicionais de X.

Para ilustrar o funcionamento de uma rede bayesiana, considere o seguinte exemplo representado na figura 7: um novo alarme contra assaltantes é instalado em uma residência, mesmo sendo muito confiável na detecção de assaltos ele pode disparar caso ocorra um terremoto. Os dois vizinhos João e Maria se disponibilizaram a telefonar caso o alarme dispare. João sempre liga quando ouve o alarme disparar, porém algumas vezes ele confunde o alarme com o telefone e também liga nestes casos. Já a Maria gosta de ouvir música alta e às vezes não ouve o alarme disparar, não ligando nestes casos.

A ação de João e Maria telefonarem são condicionalmente independentes do roubo e do terremoto dado o disparo do alarme. As probabilidades são representadas através de tabelas de probabilidades condicionais. A probabilidade em cada linha corresponde a um caso condicional. Para cada número n de nós raiz, haverá uma tabela de 2^n entradas para a variável. Caso o nó não possua um nó raiz, sua probabilidade é pré-definida.

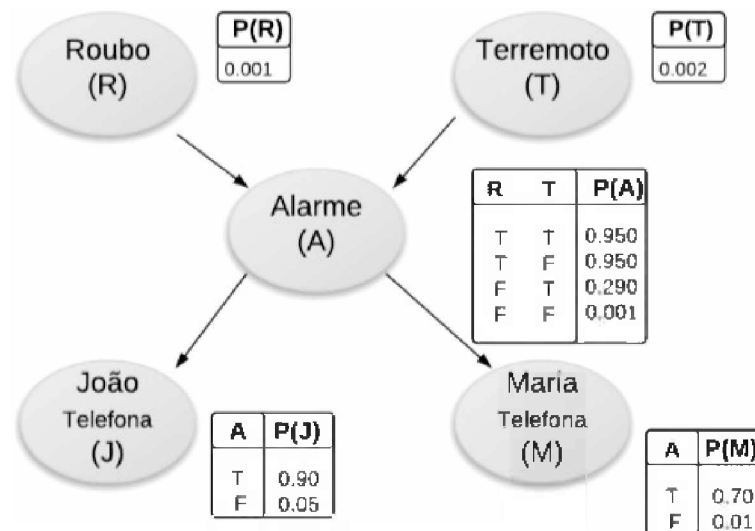


Figura 7 – Exemplo de raciocínio de uma rede Bayesiana

5 Estudo de Caso

Este capítulo apresenta o estudo de caso selecionado para validar a hipótese do trabalho de que uma revisão sistemática da literatura permite a definição de uma arquitetura básica para um sistema. Esse estudo de caso pertence à área de sistemas tutores inteligentes orientados ao ensino de programação. As fases de planejamento, execução e conclusão são apresentadas nesta ordem e contém uma descrição detalhada das atividades desempenhadas ao longo do processo de revisão.

5.1 Planejamento

Esta seção descreve a fase de planejamento da revisão, incluindo a justificativa para a sua realização e protocolo de revisão formulado para a sua execução.

5.1.1 Justificativa da Revisão

A revisão sistemática executada neste trabalho tem como objetivo primário confirmar a hipótese do trabalho de que por meio de sua realização é possível identificar uma arquitetura de software básica para um sistema. Os sistemas tutores inteligentes orientados ao ensino de programação são escolhidos como tópico de interesse para a execução da revisão e contextualizam a revisão em um cenário real.

5.1.2 Protocolo de Revisão

O protocolo de revisão tem como objetivo formalizar o processo de identificação e seleção de material literário adequado aos fins da pesquisa. Os critérios estabelecidos determinam a conformidade de cada documento encontrado durante as buscas.

5.1.3 Coleta de Material

As seguintes plataformas eletrônicas são utilizadas durante o processo de identificação e seleção de material relevante:

- ACM Digital Library ¹
- CiteSeerX ²

¹ <http://dl.acm.org>

² <http://citeseerx.ist.psu.edu>

- IEE Xplore Digital Library ³
- Google Scholar ⁴
- Microsoft Academic ⁵
- ResearchGate ⁶

O ano de publicação ou quantidade de citações de um trabalho não é um critério impeditivo durante o processo de coleta do material. Entretanto, há uma ordem de prioridade como descrita abaixo:

1. Alta prioridade para trabalhos recentes publicados entre o período de 2010 e 2016, independente do número de citações.
2. Média prioridade para trabalhos publicados antes do ano 2010 e que possuam um número superior a 10 citações.
3. Baixa prioridade para trabalhos publicados antes do ano 2010 e que não possuam um número superior a 10 citações.

5.1.4 Estratégias de Busca

Pesquisas textuais são executadas em cada uma das plataformas estipuladas. As seguintes sequências de termos são definidas para a identificação de material. As linguagens de programação Java, C++, PHP e Haskell são incluídas em termos de consulta separados devido a popularidade que possuem na comunidade computacional. As aspas indicam que o resultado deve conter exatamente a palavra ou conjunto de termos definidos, enquanto que o termo booleano AND denota que o resultado deve satisfazer ambas as sequências de termos pesquisadas.

- tutor system AND “programming”
- tutoring system AND “programming”
- tutor system AND “computer programming”
- tutoring system AND “computer programming”
- tutor system AND “java”

³ <http://ieeexplore.ieee.org>

⁴ <http://scholar.google.com>

⁵ <http://academic.microsoft.com>

⁶ <http://researchgate.net>

- tutoring system AND "java"
- tutor system AND "c++"
- tutoring system AND "c++"
- tutor system AND "php"
- tutoring system AND "php"
- tutor system AND "haskell"
- tutoring system AND "haskell"

5.1.5 Resultados e Documentação

Uma total de vinte e seis documentos foi selecionado durante o processo de revisão sistemática da literatura. Para o gerenciamento do material selecionado e indicação da relevância de cada publicação são utilizados rótulos, baseado em escala de cores. O processo de seleção e categorização é apresentado de forma detalhada na subseção 5.1.7.3. Nos casos em que uma publicação é encontrada em mais de uma plataforma, apenas o documento mais recente foi incluído.

5.1.6 Validação da Estratégia de Pesquisa

Consultas experimentais são executadas nos buscadores estipulados com o intuito de validar a qualidade da formulação dos conjuntos de termos definidos para a seleção dos estudos. O número total de documentos retornados durante a execução de cada consulta é um indicador da qualidade da formulação. Como exemplo, a execução da consulta (tutoring system AND "computer programming") no Google Scholar retornou os seguintes documentos identificados previamente como relevantes para a revisão:

- A web-based bayesian intelligent tutoring system for computer programming. *Web Intelligence and Agent Systems* (BUTZ; HUA; MAGUIRE, 2006).
- A tutoring system for parameter passing in programming languages. In: ACM. *ACM SIGCSE Bulletin* (SHAH; KUMAR, 2002).
- Expert tutoring system for teaching computer programming languages. *Expert Systems with Applications* (EL-KHOULY; FAR; KOONO, 2000).

5.1.7 Critérios de Seleção

A subseção seguinte descreve os critérios definidos para a inclusão ou exclusão de estudos durante o processo de busca.

5.1.7.1 Inclusão de Material

- Publicações somente são incluídas caso abordem o uso de sistemas tutores inteligentes no contexto de ensino de programação.
- Trabalhos que envolvam estudos empíricos são incluídos.
- Caso vários documentos reportem o mesmo estudo apenas o mais recente é incluído no processo de revisão.

5.1.7.2 Exclusão de Material

- Publicações são desconsideradas caso não abordem o uso de sistemas tutores inteligentes no contexto de ensino de programação.
- Publicações que proponham abordagens de desenvolvimento, mas que não descrevam a metodologia de desenvolvimento da proposta são desconsideradas.
- Publicações são desconsideradas caso o documento completo não esteja disponível nas plataformas de consulta utilizadas ou não seja encontrado em mecanismos de buscas auxiliares.
- Publicações são desconsideradas caso não tenham sido publicados em Português, Inglês ou Espanhol.

5.1.7.3 Processo de Seleção

O processo de seleção dos estudos acontece ao longo de todo o desenvolvimento do trabalho para garantir que estudos relevantes encontradas após a seleção inicial não sejam descartados, bem como permitir que a seleção inicial possa ser reavaliada e modificada posteriormente. Ao adotar estes critérios, é possível validar a consistência do material coletado de modo contínuo e cuidadoso. A seleção é composta por três fases:

1. A conformidade de cada estudo encontrado durante a busca inicial é analisada baseada em seu título, resumo e conclusão. Literatura que esteja claramente fora dos critérios estabelecidos é descartada sem possibilidade de análise posterior.
2. Os estudos selecionados são categorizadas de acordo com os seguintes critérios:

- Verde: publicações altamente relevantes.
- Azul: publicações altamente relevantes e complementares. Indica casos em que etapas distintas de um estudo são publicadas separadamente. Como exemplo, uma publicação descreve um sistema do ponto de vista conceitual e outra apresenta um experimento realizado com um protótipo.
- Amarelo: publicações relevantes.
- Vermelho: publicações relevantes porém desclassificadas por algum dos critérios presentes na subseção [5.1.7.2](#).

3. Análise na íntegra do conteúdo das publicações e extração dos dados relevantes de acordo com a ordem de leitura definida na fase anterior.

5.1.7.4 Avaliação da Relevância da Publicação

A relevância de cada publicação selecionada é avaliada nesta etapa. O processo de avaliação acontece durante a seleção do material para assegurar que todos os documentos selecionados são relevantes para o processo de revisão sistemática. O conjunto de 12 critérios definidos para avaliação são:

1. O conteúdo da publicação é baseado em uma pesquisa de caráter científico?
2. Os objetivos da pesquisa são claramente definidos?
3. A arquitetura do sistema apresentado é claramente definida?
4. O mecanismo de inferência de dados do sistema é claramente descrito?
5. O processo de feedback do usuário é claramente apresentado?
6. É apresentado um protótipo do sistema?
7. O protótipo, caso presente, possui suporte à personalização de preferências de uso pelo usuário?
8. É realizado um experimento contendo um grupo de controle?
9. Os dados foram coletados de forma satisfatória?
10. A análise dos dados coletados é rigorosa?
11. Os resultados obtidos foram claramente apresentados?
12. A contribuição do estudo é discutida?

Os primeiros dois critérios são utilizados para a exclusão preliminar de estudos que são claramente insatisfatórios para o processo de revisão sistemática. Os dez critérios seguintes são utilizados para determinar a relevância de cada publicação para o estudo. As respostas de cada questão são inseridas em uma planilha com atribuição de um valor 10 (“SIM”), 5 (“PARCIALMENTE”) ou 0 (“NÃO”). Consulte o apêndice [A](#) para visualização completa do formulário de avaliação.

5.1.8 Extração de Dados

Esta subseção descreve o modo de extração dos dados necessários de cada publicação selecionada para estudo. Este processo garante que as questões da pesquisa são devidamente respondidas ao término do processo de revisão.

5.1.8.1 Processo de Coleta

Todos os dados extraídos das publicações são comparados para assegurar o cumprimento do princípio da imparcialidade da revisão por meio da exclusão de dados tendenciosos. Os seguintes dados são coletados:

1. Referência bibliográfica.
2. Resumo da publicação.
3. Tipo da publicação (exemplos: artigo, livro, relatório técnico, etc.)
4. Objetivos do trabalho.
5. Metodologia de pesquisa.
6. Divisão arquitetural do sistema.
7. Técnica de inferência de dados utilizada pelo sistema.
8. Tipo(s) de feedback utilizado(s) suportados pelo módulo pedagógico.
9. Caso o estudo inclua um experimento, o número de participantes é extraído.
10. Caso o estudo inclua um experimento, a técnica de coleta de dados é extraída.
11. Resultados obtidos com a pesquisa.
12. Considerações finais sobre os resultados.
13. Avaliação da relevância do estudo.

Todos os dados extraídos são armazenados em documentos de texto identificados pela chave única da publicação e sumariados por meio de planilhas para geração de gráficos e apresentação posterior de modo intuitivo dos resultados obtidos.

5.1.8.2 Síntese dos Dados

A síntese dos dados é realizada por meio de uma narrativa descritiva contendo os principais resultados encontrados ao longo do processo de revisão sistemática. Para todos os dados apresentados são fornecidas evidências quantitativas complementares, no formato de gráficos e tabelas.

5.1.9 Limitações do Estudo

As plataformas de buscas definidas para a coleta do material representam um aspecto no qual o autor do trabalho possui pouco controle e influência devido ao mecanismo de retorno automático dos resultados pelos buscadores. Isto pode impactar negativamente sobre a possibilidade de estudos futuros replicarem os resultados obtidos por meio dos termos de consulta definidos, uma vez que é bem provável que os resultados obtidos futuramente não serão idênticos aos obtidos durante a época de realização deste estudo.

5.2 Execução

Esta seção descreve o processo de execução do protocolo de revisão sistemática formulado, incluindo aspectos tais como: coleta das publicações, avaliação da relevância dos estudos e extração de dados dos estudos relevantes.

5.2.1 Coleta das Publicações

A coleta das publicações foi conduzida segundo os critérios definidos na subseção 5.1.7.3 e resultou na identificação de vinte e seis publicações. Após a coleta, foi iniciada uma categorização preliminar das publicações para definição do nível de relevância de cada em relação ao estudo desenvolvido neste trabalho. Para isso, foram utilizados marcadores coloridos com o intuito de facilitar o gerenciamento do material e extração de dados. O resultado deste processo pode ser observado na figura 8.

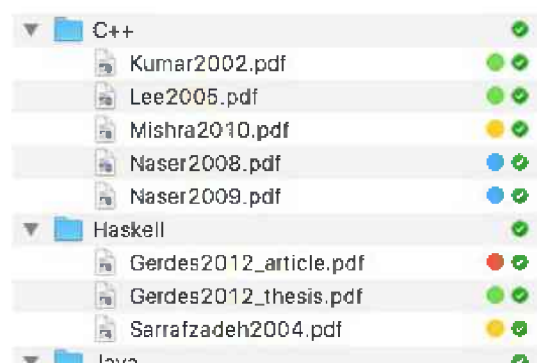


Figura 8 – Representação das publicações categorizadas

A quantidade de publicações por categoria obtida após o processo de categorização preliminar é apresentada na figura 9. Como pode ser constatado na figura, a soma das categorias altamente relevantes (verde), altamente relevantes e complementares (azul) e relevantes (amarelo) resulta em um total de vinte e três publicações. Contudo, como existem três publicações complementares pertencentes a um mesmo estudo de um determinado autor, o número total de estudos a ser analisado é de vinte estudos.

É importante destacar que, as publicações desclassificadas (vermelho) não fazem parte das etapas seguintes de análise de relevância das publicações e extração de dados. Por esta razão, estas publicações não serão mencionadas novamente.

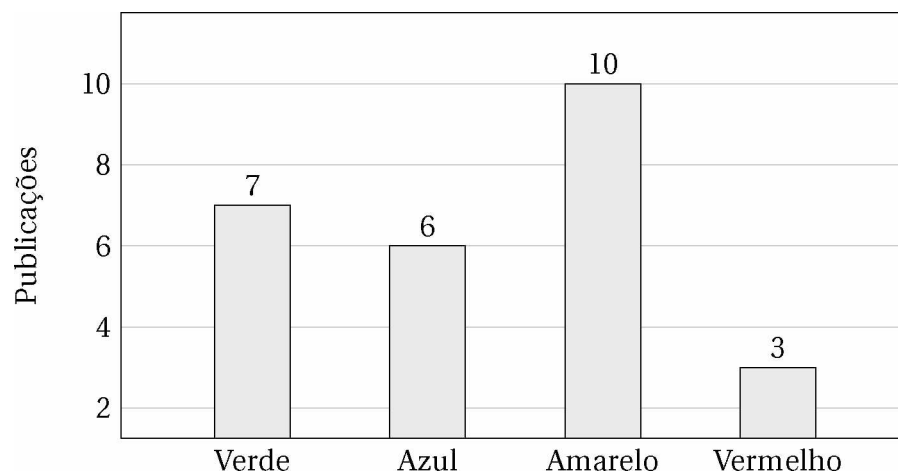


Figura 9 – Quantidade de publicações por categoria

A lista completa de publicações relevantes é: (SONG et al., 1997), (EL-KHOULY; FAR; KOONO, 2000), (KUMAR, 2002), (SHAH; KUMAR, 2002), (LANE; VANLEHN, 2004), (XU; SARRAFZADEH, 2004), (LEE; BABA, 2005), (SHAFFER, 2005), (BUTZ; HUA; MAGUIRE, 2006), [(EMURIAN et al., 2000) + (EMURIAN, 2006)], [(NASER, 2008) + (NASER, 2009)], (HOLLAND, 2009), (MISHRA; MISHRA, 2010), (SYKES, 2010), (ABU-NASER et al., 2011), (GERDES, 2012), (VESIN et al., 2013), [(WERAGAMA, 2013) + (WERAGAMA; REYE, 2014)], (ZATARAIN-CABADA et al., 2015), (HOOSHYAR et al., 2016). Os estudos complementares estão destacados entre colchetes.

5.2.2 Avaliação da Relevância dos Estudos

A avaliação da relevância dos estudos é obtida por meio da aplicação do formulário de avaliação do material coletado. O formulário de avaliação é encontrado no apêndice A. Por meio da concretização desse formulário é possível identificar os pontos fortes e fracos dos estudos selecionados, bem como verificar a relevância dos estudos para a concretização dos objetivos deste trabalho. Uma análise quantitativa dos resultados é realizada para as seguintes categorias: arquitetura, inferência de dados, feedback, protótipo, experimento e resultados.

5.2.2.1 Quadro de Pontuação Geral

Publicação	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total
Song1997	10	10	0	0	0	0	0	0	10	10	40
El2000	10	5	5	0	0	0	0	0	10	0	30
Kumar2002	5	10	10	0	0	10	10	10	10	5	70
Shah2002	0	0	10	10	0	5	10	5	10	10	60
Lane2004	5	5	5	10	0	0	0	0	10	10	45
Sarrafzadeh2004	10	10	10	10	0	0	0	0	10	5	55
Lee2005	10	10	5	10	0	10	10	5	10	10	80
Shaffer2005	0	0	5	0	0	10	10	10	10	10	55
Butz2006	10	10	10	10	5	0	0	0	10	10	65
Emurian2000 / Emurian2006	5	5	10	10	0	10	10	10	10	10	80
Naser2008 / Naser2009	10	10	5	10	0	10	10	10	10	10	85
Holland2009	10	10	10	10	5	10	10	10	10	10	95
Mishra2010	0	10	5	0	0	5	5	5	5	10	45
Sykes2010	10	10	10	10	5	10	10	10	10	10	95
Naser2011	10	0	5	10	0	5	5	0	10	0	45
Gerdes2012	10	10	10	10	5	10	5	5	10	10	85
Vesin2013	10	10	10	0	0	0	0	0	10	5	45
Weragama2013 / Weragama2014	10	10	10	10	5	10	10	10	10	10	95
Zatarain2015	10	10	5	10	0	5	0	0	10	10	60
Hooshyar2016	10	10	10	10	5	10	10	10	10	10	95

Tabela 2 – Tabela de avaliação da relevância das publicações

O resultado geral obtido após o processo de análise da relevância dos estudos é apresentado na tabela 2 e segue os critérios de classificação estipulados a seguir. Para cada questão presente no formulário de avaliação, as respostas ‘SIM’ representam 10 pontos e estão destacadas em verde, as respostas ‘PARCIAL’ representam 5 pontos e estão destacadas em amarelo, e as respostas ‘NÃO’ não pontuam e estão destacadas em vermelho.

A escala de cores também é utilizada para indicar a relevância geral de cada estudo após a soma total da pontuação obtida nas questões. Os estudos altamente relevantes estão destacados em verde e apresentam uma pontuação igual ou superior a 70 pontos. Os estudos relevantes estão destacados em amarelo e apresentam uma pontuação superior a 30 pontos e inferior a 70 pontos. Os estudos fracamente relevantes estão destacados em vermelho e apresentam uma pontuação inferior a 30 pontos.

Como pode ser evidenciado pela resultado geral, a quantidade de estudos altamente relevantes representa 45% do material selecionado, enquanto que a quantidade relevante detém os outros 55%. Este resultado é considerado extremamente satisfatório pelo autor deste trabalho, uma vez que indica a eficácia do processo de coleta.

5.2.2.2 Arquitetura

O primeiro critério analisado é a existência de uma arquitetura bem definida no estudo em análise. Caso exista, a arquitetura é documentada de forma clara e objetiva incluindo as responsabilidades de cada módulo e a interação entre eles? A questão I do formulário de avaliação tem como objetivo atribuir um peso maior aos estudos que descrevem a arquitetura do sistema em detalhes. Além disso, esta questão tem o objetivo secundário de identificar as arquiteturas interpretadas como mais apropriadas para a construção de sistemas tutores inteligentes no domínio de programação em dado contexto.

Uma arquitetura somente é apontada como bem definida caso o estudo documente todos os módulos do sistema, suas responsabilidades, e a interação existente entre eles. Uma arquitetura é apontada como parcialmente definida caso apenas alguns dos critérios anteriores sejam respondidos. Caso nenhum dos critérios sejam respondidos ou a arquitetura do sistema não esteja documentada, a arquitetura será apontada como não definida.

Do total de vinte estudos analisados, quatorze apresentam uma arquitetura bem definida, três apresentam uma arquitetura parcial e três não descrevem a arquitetura do sistema proposto. A proporção é considerada altamente satisfatória pelo autor deste trabalho visto que há um grau de possibilidade muito alto que o objetivo geral deste trabalho será alcançado após a análise detalhada de cada uma das arquiteturas. O resultado da questão 01 é ilustrado na figura 10.

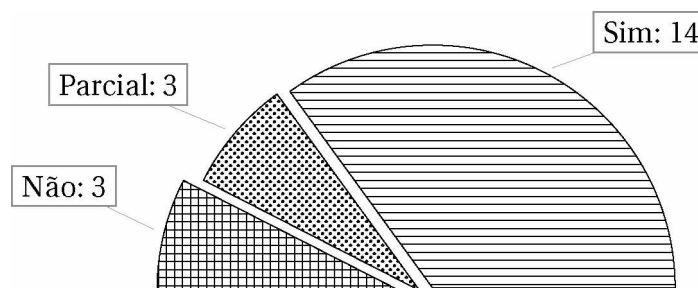


Figura 10 – Análise quantitativa da questão I do formulário de avaliação

5.2.2.3 Inferência dos dados

O segundo critério averiguado é o método de inferência de dados do sistema descrito. É apresentado o modo no qual o sistema irá tomar decisões? Caso apresentado, a técnica escolhida é justificada? A questão II do formulário de avaliação tem como objetivo atribuir um peso maior aos estudos que descrevem a técnica de inferência utilizada. Além disso, esta questão tem o objetivo secundário de identificar a técnica mais apropriada para o raciocínio em sistemas tutores inteligentes no domínio de programação, assim como vantagens e desvantagens de cada uma das técnicas.

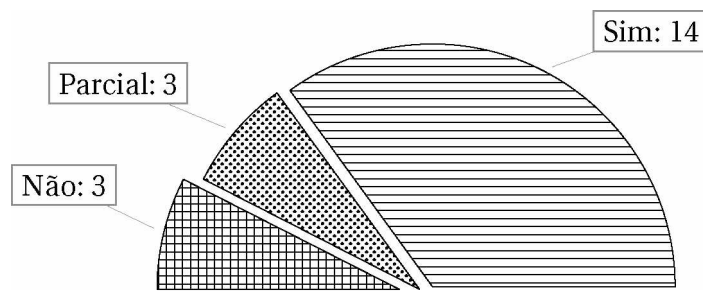


Figura 11 – Análise quantitativa da questão II do formulário de avaliação

O mecanismo de inferência será apontado como claramente descrito caso o estudo descreva o funcionamento da técnica de inferência adotada para o sistema e discuta o porquê de sua escolha perante outras técnicas existentes. O mecanismo de inferência será apontado como parcialmente descrito caso apenas alguns dos critérios anteriores sejam correspondidos. Caso o estudo não apresente a técnica utilizada ou não descreva nenhum dos critérios anteriores, o mecanismo de inferência será apontado como não descrito.

Do total de vinte estudos analisados, quatorze apresentam a técnica de inferência utilizada de forma detalhada, três apresentam a técnica utilizada de forma parcial e três não descrevem a técnica utilizada no sistema. A proporção é considerada bastante satisfatória pelo autor deste trabalho. O resultado da questão 02 é ilustrado na figura 11.

5.2.2.4 Retorno de Informações

Um dos aspectos mais importantes do processo de aprendizagem em um sistema tutor inteligente é o retorno de informações ao usuário em relação às suas ações e desempenho no sistema. É apresentado o modo no qual o usuário receberá retorno do sistema? É discutido a forma de retorno mais apropriada para o usuário do sistema desenvolvido levando em consideração os seus objetivos pedagógicos? A questão III do formulário de avaliação tem como objetivo atribuir um peso maior aos estudos que descrevem o funcionamento do processo de retorno ao usuário. Além disso, esta questão tem o objetivo secundário de identificar os aspectos mais importantes no retorno de informações de um usuário de um sistema tutor no domínio de programação.

O processo de feedback do usuário será apontado como claramente descrito caso o estudo apresente exemplos de interação com o usuário e discorra sobre o impacto do feedback em sua aprendizagem. O processo de feedback é apontado como parcialmente apresentado caso apenas alguns dos critérios anteriores sejam respondidos. Caso o estudo não apresente detalhes a respeito do processo de feedback do usuário ou não considere nenhum dos critérios anteriores, o feedback do usuário será apontado como não definido.

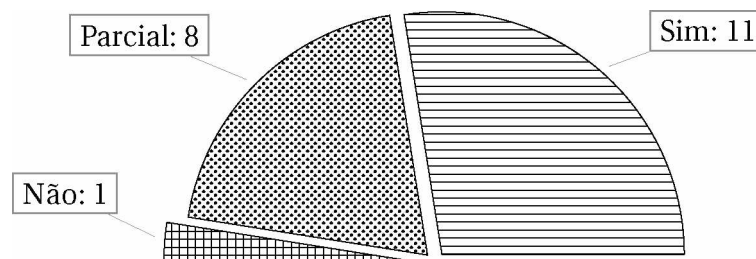


Figura 12 – Análise quantitativa da questão III do formulário de avaliação

Do total de vinte estudos analisados, onze apresentam em detalhes o processo de feedback utilizado no sistema, oito apresentam de forma parcial e apenas um não descreve. Apesar de esperada uma quantidade maior de estudos que descrevessem o processo de retorno de informações do usuário em detalhes, a proporção ainda é considerada satisfatória pelo autor do trabalho. O resultado da questão 03 é ilustrado na figura 12.

5.2.2.5 Protótipo

Um dos aspectos importantes à fase de entendimento de um sistema é construir um protótipo que ilustre e valide o conceito do projeto. Esta etapa é muito importante para avaliar o conceito elaborado e identificar aspectos do sistema a serem aprimorados. O estudo apresentado possui um protótipo? O protótipo, caso presente, é funcional e descrito em detalhes? A questão IV do formulário de avaliação tem como objetivo verificar quais dos sistemas apresentam um protótipo no qual é possível evidenciar o modo de funcionamento do sistema. Além disso, tem o objetivo secundário de identificar quais as ferramentas comumente utilizadas para a construção de protótipos.

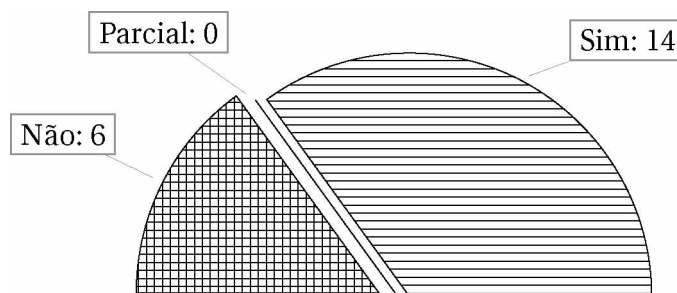


Figura 13 – Análise quantitativa da questão IV do formulário de avaliação

O protótipo é apontado como apresentado caso o estudo em análise documente a existência de um protótipo do sistema através de ilustrações e textos. O protótipo é apontado como parcialmente definido caso os critérios anteriores sejam parcialmente atendidos ou não seja possível evidenciar de forma clara o funcionamento do sistema através do protó-

tipo. Caso o estudo não apresente um protótipo do sistema, o protótipo será apontado como não apresentado.

Do total de vinte estudos analisados, quatorze apresentam um protótipo bem definido e seis não apresentam um protótipo. A proporção é considerada altamente satisfatória pelo autor deste trabalho dado que grande parte dos estudos ilustraram a arquitetura e o funcionamento do sistema proposto através de um protótipo, alguns inclusive funcionais. O resultado da questão 04 é ilustrado na figura 13.

Outro aspecto importante de um sistema tutor é oferecer suporte à personalização das preferências do sistema pelo usuário. A questão V do formulário de avaliação visa complementar a questão IV e atribuir um peso maior aos estudos que apresentam um protótipo com suporte à personalização de preferências pelo usuário. Afinal, é considerado de suma importância que o usuário tenha impacto sobre sua navegação no sistema e possa exercer influência em sua aprendizagem de acordo com os seus objetivos e metas pessoais.

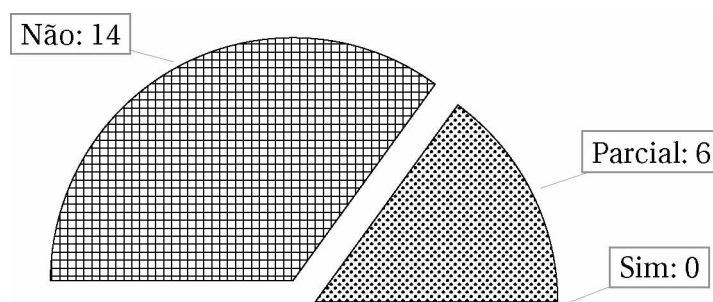


Figura 14 – Análise quantitativa da questão V do formulário de avaliação

O suporte à personalização de preferências de uso pelo usuário é apontado como presente caso o estudo documente no protótipo a possibilidade de o usuário alterar as configurações de ensino de acordo com seus objetivos e metas, e tenha impacto sobre a navegação durante sua aprendizagem. O suporte é apontado como parcialmente existente caso apenas um dos critérios anteriores seja atendido. Caso o estudo não suporte nenhum tipo de personalização pelo usuário, o suporte à personalização é apontado como inexistente.

Do total de vinte estudos analisados, nenhum apresenta suporte completo à personalização de preferências de uso pelo usuário, seis apresentam suporte parcial e quatorze não apresentam nenhum tipo de suporte. A proporção é considerada insatisfatória pelo autor deste trabalho dado que o suporte à personalização é uma das características ideais em um sistema tutor. Apesar da complexidade de desenvolvimento desse requisito, era esperado que um número maior de estudos incluísse o suporte durante o projeto do sistema. O resultado da questão 05 é ilustrado na figura 14.

5.2.2.6 Experimento

Após ou durante a fase de prototipação é importante avaliar a usabilidade do sistema através da condução de um ou mais experimentos com no mínimo um grupo de controle, especialmente em um processo de design centrado no usuário como é o caso de um sistema tutor inteligente no domínio de programação. A questão VI do formulário de avaliação visa atribuir um peso maior aos estudos que apresentam um experimento.

Este processo de participação é fundamental para compreender a percepção dos usuários durante a interação com o sistema e verificar se as expectativas e objetivos estão sendo atendidos. É importante também que os critérios de escolha dos grupos sejam transparentes e que o experimento apresente uma quantidade satisfatória de indivíduos envolvidos para que os resultados não sejam equivocados.

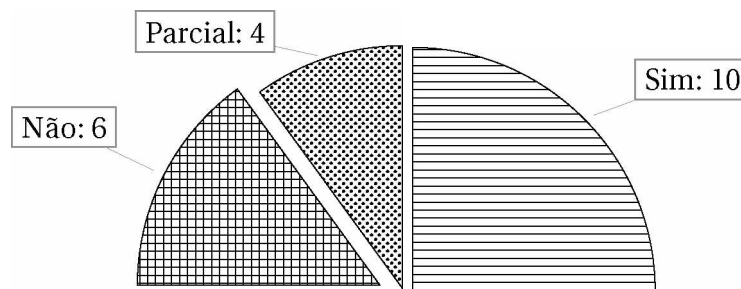


Figura 15 – Análise quantitativa da questão VI do formulário de avaliação

Um experimento é apontado como bem realizado caso o estudo documente em detalhes o modo no qual o experimento foi conduzido, incluindo os critérios de escolha do grupo de controle e quantidade de pessoas selecionadas. O experimento é apontado como parcialmente realizado caso apenas alguns dos critérios anteriores sejam atendidos. Caso o estudo não apresente um experimento ou nenhum dos critérios anteriores sejam atendidos no experimento realizado, o experimento é apontado como inexistente.

Do total de vinte estudos analisados, dez apresentam um experimento bem descrito, quatro apresentam um experimento considerado parcial pelo autor deste trabalho e seis não realizam um experimento. A proporção é considerada satisfatória pelo autor deste trabalho dado que se observa que grande parte dos estudos buscaram uma validação do usuário em relação à utilidade e usabilidade do sistema. O resultado da questão 06 é ilustrado na figura 15.

Ao conduzir um experimento é essencial que o processo de coleta dos dados seja claramente especificado e que os métodos de análise sejam atendidos. A questão VII do formulário de avaliação tem como objetivo atribuir um peso maior aos estudos que realizam a coleta dos dados de forma satisfatória.

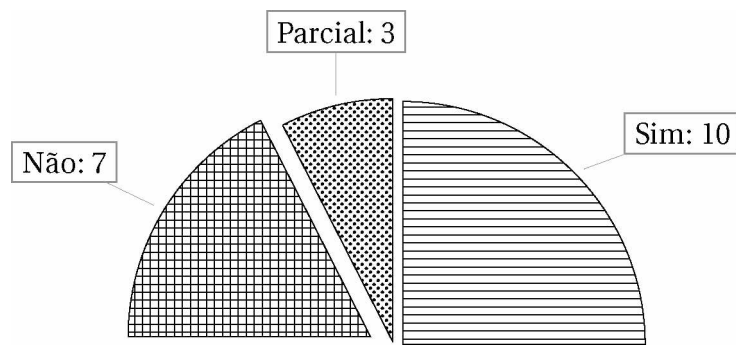


Figura 16 – Análise quantitativa da questão VII do formulário de avaliação

A coleta é considerada satisfatória caso o estudo documente em detalhes o processo no qual os dados foram coletados, bem como todas as unidades de medida e métodos de análise utilizados. A coleta é considerada parcialmente satisfatória caso apenas alguns dos critérios anteriores sejam atendidos. Caso o estudo não apresente um experimento ou nenhum dos critérios anteriores sejam atendidos, o processo de coleta dos dados do experimento é apontado como inexistente ou fracamente descrito.

Do total de vinte estudos analisados, dez descrevem em detalhes o processo de coleta, três descrevem parcialmente e sete não descrevem ou não realizaram um experimento. Proporção considerada altamente satisfatória pelo autor do trabalho visto que a grande maioria dos estudos que realizaram um experimento descrevem de forma transparente o processo de coleta dos dados. O resultado da questão 07 é ilustrado na figura 16.

Complementando o processo de realização de um experimento, é essencial que o processo de análise dos dados coletados seja rigoroso e imparcial. O raciocínio de obtenção dos resultados deve ser devidamente documentado e comprovar a realidade dos fatos apresentados. A questão VIII do formulário de avaliação visa atribuir um peso maior aos estudos que analisam os dados coletados de forma satisfatória.

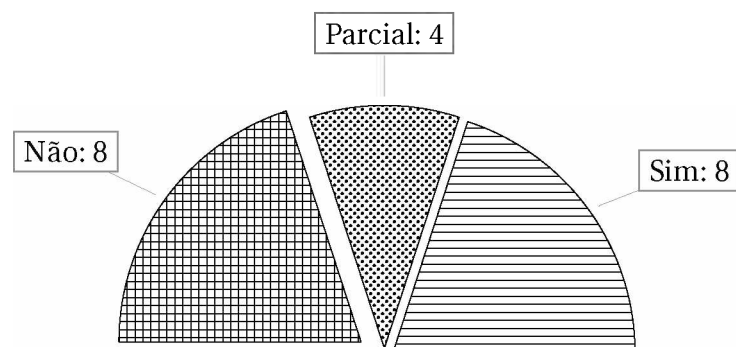


Figura 17 – Análise quantitativa da questão VIII do formulário de avaliação

A análise de dados é considerada satisfatória caso o estudo documente de forma criteriosa e imparcial o processo de análise realizado. A análise é considerada parcialmente satisfatória caso o estudo documente a análise conduzida de modo vago sem detalhar os passos realizados para obtenção de determinado resultado. Caso o estudo não apresente um experimento ou nenhum dos critérios anteriores sejam atendidos, o processo de análise dos resultados do experimento é apontado como inexistente.

Do total de vinte estudos analisados, oito descrevem a análise de forma detalhada e rigorosa, quatro descrevem parcialmente e oito não descrevem ou não realizam um experimento. A proporção é considerada altamente satisfatória pelo autor do trabalho visto que a grande maioria dos experimentos apresenta uma análise rigorosa. O resultado da questão 08 é ilustrado na figura 17.

5.2.2.7 Resultados

É fundamental que um estudo científico documente os resultados obtidos para identificar se os objetivos do pesquisa foram devidamente alcançados após o seu desenvolvimento. A questão IX do formulário de avaliação visa atribuir um peso maior aos estudos que apresentam os resultados obtidos de forma clara e objetiva.

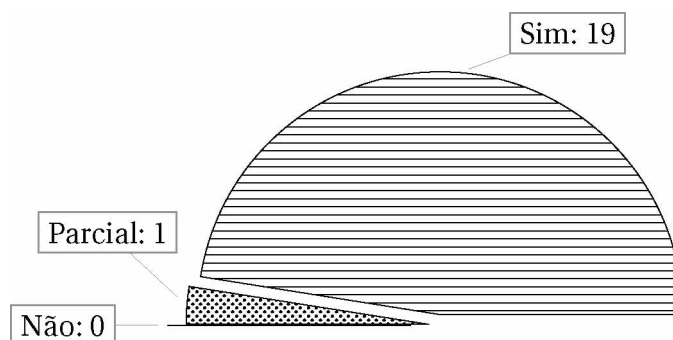


Figura 18 – Análise quantitativa da questão IX do formulário de avaliação

Os resultados obtidos são considerados claramente apresentados caso o estudo apresente uma argumentação sobre as conclusões obtidas fundamentada nos dados apresentados ao longo do estudo. Os resultados são considerados parcialmente apresentados caso os resultados obtidos não sejam documentados de acordo com os critérios anteriores. Caso os resultados obtidos com a realização do estudo não sejam percorridos ou não considerem os critérios anteriores, os resultados obtidos serão apontados como inexistentes.

Do total de vinte estudos analisados, dezenove apresentam os resultados obtidos com a realização do trabalho de forma clara e apenas um apresenta os resultados de forma parcial. A proporção, apesar de prevista, é considerada altamente satisfatória pelo autor do trabalho visto que todos apresentam os resultados obtidos. O resultado da questão 09 é ilustrado na figura 18.

Outro aspecto fundamental ao concluir uma pesquisa é discutir a contribuição do estudo para a área na qual o estudo pertence. A questão X tem como objetivo atribuir um peso maior aos estudos que apresentam uma discussão sobre a contribuição do estudo para a área estudada.

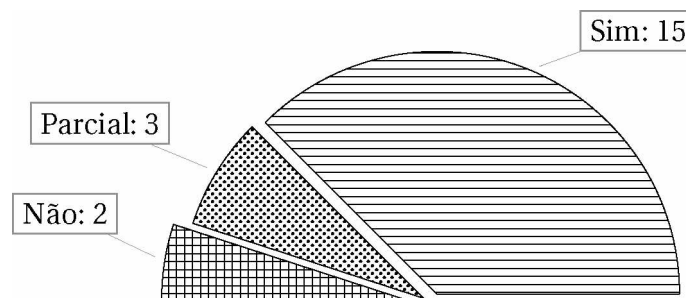


Figura 19 – Análise quantitativa da questão X do formulário de avaliação

A contribuição do estudo é considerada claramente apresentada caso a argumentação inclua a contribuição do estudo para o conhecimento já existente na área e identifique possíveis áreas onde os resultados possam contribuir. A contribuição é considerada parcialmente discutida caso apenas alguns dos critérios anteriores sejam atendidos. Caso a contribuição não seja discutida ou não considere os critérios anteriores, a discussão será apontada como inexistente.

Do total de vinte estudos analisados, quinze apresentam uma discussão detalhada sobre a contribuição do estudo, três apresentam uma discussão parcial e dois não apresentam nenhuma discussão a respeito. Proporção considerada altamente satisfatória pelo autor do trabalho visto que a grande maioria dos estudos analisados discutem a contribuição do estudo para a área após o término da pesquisa. O resultado da questão 10 é ilustrado na figura 19.

5.2.3 Análise dos dados

O processo de análise dos dados relevantes é detalhado ao longo desta subseção, incluindo características sobre as arquiteturas utilizadas e protótipos construídos. As conclusões obtidas por meio desses dados são apresentadas subsequentemente, contendo a arquitetura observada como inicial para um STI orientado ao ensino de programação.

5.2.3.1 Arquitetura

Como apontado na subseção 5.2.2.2, quatorze estudos apresentam uma arquitetura considerada bem definida, enquanto que três apresentam uma arquitetura considerada parcial. Esta subseção tem como objetivo analisar detalhadamente as arquiteturas relevantes e identificar aquelas mais apropriadas para o desenvolvimento de um sistema tutor inte-

ligente no domínio de programação. Apenas os estudos que apresentam uma arquitetura considerada bem descrita ou parcial foram levados em consideração. Além disso, nos casos em que a arquitetura é considerada parcial há uma justificativa apontando as razões pela qual obteve essa avaliação.

5.2.3.1.1 C-Tutor

([SONG et al., 1997](#)) descreve o C-Tutor, um sistema tutor voltado ao ensino de programação na linguagem C, a arquitetura do sistema é composta por dois subsistemas: um analisador de programa e um ambiente de aprendizagem. O analisador de programa é composto por um sistema de engenharia reversa e um sistema didático, enquanto que o ambiente de aprendizagem é constituído por uma rede curricular. As responsabilidades dos dois subsistemas são devidamente documentadas e evidenciam de forma clara o funcionamento do sistema. Na figura 20 é apresentada uma visão geral da arquitetura do C-Tutor.

O sistema de engenharia reversa do analisador de programa, chamado GOES, é responsável pela geração de problemas através de programas de exemplos fornecidos pelo tutor humano. A geração da descrição de cada problema ocorre através da extração de planos e objetivos dos exemplos fornecidos pelo tutor.

O sistema didático, chamado ExBug, analisa as respostas dos estudantes. A análise é um processo de três etapas. A primeira etapa é a execução do programa submetido pelo estudante. A segunda etapa é a análise da corretude do programa através da comparação do programa com os planos de programação relevantes presentes na base de conhecimento. A terceira etapa identifica os erros cometidos e apresenta uma lista contendo todos os erros identificados ao estudante.

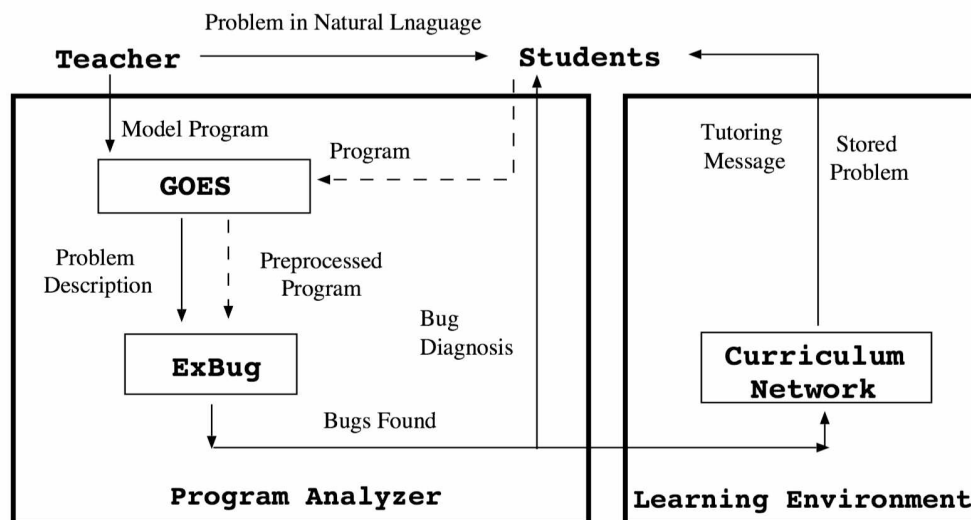


Figura 20 – Arquitetura do C-Tutor apresentada em ([SONG et al., 1997](#))

5.2.3.1.2 E-TCL

(EL-KHOULY; FAR; KOONO, 2000) apresenta o E-TCL, um sistema tutor web para o ensino de programação, a arquitetura do sistema é composta por três agentes: agente de tutoria, agente assistente pessoal para professores e agente assistente pessoal para estudantes. As responsabilidades dos agentes são devidamente documentadas e evidenciam o funcionamento por meio de comunicação cliente-servidor do sistema. É possível notar também que os módulos da arquitetura clássica podem ser identificados como subcomponentes dos agentes especificados. Uma visão geral da arquitetura do E-TCL é apresentada na figura 21.

O agente assistente pessoal para professores é composto por três partes: o módulo especialista, a base de regras semânticas e a base de conteúdo de tutoria. A principal função deste agente consiste na produção de regras semânticas e manipulação da base de conhecimento de uma linguagem de programação de modo que permita a fácil adição ou modificação das estruturas de comandos ensinadas pelos professores.

O agente assistente pessoal para estudantes é composto por três componentes: o modelo do estudante, o módulo de tutoria e o módulo de interface de usuário. Este agente é responsável pela interação do sistema com o estudante e identificação ou atualização de seu nível de conhecimento ao longo do processo de tutoria.

O agente de tutoria contém módulos de tutoria, base de dados com diálogos anteriores e base de regras semânticas. Este agente realiza o sequenciamento de conteúdo e resolução interativa de problemas enviados pelo estudante.

5.2.3.1.3 Sistema Tutor em C++

Um sistema tutor para o ensino de programação na linguagem C++ por meio da análise e depuração de segmentos de código é descrito por (KUMAR, 2002). O estudo é focado apenas na modelagem do domínio do sistema através do raciocínio baseado em modelo e, por essa razão, apresenta uma arquitetura parcial que documenta somente o processo de geração de problemas. Por meio do fragmento apresentado é possível evidenciar de forma clara as responsabilidades e interação dos módulos envolvidos no processo. O mecanismo adotado para geração de problemas do sistema é apresentado na figura 22.

Um número ilimitado de problemas pode ser gerado através do esquema ilustrado, uma vez que o raciocínio baseado em modelos é capaz de resolver problemas por conta própria sem conhecimento prévio da solução correta. Um tutor que utilize desse modelo não fica restrito apenas aos problemas que foram codificados nele e pode se tornar muito mais eficiente ao processo de ensino, pois os testes e avaliações podem se tornar individualizados e evitar o plágio.

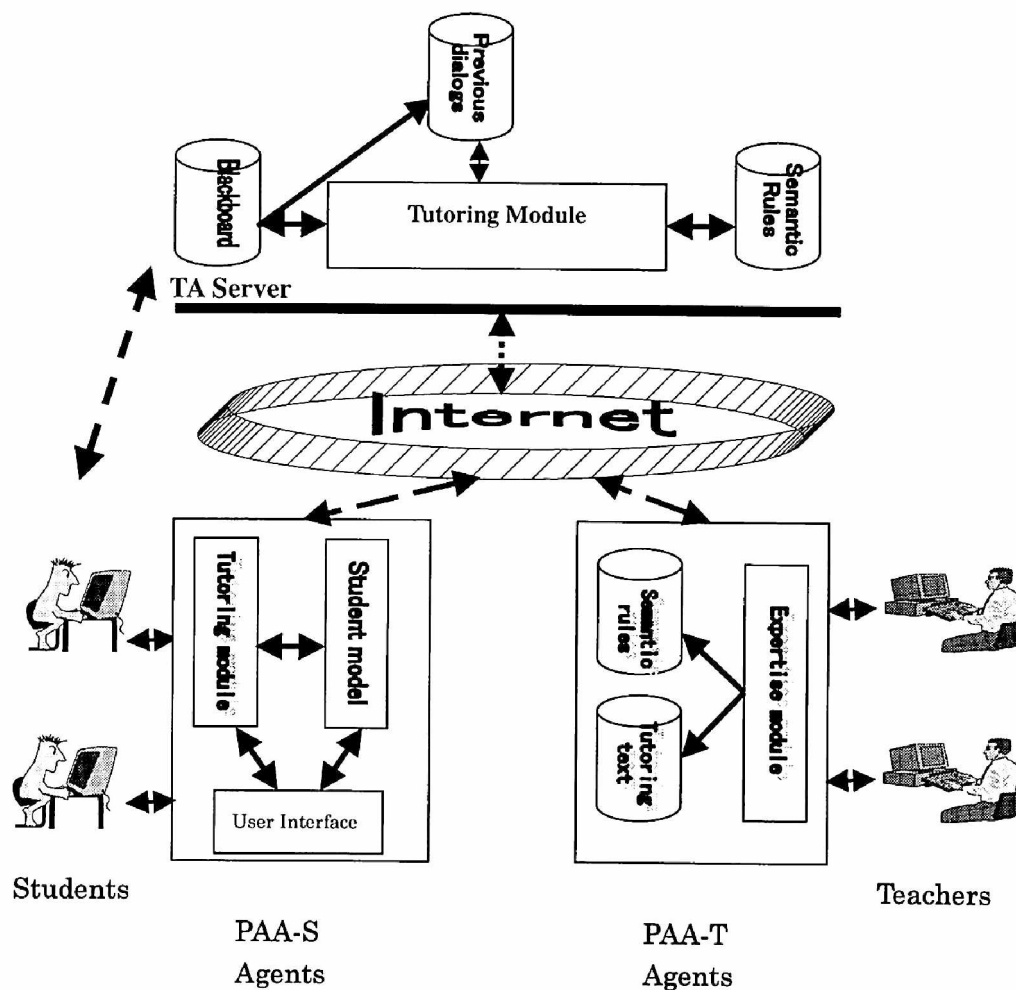


Figura 21 – Arquitetura do E-TCL apresentada em (EL-KHOULY; FAR; KOONO, 2000)

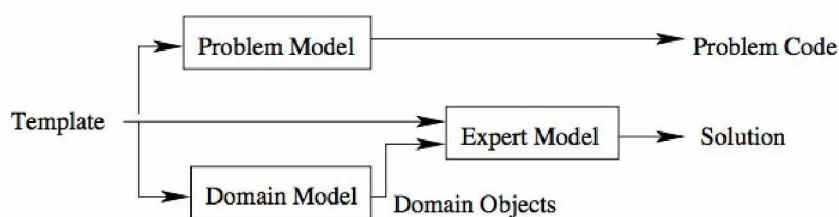


Figura 22 – Arquitetura de geração de problemas apresentada em (KUMAR, 2002)

5.2.3.1.4 ProPI

(LANE; VANLEHN, 2004) detalha o ProPI, um sistema tutor web baseado em diálogos para o ensino de programação, a arquitetura do sistema não é claramente documentada no estudo e por essa razão é considerada parcial pelo autor deste trabalho. Através da leitura do estudo, é possível evidenciar o papel da interface, do planejador de diálogos e do construtor de diálogos, contudo, o estudo carece de uma representação esquemática do processo de comunicação web realizado durante a tutoria.

O construtor de diálogos é a principal fonte de conhecimento do sistema e é representado por linhas de raciocínio tutorial. Estas linhas consistem em seqüências de objetivos de tutoria, cada uma realizada em forma de perguntas e conjuntos de respostas esperadas à essas perguntas. O planejador de diálogos é baseado em um quadro de trabalho chamado Atlas, utilizado na construção de sistemas de diálogos baseados em linguagem natural. Através dessa biblioteca é possível definir todas as interações desejadas com o estudante e planos de ações a serem executados. A interface realiza a interação com o estudante por meio de janelas em forma de diálogos.

5.2.3.1.5 Haskell-Tutor

(XU; SARRAFZADEH, 2004) descreve o Haskell-Tutor, um sistema tutor baseado em regras para o ensino de programação na linguagem Haskell, a arquitetura do sistema é composta por sete módulos: o módulo da interface, o módulo pedagógico, o módulo do estudante, o módulo do domínio, um módulo de explicações, um módulo de problemas e um módulo especialista. As responsabilidades e interações dos módulos são devidamente documentadas e fornecem uma visão clara do funcionamento do sistema. Na figura 23 são apresentados os módulos e relacionamentos da arquitetura do Haskell-Tutor.

O módulo do domínio contém a base de conhecimento do sistema representada em forma de restrições. O módulo pedagógico oferece retorno individualizado de informações de aprendizado ao estudante. O módulo do estudante lida com o nível de conhecimento do estudante e atualiza o nível com base nas soluções do estudante. O módulo de explicações fornece explicações detalhadas das razões pelas quais uma solução é considerada errada, bem como dicas para auxiliar o estudante na correção da solução. O módulo de problema contém todos os problemas disponíveis ao estudante durante o processo de ensino. O módulo especialista gera soluções corretas aos problemas apresentados ao estudante. O módulo da interface controla a interação do sistema com o estudante.

Assim que uma sessão de ensino é iniciada, um problema é selecionado seja pelo estudante ou pelo módulo pedagógico e o sistema entra em estado de espera do envio da solução do estudante. Após o envio da solução do estudante, o módulo da interface transmite a solução para o módulo do estudante para análise das restrições violadas pelo estudante e atualização de seu modelo. O módulo da interface também transmite a solução para o módulo especialista, responsável pela geração de uma solução correta usando as restrições do módulo do domínio e envio da resposta ao módulo de explicações. O módulo pedagógico gera, então, uma ação pedagógica baseada no modelo do estudante. Por fim, o retorno do módulo de explicações é apresentado ao estudante pelo módulo da interface.

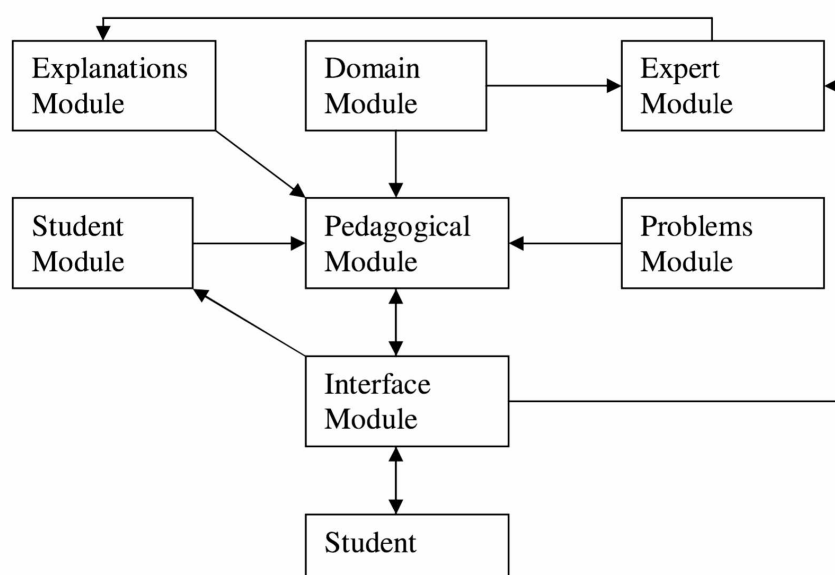


Figura 23 – Arquitetura do Haskell-Tutor apresentada em (XU; SARRAFZADEH, 2004)

5.2.3.1.6 C++ STL ITS

De acordo com a descrição de (LEE; BABA, 2005), o C++ STL ITS é um sistema tutor para o ensino de programação na linguagem C++ construído com base no framework STL, uma coleção de estruturas de dados e algoritmos. A arquitetura do sistema é baseada na arquitetura clássica e é composta por quatro módulos: módulo do domínio, módulo de estratégias de ensino, módulo do estudante e módulo da interface gráfica do usuário. As responsabilidades e interações dos módulos são devidamente documentadas e fornecem uma visão clara do funcionamento do sistema. A arquitetura completa dos módulos e relacionamentos do C++ STL ITS é apresentada na figura 24. A arquitetura web de três camadas adotada para o desenvolvimento do sistema é apresentada, em seguida, na figura 25.

O módulo do estudante é responsável pela geração e armazenamento do modelo do estudante com base no seu desempenho e informações obtidas durante suas interações com o sistema. O módulo de estratégias de ensino é composto por três componentes principais: pré-teste, tutoriais e pós-teste. O pré-teste realiza uma avaliação preliminar do conhecimento do estudante nos pré-requisitos de um conceito e o direciona aos tutoriais adequados. Após o término dos tutoriais, um pós-teste é aplicado a fim de avaliar a eficácia da sessão de tutoria e conhecimento adquirido pelo estudante. O módulo do domínio contém a base de conhecimento do sistema e é dividido em três partes: tópicos e subtópicos, especificações de programa, e tutoriais e subtutoriais. Essa divisão facilita a manutenção da base e alteração de seu conteúdo. O módulo da interface gráfica lida com a interação do usuário com o sistema e transmissão das informações aos módulos responsáveis pelo processamento.

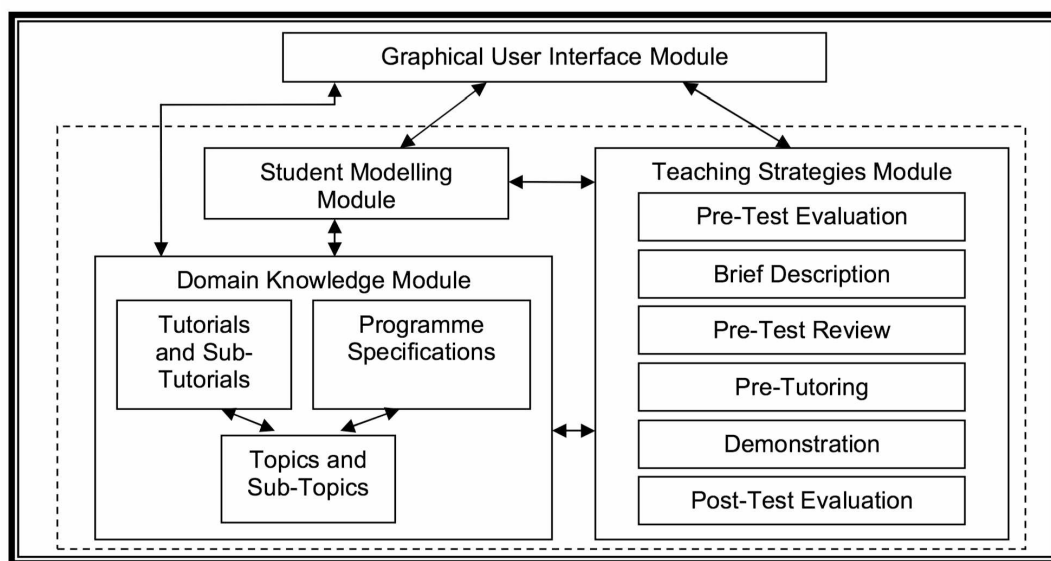


Figura 24 – Arquitetura do C++ STL ITS apresentada em (LEE; BABA, 2005)

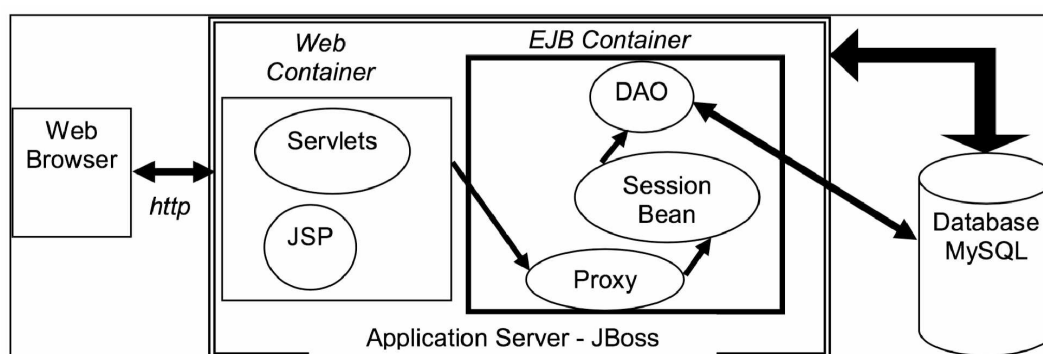


Figura 25 – Arquitetura de três camadas do C++ STL ITS apresentada em (LEE; BABA, 2005)

5.2.3.1.7 BITS

Conforme descrito em (BUTZ; HUA; MAGUIRE, 2006), o BITS é um sistema tutor web baseado em redes Bayesianas. A arquitetura do sistema é baseada na arquitetura clássica e é composta por quatro módulos: o módulo de redes Bayesianas, base do conhecimento, interface de usuário e módulo de orientação adaptativo. Além disso, são destacados dois submódulos no módulo de interface de usuário e três submódulos no módulo de orientação adaptativa que permitem uma análise precisa das responsabilidades dos módulos e funcionamento do sistema. A arquitetura do BITS é apresentada na figura 26.

O módulo de redes Bayesianas é responsável pela inferência de dados do sistema durante o processo de ensino e armazenamento do nível de conhecimento do estudante. Este módulo avalia as respostas e comportamento do estudante de modo constante a fim de detectar estratégias de ensino personalizadas mais adequadas aos objetivos de aprendizagem do estudante.

A base de conhecimento contém o material de ensino do sistema em formato de páginas web, como também um repositório de testes. Todo o material é organizado através de conceitos de conhecimento em C++ que correspondem aos nós da rede Bayesiana do sistema. Dessa forma, os conceitos de conhecimento são distinguidos do conteúdo instrucional, facilitando a alteração do conteúdo de tutoria e aplicação da técnica de inferência Bayesiana para a estimativa do nível de conhecimento do estudante.

O módulo de orientação adaptativo oferece três métodos de adaptação: suporte à navegação, recomendações de pré-requisitos para a solução de problemas e geração de sequências de aprendizado individuais para conceitos específicos. A principal função do módulo é oferecer opções de aprendizagem que melhor se adequam aos objetivos de aprendizagem e preferências do usuário.

O módulo de interface de usuário interage com o estudante através de um navegador web e utiliza janelas de diálogo para mostrar testes de avaliação e sugestões pedagógicas ao longo do processo de ensino.

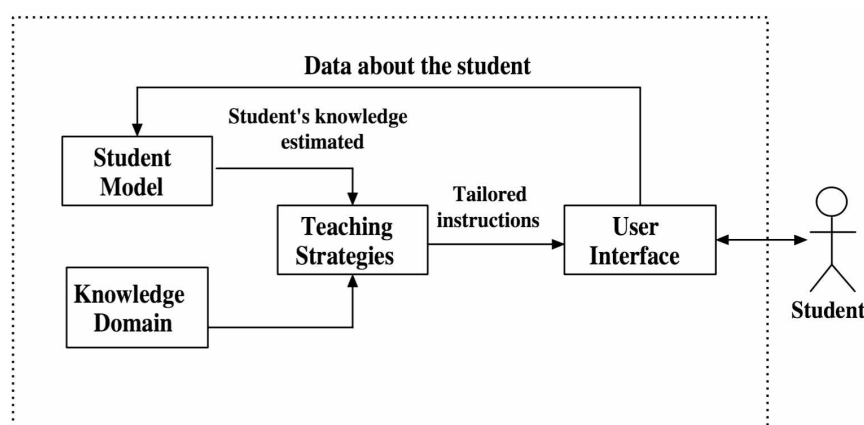


Figura 26 – Arquitetura do BITS apresentada em (BUTZ; HUA; MAGUIRE, 2006)

5.2.3.1.8 Sistema Tutor em applets Java

Um sistema tutor baseado em uma série de applets Java para o ensino de programação na linguagem Java é descrito em (EMURIAN et al., 2000) e (EMURIAN, 2006). Não há uma representação esquemática que ilustre a arquitetura do sistema no estudo, contudo, é possível induzir o modelo de funcionamento do sistema de applets Java através dos estágios de tutoria apresentados e tecnologias utilizadas para o desenvolvimento. A arquitetura foi considerada parcial em detrimento do critério anterior.

O primeiro estágio oferece ao estudante a oportunidade de observar itens individuais de código Java a fim de transmitir um conhecimento mínimo do repertório de símbolos utilizados na escrita de um programa. O segundo estágio permite que o estudante encontre um determinado item em uma lista de itens, aperfeiçoando sua capacidade de discernimento

de propriedades. O terceiro estágio ensina itens individuais de código e testa o aluno sobre o significado de cada item. O quarto estágio requer que o estudante escreva um código de 10 linhas e testa o aluno sobre o objetivo geral de cada linha. O quinto estágio solicita a escrita de um código dentro de uma janela que simula um editor de texto tradicional e testa a habilidade do aluno em escrever um programa básico por conta própria.

5.2.3.1.9 CPP-Tutor

Segundo (NASER, 2008) e (NASER, 2009), o CPP-Tutor é um sistema tutor para o ensino de programação na linguagem C++ cuja arquitetura é uma variação da arquitetura clássica e é composta por cinco módulos: base de conhecimento, módulo especialista, módulo especialista de reconhecimento de intenção, módulo de retorno de informações e interface de usuário. As responsabilidades dos módulos são claramente identificadas e detalham o funcionamento do sistema, incluindo um fluxograma do processo de tutoria. No entanto, o estudo não apresenta uma representação esquemática da arquitetura. O fluxograma do processo de tutoria é apresentado na figura 27.

A base de conhecimento contém um subconjunto do material em C++ a ser ensinado ao estudante. O sistema concentra nos seguintes tópicos: declaração de variáveis, operadores, estruturas de repetição e tarefas. O módulo especialista analisa cuidadosamente toda a informação disponível durante o processo de ensino para gerar o retorno de informações apropriado ao estudante. Entre o conjunto de informações analisadas estão: a declaração e especificação do programa, o código do estudante, o modelo do estudante atualizado e o resultado produzido pelo compilador C++. O módulo especialista de reconhecimento de intenção tem como objetivo verificar a intenção do código submetido pelo estudante. Este módulo realiza várias verificações de padrões nas respostas submetidas e sugere correções de código ao estudante quando erros são encontrados. O módulo de retorno de informações apresenta informações de desempenho ao estudante e dicas de correções. A interface do usuário interage com o estudante e transmite todas as informações enviadas do estudante para os módulos e vice-versa.

5.2.3.1.10 J-LATTE

Conforme (HOLLAND, 2009), o J-LATTE é um sistema tutor web para o ensino de programação na linguagem Java. A arquitetura do sistema é composta por vários módulos, sendo os principais: o módulo pedagógico, o módulo do estudante e o gerenciador de sessões. A comunicação realizada entre o web browser e o web server é claramente documentada, bem como o funcionamento do sistema durante a utilização por um usuário. A arquitetura do J-LATTE é apresentada na figura 28.

Toda a informação e interação é apresentada ao estudante por meio de uma interface web. O gerenciador de sessões lida com todas as requisições do servidor web e interage com

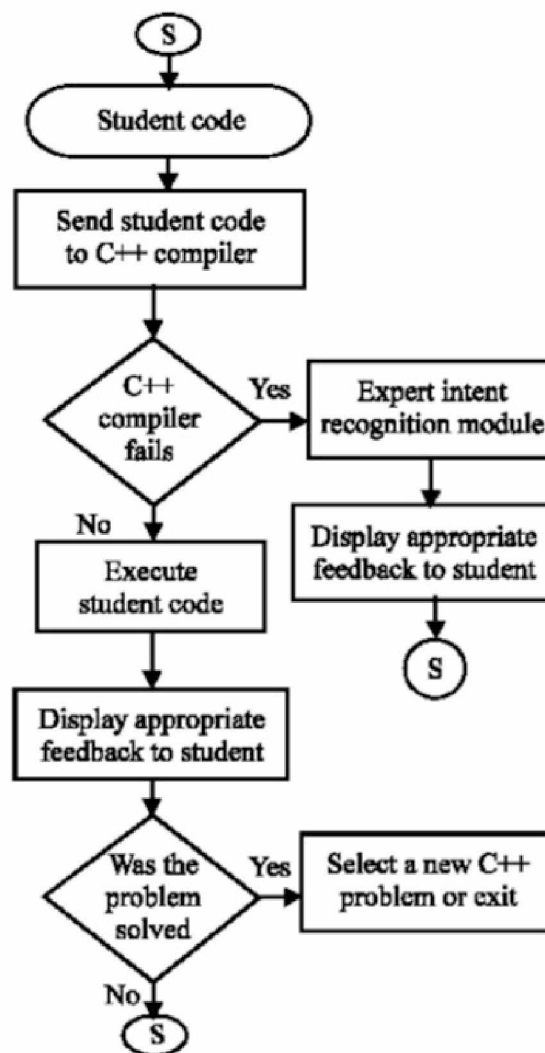


Figura 27 – Fluxograma do processo de tutoria apresentado em (NASER, 2008)

todas as outras partes do sistema. O módulo pedagógico realiza as correções de exercícios e toma as decisões pedagógicas. O módulo do estudante lida com o nível de conhecimento dos estudantes através da criação e atualização de um modelo para cada estudante, bem como avalia as soluções enviadas pelos estudantes.

Os dados de utilização de cada estudante são armazenados em uma base de dados auxiliar administrada pelo gerenciador de sessões, enquanto que os modelos dos estudantes são administrados pelo módulo do estudante. O repositório de problemas é acessado pelo módulo pedagógico durante a apresentação de novos problemas. A base de restrições é acessada tanto pelo módulo pedagógico quanto pelo módulo do estudante ao longo do processo de correção e avaliação das respostas submetidas pelo estudante.

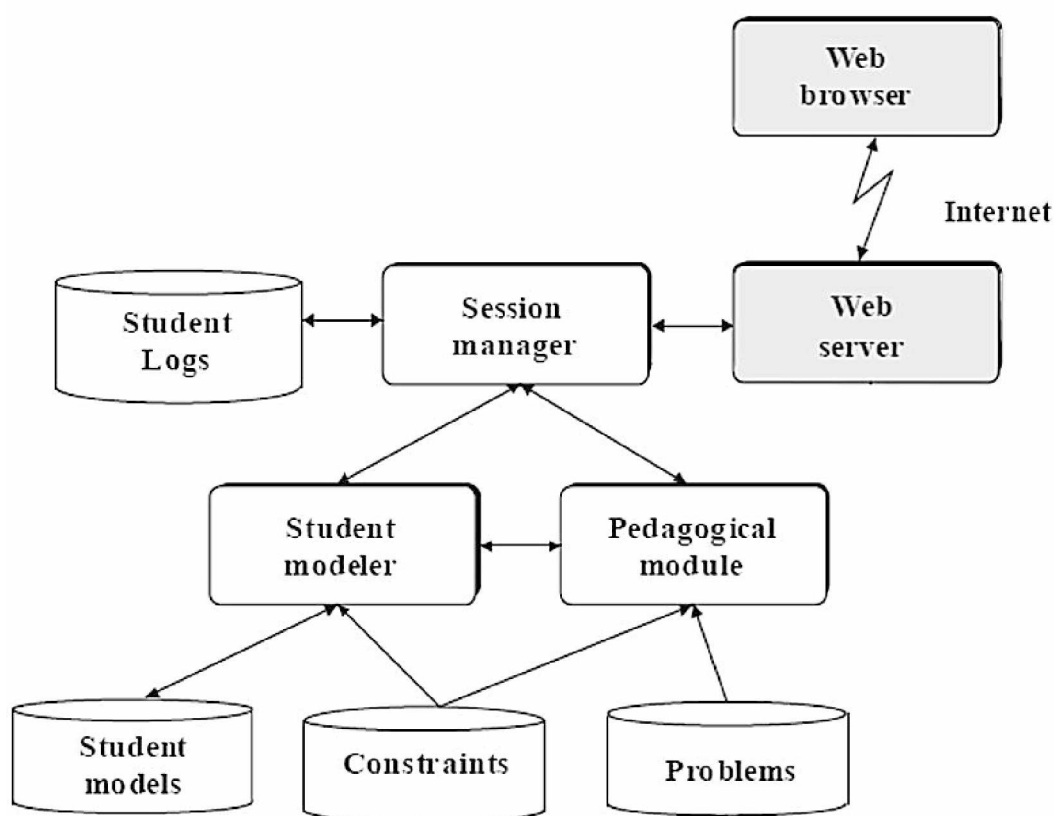


Figura 28 – Arquitetura do J-LATTE apresentada em (HOLLAND, 2009)

5.2.3.1.11 JITS

Segundo (SYKES, 2010), o JITS é um sistema tutor web para o ensino de programação na linguagem Java. A arquitetura do sistema é personalizada e apresenta dois módulos principais: o módulo front-end de apresentação e o módulo JECA de processamento. O módulo de apresentação é responsável pela interação do estudante com o tutor e envio dos dados para o módulo de processamento. O módulo JECA de processamento contém um algoritmo de correção de erros Java que realiza a análise das respostas do estudante. Este módulo possui um analisador léxico e um recuperador de erros do analisador.

O analisador léxico interpreta os identificadores indexados pelo escaneador e os compara com palavras-chave, palavras reservadas e identificadores válidos da linguagem Java. Dessa forma, o módulo tenta entender a intenção do estudante por meio de uma série de substituições e refinamentos de código. O recuperador de erros, por sua vez, tem como objetivo recuperar o analisador na ocorrência de erros através da inserção, remoção e substituição de termos em um espaço de N termos que antecedem um erro. Um fluxograma de ações é apresentado para cada um dos módulos no estudo, contudo não há uma representação esquemática da arquitetura completa do sistema JITS.

5.2.3.1.12 JO-Tutor

(ABU-NASER et al., 2011) descreve o JO-Tutor, um sistema tutor web para o ensino de programação de objetos na linguagem Java, a arquitetura do sistema é composta por seis módulos: módulo especialista, módulo do estudante, módulo pedagógico, módulo de geração de problemas, módulo do processo de tutoria e módulo de interface de usuário. As responsabilidades de cada módulo são devidamente documentadas e fornecem uma visão geral do funcionamento do sistema. Uma visão geral da arquitetura do JO-Tutor é apresentada na figura 29.

O módulo pedagógico faz a geração automática de problemas em Java para os conceitos de classes e objetos, herança e polimorfismo. O módulo de geração de problemas é responsável pela geração automática de problemas através de modelos de código. O módulo especialista coleta toda a informação necessária para fornecer um retorno de desempenho adequado ao estudante e é capaz de resolver os problemas gerados pelo módulo de geração de problemas. Este módulo é capaz ainda de verificar a corretude da solução apresentada pelo estudante e fornecer a resposta correta quando requisitado. O módulo do estudante armazena as informações pessoais do estudante como preferências pessoais e histórico de desempenho. O módulo do processo de tutoria age como um coordenador que controla toda a funcionalidade do sistema. A interface do usuário interage com o usuário por meio janelas, ícones e menus cascata.

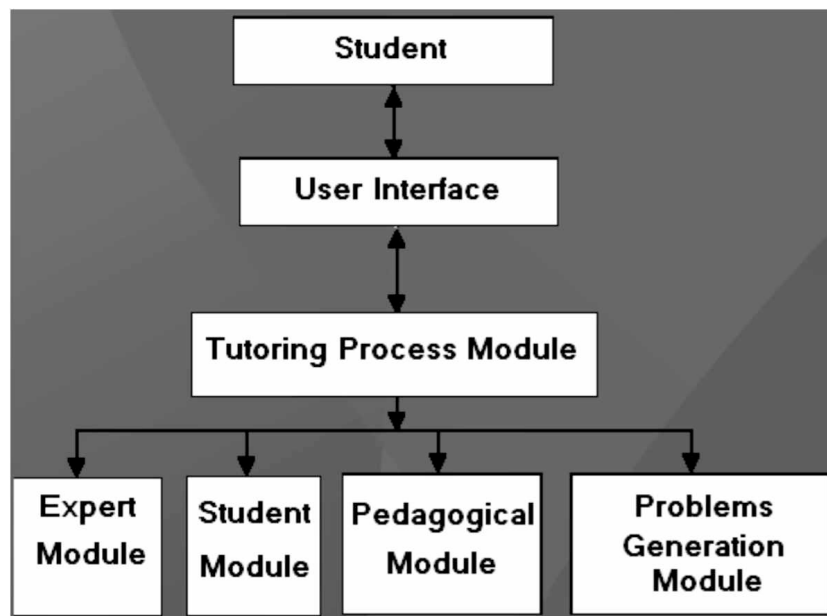


Figura 29 – Arquitetura do JO-Tutor apresentada em (ABU-NASER et al., 2011)

5.2.3.1.13 ASK-ELLE

Segundo (GERDES, 2012), o ASK-ELLE é um sistema tutor web para o ensino de programação na linguagem Haskell, a arquitetura do sistema segue os padrões arquiteturas de aplicações web e é composta por dois módulos: o módulo front-end e o módulo back-end. O módulo front-end é responsável pela interação com o usuário, enquanto que o módulo back-end cuida da geração de retorno de informações. O processo de comunicação e compilação executado pelo sistema é claramente documentado e oferece uma visão precisa do funcionamento da aplicação. A arquitetura web adotada para o ASK-ELLE é apresentada em detalhes na figura 30.

O front-end faz chamadas assíncronas para o back-end sempre que o estudante requisitar alguma informação ou enviar uma resposta por meio da aplicação web. O back-end, ao receber uma requisição, compila o programa enviado pelo estudante e retorna o seu desempenho. O script de retorno entra em ação e diagnostica o retorno do compilador a fim de produzir mensagens de erro ou dicas a serem apresentadas ao estudante. É importante destacar que o estado de raciocínio não é armazenado pelo back-end, ou seja, todo tipo de informação necessária pelo back-end para se processar uma requisição deve ser incluída no corpo da requisição.

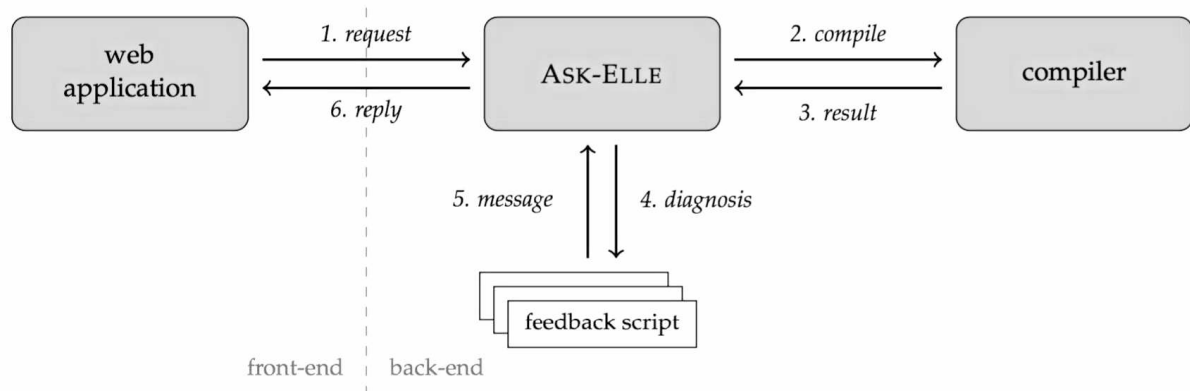


Figura 30 – Arquitetura do ASK-ELLE apresentada em (GERDES, 2012)

5.2.3.1.14 Protus 2.0

Conforme (VESIN et al., 2013), o Protus 2.0 é um sistema tutor para o ensino de programação na linguagem Java. A arquitetura do sistema é baseada em ontologias, isto é, o sistema representa cada componente em formas de ontologia: ontologia do domínio, ontologia de tarefas, ontologias de estratégias de ensino e ontologia do modelo do estudante. Esta abordagem visa promover a modularidade entre os componentes e permitir maior flexibilidade de manutenção. As responsabilidades de cada ontologia são claramente funda-

mentadas e exemplificadas e permitem uma fácil compreensão da arquitetura proposta no estudo. A arquitetura completa do Protus 2.0 é apresentada na figura 31.

A ontologia de domínio armazena todo o material de ensino, tutoriais e testes do sistema e descreve como este conteúdo deve ser estruturado. A estruturação geralmente ocorre em forma de taxonomia de conceitos, com atributos e relações conectando os conceitos uns aos outros. Para cada conceito há quatro propriedades que o descrevem: descrição, nome, valor e tipo.

A ontologia de tarefas é um sistema de vocabulário para definição da estrutura de resolução de problemas de todos os tipos de forma independente. Esta ontologia complementa a ontologia de domínio e especifica papéis específicos à cada objeto do domínio, bem como as relações entre os objetos.

A ontologia de estratégias de ensino contém as técnicas de ensino adaptativas utilizadas no sequenciamento de conteúdo apropriado ao estudante de acordo com o seu estilo de aprendizagem. Esta ontologia decide qual o conceito e recurso do sistema a ser apresentado ao estudante com base em um conjunto de condições pré-determinado.

A ontologia do modelo do estudante armazena todas as informações de desempenho do estudante, assim como suas preferências pessoais. Esta informação é atualizada constantemente a medida que o estudante interage com o conteúdo ensinado pelo sistema e é utilizada pela ontologia de estratégias de ensino no processo de adaptação de ensino.

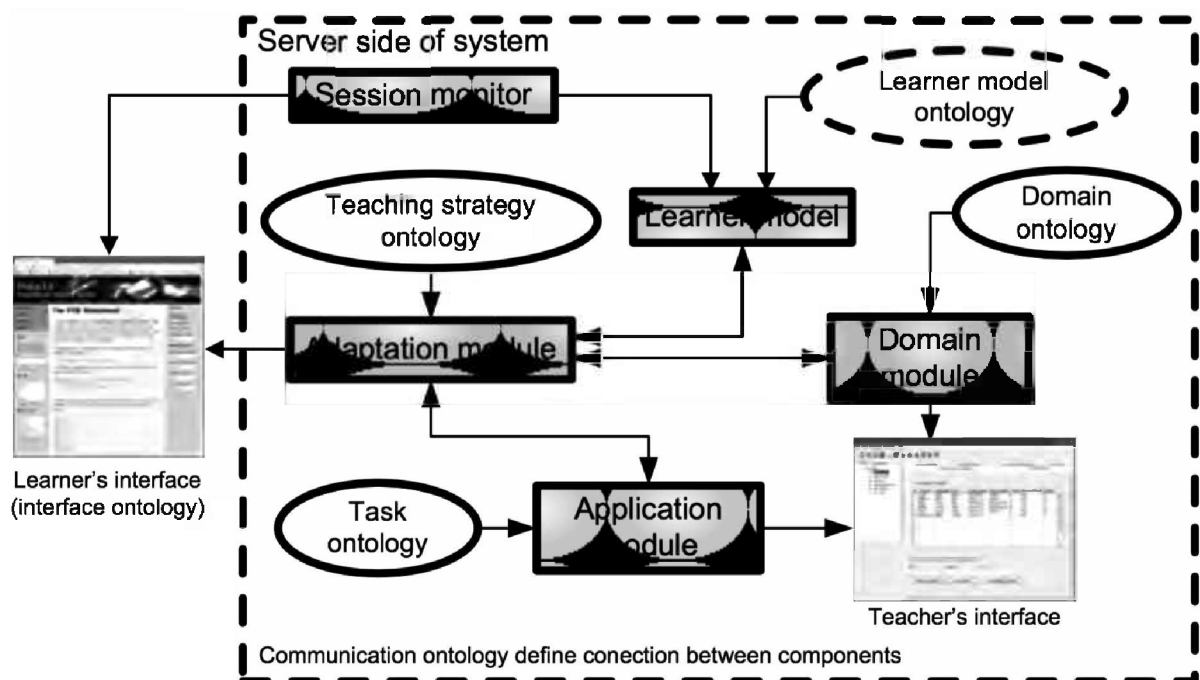


Figura 31 – Arquitetura do Protus 2.0 apresentada em (VESIN et al., 2013)

5.2.3.1.15 PHP ITS

Segundo (WERAGAMA, 2013) e (WERAGAMA; REYE, 2014), o PHP ITS é um sistema tutor para o ensino de programação na linguagem PHP, a arquitetura do sistema é baseada na arquitetura clássica e é composta por 4 módulos: módulo do domínio, módulo do estudante, módulo de ensino e módulo da comunicação. As responsabilidades de todos os módulos são claramente documentadas, assim como o processo de comunicação dos módulos durante a tutoria. Um aspecto interessante do estudo é que ele aponta que não há necessidade em utilizar uma arquitetura genérica quando os requisitos de um tutor em particular não estão pré-definidos. A arquitetura clássica adotada para o sistema PHP ITS é apresentada na figura 32. A arquitetura web adotada para o desenvolvimento do sistema é apresentada, em seguida, na figura 33.

As responsabilidades de cada módulo são as mesmas da arquitetura clássica. O módulo do domínio concentra o material de ensino utilizado pelo sistema no processo de tutoria. O módulo de ensino contém os métodos de suporte ao ensino do estudante. O módulo do estudante armazena as características pessoais do estudante e seu nível de conhecimento. O módulo da comunicação contém a interface de usuário do sistema responsável pela interação do sistema com o estudante.

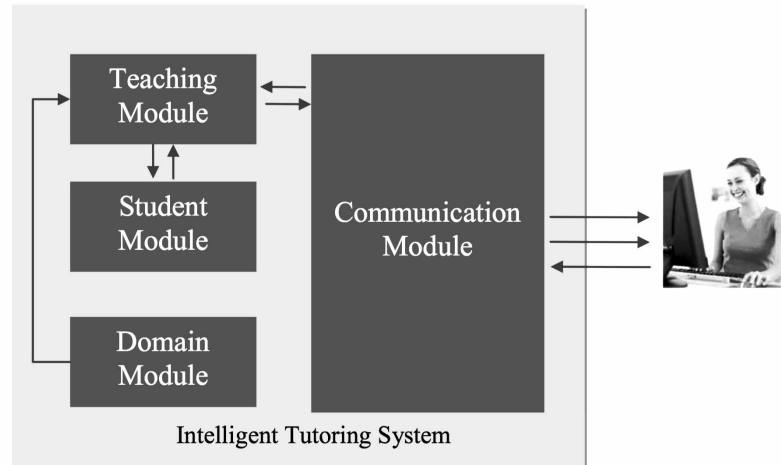


Figura 32 – Arquitetura do PHP ITS apresentada em (WERAGAMA, 2013)

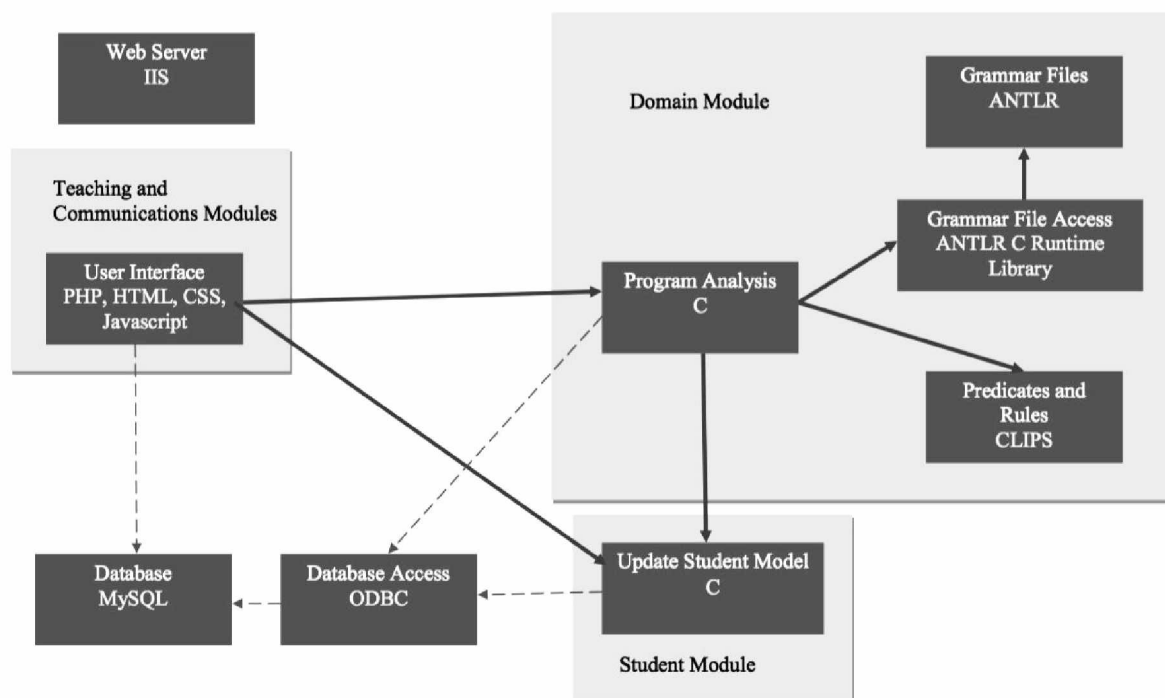


Figura 33 – Arquitetura web do PHP ITS apresentada em (WERAGAMA, 2013)

5.2.3.1.16 Java Sensei

(ZATARAIN-CABADA et al., 2015) descrevem o Java Sensei, um sistema tutor para o ensino de programação na linguagem Java. A arquitetura do sistema é composta por sete módulos: quadro de trabalho afetivo, módulo do estudante, agente pedagógico, módulo de adaptação, módulo pedagógico, módulo especialista e um mecanismo de tutoria. As responsabilidades de cada um dos módulos são claramente documentadas e fornecem uma visão precisa do funcionamento do sistema. Uma visão geral dos módulos que compõem a arquitetura do Java Sensei é apresentada na figura 34.

O quadro de trabalho afetivo implementa um sistema de lógica Fuzzy e uma rede neural para reconhecimento das emoções do estudante. O modelo do estudante representa o aspecto cognitivo e emocional do estudante e considera quadro aspectos: emoção atual, emoção anterior, habilidades gerais do estudante e qualidade da resposta apresentada. O agente pedagógico transmite as emoções e mensagens que um tutor humano normalmente apresentaria em um processo de ensino. O módulo de adaptação é responsável pela análise das características do estudante e adaptação do processo de ensino através de recomendações de mudança. O módulo pedagógico realiza a correção dos exercícios enviados pelo estudante. O módulo especialista contém o conhecimento no qual o estudante deseja aprender e praticar, representado em formato JSON. O mecanismo de tutoria guia os estudantes ao longo do processo de ensino e fornece múltiplas estratégias para a solução de problemas.

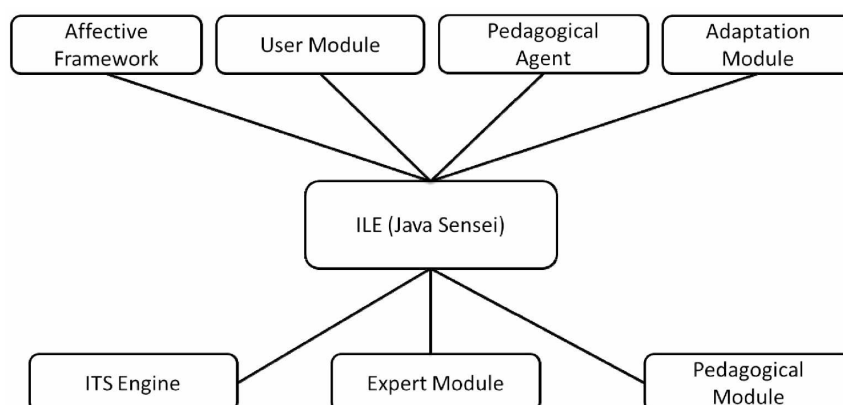


Figura 34 – Arquitetura do Java Sensei apresentada em (ZATARAIN-CABADA et al., 2015)

5.2.3.1.17 SITS

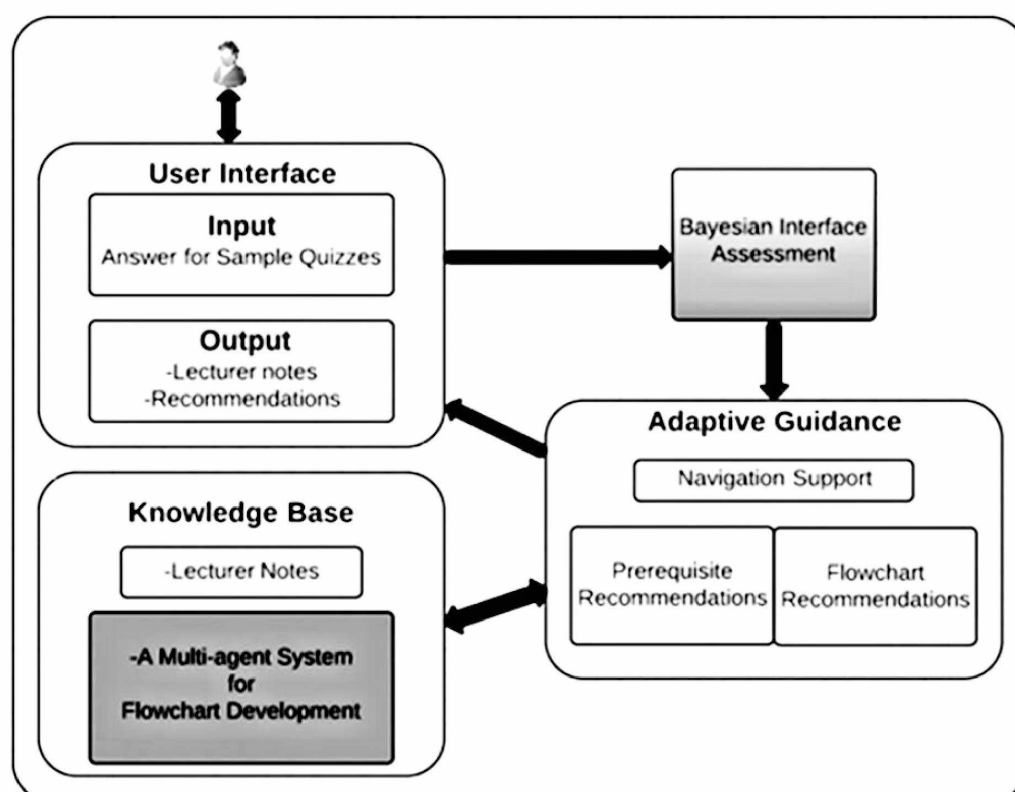


Figura 35 – Arquitetura do SITS apresentada em (HOOSHYAR et al., 2016)

(HOOSHYAR et al., 2016) descrevem o SITS, um sistema tutor baseado em soluções para a avaliação de jogos colaborativos. A arquitetura do sistema é composta por quatro módulos: módulo de redes Bayesianas, base do conhecimento, interface de usuário, orientação adaptativa. Como pode-se observar, a arquitetura do SITS é extremamente baseada na arquitetura do BITS apresentada em (BUTZ; HUA; MAGUIRE, 2006), e compartilha da mesma nomenclatura para os módulos. A arquitetura completa do SITS é apresentada na figura 35.

O módulo de redes Bayesianas realiza a inferência de dados do sistema durante o processo de ensino e armazena o nível de conhecimento do estudante. A base de conhecimento agrega o material de ensino do sistema em formato de páginas web, como também um repositório de testes. O módulo de orientação adaptativo oferece as opções de aprendizagem que melhor se adequam aos objetivos de aprendizagem e preferências do estudante. O módulo de interface de usuário interage com o estudante por meio de um navegador web e apresenta testes e sugestões pedagógicas ao longo do processo de ensino.

5.2.3.2 Protótipo

Como apontado na subseção 5.2.2.5, quatorze estudos apresentam um protótipo do sistema, enquanto que seis não apresentam. Esta subseção tem como objetivo analisar detalhadamente os protótipos apresentados e identificar as técnicas e tecnologias utilizadas no desenvolvimento de cada um deles. Lembrando que, apenas os estudos que apresentam um protótipo foram considerados.

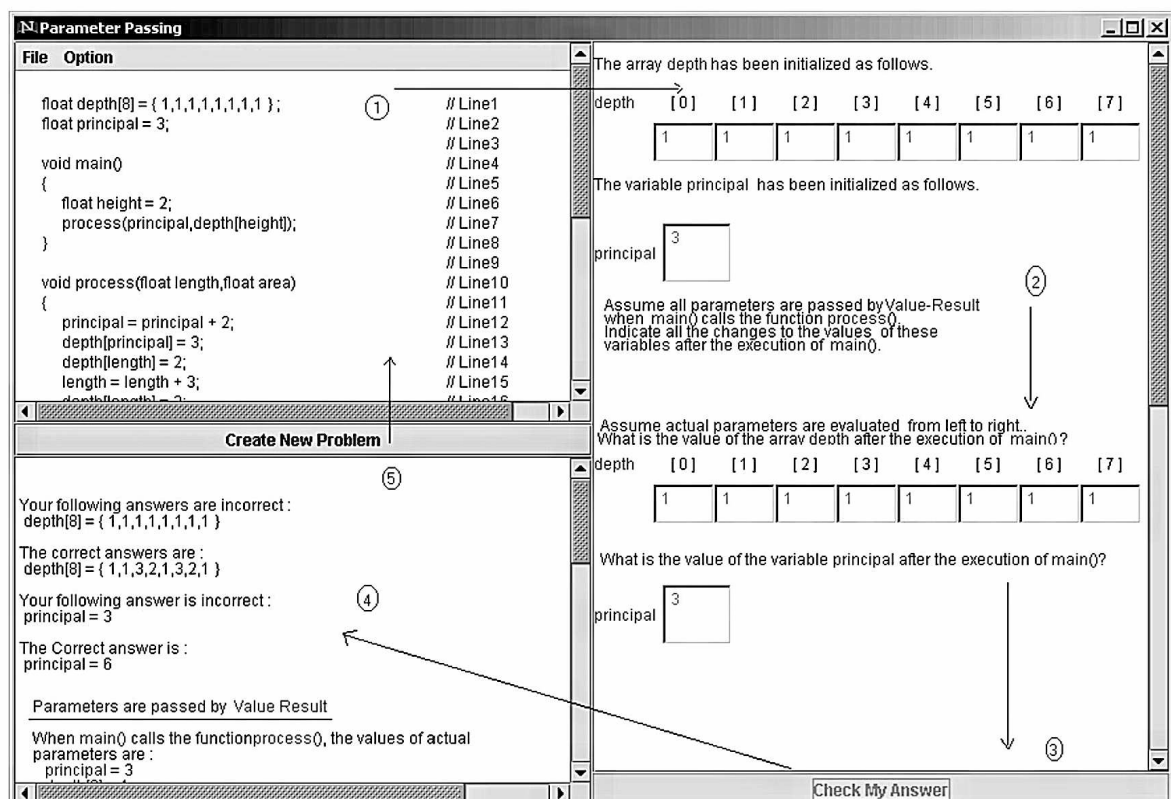


Figura 36 – Interface com o curso de ações do tutor extraída de (SHAH; KUMAR, 2002)

O sistema tutor descrito por (SHAH; KUMAR, 2002), é mostrado na figura 36 por meio de uma captura de tela com o curso de ações do sistema, realizado em sentido horário. O indicador 1 apresenta o painel com o código do programa. O indicador 2 apresenta o painel com o enunciado do problema e controles do usuário. O indicador 3 apresenta o botão para

verificação da resposta. O indicador 4 apresenta o painel com o feedback do usuário. Por último, o indicador 5 apresenta o botão para criação de um novo problema. É importante citar que, os botões só se tornam disponíveis no contexto apropriado a fim de evitar confusão por parte do usuário. O sistema é baseado em web e implementado por meio de applets Java, o que permite sua disponibilidade sem restrições de horário e ambiente físico. Há no total um conjunto de 18 classes Swing responsáveis pelo funcionamento do sistema. Ainda, segundo os autores, a interface de usuário, o módulo do estudante e partes do módulo tutor podem ser reutilizadas em outros sistemas tutores, caso desejado.

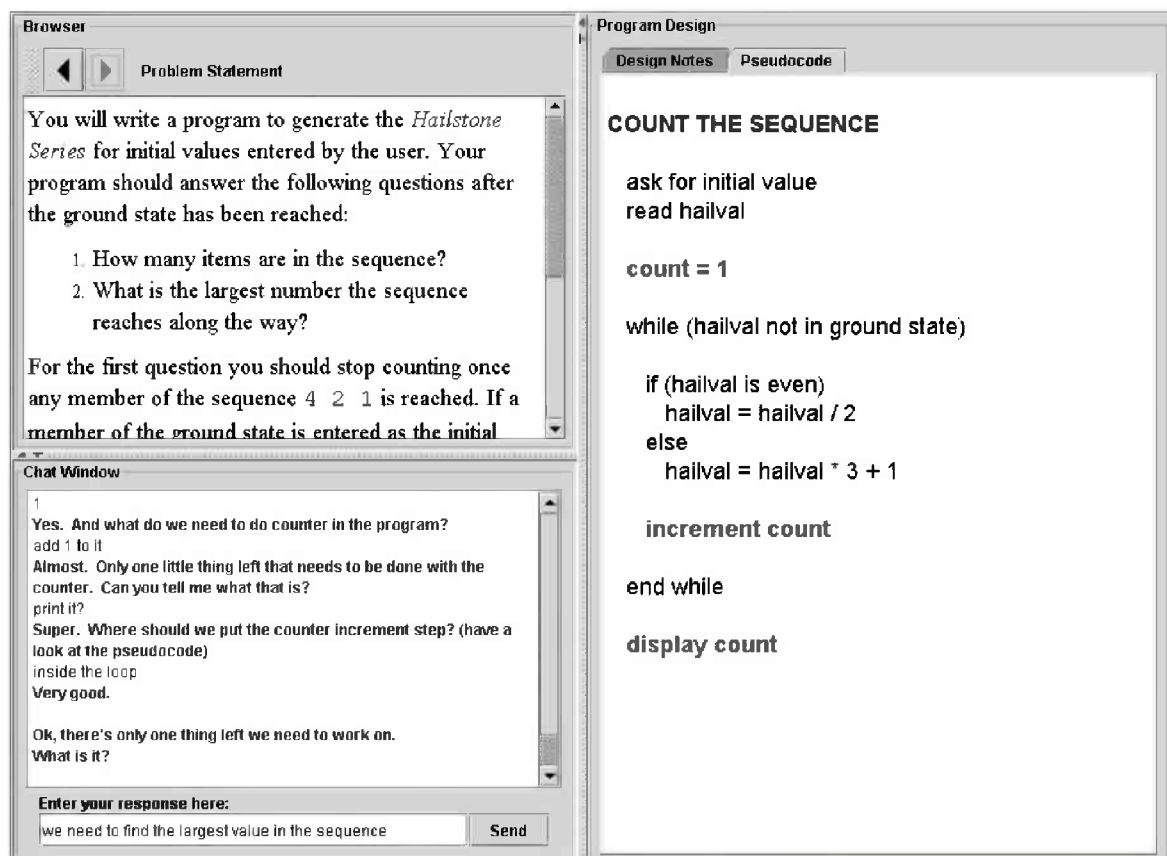


Figura 37 – Interface principal do ProPl extraída de (LANE; VANLEHN, 2004)

O sistema ProPl, apresentado em (LANE; VANLEHN, 2004), é mostrado na figura 37 por meio de uma captura de tela com a interface do sistema. A interface do ProPl consiste em três janelas principais. No topo esquerdo da tela há um pequeno navegador que apresenta o enunciado do problema e materiais de ensino. Abaixo deste navegador existe uma janela de chat responsável pelo diálogo entre o tutor e o estudante. Preenchendo a metade direita da tela encontra-se um painel com duas abas: a primeira aba apresenta notas de design e a outra apresenta a solução em pseudocódigo. O sistema é baseado em web e sua interface roda em qualquer navegador que tenha o Java habilitado. O controle do processo de tutoria é realizado pelo back-end desenvolvido na linguagem Lisp.

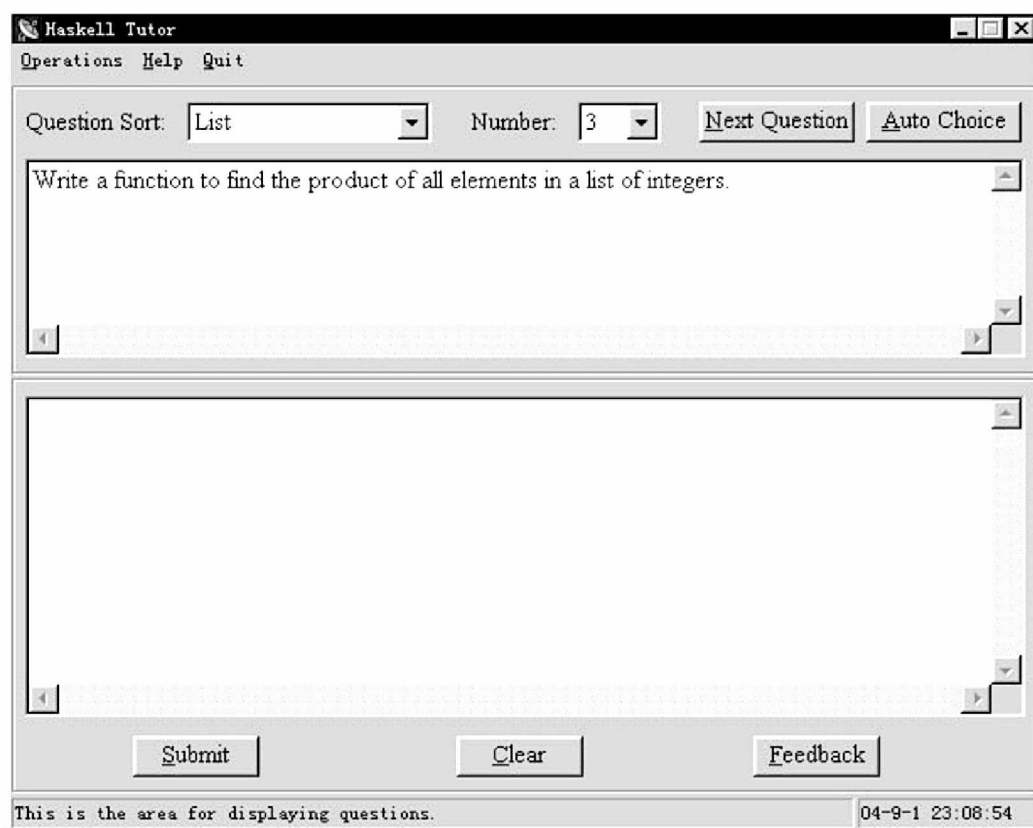


Figura 38 – Interface principal do Haskell-Tutor extraída de (XU; SARRAFZADEH, 2004)

O sistema Haskell-Tutor, apresentado em (XU; SARRAFZADEH, 2004), possui três níveis de conhecimento que podem ser selecionados pelo usuário: novato, médio e especialista. A escolha ou identificação de um problema pode ser realizada pelos campos localizados no topo superior da tela, conforme visto na figura 38. Na janela superior, é apresentada a declaração do problema. Na janela inferior, o estudante colocará sua resposta e, utilizará o botão Submit para submeter a solução para o compilador do sistema. O botão clear tem como objetivo facilitar a limpeza da janela inferior de resposta, enquanto que o botão feedback tem como objetivo apresentar mensagens de feedback ao estudante com relação ao seu desempenho.

O sistema C++ STL ITS, apresentado em (LEE; BABA, 2005), é mostrado na figura 39. Como visto anteriormente, o sistema é baseado em uma arquitetura de três camadas: cliente, mediador e persistência. A interface se encontra na camada cliente e é composta por duas partes: páginas web dinâmicas geradas em HTML pelo mediador e um navegador para apresentação do conteúdo e interação com o usuário. Como pode ser verificado na figura, para cada tutorial há subtutoriais com diferentes níveis de estratégias de ensino baseados nos resultados de um pré-teste aplicado ao estudante. Maiores informações sobre o questionário pré-teste são encontradas no estudo. O estudante pode ainda requisitar ajuda através do botão de dicas localizado no canto inferior direito da tela.

```
#include <iostream>
#include <>

using namespace std;

int main()
{
    // Declare a vector of 10 integers, myVector
    
    int i;
```

Sub Tutorials

Sub-Tutorial 1 Sub-Tutorial 2 Sub-Tutorial 3

Write the statement to create a vector called myVector that can store up to 10 integers.

Hint ? Submit All Answers

Figura 39 – Interface de ensino do C++ STL ITS extraída de (LEE; BABA, 2005)

O sistema BITS, apresentado em (BUTZ; HUA; MAGUIRE, 2006), é exemplificado na figura 40 por meio de uma captura de tela mostrando um curso para o conceito de números de ponto-flutuante. Por meio da figura, pode-se compreender o processo de aprendizagem de novos conceitos no sistema. O material de leitura é apresentado na janela esquerda da tela, enquanto que os pré-requisitos são identificados na janela no topo direito. Após a leitura, o estudante indica o seu nível de compreensão por meio do formulário de seleção localizado no canto inferior direito da tela. A rede Bayesiana e o menu de navegação do sistema são atualizados sempre que houver uma mudança no estado de aprendizagem do estudante.

O sistema BITS é baseado em web e possui a interface baseada em páginas HTML e arquivos multimídia Flash combinados com documentos XML. A sua exibição é realizada por um navegador web integrado no sistema, que também permite o acesso a outros materiais de C++ na Internet. Para o desenvolvimento do framework do BITS foram usadas as seguintes ferramentas: Visual Studio.net, Dreamweaver MX, Flash MX e Fireworks MX. Mais detalhes da implementação do sistema são encontrados no estudo.

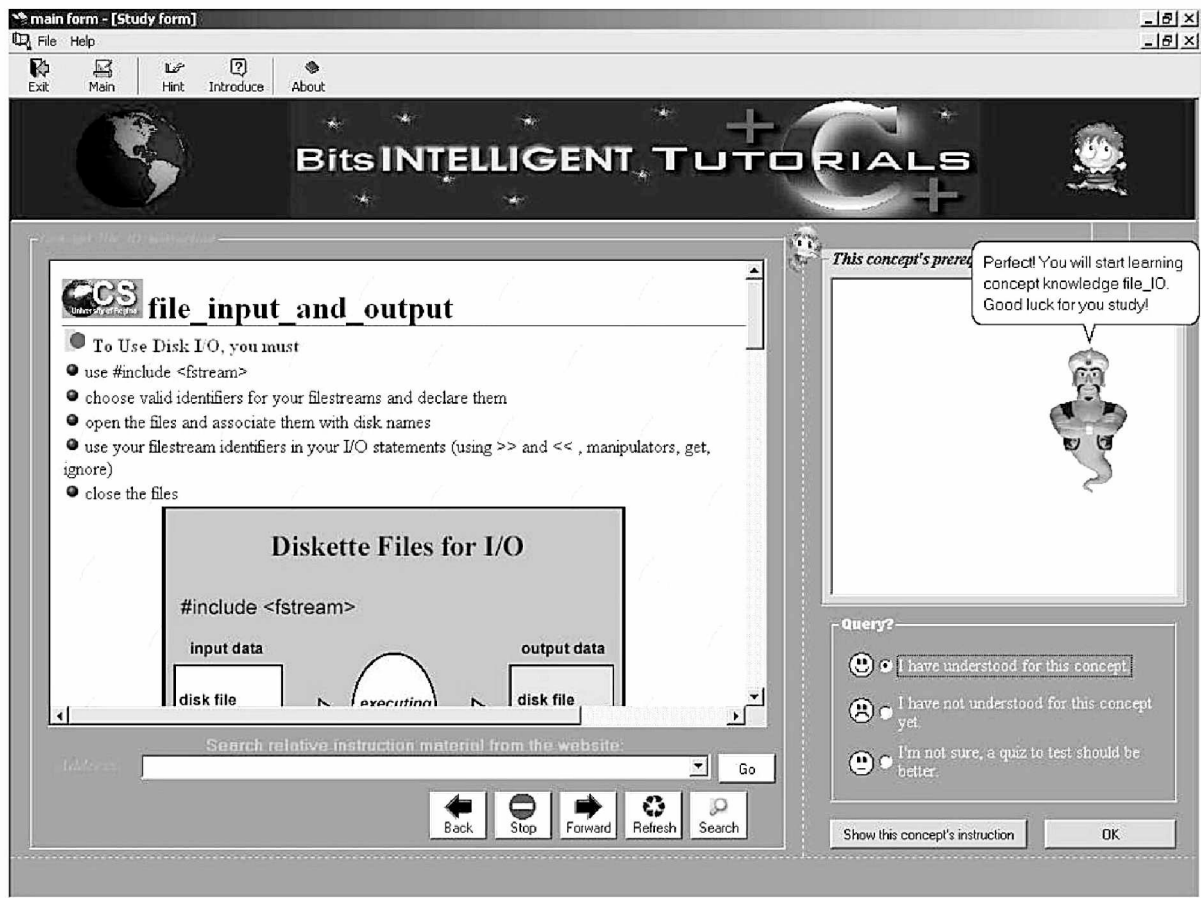


Figura 40 – Interface de ensino do BITS extraída de (BUTZ; HUA; MAGUIRE, 2006)

A interface do sistema tutor desenvolvido por (EMURIAN, 2006) é exibida na figura 41. A captura de tela selecionada apresenta um exemplo de teste de múltipla escolha. Como pode ser observado, o enunciado é apresentado na região superior da janela, enquanto que a pergunta e as opções de resposta se encontram na região inferior. O sistema é baseado em web e objetiva o ensino de um applet Java básico.

A interface de usuário do sistema CPP-Tutor, apresentado em (NASER, 2009), é mostrada na figura 42. Na metade superior da tela encontra-se um problema a ser solucionado pelo estudante, assim como as especificações do programa a ser codificado. Na metade inferior, o estudante tem acesso a um editor de código, botões de interação com o tutor e o resultado gerado pelo sistema. As seguintes funcionalidades são disponibilizadas através dos botões de interação: requisição de um novo problema, submissão da solução do estudante para avaliação, requisição de dicas, visualização de estatísticas de desempenho, visualização da solução do problema e saída do problema para escolha de um novo.

Uma representação da interface de usuário do sistema J-LATTE, apresentado em (HOLLAND, 2009), é mostrada na figura 43. A interface é dividida em vários painéis e possibilita ao estudante a realização de todas as interações necessárias, com exceção do login.

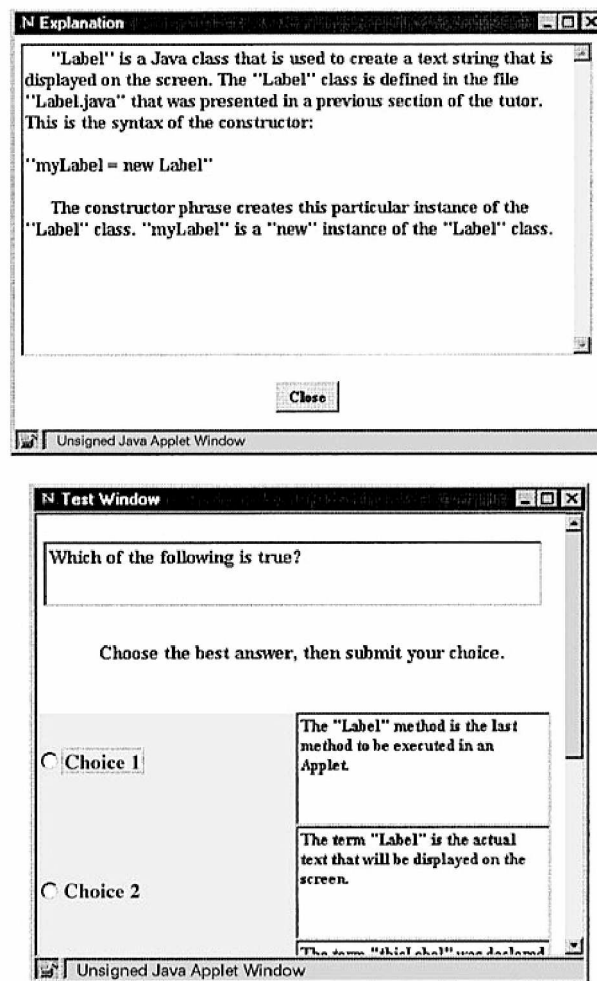


Figura 41 – Interface principal do tutor extraída de (EMURIAN, 2006)

Isto inclui operações como a seleção de problemas, solução de problemas e submissão de uma solução. A leitura e resolução de problemas é realizada nos painéis localizados no lado esquerdo, enquanto que, a apresentação de feedback ao estudante é realizada no painel direito. O sistema é baseado em web e foi desenvolvido utilizando tecnologias AJAX, uma combinação de XHTML dinâmico, Javascript e requisição assíncrona de objetos. Isto possibilitou que as páginas fossem dinâmicas e não exigissem a atualização da página a cada vez que houvesse uma interação com o servidor. O servidor, por sua vez, foi implementado usando o AllegroServe, que é um servidor web baseado em Lisp. Mais informações a respeito da interação cliente-servidor são encontradas no estudo.

A interface do sistema JITS, apresentado em (SYKES, 2010), é mostrada na figura 44. O indicador 1 apresenta uma mensagem personalizada ao estudante assim que o login é realizado. O indicador 2 apresenta uma nota relativa ao estado de solução do problema. O indicador 3 apresenta o problema incluindo sua declaração, especificações e retorno esperado. O indicador 4 apresenta uma imagem caso o problema tenha um apoio visual. O indicadores 5 mostra o modelo do sistema para apresentação de cada problema. O indicador

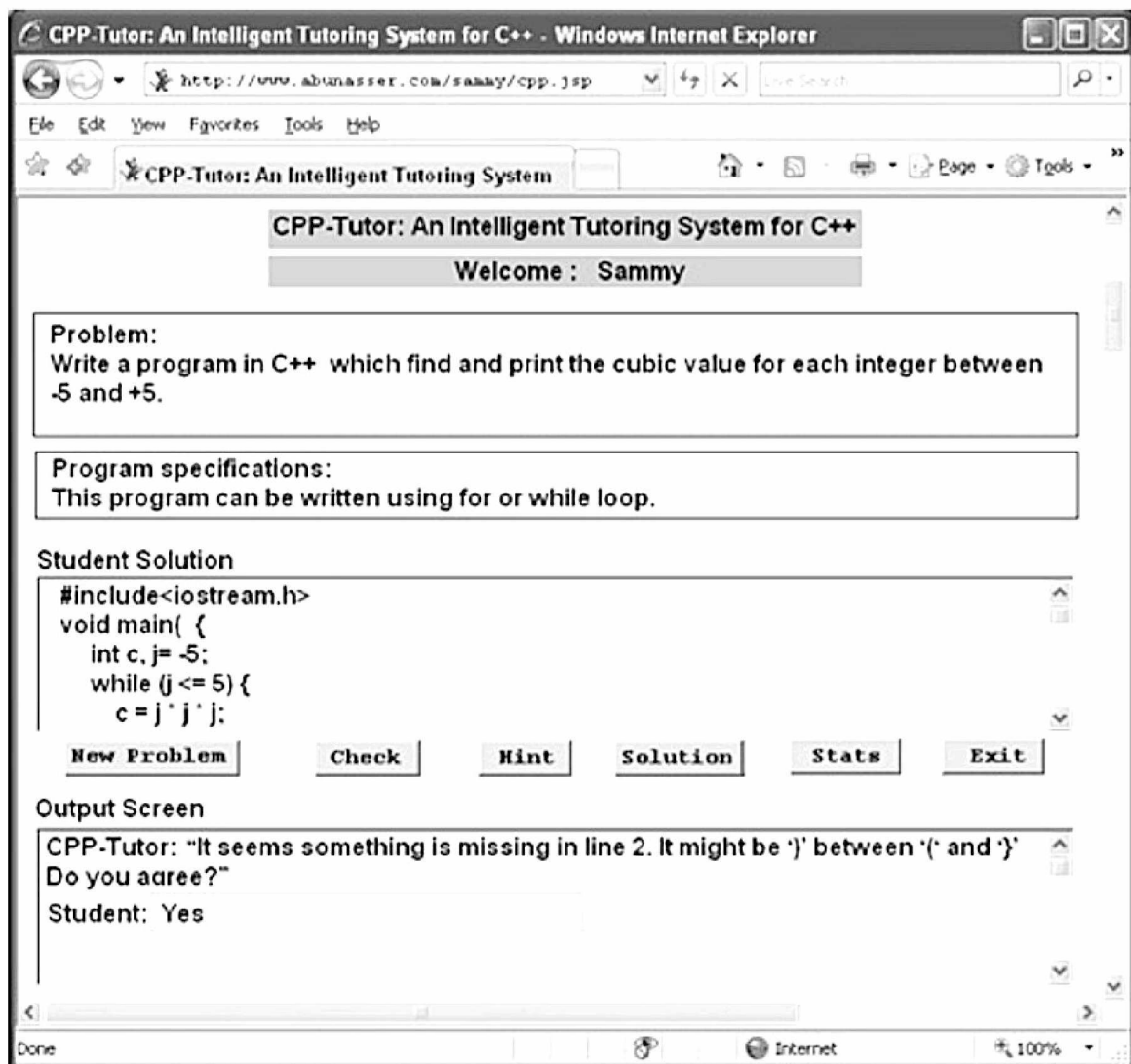


Figura 42 – Interface principal do CPP-Tutor extraída de (NASER, 2009)

6 apresenta a região na qual o estudante irá editar sua resposta. O indicador 7 apresenta os vários botões utilizados para interação com o sistema. O indicador 8 apresenta o local onde dicas, soluções, erros e estatísticas de desempenho são apresentadas. O indicador 9 apresenta os tópicos que podem ser selecionados pelo estudante. O indicador 10 apresenta o botão utilizado para abertura de uma janela de tutorial complementar. O indicador 11 mostra o botão de ajuda oferecido para auxiliar o estudante a navegar pelo sistema. O indicador 12 mostra o botão de saída do sistema.

O sistema JITS é baseado em web e foi implementado utilizando o padrão de projeto MVC, Model-View-Controller, em conjunto com as tecnologias Java ServerPages, também conhecida como JSP, e JavaBeans. Todo o processamento é realizado na camada mediadora, responsável pela extração dinâmica das informações do banco de dados Oracle e apresentação apropriada dos dados nas páginas JSP. A arquitetura distribuída do JITS confere algumas

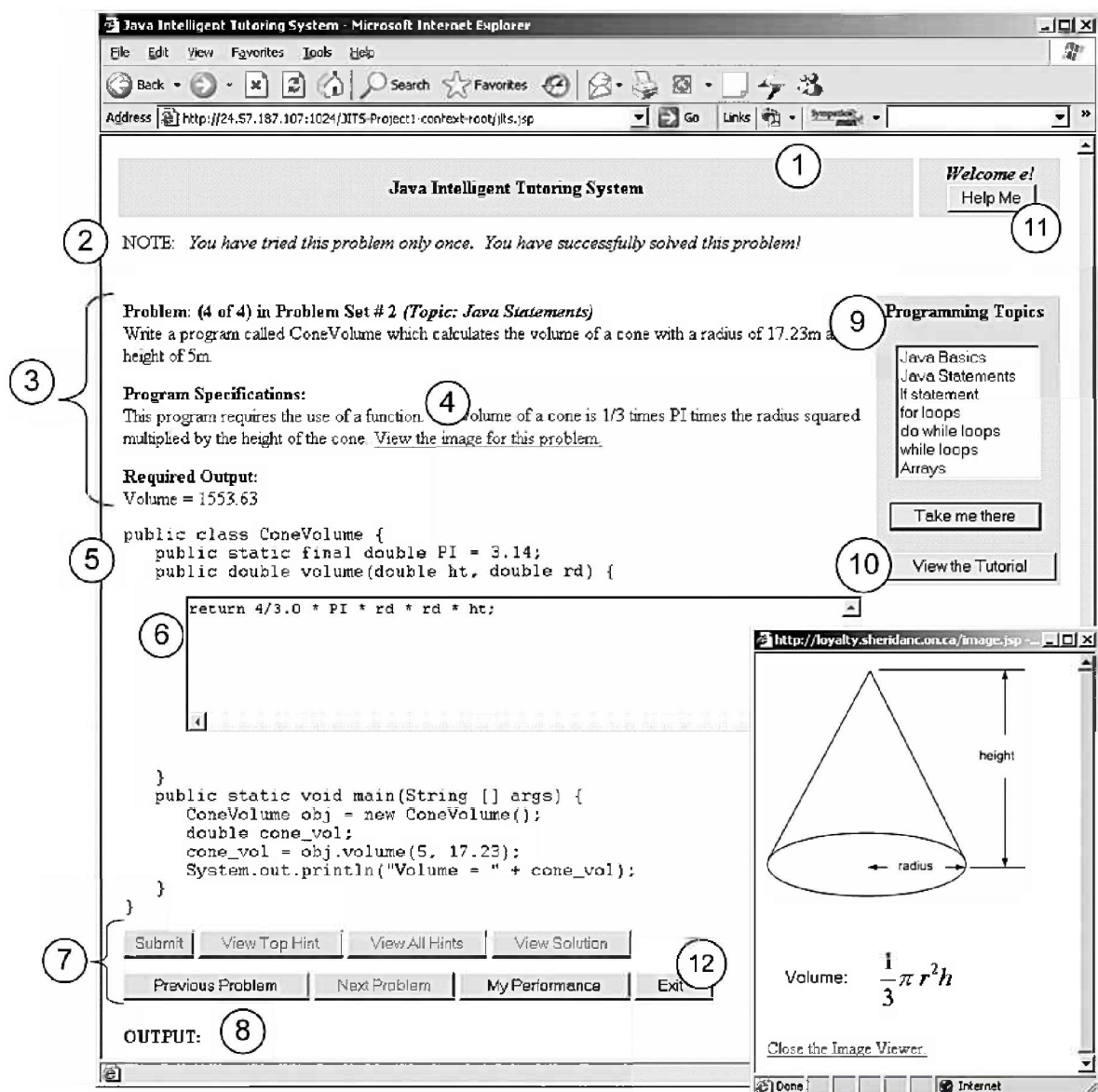


Figura 44 – Interface principal do JITS extraída de (SYKES, 2010)

mentares de navegação do sistema e feedback estão disponíveis no painel direito. O sistema é baseado em web e acessível através de um navegador. A interface de usuário foi desenvolvida em PHP, HTML, CSS e Javascript. O banco de dados utilizado foi o MySQL com o acesso provido pelo ODBC. Os módulos de análise do domínio e atualização do modelo do estudante foram codificados na linguagem C. Outras bibliotecas adicionais complementam o módulo do domínio e são descritas em detalhes no estudo.

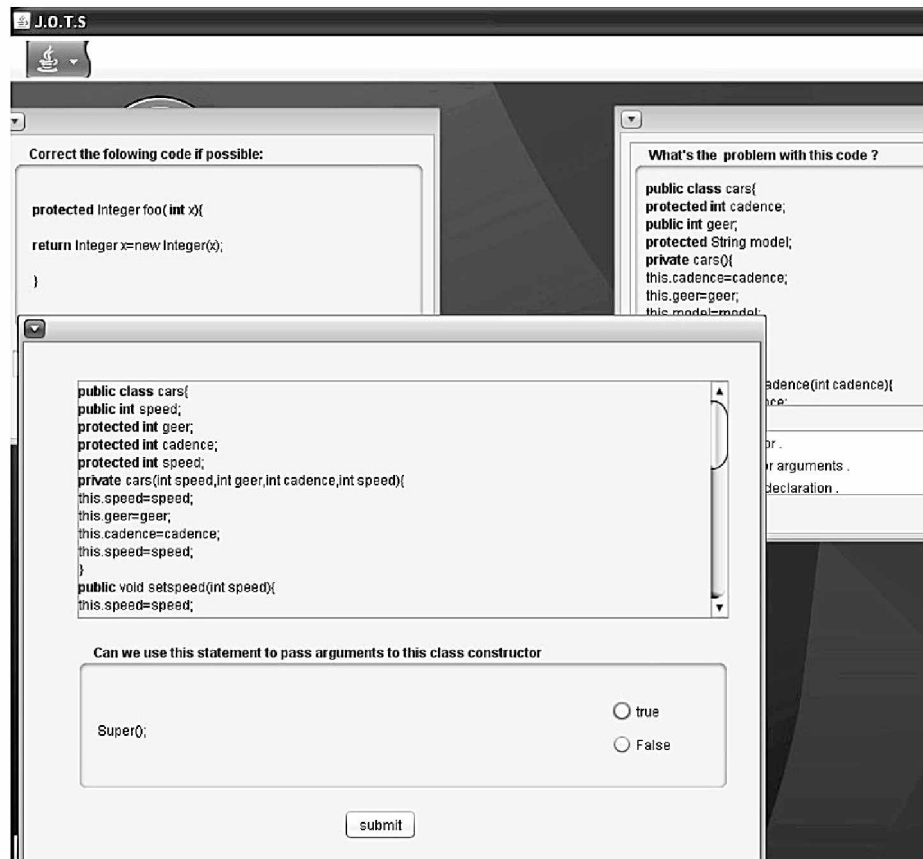


Figura 45 – Interface principal do JO-Tutor extraída de (ABU-NASER et al., 2011)

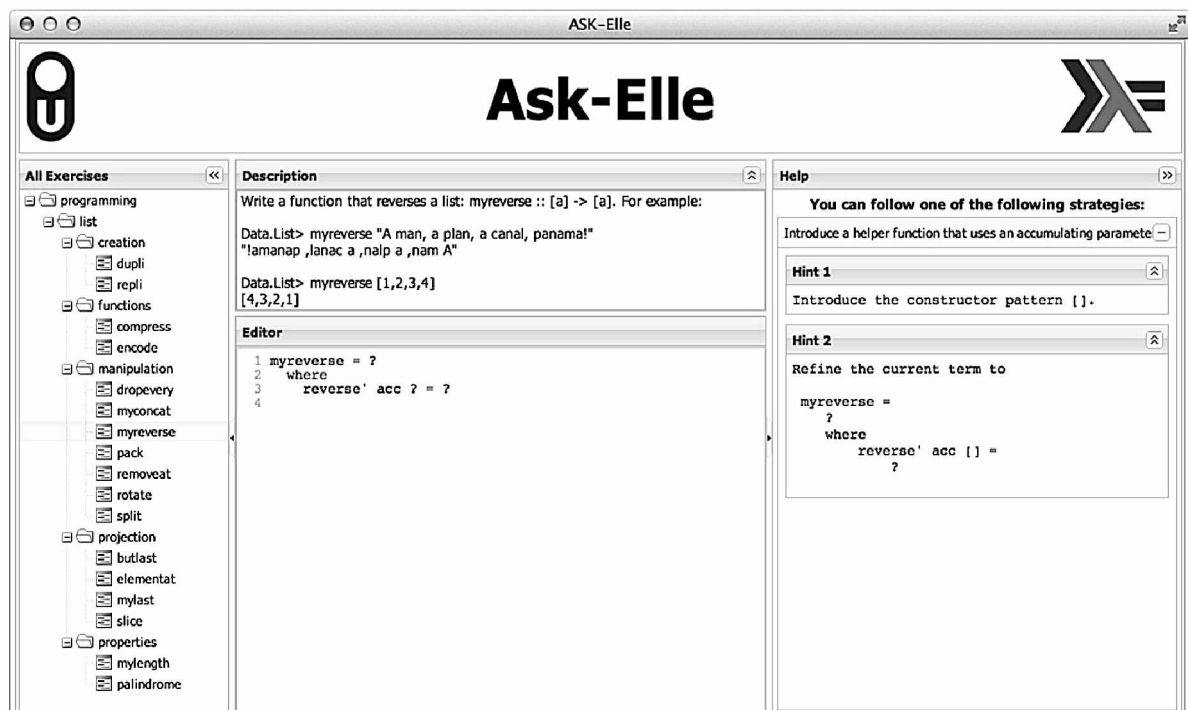


Figura 46 – Interface principal do ASK-ELLE extraída de (GERDES, 2012)

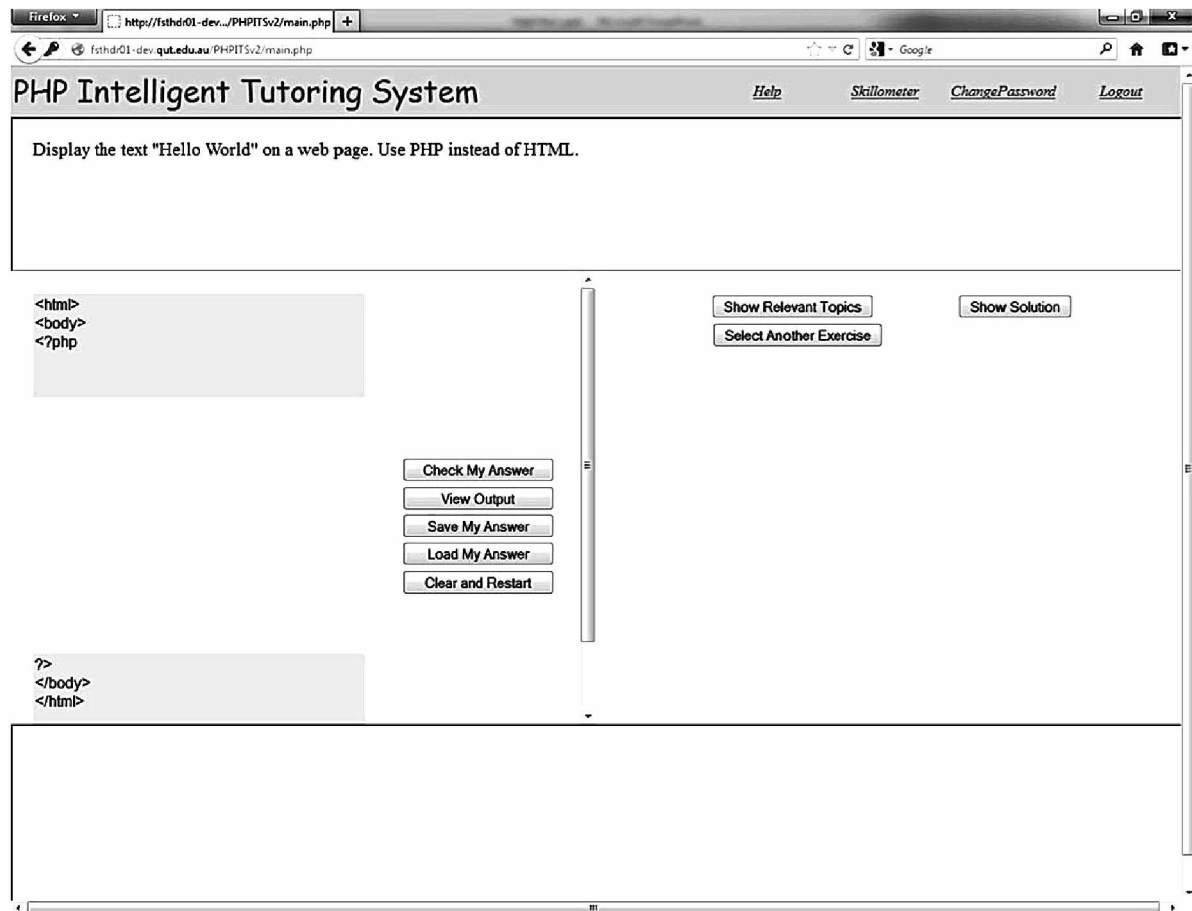


Figura 47 – Interface de solução de problemas do PHP ITS extraída de (WERAGAMA, 2013)

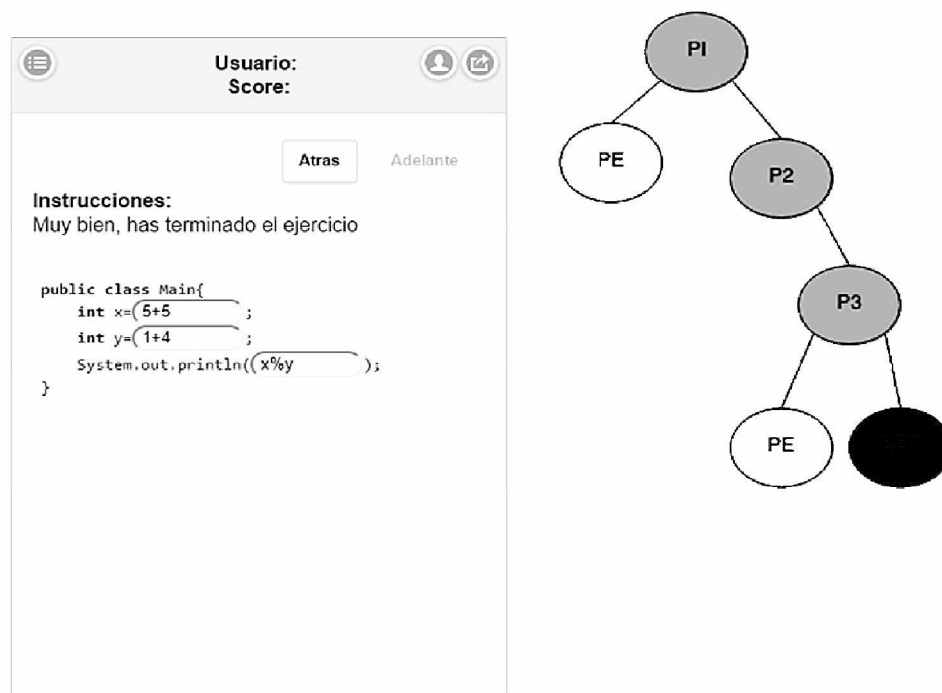


Figura 48 – Protótipo do Java Sensei apresentado em (ZATARAIN-CABADA et al., 2015)

A interface do sistema Java Sensei, apresentado em (ZATARAIN-CABADA et al., 2015), é baseada em web e exemplificada na figura 48. A captura de tela mostra uma sessão de um estudante em que obtêm sucesso em sua solução. A cor verde escura utilizada para o plano de fundo do nó PFO simboliza uma solução correta. Caso o estudante tivesse selecionado o nó PE, a cor vermelha seria utilizada para notificar que há um erro na solução. As tecnologias e bibliotecas utilizadas para o desenvolvimento do sistema são: Java, Javascript, MongoDB, JSON, jFuzzyLogic, Apache Mahout, Weka, jQuery, jQuery Mobile, entre outras.

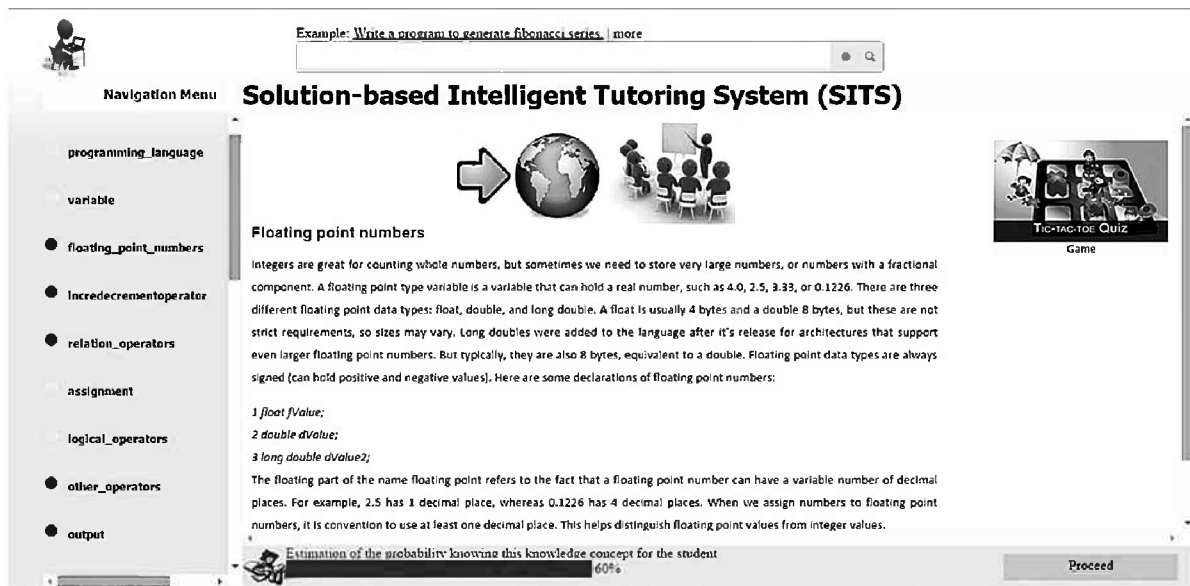


Figura 49 – Interface de ensino de conceitos do SITS extraída de (HOOSHYAR et al., 2016)

A interface de usuário do sistema SITS, apresentado em (HOOSHYAR et al., 2016), é exemplificada na figura 49. A captura de tela ilustra a apresentação de materiais de aprendizagem para o conceito de números de ponto flutuante. Como pode ser observado através da figura, o menu de navegação do sistema é exibido no lado esquerdo da tela e inclui suporte à orientação adaptativa por meio da marcação dos conceitos com cores diferentes. A cor verde representa um conceito desconhecido, mas pronto para aprender, a cor amarela representa um conceito já conhecido, e a cor vermelha representa um conceito desconhecido e não indicado para aprendizagem dado o atual estado de conhecimento do estudante. O sistema é baseado em web e apresenta a interação com o usuário realizada por meio de um navegador. Para mais informações e visualização de outras telas dirigir-se ao estudo.

5.2.4 Conclusões do processo de revisão

Os estudos são divididos em dois conjuntos, AC e AS, a fim de facilitar o processo de conclusão dos dados extraídos e definição de uma arquitetura inicial. O conjunto AC contém nove dos dezessete estudos analisados e abrange os estudos que apresentam uma mesma base arquitetural, enquanto que, o conjunto AS contém oito dos dezessete estudos analisados e engloba os estudos que apresentam arquiteturas singulares.

O conjunto AC, que representa uma porcentagem aproximada de 53% do total, abrange estudos que descrevem uma arquitetura de sistema baseada na arquitetura clássica de quatro componentes vista no capítulo 4: módulo do domínio, módulo do estudante, módulo pedagógico e módulo da comunicação. Um excelente indício, pois, indica que esta arquitetura é vista como o ponto de partida na maior parte dos estudos considerados. Os estudos pertencentes a esse conjunto são: (EL-KHOULY; FAR; KOONO, 2000), (XU; SARRAFZADEH, 2004), (LEE; BABA, 2005), (BUTZ; HUA; MAGUIRE, 2006), (NASER, 2008) + (NASER, 2009), (ABU-NASER et al., 2011), (WERAGAMA, 2013) + (WERAGAMA; REYE, 2014), (ZATARAIN-CABADA et al., 2015) e (HOOSHYAR et al., 2016). O símbolo de adição '+' retrata que um conjunto de publicações é considerado como um único estudo.

O conjunto AS que representa uma porcentagem aproximada de 47% do total, engloba estudos que contém arquiteturas únicas e dissemelhantes. Isto quer dizer que todas as arquiteturas são válidas, contudo a peculiaridade de cada uma impossibilita a inclusão das suas características na arquitetura recomendada neste trabalho, uma vez que as arquiteturas atendem propósitos específicos e não podem ser caracterizadas como básicas ou iniciais. Os estudos pertencentes a esse conjunto são: (SONG et al., 1997), (KUMAR, 2002), (LANE; VANLEHN, 2004), (EMURIAN et al., 2000) + (EMURIAN, 2006), (HOLLAND, 2009), (SYKES, 2010), (GERDES, 2012) e (VESIN et al., 2013).

Como descrito acima, todos os estudos pertencentes ao conjunto AC possuem a mesma base arquitetural, fazendo com que este conjunto seja escolhido para a definição de uma arquitetura básica para sistemas tutores inteligentes orientados ao ensino de programação. Uma análise das semelhanças e disparidades encontradas entre as nove arquiteturas descritas neste conjunto de estudos.

Como previamente indicado, a principal semelhança entre os sistemas é o fato que apresentam a arquitetura clássica de 4 componentes utilizada como arquitetura final do sistema ou embutida como base arquitetural. É importante observar que, entre os estudos considerados há divergências na nomenclatura dos módulos, isto é, o mesmo módulo é referenciado com dois ou mais nomes distintos ao decorrer da leitura dos trabalhos. A identificação e equiparação é possível através da leitura e observação das responsabilidades atribuídas a cada um dos módulos para o funcionamento dos sistemas.

A arquitetura clássica é uma arquitetural inicial sólida para um sistema tutor inteligente como demonstrado pela literatura e comprovado pelos sistemas tutores do conjunto AC. Por esta razão, os quatro módulos: módulo do domínio, módulo do estudante, módulo pedagógico e módulo da comunicação, serão mantidos na arquitetura sugerida por este trabalho, visto que já é um quadro de trabalho arquitetural bem consolidado para a construção de um STI. Entretanto, as particularidades do domínio de programação também são levadas em consideração assim para a inclusão de módulos complementares.

O aspecto que mais se destacou entre os sistemas do conjunto AC, foi a inclusão do módulo especialista como um módulo distinto em cinco dos nove estudos do conjunto (EL-KHOULY; FAR; KOONO, 2000), (XU; SARRAFZADEH, 2004), (NASER, 2008) + (NASER, 2009), (ABU-NASER et al., 2011) e (ZATARAIN-CABADA et al., 2015). As atribuições do módulo especialista são geralmente observadas no módulo pedagógico, contudo, há várias justificativas válidas para a sua segregação como:

- Aumento da flexibilidade arquitetural do sistema.
- Manutenção corretiva mais eficiente, uma vez que permite mudanças nas atribuições do módulo sem impacto direto em outros módulos do sistema.
- Possibilidade de alteração de algoritmos de correção de exercícios sem interferência nos dados do módulo pedagógico.

Outro módulo merecedor de atenção é a inclusão do módulo de geração de problemas (XU; SARRAFZADEH, 2004) e (ABU-NASER et al., 2011). Este módulo é responsável pela geração de problemas em sistemas nos quais o conjunto de questões é limitado e foi observado como bastante útil para o funcionamento geral dos sistemas em que foi introduzido, principalmente na geração de modelos de código de linguagens de programação. Conforme (ABU-NASER et al., 2011) exemplifica em seu estudo, o módulo pode ser utilizado para criação aleatória de exercícios de programação por meio da estruturação de pedaços de código tais como: tipos de retorno, tipos de modificadores de acesso, tipo de argumentos de entrada, métodos, classes e nomes de argumentos.

A figura 50 apresenta a arquitetura de seis módulos definida neste trabalho para um sistema tutor educacional orientado ao ensino de programação. Os módulos são: módulo do domínio, módulo do estudante, módulo pedagógico, módulo da comunicação, módulo de correção de problemas e o módulo de geração de problemas.

As responsabilidades do módulo especialista são executadas pelos módulos de correção e geração de problemas. Esta divisão da responsabilidade do especialista em dois módulos distintos aprimora a flexibilidade de um sistema tutor educacional e possibilita um melhor controle do ciclo evolutivo de desenvolvimento, uma vez que permite que modificações nas técnicas de raciocínio de cada um dos módulos sejam realizadas separadamente.

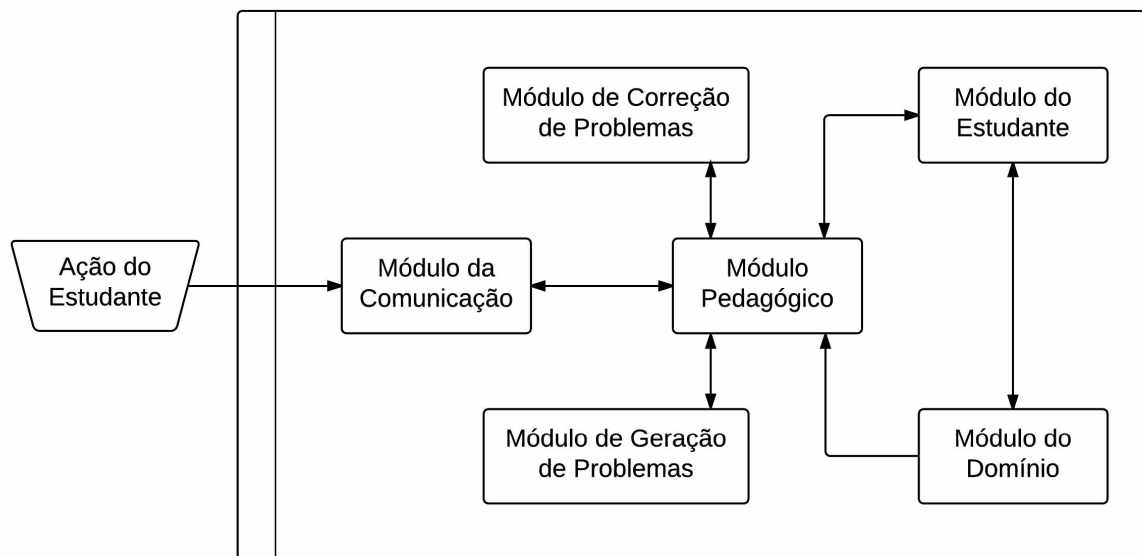


Figura 50 – Arquitetura sugerida para um sistema tutor educacional

A seguir é apresentado um exemplo de interação do usuário com o sistema. O estudante inicia sua sessão por meio do módulo da comunicação. As informações individuais do estudante são recuperadas do módulo do estudante e o sequenciamento de conteúdo de curso é iniciado pelo módulo pedagógico. O conteúdo didático é solicitado pelo módulo pedagógico ao módulo do domínio e exibido ao estudante. O estudante ao solicitar uma questão, aguarda o retorno do módulo pedagógico. Lembrando que, a questão pode estar presente no módulo do domínio ou ser gerada em tempo real pelo módulo de geração de problemas, nos sistemas em que está presente. Em ambos os casos, o módulo pedagógico faz o intermeio da comunicação. Ao solucionar a questão solicitada, o estudante envia sua resposta. O módulo pedagógico é invocado pelo módulo da comunicação mais uma vez e atribui a responsabilidade de correção do exercício para o módulo de correção de problemas, que corrige o exercício e notifica o módulo pedagógico a respeito do desempenho do estudante. O módulo do estudante prontamente armazena as novas informações de utilização do estudante e o retorno é apresentado ao estudante.

6 Conclusão

Este trabalho apresentou em detalhes a metodologia de revisão sistemática da literatura como instrumento de determinação de uma arquitetura básica de um sistema tutor inteligente orientado ao ensino de programação. Todas as etapas da revisão foram cuidadosamente definidas e apresentadas a fim de assegurar a rigorosidade e a imparcialidade do processo de revisão, como também a reprodutibilidade do estudo por outros pesquisadores da área computacional.

A metodologia se mostrou bastante eficiente para a coleta de material relevante aos fins da pesquisa e também para a análise do estado da arte do domínio do conhecimento escolhido, o que possibilitou atingir todos os objetivos do trabalho. Através do estudo de caso apresentado no capítulo 5, foi mostrado que a metodologia de revisão sistemática de literatura pode ser considerada uma boa técnica para a determinação de uma arquitetura básica para um sistema, e pode-se mostrar eficiente também no estudo de outros aspectos de um sistema tutor inteligente como a determinação da técnica mais apropriada de inferência de dados e tecnologias adequadas para a construção de um protótipo.

Todas as fases de planejamento, execução e conclusão da revisão sistemática apresentadas no estudo de caso do capítulo 5 são essenciais para a execução da metodologia, contudo, a etapa de definição do protocolo de revisão foi observada como a etapa mais importante de todo o processo, visto que é responsável por guiar toda a etapa de execução da revisão e consequentemente influencia nos resultados obtidos por meio do processo. Além disso, esta etapa influi consideravelmente no tempo de execução da revisão, pois quanto melhor definido for o protocolo mais provável é que menor será o tempo total gasto no processo de revisão.

Uma das contribuições do trabalho é a verificação da possibilidade de emprego da metodologia de revisão sistemática como um instrumento de apoio na definição de uma arquitetura de software básica para um sistema, uma vez que o uso da metodologia ainda não é uma prática comumente observada nos estudos da área computacional.

O protocolo de revisão definido nesta pesquisa é outra contribuição relevante, visto que pode ser facilmente replicado e adaptado em outros estudos da área computacional por seguir um fluxo de execução bem definido e ser modelado de acordo com as particularidades da área de engenharia de software. O único cuidado a se observar é a necessidade de desconsideração das características inerentes ao estudo de caso apresentado neste trabalho, no âmbito de sistemas tutores inteligentes.

6.1 Trabalhos Futuros

O trabalho futuro nesta pesquisa pode incluir a expansão do processo de revisão para a definição de outros aspectos de um sistema tutor inteligente como a técnica de inferência de dados mais apropriada para o domínio de programação e tipo de feedback mais benéfico para o aprendizado de um estudante. É importante lembrar que a revisão executada neste trabalho já contém uma base sólida para continuação uma vez que já avalia os aspectos acima de forma quantitativa. Ou seja, uma expansão envolveria somente as etapas de extração dos dados relevantes e síntese.

Além disso, uma expansão poderia incluir também o desenvolvimento de um protótipo de um sistema tutor inteligente contendo todas as características importantes observadas através do processo de revisão, como também a realização de um experimento com grupos de estudantes a fim de verificar a eficiência do sistema tutor sugerido para o aprendizado de conceitos de programação.

Referências

- ABU-NASER, S. et al. An intelligent tutoring system for learning java objects. *International Journal of Artificial Intelligence and Applications (IJIA)*, v. 2, n. 2, 2011. Disponível em: <https://www.researchgate.net/profile/Samy_Abu_Naser/publication/267418395_An_Intelligent_Tutoring_System_for_Learning_Java_Objects/links/551a4f270cf26cbb81a2bea4.pdf>. Acesso em: 27 nov. 2016. Citado 8 vezes nas páginas 6, 7, 41, 61, 74, 76, 79 e 80.
- ARORA, N.; SAINI, J. R. A literature review on recent advances in neuro-fuzzy applications. *International Journal of Advanced Networking & Applications*, 2014. Disponível em: <https://www.researchgate.net/profile/Jatinderkumar_Saini/publication/281965280_A_Literature_Review_on_Recent_Advances_in_Neuro-Fuzzy_Applications/links/55ffd8e208aec948c4f9bc5c.pdf>. Acesso em: 27 nov. 2016. Citado na página 31.
- BOSCH, J. Software architecture: The next step. In: SPRINGER. *European Workshop on Software Architecture*. 2004. p. 194–199. Disponível em: <https://www.researchgate.net/profile/Jan_Bosch/publication/221105839_Software_Architecture_The_Next_Step/links/02e7e51b6ef86b844b000000.pdf>. Acesso em: 27 nov. 2016. Citado na página 16.
- BRUSILOVSKY, P.; SCHWARZ, E. W.; WEBER, G. Elm-art: An intelligent tutoring system on world wide web. In: FRASSON, C.; GAUTHIER, G.; LESGOLD, A. M. (Ed.). *Intelligent Tutoring Systems*. Springer, 1996. (Lecture Notes in Computer Science, v. 1086), p. 261–269. ISBN 3-540-61327-7. Disponível em: <https://www.researchgate.net/profile/Gerhard_Weber3/publication/221413291_ELM-ART_An_intelligent_tutoring_system_on_world_wide_web/links/0a85e5373ea8b460b4000000.pdf>. Acesso em: 27 nov. 2016. Citado na página 23.
- BUTZ, C. J.; HUA, S.; MAGUIRE, R. B. A web-based bayesian intelligent tutoring system for computer programming. *Web Intelligence and Agent Systems: An International Journal*, IOS Press, v. 4, n. 1, p. 77–97, 2006. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.3611&rep=rep1&type=pdf>>. Acesso em: 27 nov. 2016. Citado 12 vezes nas páginas 6, 7, 24, 25, 36, 41, 56, 57, 66, 70, 71 e 79.
- CRUZ, V. M. *Diseño e Implementación de planificadores instruccionales en sistemas tutoriales inteligentes mediante el uso combinado de metodologías borrosa y multiagente*. Tese (Doutorado) — Universidad de La Laguna, 2007. Disponível em: <<https://dialnet.unirioja.es/servlet/tesis?codigo=19459>>. Acesso em: 27 nov. 2016. Citado 5 vezes nas páginas 25, 27, 30, 31 e 32.
- DORÇA, F. A. *Um Sistema Inteligente Multiagente para Educação a Distância Apoiado em Web*. Tese (Doutorado) — Universidade Federal de Uberlândia, 2004. Disponível em: <http://www.facom.ufu.br/~fabiano/publicacoes_arquivos/dissertacao.pdf>. Acesso em: 27 nov. 2016. Citado 2 vezes nas páginas 24 e 26.
- EL-KHOULY, M.; FAR, B. H.; KOONO, Z. Expert tutoring system for teaching computer programming languages. *Expert Systems with Applications*, Elsevier, v. 18, n. 1, p. 27–32, 2000. Disponível em: <https://www.researchgate.net/profile/Mahmoud_Mohamed12/publication/222545795_Expert_tutoring_system_for_teaching_computer_programming_languages/links/53dc54b40cf216e4210c09af.pdf>. Acesso em: 27 nov. 2016. Citado 7 vezes nas páginas 6, 36, 41, 52, 53, 79 e 80.

- EMURIAN, H. et al. Learning java: A programmed instruction approach using applets. *Computers in Human Behavior*, Elsevier, v. 16, n. 4, p. 395–422, 2000. Disponível em: <https://www.researchgate.net/publication/223910631_Learning_Java_A_programmed_instruction_approach_using_Applets>. Acesso em: 27 nov. 2016. Citado 3 vezes nas páginas 41, 57 e 79.
- EMURIAN, H. H. A web-based tutor for java (tm): Evidence of meaningful learning. *International Journal of Distance Education Technologies*, IGI Global, v. 4, n. 2, p. 10, 2006. Disponível em: <<http://dl.acm.org/citation.cfm?id=1083464>>. Acesso em: 27 nov. 2016. Citado 6 vezes nas páginas 7, 41, 57, 71, 72 e 79.
- FEINBERG, E. A.; SHWARTZ, A. *Handbook of Markov decision processes: methods and applications*. Springer Science & Business Media, 2002. v. 40. Disponível em: <<https://books.google.com.br/books?hl=pt-BR&lr=&id=TpwKCAAAQBAJ&oi=fnd&pg=PA21&ots=AoUx1G86oG&sig=hQWM0otBHe9oY-RceVSs7Mteews#v=onepage&q&f=false>>. Acesso em: 27 nov. 2016. Citado na página 29.
- GARLAN, D. Software architecture: a travelogue. In: ACM. *Proceedings of the on Future of Software Engineering*. 2014. p. 29–39. Disponível em: <<http://www.cs.cmu.edu/~garlan/publications/DG14.pdf>>. Acesso em: 27 nov. 2016. Citado na página 14.
- GERDES, A. *Ask-Elle: a Haskell Tutor*. Tese (Doutorado) — Universiteit Utrecht, 2012. Disponível em: <<http://www.botkes.nl/wp-content/uploads/HaskellTutor.pdf>>. Acesso em: 27 nov. 2016. Citado 7 vezes nas páginas 6, 7, 41, 62, 74, 76 e 79.
- GORTON, I. *Essential software architecture*. Springer Science & Business Media, 2006. Disponível em: <<http://sites.google.com/site/nelsonjuniorbr2/EssentialSoftwareArchitecture.pdf>>. Acesso em: 27 nov. 2016. Citado 2 vezes nas páginas 14 e 15.
- HADJI, H. B.; CHOI, H.-J.; JEMNI, M. Pedagogically-driven courseware content generation for intelligent tutoring systems. *International Journal of Fuzzy Logic and Intelligent Systems*, Korean Institute of Intelligent Systems, v. 12, n. 1, p. 77–85, 2012. Disponível em: <https://www.researchgate.net/profile/Mohamed_Jemni/publication/264023577_Pedagogically-Driven_Courseware_Content_Generation_for_Intelligent_Tutoring_Systems/links/548db9930cf2d1800d841dd6.pdf>. Acesso em: 27 nov. 2016. Citado na página 26.
- HOLLAND, J. *A Constraint-Based ITS for the Java Programming Language*. Tese (Doutorado) — University of Canterbury, 2009. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.151.4812&rep=rep1&type=pdf>>. Acesso em: 27 nov. 2016. Citado 8 vezes nas páginas 6, 7, 41, 58, 60, 71, 74 e 79.
- HOOSHYAR, D. et al. A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers. *Journal of Computer Assisted Learning*, Wiley Online Library, v. 31, n. 4, p. 345–361, 2015. Disponível em: <https://www.researchgate.net/profile/Moslem_Yousefi2/publication/275652566_A_flowchart-based_intelligent_tutoring_system_for_improving_problem-solving_skills_of_novice_programmers_Flowchart-based_Intelligent_Tutoring_System/links/5542e2ad0cf234bdb21a2eca.pdf>. Acesso em: 27 nov. 2016. Citado na página 26.
- HOOSHYAR, D. et al. A solution-based intelligent tutoring system integrated with an online game-based formative assessment: development and evaluation. *Educational Technology*

Research and Development, Springer, p. 1–22, 2016. Disponível em: <https://www.researchgate.net/profile/Moslem_Yousefi2/publication/294288569_A_solution-based_intelligent_tutoring_system_integrated_with_an_online_game-based_formative_assessment_development_and_evaluation/links/56c1bbf308aeedba056724cd.pdf>. Acesso em: 27 nov. 2016. Citado 6 vezes nas páginas 6, 7, 41, 66, 78 e 79.

HUANG, M.-L. et al. Usage of case-based reasoning, neural network and adaptive neuro-fuzzy inference system classification techniques in breast cancer dataset classification diagnosis. *Journal of medical systems*, Springer, v. 36, n. 2, p. 407–414, 2012. Disponível em: <<https://ir.nctu.edu.tw/bitstream/11536/16050/1/000303825500006.pdf>>. Acesso em: 27 nov. 2016. Citado na página 27.

JEREMIĆ, Z.; JOVANOVIĆ, J.; GAŠEVIĆ, D. Evaluating an intelligent tutoring system for design patterns: The depths experience. *Journal of Educational Technology & Society*, JSTOR, v. 12, n. 2, p. 111–130, 2009. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.634.3600&rep=rep1&type=pdf>>. Acesso em: 27 nov. 2016. Citado 2 vezes nas páginas 25 e 26.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007. Disponível em: <<https://community.dur.ac.uk/ebse/resources/guidelines/Systematic-reviews-5-8.pdf>>. Acesso em: 27 nov. 2016. Citado 2 vezes nas páginas 18 e 19.

KRUCHTEN, P. B. The 4+ 1 view model of architecture. *IEEE software*, IEEE, v. 12, n. 6, p. 42–50, 1995. Disponível em: <<http://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6310/readings/gt-sad-kruchten-paper.pdf>>. Acesso em: 27 nov. 2016. Citado 3 vezes nas páginas 6, 15 e 16.

KUMAR, A. N. Intelligent tutoring systems: 6th international conference, its 2002 biarritz, france and san sebastian, spain, june 2–7, 2002 proceedings. In: _____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. cap. Model-Based Reasoning for Domain Modeling in a Web-Based Intelligent Tutoring System to Help Students Learn to Debug C++ Programs, p. 792–801. ISBN 978-3-540-47987-1. Disponível em: <https://www.researchgate.net/profile/Amruth_Kumar2/publication/221413997_Model-Based_Reasoning_for_Domain_Modeling_in_a_Web-Based_Intelligent_Tutoring_System_to_Help_Students_Learn_to_Debug_C_Programs/links/00b4951d4aba435b26000000.pdf>. Acesso em: 27 nov. 2016. Citado 5 vezes nas páginas 6, 41, 52, 53 e 79.

LANE, H. C.; VANLEHN, K. A dialogue-based tutoring system for beginning programming. In: *FLAIRS Conference*. [s.n.], 2004. p. 449–454. Disponível em: <<https://pdfs.semanticscholar.org/123f/64cad1d6a4627405176d22fb8487da6889c1.pdf>>. Acesso em: 27 nov. 2016. Citado 5 vezes nas páginas 7, 41, 53, 68 e 79.

LEE, C.; BABA, M. S. The intelligent web-based tutoring system using the c++ standard template library. *Malaysian Online Journal of Instructional Technology (MOJIT)*, Malaysian Educational Technology Association, v. 2, n. 3, p. 34–42, 2005. Disponível em: <https://www.researchgate.net/profile/Mohd_Sapiyan_Baba/publication/233988713_The_Intelligent_Web-Based_Tutoring_System_using_the_C_Standard_Template_Library/links/00b49532ef75785a21000000.pdf>. Acesso em: 27 nov. 2016. Citado 8 vezes nas páginas 6, 7, 41, 55, 56, 69, 70 e 79.

- LUNA, J. E. O. *Algoritmos EM para aprendizagem de redes Bayesianas a partir de dados incompletos*. Tese (Doutorado) — Universidade Federal de Mato Grosso do Sul, 2004. Disponível em: <https://www.researchgate.net/publication/242449477_Algoritmos_EM_para_Aprendizagem_de_Redes_Bayesianas_a_partir_de_Dados_Incompletos>. Acesso em: 27 nov. 2016. Citado na página 32.
- MAJOR, L.; KYRIACOU, T.; BRERETON, O. P. Systematic literature review: teaching novices programming using robots. *IET software*, IET, v. 6, n. 6, p. 502–513, 2012. Disponível em: <https://www.researchgate.net/profile/Louis_Major/publication/260649955_Systematic_literature_review_teaching_novices_programming_using_robots/links/55d43aea08aec1b0429f4a28.pdf>. Acesso em: 27 nov. 2016. Citado na página 12.
- MISHRA, K.; MISHRA, R. B. An intelligent tutoring system for c++. In: *Electronics and Information Engineering (ICEIE), 2010 International Conference On*. [s.n.], 2010. v. 2, p. V2–454–V2–458. Disponível em: <https://www.researchgate.net/publication/241172318_An_intelligent_tutoring_system_for_C>. Acesso em: 27 nov. 2016. Citado na página 41.
- NASER, S. Evaluating the effectiveness of the cpp-tutor an intelligent tutoring system for students learning to program in c++. *Journal of Applied Sciences Research*, v. 5, n. 1, p. 109–114, 2009. Disponível em: <https://www.researchgate.net/profile/Samy_Abu_Naser/publication/265991384_Evaluating_the_Effectiveness_of_the_CPP-Tutor_an_Intelligent_Tutoring_System_for_Students_Learning_to_Program_in_C/links/551a4f290cf244e9a4585944.pdf>. Acesso em: 27 nov. 2016. Citado 7 vezes nas páginas 7, 41, 58, 71, 73, 79 e 80.
- NASER, S. S. A. Developing an intelligent tutoring system for students learning to program in c++. *Information technology journal*, v. 7, n. 7, p. 1055–1060, 2008. Disponível em: <https://www.researchgate.net/profile/Samy_Abu_Naser/publication/26557161_Developing_an_Intelligent_Tutoring_System_For_Students_Learning_To_Program_in_C/links/551a4f2b0cf244e9a4585946.pdf>. Acesso em: 27 nov. 2016. Citado 6 vezes nas páginas 6, 41, 58, 59, 79 e 80.
- NORTHROP, L. The importance of software architecture. *Software Engineering Institute, Carnegie Mellon University*, 2003. Disponível em: <<http://sunset.usc.edu/GSAW/gsaw2003/s13/northrop.pdf>>. Acesso em: 27 nov. 2016. Citado na página 15.
- NWANA, H. S. Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, Springer, v. 4, n. 4, p. 251–277, 1990. Disponível em: <<https://pdfs.semanticscholar.org/ad07/945c80fb233c143085fb76f238681164a345.pdf>>. Acesso em: 27 nov. 2016. Citado na página 23.
- OKOLI, C.; SCHABRAM, K. A guide to conducting a systematic literature review of information systems research. *Sprouts Work. Pap. Inf. Syst*, v. 10, p. 26, 2010. Disponível em: <<https://pdfs.semanticscholar.org/31dc/753345d5230e421ea817dd7dcdd352e87ea2.pdf>>. Acesso em: 27 nov. 2016. Citado na página 18.
- ONTAÑÓN, S.; RAM, A. Case-based reasoning and user-generated artificial intelligence for real-time strategy games. In: *Artificial Intelligence for Computer Games*. Springer, 2011. p. 103–124. Disponível em: <https://www.researchgate.net/profile/Santiago_Ontanon2/publication/278717489_Case-Based_Reasoning_and_User-Generated_Artificial_Intelligence_for_Real-Time_Strategy_Games/links/558811fe08ae1dfa49d229c5.pdf>. Acesso em: 27 nov. 2016. Citado na página 27.

- PIÑERES, M. C.; JOSYULA, D. P.; BUILES, J. A. J. Multi-level pedagogical model for the personalization of pedagogical strategies in intelligent tutoring systems. *Dyna*, Universidad Nacional de Colombia, v. 82, n. 194, p. 185, 2015. Disponível em: <https://www.researchgate.net/profile/Jovani_Jimenez_Builes/publication/287800372_Multi-level_pedagogical_model_for_the_personalization_of_pedagogical_strategies_in_intelligent_tutoring_systems/links/5679560808ae70e8e4816950.pdf>. Acesso em: 27 nov. 2016. Citado na página 26.
- PRENTZAS, J.; HATZILYGEROUDIS, I. Categorizing approaches combining rule-based and case-based reasoning. *Expert Systems*, Wiley Online Library, v. 24, n. 2, p. 97–122, 2007. Disponível em: <https://www.researchgate.net/profile/Jim_Prentzas/publication/220658187_Categorizing_approaches_combining_rule-based_and_case-based_reasoning/links/0912f5073cb934f845000000.pdf>. Acesso em: 27 nov. 2016. Citado na página 27.
- PRENTZAS, J.; HATZILYGEROUDIS, I. Case-based reasoning integrations: Approaches and applications. *Case-based reasoning: processes, suitability and applications*. New York: Nova Science Publishers Inc, 2011. Disponível em: <https://www.researchgate.net/profile/Jim_Prentzas/publication/228446940_CASE-BASED_REASONING_INTEGRATIONS_APPROACHES_AND_APPLICATIONS/links/0912f5073cb91d4f66000000.pdf>. Acesso em: 27 nov. 2016. Citado na página 27.
- PUTERMAN, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2005. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=VvBjBAAQBAJ&oi=fnd&pg=PT9&ots=rqnvvLT_ON&sig=UKqkHSLbRIcW2vQFarNTf19I8Hw#v=onepage&q&f=false>. Acesso em: 27 nov. 2016. Citado na página 29.
- SAWICKI, S. et al. Characterising enterprise application integration solutions as discreteevent system. *IGI Global*, 2015. Disponível em: <<http://www.gca.unijui.edu.br/members/rzfrantz/publications/sawicki-igi-2015.pdf>>. Acesso em: 27 nov. 2016. Citado 2 vezes nas páginas 6 e 30.
- SEVARAC, Z. Neuro fuzzy reasoner for student modeling. In: IEEE. *null*. 2006. p. 740–744. Disponível em: <<https://pdfs.semanticscholar.org/75da/75123ad3fc8666e89f0c25f6e50bd80f1d2d.pdf>>. Acesso em: 27 nov. 2016. Citado na página 31.
- SHAFFER, S. C. Ludwig: An online programming tutoring and assessment system. *SIGCSE Bull.*, ACM, New York, NY, USA, v. 37, n. 2, p. 56–60, jun. 2005. ISSN 0097-8418. Disponível em: <<http://doi.acm.org/10.1145/1083431.1083464>>. Acesso em: 27 nov. 2016. Citado na página 41.
- SHAH, H.; KUMAR, A. N. A tutoring system for parameter passing in programming languages. In: ACM. *ACM SIGCSE Bulletin*. 2002. v. 34, n. 3, p. 170–174. Disponível em: <https://www.researchgate.net/profile/Amruth_Kumar2/publication/220807551_A_Tutoring_System_for_Parameter_Passing_in_Programming_Languages/links/02e7e51f2ec144858c000000.pdf>. Acesso em: 27 nov. 2016. Citado 4 vezes nas páginas 7, 36, 41 e 67.
- SOMMERVILLE, I. *Software Engineering*. 9th. ed. USA: Addison-Wesley Publishing Company, 2010. ISBN 0137035152, 9780137035151. Citado na página 14.

- SONG, J. et al. An intelligent tutoring system for introductory C language course. *Computers & Education*, Elsevier, v. 28, n. 2, p. 93–102, 1997. Disponível em: <<https://pdfs.semanticscholar.org/947f/453be7254b18d3920cb77264278c49974298.pdf>>. Acesso em: 27 nov. 2016. Citado 4 vezes nas páginas 6, 41, 51 e 79.
- SOTTILARE, R. A. *Fundamentals of Adaptive Intelligent Tutoring Systems for Self-Regulated Learning*. [S.l.], 2015. Disponível em: <https://www.researchgate.net/profile/Bob_Sottolare/publication/274081076_Fundamentals_of_Adaptive_Intelligent_Tutoring_Systems_for_Self-Regulated_Learning/links/551574460cf2d70ee2702bb1.pdf>. Acesso em: 27 nov. 2016. Citado na página 23.
- SYKES, E. R. Design, development and evaluation of the java intelligent tutoring system. *International Journal of Technology, Instruction, Cognition and Learning*, v. 7, n. 4, 2010. Disponível em: <https://www.researchgate.net/profile/Edward_Sykes2/publication/260790570_Design_Development_and_Evaluation_of_the_Java_Intelligent_Tutoring_System/links/0deec53b8b8b716286000000.pdf>. Acesso em: 27 nov. 2016. Citado 6 vezes nas páginas 7, 41, 60, 72, 75 e 79.
- VESIN, B. et al. Ontology-based architecture with recommendation strategy in java tutoring system. *Computer Science and Information Systems*, v. 10, n. 1, p. 237–261, 2013. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.715.700&rep=rep1&type=pdf>>. Acesso em: 27 nov. 2016. Citado 5 vezes nas páginas 6, 41, 62, 63 e 79.
- WERAGAMA, D.; REYE, J. Analysing student programs in the php intelligent tutoring system. *International Journal of Artificial Intelligence in Education*, Springer, v. 24, n. 2, p. 162–188, 2014. Disponível em: <<http://link.springer.com/article/10.1007/s40593-014-0014-z>>. Acesso em: 27 nov. 2016. Citado 3 vezes nas páginas 41, 64 e 79.
- WERAGAMA, D. S. *Intelligent tutoring system for learning PHP*. Tese (Doutorado) — Queensland University of Technology, 2013. Disponível em: <<http://eprints.qut.edu.au/63202/>>. Acesso em: 27 nov. 2016. Citado 9 vezes nas páginas 6, 7, 26, 41, 64, 65, 74, 77 e 79.
- WOOLF, B. P. *Building Intelligent Interactive Tutors: student-centered strategies for revolutionizing e-learning*. Burlington, MA: Morgan Kaufmann Publishers, 2009. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=MnrUj3J_VuEC&oi=fnd&pg=PP2&ots=lwzXj4mBli&sig=tKYjSW3BPvsug0n3FbVS2ujMSwA>. Acesso em: 27 nov. 2016. Citado 2 vezes nas páginas 24 e 25.
- XU, L.; SARRAFZADEH, A. Haskell-tutor: An intelligent tutoring system for haskell programming language. In: *Postgraduate Conference of the Institute of Information and Mathematical Sciences*. [s.n.], 2004. p. 108. Disponível em: <https://www.researchgate.net/profile/Abdolhossein_Sarrafzadeh/publication/239591295_Haskell-Tutor_An_Intelligent_Tutoring_System_for_Haskell_Programming_language/links/0f3175312463d2eb74000000.pdf>. Acesso em: 27 nov. 2016. Citado 8 vezes nas páginas 6, 7, 41, 54, 55, 69, 79 e 80.
- ZATARAIN-CABADA, R. et al. Java tutoring system with facial and text emotion recognition. *Intelligent Learning Environments*, p. 49, 2015. Disponível em: <<https://pdfs.semanticscholar.org/0534/775b9c350e42e90844781dbb3f0d0bdf63b.pdf>>. Acesso em: 27 nov. 2016. Citado 9 vezes nas páginas 6, 7, 41, 65, 66, 77, 78, 79 e 80.

Apêndices

APÊNDICE A – Formulário de Avaliação da Relevância da Publicação

Publicação:

O conteúdo da publicação é baseado em uma pesquisa de caráter científico?

- Há um estudo empírico ou relatório técnico com base na opinião de um especialista?

SIM

NÃO

Os objetivos da pesquisa são claramente definidos?

- Há uma declaração clara sobre os objetivos no resumo da publicação?
- São apresentadas as razões para a realização da pesquisa?
- O foco principal do trabalho envolve a utilização de sistemas tutores inteligentes no contexto de ensino de programação?

SIM

NÃO

Caso a resposta para alguma das questões anteriores seja NÃO, o procedimento de avaliação da relevância não será continuado e a publicação será considerada irrelevante para o estudo. A partir deste ponto haverá uma pontuação para as respostas das questões seguintes. A resposta SIM representa 10 pontos, a resposta PARCIAL constitui 5 pontos e a resposta NÃO é igual a 0 e não pontua. O resultado total será calculado através da soma dos pontos obtidos e será utilizado para geração de gráficos comparativos.

I. A arquitetura do sistema é claramente definida?

- São descritos todos os módulos que compõem o sistema?
- As responsabilidades de todos os módulos são devidamente apresentadas?
- O funcionamento do sistema é claramente descrito?

SIM

PARCIAL

NÃO

II. O mecanismo de inferência de dados do sistema é claramente descrito?

- Há uma descrição detalhada da técnica de inferência escolhida?
- A utilização da técnica escolhida é justificada?
- O contexto da pesquisa é levado em consideração na justificativa?

SIM

PARCIAL

NÃO

III. O processo de feedback do usuário é claramente apresentado?

- Há uma descrição detalhada de como o usuário receberá feedback do sistema?
- São apresentados exemplos de interação do sistema com o usuário?
- Os objetivos pedagógicos do feedback são discutidos?

SIM

PARCIAL

NÃO

IV. É apresentado um protótipo do sistema?

- O protótipo apresentado é funcional?
- O funcionamento do sistema pode ser evidenciado pelo protótipo?
- Há uma descrição detalhada do modo no qual o protótipo foi desenvolvido?

SIM

PARCIAL

NÃO

V. O protótipo, caso presente, possui suporte à personalização de preferências pelo usuário?

- Há como o usuário alterar as configurações de ensino do sistema de acordo com os seus objetivos e metas de aprendizagem pessoal?
- É possível que o usuário tenha impacto sobre a navegação durante o processo de aprendizagem?

SIM

PARCIAL

NÃO

VI. É realizado um experimento?

- O processo de realização do experimento é claramente definido?
- O critérios de escolha do grupo de pessoas é apresentado?
- A quantidade de pessoas selecionadas é satisfatória?

SIM

PARCIAL

NÃO

VII. Caso um experimento tenha sido realizado, os dados foram coletados de forma satisfatória?

- O processo de coleta é claramente descrito?
- Todas as unidades de medida foram especificadas?
- A seleção dos métodos de análise é justificada?

SIM

PARCIAL

NÃO

VIII. Caso os dados tenham sido coletados, a análise dos dados coletados é rigorosa?

- É fornecida uma descrição detalhada do processo de análise?
- A análise é imparcial e apresenta também dados contraditórios?

SIM

PARCIAL

NÃO

IX. Os resultados obtidos foram claramente documentados?

- Os resultados são devidamente descritos?
- Há uma argumentação clara sobre as conclusões obtidas com o estudo realizado?
- As conclusões são justificadas pelos resultados apresentados?

SIM

PARCIAL

NÃO

X. A contribuição do estudo é discutida?

- Há uma discussão sobre a contribuição do estudo para o conhecimento existente na área estudada?
- São identificadas áreas onde os resultados podem contribuir?

SIM

PARCIAL

NÃO

Pontos: