# Desenvolvimento de Pacote Python para Migração entre Juízes Eletrônicos

Tiago de Souza Fernandes
*Depto. de Ciência da Computação*
*Universidade de Brasília*
tiagotsf2000@gmail.com

Guilherme N. Ramos
*Depto. de Ciência da Computação*
*Universidade de Brasília*
gnramos@unb.br

***Resumo*—Boas práticas no ensino de programação são essenciais para a consolidação de conteúdos por parte dos alunos. Uma abordagem atrativa e eficiente é a utilização de ferramentas de correção automática de código, conhecidos como juízes eletrônicos, pois além de agilizarem o processo de aprendizagem do aluno, permitem uma aplicação prática de conceitos, com *feedback* instantâneo, sem sobrecarregar o docente com correções manuais de códigos. Contudo, o processo de elaboração de questões ainda é manual e demorado, e existem muitas dificuldades em reaproveitar questões de diferentes plataformas, visto que cada juiz eletrônico trabalha com um formato próprio. Nesse contexto, este trabalho propõe a criação de um software para tradução de questões entre juízes eletrônicos, que busca contemplar três plataformas: CodeForces, BOCA e Moodle. Essa ferramenta facilita a importação e exportação de questões entre as plataformas, permitindo que sejam reaproveitadas sem os custos do processo manual de tradução dos diferentes formatos. O software se encontra disponível para uso no repositório Pypi, de pacotes Python, de forma gratuita.**

***Index Terms*—Juiz eletrônico, ferramenta de software, ambiente virtual de aprendizado, educação na ciência da computação.**

## I. Introdução

O advento da revolução tecnológica trouxe uma série de mudanças no funcionamento da sociedade, tornando a tecnologia uma ferramenta poderosa e indispensável para processos de diversas áreas [1]. A educação foi amplamente afetada pela tecnologia, o desenvolvimento da internet transformou a forma como a informação e o conhecimento são obtidos, tornando o processo de aprendizagem muito mais eficiente [2]. Nesse contexto, nasceram os ambientes virtuais de aprendizagem (AVA), plataformas online de educação que auxiliam docentes na disponibilização de conteúdos e questões para seus alunos, muito utilizadas em ensino remoto e a distância, bem como em disciplinas presenciais [3].

O Moodle é o principal e mais utilizado software de aprendizagem em ambiente virtual do mundo [4], oferecendo uma série de recursos para o desenvolvimento de atividades e materiais, além de contar com sistemas de avisos e avaliação dos alunos. A plataforma é amplamente utilizada[1] por diversos departamentos da Universidade de Brasília (UnB), como forma de extensão das disciplinas presenciais para o ambiente virtual.

O departamento de Ciência da Computação da UnB é responsável por ministrar as disciplinas iniciais dos cursos de computação, e conta com a plataforma do Moodle para isso. Contudo, o ensino de programação não é algo simples, e muitos alunos se sentem desmotivados nas disciplinas, corroborando para os altos índices de desistência nos cursos de computação e gerando desinteresse na área [5]. Esse problema está atrelado a dificuldade de se explicar programação apenas com a apresentação de conceitos, sem a presença de uma parte prática, abordagem utilizada por muitos docentes [6].

Nesse contexto, a correção automática de questões de programação disponível no Moodle, realizada por sistemas conhecidos como *juízes eletrônicos*, vem tomando espaço como uma solução eficaz para a implementação de práticas nas disciplinas da computação [7]. A correção automática de código é rápida, o que significa que o aluno pode errar e ser alertado imediatamente do erro para corrigi-lo, e é imparcial, logo, a correção não é sujeita a subjetividade de quem está corrigindo, proporcionando condições iguais de correção para todos os alunos [8]. A automação do processo também facilita a correção por parte do professor, trabalho esse que poderia se tornar mecânico e propício a erros se feito manualmente, liberando-o para se dedicar a outros aspectos da disciplina.

Apesar dos benefícios, o Moodle não possui muitas questões de programação prontas, pois o uso do serviço de correção automática de código disponível na plataforma é recente no departamento, e gerá-las requer tempo, pois é um processo criativo que envolve criar problemas que abordem bem os conteúdos desejados, e manual, pois é necessário redigir e formatar os textos, gerar testes e escolher exemplos.

Existem, contudo, questões de programação prontas em outras plataformas de programação que também oferecem o serviço de correção automática. A Universidade de Brasília possui um longo histórico de participação em eventos de programação competitiva[2], e existem duas plataformas muito utilizadas, tanto pelos alunos quanto pelos professores que ministram disciplinas na área, o CodeForces[3] e o BOCA[4]. Ambas as plataformas possuem algumas centenas de questões de programação, elaboradas pelos próprios docentes da Universidade ao longo dos anos, que poderiam ser utilizadas nas disciplinas de programação. Contudo, são plataformas voltadas

[1] http://aprender.unb.br/

[2] http://maratona.ime.usp.br/

[3] https://codeforces.com/

[4] https://www.ime.usp.br/~cassio/boca/

para atividades de programação competitiva e possuem poucos recursos que possam ser utilizados em ensino de programação. O Moodle, no entanto, dispõe de diversas ferramentas da própria plataforma integradas ao juiz eletrônico, e de acordo com uma pesquisa da Universidade Federal do ABC, o Moodle fornece um maior suporte ao desenvolvimento de estratégias metacognitivas quando comparado com outras plataformas analisadas no escopo do ensino de programação [9].

A Universidade de Brasília possui muita proximidade com a plataforma do Moodle, e os benefícios do uso de juízes eletrônicos nas disciplinas da computação são significativos. Contudo, atualmente não existe qualquer forma prática de utilizar das questões do CodeForces e do BOCA na plataforma do Moodle, algo interessante tendo em vista a existência de questões prontas nessas plataformas.

Nesse contexto, surge a seguinte pergunta: como reaproveitar questões entre essas plataformas? As três plataformas trabalham com formatos diferentes de questão, e a tradução manual é um trabalho mecânico que toma muito tempo. Com o intuito de oferecer uma solução eficiente para o problema, este trabalho propõe e implementa um software que automatiza o processo de tradução de questões do CodeForces e do BOCA para o Moodle, facilitando e agilizando o processo de exportação/importação de questões entre as plataformas quanto comparado com o processo manual.

Os assuntos abordados neste projeto estão organizados da seguinte forma: a Seção II possui uma descrição dos juízes eletrônicos e dos diferentes formatos de questão de programação, a Seção III contém um planejamento da arquitetura do software, a Seção IV descreve a implementação do software e dos processos de tradução, bem como problemas enfrentados, a Seção V mostra alguns resultados do uso do software, e por fim, a Seção VI apresenta conclusões alcançadas e futuras melhorias para o projeto.

## II. Juízes Eletrônicos

Os juízes eletrônicos são sistemas de correção automatizados que executam e testam uma solução, programa que resolve o problema descrito na questão, geralmente utilizando de um conjunto de casos de teste [10]. O funcionamento interno da maioria dos juízes eletrônicos é semelhante, o código submetido é executado com a entrada de dados conhecida, e a saída obtida é comparada ao resultado esperado; a solução é aceita caso sejam iguais. As soluções enviadas também devem respeitar os limites estabelecidos de tempo de execução e memória alocada, estarem restritas às linguagens de programação aceitas pelo juiz e respeitarem o limite de tamanho do arquivo enviado.

Apesar de serem ferramentas semelhantes, cada juiz eletrônico trabalha com seu próprio formato de questão, contendo diversas informações estruturadas de diferentes formas. A seguir são descritos os detalhes específicos dos juízes considerados neste projeto, que definem o *formato* de seus arquivos.

### A. Formato Polygon

O CodeForces é uma plataforma online que hospeda questões e competições voltadas para a programação competitiva, contando com um sistema de correção automática próprio e uma plataforma de criação e teste de questões chamada Polygon[5]. O Polygon estrutura as questões em um arquivo no formato `ZIP`, de compactação de arquivos, onde está armazenado um conjunto de diretórios e arquivos com informações da questão. Nos arquivos podem ser encontrados o texto (enunciados, imagens e exemplos), soluções, casos de teste, conteúdos abordados (*tags*), nome identificador da questão, entre outros dados e arquivos necessários para que a questão seja reconhecida pela plataforma.

Os arquivos também possuem um comparador de saída, programa feito para verificar, de forma inteligente, se a saída obtida pela solução do aluno é igual ao resultado esperado, e um gerador de casos de teste, programa feito pelo criador da questão para gerar os testes de forma aleatória.

### B. Formato CodeRunner

O Moodle conta com um plugin chamado CodeRunner[6], que permite a criação e uso de questões de programação com correção automática dentro da própria plataforma. O CodeRunner trabalha com questões no formato de arquivo `XML`, que estrutura as informações em elementos hierárquicos.

Neste arquivo também são encontrados o texto, casos de teste, solução, *tags* e, por ter um foco de integração com a plataforma de ensino, parâmetros referentes à avaliação da questão na plataforma, como "penalty", relativo penalidade sofrida na nota do aluno ao enviar uma solução errada, "all_or_nothing", que diz respeito a necessidade da solução ser aceita em todos os casos de teste para o aluno ser avaliado, e a linguagem de programação utilizada para aquela questão. As questões do CodeRunner são atreladas a uma única linguagem de programação, na qual a solução deve ser escrita.

### C. Formato BOCA

O BOCA é um software brasileiro, criado com a função de hospedar e controlar competições de programação aos moldes da Maratona de Programação da Sociedade Brasileira de Computação. O software trabalha com questões em um formato `ZIP`, assim como o Polygon, porém com um conjunto de diretórios diferentes, onde podem ser encontrados o texto, os casos de teste, os limites, entre outras informações. A estrutura é simples e semelhante a dos formatos anteriores, porém não possui uma formatação padrão para armazenamento dos enunciados, visto que a plataforma é voltada para maratonas, os enunciados normalmente são agrupados em um único arquivo de prova, com uma formatação à escolha do usuário.

### D. Síntese

Apesar das plataformas descritas trabalharem com formatos diferentes, percebe-se uma semelhança nas informações contidas nas questões, algo que pode auxiliar a implementação dos

[5]https://polygon.codeforces.com/

[6]http://coderunner.org.nz/

processos de tradução. Na seção seguinte será apresentado um planejamento da arquitetura do software e como a semelhança entre os formatos viabiliza o uso de uma abordagem mais simples para a implementação do programa.

## III. Projeto

A organização e o planejamento prévio de um software podem evitar esforços desnecessários em sua implementação, pois identificar e tratar possíveis problemas pode ser feito de forma mais simples quando a ideia ainda está sendo planejada, reduzindo a necessidade de reescrever código [11]. Para isso, é interessante entender bem o propósito do programa e ter em mente suas funcionalidades principais.

Este software possui o objetivo de traduzir diferentes formatos de questões de programação com correção automática. Esse objetivo pode ser alcançado a partir de duas funcionalidades: a de **leitura**, onde o arquivo da questão em um formato específico é aberto para se adquirir as informações necessárias, e a de **escrita**, onde um novo arquivo de questão em outro formato é gerado com as informações previamente obtidas no processo de leitura.

Para planejar a implementação dessas funcionalidades, é necessário entender quais são as informações manipuladas por esses processos. Apesar de distintos em estrutura, os diferentes formatos vistos armazenam informações similares a respeito da questão, que no geral possuem dois propósitos principais: o de **apresentar** o leitor à questão, como o texto, e o de **avaliar** uma solução enviada, como os casos de teste e os limites de tempo e memória. Essas duas ideias são a base para a maioria das questão de programação com correção automática e devem estar presentes para que a questão de fato exista.

### A. Tradução entre Juízes

A princípio, para que o software ofereça todas as traduções dos formatos de questão, devem ser implementados os processos de leitura e escrita para cada combinação possível entre formato de entrada (da questão a ser traduzida) e formato de saída (desejado pelo usuário). Essa abordagem se torna custosa a medida que o número de plataformas incorporadas pelo software aumenta, pois requer que muitos processos sejam implementados.

Uma abordagem mais simples é o uso de uma estrutura intermediária de informações entre os formatos de entrada e saída, onde os dados da questão são armazenados durante a tradução. Dessa forma, é possível incorporar uma nova plataforma apenas implementando processos de leitura e escrita que interagem a estrutura intermediária, não existindo a necessidade de implementações diretas desse formato para todos os outros formatos incorporados ao software. Contudo, para viabilizar essa abordagem, é necessário que as informações contidas nas questões sejam semelhantes, para que possam ser armazenadas em uma mesma estrutura de dados.

### B. Formato intermediário

Pensando nas ideias mencionadas acima, um formato intermediário de questão foi implementado, constituído de informações base abstraídas dos formatos de questão vistos, com o intuito de simplificar a implementação dos processos de tradução.

Para compor o formato intermediário, foram utilizadas as informações necessárias para **apresentar** e **avaliar** uma questão de programação com correção automática, são elas o **texto**, responsável por apresentar a questão ao leitor, os **casos de teste** e os **limites de tempo e memória**, responsáveis por avaliar uma solução enviada. Contudo, como o proposito do projeto é utilizar de questões para o ensino, foram incluídos também a **solução** esperada da questão, pois oferece ao aluno uma abordagem diferente e muitas vezes melhor implementada para comparação, auxiliando no seu aprendizado, as ***tags***, pois permite a indexação das questões por conteúdo, facilitando o acesso para elaboração de provas, e por simplicidade, o **identificador**. As informações utilizadas para compor a classe do formato intermediário podem ser vistas em cinza claro na Tabela I.

Tabela I: Comparativo entre formatos.

| | **Polygon** | **Coderunner** | **BOCA** |
|---|---|---|---|
| **Texto** | Sim | Sim | Sim |
| **Casos de teste** | Sim | Sim | Sim |
| **Limites de tempo e memória** | Sim | Sim | Sim |
| **Solução** | Sim | Sim | |
| **Tags** | Sim | Sim | |
| **Identificador** | Sim | | Sim |
| **Comparador de saída** | Sim | | Sim |
| **Gerador de casos teste** | Sim | | |
| **All_or_nothing** | | Sim | |
| **Penalty** | | Sim | |

Utilizando essa abordagem, cada formato de questão possui suas próprias funções de leitura e de escrita, que trabalham somente com o formato intermediário.

### C. Funções de leitura e escrita

As funções de leitura/escrita de cada um dos formatos realizam os mesmos processos: leem/escrevem os dados existentes no formato intermediário. Com o propósito de aproveitar dessa semelhança, uma abordagem de orientação a objeto e herança foi utilizada. A orientação a objeto é um paradigma de programação que estrutura dados e funções na forma de objetos, chamados classes, já a ideia de herança permite que classes herdem características de outras classes, chamadas classes abstratas, na intenção de reaproveitar comportamentos semelhantes. Essa abordagem permite uma padronização na implementação dos processos de leitura e escrita, evitando códigos repetitivo e tornando a incorporação de novos formatos simples e semelhante a de formatos já presentes.

Para estruturar essas funções, duas classes abstratas foram implementadas, uma para leitura e outra para escrita, possuindo um método principal, o método `read`, na de leitura, e o método `write`, na de escrita, e outros métodos abstratos para manipular cada um dos dados do formato intermediário. Nessa abordagem, o funcionamento do software gira em torno do uso dos dois métodos principais `read` e `write`, responsáveis por chamar todos os métodos abstratos. Com essas classes

abstratas é possível criar classes que herdem esses métodos e os implementem para os diferentes formatos de questão.

Dessa forma, a implementação de um processo de tradução se resume a implementação dos métodos `read` ou `write` e de seus respectivos métodos abstratos, que procuram a informação desejada no formato da questão para adicioná-la ao formato intermediário, nos de leitura, ou que adicionam uma informação do formato intermediário ao formato desejado, nos de escrita.

Como o propósito do projeto é tornar possível o uso de questões do CodeForces e do BOCA na plataforma do Moodle, os únicos métodos funcionais são os necessários para tornar esses dois processos possíveis (Polygon `read`, BOCA `read` e CodeRunner `write`).

## IV. Implementação

A tradução dos formatos envolve muitos processos diferentes e é essencial que eles sejam implementados de forma legível, organizada e eficiente, dessa forma o código pode ser compreendido por outras pessoas que virão a utilizá-lo. A seguir, serão apresentados os detalhes técnicos da implementação do software.

### A. Tecnologias utilizadas

Este projeto foi estruturado na linguagem de programação Python 3 [12] devido a simplicidade de sua sintaxe, que permite que um código conciso e legível seja elaborado em pouco tempo, além da existência de módulos de terceiros, de onde muitas ferramentas prontas e eficientes podem ser utilizadas [13]. Python é também uma linguagem orientada a objetos, abordagem utilizada na implementação do software. Existem outras linguagens de programação que lidam de forma mais eficiente com a memória, como C, que poderiam ser utilizadas neste projeto, porém como a tradução das questões não possui um custo computacional alto, a simplicidade e organização foram priorizadas.

Para ter um maior controle sobre as mudanças no software, foi utilizada uma ferramenta de controle de versão, um sistema que registra todas as alterações de um projeto ou arquivo, possibilitando o acesso a versões antigas. Foi escolhida a ferramenta *git*, de controle de versões, e os arquivos foram hospedados na plataforma do *GitHub*[7] onde estão disponíveis gratuitamente. Existem outras plataformas de controle de versão, mas que não oferecem qualquer diferencial para este projeto, dessa forma, o *GitHub* foi escolhido por existir uma maior familiaridade com a plataforma.

### B. Formato intermediário

As informações presentes nas questões possuem dois propósitos principais: o entendimento e a avaliação da questão. Para implementar um formato intermediário que abstraia bem as informações presentes nos diferentes formatos dos juízes eletrônicos, foram utilizadas duas classes: a classe `Statement`, que armazena informações utilizadas no entendimento da questão, e a classe `Evaluation`, que armazena informações utilizadas na avaliação de uma solução. O diagrama das classes pode ser visto na Figura 1.

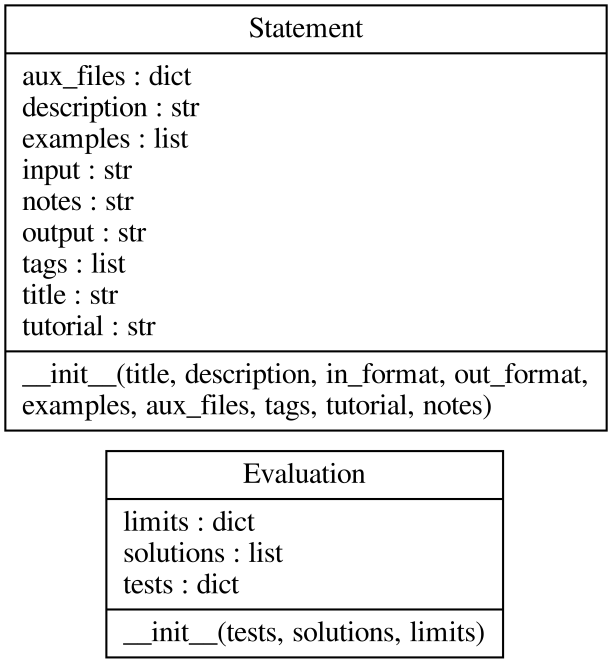



Figura 1: Diagramas das classes `Statement` e `Evaluation`.

A classe `Statement` possui todos os enunciados do texto separados em seis seções: nome, contexto, entrada, saída, tutorial e notas, para que a formatação do texto da questão seja facilitada. Todos os enunciados são armazenados no formato LaTeX, por ser uma ferramenta com diversos recursos para a formatação de textos e equações matemáticas, e por já ser utilizada nas plataformas do BOCA e Polygon, facilitando a tradução desses formatos. A classe possui também os casos de teste de exemplo e as *tags*, armazenados em listas, além de alguns arquivos auxiliares, como imagens, que são armazenados em memória como `binary objects`.

A classe `Evaluation` possui todos os casos de teste da questão, armazenados em um dicionário, estrutura de dados do Python, onde é possível acessar de forma eficiente os casos de exemplo e os casos que não aparecem para o aluno, separados em entrada e saída. A classe também possui os limites de tempo de execução (segundos), memória alocada (megabytes) e do tamanho máximo do arquivo de solução (kilobytes). As soluções são armazenadas juntamente com a sua linguagem de programação em uma lista, onde sempre existe uma solução identificada como principal, escolhida pelo autor da questão como a solução de referência.

As classes `Statement` e `Evaluation` compõem a classe `Problem`, cujo diagrama é apresentado na Figura 2, que também conta com o identificador da questão. Essa classe representa o formato intermediário em si, e pode ser instanciada e manipulada entre os processos de tradução.

[7] https://github.com/gnramos/convert-ej

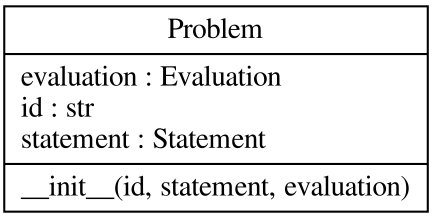



Figura 2: Diagrama da classe `Problem`.

### C. Leitura e escrita

Duas classes abstratas responsáveis pelos processos de tradução foram implementadas, a `Reader` e a `Writer`, que possuem métodos abstratos de leitura/escrita para cada um dos dados no formato intermediário. Os métodos de cada uma das classes abstratas podem ser vistos nas Figuras 3a e 3b.

Essas classes abstratas são as responsáveis por terem os métodos abstratos herdados e implementados pelas classes dos diferentes formatos de questão. Essa abordagem permite que os processos de tradução sejam forçados a possuir uma mesma estrutura definida pelas classes `Reader` e `Writer`. Segue a descrição da implementação desses métodos específicos para cada um dos formatos.

*1) Função de leitura do Polygon:* O Polygon trabalha com questões no formato `ZIP`, e para ter acesso aos arquivos foi utilizada uma ferramenta do módulo "zipfile" do Python, extraindo os arquivos e os manipulando em memória.

Todas as questões do Polygon possuem um arquivo chamado `problem.xml` no diretório principal, com informações a respeito da questão, como o nome, limites de tempo e memória, *tags*, e o diretório de diversos arquivos como casos teste, textos e soluções. Para manipular arquivos `XML` foi utilizado o módulo "ElementTree" do Python, que facilita o acesso a informações dentro de arquivos `XML`.

Nos arquivos os enunciados estão divididos por seções (nome, contexto, entrada, saída, tutorial e notas) em arquivos LaTeX separados, que são as mesmas seções definidas na classe `Statement`, onde também serão armazenados em LaTeX, logo, nenhuma transformação é necessária para armazenar esses arquivos. Os arquivos auxiliares, como imagens, tem suas informações armazenadas no formato de `binary object`.

Os casos de teste são compostos por arquivos de entrada e arquivos de saída, presentes em um mesmo diretório, identificados com um número diferente por teste, e a partir do arquivo `XML` é possível identificar quais casos são de exemplo. Esses casos são armazenados em um dicionário, de forma a ser possível identificar os casos normais e os de exemplo, e se são a entrada ou a saída. O arquivo da questão pode conter várias soluções, até mesmo soluções que não funcionam, e é preciso escolher quais delas vão para o formato intermediário. No arquivo `XML` é possível obter a informação de qual solução é a principal, quais funcionam e em quais linguagens as soluções estão escritas.

Outras informações como limites de memória e de tempo, tamanho máximo do arquivo da solução, os conteúdos abordados (*tags*) e o nome identificador da questão (*handle*) são retiradas diretamente do arquivo `problem.xml`.

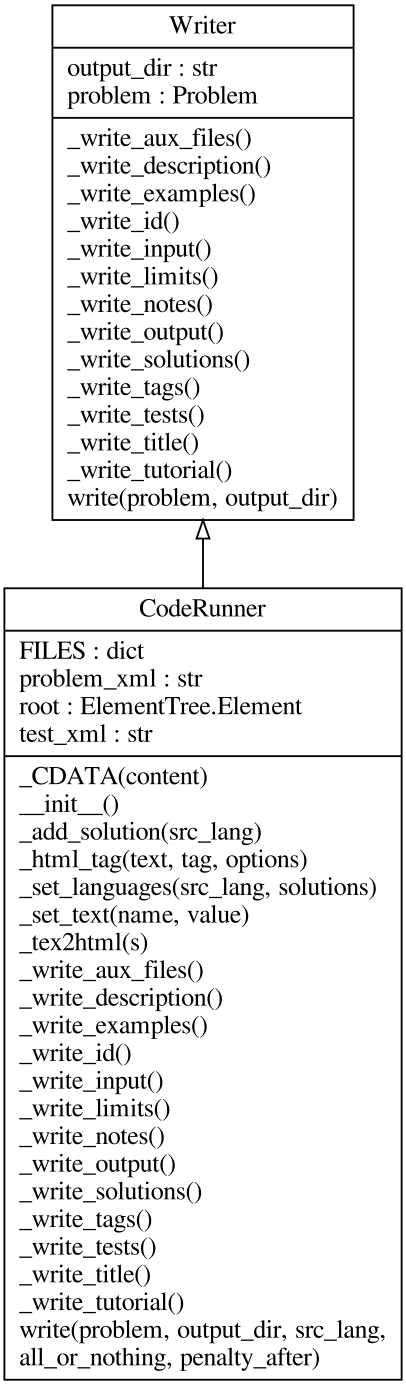
*2) Função de leitura do BOCA:* O BOCA trabalha com questões no formato `ZIP` e para extrair os dados também foi utilizado do módulo "zipfile". O acesso a informações como casos de teste, limites e identificador são feitas de forma simples e direta, contudo, o formato não possui necessariamente informações como *tags*, soluções, e nem uma estrutura padrão para o enunciado. Dessa forma, foi definida uma estrutura especifica para armazenar essas informações ausentes, e para que a tradução ocorra admite-se que a questão esteja de acordo com essa estrutura, que inclui as soluções, as `tags` e uma pasta com os textos da questão, que se assemelha à estrutura do formato intermediário por simplicidade. Mais informações podem ser obtidas na documentação do software.

A ausência dessas informações no formato BOCA é um problema no processo de tradução desse formato, que fica limitado ao uso de questões em uma estrutura não padrão com as novas informações necessárias. Entretanto, as questões criadas no âmbito da Universidade de Brasília têm os arquivos de descrição disponíveis no formato LaTeX e o ajuste para a estrutura descrita pode ser feito de forma relativamente simples.

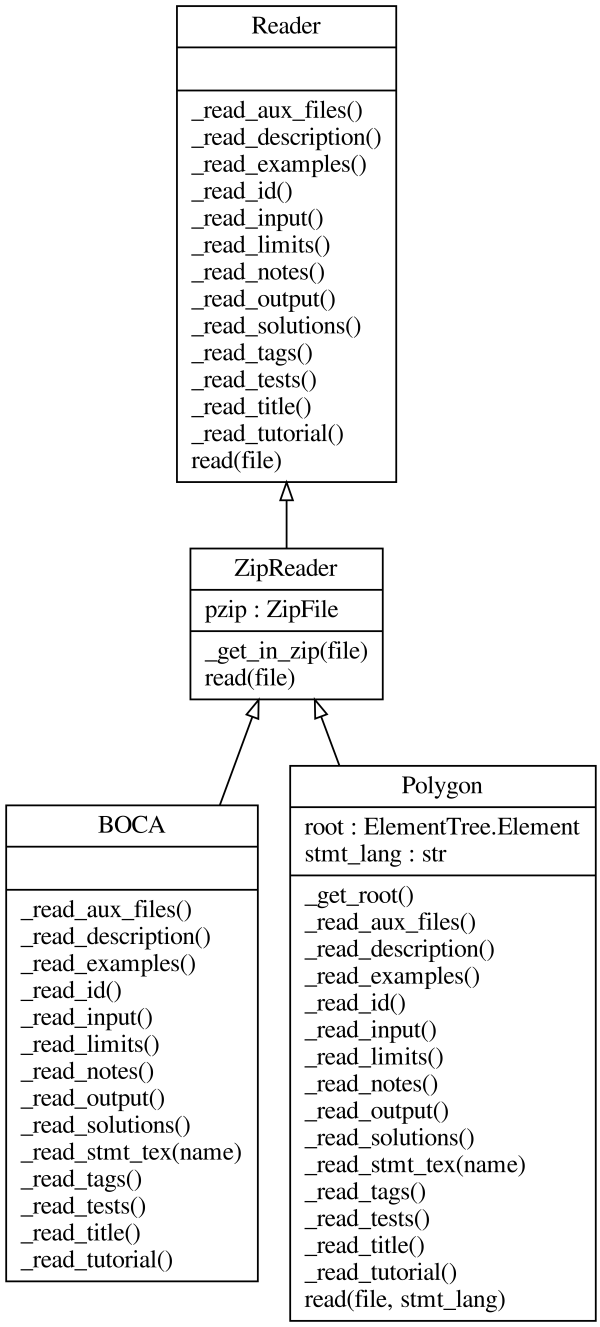
*3) Função de escrita do CodeRunner:* O CodeRunner trabalha com questões em um formato `XML`, e para gerar esse arquivo, o programa conta com dois modelos (*templates*) prontos, um com a estrutura da questão e outro com uma estrutura para os casos de teste. Para manipular esses modelos, foi utilizado o módulo "ElementTree", e uma vez com o *template* pronto, é possível preencher essa estrutura com as informações do formato intermediário.

O texto da questão no formato do CodeRunner é escrito em HTML, diferente do texto do formato intermediário, que se encontra em LaTeX. A abordagem inicial era a de utilizar um módulo do Python que seria responsável por traduzir o texto de LaTeX para HTML, contudo as ferramentas testadas transformavam equações matemáticas em imagens, complicando o processo de tradução. A solução encontrada foi utilizar expressões regulares para gerar uma tradução manual dos textos, e como a ferramenta MathJax [14], que suporta equações matemáticas em LaTeX para um texto em HTML, está disponível no CodeRunner, não houve a necessidade de transformar as equações em imagens. Como normalmente os textos não utilizam de ferramentas de formatação muito específicas do LaTeX, apenas inserção de imagens ou textos em itálico ou negrito, uma tradução manual simples é o suficiente para traduzir corretamente a maioria dos enunciados das questões.

O `XML` possui alguns caracteres reservados, ou seja, que possuem um significado especial para a linguagem, e a aparição desses nos textos ou na solução da questão pode ser um problema, pois é esperado que o analisador trate os caracteres do texto ou da solução de forma literal, e não como uma possível marcação da linguagem. Os blocos CDATA são blocos de textos cujos caracteres não serão analisados, e a utilização desse bloco resolve o problema descrito acima.




(a) Classe Writer.




(b) Classe Reader.

Figura 3: Diagramas de classe.

Uma pequena alteração no método de serialização do módulo ElementTree (que não possui suporte a blocos CDATA) foi feita para tratar a inserção de uma informação como um bloco CDATA, sempre que possuir uma tag específica, usada pelo programa ao incluir os textos ou a solução no *template*.

As imagens do texto são armazenadas dentro do arquivo `XML`, codificadas em "Base 64" [15]. Para codificar as imagens nesse formato foi utilizado o módulo "base64" do Python, que possui ferramentas de tradução que recebem uma imagem e codificam sua informação. Um dos problemas encontrados é quando alguma imagem da questão não está em um formato aceito pelo HTML, como é o caso de imagens no formato `EPS` ou `PDF`, nesse caso, o usuário é informado que o formato da imagem deve ser convertido para um aceito pelo HTML (`PNG`, `JPEG`, `GIF`, entre outros) para que a tradução ocorra.

Cada caso de teste deve ser armazenado em uma estrutura especifica dentro do arquivo `XML`, essa estrutura está presente em um dos *templates* do programa. Para cada caso de teste essa estrutura é preenchida com a entrada e a saída, e com um parâmetro que indica se esse caso deve aparecer de exemplo junto ao texto ou não, após preenchida ela é incluída na estrutura da questão.

Existem duas informações importantes na construção da questão do CodeRunner que não estão presentes no formato intermediário, elas são o parâmetro "penalty" e o parâmetro "all_or_nothing", que dizem a respeito da penalidade de uma resposta errada, e se um programa deve passar em todos os casos de teste para receber uma nota, respectivamente. Esses parâmetros possuem um valor padrão que pode ser alterado pelo usuário ao executar a função de tradução. Para mais informações consultar a mensagem de ajuda do programa. As demais informações como limites de tempo e memória, tamanho máximo do arquivo de solução, *tags* e outros, são facilmente inseridos no *template* com o uso do módulo "ElementTree", sem passar por qualquer transformação.

Cada questão do CodeRunner é atrelada a uma única linguagem de programação, contudo, podem existir múltiplas soluções armazenadas no formato intermediário, e por isso o programa gera uma questão diferente para cada solução existente no formato intermediário, ficando à opção do usuário qual utilizar.

### *D. Uso do software*

Por fim, foi utilizado o módulo "argparse" do Python para lidar com a interação do software com o usuário, como escolha da tradução desejada, alteração de parâmetros, entre outros. O uso do software gerado, após instalação via `pip`, é feito de maneira simples pela linha de comando:

```
convert-ej reader writer nome_arquivo
```

O programa `convert-ej` considera o primeiro argumento (`reader`) como o formato de origem da questão (*Polygon* ou *BOCA*), o segundo (`writer`) como formato destino, para qual a questão será traduzida (*CodeRunner*) e o último argumento (`nome_arquivo`) como o arquivo da questão, ou nome do diretório contendo os arquivos das questões a serem processadas.

## V. Resultados experimentais

Para testar a eficácia do software, foram traduzidas algumas questões nos formatos Polygon e BOCA para o CodeRunner, e as questões geradas foram testadas na plataforma do Moodle, verificando a integridade do texto, dos casos de teste, da solução e de outras informações e parâmetros.

Um dos testes pode ser visto na Figura 4, onde a questão "Neologismos" do Polygon foi traduzida para o CodeRunner. É possível perceber que as seções do texto continuaram bem formatadas, a imagem está presente e continua igual, e os casos de teste visíveis são os mesmos nas duas plataformas. As aspas, que possuem sintaxes diferentes nas duas linguagens utilizadas (LaTeX e HTML), estão presentes nas duas questões, bem como o destaque na palavra "triggerou", apesar de apresentar interpretações diferentes nas duas linguagens (sublinhado e itálico). Os casos de teste estão todos presentes, e a solução da questão é aceita em todos os casos.

Outro teste pode ser visto na Figura 5, onde a questão "Sem número" do BOCA que foi traduzida para o CodeRunner. O texto da questão continua bem formatado e os casos de teste de exemplo são iguais. É possível perceber que as partes do texto tratadas como equação matemática pelo LaTeX, como o símbolo de menor ou igual ($\leq$), estão idênticas na questão do CodeRunner, graças a ferramenta MathJax. Os casos de teste também estão todos presentes, e a solução é aceita em todos eles.

O processo de tradução de uma questão pode levar de muitos minutos a horas, se feito manualmente, pois todas as equações matemáticas precisam ser refeitas, texto e imagens precisam ser copiados, e talvez convertidos para funcionar em HTML. Além disso, é preciso lidar com cada um dos casos de teste, que em algumas questões podem chegar a centenas de arquivos, com milhares ou até milhões de caracteres cada. Todo esse trabalho é feito quase que instantaneamente pelo software, que consegue traduzir uma questão em menos de 1 segundo, como pode ser observado pelos resultados de testes apresentados na Tabela II. Os testes foram conduzidos em uma máquina com um Intel Core i7-8550U e 8GB de memória RAM, rodando o sistema operacional Ubuntu versão 20.04 LTS.

Tabela II: Tabela de resultados das traduções.

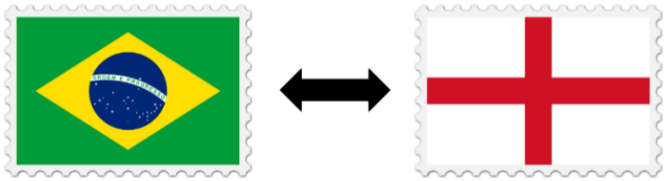
| Questão | Formato | Tempo gasto | Texto bem formatado | Casos de teste e solução funcionando |
|---|---|---|---|---|
| neologismo | BOCA | 7 ms | Sim | Sim |
| sem-numeros | BOCA | 9 ms | Sim | Sim |
| neologismo | Polygon | 8 ms | Sim | Sim |
| sem-numeros | Polygon | 8 ms | Sim | Sim |
| fizzbusao | Polygon | 9 ms | Sim | Sim |
| entao-e-natal | Polygon | 13 ms | Sim | Sim |

O software foi implementado de forma a facilitar a incorporação de novos juízes eletrônicos, graças ao uso do formato intermediário, tornando possível que contribuições

## Neologismos

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 4 megabytes

O avanço dos maratonistas triggerou a incorporação de neologismos à comunicação, dificultando a interface com os técnicos. Sem perspectivas de reverter a situação, e irredutíveis quanto ao uso dos "novos" termos, os técnicos se viram forçados à comunicação por meio de um tradutor. O processo é simples, basta acrescentar "ar" ao termo dado. A única exceção é o termo "code", que os maratonistas insistem em declamar como "codar".



Defina um programa que, dada uma instrução em inglês, a traduza como um neologismo de maratonista.

### Input

A entrada consiste em um string, com não mais de 50 letras minúsculas, definindo um único termo em inglês.

### Output

A saída deve ser a versão do termo dado, traduzida como um neologismo verbal de maratonista.

### Examples

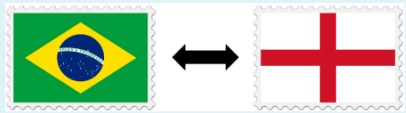
| standard input | standard output |
|---|---|
| print | printar |
| setup | setupar |

### Note

O primeiro caso é bem comum.

(a) Polygon.

O avanço dos maratonistas *triggerou* a incorporação de neologismos à comunicação, dificultando a interface com os técnicos. Sem perspectivas de reverter a situação, e irredutíveis quanto ao uso dos "novos" termos, os técnicos se viram forçados à comunicação por meio de um tradutor. O processo é simples, basta acrescentar "ar" ao termo dado. A única exceção é o termo "code", que os maratonistas insistem em declamar como "codar".



Defina um programa que, dada uma instrução em inglês, a traduza como um neologismo de maratonista.

**Entrada**
A entrada consiste em um string, com não mais de 50 letras minúsculas, definindo um único termo em inglês.

**Saída**
A saída deve ser a versão do termo dado, traduzida como um neologismo verbal de maratonista.

**Observações**
O primeiro caso é bem comum.

**For example:**

| Input | Result |
|---|---|
| print | printar |
| setup | setupar |

**Answer:** (penalty regime: 0, 0, 10, 20, ... %)

```
1
```

(b) CodeRunner.

Figura 4: Questão do Polygon no CodeRunner.

**Problema A**

# Sem número

Limite de tempo: 1 s

Polly quer um biscoito. E contar de 0 até $N$. O problema é que Polly tem uma abordagem caótica e sempre pula um número, mas nunca se lembra qual.

Dada uma sequência de $N$ inteiros, indique qual deles Polly pulou.

## Entrada

A primeira linha da entrada apresenta o valor de $N$ ($1 \leq N \leq 100$). A segunda linha fornece $N$ inteiros distintos $i$ ($0 \leq i \leq N$), separados por espaço.

## Saída

Apresente o número no intervalo $[0, N]$ que não foi dito por Polly.

## Exemplos

| Entrada | Saída |
|---|---|
| 10<br>1 2 3 4 5 6 7 8 9 10 | 0 |
| 10<br>9 8 7 6 5 4 3 2 1 0 | 10 |
| 5<br>3 1 5 0 2 | 4 |

(a) BOCA.

Polly quer um biscoito. E contar de 0 até $N$. O problema é que Polly tem uma abordagem caótica e sempre pula um número, mas nunca se lembra qual.

Dada uma sequência de $N$ inteiros, indique qual deles Polly pulou.

**Entrada**

A primeira linha da entrada apresenta o valor de $N$ ($1 \leq N \leq 100$). A segunda linha fornece $N$ inteiros distintos $i$ ($0 \leq i \leq N$), separados por espaço.

**Saída**

Apresente o número no intervalo $[0, N]$ que não foi dito por Polly.

**Observações**

No primeiro caso, os números listados são de 1 a 10, portanto Polly pulou o 0.

No segundo exemplo, são dados os números de 0 a 9, e Polly pulou o 10.

No terceiro exemplo, Polly pulou 4.

**For example:**

| Input | Result |
|---|---|
| 10<br>1 2 3 4 5 6 7 8 9 10 | 0 |
| 10<br>9 8 7 6 5 4 3 2 1 0 | 10 |
| 5<br>3 1 5 0 2 | 4 |

**Answer:** (penalty regime: 0, 0, 10, 20, ... %)

```
1
```

(b) CodeRunner.

Figura 5: Questão do BOCA no CodeRunner.

a este projeto sejam facilmente implementadas. O código foi escrito dentro das recomendações descritas nas PEP's[8] (*Python Enhancement Proposals*) e foi propriamente documentado. Para facilitar o acesso a este projeto, o software desenvolvido foi estruturado na forma de um pacote Python e disponibilizado gratuitamente no Pypi[9], repositório oficial de software de terceiros do Python, onde é possível encontrar um texto explicando como instalar o programa utilizando o `pip`, gerenciador de pacotes incluso nas versões mais recentes do Python 3, e como utilizá-lo, descrevendo os parâmetros e exibindo alguns exemplos de uso.

Alguns tutores e docentes da Universidade de Brasília testaram o software e comentaram sobre suas experiências. O aluno do curso de Ciência da Computação e tutor da disciplina Algoritmos e Programação de Computadores, José Leite Marcos, utilizou o software para traduzir questões do Polygon para o CodeRunner no Moodle: "Tive alguns problemas iniciais pois o conversor não aceita imagens em formato PDF. Ao usar um formato aceito, foi tudo muito prático e em poucos segundos a questão completa estava convertida para o outro juíz, incluindo latex, imagens e centenas de casos de teste! Aumenta muito a produtividade na conversão de questões, principalmente nas com muitos casos de teste."

O docente Dr. Guilherme Novaes Ramos, que ministra as disciplinas de programação Algoritmos e Programação de Computadores e Estruturas de Dados testou o software: "Traduzi 20 questões do Polygon para o CodeRunner e para o BOCA, e os resultados foram aceitáveis, houve pequenos problemas de mapeamento de elementos LaTeX para HTML, no caso de imagens, os problemas de incompatibilidade foram indicados pelo programa"

Por último, professor Dr. Vinicius Ruela Pereira Borges, que ministra as disciplinas Algoritmos e Programação de Computadores e Programação Competitiva também testou o software: "Utilizei a ferramenta convert-ej para criar problemas (exercícios) para o banco de questões da disciplina Algoritmos e Programação de Computadores (APC). No meu caso, utilizei as questões do repositório que tenho na plataforma Polygon (Codeforces), uma vez que eu ministrava a referida disciplina utilizando o Codeforces. Primeiramente, baixei o pacote (Linux) de cada questão do Polygon e converti cada um para o formato do Coderunner. Consegui importar os arquivos XML gerados para o banco de questões do Coderunner e incluí normalmente nos questionários da disciplina no Aprender3/Moodle da disciplina APC. Por isso, posso dizer que a ferramenta cumpriu seu objetivo e atendeu as minhas necessidades. Os enunciados dos problemas foram convertidos corretamente para o formato do Coderunner, inclusive com os comandos LaTeX, e também todos os casos de teste (exemplo, visíveis e ocultos) e as soluções. Certamente continuarei utilizando a ferramenta, pois economizei tempo reaproveitando questões já criadas no Polygon, ao invés de recriar elas novamente no Coderunner, em que o processo de criação de questões é essencialmente manual."

Nos *feedbacks* recebidos, ocorreram dois problemas durante o processo de tradução: com o formato da imagem, e com a tradução do LaTeX para HTML. O primeiro problema mencionado ocorre pois o HTML (utilizado pelo CodeRunner) não aceita imagens em formato PDF, assim, para concluir a tradução, o usuário é informado que deve converter as imagens para um formato aceito para poder prosseguir. Já o problema de mapeamento de sintaxes do LaTeX para HTML, ocorre pelo fato da função de tradução manual, desenvolvida no projeto, não abranger todas as sintaxes existentes no LaTeX, apenas as principais (sublinhado, negrito, etc). O uso dessa tradução manual é uma limitação do projeto, contudo, é possível contornar o problema com uma simples e rápida alteração manual no texto, corrigindo as sintaxes não traduzidas.

## VI. Conclusão

A automação da correção de questões de programação tornou o processo de ensino desta área muito mais eficiente, e seu uso pode evitar evasões e gerar mais interesse nas disciplinas da computação. Portanto, é interessante que exista uma forma eficiente de reaproveitar de questões prontas entre plataformas de programação com esse recurso, para que seja possível economizar tempo em vez de elaborar novas questões ou traduzir manualmente os formatos de diferentes plataformas.

Neste trabalho, foi proposto e implementado um software de tradução de questões de programação entre diferentes juízes eletrônicos, automatizando o processo de tradução dos diferentes formatos adotados pelas plataformas. Para tanto, foram incorporados os formatos de três plataformas principais no escopo da Universidade de Brasília, o CodeForces, o BOCA e o Moodle, com o intuito de otimizar o uso de questões prontas em disciplinas de programação na universidade.

Para traduzir os formatos, o software conta com algumas funcionalidades principais, como os métodos de leitura e de escrita, responsáveis pela manipulação dos dados das questões, e o uso de um formato intermediário entre esses processos, que trouxe uma abordagem mais simples para a implementação do programa. Uma vez que a motivação inicial do projeto era de apenas reaproveitar de questões do CodeForces e do BOCA na plataforma do Moodle, apenas os métodos leitura e escrita necessários para tornar esses processos possíveis foram implementados (métodos de leitura do Polygon e do BOCA e de escrita do CodeRunner).

Os resultados mostraram que o software traduz bem os formatos de questão, sem qualquer problema ou erro nos testes feitos, mantendo a estrutura dos enunciados e conservando todas as informações necessárias para que a correção automática aconteça. O programa torna um processo manual de tradução que pode levar horas para ser realizado, em um processo automatizado que leva menos de um segundo para ser feito, essa diferença agiliza o processo de importação/exportação de questões entre juízes, permitindo o reaproveitamento de questões de forma eficiente entre as plataformas.

Como trabalhos futuros neste projeto, pretende-se completar os processos de tradução que não foram implementados

[8]https://www.python.org/dev/peps/

[9]https://pypi.org/project/convert-ej/

(métodos de escrita do Polygon e do BOCA e método de leitura do CodeRunner), e incorporar outros juízes eletrônicos, como o URI, UVA, entre outros. Além disso, pode-se melhorar os processos de tradução, como por exemplo, a tradução do formato do enunciado de LaTeX para HTML, que foi implementada manualmente e ainda não abrange toda a sintaxe desses formatos, e a implementação de uma conversão automática de imagens, para que seja possível converter um formato qualquer de imagem existente na questão nos formatos aceitos pelo HTML. Por fim, é possível realizar uma integração com a API de comunicação do Polygon e com o banco de dados do Moodle, para facilitar a manipulação dos arquivos de questão.

## REFERÊNCIAS


[1] R. H. Franke, "Technological revolution and productivity decline: Computer introduction in the financial industry," *Technological Forecasting and Social Change*, vol. 31, no. 2, pp. 143 – 154, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0040162587900461

[2] S. Ryan, H. Freeman, B. Scott, and D. Patel, *The virtual university: The internet and resource-based learning*. Psychology Press, 2000.

[3] P. Dillenbourg, D. Schneider, and P. Synteta, "Virtual Learning Environments," in *3rd Hellenic Conference "Information & Communication Technologies in Education"*, A. Dimitracopoulou, Ed. Rhodes, Greece: Kastaniotis Editions, Greece, 2002, pp. 3–18. [Online]. Available: https://telearn.archives-ouvertes.fr/hal-00190701

[4] L. Hasan, "The usefulness and usability of moodle LMS as employed by Zarqa University in Jordan," *JISTEM - Journal of Information Systems and Technology Management*, vol. 16, 00 2019. [Online]. Available: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1807-17752019000100308&nrm=iso

[5] R. M. Hoed, "Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de computação," *MPCA - Mestrado Profissional em Computação Aplicada*, 2016. [Online]. Available: https://repositorio.unb.br/handle/10482/22575

[6] D. Brauner, P. Margreff, T. Tavares, V. da Costa, and A. Silva, "Estímulo à prática multidisciplinar no ensino de computação e design através de um evento de programação focado em problemas," in *Anais do XXIV Workshop sobre Educação em Computação*. Porto Alegre, RS, Brasil: SBC, 2020, pp. 131–140. [Online]. Available: https://sol.sbc.org.br/index.php/wei/article/view/9656

[7] B. Cheang, A. Kurnia, A. Lim, and W.-C. Oon, "On automated grading of programming assignments in an academic institution," *Computers & Education*, vol. 41, no. 2, pp. 121 – 131, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131503000307

[8] B. Cheang, A. Kurnia, A. Lim, and W. Oon, "Automated grading of programming assignments," in *Proceedings of the 11th International Conference on Computers in Education (ICCE 2003)*. Citeseer, 2003, pp. 45–52.

[9] S. R. e Mirtha Venero e Carla Rodriguez e Denise Goya e Rafaela Rocha, "Avaliando ambientes para ensino de programação com suporte para o desenvolvimento da metacognição," *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, vol. 8, no. 1, p. 417, 2019. [Online]. Available: https://www.br-ie.org/pub/index.php/wcbie/article/view/8983

[10] S. Zhigang, S. Xiaohong, Z. Ning, and C. Yanyu, "Moodle plugins for highly efficient programming courses," *1st Moodle Research Conference*, 2001.

[11] R. S. Pressman, *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

[12] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. [Online]. Available: https://docs.python.org/3/reference/

[13] G. Lindstrom, "Programming with python," *IT Professional Magazine*, vol. 7, no. 5, p. 10, 2005.

[14] D. Cervone, "Mathjax: a platform for mathematics on the web," *Notices of the AMS*, vol. 59, no. 2, pp. 312–316, 2012.

[15] S. Josefsson, "The base16, base32, and base64 data encodings (rfc 4648)," *Network Working Group*, p. 44, 2006.