UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

JOÃO PAULO MEDEIROS

REVISÃO SISTEMÁTICA SOBRE MECANISMOS DE AVALIAÇÃO AUTOMÁTICA EM OBJETOS DE APRENDIZAGEM SOBRE PROGRAMAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO - PR

2015

JOÃO PAULO MEDEIROS

REVISÃO SISTEMÁTICA SOBRE MECANISMOS DE AVALIAÇÃO AUTOMÁTICA EM OBJETOS DE APRENDIZAGEM SOBRE PROGRAMAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Dr. Marco Aurélio Graciotto Silva

CAMPO MOURÃO - PR

2015

ATA DA DEFESA DO TABALHO DE CONCLUSÃO DE CURSO

Às **dezenove horas** do dia **seis de julho de dois mil e quinze** foi realizada no Sala E-103 a sessão pública da defesa do Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas para Internet do acadêmico **João Paulo Medeiros** com o título **REVISÃO SISTEMÁTICA SOBRE MECANISMOS DE AVALIAÇÃO AUTOMÁTICA EM OBJETOS DE APRENDIZAGEM SOBRE PROGRAMAÇÃO**. Estavam presentes, além do acadêmico, os membros da banca examinadora composta pelo professor **Dr. Marco Aurélio Graciotto Silva** (Orientador-Presidente), pelo professor **Dr. Radames Juliano Halmeman** e pela professora **Dra. Aretha Barbosa Alencar**. Inicialmente, o aluno fez a apresentação do seu trabalho, sendo, em seguida, arguido pela banca examinadora. Após as arguições, sem a presença do acadêmico, a banca examinadora o considerou **APROVADO** na disciplina de Trabalho de Conclusão de Curso e atribuiu, em consenso, a nota 7,5 (sete e meio). Este resultado foi comunicado ao acadêmico e aos presentes na sessão pública. A banca examinadora também comunicou ao acadêmico que este resultado fica condicionado à entrega da versão final dentro dos padrões e da documentação exigida pela UTFPR ao professor Responsável do TCC no prazo de **onze dias**. Em seguida foi encerrada a sessão e, para constar, foi lavrada a presente Ata que segue assinada pelos membros da banca examinadora, após lida e considerada conforme.

Observações:

Sem observações._____

_____

_____

_____

_____

Campo Mourão, 06 de julho de 2015.

A folha de aprovação assinada encontra-se na coordenação do curso

Dedico este trabalho a minha família pelo apoio e carinho. Ao meu orientador pela paciência nas orientações deste trabalho. E a todos professores, amigos, parentes que de alguma forma contribuíram com conhecimento, carinho e apoio, para que este sonho fosse realizado.

AGRADECIMENTOS

Quero agradecer em primeiro lugar a Deus, pois se cheguei até aqui foi porque ele permitiu. A bíblia diz que nenhuma folha cai da árvore se não for com a permissão de Deus.

Agradeço a todos os docentes que contribuíram na minha formação, em especial ao meu orientador Prof. Dr. Marco Aurélio Graciotto Silva, pela dedicação, confiança e paciência que teve comigo.

Aos meus pais José Medeiros e Maria Medeiros, meu irmão Alex Medeiros e minha namorada Karina Alves que sempre me apoiaram e compreenderam as minhas ausências quando precisei estudar, sempre me dando força para que eu pudesse alcançar este sonho na minha vida.

E a todos que direta ou indiretamente fizeram parte da minha formação e conquista, os meus sinceros agradecimentos.

RESUMO

MEDEIROS, João Paulo. REVISÃO SISTEMÁTICA SOBRE MECANISMOS DE AVALIAÇÃO AUTOMÁTICA EM OBJETOS DE APRENDIZAGEM SOBRE PROGRAMAÇÃO. 25 f. Trabalho de Conclusão de Curso – Curso Superior de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Campo Mourão - PR, 2015.

Em disciplinas de programação, os alunos devem estar sempre atentos e concentrados nos conceitos, sintaxe de código e na lógica dos algoritmos para que se possa tornar um bom programador. Uma grande quantidade de informações é apresentada aos alunos nas aulas, tornando assim muito importante a aplicação de exercícios práticos para melhorar o aprendizado em programação. Afinal, segundo Piaget, o aprendizado está fortemente ligado à experiência e prática de alguma determinada tarefa. Entretanto, professores das disciplinas de programação tem dificuldade em aplicar exercícios no final de cada tópico, devido ao tempo necessário para a correção e *feedback* ao aluno. Uma solução para este problema seria a utilização de sistema de avaliação automática. Sistemas de avaliação automática podem trazer benefícios ao aprendizado, pois os alunos podem resolver exercícios, submeter ao programa de avaliação automática e obter um *feedback* sobre o seu trabalho ao final de cada tarefa. De forma especial, deve-se observar a integração desses mecanismos e o conteúdo das disciplinas, constituindo objetos de aprendizagem sobre programação. O objetivo deste projeto foi realizar um mapeamento sistemático sobre mecanismos de avaliação automática e a aplicação desses em objetos de aprendizagem sobre programação. Para esta revisão sistemática foram coletados 17 trabalhos. Através da leitura destes trabalhos foram identificadas 10 ferramentas que já estão sendo utilizadas por cursos introdutórios de computação. Foi observado o objetivo das ferramentas de avaliação automática bem como suas características. Tais ferramentas possuem a característica de contribuição com o ensino e o desenvolvimento do aluno em disciplinas introdutórias de programação, visando também o ensino de conceitos primários de programação orientada a objetos e de teste de software no início do aprendizado em programação. As ferramentas em geral corrigem e retornam automaticamente texto de correção aos alunos e, em alguns casos, ainda verificam a originalidade dos códigos procurando plágio entre as respostas dos alunos. Pode-se também identificar trabalhos com características de incentivar o aluno a realizar correções colaborativas e até mesmo se auto avaliar em seus trabalhos. Foram verificados resultados positivos com a integração destes mecanismos em objetos de aprendizagem. Tais mecanismos acrescentam na didática das aulas, contribuindo no aprendizado dos alunos de programação.

Palavras-chave: Avaliação automática, Objetos de aprendizagem, Programação, Mecanismos de avaliação, Revisão Sistemática.

ABSTRACT

MEDEIROS, João Paulo. . 25 f. Trabalho de Conclusão de Curso – Curso Superior de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Campo Mourão - PR, 2015.

In programming courses, students should be attentive and focused on the concepts, code syntax, and logic of algorithms in order to become good programmers. A great deal of information is presented to students in classroom, thus making very important the application of practical exercises to improve the learning experience in computing. After all, according to Piaget, learning is strongly linked to experience and practice. However, instructors of programming disciplines faces difficulties to apply exercises at the end of each topic due to the time required for correction and feedback to the student. One solution to this problem would be to use automatic assessment tools. Automatic evaluation systems can bring benefits to learning because students can solve exercises, undergo the automatic evaluation program and get feedback on their work at the end of each task. Thus, special attention must be taken on the integration of these mechanisms and content of disciplines, providing proper learning objects about programming. The objective of this project was to conduct a systematic mapping of automatic evaluation mechanisms and the application of them in learning objects about programming. For this systematic review were collected 17 studies. By reading them, we identified 10 tools that are already being used for introductory courses in computing. It was observed the purpose of automated evaluation tools and its features. Such tools contributes to the teaching and development of students in introductory programming courses, also aiming at teaching basic concepts of object-oriented programming and software testing. Most of the tools evaluates and provides automatic grades to students, and in some cases even verify the originality of code is assessed, looking for plagiarism among students' responses. We could also identify studies that spurs student to perform collaborative corrections and even to self assess their work. Overall, we identified positive results with the integration of these mechanisms in learning objects. Such mechanisms improve the didactic of classes, contributing to the effectiveness of learning of programming.

Keywords: Automatic evaluation, learning objects, programming, evaluation mechanisms, Systematic Review.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

RS	Revisão Sistemática
UTFPR	Universidade Tecnológica Federal do Paraná
EAD	Ensino a Distância
SBIE	Simpósio Brasileiro de Informática na Educação
FIE	*Frontiers in Education Conference*
CSEET	*Conference on Software Engineering Education and Training*
SIGCSE	*Technical Symposium on Computer Science Education*
YAP	*Yet Another Plague*
EPGY	*Education Program for Gifted Youth*
LMS	*Learning Management System*

SUMÁRIO

1 INTRODUÇÃO

Em disciplinas de programação os alunos devem estar sempre atentos e concentrados nos conceitos, sintaxe de código e na lógica dos algoritmos para que se possam tornar bons programadores. Uma grande quantidade de informações é apresentada aos alunos nas aulas, tornando assim importante a aplicação de exercícios práticos para melhorar o aprendizado em programação. Segundo Piaget (2002), o aprendizado está fortemente ligado a experiência e prática de alguma determinada tarefa. Porém nem sempre é possível aplicar a quantidade de exercícios necessária para fixar bem o conteúdo das aulas e com isso ainda existem dificuldades no ensino introdutório de computação.

Partindo deste princípio, sabemos que uma boa base de programação é essencial para o melhor desenvolvimento dos alunos de computação. Conforme Gomes et al. (2008), o mínimo que se espera dos alunos de programação é que eles sejam capazes de resolver problemas simples em tarefas de programação, mas na realidade os níveis de reprovação nessa área estão altos.

Segundo Lahtinen et al. (2005), o desenvolvimento insatisfatório de alunos nos cursos de programação está relacionado ao não entendimento dos conceitos iniciais e fundamentais de programação. Portanto os alunos devem evoluir sequencialmente sem perder nenhuma fase do ensino de programação. Além dessas fases importantes do ensino, Edwards (2004) diz que, desde o começo da aprendizagem em algoritmos, os alunos devem ser ensinados a se auto avaliar aprendendo a corrigir seus próprios erros e realizar os testes de cada exercício.

Para ajudar a amenizar este problema, algumas abordagens como métodos de avaliação automática e objetos de aprendizagem vem sendo utilizadas para melhorar a qualidade dos recursos educacionais.

Objetos de aprendizagem é definido segundo Wiley (2000), como elementos de ensino instrucionais construído com qualquer tipo de mídias digitais, e que possam ser reutilizados em cenários diferentes dentro do ensino, e preferencialmente de forma automática. Já Fabre et al. (2003) define objetos de aprendizagem de forma geral e menos detalhada como qualquer tipo

de recurso que contribua para o ensino, e que possa ser reutilizado em contextos diferentes.

Em disciplinas introdutórias de computação, objetos de aprendizagem tais como ferramentas de avaliação automática, vídeos, imagens, sites, jogos ou rede social de aprendizagem, podem ser utilizados como ferramentas pedagógicas, trazendo benefícios ao ensino. O uso destes objetos de aprendizagem em sala de aula é visto como base de obtenção de conhecimento, ampliando o campo didático das aulas e provendo formas diferentes de interação com os alunos.

Neste trabalho são apresentadas algumas ferramentas de avaliação automática e a utilização das mesmas em objetos de aprendizagem. Estes recursos têm contribuído e mostrado resultados positivos contribuindo com o aprendizado em disciplinas introdutórias de programação. Por exemplo, Verdú et al. (2012) apresentam que, o uso de um objeto de aprendizagem e da ferramenta EduJudge, um programa utilizado para avaliar automaticamente e retornar *feedbacks* automáticos e imediatos nos exercícios de programação dos alunos, mostra que alunos utilizando o sistema, obtiveram melhor nota no exame do que os que não utilizaram. Além de obter o *feedback* automático, isso faz com que o aluno seja motivado a se auto avaliar-se em suas correções, procurando encontrar a solução correta do problema apresentado.

Assim como a questão pedagógica, estes objetos de aprendizagens utilizados em computação também auxiliam os professores no tempo e esforço. As correções dos exercícios e as avaliações são feitas automaticamente, retornando ao aluno um *feedback* automático sobre a correção dos exercícios.

De acordo com o contexto deste trabalho, foi realizada uma revisão sistemática sobre mecanismos de avaliação automática em objetos de aprendizagem sobre programação. A proposta é conhecer quais são os métodos de avaliação automática de programação utilizados em disciplinas introdutórias de computação.

O restante deste trabalho organiza-se da seguinte forma. O Capítulo 2 aborda conceitos de revisão sistemática e seus elementos segundo Biolchini et al. (2005). No Capítulo 3 é apresentado o protocolo desta revisão sistemática tal como suas subseções: questão de pesquisa, caracterização da questão de pesquisa, critérios para seleção de fontes, seleção de estudos, análise de qualidade, extração de dados, análise de estudos e considerações finais. Por fim, o Capítulo 4 fala da conclusão do trabalho.

2 REVISÃO SISTEMÁTICA

Revisão sistemática (RS) é um método de pesquisa para identificar, avaliar e interpretar trabalhos relacionados a um tema (KITCHENHAM, 2004). Esse tipo de trabalho integra dados relevantes coletados de estudos separados que, quando avaliados juntos, podem proporcionar resultados conflitantes ou coincidentes e muitas vezes produzem novos rumos de investigação. Kitchenham (2004) defende que uma boa revisão sistemática deve conter alguns requisitos importantes que são indispensáveis e devem andar juntos: transparência, replicabilidade e auditabilidade. A omissão desses requisitos pode comprometer o bom desenvolvimento da pesquisa. Elementos estes que são cruciais à RS incluem também detalhes dos procedimentos na seleção de estudos (FERRARI; MALDONADO, 2008).

Para desenvolver uma revisão sistemática, um protocolo com alguns passos metodológicos deve ser seguido em torno de um plano de foco central. No protocolo são definidos os passos do desenvolvimento da revisão sistemática, como formulação da questão, seleção de fontes, seleção de estudos, extração de informação, resultados e conclusões. O protocolo define o percurso da revisão sistemática até encontrar uma resposta para a questão problema. Segundo Biolchini et al. (2005), uma revisão sistemática pode ser dividida em três módulos, mostrados na Figura 1.

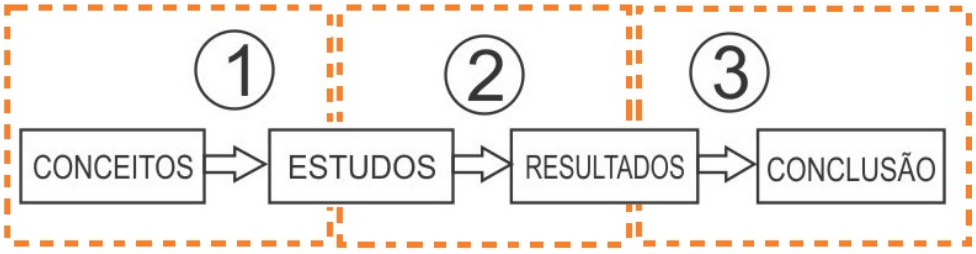

Figura 1: Módulos Revisão Sistemática (BIOLCHINI et al., 2005)

No primeiro módulo deve ser feito um estudo dos conceitos relacionados e uma pesquisa em busca de materiais disponíveis que possam fornecer informações relevantes ao tema da pesquisa. No segundo módulo, todos os trabalhos pesquisados e coletados nas pesquisas do primeiro módulo são estudados e comparados entre si. Quando isso é feito o pesquisador pode

analisar as medições e os resultados de cada trabalho, podendo muitas vezes levar a descoberta de novos tipos de provas. O terceiro módulo começa a partir dos resultados do segundo módulo. Por meio de um processo de análise, são extraídos novos arranjos de dados para as conclusões, que implicam na aquisição de novos conhecimentos sobre o tema, servindo de base à algumas tomadas de decisão relacionadas a ele (BIOLCHINI et al., 2005).

De maneira mais detalhada, Biolchini et al. (2005) explica em alguns passos uma revisão sistemática: Formulação da Questão, Seleção de Fontes, Seleção de Estudos, Análise de qualidade de estudos, Extração de Informação, Resultados e Conclusões. Conforme apresentado na Figura 1, estes passos podem ser organizados em três módulos. O primeiro módulo é iniciado à partir dos conceitos, englobando a formulação da questão, seleção de fontes e a seleção de estudos. Neste módulo é apresentado o tema, assunto do trabalho e a seleção dos estudos primários (estudos que podem produzir boas evidências sobre o tema do trabalho). A análise de qualidade de estudos e a extração de informações estão relacionadas ao segundo módulo e por fim, os resultados e conclusões começam e terminam no terceiro módulo. Todos esses passos são discutidos nas subseções seguintes.

2.1 FORMULAÇÃO DA QUESTÃO

Na formulação da questão o pesquisador aponta os objetivos da revisão sistemática. Estes objetivos devem ser claros e bem definidos, estabelecendo-se qual é a questão que deverá ser respondida no final da revisão sistemática. Outro item que deve ser pensado cuidadosamente é na qualidade da questão e amplitude, permitindo a definição da sintaxe das expressões de busca com uma atenção maior para que possa realizar pesquisa por bons trabalhos que ajudarão a responder a questão da revisão sistemática. Outros itens que compõem a qualidade da questão e amplitude são: Problema, Intervenção, Controle, Efeito, Medida de Resultado, População e Aplicação. A descrição de cada item é comentada a seguir.

- **Problema:** o problema é a meta da revisão sistemática, o que o pesquisador pretende descobrir. Por exemplo, neste trabalho pretendem-se saber quais são os mecanismos de avaliação automática utilizados em objetos de aprendizagem sobre programação.
- **População:** a população será o grupo populacional que será observado pela intervenção como por exemplo, os alunos das disciplinas inicias de programação da UTFPR.
- **Intervenção:** trata-se do que o pesquisador deseja observar no contexto da sua revisão sistemática. Por exemplo, neste trabalho será observado o uso de mecanismos de avaliação automática no ensino de programação.

- **Controle:** elemento utilizado para comparar o resultado.
- **Efeito:** são os tipos de resultados esperados no final da revisão sistemática.
- **Medida de Resultado:** medidas utilizadas para medir os resultados esperados.
- **Aplicação:** áreas que irão se beneficiar a partir do resultado da pesquisa, tipos de profissionais, papéis.

Neste caso, partir desta definição da amplitude, são obtidas as palavras chaves e sinônimos para seleção de fontes e estudos. Depois de concluída a formulação da questão, procede-se a definição de seleção de fontes.

2.2 SELEÇÃO DE FONTES

A seleção de fontes trata do estabelecimento de critérios para selecionar fontes de estudo relevantes à revisão sistemática. Segundo Biolchini et al. (2005), na execução desta seleção deve-se considerar alguns itens como: critérios para seleção de fontes, idioma de estudo e identificação de fontes.

Os critérios de seleção de fontes devem ser definidos para selecionar boas fontes de pesquisa para a revisão sistemática, como por exemplo: bases que indexam artigos das principais conferências, periódicos e análise de citações *(Snowballing)*. O pesquisador precisa saber quais são as características que tornam essas fontes de pesquisas relevantes para a realização da revisão sistemática.

Definido os critérios de seleção de fontes, outro item importante é o idioma de estudo. O pesquisador deverá definir o idioma de pesquisa de forma que isso não dificulte o bom entendimento dos estudos selecionados.

Na identificação de fontes, o objetivo é selecionar fontes para a execução das buscas e recuperação de artigos. As seleções destas fontes podem ser feitas através de motores de busca, bases de dados ou manual. Frases e palavras chaves devem ser relacionados com o tema para realizar boas pesquisas de trabalhos relacionados.

Considerando os itens que compõem a seleção de estudos, é possível utilizar várias fontes distintas, usando estratégias de buscas combinando, por exemplo, uso de bibliotecas digitais e o uso dos próprios estudos para selecionar fontes, empregando, com eles, análise de citações *(snowballing e foward balling)* para selecionar outros estudos.

2.3 SELEÇÃO DE ESTUDOS

Durante a seleção de estudos, definem-se os critérios de seleção de estudos e avaliação, ou seja, a forma como os estudos serão escolhidos. Também deve ser definidos critérios de exclusão de estudos, motivos por qual um trabalho não será incluso na revisão sistemática. Isto deve ser feito porque em uma pesquisa realizada em um motor de busca pode retorna um grande resultado de artigos que não respondem a questão da revisão sistemática. Isso acontece porque uma palavra chave de uma pesquisa pode ter significados diferentes em contextos diferentes e pode ser chave para outros tipos de trabalhos, por isso torna muito importante definir os critérios de inclusão e exclusão de trabalhos.

Na realização da etapa de controle quando o autor está montando sua *baseline*, os critérios de inclusão e exclusão já podem ser executados, incluindo e excluindo trabalhos não relevantes para a revisão sistemática.

2.4 ANÁLISE DE QUALIDADE DE ESTUDOS

Em análise de qualidade de estudos, identificam-se tipos de estudos que serão considerados para a RS. Cada tipo de estudo produz evidências com certa qualidade para a RS. Os estudos podem ser classificados em: experimentos controlados, estudo de caso, levantamento (survey) ou outros.

2.5 EXTRAÇÃO DE INFORMAÇÃO

A partir dos estudos primários, um protocolo de extração de informações relevantes deve ser descrito. Um padrão de coleta de informações deve ser utilizado para que todos possam ser comparados depois. Esse padrão de coleta pode variar de acordo com as necessidades de cada pesquisa. À partir dessa extração de informação podemos ter dois tipos de dados: resultados objetivos e resultados subjetivos.

- Resultados Objetivos: os resultados objetivos são aqueles que são extraídos diretamente dos estudos selecionados.
- Resultados Subjetivos: esse tipo de resultado é aquele que não é extraído diretamente de estudos selecionados. Ele pode ser coletado de duas maneiras distintas: entrando em contato com os autores dos trabalhos para tirar dúvidas e colher mais informações ou os revisores buscam aumentar suas próprias conclusões após os estudos dos trabalhos.

2.6 RESULTADOS E CONCLUSÕES

Nesta seção os resultados dos estudos selecionados devem responder as questões inicialmente formuladas para esta revisão sistemática. Tais resultados são obtidos a partir da síntese das evidências dos estudos selecionados. Inicialmente um mapeamento sistemático deve ser feito para obter a visão geral, definindo os tipos de estudos que serão abordados, locais de publicação, bases indexadas e tipos de resultados encontrados. Em seguida através de uma análise justa nos trabalhos selecionados, é feita uma revisão da leitura que visa avaliar, interpretar e identificar trabalhos relevantes para a RS. Após obter parâmetros finais sobre o observado através do processo de experimentação ou desenvolvimento, é feita a conclusão da revisão sistemática.

2.7 CONSIDERAÇÕES FINAIS

Revisões sistemáticas são um método para síntese de resultados de estudos selecionados. A definição do protocolo reduz o viés da aplicação e contribui para a qualidade dos resultados finais. Desta forma, a revisão sistemática demonstra-se apropriada para a identificação de mecanismos de avaliação automática de programas em disciplinas introdutórias de programação e suas características, bem como seu emprego em objetos de aprendizagem e os resultados desta interação.

3 PROTOCOLO DA REVISÃO SISTEMÁTICA

Este trabalho tem como objetivo realizar uma RS sobre mecanismos de avaliação automática de programação em disciplinas introdutórias de programação. Tais mecanismos vêm sendo utilizados por docentes em cursos de programação, tanto presencial como EAD (ensino a distância). A importância desta RS está em conhecer os diferentes mecanismos de avaliação automática junto de suas características, permitindo o uso e integração desses em objetos de aprendizagem.

3.1 QUESTÃO DE PESQUISA

Este trabalho relata uma pesquisa por mecanismos de avaliação automática sobre programação e sua utilização em objeto de aprendizagem, identificando quais são os mecanismos de avaliação automática utilizados. E como questão secundária, identificar as vantagens do ponto de vista pedagógico em utilizar esse mecanismo. As questões deste trabalho são:

- **Primária:** quais são os métodos de avaliação automática utilizados em objetos de aprendizagem sobre programação?
- **Secundária:** quais são os benefícios da utilização de mecanismos de avaliação automática no aprendizado sobre programação?

3.2 CARACTERIZAÇÃO DA QUESTÃO DE PESQUISA

Para a realização desta RS, foi utilizado como população, disciplinas introdutórias de programação. Não foram consideradas as disciplinas intermediárias, avançadas ou qualquer outra disciplina que não seja de programação. A intervenção desta RS está focada em mecanismos de avaliação automática no ensino de programação e sua utilização em objetos de aprendizagem. Os mecanismos esperados são ferramentas de avaliação automática que empreguem ou não teste de *software*, verificação de qualidade de código, tal como ProgTest, JaBUTi,

Poke-Tool, VETesting e outros. Espera-se encontrar diferentes ferramentas, identificar seus diferentes métodos e abordagens e o quanto elas contribuem no ensino de disciplinas introdutórias de programação.

Neste trabalho não será utilizado controle como citado entre os elementos que compõe a qualidade e amplitude da formulação da questão, na Seção 2.1 do Capítulo 2. Esta revisão sistemática possui uma natureza de mapeamento sistemático, e nesse tipo de estudo, não se utiliza controle desta forma. Para este trabalho foram selecionados estudos relevantes ao tema desta revisão sistemática para fins de controle da pesquisa mas não dos estudos selecionados.

3.3 CRITÉRIOS PARA SELEÇÃO DE FONTES

Após consultar alguns especialistas, entende-se que tais conferências de informática são importantes para este trabalho: SBIE (Simpósio Brasileiro de Informática na Educação), FIE *(Frontiers in Education Conference)*, CSEET *(Conference on Software Engineering Education and Training)*, SIGCSE *(Technical Symposium on Computer Science Education)* ou qualquer outro mecanismos de buscas que indexem essas bases listadas acima. Em relação aos critérios, somente foram aceitos trabalhos nas línguas portuguesa e inglesa: português porque nos possibilita a inclusão de vários trabalhos brasileiros com informações relevantes, e inglês por ter vários trabalhos relacionados a essa área.

Considerando esse critério, foram escolhidas como fontes para buscas iniciais as bibliotecas digitais: ACM Digital Library, IEEE Explorer e SCOPUS. Para realizar as buscas iniciais foram criados expressões de busca. Essas expressões foram criadas por meio de termos e sinônimos obtidos através da leitura de trabalhos relacionados. As palavras devem ser comuns e caracterizar bem o tema da revisão sistemática.

Com base no grupo populacional e no assunto que o pesquisador deseja observar para esta revisão sistemática, os sinônimos obtidos a partir da caracterização da questão de pesquisa foram: *programming, evaluation, evaluated, learning, e-learning, object, automated, automatic e programs*. Estas palavras podem ser comuns a outros tipos de trabalhos em computação, porém quando colocadas juntas criando termos mais específicos, caracterizam bem o tema desta revisão sistemática. Estas palavras também aparecem com frequência formando termos comuns como: *learning programming, automatic evaluation, e-learning, automated evaluate, evaluate of programs*.

Na pesquisa por estudos relacionados a está revisão sistemática foram utilizadas duas variações de expressão de busca, sendo elas:

1. *("automated evaluation"OR "automatic evaluation") AND "programming".*
2. *"learning object"AND ("automated evaluation"OR "automatic evaluation") AND "programming".*

Observando as duas expressões, nota-se que elas são equivalentes em questão de palavras e termos. Porém na segunda expressão utiliza-se o termo *"learning object"*, que no português fica (objetos de aprendizagem). Esse termo foi utilizado porque foi realizada uma pesquisa por ferramentas de avaliação automática bem como o emprego destas ferramentas em objetos de aprendizagem sobre programação.

3.4 SELEÇÃO DE ESTUDOS

Os estudos retornados das buscas foram submetidos aos critérios de inclusão e exclusão para selecionar trabalhos relevantes. Todos os critérios são definidos com base nas regras que melhor se encaixam nesta RS. Tais critérios são definidos a seguir.

Como critérios de inclusão foram definidos que no presente trabalho serão incluídos trabalhos escritos nas línguas inglesa ou portuguesa, trabalhos que utilizam mecanismos de avaliação automática em disciplinas introdutórias de programação e trabalhos de objetos de aprendizagem relacionados a programação.

Em contra partida, critérios de exclusão neste trabalho foi definido que, se houver um ou todos os critérios de inclusão, não serão aceitos trabalhos que estiverem algum dos critérios de exclusão como: trabalhos curtos com menos de quatro páginas de texto. Os trabalhos devem ter um número de página que possa de fato caracterizar que houve um estudo sobre o tema da RS. Trabalhos de mecanismos de avaliação automática que não sejam de programação e que somente realizam avaliação automática de similaridade de código serão excluídos desta RS.

As buscas foram feitas com as expressões 1 e 2 em cada uma das bibliotecas digitais IEEE Explore, ACM Digital library e SCOPUS. A Tabela 1 mostra os resultados das buscas.

Biblioteca	Resultados Expressão 1	Resultados Expressão 2	Selecionados
IEEE	114	94	7
ACM	38	1	6
SCOPUS	93	6	4
TOTAL	245	101	17

Tabela 1: Resultados da pesquisa primaria de estudos

Executando a expressão 1 *("automated evaluation"OR "automatic evaluation") AND "programming"*, o retorno de resultados são maiores, sendo que com estes termos na expressão contém muitos trabalhos relacionados, porém trazem muitos estudos irrelevantes que não passam pelos critérios de inclusão ou que não são específicos sobre programação. Tal como trabalhos sobre avaliação automática de processos de engenharia de software ou trabalhos que somente avaliam similaridade de código ou texto, focados em detecção de plágio.

Executando a expressão 2 *"learning object"AND ("automated evaluation"OR "automatic evaluation") AND "programming"*, a busca feita com o termo em inglês *"learning object"* nas três bibliotecas os resultados são bem menores. O uso destes termos entre aspas faz com que sejam pesquisados trabalhos com exatamente estes termos contido neles e nessa ordem. Neste caso tem-se a vantagem de remover muitos trabalhos irrelevantes, porém isto não quer dizer que é o melhor caso. Na ACM Digital Library neste caso não retornou trabalhos relevantes como na execução da primeira expressão. Ao todo foram selecionados 17 trabalhos e na biblioteca IEEE Explorer a expressão 2 retornou menos resultados do que a expressão 1. Está biblioteca foi a que mais retornou trabalhos relevantes.

Aplicando estes critérios, foram selecionados 17 trabalhos relevantes sobre avaliação automática de programação. A Tabela 2 mostra quais foram os estudos selecionados e as fontes.

Fontes	Estudo selecionado	Citação
ACM	A distributed system for learning programming on-line	(VERDÚ et al., 2012)
ACM	Automatic evaluation of correctness and originality of source codes	(POHUBA et al., 2014)
ACM	Individualized Exercises for Self-Assessment of Programming Knowledge: An Evaluation of QuizPACK	(BRUSILOVSKY; SOSNOVSKY, 2005)
ACM	Monitoring System for the Semi-Automatic Evaluation of Programs Written During Classroom Lectures	(KOGURE et al., 2013)
ACM	Software Verification and Graph Similarity for Automated Evaluation of Students' Assignments	(VUJOŠEVIĆ-JANIČIĆ et al., 2013)
ACM	Orchestration of E-Learning Services for Automatic Evaluation of Programming Exercises	(QUEIRÓS; LEAL, 2012)
IEEE	Automated Evaluation of Programming Assignments	(KAUSHAL; SINGH, 2012)
IEEE	Web-Based System for Automatic Evaluation of Java Algorithms	(PINTO, 2013)
IEEE	Automated evaluation methods with attention to individual differences-a study of a computer-based course in C#	(ROSENTHAL et al., 2002)
IEEE	Automatic evaluation of algorithms over the Internet	(LUCAS et al., 2000)
IEEE	Automatic evaluation of correctness and originality of source codes	(POHUBA et al., 2014)
IEEE	Automatic evaluation of adaptive algorithms over the Internet	(LUCAS et al., 2000)
IEEE	PROGTEST: An Environment for the Submission and Evaluation of Programming Assignments based on Testing Activities	(SOUZA et al., 2011)
SCOPUS	Aprendizagem de iniciantes em algoritmos e programação: foco nas competências de auto avaliação	(SIROTHEAU et al., 2011)
SCOPUS	Testador Automático e Método de Avaliação de Programas em Java	(NUNES; LISBÔA, 2004)
SCOPUS	AnimalSense: Combining Automated Exercise Evaluations with Algorithm Animations	(RÖSSLING et al., 2011)
SCOPUS	Review of Recent Systems for Automatic Assessment of Programming Assignments	(IHANTOLA et al., 2010)

Tabela 2: Estudos Selecionados para revisão sistemática

Com base nos estudos selecionados foi observada a publicação dos estudos em anos cronológicos verificando a partir de quando esse tipo de trabalho começou a ser mais desenvol-

vido. A primeira apareceu em 2000 e até 2010 tinha uma média de uma publicação por ano, já em 2011 até 2014 o número de publicações encontrado aumentou em mais de 100% em relação aos anos anteriores. Então podemos perceber que o tema tem ganhado força de 2011 em diante. A Figura 2 mostra esses dados.

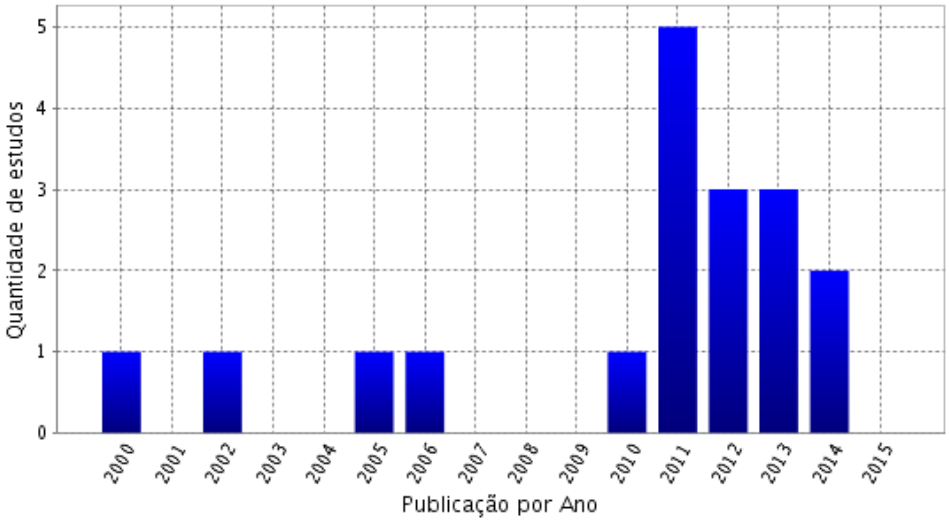

Figura 2: Publicações em tempo sequencial

3.5 ANÁLISE DE QUALIDADE

Neste trabalho foram coletados o total de 17 estudos sobre mecanismos de avaliação automática de programas, 6 trabalhos coletados na ACM (38%), 7 coletados na IEEE (43%) e 4 na SCOPUS (25%). Estes estudos apresentam ferramentas de avaliação automática de programa escrito por alunos, com o propósito de contribuir com o ensino de programação e são considerados estudos de caso. Entretanto pode se identificar mais de um tipo de estudo na leitura desses trabalhos, em 3 dos estudos selecionados os alunos são inquiridos com relação ao uso das ferramentas, qualificando as mesmas. Em outros 3 trabalhos foram identificados os tipos de estudos quantitativos por conter resultados que são quantificados por meio das análises dos dados e 1 trabalho que são os dois tipos. A Figura 3 com diagrama de Venn mostra a intersecção entre dois tipos de estudos encontrado, quantitativo e qualitativo.

Para ficar mais claro o entendimento do diagrama o mesmo foi divido em dois diagramas. Segue a Figura 4 demonstrativa da intersecção entre os tipos de estudo experimental, estudo de caso e ferramentas. Os estudos apresentam uma ferramenta ou um método de avaliação automática, porém ainda nem todos estão consolidados com um nome, ou não foi identificado o mesmo, alguns ainda estão em fase de teste e desenvolvimento por isso eles não entraram na quantidade de ferramentas no Diagrama 4.

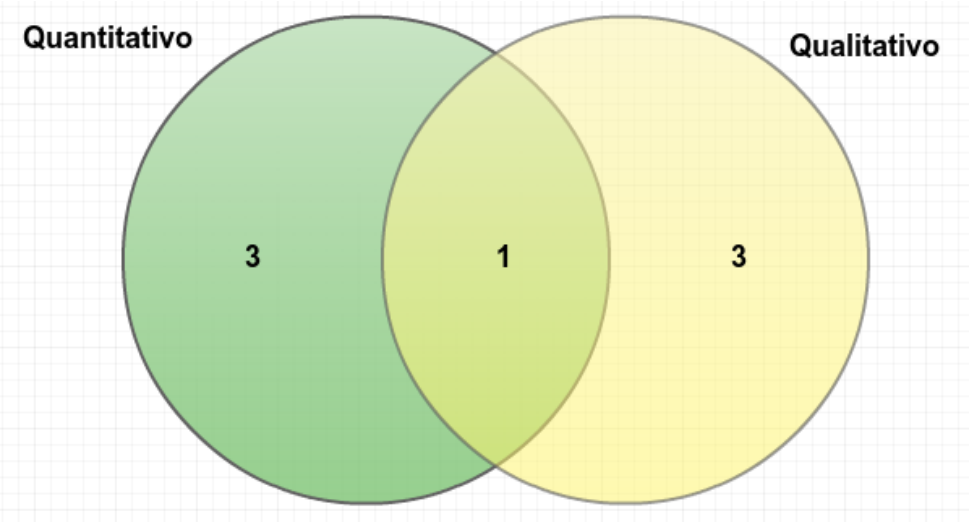

Figura 3: Diagrama de Venn Qualitativo e Quantitativo

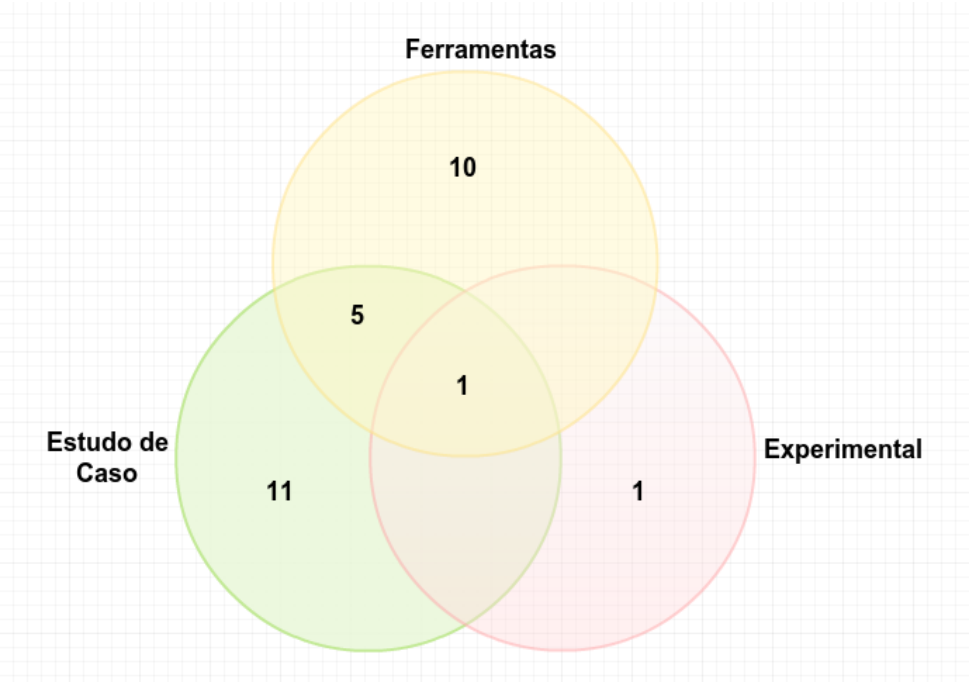

Figura 4: Diagrama de Venn Estudo de caso, Ferramentas e Experimental

Ao todo quatro tipos de estudos foram identificados: estudo de caso, qualitativo, quantitativo e experimental. Nesta figura foram identificados dez trabalhos que mostram ferramentas de avaliação automáticas e entre elas cinco dos trabalhos também foram identificados como estudo de caso. Onze dos 17 trabalhos eram estudos de caso, um experimental e um que apresentava uma ferramenta e era estudo de caso e experimental.

3.6 EXTRAÇÃO DE DADOS E ANÁLISE DOS ESTUDOS

A Tabela 3 mostra quais foram os dados investigados nos estudos selecionados para esta revisão sistemática. Esta Tabela é composta por onze itens que serão definidos com uma descrição explicando qual é o dado específico que pretende saber de cada item:

Item	Descrição
1	Qual é o nome da ferramenta de avaliação automática?
2	Linguagem de programação utilizada?
3	Utilização da ferramenta em sala de aula?
4	Objetivo na utilização da ferramenta?
5	Estudo é sobre as atribuições de programação?
6	Avaliação de testes?
7	Estudante cria e submete casos de teste?
8	Estudante tem acesso a programas desenvolvidos por outros estudantes?
9	Programas dos alunos são avaliados com o do professor?
10	Na avaliação automática os programas são classificados com pontuação?
11	Classificação considera o desempenho da execução do programa?

Tabela 3: Dados analisados

- **Qual é o nome da ferramenta de avaliação automática:** aqui somente descrever o nome da ferramenta.
- **Linguagem utilizada:** Dizer qual é a linguagem aceita pela ferramenta para realizar a avaliação automática. Ex: Java, PHP, Pascal, C++.
- **Utilização da ferramenta em sala de aula:** descrever se a ferramenta é utilizada pelos próprios alunos para submeter e receber *feedbacks* e se ela é utilizada para avaliar testes, algoritmos ou os dois.
- **Objetivo na utilização da ferramenta:** saber quais são os objetivos da ferramenta, que podem ser de apoio ao ensino de programação, incentivar o aluno a se auto avaliar ou fornecer avaliações colaborativas para outros alunos.
- **Estudo é sobre as atribuições de programação:** o estudo precisa ser sobre programação.
- **Avaliação de casos de testes:** saber se a ferramenta avalia casos de teste de software.
- **Estudante cria e submete casos de teste:** saber se os teste são criados pelos alunos ou se são fornecidos pelo professor.

- **Estudante tem acesso a programas desenvolvidos por outros estudantes:** saber se o estudante tem acesso a programas desenvolvido por outros alunos.
- **Programas dos alunos são avaliados com o do professor:** saber se os exercícios de programação dos alunos são comparados na avaliação junto com um exemplo do professor para realizar a correção.
- **Na avaliação automática os programas são classificados com pontuação:** saber se os programas são classificados com uma nota em relação a correção.
- **Classificação considera o desempenho da execução do programa:** saber se a classificação utiliza o desempenho como critério para atribuição de nota final.

Os estudos foram selecionados com ênfase em trabalhos de avaliação automática de códigos de programação. Entre os trabalhos abordados, pode se observar que 100% dos estudos selecionados são estudos de avaliação automática que visam apoiar o aprendizado de programação em salas de aula e apoiar os professores nas correções automáticas dos programas. A originalidade de código e a correção dos programas é realizado por 31% dos estudos, evidenciando quando houvesse plágio e 69% dos trabalhos não avaliava a originalidade de códigos. Outro dado encontrado é que 56% dos estudos também avaliavam casos de testes de software.

Na leitura dos artigos destacou-se a questão pedagógica e qual seria o objetivo da utilização dessa ferramenta de avaliação automática e como ela era usada em sala de aula. No trabalho de Verdú et al. (2012) tinham como objetivo fornecer novas estratégias de aprendizagem que motivassem os estudantes a aprenderem programação, assim como Sirotheau et al. (2011), tinham como objetivo auxiliar o ensino inicial de programação e incentivar o aluno a se auto avaliar e fornecer avaliações colaborativas. Neste trabalho outros alunos podem sugerir correções nos trabalhos dos colegas de classe. No trabalho de Corte et al. (2006), o objetivo é apoiar o ensino de teste de software com fundamentos de programação, defendendo que o bom entendimento dos conceitos fundamentais de programação, somado com desenvolvimento de casos de teste, são pontos cruciais para ser um bom programador.

Além disso, 75% das ferramentas são para avaliação automática de códigos Java, e outros percentuais são mostrados no Gráfico 5.

Uma das vantagens que estes mecanismos proporcionam ao aluno é o *feedback* automático na hora em que suas dúvidas ocorrem. Pensando neste tipo de problema, Wang e Wong (2008) criaram um projeto chamado EduJudge que visa proporcionar um "juiz online", ou seja, um programa utilizado para fornecer *feedbacks* automáticos nas correções dos exercícios de

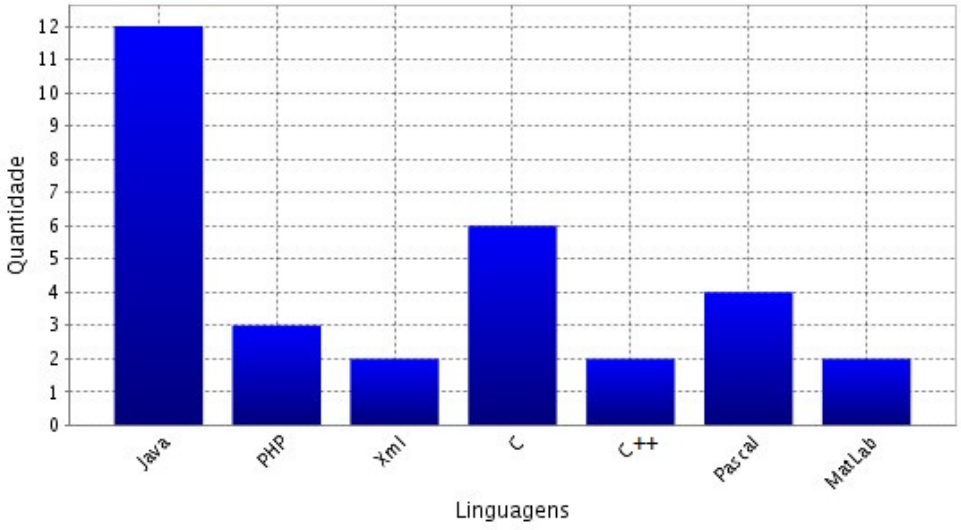

Figura 5: Linguagens avaliadas

programação nas linguagens C++ e Java. Esta ferramenta está incorporada na plataforma Moodle, utilizada online. Outro sistema utilizado para avaliação automática integrado ao Moodle é o YAP (POHUBA et al., 2014). Este sistema verifica a implementação correta dos algoritmos, dando ênfase na originalidade dos códigos. Ele também detecta automaticamente quando houver plágios. Além disso, ele é capaz de avaliar exercícios de programação, assim como casos de testes, verificando a justeza dos códigos, retornando *feedbacks* sobre a correção dos códigos aos alunos. O sistema YAP realiza a avaliação nas seguintes linguagens de programação C, C++, Java, Pascal, Php e MatLab.

Existem sistemas que geram exercícios tal como o QuizPACK, trabalho este desenvolvido por Brusilovsky e Sosnovsky (2005). Ele é capaz de gerar exercícios parametrizados na linguagem C e avaliar automaticamente as respostas de cada aluno. O uso deste sistema aumentou significativamente o conhecimento semântico e o nível de conhecimento dos alunos em relação a programação (BRUSILOVSKY; SOSNOVSKY, 2005). Vários alunos relatam uma avaliação positiva sobre o QuizPACK.

Vujošević-Janičić et al. (2013) desenvolveram um trabalho de avaliação automática que realiza três tipos de avaliações diferentes: verificação de erros em código, controle de fluxo dos programas e medidor de similaridade em programas escrito pelos alunos. Este sistema possui um gráfico utilizado para mostrar o nível de similaridade entre programas. Tanto na avaliação dos programas como na avaliação de similaridade entre os programas, o sistema disponibiliza *feedbacks* compreensíveis para ajudar os alunos a melhorar a qualidade de seus programas. Os resultados foram bons em relação às estimativas de notas dos alunos que começaram a praticar este sistema. Sistema este que tem ajudado tanto alunos como professores. Para os professores

a avaliação automática é vantajosa na classificação e correção dos exercícios, deixando os professores com mais tempo para outras atividades. Já para os alunos tem-se a vantagem de obter um imediato *feedback* dos exercícios.

No trabalho de Queirós e Leal (2012) é apresentado um sistema de gerenciamento de exercícios de programação que utiliza outros sistemas heterogêneos, tais como motores de avaliação, repositórios de objetos de aprendizagem e ambientes de resolução de exercícios. Tais sistemas são heterogêneos, portanto a coordenação e integração de todos eles ficam mais complexas. Estas ferramentas são específicas para serem incorporados em plataformas de *e-Learning*. Existe um componente pivô para administrar todos os sistemas em conjunto. Neste cenário o professor seleciona um conjunto de exercícios e os alunos tentam resolver. Após a resolução os alunos submetem os exercícios, recebendo um *feedback* automático. A vantagem desta arquitetura de sistema utilizando outros sistemas heterogêneos é que pode ser reutilizada em outros contextos, não se limitando estritamente a avaliação automática de uma linguagem específica. Ela pode ser reutilizada em outras linguagens, tais como SQL, UML, HTML, JAVA e outras (QUEIRÓS; LEAL, 2012).

Kaushal e Singh (2012) apresentam uma ferramenta de avaliação automática de trabalhos de programação em nível de graduação, acessível através da Web. Tal sistema foi desenvolvido principalmente para detecção de plágio e avaliação automatizada de casos de testes. Os casos de testes podem ser gerados manualmente ou automaticamente. As correções feitas pelo sistema levam em consideração quatro parâmetros importantes que são integridade, regularidade, eficiência e precisão. A conclusão deste trabalho foi que os alunos melhoraram suas habilidades em relação aos quatros parâmetros em termos de regularidade, integridade, precisão e eficiência em programação.

Pinto (2013) desenvolveu um trabalho de avaliação automática de programas e casos de testes de software. Tal sistema foi dividido em quatro estágios para a avaliação automática de código fonte: I) código fonte é avaliado e um verificador de erro é especificado; II) código é testado no JUnit, com outros casos de testes estabelecidos pelos professores; III) conjunto de medições de engenharia de software é estabelecido para comparar os programas dos alunos com o do professor; IV) e por fim, no último estágio, um *feedback* automático é fornecido com base nos resultados dos estágios anteriores. O *feedback* final é muito importante para dizer aos alunos em quais áreas eles precisam se corrigir, melhorando seus conhecimentos em programação.

Pohuba et al. (2014) apresenta um sistema de verificação de algoritmos e correção de originalidade de código fonte utilizando API de plágio YAP. O sistema é integrado à plataforma Moodle e possui licença GPLv3, podendo ser utilizado por outras instituições. Neste sistema

existem algumas regras básicas para definir a pontuação final da avaliação automática. As regras são: se o aluno tem alguns erros de compilação, a pontuação é atribuída a zero; acima de duas advertências, a pontuação cai em 10%; nas linguagens C / C++, se as saídas forem semelhantes o peso é de 50%, caso contrário é de 90%; no caso de vazamento de memória o peso é diminuído em 40%.

Na Stanford University, Rosenthal et al. (2002) apresentaram o EPGY *(Education Program for Gifted Youth),* programa de educação para jovens superdotados. Essa iniciativa tem por finalidade disponibilizar um curso de programação em C. Neste programa um sistema é utilizado com o objetivo principal de testar a viabilidade de correção automática de algoritmos. Este software é composto por palestras de vídeos e também com questões e exercícios ao final de cada palestra. Para cada respostas que os alunos submetem aos exercícios do programa, um comentário automático é retornado de imediato ao aluno, especificando erros ou complementando respostas corretas. O software registra a localização atual para que na próxima aula o aluno possa continua de onde parou na aula anterior.

ProgTest é uma ferramenta de avaliação automática de programas desenvolvido por Souza et al. (2011). Ela é uma ferramenta Web de submissão de trabalhos práticos de programação, com base em casos de teste de software. Esta ferramenta tem sido utilizada como mecanismo de apoio no ensino de conceitos fundamentais em teste de software. Ela fornece suporte automatizado, avaliando programas e conjuntos de casos de testes feitos pelos alunos. A ProgTest requer uma atribuição de um oráculo para realizar a avaliação dos programas dos alunos. Esta atribuição é fornecida de duas maneiras, através da apresentação do conjunto de teste fornecido pelo instrutor ou selecionando uma atribuição da base de atribuições do oráculo no ProgTest. Mais especificamente, esta atribuição do oráculo são exercícios e casos de testes adequados já selecionados de livros introdutórios de programação, com o objetivo de facilitar o trabalho dos instrutores. Os testes são compilados e executados com JUnit. O ProgTest contém critérios de correção estabelecidos no programa que podem ser selecionados pelo instrutor ou não, podendo estabelecer pesos para estes critérios. Os testes e programas dos alunos também são comparados com o do instrutor. Após a avaliação, o ProgTest retorna um *feedback* adequado a correção dos exercícios de cada aluno.

Sirotheau et al. (2011) apresentam um trabalho focado nas competências de auto avaliação em trabalhos de programação dos alunos. Recursos incorporados ao JavaTool e a plataforma Moodle permitem avaliar trabalhos com foco em *feedbacks* colaborativos. O *feedback* é considerado muito importante na autorregulação da aprendizagem de programação. Entretanto, quando isso não ocorre ou é demorado para se obter um *feedback*, acaba prejudicando o

aprendizado do aluno (SCHUNK, 1989). A partir desta experiência os educadores perceberam que esta é uma forma de colocar os alunos na ativa, proporcionando uma correção colaborativa automática, ajudando-os a pensar e se auto avaliar em seus trabalhos.

No trabalho de Nunes e Lisbôa (2004) foi desenvolvido um método em Java que serve para avaliar classes, também escritas em Java. Tanto testes de software como programas são avaliados, identificando problemas com construtores ou métodos. Para as classes testadas é atribuída uma nota com base em alguns critérios de avaliação de forma que se alguma classe não compilar será atribuída a nota zero. Erros de retorno também serão contabilizados na nota final da avaliação.

Os trabalhos apresentados focam no desenvolvimento de objetos de aprendizagem junto com avaliação automática de programas escritos por alunos. Cada uma dessas ferramentas tem características peculiares, decorrente dos objetivos que os autores desejam chegar com o uso delas. A fim de avaliar as ferramentas encontradas para esta revisão sistemática, foram selecionadas 9 características com o intuito de identificar se as ferramentas possuem as mesmas. Na Tabela 5 são apresentadas as ferramentas com suas características nomeadas de C1 a C9. As características são classificadas com V (verdadeiro) quando a ferramenta possui esta característica, ou F (Falso) quando não possui. Cada característica é definida e detalhada na Tabela 4 a seguir:

Sigla	Descrição
C1	Avalia mais de um tipo de linguagem de programação?
C2	Esta ferramenta somente avalia teste de software?
C3	O objetivo desta ferramenta é contribuir com processo de ensino ou auxiliar os professores na aplicação e correção de exercícios?
C4	Estudo é sobre as atribuições de programação?
C5	O Estudante cria ou submete casos de teste?
C6	Estudante tem acesso a programas desenvolvidos por outros estudantes?
C7	Os programas dos alunos são avaliados com o do professor?
C8	Na avaliação automática os programas são classificados com pontuação?
C9	Classificação considera o desempenho do programa?

Tabela 4: Características

A partir da descrição das características apresentadas, a Tabela 5 ilustra as ferramentas que mais se encaixam nessas características. Pode se observar que todas as ferramentas tem em comum a responsabilidade de contribuir com o ensino de programação e auxiliar os professores com monitoramento, correção e *feedback* automático dos exercícios de programação. Também foi identificado que 60% das ferramentas foram desenvolvidas para avaliar mais de um tipo de linguagem de programação incluindo algoritmos e casos de teste de software. E 40% delas per-

mitem que os alunos criem e submetam casos de teste e os outros disponibilizam os exercícios através do programa. Alguns sistemas permitem que alunos tenham acesso aos programas de outros alunos. Aqui temos o EduJudge e o JavaTool que permitem os alunos terem acesso ao programa dos outros alunos e poderem colaborar com o trabalho de outros alunos. Na ferramenta EduJudge os autores pensaram nessa estratégia de colaboração entre os alunos visando motivá-los a avaliar e contribuir, como um desafio fácil, atraente e cooperar com o aprendizado de todos. Foi identificado que 30% das ferramentas usam uma métrica para avaliar os programas dos alunos com o do professor. Nesses casos o professor deixa uma resposta padrão dos exercícios que quando comparada com o dos alunos, o programa consiga avaliar redundância de código, desperdício de uso indevido de memória e número de linhas. Essas avaliações ajudam a classificar os programas com pontuação. Nesta revisão sistemática, todas as ferramentas retornam *feedback* das correções dos trabalhos. Duas ferramentas se destacaram por atender 70% dos requisitos, o ProgTest e EduJudge, mas em contra partida o QuizPack atende apenas 20% dos requisitos.

Ferramenta	c1	c2	c3	c4	c5	c6	c7	c8	c9
ProgTest	V	F	V	V	V	F	V	V	V
JavaTool	F	F	V	V	F	V	F	V	V
AEPA	V	F	V	V	V	F	F	V	V
EduJudge	V	F	V	V	V	V	F	V	V
Yap	V	F	V	V	F	F	F	V	V
QuizPACK	F	F	V	V	F	F	F	F	F
Sistema Monitoração	V	F	V	V	F	F	V	F	F
Acode System	V	F	V	V	F	F	V	V	V
EPGY	F	F	V	V	F	F	F	V	V
Algoval	F	F	V	V	V	F	F	V	V

Tabela 5: Ferramentas e Características

As conclusões destes trabalhos revelam que o uso de mecanismos de correções automáticas junto com os *feedbacks* automáticos fornecidos aos alunos de forma rápida, tem causado melhora no aprendizado dos alunos. O *feedback* proporciona ao aluno uma resposta rápida à correção dos exercícios. E com isso o aluno não perde tempo esperando o *feedback* do professor, e também pode se auto avaliar no desenvolvimento dos seus códigos.

3.7 CONSIDERAÇÕES FINAIS

A partir das análises dos estudos selecionados e das características observadas, pode-se conhecer algumas ferramentas de avaliação automática, bem como as vantagens de uso como ferramenta de apoio e como objetos de aprendizagem. Estas ferramentas podem ser integradas

a algum LMS-*Learning Management System* (Sistema de gestão de aprendizagem), que são plataformas acessíveis através da Internet. Estas práticas têm sido estudadas e utilizadas como formas de apoio pedagógico no ensino.

Neste trabalho foi observado que o uso destes mecanismos assim como a utilização deles em objetos de aprendizagem tem mostrado bons resultados nos alunos que começaram a utilizar estas ferramentas. O uso destas ferramentas de aprendizagem tem causado o efeito de colaboração e auto-avaliação nos alunos de programação. Em algumas ferramentas é possível que os alunos tenham acesso a programas de outros alunos, podendo contribuir com sugestões de correção. Os *feedbacks* automáticos contribuem com professores e alunos, agilizando no tempo gasto para corrigir o trabalho de todos os alunos e disponibilizando correção imediata nos trabalhos dos alunos.

4 CONCLUSÃO

Através da leitura dos estudos selecionados foram identificados os métodos de avaliação automática utilizados. Todos os trabalhos apresentam ferramentas que somente realizam avaliações automáticas ou avaliação automática e verificação de originalidade de códigos, procurando plágio entre os trabalhos escritos pelos alunos. Em 7 ferramentas ou 41% estão integradas ao LMS Moodle e estão sendo utilizadas em cursos de informática. Os estudos mostraram que os alunos que começaram a utilizar as ferramentas de avaliação automática em objetos de aprendizagem tem demonstrado notas finais mais altas do que os alunos que não utilizaram. De modo geral os objetos de aprendizagem contribuem com o aprendizado e ensino de formas diferentes, estimulando os alunos a realizar correções colaborativas, se auto avaliar nos seus exercícios e até mesmo deixando a ferramenta de avaliação automática de forma divertida e atraente, para que o aluno goste de programar como se estivesse se divertindo ou jogando.

Considerando os benefícios destas ferramentas, tanto para o professor quanto para os alunos, ela fica melhor ainda e traz mais benefícios quando empregada a uma plataforma LMS, pois isto remove qualquer tipo de incômodo com instalação de software local e, considerando que ela seja uma ferramenta baseada na Web, pode alcançar pessoas de qualquer lugar, desde que tenha um computador conectado à Internet. Mas em contrapartida existe uma falta de integração entre ferramentas de avaliação automáticas e objetos de aprendizagens LMS, pois tais sistemas precisam ser compatíveis com plataforma de execução e linguagem. A falta de estudos que analisam os efeitos da integração entre os mecanismos e LMS, visto como uma limitação. Desta forma, são necessárias pesquisas para tratar desta integração e como realizá-la adequadamente.

Esta revisão sistemática contém algumas limitações que são: busca por trabalhos em português, pois ainda existem poucos, expressões de busca que retornaram muitos trabalhos irrelevantes, avaliar mecanismos de avaliação automática em objetos de aprendizagem limitou bem os resultados da pesquisa.

Em trabalhos futuros pode ser feita uma revisão sistemática visando entender em deta-

lhes como são realizadas as avaliações automáticas na integra, observando os objetivos e estratégias pedagógicas de cada ferramenta, também verificar objetos de aprendizagem que podem ser reaproveitados em outras disciplinas dentro da computação.

REFERÊNCIAS

BIOLCHINI, J.; MIAN, P. G.; NATALI, A. C. C.; TRAVASSOS, G. H. **Systematic Review in Software Engineering**. Rio de Janeiro, RJ,, maio 2005. Disponível em: `http://alarcos.inf-cr.uclm.es/doc/MetoTecInfInf/Articulos/es67905.pdf`.

BRUSILOVSKY, P.; SOSNOVSKY, S. Individualized exercises for self-assessment of programming knowledge: An evaluation of quizpack. **Journal on Educational Resources in Computing (JERIC)**, ACM, v. 5, n. 3, p. 6, 2005.

CORTE, C. K. D.; BARBOSA, E. F.; MALDONADO, J. C. Ensino integrado de fundamentos de programaç ao e de teste de software. 2006.

EDWARDS, S. H. Using software testing to move students from trial-and-error to reflection-in-action. **SIGCSE Bull.**, ACM, New York, NY, USA, v. 36, n. 1, p. 26–30, mar. 2004. ISSN 0097-8418. Disponível em: `http://doi.acm.org/10.1145/1028174.971312`.

FABRE, M.-C. J.; TAMUSIUNAS, F.; TAROUCO, L. M. R. Reusabilidade de objetos educacionais. **RENOTE**, v. 1, n. 1, 2003.

FERRARI, F. C.; MALDONADO, J. C. Experimenting with a multi-iteration systematic review in software engineering. In: . [S.l.: s.n.], 2008. p. 1–10.

GOMES, A.; HENRIQUES, J.; MENDES, A. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. **Educação, Formação & Tecnologias-ISSN 1646-933X**, v. 1, n. 1, p. 93–103, 2008.

IHANTOLA, P.; AHONIEMI, T.; KARAVIRTA, V.; SEPPÄLÄ, O. Review of recent systems for automatic assessment of programming assignments. In: ACM. **Proceedings of the 10th Koli Calling International Conference on Computing Education Research**. [S.l.], 2010. p. 86–93.

KAUSHAL, R.; SINGH, A. Automated evaluation of programming assignments. In: IEEE. **Engineering Education: Innovative Practices and Future Trends (AICERA), 2012 IEEE International Conference on**. [S.l.], 2012. p. 1–5.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, p. 2004, 2004.

KOGURE, N.; RIKI, M.; KANAE, Y.; KOICHI, K.; ITOH, T. .; YUKIHIRO. Monitoring system for the semi-automatic evaluation of programs written during classroom lectures. ACM, 2013.

LAHTINEN, E.; ALA-MUTKA, K.; JÄRVINEN, H.-M. A study of the difficulties of novice programmers. In: ACM. **ACM SIGCSE Bulletin**. [S.l.], 2005. v. 37, n. 3, p. 14–18.

LUCAS, S.; BEATTIE, P.; COY, J. Automatic evaluation of adaptive algorithms over the internet. In: IEEE. **Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop**. [S.l.], 2000. v. 2, p. 886–895.

NUNES, I. O. de; LISBÔA, M. L. B. Testador automático e método de avaliação de programas em java. 2004.

PIAGET, J. **Judgement and reasoning in the child**. [S.l.]: Routledge, 2002.

PINTO, M. Web-based system for automatic evaluation of java algorithms. In: **EUROCON, 2013 IEEE**. [S.l.: s.n.], 2013. p. 2123–2128.

POHUBA, D.; DULIK, T.; JANKU, P. Automatic evaluation of correctness and originality of source codes. In: IEEE. **Microelectronics Education (EWME), 10th European Workshop on**. [S.l.], 2014. p. 49–52.

QUEIRÓS, R.; LEAL, J. P. Orchestration of e-learning services for automatic evaluation of programming exercises. **J. UCS**, v. 18, n. 11, p. 1454–1482, 2012.

ROSENTHAL, T.; SUPPES, P.; BEN-ZVI, N. **Automated evaluation methods with attention to individual differences-a study of a computer-based course in C**. [S.l.]: IEEE, 2002.

RÖSSLING, G.; MIHAYLOV, M.; SALTMARSH, J. Animalsense: combining automated exercise evaluations with algorithm animations. In: ACM. **Proceedings of the 16th annual joint conference on Innovation and technology in computer science education**. [S.l.], 2011. p. 298–302.

SCHUNK, D. H. Self-efficacy and cognitive skill learning. **Research on motivation in education**, v. 3, p. 13–44, 1989.

SIROTHEAU, S.; BRITO, S.; SILVA, A.; ELIASQUEVICI, M. K.; FAVERO, E. L.; TAVARES, O. Aprendizagem de iniciantes em algoritmos e programação: foco nas competências de autoavaliação. **Simpósio Brasileiro de Informática na Educação (SBIE 2011)**, v. 22, 2011.

SOUZA, D. M. de; MALDONADO, J. C.; BARBOSA, E. F. Progtest: An environment for the submission and evaluation of programming assignments based on testing activities. In: IEEE. **Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on**. [S.l.], 2011. p. 1–10.

VERDÚ, E.; REGUERAS, L. M.; VERDÚ, M. J.; LEAL, J. P.; CASTRO, J. P. de; QUEIRÓS, R. A distributed system for learning programming on-line. **Computers & Education**, Elsevier, v. 58, n. 1, p. 1–10, 2012.

VUJOŠEVIĆ-JANIČIĆ, M.; NIKOLIĆ, M.; TOŠIĆ, D.; KUNCAK, V. Software verification and graph similarity for automated evaluation of students' assignments. **Information and Software Technology**, Elsevier, v. 55, n. 6, p. 1004–1016, 2013.

WANG, F. L.; WONG, T.-L. Designing programming exercises with computer assisted instruction. In: **Hybrid Learning and Education**. [S.l.]: Springer, 2008. p. 283–293.

WILEY, D. A. **Learning Object Design and Sequencing Theory**. Tese (Doutorado) — Brigham Young University, Provo, Utah,, 2000.