UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Ampliação das linguagens suportadas pelo Ambiente de Simulação Distribuída Automático (ASDA)

Nathalia Lopes

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Nathalia Lopes

Ampliação das linguagens suportadas pelo Ambiente de Simulação Distribuída Automático (ASDA)

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestra em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Profa. Dra. Sarita Mazzini Bruschi

USP – São Carlos
Abril de 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

L864a

Lopes, Nathalia
Ampliação das linguagens suportadas pelo Ambiente de Simulação Distribuída Automático (ASDA) / Nathalia Lopes; orientadora Sarita Mazzini Bruschi. -- São Carlos, 2023.
134 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Simulação de Sistemas. 2. Ambiente Automático de Simulação. 3. Avaliação de desempenho. 4. Ensino-Aprendizagem. I. Bruschi, Sarita Mazzini , orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Nathalia Lopes

Expansion of the languages supported by the Ambiente de Simulação Distribuída Automático (ASDA)

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Sarita Mazzini Bruschi

USP – São Carlos
April 2023

Este trabalho é dedicado aos amigos que fiz no LaSDPC.

AGRADECIMENTOS

Agradeço à minha família por todo apoio incondicional e incentivo nos estudos, em especial ao meu irmão Bruno Lopes por sempre me apoiar em minhas escolhas e tentar me ajudar de todas as formas possíveis, não poderia ter um irmão melhor.

Gostaria também de expressar toda minha gratidão à minha orientadora Dra. Sarita Mazzini Bruschi, por todo o apoio, consideração e paciência que dedicou a mim durante esta jornada, além de ser uma excelente profissional, também tem um coração maravilhoso. Sua orientação e encorajamento foram fundamentais para o sucesso deste projeto.

Agradeço aos incríveis amigos que fiz no LaSDPC e vou levar comigo para o resto da vida, Gabriel Tomiatti Andreazi, Lucélia Cunha da Rocha Santos, Guilherme Martins, Welington da Silva Martins e em especial agradeço a uma das melhores pessoas que conheci nesta jornada Matheus Henrique Junqueira Saldanha, sem você nada disso seria possível.

Agradeço também ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro à pesquisa.

RESUMO

LOPES, N. **Ampliação das linguagens suportadas pelo Ambiente de Simulação Distribuída Automático (ASDA)**. 2023. 134 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

A simulação é uma técnica da avaliação de desempenho de sistemas computacionais, que por meio de um modelo do sistema real realiza experimentos para se obter informações sobre o sistema. Tratando-se de um processo complexo, a simulação demanda por profundos conhecimentos em diversas áreas. No contexto acadêmico, para diminuir a complexidade são utilizados ambientes automáticos, um exemplo é o Ambiente de Simulação Distribuída Automático (ASDA) que automatiza todo o processo de geração do programa de simulação para o usuário. O ASDA é utilizado como auxílio para o ensino de simulação, porém tem sua utilidade limitada, pois encontra-se defasado. Este projeto tem como objetivo proporcionar uma melhora na qualidade da aprendizagem de avaliação de desempenho, com o auxílio da ferramenta ASDA. Para realizar este objetivo, foi realizada a implementação de uma extensão para a ferramenta ASDA, que permite a geração de código de simulação em linguagens de programação de alto nível. Para validação da proposta foi realizado dois experimentos com aplicação de questionários qualitativos que foi conduzido durante algumas aulas, utilizando a ferramenta implementada. Os resultados foram positivos, os alunos tiveram uma boa adesão ao uso da ferramenta e a consideraram de grande apoio durante as aulas.

Palavras-chave: Simulação, Ambiente Automático de Simulação, Avaliação de desempenho, Aprendizagem.

ABSTRACT

LOPES, N. **Expansion of the languages supported by the Ambiente de Simulação Distribuída Automático (ASDA)** . 2023. 134 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Simulation is a technique for evaluating the performance of computer systems, which uses a model of the real system to carry out experiments to obtain information about the system. As it is a complex process, simulation demands deep knowledge in several areas. In the academic context, automatic environments are used to reduce complexity, an example being the Automatic Distributed Simulation Environment (ASDA), which automates the entire process of generating the simulation program for the user. ASDA is used as an aid for teaching simulation, but its usefulness is limited because it is outdated. This project aims to provide an improvement in the quality of performance assessment learning, with the help of the ASDA tool. To accomplish this objective, an extension to the ASDA tool was implemented, which allows the generation of simulation code in high-level programming languages. For validation of the proposal, two experiments were carried out with the application of qualitative questionnaires that were conducted during some classes, using the implemented tool. The results were positive, the students had a good adhesion to the use of the tool and considered it to be of great support during the classes.

Keywords: Simulation, Automatic Simulation Environment, Performance Evaluation, Learning.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CAD	*Computer-aided design*
CNAP	*Cisco Networking Academy Program*
DSL	*Domain-Specific Language*
FEA	*Finite element analysis*
ICMC	Instituto de Ciências Matemáticas e de Computação
IFSP	Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
MRIP	*Multiple Replication in Parallel*
SRAC	*Structural Research & Analysis Corporation*
UML	*Unified Modeling Language*
USP	Universidade de São Paulo
Yasc	*Yes, a simulator's compiler*

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

1.1 Considerações Iniciais

Neste capítulo é denotada uma introdução sobre a temática do trabalho. Na Seção 1.2 são apresentadas a contextualização do tema e uma introdução sobre o conceito geral deste projeto de mestrado. Na Seção 1.3 é discutida a motivação deste trabalho e os objetivos propostos que foram atingidos durante o desenvolvimento do projeto. Por fim, na Seção 1.4 é apresentada a estrutura completa do trabalho.

1.2 Contextualização

A avaliação de desempenho é um tópico importante quando se trata de sistemas computacionais. Ao se realizar uma avaliação de desempenho de um sistema computacional, é possível prever problemas, analisar situações de gargalo, verificar como o sistema pode reagir a uma determinada carga de trabalho, comparar sistemas, dentre outros.

Para realizar uma avaliação de desempenho, várias técnicas podem ser utilizadas. Uma que é bastante flexível é a simulação, que utiliza modelos que representam sistemas reais para realizar diversos testes e se obter análises acerca do comportamento do sistema em diferentes cenários (CASACA, 2005).

A realização de uma simulação demanda várias etapas com diferentes tipos de atividades, que requerem conhecimento profundo da ciência da simulação e do sistema a ser simulado levando em consideração que quanto mais complexo for o sistema, mais complicada se torna a criação do modelo para a simulação. Além disso, há a necessidade de se realizar todas as atividades com competência, a fim de se obter a compreensão dos resultados da simulação (WHITE; INGALLS, 2018).

Para o processo de tradução do modelo criado são utilizadas bibliotecas de simulação que

fornecem funções que facilitam a criação do programa de simulação. Porém ainda é necessário o conhecimento de como utilizar a biblioteca de simulação e conhecimentos de programação para a implementação do programa de simulação, na linguagem de programação escolhida.

Como uma forma de resolver esse problema, foi introduzido na literatura o conceito de Ambientes Automáticos, com o objetivo de simplificar o processo de simulação, e assim surgiram gradualmente, no mercado e no meio acadêmico, ferramentas capazes de automatizar grande parte do processo de criação de uma simulação (BRUSCHI, 2002).

1.3 Motivação e Objetivo

Considerando o contexto acadêmico, deve-se considerar a importância do ensino de Avaliação de Desempenho para a formação de alunos em cursos de computação. De modo a facilitar que o aluno adquira esse conhecimento, o uso de ambientes automáticos para o ensino de simulação é uma boa alternativa para contornar a curva de aprendizado desfavorável.

Um exemplo de ambiente automático é a ferramenta ASDA (Ambiente de Simulação Distribuída Automático) (BRUSCHI, 2002), que tem como objetivo facilitar a realização de uma avaliação de desempenho, automatizando todo o processo de criação do programa de simulação, deixando o usuário apenas com a tarefa de modelagem do sistema na interface gráfica da ferramenta.

O ASDA atualmente é utilizado para auxiliar o ensino durante as aula da disciplina de Avaliação de Desempenho, na Universidade de São Paulo (USP). Porém o mesmo apresenta alguns problemas que comprometem a qualidade do ensino. Em especial, nota-se que a ferramenta encontra-se defasada, uma vez que a geração dos programas de simulação apenas é realizada na linguagem de programação C, que possui uma curva de aprendizado desaforável, e utilizando bibliotecas de simulação antigas.

Partindo deste cenário, é possível melhorar a experiência dos alunos, a percepção de uso e a qualidade do aprendizado atual, utilizando a ferramenta ASDA como o principal elemento no ensino de avaliação de desempenho.

Este projeto de pesquisa tem como objetivo geral proporcionar uma melhora na qualidade da aprendizagem de avaliação de desempenho, com o auxílio da ferramenta ASDA. Visando a realização deste objetivo, foi proposta a implementação de uma extensão para a ferramenta ASDA, permitindo a geração de código de simulação também em outras linguagens de programação que possuam uma biblioteca de simulação baseada em redes de filas.

Como forma de avaliação foi proposta a realização de testes para comprovar a qualidade da extensão implementada e um experimento com aplicação de questionário, para qualificar o uso da ferramenta ASDA durante as aulas. Espera-se como resultado principal a melhora na qualidade do aprendizado no ensino de Avaliação de Desempenho, gerando como contribuição

uma nova forma de ensino de Avaliação de Desempenho e uma ferramenta atualizada.

O objetivo específico deste projeto de mestrado é:

- Avaliar diferentes exemplos tradicionais em códigos gerados em outras linguagens.

1.4 Estrutura e Organização do Trabalho

Este trabalho está organizado da seguinte maneira:

- **Capítulo 2** - É apresentada a Fundamentação Teórica necessária para o desenvolvimento do projeto, mais especificamente os conceitos referentes a: Simulação, Geração de Código, Ambientes de Simulação Automáticos e a ferramenta ASDA.
- **Capítulo 3** - São apresentados os Trabalhos Relacionados a este projeto de pesquisa juntamente da revisão da literatura realizada para se obter os trabalhos mais relevantes.
- **Capítulo 4** - É apresentado o processo de implementação e validação da extensão.
- **Capítulo 5** - São apresentadas as validações da proposta de pesquisa e os resultados gerados.
- **Capítulo 6** - São feitas as considerações finais sobre todo o desenvolvimento do projeto de pesquisa e seus resultados.

1.5 Considerações Finais

Neste capítulo foi realizada uma introdução sobre a temática abordada neste projeto de pesquisa, abordando como pontos principais os objetivos da pesquisa e as motivações que levaram a escolha dos mesmos. O próximo capítulo apresenta os conceitos estudados para o desenvolvimento do projeto.

CAPÍTULO

2

FUNDAMENTAÇÃO TEÓRICA

2.1 Considerações Iniciais

Com o objetivo de conhecer melhor as tecnologias e os conceitos aqui utilizados, apresenta-se, neste capítulo, os conceitos necessários para a construção e o entendimento desta pesquisa.

Este capítulo encontra-se organizado da seguinte maneira: na Seção 2.2 é apresentado o conceito geral de Simulação. Na Seção 2.3 são descritos alguns conceitos teóricos sobre Geração Automática de Código. Na Seção 2.4 são apresentados os Ambientes Automáticos, bem como seus exemplos mais conhecidos. Na Seção 2.5 é apresentado o ASDA, o ambiente de simulação automático que é a base desta pesquisa.

2.2 Simulação

O conceito de simulação vem sendo estudado desde a década de 1960 e evoluiu consideravelmente com o avanço das tecnologias e da computação, espalhando-se por diversas áreas além da computação, como setores financeiros e bancários, área da saúde, transportes, engenharias no geral, dentre outras (CASACA, 2005).

Dentre as várias definições de simulação, Shannon (1998) definiu como um processo de projetar um modelo que faz referência a um sistema real, seguido da realização de diversos experimentos com esse modelo para compreender o comportamento do sistema e avaliar as estratégias de operação do sistema. Maria (1997) considera a simulação uma ferramenta para avaliar o desempenho de um sistema já existente ou proposto, com diferentes configurações e por longos períodos de tempo real. Também complementa que o modelo de um sistema pode ser reconfigurado e experimentado, algo considerado infactível de ser realizado na prática com um sistema real, devido ao alto custo de tempo e de recursos computacionais e humanos necessários

para realizar tal feito. Além disso, muitas vezes há ações que são impraticáveis de serem feitas no sistema, contribuindo para a dificuldade de realizar numerosos experimentos para avaliar seu desempenho.

Existem vários tipos de simulação, e neste trabalho vamos abordar a simulação discreta baseada em eventos.

Simulação Discreta Baseada em Eventos

Na simulação de eventos discretos, as ações que ocorrem no sistema são modeladas como uma série de eventos em diferentes instantes de tempo, ou seja, cada mudança de estado do sistema é considerado um evento, como a chegada de um cliente ou o inicio do serviço (ROBINSON, 2004).

Os modelos de simulação são uma representação do sistema real que será simulado. Tais modelos são analisados por métodos numéricos, os quais utilizam de procedimentos computacionais para realizar as análises, isto é, os modelos de simulação são executados diversas vezes e são coletados os dados dos resultados das simulações para gerar uma análise concreta, que evidencia as medidas de desempenho do sistema simulado. Devido à grande quantidade de dados que são manipulados durante a simulação, em geral as execuções são realizadas de forma computacional, mas modelos de sistemas pequenos podem ser simulados manualmente (BANKS *et al.*, 2005).

Bibliotecas de Simulação

Implementar uma simulação pode se tornar uma atividade complexa, uma vez que o usuário necessita de conhecimentos profundos de programação para escrever o código de simulação. Diante de tais desafios, as linguagens de simulação foram propostas para facilitar a criação de códigos de simulação pelo programador (MATLOFF, 2008). As bibliotecas de simulação ou extensões funcionais são consolidadas na literatura como linguagens de simulação, apesar de não serem de fato linguagens de programação. Tais bibliotecas possuem funções para efetuar simulações de acordo com o tipo de simulação a ser realizado, sempre atrelados a uma linguagem de programação especifica (SPOLON, 1994).

2.3 Geração de Código

A Geração de Código revoluciona a maneira como se escreve programas, pois permite a construção de uma aplicação de alta qualidade, pronta para mudanças durante o desenvolvimento e durante todo o ciclo de vida da aplicação, com um tempo reduzido. Esse nível de agilidade de software é oferecido pela geração de código, além de permitir ao mesmo tempo a criação de uma grande quantidade de código automaticamente (DOLLARD, 2008).

Definir o termo Geração de Código pode ser considerado uma tarefa fácil, entretanto o termo pode ser usado em diferentes cenários da computação, tornando assim o seu compreendimento um pouco confuso.

Em termos gerais, a Geração de Código é qualquer forma de criar código diferente de escrevê-lo linha por linha em um editor, é simplesmente um código que escreve código. À primeira vista pode parecer um pouco complexo, porém seus benefícios são evidentes quando implementado (DOLLARD, 2008). Outros autores definem a Geração de Código como uma técnica usada para construir e usar programas para escrever outros programas, e com isso criar código consistente e de alta qualidade mais rapidamente (HERRINGTON, 2003).

Segundo Rumpe (2017), a geração de código é um fator significativo de sucesso para o uso de modelos no processo de desenvolvimento de software. Podemos gerar código de forma eficiente para os sistemas, melhorando assim a consistência entre o modelo e sua implementação, bem como economizando recursos.

Um dos pontos fortes da geração de código é que ela agiliza a replicação de fragmentos de código semelhantes que aparecem com frequência no código-fonte do programa. Isso reduz drasticamente o tamanho e a complexidade dos artefatos que precisam ser criados manualmente. Por sua vez, isso reduz o número de erros de programação e o código gerado tem uma conformidade ainda melhor com padrões de qualidade de software. Além disso, a geração de código também permite o rápido desenvolvimento do sistema, assim como permite adaptá-lo com mais flexibilidade a partir de modelos (RUMPE, 2017).

2.3.1 Programação Generativa

Entende-se programação generativa como o processo de projetar e implementar código ou módulos de software de maneira automática, atendendo a requisitos específicos com alta otimização e confiabilidade, podendo até mesmo gerar sistemas inteiros a partir de especificações (EISENECKER, 1997). Schlee e Vanderdonckt (2004) definem programação generativa como um paradigma de engenharia de software baseado na modelagem de famílias, que tem como intuito gerar códigos mais customizados e otimizados, utilizando componentes reutilizáveis.

A programação generativa possui uma abordagem que pode gerar famílias inteiras de produtos, denominada modelo de domínio generativo, que é composta pelos três elementos apresentados na Figura 1 (SCHLEE; VANDERDONCKT, 2004):

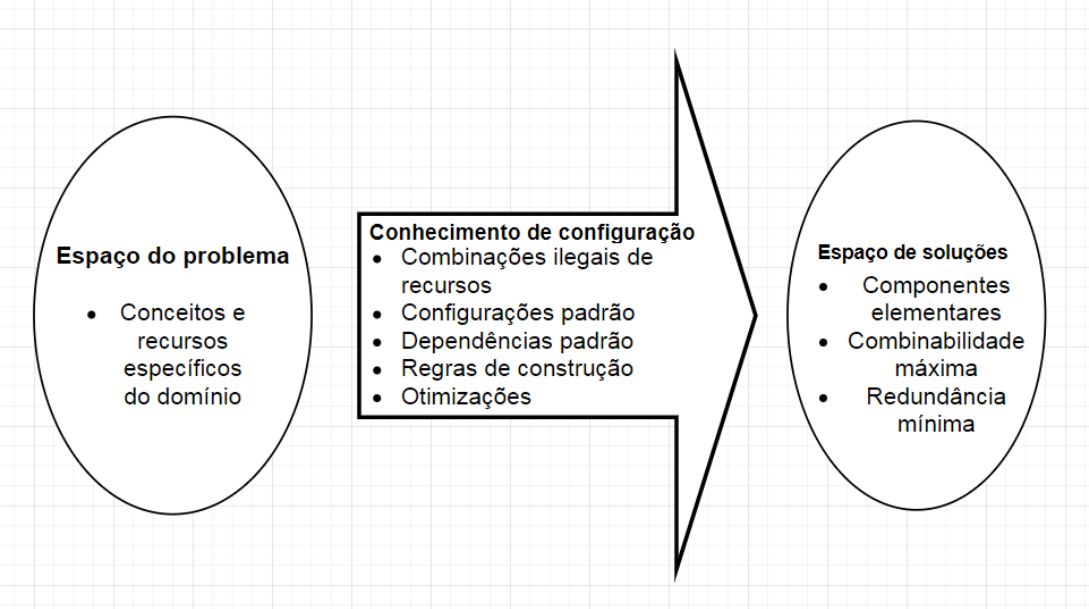
1. No “Espaço do problema”, os usuários utilizam de uma *Domain-Specific Language* (DSL), para representar um sistema de forma mais consistente, por meio de textos, formulários ou uma representação gráfica do sistema.

2. No “Conhecimento de configuração” a montagem do produto é automatizada por um gerador de configuração, a partir de uma especificação DSL com os detalhes dos componentes de software a serem produzidos.

3. No “Espaço de soluções” encontram-se os componentes de software desenvolvidos, a partir dos quais pode-se montar um sistema por meio da combinação de tais componentes,

os quais devem possuir combinabilidade máxima e redundância mínima.

Figura 1 – Modelo de Domínio Generativo.



Fonte – Adaptado de Czarnecki e Eisenecker (1999).

2.3.2 Geradores

Geradores de código, ou simplesmente Geradores, são ferramentas capazes de gerar código de maneira automática a partir de modelos e *templates*. Estas ferramentas automatizam grande parte do trabalho repetitivo ou complexo, otimizando o tempo do programador ou proporcionando ao usuário acesso à informação sem a necessidade de saber o complexo processo até chegar a essa informação (ARNOLDUS *et al.*, 2012).

O uso de geradores de código apresenta vários benefícios para o programador e também para a aplicação gerada, sendo alguns deles (HERRINGTON, 2003):

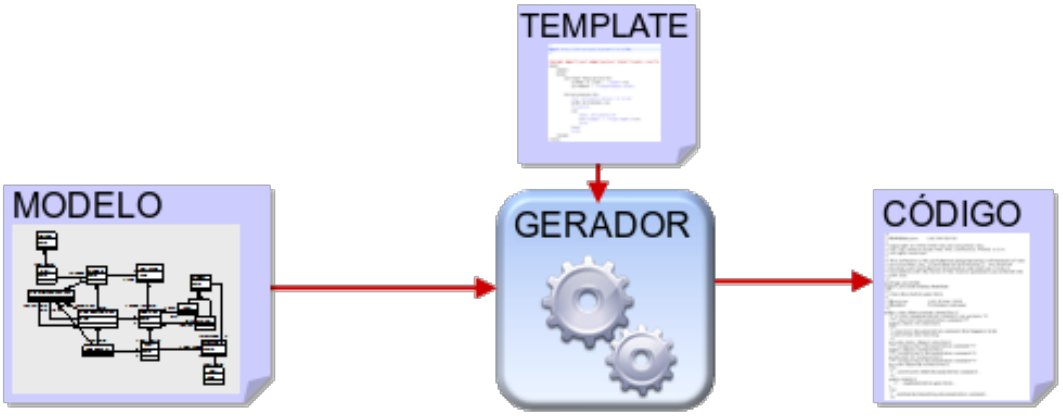
- **Qualidade** – Códigos volumosos gerados à mão tendem a ter inconsistências e erros devido a inúmeros motivos, tais como, por exemplo, mudanças de abordagens de durante o período de desenvolvimento. Com o uso de Geradores, o código se torna mais consistente e as trocas de abordagens não afetam a qualidade do código, pois é alterado apenas o modelo;
- **Consistência** – O código desenvolvido por um gerador é consistente ao projeto da *Application Programming Interface* (API), tornando a interface mais fácil e prática de se usar;
- **Ponto único de conhecimento** – Torna as mudanças no sistema mais rápidas e precisas. Uma mudança no arquivo de esquema gera um efeito cascata que altera todos os demais ar-

quivos, completando a mudança em todo o sistema de forma automática sem a necessidade de alterar muitos arquivo à mão;

- **Mais tempo de Projeto** – O tempo de um projeto com a geração de código é menor comparado a um projeto codificado manualmente, e este ganho de tempo favorece o desenvolvedor, permitindo o uso desse tempo para projeto, teste e adequações do sistema, que muitas vezes não recebem a devida atenção em projetos codificados à mão;
- **Arquitetura consistente** – Um gerador de código em um projeto reflete as decisões de arquitetura tomadas no ciclo de desenvolvimento. Isto ajuda a manter um padrão de arquitetura para todo o sistema;
- **Desenvolvimento ágil** – A maleabilidade é um recurso chave nos geradores de código, e contribui em tornar o sistema mais acessível a mudanças e atualizações durante um longo prazo.

A Figura 2, a seguir, exemplifica de maneira simples o funcionamento de um Gerador. No centro do processo se encontra a ferramenta de geração (Gerador). Para dar inicio ao processo de geração de código o gerador recebe como entrada um arquivo com as especificações em alto nível do sistema (Modelo), muitas vezes se tratando de um Diagrama do tipo *Unified Modeling Language* (UML) ou um Esquema de Banco de Dados. O gerador também recebe um arquivo que contém uma estrutura base para auxiliar na criação do código (*Template*). Após receber essas informações, acontece o processo de geração de código dentro do gerador, e a saída final é o código criado a partir de todas as especificações recebidas (Código).

Figura 2 – Esquema geral do funcionamento de um Gerador de Código.



Fonte – Adaptado de SogetiLabs (2013).

2.4 Ambientes de Simulação Automáticos

A concepção do termo Ambientes Automáticos se dá pela necessidade de simplificar o processo de simulação, criando assim ferramentas capazes de automatizar grande parte do

processo de criação de uma simulação, facilitando tal processo para o usuário (BRUSCHI, 2002). Tais ambientes possuem características próprias que os diferem das demais ferramentas. A programação visual é que mais se destaca, pois os ambientes automáticos contam com uma interface gráfica que proporciona toda uma representação do modelo e interação com a simulação para o usuário.

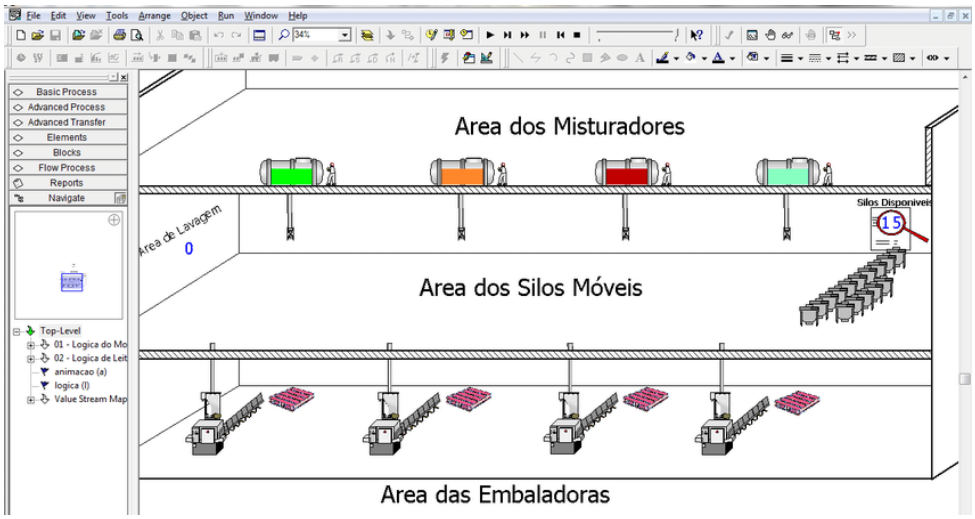
O uso da simulação vem se expandindo e fazendo parte de muitas áreas além da Computação. Cada vez mais investe-se na criação de ambientes de simulação, a maioria sendo software comerciais. Os próximos tópicos apresentados trarão exemplos de ambientes de simulação comerciais e acadêmicos.

2.4.1 ARENA

O ARENA[1] foi desenvolvido pela empresa americana Systems Modeling no ano de 1993. O software resulta da junção do SIMAN (software de simulação para computadores) e do CINEMA (software de animação para computadores), que também pertencem à mesma empresa americana, possibilitando a criação de um dos primeiros softwares de simulação com interface gráfica. A interface gráfica do ARENA proporciona ao usuário uma experiência mais interativa e facilita a criação de modelos de simulação, pois o ARENA possui um conjunto de blocos para descrever uma aplicação real, e esses blocos representam os comandos de uma linguagem de programação. Desta forma, o usuário não precisa utilizar de programação para construir seu modelo, podendo tudo ser feito de forma simplificada e intuitiva por meio da interface gráfica (PRADO, 2014).

A Figura 3 ilustra uma simulação no ambiente gráfico do ARENA.

Figura 3 – Tela do Software ARENA.



Fonte – Oliveira, Correa e Nunes (2014).

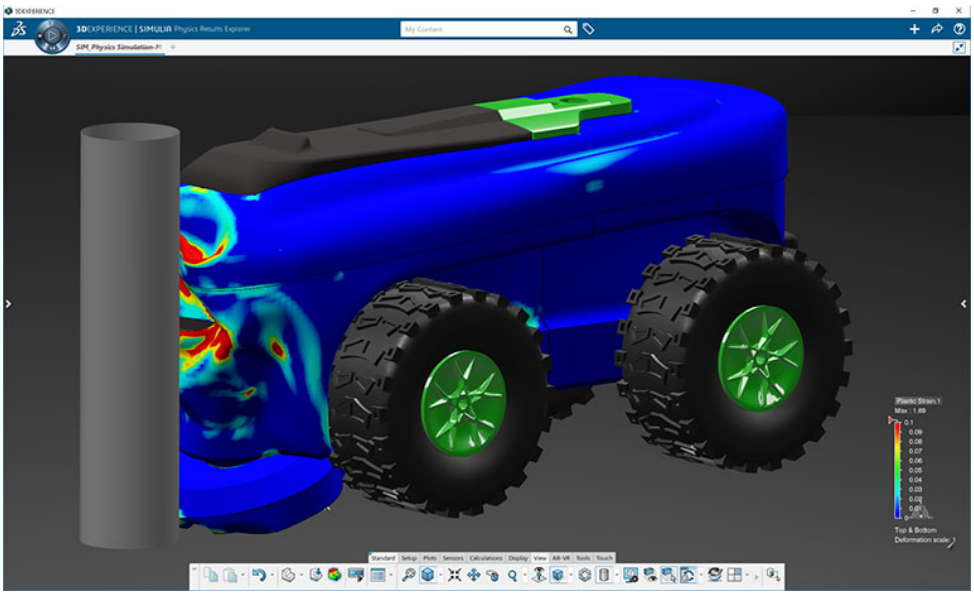
[1] <https://www.paragon.com.br/arena-academico-student/>

2.4.2 SOLIDWORKS Simulation

O SOLIDWORKS Simulation[2] originalmente foi lançado em 1995 com o nome de COSMOSWorks por meio de uma parceria entre a empresa *Structural Research & Analysis Corporation* (SRAC) e a SOLIDWORKS Corporation. Anos depois, adquiridos pela Dassault Systèmes, o COSMOSWorks foi renomeado para SOLIDWORKS Simulation. O SOLIDWORKS Simulation é integrado ao SOLIDWORKS CAD, permitindo ao usuário a criação e edição dos modelos que serão usados para gerar a simulação. O ambiente de simulação comercial realiza análises estruturais utilizando *Finite element analysis* (FEA) para prever o comportamento real de um produto, a partir de um modelo de *Computer-aided design* (CAD) (KUROWSKI, 2015).

A Figura 4 apresenta a interface do SOLIDWORS Simulation.

Figura 4 – Tela do Software Solidworks.



Fonte – Systèmes (2020).

2.4.3 JMP

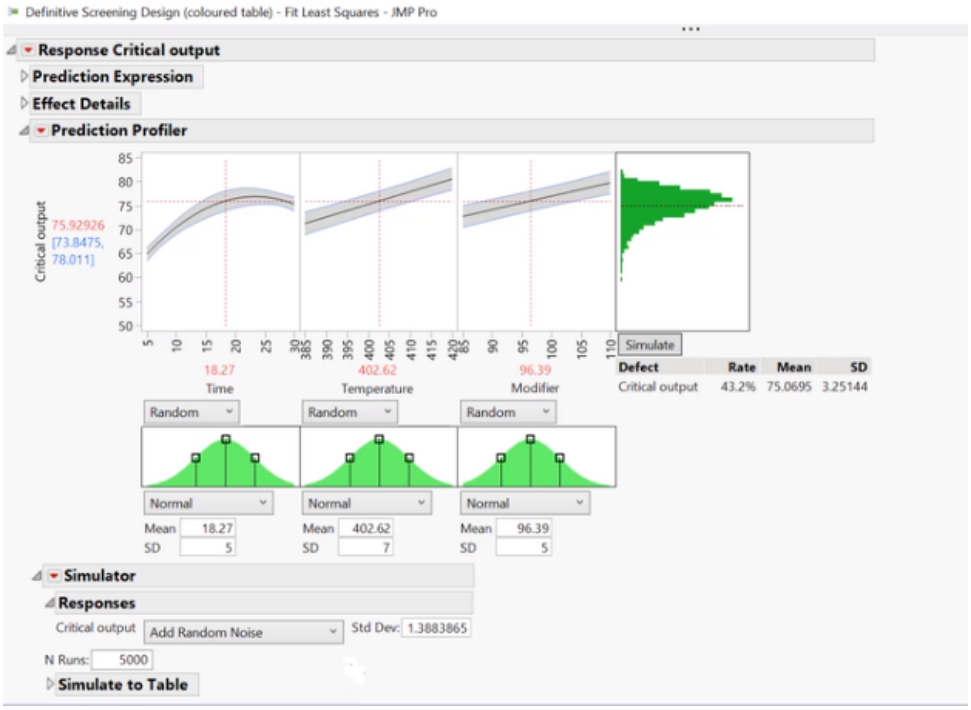
O JMP [3] é um software estatístico desenvolvido pela empresa SAS Intitute em 1989. A ferramenta permite a análise de dados e atualmente está disponível para os sistemas operacionais Mac e Windows (JONES; SALL, 2011).

O JMP conta com a interatividade como uma de suas principais características, disponibilizando a seus usuários uma interface gráfica que permite ver todas as estatísticas em forma de gráficos, o que facilita a análise e exploração de dados, gerando uma melhor compreensão dos resultados e levando a descobertas estatísticas. Dentre os muitos recursos oferecidos pelo JMP,

[2] <https://www.solidworks.com/pt-br/domain/simulation>
[3] <https://www.jmp.com/>

está a possibilidade de gerar e executar simulações (SALL *et al.*, 2017). A Figura 5 ilustra uma simulação que permite descobrir a distribuição das saídas do modelo de dados.

Figura 5 – Tela do Software JMP.



Fonte – SAS (2020).

2.4.4 Cisco Packet Tracer

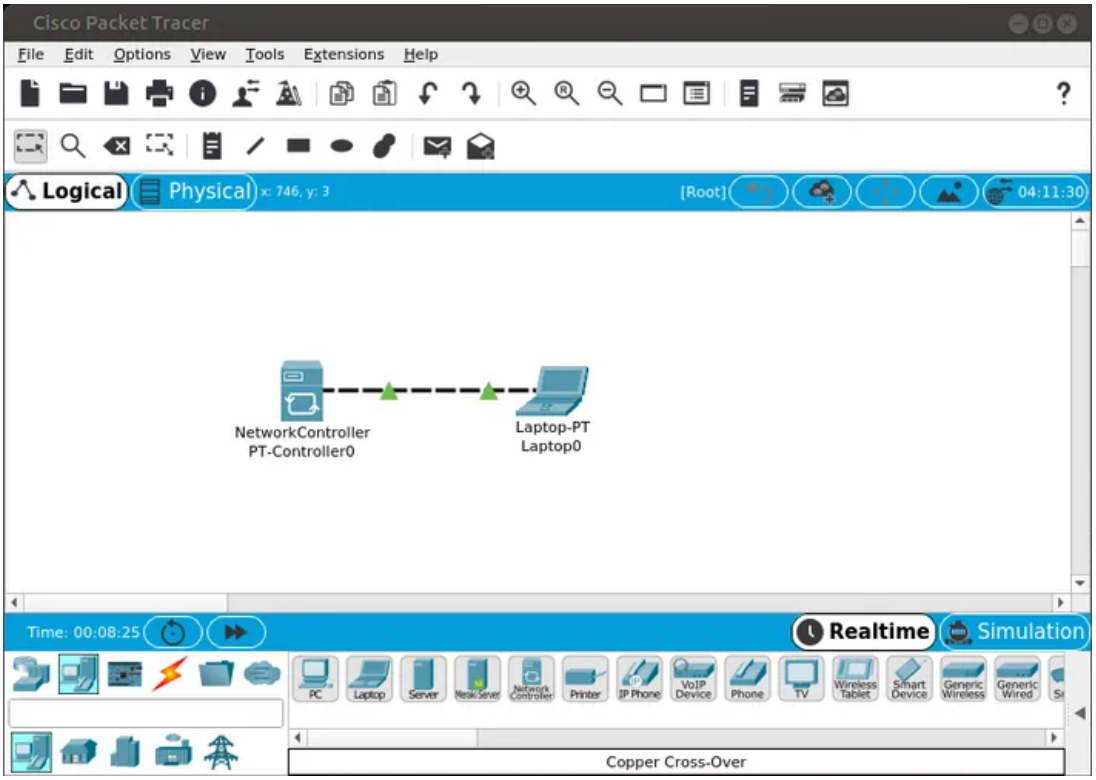
O Cisco Packet Tracer[4] é um software de "simulação, visualização, colaboração e avaliação para o ensino de redes" (NETWORKING, 2020) criado pela empresa Cisco. O software foi proposto pela *Cisco Networking Academy Program* (CNAP) como uma forma de promover o ensino e aprendizagem de redes de computadores para diferentes públicos, como estudantes, educadores, entusiastas da área de redes e os demais interessados (NOOR; YAYAO; SULAIMAN, 2018). A CNAP distribui o Packet Tracer de forma gratuita apenas para fins acadêmicos, e também oferece um curso introdutório para iniciantes.

O Packet Tracer está disponível para Windows e Linux, e oferece aos usuários a oportunidade de se familiarizarem com diversos dispositivos, como roteadores, *switches*, computadores, entre outros, tudo em uma plataforma amigável. Tais facilidades fazem com que o Packet Tracer seja muito utilizado nas universidades para auxiliar o ensino de redes de computadores (NOOR; YAYAO; SULAIMAN, 2018).

A Figura 6 apresenta a interface do Cisco Packet Tracer 8.0.0.

[4] <https://www.netacad.com/pt-br/courses/packet-tracer>

Figura 6 – Tela do Software Cisco Packet Tracer.



Fonte – Network (2021).

2.5 ASDA

O ASDA é um ambiente de simulação proposto para atender demandas de diversos públicos-alvos interessados em avaliação de desempenho de sistemas computacionais, como alunos, professores e pesquisadores. Sua proposta é ser uma forma rápida e fácil de se obter simulações utilizando modelos que assemelham ao sistema real.

Possuindo as principais características dos ambientes de simulação (como uma interface intuitiva e de fácil uso, e a geração de código de simulação), o objetivo do ASDA é prover um ambiente para desenvolver simulações de forma automática, auxiliando os usuários com diferentes níveis de conhecimento sobre simulação de sistemas computacionais (BRUSCHI, 2002).

2.5.1 Módulos

Para facilitar o entendimento e funcionamento, o ASDA foi dividido em diversos módulos que possuem funcionalidades próprias. A seguir serão apresentados tais módulos:

- **Módulo de Interface com o usuário:** Este módulo oferece ao usuário acesso às funcionalidades do ambiente por intermédio de uma interface, a qual contém um editor gráfico

que possibilita ao usuário maior flexibilidade para criar seus modelos de acordo com suas necessidades. Para a interação entre máquinas e seres humanos a interface é muito importante, pois é o que o usuário entende como sistema. Alguns requisitos precisam ser levados em consideração para se ter uma interface de boa qualidade, tais como: facilidade de aprendizado e facilidade de uso; flexibilidade; simplicidade; interface adequada ao usuário que utilizara o sistema; dentre outros (BRUSCHI, 2002).

- **Módulo Avaliador:** Responsável por auxiliar o usuário na tomada de decisões para realização da simulação da melhor forma. O módulo avaliador é dividido em três níveis. O primeiro é responsável por verificar se o modelo foi especificado corretamente pelo usuário; o segundo orienta na escolha da simulação sequencial ou distribuída; e o terceiro orienta na escolha dos protocolos de simulação. Se no nível dois for escolhido simulação distribuída, o módulo também tem a capacidade de decidir as melhores especificações para simulação sem a necessidade de escolha do usuário (BRUSCHI, 2002).

- **Módulo Gerador:** A partir do modelo montado pelo usuário no editor gráfico e das especificações escolhidas no módulo avaliador, é gerado o código do programa de simulação. O módulo conta com o auxílio de um *template* e de bibliotecas de simulação para gerar os códigos (BRUSCHI, 2002).

- **Módulo Executor:** Este módulo é responsável pela execução do código de simulação criado pelo módulo gerador, e a simulação ocorre de acordo com o escolhido no módulo avaliador, ou seja, simulação sequencial ou simulação distribuída (BRUSCHI, 2002).

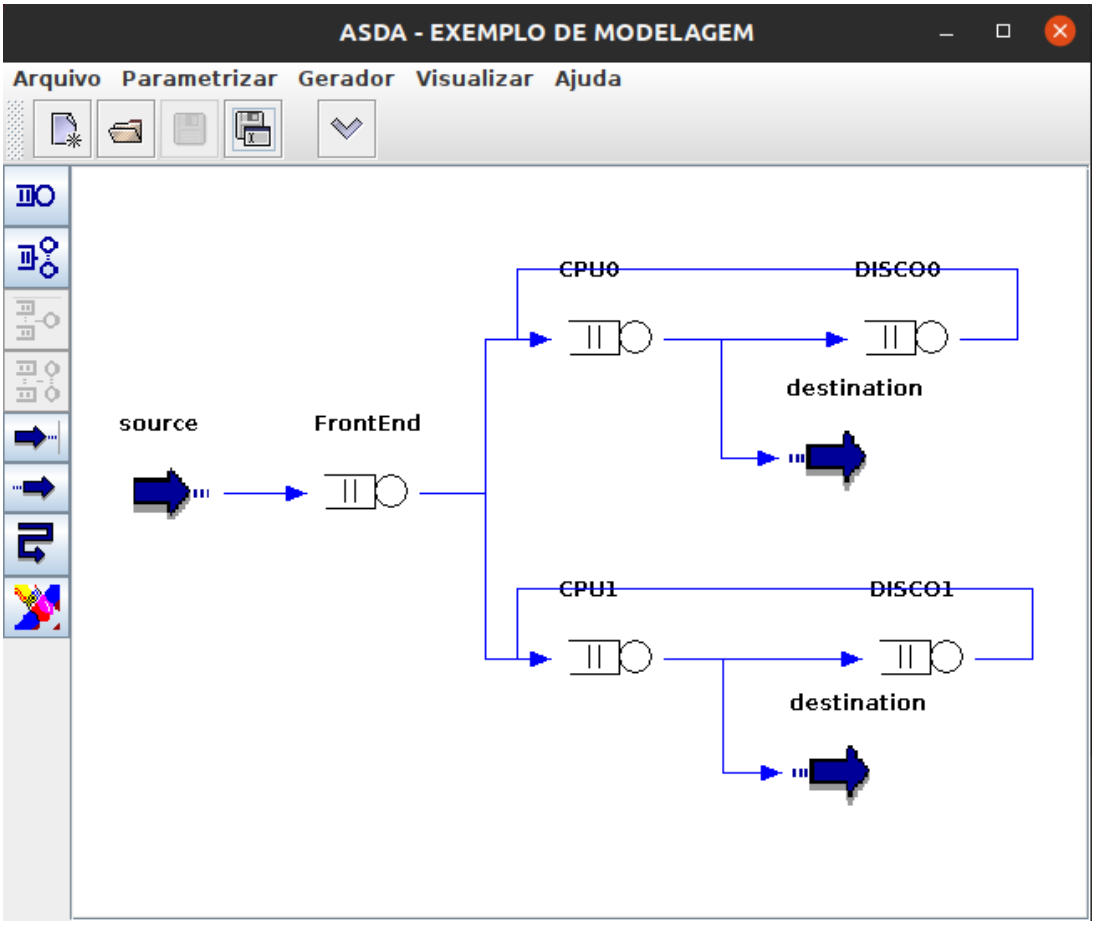
- **Módulo Escalonamento:** Este módulo tem a finalidade de escalonar os processos criados no módulo executor, distribuindo entre os recursos de processamento da máquina (BRUSCHI, 2002).

- **Módulo Replicador:** Este módulo é acionado quando é escolhida a simulação distribuída com a abordagem *Multiple Replication in Parallel* (MRIP), para garantir todas as replicações da execução (BRUSCHI, 2002).

- **Módulo de Interface de Software:** Este módulo fornece uma interface de comunicação entre os programas de simulação já desenvolvidos com o módulo replicador, garantindo assim que o programa seja alterado para ser executado com as abordagem MRIP (BRUSCHI, 2002).

A Figura 7 ilustra um exemplo de modelagem gráfica de um sistema computacional desenvolvido no ambiente de simulação ASDA.

Figura 7 – Tela do Ambiente de Simulação ASDA.



Fonte – Elaborada pela autora.

2.6 Considerações Finais

Neste capítulo foram apresentados conceitos considerados fundamentais para este trabalho. Buscou-se, nesse sentido, absorver um conhecimento aprofundado sobre geração automática de código, além de adquirir um conhecimento geral a respeito do cenário estudado, Ambientes de Simulação Automáticos. Vale destacar que os conceitos apresentados neste capítulo agregaram a possibilidade de identificar e explorar com clareza as oportunidades de desenvolvimento, bem como facilitar o processo de simulação para o usuário. O próximo capítulo apresenta os trabalhos científicos que possuem temas relacionados ao projeto desenvolvido.

CAPÍTULO

3

REVISÃO DA LITERATURA

3.1 Considerações Iniciais

Neste capítulo são abordados alguns dos principais trabalhos que contribuíram para o desenvolvimento deste projeto. Os trabalhos apresentados neste capítulo seguem a mesma base da proposta desta pesquisa de geração de código para realização de simulações automáticas. Na Seção 3.2 é apresentado como foi conduzida a seleção dos estudos primários mais importantes no contexto deste projeto. Na Seção 3.3 foi realizada uma análise dos trabalhos que foram selecionados para a extração de dados referentes ao contexto do trabalho.

3.2 Revisão da Literatura

Para a realização da revisão da literatura foi escolhida a técnica de mapeamento da literatura para identificar os principais trabalhos relacionados ao tema proposto. A pesquisa foi realizada nas principais bases de dados relacionadas a ciência da computação:

- Scopus[1];
- IEEE Xplore[2];
- ACM Digital Library[3].

A fim de se obter uma busca estruturada e objetiva, o Quadro 1 foi constituída possuindo as principais palavras chaves e seus sinônimos relacionados ao tema da pesquisa, para auxiliar na formação de uma String de busca. A seleção levou em consideração todos os trabalhos que possuíam uma ou mais palavras chaves em seu título, *abstract* ou palavras chaves.

[1] <https://www.scopus.com>
[2] <http://ieeexplore.ieee.org>
[3] <http://dl.acm.org>

Quadro 1 – Palavras Chaves de Pesquisa.

Palavras Chave	**Sinônimos**	**Abreviações**
Generator	Code Generator	-
Simulation	Discrete Simulation, Discrete-event Simulation	-
Code Generation	Automatic Code Generation, Source Code Generation	-

A seguir é apresentada a String de busca elaborada que contém operadores lógicos que auxiliam a filtrar os resultados da busca para os trabalhos que mais se encaixam com o tema proposto.

(“Generator” **OR** “Code Generator”) **AND** (“Simulation” **OR** “Discrete Simulation” **OR** “Discrete-event Simulation”) **AND** (“Code Generation” **OR** “Automatic Code Generation” **OR** “Source Code Generation”)

Seleção de Estudos Primários

Após a realização das buscas nas bases de dados, é imprescindível a execução da fase de seleção de estudos primários, a fim de filtrar nos resultados das buscas os trabalhos de maior relevância para o projeto. Critérios de inclusão e exclusão devem ser definidos para auxiliar o processo de filtragem que é constituído de três etapas, sendo que em cada etapa é possível filtrar melhor os trabalhos de maior relevância.

1. **Leitura de Títulos e Resumos:** Nesta etapa é realizada a leitura dos títulos e resumos de todos os estudos resultados da busca. Os trabalhos que possuem elementos correlatos ao tema são selecionados para a segunda etapa da seleção e os demais são descartados;
2. **Leitura das seções de Introdução e Conclusão:** Nesta etapa, para averiguar se os trabalhos selecionados na etapa anterior possuem relevância para o tema do projeto, é realizada a leitura das seções de introdução e conclusão;
3. **Leitura completa dos trabalhos:** Nesta etapa é realizada uma leitura completa dos trabalhos selecionados na etapa anterior, com o intuito de extrair as informações fundamentais necessárias à pesquisa.

Critérios de Inclusão

CI 1: Estudos que abordem aspectos de simulação de sistemas computacionais;
CI 2: Estudos que abordem aspectos de geração de código baseado em *templates*.

Critérios de Exclusão

CE 1: Estudos que não abordam Simulação;
CE 2: Estudos que não abordam Geração de Código;
CE 3: Estudos que não possuam versões completas disponíveis;
CE 4: Artigos que não sejam focados nas áreas de Ciência e Engenharia da Computação;
CE 4: Resultados correspondentes aos capítulos de livros (Estudos secundários);
CE 5: Resultados correspondentes a literatura cinza (Estudos terciários).

Nas Tabelas 1 e 2 são apresentadas as possíveis conferências e periódicos para a publicação dos resultados deste projeto.

Tabela 1 – Principais Conferências para Publicação dos Resultados

Conferência	Sigla	Qualis
EEE International Parallel & Distributed Processing Symposium	IPDPS	A1
Winter Simulation Conference	WSC	A2
International Symposium on Code Generation and Optimization	CGO	A2
IEEE International Symposium on Performance Analysis of Systems and Software	ISPASS	A2
European Conference on Modelling and Simulation	ECMS	B1
International Conference on Simulation Tools and Techniques	SIMUTOOLS	B1
International Conference on Computer Modelling and Simulation	UKSIM	B1
International Conference on High Performance Computing & Simulation	HPCS	B1
ACM International Conference on Generative Programming: Concepts and Experiences	GPCE	B2
International Conference on Simulation and Modeling Methodologies, Technologies and Applications	SIMULTECH	B2
International Workshop on Power and Timing Modeling, Optimization and Simulation	PATMOS	B3
International Symposium on Performance Evaluation of Computer and Telecommunications Systems	SPECTS	B5

Tabela 2 – Principais Periódicos para Publicação dos Resultados

Periódicos	ISSN	Qualis
IEEE Transactions on Parallel and Distributed Systems	1045-9219	A1
Performance Evaluation	0166-5316	A2
Future Generation Computer Systems	0167-739X	A2
Journal of Computer and System Sciences	1090-2724	A2
Simulation Modelling Practice and Theory	1569-190X	B1
International Journal of Simulation: Systems, Science and Technology	1473-8031	B1
Simulation (San Diego, Calif.)	0037-5497	B1
ACM Transactions an Modeling and Computer Simulation	1049-3301	B1
International Journal Of Distributed Systems and Technologies	1947-3532	B2
IEEE Latin America	1548-0992	B4

3.3 Análise dos Trabalhos

Nesta seção de análise dos trabalhos foram agrupados os principais trabalhos relacionados ao tema proposto neste projeto de mestrado, sendo discutido o que foi proposto e desenvolvido em cada trabalho. A ordem adotada para a apresentação dos trabalhos foi a partir dos mais relevantes e conceitualmente próximos à proposta do projeto de mestrado.

Spolon (1994) apresenta uma discussão sobre os temas de simulação, ambientes de simulação e geradores de aplicação. Ao explorar cada tema, são discutidas lacunas no estado da

arte que podem ser exploradas. O trabalho propõe o desenvolvimento de um gerador de aplicação para simulação de sistemas discretos, com objetivo principal de proporcional ao usuário uma forma de criar programas de simulação de maneira automática, de forma que o usuário não precise necessariamente ter conhecimento prévio sobre programação.

O gerador é composto por uma interface textual básica para testes e um módulo responsável pela geração dos códigos. O módulo de geração utiliza a especificação textual do modelo, uma extensão funcional (biblioteca de simulação) e a descrição do produto que oferece uma estrutura do código final. Todos os códigos de simulação são gerados na linguagem de programação C, utilizando a biblioteca de simulação SMPL, que é uma extensão funcional desenvolvida para gerar simulação orientada a eventos na linguagem C. Este trabalho pode ser considerado de extrema importância, pois é o trabalho que mais se assemelha à proposta deste projeto de pesquisa.

Vavilina e Gaigals (2015) propõem uma maneira de simplificar a geração de códigos de simulação para o usuário. Ao expor as dificuldades na geração de código de simulação de processamento de sinal paralelo complexo, os autores apresentam o ambiente de simulação gráfica LabVIEW[4], que contém diversas ferramentas para a geração de programas de simulação com alta complexidade, garantindo a exatidão dos códigos.

O trabalho tem o foco em um dos métodos de geração oferecidos pelo LabVIEW, a geração de código usando funções de *script*. Tal método se mostra muito eficaz, mas apresenta desvantagens à medida que os princípios da geração de código são significativamente difíceis de serem compreendidos pelo usuário, dificultando até mesmo a criação de programas de simulação simples. Para resolver esse problema é proposto o desenvolvimento de uma ferramenta chamada Scripting baseada nos *scripts* oferecidos pelo LabVIEW com funções que simplificam ao máximo o aplicativo gerado e diminuem a quantidade de ferramentas utilizadas para a geração de código.

A caixa de ferramentas Scripting também foi submetida a uma avaliação, utilizando um algoritmo clássico para desenvolver os programas de simulação nas funções padrões do LabVIEW e nas funções aprimoradas do Scripting. A avaliação demonstrou resultados positivos, e o Scripting se mostrou fácil de usar e mais compreensível em relação as funções padrões do LabVIEW.

Furlanetto (2016) propõe o desenvolvimento de uma ferramenta capaz de gerar simuladores para diversos contextos que se encaixem em filas básicas. A ferramenta *Yes, a simulator's compiler* (Yasc) foi desenvolvida na linguagem de programação Java com uma arquitetura baseada em módulos, e o principal objetivo do trabalho é suprir a falta que há na literatura de uma ferramenta capaz de permitir ao usuário criar de forma simples e fácil seu próprio simulador de filas.

O funcionamento da ferramenta Yasc é descrito detalhadamente em (FURLANETTO,

[4] <https://www.ni.com/pt-br/shop/labview.html>

2016). O usuário fornece as especificações do simulador que deseja criar por meio da interface gráfica, e o módulo gerador cria, a partir dos parâmetros dados, uma biblioteca de simulação que contém um arquivo de texto com os detalhes de funcionamento da biblioteca e as imagens que os caracterizam. Por último, o módulo interpretador cria a interface do simulador com as informações da biblioteca, e realiza a união com o motor de simulação responsável por realizar as simulações e apresentar os resultados no novo simulador criado.

Para avaliar o Yasc, o autor realizou um teste de usabilidade com usuários reais para caracterizar o grau de eficácia da ferramenta. O teste foi realizado com o auxilio de um questionário capaz de mostrar a satisfação dos usuários quanto ao uso da ferramenta. Além do teste de usabilidade, um estudo de caso também foi realizado. Como resultado do teste de usabilidade, diversos simuladores de redes de computadores foram gerados, e a partir disto foi proposto mais um teste. No segundo teste dois modelos tradicionais foram selecionados e simulados nos simuladores criados no primeiro teste, para realizar a avaliação os dois modelos foram simulados na ferramenta de simulação de redes de computadores NS-3[5], para se obter um parâmetro de comparação.

Xiao *et al.* (2019) apresentam uma proposta de trabalho voltado para a geração de código de simulação baseada em agente para ambientes de hardware heterogêneos. O objetivo do trabalho é cobrir a lacuna que existe nas simulações de ambientes de hardware heterogêneos, causada principalmente porque, devido à grande diversidade de plataformas heterogêneas, é demandado dos pesquisadores profundo conhecimento do hardware, além da necessidade da criação de modelos de especificação únicos para cada ambiente de hardware.

Para atingir tal objetivo, o trabalho contou com o desenvolvimento do backend OpenCL, que é uma extensão para a linguagem de modelagem OpenABL[6] que permite a co-execução de modelos baseados em agentes arbitrários em hardware heterogêneo, e oferece a capacidade de simular espaços baseados em gráficos (modelagem em que o ambiente é descrito por meio de nós e arestas). Também foi implementado um mecanismo semiautomático para realizar correções nas simulações paralelizadas. Para a avaliação, o backend OpenCL foi comparado com outros modelos de simulação existentes, e demonstrou resultados favoráveis, superando os outros modelos em eficiência e desempenho.

3.4 Considerações Finais

De acordo com a revisão da literatura realizada, foi possível compreender que os temas de simulação e geração de código, estão sendo amplamente utilizados na literatura em diversos contextos, além da avaliação de sistemas computacionais. A combinação dos temas tem gerado uma ampla gama de trabalhos que impulsiona os avanços tecnológicos, proporcionando cada vez

[5] <https://www.nsnam.org/>
[6] <https://github.com/OpenABL>

mais qualidade e facilidade no uso da tecnologia.

CAPÍTULO

4

IMPLEMENTAÇÃO DA EXTENSÃO PARA O ASDA

4.1 Considerações Iniciais

Este capítulo apresenta a modelagem e desenvolvimento da extensão para o sistema ASDA. A Seção 4.2 apresenta a escolha das linguagens e bibliotecas selecionadas para implementação. Na Seção 4.3 é discutido sobre o modelo conceitual da arquitetura na qual são expostos os principais componentes e como estes interagem entre si. Na Seção 4.4 são apresentados os diagramas UML desenvolvidos para documentar a ferramenta implementada.

4.2 Seleção das Linguagens de Programação

Nesta etapa de desenvolvimento da ferramenta foi realizado o estudo das linguagens de programação selecionadas e das bibliotecas de simulação. Para a seleção das linguagens de programação a serem implementadas, foram utilizadas as informações dos maiores *ranks* de popularidade de linguagens de programação apresentados a seguir:

- ***PopularitY of Programming Language*:** O rank do PYPL é gerado com base na busca por tutoriais da linguagem de programação na internet (CARBONNELLE, 2020);
- **TIOBE:** O rank do Tiobe é gerado com base em motores de busca populares que contenham o nome da linguagem de programação na sentença de busca, também conta com algoritmos para filtrar os resultados relacionados a linguagem de programação (TIOBE, 2021);
- **GitHub:** O rank do GitHub é gerado com base nas linguagens mais utilizadas na plataforma do GitHub (GITHUB, 2021);

- **RedMonk:** O rank da RedMonk é gerado com base em uma correlação entre as linguagens de programação usadas no Github e as linguagens de programação mais discutidas no StackOverflow (O'GRADY, 2021).

Quadro 2 – Rank das Linguagens de Programação mais Populares

	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º
PYPL	Python	Java	JavaScript	C#	C/C++	PHP	R	Objective-C	TypeScript	Swift
TIOBE	C	Python	Java	C++	C#	Visual Basic	JavaScript	PHP	Assembly	SQL
GitHub	JavaScript	Python	Java	Go	Ruby	C++	TypeScript	PHP	C#	C
RedMonk	JavaScript	Python	Java	PHP	C#	C++	CSS	TypeScript	Ruby	C

O Quadro 2 apresenta as dez primeiras colocações nos *ranks* citados anteriormente. A partir destes dados foram selecionadas as linguagens de programação que têm maior colocação nos *ranks* e que possuem uma biblioteca de simulação baseada em eventos:

- Python;
- Java;
- Java Script;
- C#;
- R.

Para complementar a escolha das linguagens de programação, foram selecionadas as principais bibliotecas de simulação para cada linguagem e apresentadas no Quadro 3 a seguir:

Quadro 3 – Bibliotecas de Simulação Discreta Baseada em Eventos

	Python	**Java**	**JavaScript**	**C#**	**R**
1º	Simpy	JavaSim	SIM.JS	SharpSim	Simmer
2º	Salabim	JSL	SimPackJ/S	NSimulate	R DES package
3º	DE-Sim	SimPackJ/S	-	SimManning	Queuecomputer

Baseando-se nas informações coletadas, foram selecionados apenas as linguagens de programação **Python, Java e R** com as respectivas bibliotecas de simulação **Simpy** [1]**, JavaSim**

[1] <https://simpy.readthedocs.io/en/latest/>

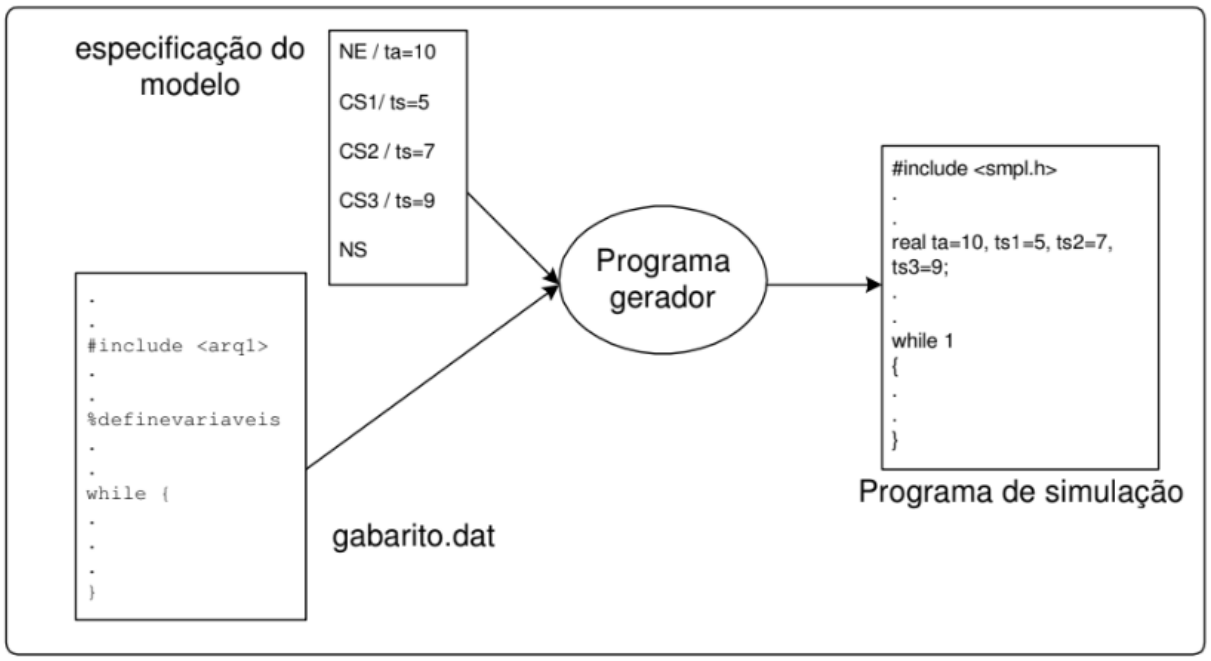
[2] **e Simmer** [3]. Embora a linguagem de programação R só apareça em um *rank*, a mesma possui uma das bibliotecas de simulação discreta mais completa e atualizada.

4.3 Especificações da Extensão

A implementação foi dividida em três partes: Modelos, Gabaritos e os Programas Geradores. Todo o código, bem como as especificações e tutoriais de instalação da ferramenta encontram-se disponíveis e podem ser acessados em <https://github.com/nathlp/extension-generator-asda>. As Telas da ferramenta poder ser vistas no Apêndice F.

A Figura 8 apresenta a estrutura de arquivos do módulo gerador do ambiente de simulação ASDA e a mesma estrutura foi adotada no projeto, a fim de que seja possível posteriormente a integração da implementação. O módulo gerador é composto pelos seguintes elementos:

Figura 8 – Estrutura de arquivos do Módulo Gerador



Fonte – Adaptado de Bruschi (2002)

- **Especificação do Modelo:** A especificação do modelo é um dos elementos principais na geração do código de simulação, pois contém todos os detalhes do sistema a ser simulado;
- **Gabarito.dat:** O arquivo gabarito.dat contém a estrutura do código de simulação que será gerado;
- **Programa Gerador:** O programa gerador é o responsável por ler as informações do modelo e baseado no arquivo gabarito.dat gerar o programa de simulação;

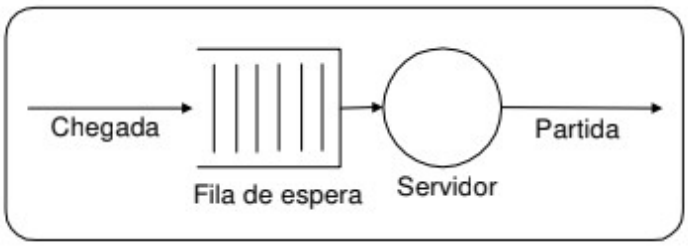
[2] <https://github.com/nmcl/JavaSim>
[3] <https://r-simmer.org/>

- **Programa de Simulação:** O programa de simulação é produto final do processo de geração de código do módulo gerador.

4.3.1 Modelos

A primeira fase da implementação consistiu no desenvolvimento de uma modelagem dos sistemas para que a extensão pudesse ser testada individualmente. A modelagem de sistemas para simulações é uma técnica que coleta as informações do sistema real para criar um modelo condizente, utilizando uma abordagem apropriada para representar o sistema. A abordagem selecionada foi "rede de filas," um ramo da probabilidade que estuda filas e seus principais elementos são os clientes, as filas e os servidores (BRUSCHI, 2002).

Figura 9 – Modelagem de um centro de serviço.



Fonte – Adaptado de Bruschi (2002)

A Figura 9 apresenta a descrição gráfica de um modelo com um único centro de serviço. Um centro de serviço é um conjunto de filas e servidores que representam os recursos do sistema.

Os modelos utilizados foram criados com auxílio da linguagem de descrição de grafos Dot. O arquivo modelo.gv possui a descrição textual de um grafo que representa o sistema, e nesse arquivo é possível descrever os nós (centros de serviço) e as arestas que representam o percurso do cliente no sistema.

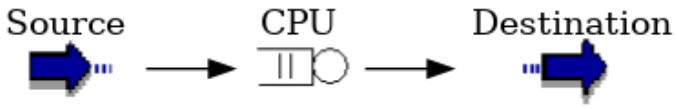
A linguagem Dot foi selecionada para modelagem dos sistemas, pois é de fácil entendimento e consegue representar bem a modelagem de sistemas baseados em redes de fila. Com o auxílio da biblioteca Graphviz[4] do Python é possível gerar os arquivos .gv e uma visualização gráfica que se assemelha aos sistemas modelados na interface gráfica do ASDA. Na Figura 10 é possível observar a representação gráfica gerada pela biblioteca Graphviz, que demonstra um sistema baseado em filas com apenas um centro de serviço (CPU).

4.3.2 Gabaritos

Após o desenvolvimento dos modelos, foram criados os gabaritos específicos para cada linguagem de programação selecionada e suas respectivas bibliotecas de simulação apresentadas

[4] <https://graphviz.org/>

Figura 10 – Representação gráfica gerada pela biblioteca Graphviz.



Fonte – Elaborado pela autora.

anteriormente. O arquivo gabarito.txt é constituído da junção de trechos de código e funções da biblioteca de simulação, tornando-se um *template* para a geração dos códigos de simulação. Para auxiliar o programa gerador o gabarito possui comandos que, ao serem lidos pelo programa gerador, indicam onde as informações importantes do arquivo estão, possibilitando atrelar as informações contidas na especificação do modelo para gerar o código de simulação.

4.3.3 Programas Geradores

O programa gerador é responsável por criar o código de simulação a partir das informações da especificação do modelo e do arquivo gabarito.txt. O programa gerador realiza a leitura do arquivo gabarito.txt, utiliza a estrutura do *template*, identifica as informações necessárias para gerar o programa de simulação e busca no arquivo da especificação do modelo os valores necessários.

A codificação dos programas geradores foi realizada utilizando a linguagem de programação Python com o paradigma de programação orientada a objetos (POO), e foi criado um arquivo.py para cada linguagem selecionada. Para auxiliar na obtenção das informações do arquivo dos modelos.gv foram implementadas três classes: *Graph*, *Nodes* e *Edges*. Essas classes são responsáveis por ler o arquivo do modelo.gv, obter a informações e salvá-las em um objeto, que é utilizado pelos programas geradores.

4.4 Validação

Para validar a ferramenta implementada foram realizados alguns testes nos códigos gerados pela ferramenta. O modelo escolhido para ser testado é um sistema M/M/1, que possui apenas um servidor. Os parâmetros do modelo para resolução analítica são:

- Taxa de Chegada: 0,667 req/s;
- Tempo Médio entre as Chegadas: 1,5 s;
- Tempo Médio de serviço do Servidor: 0.9 s;

Para esse modelo, a resolução analítica apresenta os seguintes resultados:

- Utilização: 0,6 = 60%.

- *Throughput*: 0,667 req/s (Quando o sistema esta em equilíbrio a Taxa de chegada = *Throughput* (Vazão)).

Foram gerados códigos de simulação para o modelo M/M/1 com distribuições exponencias para os valores de tempo médio entre as chegadas de 1,5 segundos e tempo médio de serviço do servidor de 0.9 segundos e para comprovar que o ambiente fornece resultados confiáveis foi realizado a execução dos códigos 20 vezes com sementes de geração de números aleatórios diferentes em cada execução e calculado valores de média, variância, desvio padrão e intervalo de confiança para a utilização e *throughput*.

Nos Quadros 4 e 5, apresentam-se os resultados das execuções realizadas, e é possível observar que as médias obtidas estão muito próximas dos valores esperados pela resolução analítica. Para garantir a confiabilidade desses resultados, foi calculado um intervalo de confiança com um nível de confiança de 95%, proporcionando uma certeza estatística na consistência dos dados obtidos. Isso reforça a validade dos resultados e a precisão das medições realizadas.

Com base nos resultados apresentados, pode-se afirmar que a ferramenta atinge os níveis de confiança estabelecidos. As medições e análises realizadas demonstraram uma consistência e precisão que sustentam a confiabilidade dos dados obtidos. A proximidade das médias com os valores da resolução analítica, aliada ao cálculo do intervalo de confiança com um nível de confiança de 95%, fortalece a confiança na capacidade da ferramenta em produzir resultados confiáveis. Esses resultados reforçam a qualidade e eficácia da ferramenta no contexto em que é aplicada.

Quadro 4 – Resultados *Throughput* das três linguagens

	Média	**Variância**	**Desvio Padrão**	**Intervalo de Confiança**
Throughput - Python	0,66155	0,0011834184	0,0344008491	0,64647344 - 0,67662656
Throughput - Java	0,64215	0,0008582394	0,0292957244	0,62931081 - 0,65498918
Throughput - R	0,654416	0,0011299051	0,0336140615	0,63968505 - 0,66914854

Quadro 5 – Resultados Utilização das três linguagens

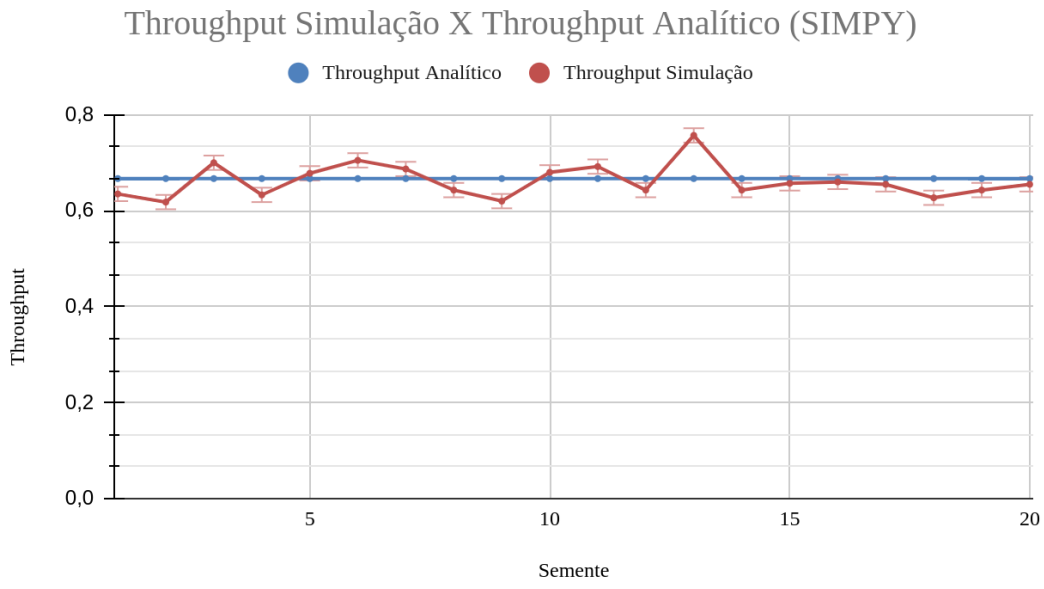
	Média	**Variância**	**Desvio Padrão**	**Intervalo de Confiança**
Utilização - Python	0,60175	0,00186325	0,04316537965	0,58283228 - 0,62066771
Utilização - Java	0,58105	0,000786997	0,02805347338	0,56875524 - 0,59334475
Utilização - R	0,58735	0,002448501	0,04948233886	0,56567320 - 0,60904559

4.4.1 Simpy

O código de simulação gerado para este teste em Python, utilizando a biblioteca de simulação Simpy se encontra no Apêndice G.1. Este código foi executado 20 vezes trocando a semente de geração de números aleatórios em cada execução a fim de se obter um resultado mais preciso.

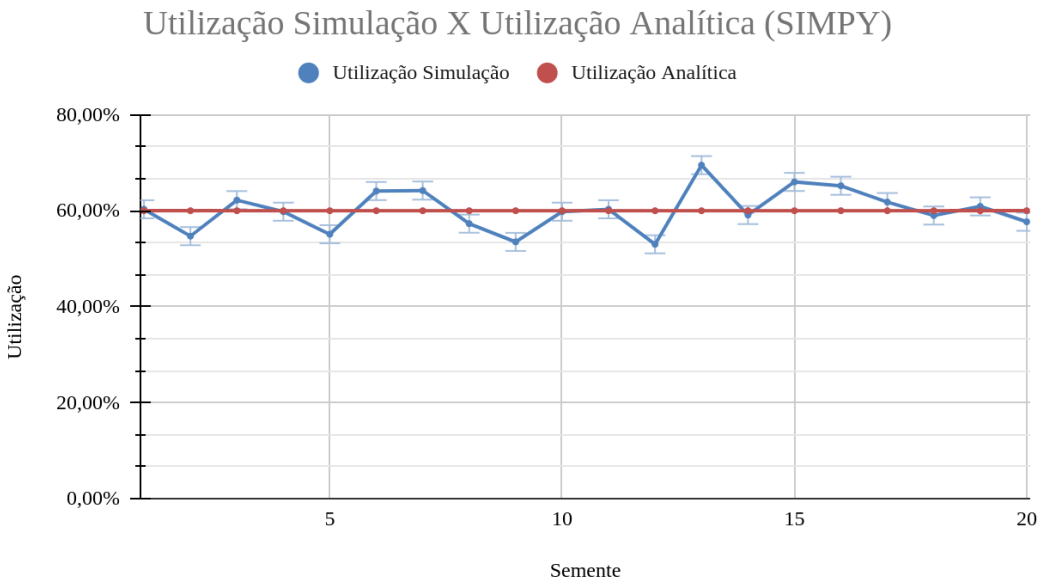
Nas Figuras 11 e 12 é possível observar uma comparação dos valores obtidos no *Throughput* e Utilização, respectivamente, em cada execução do código de simulação gerado em Python, com o valor da resolução analítica.

Figura 11 – Gráfico do Throughput Simpy.



Fonte – Elaborado pela autora.

Figura 12 – Gráfico de Utilização Simpy.



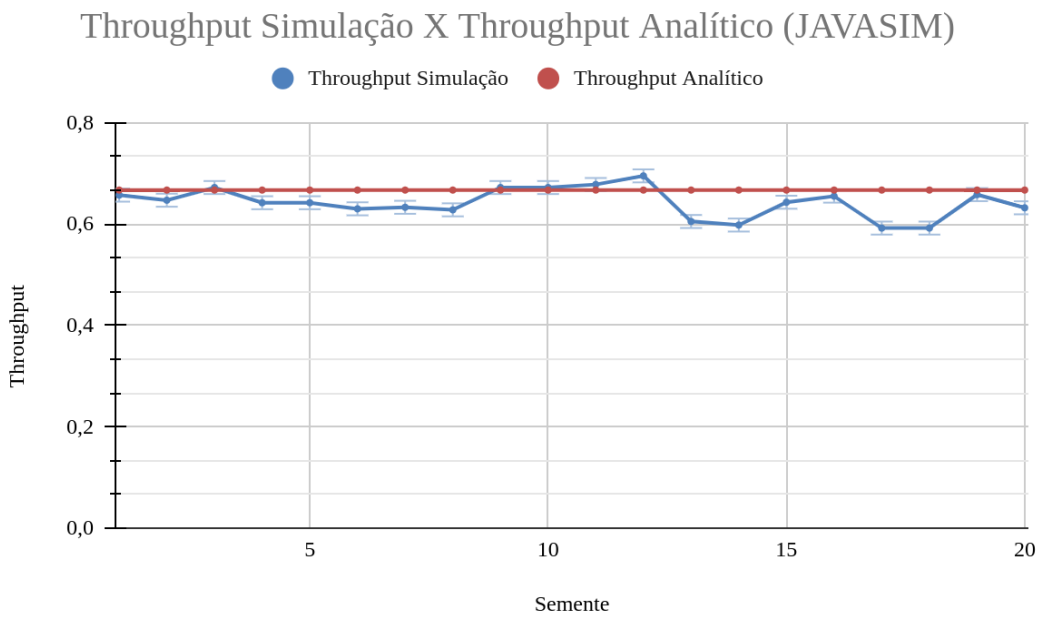
Fonte – Elaborado pela autora.

4.4.2 JavaSim

O código de simulação gerado para este teste em Java e utilizando a biblioteca de simulação JavaSim se encontra no Apêndice G.2. Este código foi executado 20 vezes trocando a semente de geração de números aleatórios em cada execução a fim de se obter um resultado mais preciso.

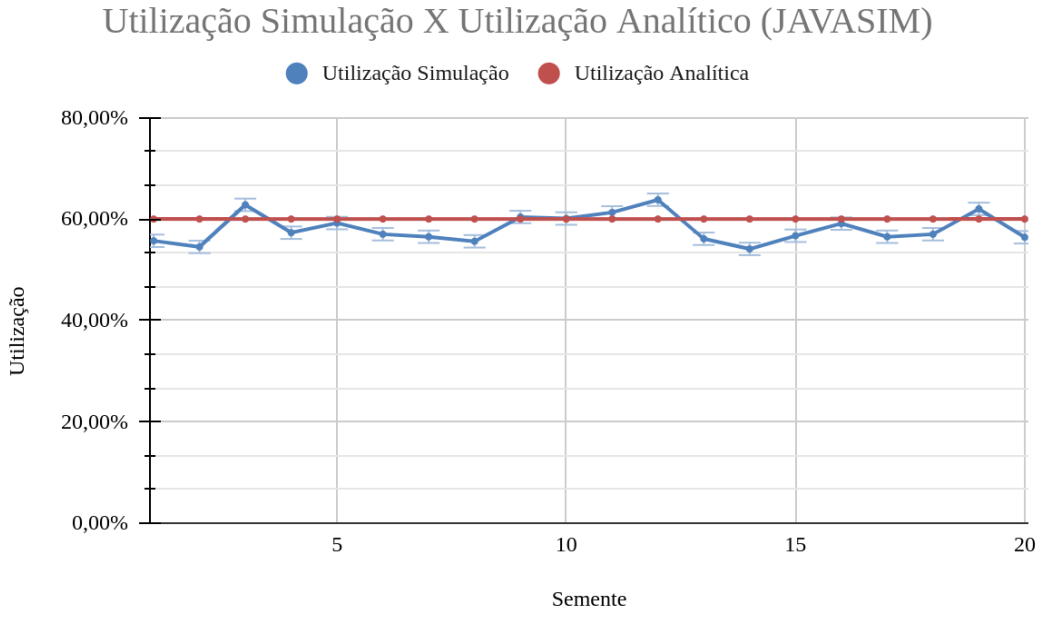
Nas Figuras 13 e 14 é possível observar uma comparação dos valores obtidos no *Throughput* e Utilização, respectivamente, em cada execução do código de simulação gerado em Java, com o valor da resolução analítica.

Figura 13 – Gráfico do Throughput JavaSim.



Fonte – Elaborado pela autora.

Figura 14 – Gráfico de Utilização JavaSim.



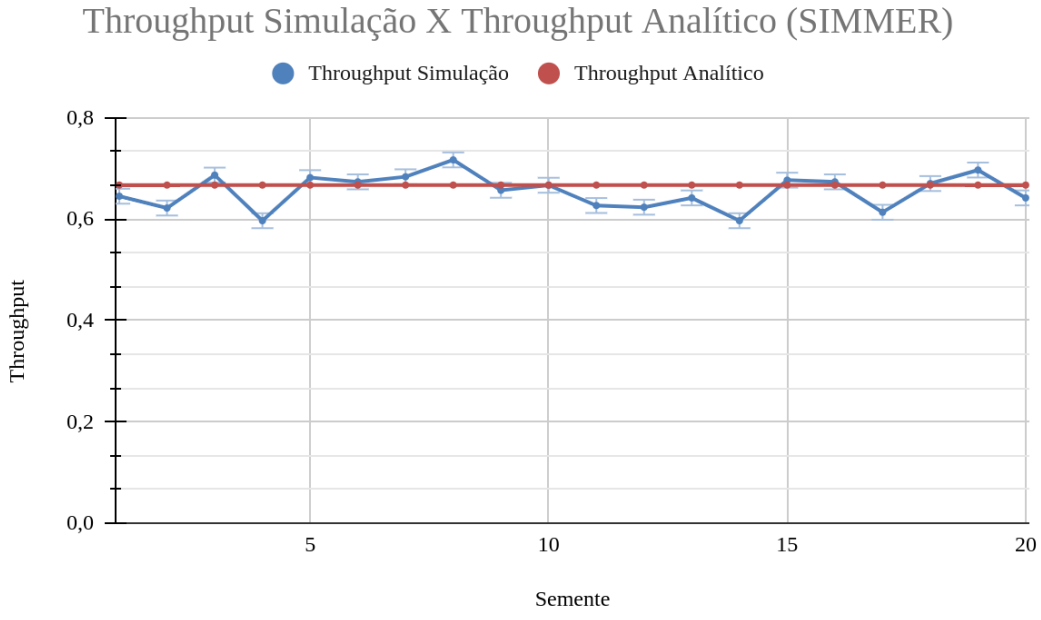
Fonte – Elaborado pela autora.

4.4.3 Simmer

O código de simulação gerado para este teste em R e utilizando a biblioteca de simulação Simmer se encontra no Apêndice G.3. Este código foi executado 20 vezes trocando a semente de geração de números aleatórios em cada execução a fim de se obter um resultado mais preciso.

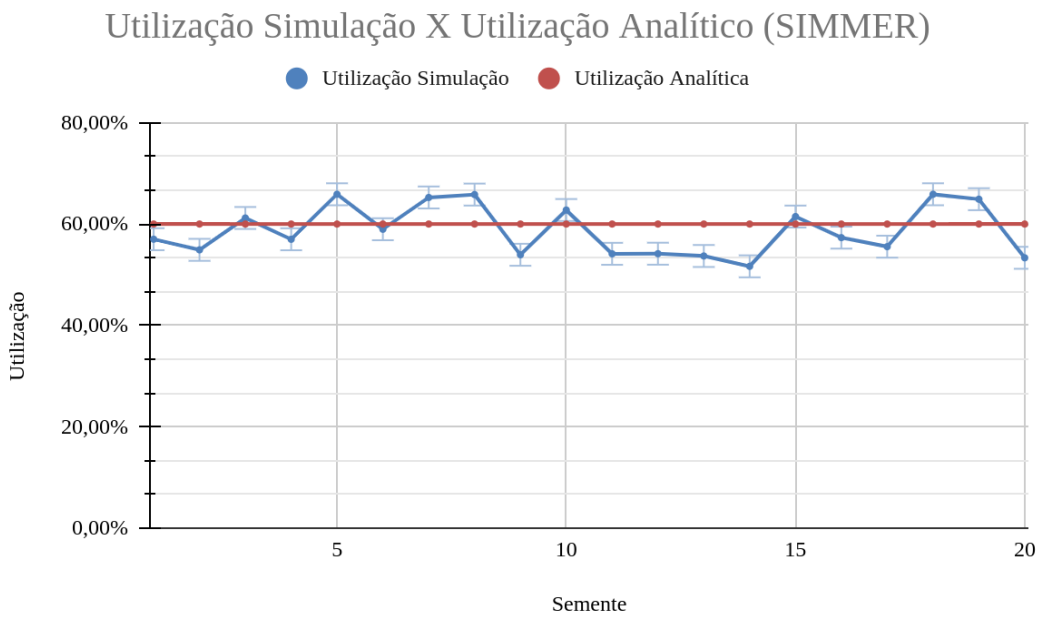
Nas Figuras 15 e 16 é possível observar uma comparação dos valores obtidos no *Throughput* e Utilização respectivamente, em cada execução do código de simulação gerado em R, com o valor da resolução analítica.

Figura 15 – Gráfico do Throughput Simmer.



Fonte – Elaborado pela autora.

Figura 16 – Gráfico de Utilização Simmer.



Fonte – Elaborado pela autora.

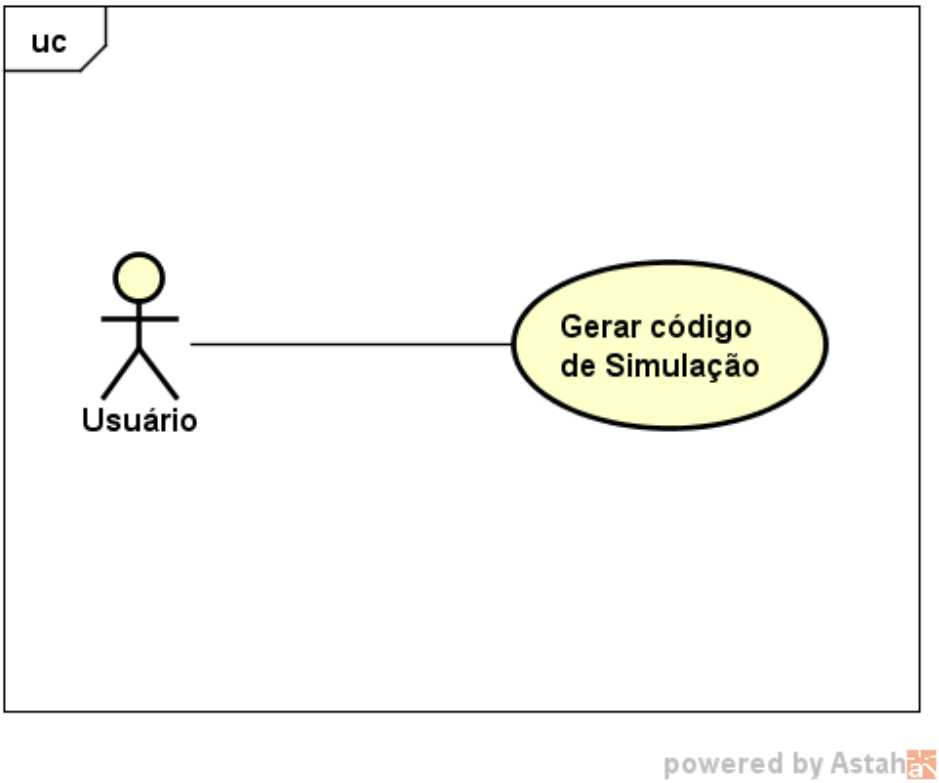
4.5 Diagramas UML

Para melhor detalhar o funcionamento da extensão, foram construídos diagramas de caso de uso, de atividades e de classes, que são apresentados nesta seção.

4.5.1 Diagrama de Caso de Uso e de Atividades

A extensão possui um caso de uso que pode ser representado pelo Diagrama de Casos de Uso na Figura 17. Para descrever o funcionamento deste caso de uso, foi feito um Diagrama de Atividades (Figura 18) que detalha todos os passos que o usuário e o sistema realizam para gerar um código de simulação e uma especificação textual.

Figura 17 – Diagrama de Casos de Uso.



Fonte – Elaborado pela autora.

Caso de Uso: Gerar Código de Simulação

Ator Principal: Usuário.

Objetivo: Usuário gerar um código de simulação na linguagem de programação que deseja.

Garantia de Sucesso: O usuário ter acesso ao código salvo.

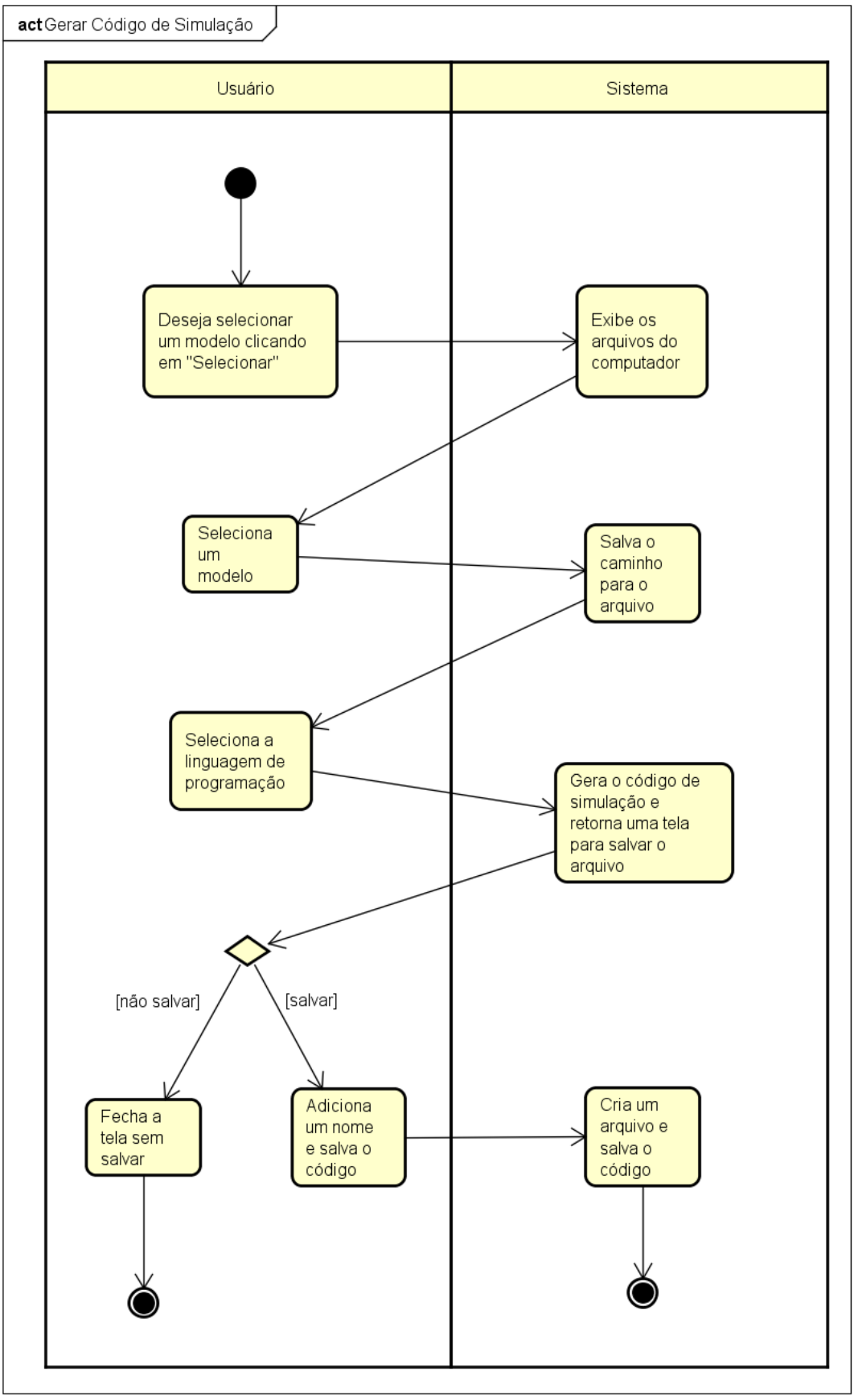
Cenário de Sucesso Principal (ou Fluxo Básico):

1. Na tela principal o usuário deseja selecionar um modelo.
2. O sistema exibe em uma nova tela todos os arquivos do computador.
3. O usuário seleciona um modelo.
4. O usuário seleciona a linguagem de programação.
5. O sistema gera o código de programação baseado nos parâmetros definidos pelo usuário e retorna uma tela para salvar o arquivo.
6. O usuário adiciona um nome ao arquivo e escolhe o local para salvar o código.
7. O sistema cria um arquivo com o código e salva no local definido.

Fluxos Alternativos:

6a. Usuário cancela a operação e fecha a tela sem salvar o arquivo.

Figura 18 – Diagrama de Atividades do Caso de Uso “Gerar Código de Simulação”.

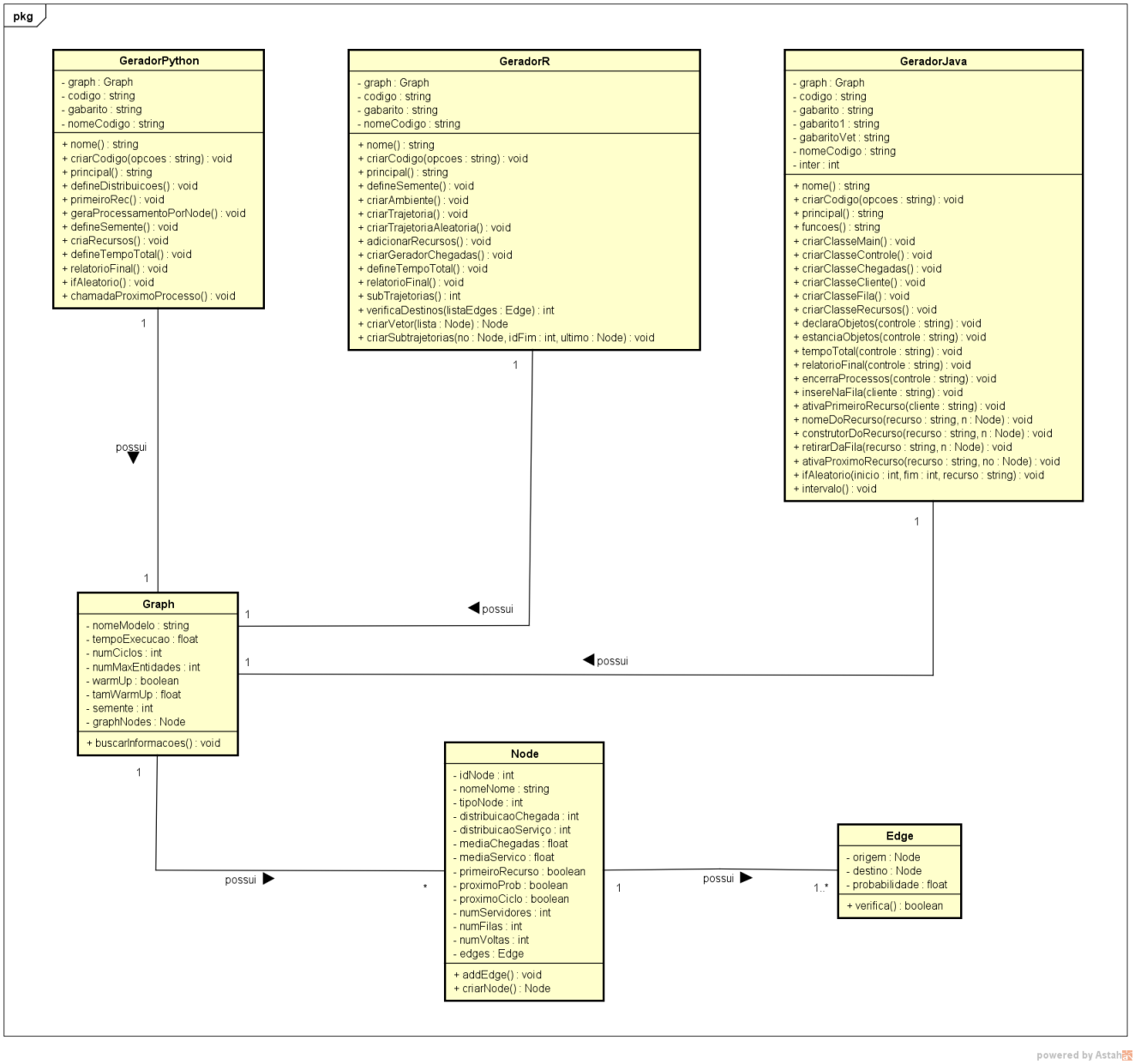


Fonte – Elaborado pela autora.

4.5.2 Diagrama de Classes

A Figura 19 apresenta o Diagrama de Classes da extensão, que permite visualizar os componentes da extensão e como se relacionam entre si.

Figura 19 – Diagrama de Classes.



Fonte – Elaborado pela autora.

4.6 Considerações Finais

Neste capítulo, foram detalhados os aspectos relacionados à implementação da ferramenta, incluindo a escolha das linguagens de programação e bibliotecas utilizadas, bem como a descrição dos componentes e a apresentação dos diagramas UML que documentam os ele-

mentos do ambiente de simulação. Essas etapas foram fundamentais para garantir uma boa funcionalidade da ferramenta.

CAPÍTULO

5

RESULTADOS

5.1 Considerações Iniciais

Neste capítulo, detalhadamente, são descritos os resultados obtidos com a aplicação da ferramenta gerada neste projeto de pesquisa. Na Seção 5.2 apresenta-se uma visão geral dos experimentos e os métodos adotados para sua realização. Nas Seções 5.3 e 5.4 são descritos e explicados em detalhes os experimentos realizados. Os resultados obtidos foram apresentados e analisados minuciosamente, levando em consideração as questões de pesquisa formuladas na Seção 5.3.2.

5.2 Relação de Experimentos

Para alcançar os objetivos propostos no Capítulo 1 foram realizados 2 experimentos:

- Experimento I: Aula de Avaliação de Desempenho de Sistemas ofertada no formato online (2022).
- Experimento II: Aula de Avaliação de Desempenho de Sistemas ofertada presencialmente (2022).

Apesar dos dois experimentos possuírem o mesmo objetivo, foram conduzidos de formas diferentes, a fim de se obter um comparativo dos resultados dos experimentos realizados em condições diferentes de aula, sendo um experimento online e o outro presencial. Para a avaliação optou-se por uma avaliação qualitativa do estudo com a aplicação de questionários de natureza qualitativa desenvolvidos pela pesquisadora.

Neste estudo, o planejamento dos experimentos foi realizado seguindo os fundamentos propostos na publicação de Wohlin *et al.* (2012), com o objetivo de garantir que os experimentos

possam ser reproduzidos e validados de forma confiável. A estrutura conta com os seguintes elementos:

- Escopo;
- Planejamento;
- Execução;
- Resultados;
- Análise.

Nas seções seguintes os experimentos são detalhados de acordo com a estrutura proposta anteriormente.

5.3 Experimento I: Aula Online

Este experimento foi realizado em outubro de 2022, com uma turma do curso de Sistemas da Informação do Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP). O experimento foi realizado de forma online e foi conduzido pela pesquisadora.

5.3.1 Escopo

Objetivo

O objetivo deste experimento é avaliar a utilização e aceitação da ferramenta de geração de código desenvolvida e apresentada no Capítulo 4 para auxiliar o ensino de avaliação de desempenho, visando a diminuição da complexidade do aprendizado no primeiro contato com a área.

Objeto

O objeto de estudo deste experimento é o uso de um ambiente automático de simulação para o ensino de avaliação de desempenho.

Foco Qualitativo

O foco qualitativo do experimento é averiguar os impactos positivos e negativos da utilização do ambiente automático de simulação como auxílio durantes as aulas.

Perspectiva

A perspectiva é que este experimento seja realizado em um ambiente com infraestrutura computacional adequada para a utilização do ambiente automático de simulação.

Contexto

O contexto deste experimento foi a disciplina de Sistemas Operacionais I, oferecida como matéria obrigatória para o curso de Sistemas da Informação (SSC0541) do ICMC/USP, oferecida no 4º período do curso. O estudo foi realizado em outubro de 2022, e contou com uma aula de 1 hora e 40 minutos de duração, com um total de 31 participantes. A aula foi ministrada de forma remota pelo palestrante, entretanto os alunos se encontravam em um laboratório de ensino com computadores disponíveis para o uso durante a aula.

5.3.2 Planejamento

Definição das Questões de Pesquisa

Em função dos objetivos estabelecidos previamente, definiram-se as seguintes questões de pesquisa:

QP1 Qual é a percepção dos alunos sobre a eficácia da ferramenta de geração de código no processo de aprendizagem de Avaliação de Desempenho durante seu primeiro contato com a área?

QP2 Os alunos aprovaram o uso da ferramenta de geração de código para auxiliar no processo de aprendizagem?

Seleção de Sujeitos

A seleção de sujeitos para o desenvolvimento do experimento foi constituída de alunos regularmente matriculados na disciplina de Sistemas Operacionais I, de tal forma que os pesquisadores não exerceram qualquer influência sobre o a seleção destes alunos.

Descrição da Instrumentação

O experimento foi estruturado como uma aula divida em duas partes. Antes da primeira parte da aula foi aplicado um questionário de conhecimento inicial, para se entender o conhecimento dos participantes sobre avaliação de desempenho. Este questionário se encontra no Apêndice A.

A primeira parte tratou de apresentar os principais conceitos da área de Avaliação de Desempenho de Sistemas, focando em simulação discreta baseada em eventos. Na segunda parte, foi apresentado o ambiente de simulação e proposto um exercício, apresentado no Apêndice E.1, para ser resolvido utilizando os conhecimentos adquiridos na primeira parte da aula e contando com o auxílio do ambiente automático. A aula teve uma duração total de 1 hora e 40 minutos e, ao final, foi aplicado um questionário qualitativo a fim de avaliar a percepção e a experiência de uso dos alunos. Este questionário encontra-se no Apêndice B.

Planejamento de Aula

O Quadro 6 apresenta os tópicos teóricos abordados na aula.

Quadro 6 – Planejamento de aula.

Aula	Conteúdo
1	- Conceitos Básicos - Simulação de Sistemas - Redes de Fila - Modelagem de Sistemas - Bibliotecas de Simulação - Apresentação do Ambiente de Simulação - Aplicação do exercício utilizando o Ambiente Automático

5.3.3 Execução

Preparação

Baseado no planejamento da aula foram selecionados os materiais e montado os slides utilizados como apoio durante a aula, e também foi elaborado, descrito e implementado o exercício aplicado durante a aula. Além disso foram realizadas as configurações necessárias para execução do ambiente automático nas máquinas do laboratório de ensino.

Operação

O experimento foi realizado durante o horário de aula da turma de Sistemas Operacionais I, que tinha duração de 1 hora e 40 minutos. Os alunos responderam ao questionário inicial e em seguida foram apresentados todo o conteúdo teórico previsto no Quadro 6 e o ambiente automático. Foi também explicado o uso do ambiente e desenvolvido um código de simulação na ferramenta junto dos alunos para exemplificar o seu uso. Logo após, foi proposta a realização de um exercício e, ao final da aula, foi aplicado o questionário qualitativo.

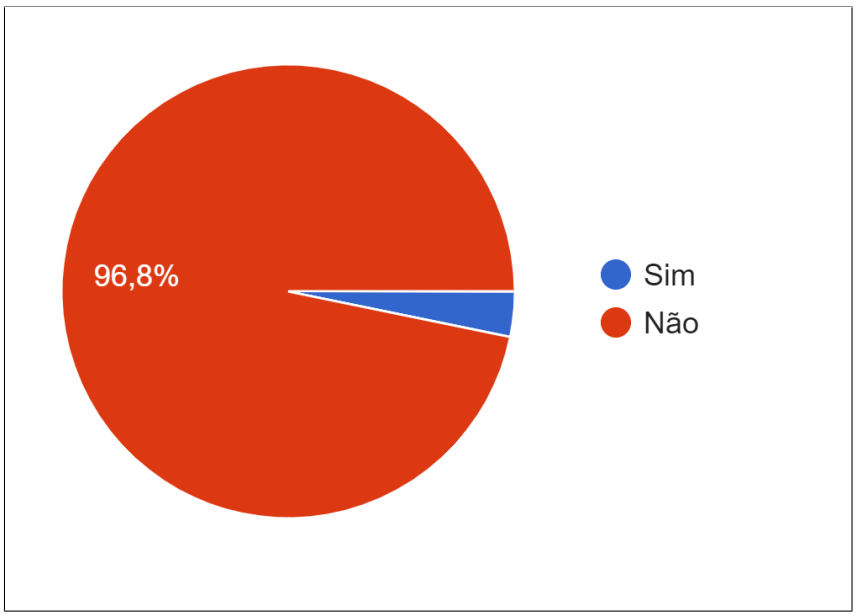
5.3.4 Descrição dos Resultados

Avaliação Inicial

Antes de iniciar o experimento, foi aplicado um questionário inicial online contendo 10 questões, das quais 5 eram questões específicas que possuíam 5 alternativas cada, para situar o conhecimento da amostra sobre o conteúdo da aula. O objetivo do questionário inicial foi coletar informações sobre a familiaridade e compreensão dos participantes em relação ao assunto que seria estudado, a fim de se obter um panorama geral do nível de conhecimento prévio da amostra. Com os resultados do questionário, foram gerados gráficos para melhorar a visualização das informações coletadas, a seguir são apresentados os principais resultados.

5 - Você já cursou a disciplina de Avaliação de Desempenho de Sistemas?

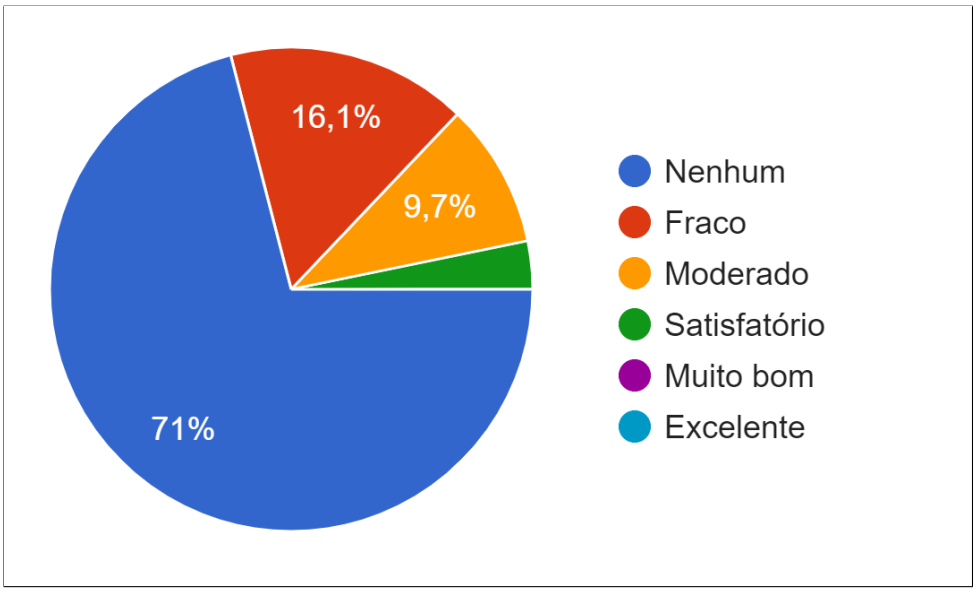
Figura 20 – Resposta da 5º questão do questionário inicial



Fonte – Elaborado pela autora.

6 - Sobre o tema de Avaliação de Desempenho, qual seu nível de conhecimento?

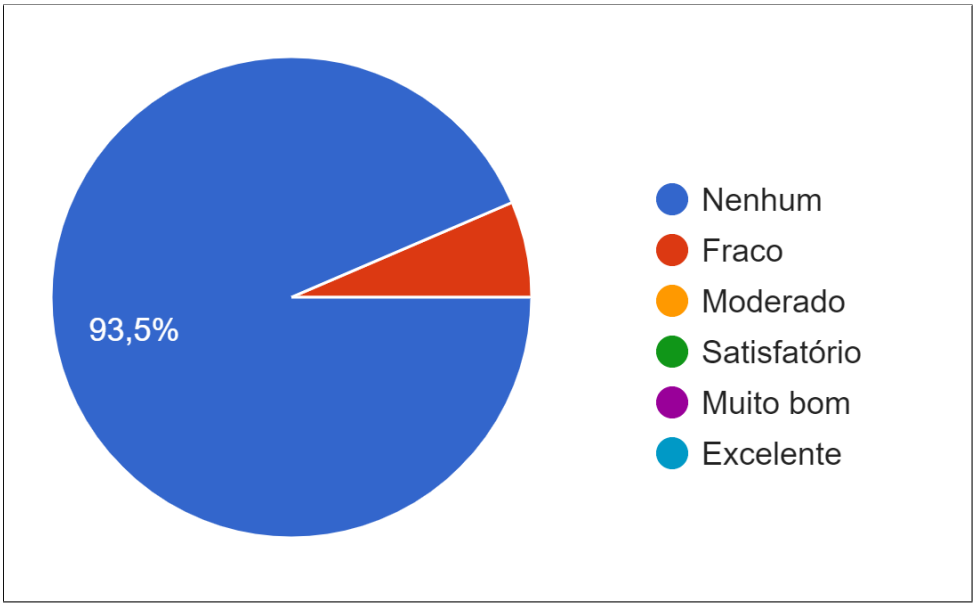
Figura 21 – Resposta da 6º questão do questionário inicial



Fonte – Elaborado pela autora.

7 - Sobre o tema de Simulação Discreta Baseada em Eventos, qual seu nível de conhecimento?

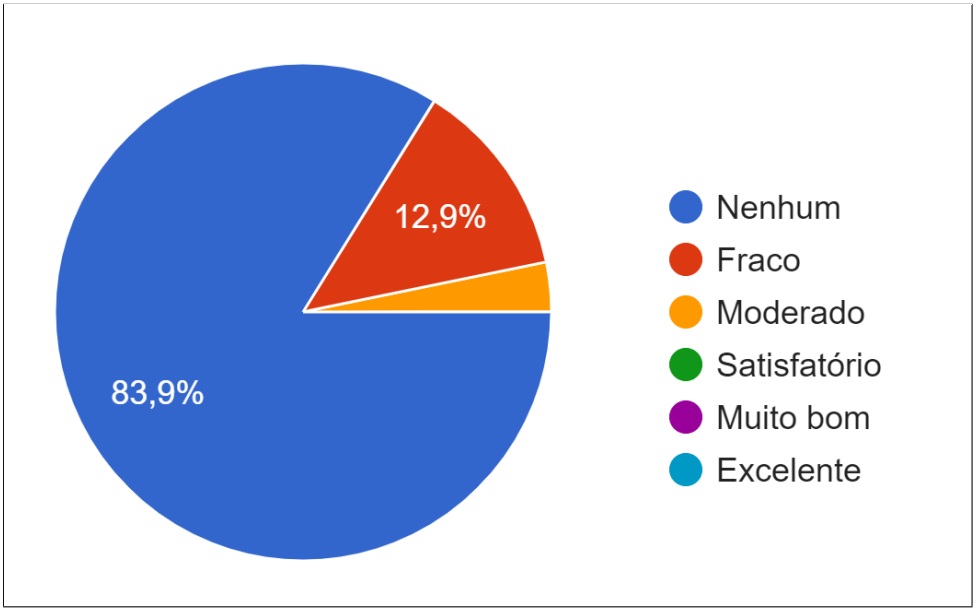
Figura 22 – Resposta da 7º questão do questionário inicial



Fonte – Elaborado pela autora.

8 - Sobre o tema de Bibliotecas de Simulação, qual seu nível de conhecimento?

Figura 23 – Resposta da 8º questão do questionário inicial



Fonte – Elaborado pela autora.

O questionário foi respondido por 31 alunos e com base nas respostas obtidas, pode-se afirmar que a maioria dos alunos da amostra não tinha experiência prévia em relação à disciplina de Avaliação de Desempenho de Sistemas, já que aproximadamente 97% dos participantes não a cursaram, como mostra o gráfico na Figura 20. Apesar de uma porcentagem significativa de

aproximadamente 26% tenha indicado conhecimento fraco ou moderado em relação ao tema principal de Avaliação de Desempenho, a grande maioria não possuía nenhum conhecimento prévio, como ilustra a Figura 21.

Nas Figuras 22 e 23 são apresentadas as porcentagens de nível de conhecimento sobre dois subtemas da área de Avaliação de Desempenho, Simulação Discreta Baseada em Eventos e Bibliotecas de Simulação respectivamente. Verifica-se que a maioria dos alunos, aproximadamente 93% dos alunos na 7º questão e 84% na 8º questão indicaram a opção "nenhum", isso indica que esses temas também são novos para a maioria dos participantes.

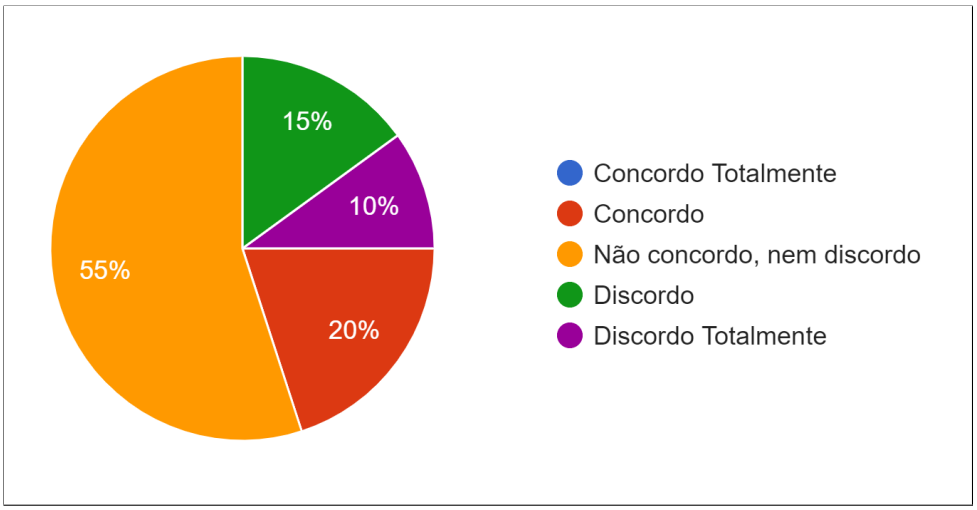
De modo geral, os resultados indicam que a amostra tinha um baixo nível de conhecimento prévio em relação aos temas estudados.

Avaliação qualitativa

Ao término da realização do experimento, foi aplicado um questionário online de natureza qualitativa aos participantes, cujas respostas foram de caráter opcional, a escolha de responder ou não o questionário ficou inteiramente a critério dos participantes. O questionário foi composto por um total de 19 questões, dentre elas 3 questões discursivas e obteve 20 respostas, o que corresponde a 64% dos participantes. Os resultados foram organizados em gráficos e a seguir são apresentados as respostas das principais questões.

5 - A atividade proposta era muito complexa.

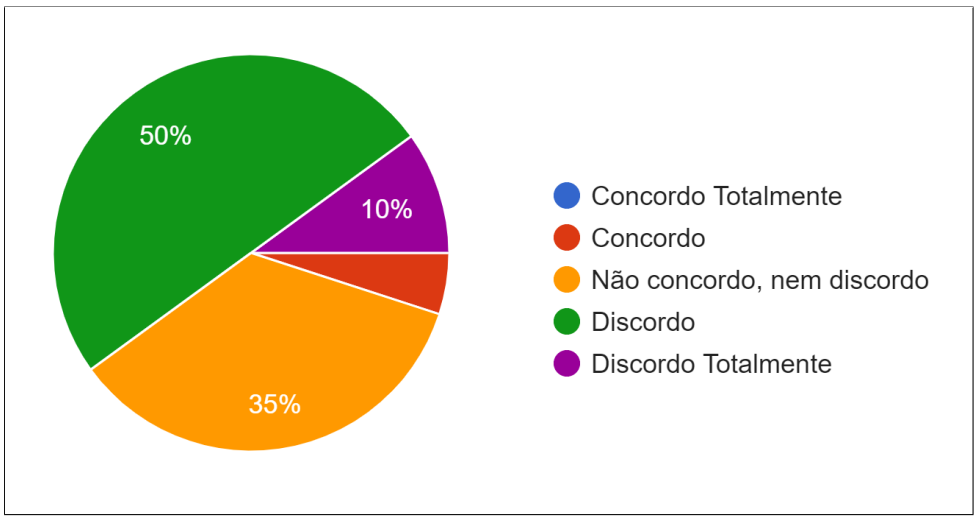
Figura 24 – Gráfico de Resposta da Questão 5 do questionário qualitativo



Fonte – Elaborado pela autora.

6 - Seria mais fácil desenvolver a atividade SEM o uso da ferramenta.

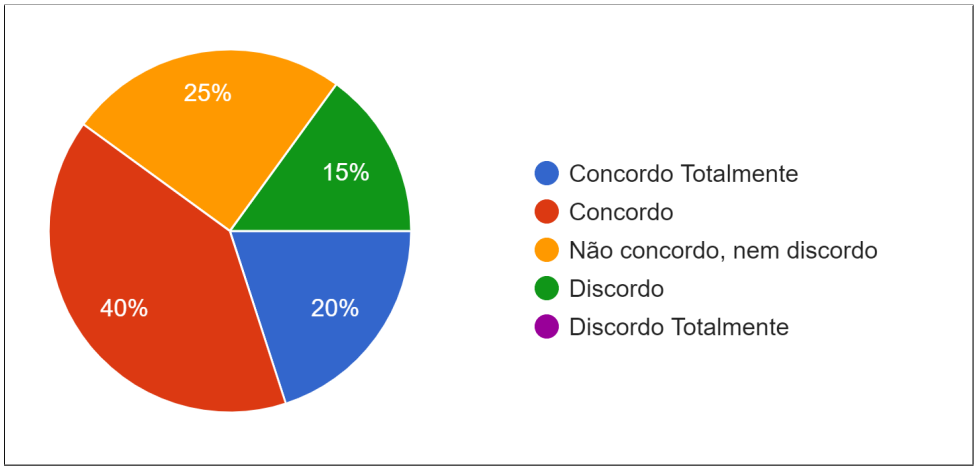
Figura 25 – Gráfico de resposta da Questão 6 do questionário qualitativo



Fonte – Elaborado pela autora.

7 - Ter um exemplo básico de um código de simulação gerado pela ferramenta, facilitou o entendimento sobre os códigos de simulação.

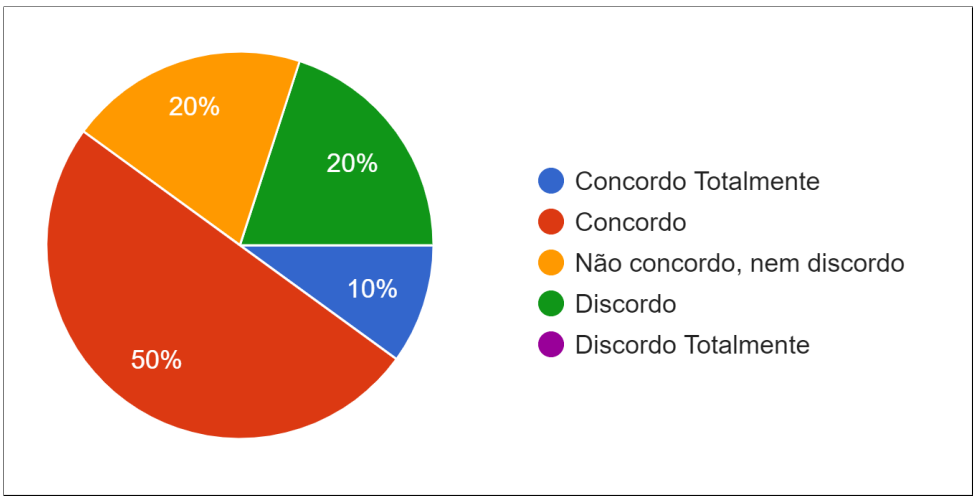
Figura 26 – Gráfico de resposta da Questão 7 do questionário qualitativo



Fonte – Elaborado pela autora.

8 - Eu gostei do uso da ferramenta na aula.

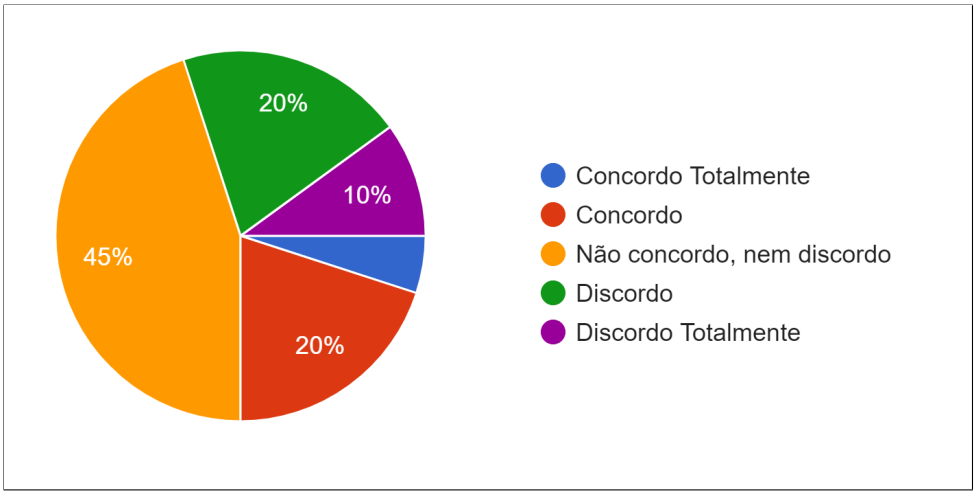
Figura 27 – Gráfico de resposta da Questão 8 do questionário qualitativo



Fonte – Elaborado pela autora.

9 - A ferramenta foi fácil de utilizar.

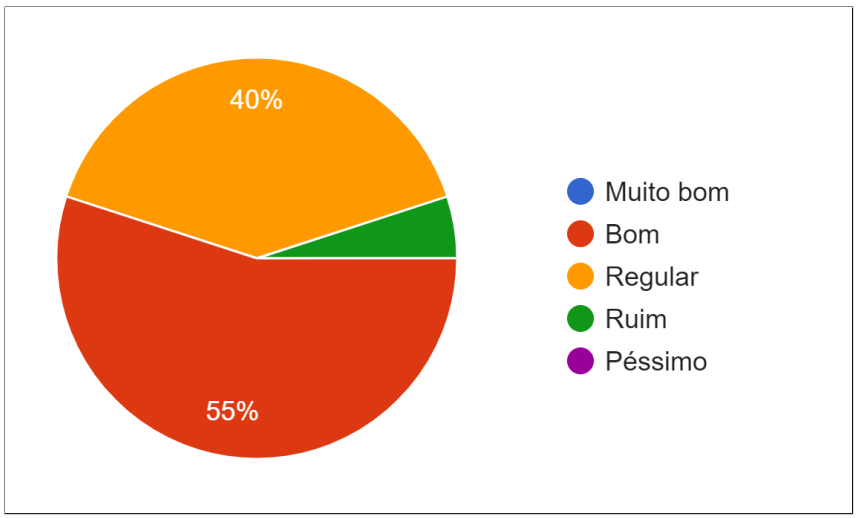
Figura 28 – Gráfico de resposta da Questão 9 do questionário qualitativo



Fonte – Elaborado pela autora.

12 - O método de aula utilizando uma ferramenta para auxiliar no ensino foi:

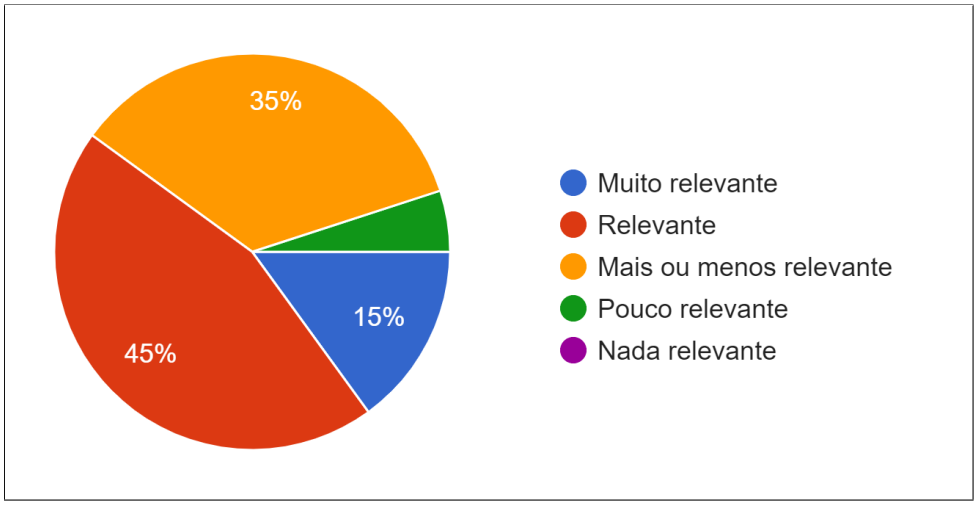
Figura 29 – Gráfico de resposta da Questão 12 do questionário qualitativo



Fonte – Elaborado pela autora.

13 - O uso da ferramenta foi relevante para a aula?

Figura 30 – Gráfico de resposta da Questão 13 do questionário qualitativo



Fonte – Elaborado pela autora.

16 - Quais as suas impressões sobre a ferramenta? Foi uma boa proposta utilizá-la em aula? É útil? Precisa de melhorias?

- "Acho que faz sentido, dando um esqueleto geral do caminho que tínhamos que seguir para concluir o exercício."
- "Sim. Acredito que a análise de código é de muita utilidade quando se quer aprender sobre algum assunto, desde que seja dado um tempo proporcional ao tamanho do código. Desse modo, a ferramenta é sim útil. (...)"

- “Foi uma boa proposta, me pareceu bem útil, mas precisaria analisar melhor a ferramenta para indicar alguma mudança.”
- “A ferramente em si pareceu interessante e útil quanto o entendimento do conteúdo proposto.”
- “A ferramenta em si foi uma boa proposta.”

17 - Deixe um feedback sobre a aula e o uso ferramenta, destacando aspectos positivos e negativos que observou. Você também pode dar sugestões, fazer elogios e/ou críticas. Seus comentários são muito importantes para a melhoria das aulas e da ferramenta.

- “Má qualidade do som na transmissão da aula(...) Infelizmente não consegui acompanhar o início da atividade por causa da falta de som na sala. A ferramenta em si foi uma boa proposta e a forma em que o conteúdo foi ensinado também foi de boa qualidade.”
- “Dificuldades na execução da ferramenta.”
- “A ferramenta foi sim bem utilizada e facilitou o entendimento da disciplina. Aspectos positivos se concentram entorno da ferramenta visto que facilitou o entendimento do assunto estudado. Não vejo aspecto negativo suficientemente relevante para ser citado.”
- “A aula teria sido boa se eu tivesse conseguido instalar a ferramenta no WSL do computador do laboratório, mas não consegui, então fiquei apenas assistindo a transmissão.”
- “Achei a aula interessante, um ponto negativo foi o tempo gasto configurando o ambiente no computador do laboratório.”
- “Configurar o ambiente antes da aula.”

5.3.5 Análise

Nesta etapa serão apresentadas as respostas para as questões de pesquisa levantadas na Subseção 5.3.2 a partir dos resultados obtidos e apresentados na etapa anterior.

[QP1] Qual é a percepção dos alunos sobre a eficácia da ferramenta de geração de código no processo de aprendizagem de Avaliação de Desempenho durante seu primeiro contato com a área?

A análise dos resultados do experimento mostrou que a maioria dos alunos teve uma percepção positiva sobre a ferramenta de geração de código utilizada como auxiliar no processo de aprendizagem de Avaliação de Desempenho. A Figura 30 apresenta informações importantes sobre a relevância da ferramenta na aula. De acordo com os resultados apresentados, é possível observar que a maioria dos alunos consideraram a ferramenta relevante para a aula, totalizando

60% dos participantes. Além disso, 35% dos alunos consideraram a ferramenta moderadamente relevante, demonstrando um nível de importância intermediário. Somente 5% dos alunos avaliaram a ferramenta como pouco relevante para o aprendizado. Esses resultados são bastante significativos, já que a grande maioria dos alunos percebeu o valor da ferramenta como um recurso útil para o processo de aprendizagem, o que pode ser um indicativo positivo para a sua utilização futura em outras aulas ou cursos.

Na Figura 26, é possível perceber que apenas 20% dos alunos não concordam que o exemplo básico de um código de simulação gerado pela ferramenta tenha facilitado o entendimento sobre os códigos de simulação. Em contrapartida, a maioria dos alunos aprovou a utilização deste exemplo como forma de auxiliar no processo de aprendizado. Com esse resultado é possível destacar que o uso de exemplos práticos e simplificados são uma estratégia eficaz para o entendimento de conceitos complexos.

Além disso, na Figura 25, pode-se observar que a maioria dos participantes concorda que a atividade proposta seria difícil de ser desenvolvida sem a ajuda da ferramenta. Porém, na Figura 24, a maioria considera o exercício proposto como sendo de complexidade mediana. Esses resultados comprovam a utilidade da ferramenta ao proporcionar aos participantes a realização de simulações no primeiro contato com a área, diminuindo as barreiras de complexidade do tema estudado. Dessa forma, não é necessário um conhecimento aprofundado de diversos temas de inicio, para começar a utilizar os conceitos aprendidos em prática. Assim, a ferramenta atua como uma facilitadora no processo de aprendizagem, tornando o conteúdo mais acessível e compreensível para os alunos.

[QP2] Os alunos aprovaram o uso da ferramenta de geração de código para auxiliar no processo de aprendizagem?

A análise dos resultados obtidos neste estudo revela que a ferramenta de geração de código desenvolvida obteve uma boa aceitação pelos alunos. De acordo com a Figura 27, apenas 20% dos participantes não gostaram do uso da ferramenta, o que indica que de modo geral os alunos ficaram satisfeitos com a ferramenta. Já a Figura 28 mostrou que 30% dos alunos tiveram dificuldades em utilizar a ferramenta, o que pode ser um indicativo para possíveis melhorias em sua usabilidade, mas é importante destacar que a maioria teve uma percepção positiva.

A Figura 29 evidenciou que os alunos aprovaram o método de aula utilizando uma ferramenta, indicando que ela foi útil no processo de aprendizagem. Por fim, a Questão 16, que perguntava de modo geral sobre a opinião dos alunos em relação à ferramenta, mostrou que a maioria dos alunos aprovou o uso da ferramenta na aula e a considerou útil. Contudo, na Questão 17 foi possível evidenciar críticas quanto ao processo de instalação da ferramenta utilizada para o auxílio do ensino de avaliação de desempenho. Em alguns casos, foram relatados problemas e erros na instalação da ferramenta em algumas máquinas, o que gerou insatisfação por parte dos alunos e acarretou em uma dificuldade na utilização da ferramenta, anteriormente apontada na Figura 28.

Além disso, também foram apontados problemas na transmissão das aulas, o que dificultou o entendimento dos conceitos abordados na aula e gerou frustração na turma. É importante ressaltar esses fatores, uma vez que afetam diretamente a qualidade do processo de ensino-aprendizagem e podem comprometer os resultados esperados.

5.4 Experimento II: Aula Presencial

No mês de novembro de 2022, o experimento foi realizado em um dos laboratórios de ensino do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP), Câmpus Presidente Epitácio. A amostra utilizada no estudo foi composta por alunos do 6º período matriculados no curso de Bacharelado em Ciência da Computação da instituição, os quais participaram do experimento presencialmente conduzido pela pesquisadora.

5.4.1 Escopo

Objetivo

O objetivo deste experimento é avaliar a utilização e aceitação da ferramenta de geração de código desenvolvida e apresentada no Capítulo 4 para auxiliar o ensino de avaliação de desempenho, visando a diminuição da complexidade do aprendizado no primeiro contato com a área.

Objeto

O objeto de estudo deste experimento é o uso de um ambiente automático de simulação para o ensino de avaliação de desempenho.

Foco Qualitativo

O foco qualitativo do experimento é averiguar os impactos positivos e negativos da utilização do ambiente automático de simulação como auxílio durantes as aulas.

Perspectiva

Apesar de manter os mesmos objetivos do experimento anterior, espera-se que o estudo seja conduzido no ambiente de sala de aula dos participantes com o palestrante presente. O mesmo ambiente deve oferecer recursos computacionais adequados para o uso da ferramenta desenvolvida.

Contexto

O experimento foi planejado no formato de minicurso e oferecido nas dependências do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Câmpus Presidente Epitácio. O estudo foi realizado em novembro de 2022 e contou com duas aulas de 1 hora e 40 minutos de duração. O minicurso foi ministrado presencialmente pelo palestrante, em um laboratório de computação, com infraestrutura adequada para a realização do estudo.

5.4.2 Planejamento

Definição das Questões de Pesquisa

Em função dos objetivos estabelecidos previamente, definiram-se as seguintes questões de pesquisa:

QP1 Qual é a percepção dos alunos sobre a eficácia da ferramenta de geração de código no processo de aprendizagem de Avaliação de Desempenho durante seu primeiro contato com a área?

QP2 Os alunos aprovaram o uso da ferramenta de geração de código para auxiliar no processo de aprendizagem?

Seleção de Sujeitos

A seleção de sujeitos para o desenvolvimento do experimento foi constituída de alunos regularmente matriculados no curso de Bacharelado em Ciência da Computação que estavam no 6º período da graduação, de tal forma que os pesquisadores não exerceram qualquer influência sobre o a seleção destes alunos.

Descrição da Instrumentação

O experimento consistiu em um minicurso, não obrigatório e preferencialmente destinado a alunos do 6º período do curso de Bacharelado em Ciência da Computação, devido aos mesmos estarem cursando a disciplina de Sistemas Distribuídos (PEPSDIS) que aborda, de forma introdutória, o tema de Avaliação de Desempenho. O curso teve um total de 17 participantes e foi dividido em duas aulas com duração total de 3 horas e 20 minutos, sendo cada aula com duração de 1 hora e 40 minutos. O início da primeira aula foi aplicado um questionário de conhecimento inicial, para situar o conhecimento geral dos alunos sobre Avaliação de Desempenho. Este questionário se encontra no Apêndice C.

A primeira aula abordou tópicos teóricos que apresentavam os principais conceitos da área de Avaliação de Desempenho de Sistemas. Na segunda aula colocou-se em prática os conceitos aprendidos para gerar códigos de simulação e como auxílio foi utilizado o ambiente automático. Foi proposto também o desenvolvimento de um exercício, o qual é descrito no Apêndice E.2. Antes do encerramento da aula foi aplicado um questionário qualitativo, apresentado no Apêndice D.

Planejamento de Aula

O Quadro 7 apresenta os tópicos abordados nas aulas.

5.4.3 Execução

Preparação

Durante a fase de preparação, foram realizadas diversas atividades para garantir o sucesso

Quadro 7 – Planejamento de aula.

Aula	Conteúdo
1	- Introdução - Motivação - Avaliação de Desempenho - Etapas de uma Avaliação de Desempenho - Técnicas para Avaliação de Desempenho - Técnicas de Aferição: Protótipos, Benchmarks e Monitores - Técnicas de Modelagem: Solução Analítica e por Simulação
2	- Modelagem de Sistemas - Bibliotecas de Simulação - Apresentação do Ambiente de Simulação - Aplicação do exercício utilizando o Ambiente Automático

da aula teórica. Inicialmente, foi selecionado cuidadosamente o material didático a ser apresentado aos alunos. Em seguida, foi elaborado um exercício prático para os alunos resolverem, a fim de consolidar os conhecimentos adquiridos durante a aula teórica. Adicionalmente, foi necessário instalar e testar o ambiente automático em todos os computadores do laboratório de ensino, a fim de assegurar que estaria pronto para uso durante a aula.

Operação

A primeira aula teve uma metodologia tradicional de ensino utilizando um slide como material de apoio e o palestrante abordou os tópicos teóricos previsto para primeira aula no Quadro 7. Na segunda aula do minicurso, foi realizada a apresentação do ambiente automático previamente instalado, acompanhada da demonstração prática do seu uso pelo palestrante. Em seguida, foram realizados exemplos em conjunto com os participantes, a fim de consolidar a compreensão da ferramenta. Por fim, foi proposto aos participantes a resolução de um exercício, utilizando os conceitos aprendidos e a ferramenta, para aplicação prática dos conhecimentos adquiridos durante o minicurso.

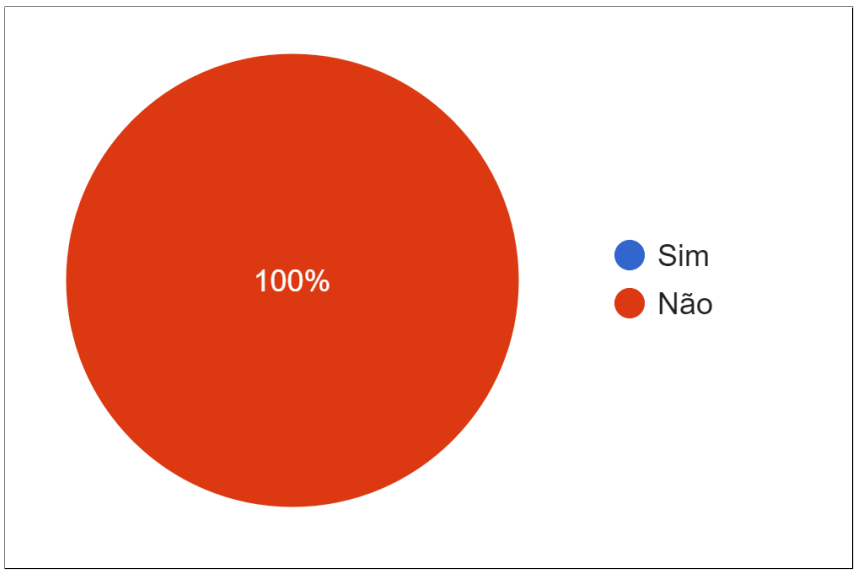
5.4.4 Descrição dos Resultados

Avaliação Inicial

Ao inicio do experimento, foi aplicado um questionário inicial online, cuja resposta era anônima para situar o conhecimento da amostra sobre o conteúdo da aula. O objetivo do questionário inicial, assim como no primeiro experimento, foi coletar informações sobre a familiaridade e compreensão dos participantes em relação ao assunto que seria estudado, a fim de se obter um panorama geral do nível de conhecimento prévio da amostra. Com os resultados do questionário, foram gerados gráficos para melhorar a visualização das informações coletadas, a seguir são apresentados os principais resultados.

4 - Você já cursou a disciplina de Avaliação de Desempenho de Sistemas?

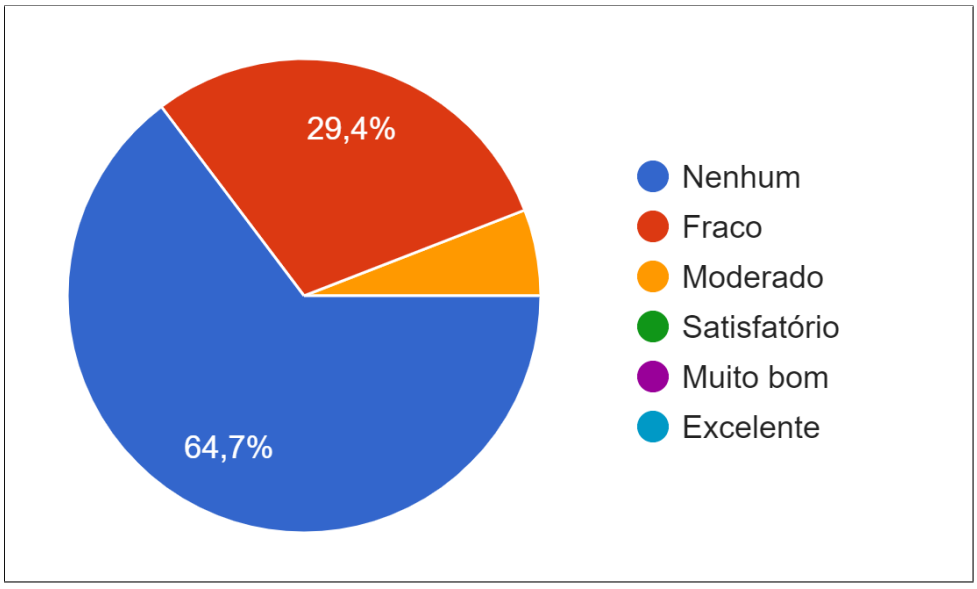
Figura 31 – Resposta da 4º questão do questionário inicial



Fonte – Elaborado pela autora.

5 - Sobre o tema de Avaliação de Desempenho, qual seu nível de conhecimento?

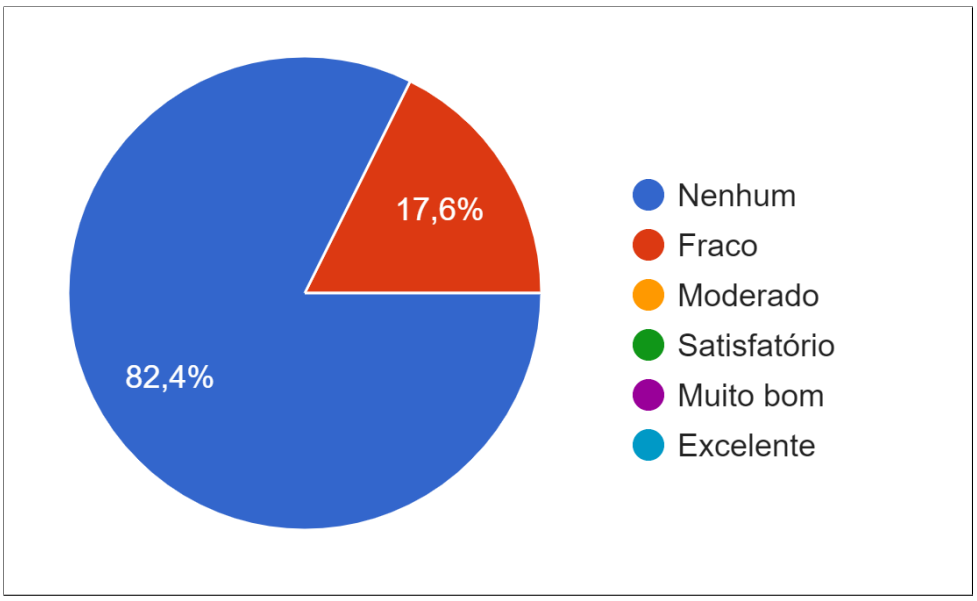
Figura 32 – Resposta da 5º questão do questionário inicial



Fonte – Elaborado pela autora.

6 - Sobre o tema de Simulação Discreta Baseada em Eventos, qual seu nível de conhecimento?

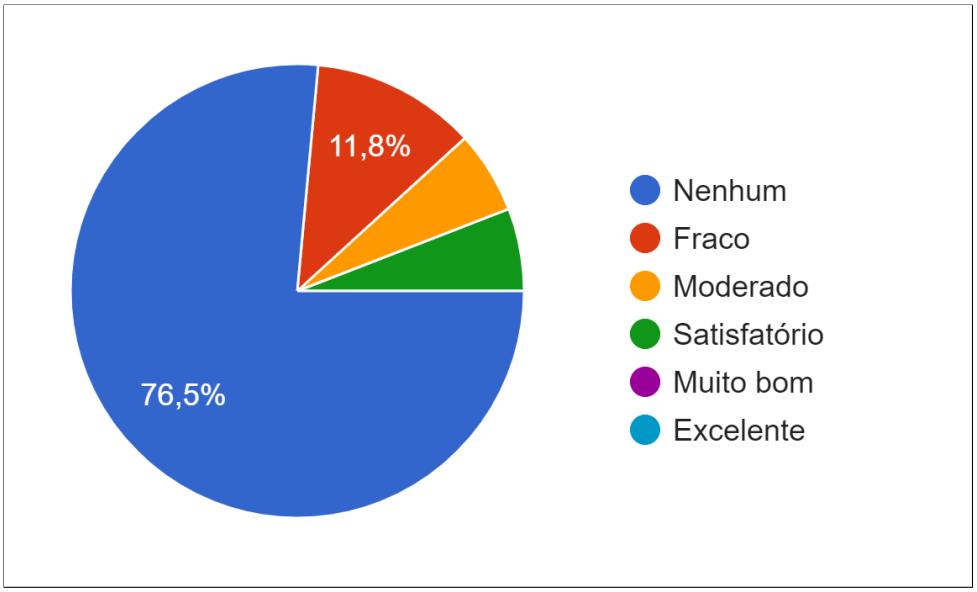
Figura 33 – Resposta da 6º questão do questionário inicial



Fonte – Elaborado pela autora.

7 - Sobre o tema de Bibliotecas de Simulação, qual seu nível de conhecimento?

Figura 34 – Resposta da 7º questão do questionário inicial



Fonte – Elaborado pela autora.

O questionário foi respondido por 17 alunos e com base nas respostas obtidas, percebe-se que nenhum aluno tinha cursado a disciplina de Avaliação de Desempenho de Sistemas como mostra o gráfico na Figura 31. Entretanto, é relevante notar na Figura 32 que cerca de 35% dos alunos possuíam um conhecimento fraco ou moderado em relação à Avaliação de Desempenho, mesmo sem ter cursado a disciplina correspondente.

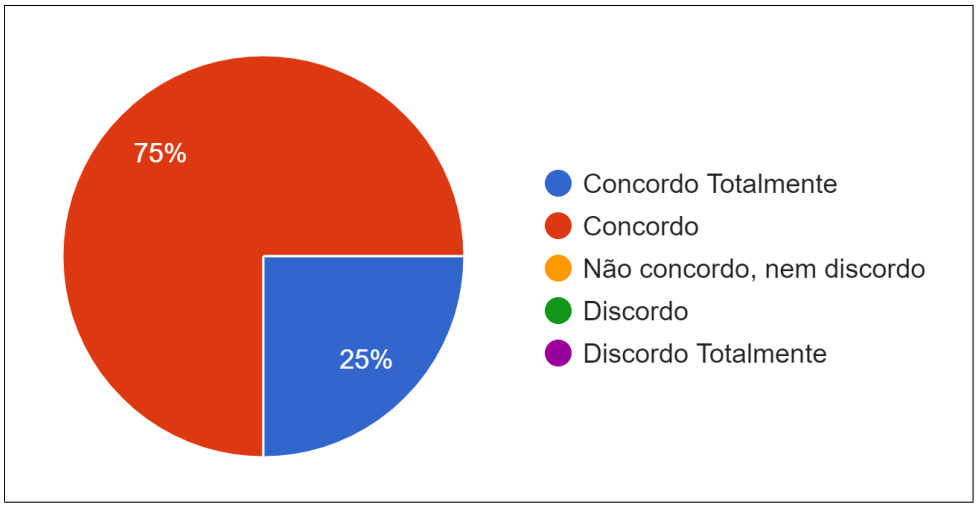
No que se refere à Simulação Discreta Baseada em Eventos, pode-se observar na Figura 33 que a maioria dos alunos (aproximadamente 84%) não possuíam nenhum conhecimento sobre o assunto. Por fim, em relação ao conhecimento sobre Bibliotecas de Simulação, os dados indicam que a maioria dos alunos (cerca de 76%) não possuíam conhecimento prévio sobre o assunto, e os cerca de 24% restantes declararam possuir conhecimento fraco, moderado ou satisfatório, como apresentado na Figura 34. De modo geral, os resultados indicam que a amostra tinha um baixo nível de conhecimento prévio em relação aos temas estudados.

Avaliação qualitativa

Ao término da realização do experimento, foi aplicado um questionário online de natureza qualitativa aos participantes, cujas respostas foram anônimas e de caráter opcional, a escolha de responder ou não o questionário ficou inteiramente a critério dos participantes. O questionário foi composto por um total de 18 questões, dentre elas 3 questões discursivas e obteve-se 12 respostas, o que corresponde a aproximadamente 71% dos participantes. Os resultados foram organizados em gráficos e a seguir são apresentados as respostas das principais questões.

6 - Eu gostei do uso da ferramenta na aula.

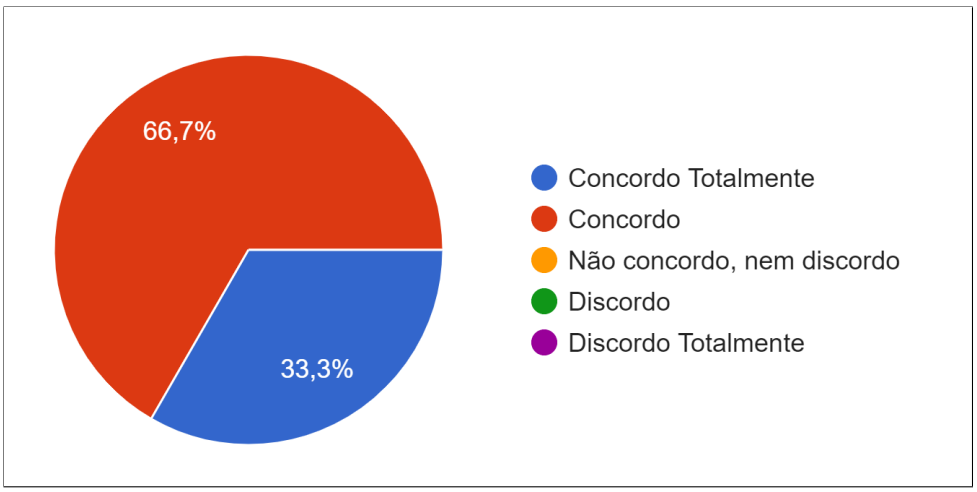
Figura 35 – Gráfico de resposta da Questão 6 do questionário qualitativo



Fonte – Elaborado pela autora.

7 - A ferramenta foi fácil de utilizar.

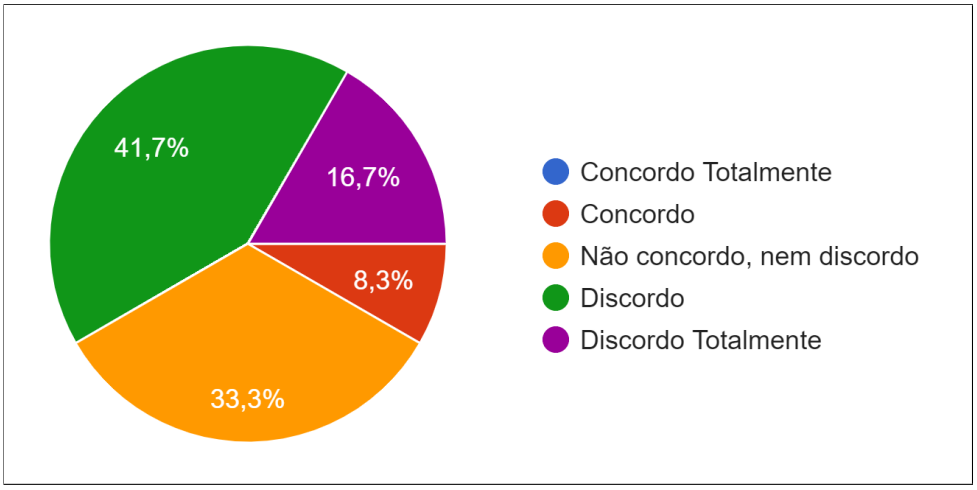
Figura 36 – Gráfico de resposta da Questão 7 do questionário qualitativo



Fonte – Elaborado pela autora.

8 - A atividade proposta era muito complexa.

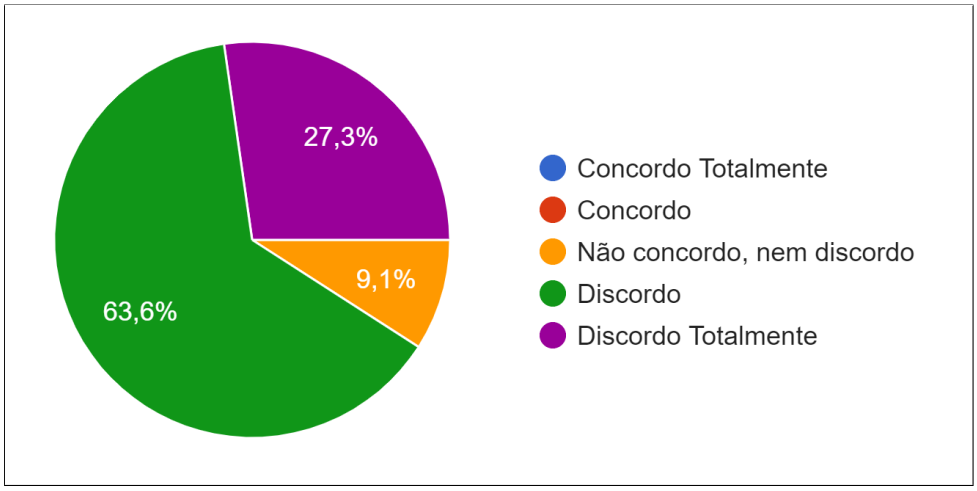
Figura 37 – Gráfico de resposta da Questão 8 do questionário qualitativo



Fonte – Elaborado pela autora.

9 - Seria mais fácil desenvolver a atividade SEM o uso da ferramenta.

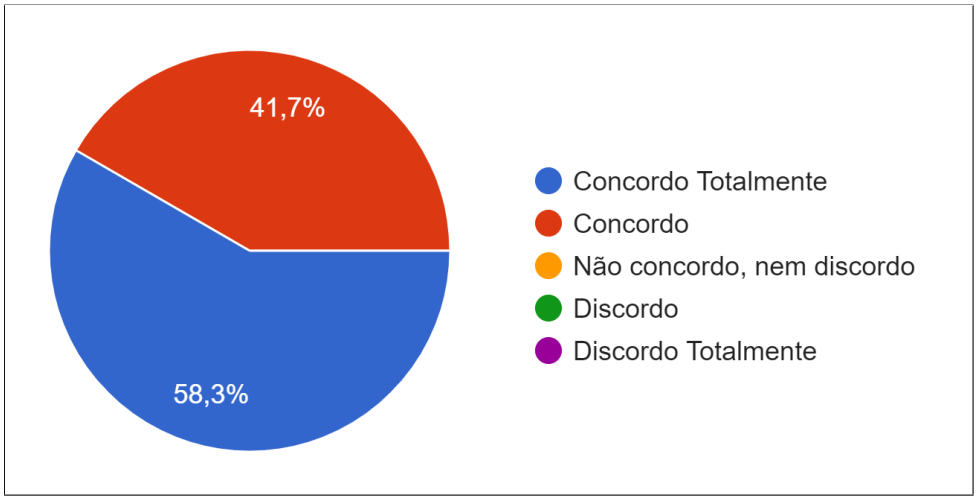
Figura 38 – Gráfico de resposta da Questão 9 do questionário qualitativo



Fonte – Elaborado pela autora.

10 - Ter um exemplo básico de um código de simulação gerado pela ferramenta, facilitou o entendimento sobre os códigos de simulação.

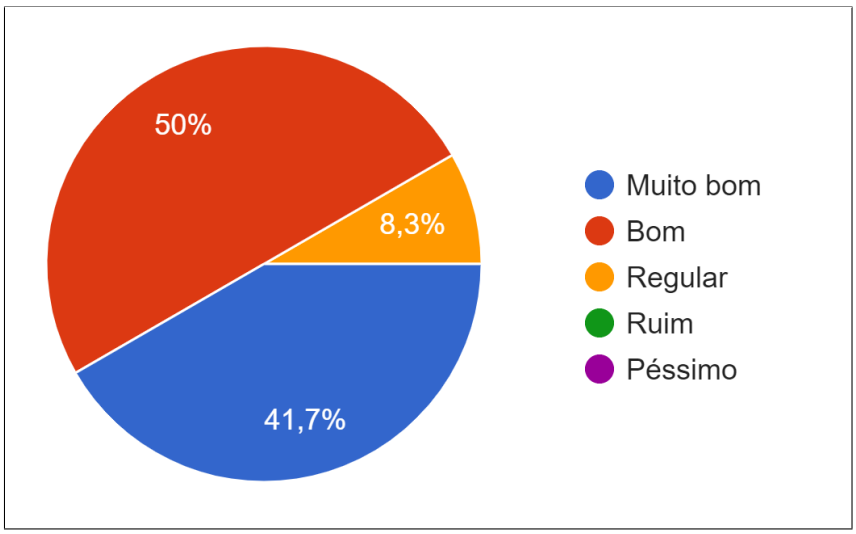
Figura 39 – Gráfico de resposta da Questão 10 do questionário qualitativo



Fonte – Elaborado pela autora.

13 - O método de aula utilizando uma ferramenta para auxiliar no ensino foi:

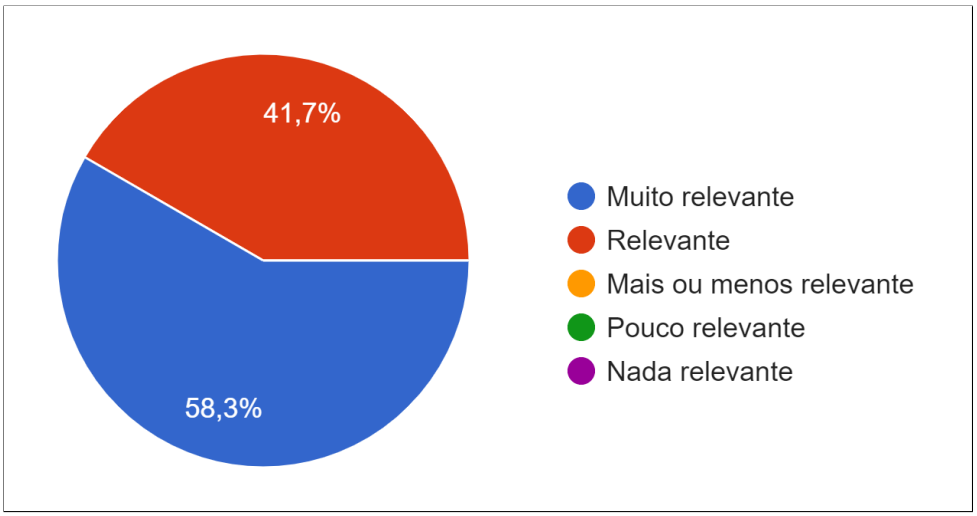
Figura 40 – Gráfico de resposta da Questão 13 do questionário qualitativo



Fonte – Elaborado pela autora.

14 - O uso da ferramenta foi relevante para a aula?

Figura 41 – Gráfico de resposta da Questão 14 do questionário qualitativo



Fonte – Elaborado pela autora.

17 - Quais as suas impressões sobre a ferramenta? Foi uma boa proposta utilizá-la em aula? É útil? Precisa de melhorias?

A seguir apresenta-se algumas respostas discursivas dadas pelos participantes para a questão 17.

- "Aulas práticas utilizando ferramentas em conjunto com o conhecimento que é passado sempre são melhores e mais fáceis de se entender e fixar o conteúdo que foi abordado na aula, achei muito útil e uma ótima proposta para um melhor entendimento."

- “A ferramenta é muito boa e ajuda a compreender todo o processo de medição, acredito que é um ótimo passo inicial para quem está começando os estudos na área.”

- “A ferramenta é extremamente intuitiva, fácil de usar e eficiente. Uma recomendação a respeito da interface utilizada para selecionar o modelo de teste, seria a criação de pastas dentro da caixa de diálogo, mas fora isso cumpre o papel.”

- “Eu achei uma boa ferramenta, sem ela teria demorado mais para entender os códigos.”

- “A ferramenta é útil, fácil e simples de se usar, com ela o código foi gerado e isso facilita muito para quem não sabe muito sobre como construir o código ou não tem o entendimento sobre a linguagem.”

- “A ferramenta foi extremamente útil para a atividade desenvolvida em aula.”

18 - Deixe um feedback sobre a aula e o uso ferramenta, destacando aspectos positivos e negativos que observou. Você também pode dar sugestões, fazer elogios e/ou críticas. Seus comentários são muito importantes para a melhoria das aulas e da ferramenta.

A seguir apresenta-se algumas respostas discursivas dadas pelos participantes para a questão 18.

- “A ferramenta é fácil de utilizar, ter o código pronto facilitou o entendimento da sua funcionalidade. A aula foi interessante, aprendi sobre um novo tema e a explicação foi fácil de entender.”

- “Eu gostei da apresentação, apesar do tempo fornecido foi pouco, mas conseguimos ver a introdução sobre o tema e alguns códigos, foi interessante observar o comportamento das análises de informação e sobre o uso computacional”

- “Explicação muito boa e de fácil entendimento, bom carisma e compreensão. Curso de conteúdo regular com uso útil de ferramenta de auxílio.”

- “Não tenho nada a acrescentar, achei um ótimo curso com um tema muito interessante e provavelmente venha a ser útil no meu futuro acadêmico.”

- “A aula foi muito boa, a professora dominava o conteúdo e sempre apresentava exemplos reais além de auxiliar quando necessário.”

- “As explicações, exemplos e interações que ocorreram na aula foram excepcionais, sem nada a criticar.”

5.4.5 Análise

Nesta etapa serão apresentadas as respostas para as questões de pesquisa levantadas na Subseção 5.3.2 a partir dos resultados obtidos e apresentados na na etapa anterior.

[QP1] Qual é a percepção dos alunos sobre a eficácia da ferramenta de geração de código no processo de aprendizagem de Avaliação de Desempenho durante seu primeiro contato com a área?

Na Figura 41, é notável que todos os alunos consideraram a ferramenta relevante para a aula, demonstrando a importância e o valor agregado que a ferramenta trouxe para o processo de aprendizagem. Isso evidencia a eficácia da utilização da ferramenta como auxílio no ensino de avaliação de desempenho, corroborando com o objetivo do experimento. Já na Figura 36, é possível perceber que os alunos concordam que a ferramenta é fácil de ser utilizada. Essa percepção é importante, já que a facilidade de uso da ferramenta pode impactar diretamente na aceitação e na adesão dos alunos ao seu uso.

A Figura 39 apresenta uma resposta unânime dos alunos quanto ao exemplo básico de um código de simulação gerado pela ferramenta. Todos os participantes afirmaram que esse exemplo facilitou o entendimento sobre os códigos de simulação, o que demonstra a eficácia da ferramenta no auxílio ao aprendizado dos alunos. Com a utilização da ferramenta, os alunos puderam visualizar um exemplo prático de aplicação, o que os ajudou a compreender de forma mais clara os conceitos e técnicas envolvidos na simulação.

Nota-se na Figura 38 que a maioria dos participantes reconheceu que não seria fácil desenvolver a atividade proposta sem o auxílio da ferramenta. Note, também, que a maioria apontou na Figura 37 que o exercício proposto não foi excessivamente complexo. Essas duas observações indicam que a ferramenta foi bem sucedida em proporcionar aos participantes uma experiência introdutória mais suave, apesar da alta complexidade dessa área de conhecimento. Mais do que isso, a ferramenta tornou possível aos alunos a realização de simulações em seu primeiro contato com a área, sem a necessidade de um conhecimento profundo e avançado sobre a área de simulação, facilitando o aprendizado e tornando-o mais acessível.

[QP2] Os alunos aprovaram o uso da ferramenta de geração de código para auxiliar no processo de aprendizagem?

Com base nos resultados apresentados na Figura 35, podemos afirmar que o uso da ferramenta foi bem aceito pelos alunos. As respostas coletadas variam entre “Concordo” com 75% e “Concordo Totalmente” com 25%, a partir desses resultados é possível concluir que a ferramenta foi bem recebida pelos estudantes. Esse resultado é bastante relevante, uma vez que a aceitação da ferramenta é um indicador importante da sua utilidade no processo de ensino-aprendizagem.

Além disso, na Figura 40, a maioria dos alunos considerou o método de aula utilizando a ferramenta para auxiliar o processo de aprendizado bom ou muito bom com cerca de 50% e 42%,

respectivamente. Apenas 8% dos alunos consideraram a aula com a ferramenta como regular, demonstrando uma alta taxa de aprovação dos estudantes em relação ao uso da tecnologia. Esse resultado pode ser visto como um indicador de que o uso de ferramentas educacionais pode ser eficaz no processo de aprendizagem, tornando o conteúdo mais acessível e dinâmico para os alunos.

Na questão 17, os alunos tiveram a oportunidade de expressar sua opinião sobre a ferramenta utilizada durante as aulas de avaliação de desempenho. Os resultados indicam que a maioria dos alunos demonstrou uma grande aprovação pela ferramenta, considerando-a útil e fácil de usar. Além disso, o fato de os alunos considerarem a ferramenta fácil de usar é um aspecto importante, pois indica que ela não representou uma barreira adicional para o aprendizado, permitindo que os alunos se concentrassem no conteúdo abordado em vez de ter que lidar com dificuldades técnicas.

A questão 18 foi utilizada para coletar as opiniões gerais dos alunos sobre a aula, e os resultados revelaram uma resposta extremamente positiva por parte dos participantes. A maioria dos alunos expressou sua satisfação em relação à aula, destacando aspectos como o conteúdo aprendido e o uso da ferramenta de forma favorável. No entanto, uma reclamação apontada foi a respeito do tempo de duração da aula, sugerindo que os alunos desejavam ter mais tempo para aprofundar o aprendizado.

CAPÍTULO

6

CONCLUSÕES

6.1 Considerações Iniciais

Neste capítulo são apresentadas as conclusões obtidas a partir dos resultados deste trabalho. Na Seção 6.2 são apresentadas as conclusões gerais do projeto. Na Seção 6.3 são apresentados as principais contribuições que o projeto gerou e as limitações envolvidas no desenvolvimento. A Seção 6.4 conta com as publicações científicas. Por fim, a Seção 6.5 apresenta os possíveis trabalhos futuros relacionados ao projeto.

6.2 Conclusões Gerais

O projeto em questão teve como objetivo principal aprimorar a qualidade da aprendizagem de avaliação de desempenho, por meio do uso da ferramenta ASDA. Para alcançar esse objetivo, foi proposta a implementação de uma extensão na ferramenta, permitindo a geração de código de simulação em outras linguagens de programação que possuem biblioteca de simulação baseada em redes de filas.

Para atingir esse objetivo, foram realizadas diversas etapas, como a fundamentação teórica necessária para o desenvolvimento do projeto, que envolveu conceitos sobre simulação, geração de código, ambientes de simulação automáticos e a própria ferramenta ASDA. Em seguida, foi realizado um estudo aprofundado do código da ferramenta ASDA, visando compreender seu funcionamento e identificar as oportunidades de melhoria para a extensão proposta.

Após o estudo detalhado do código da ferramenta ASDA, foi realizado a implementação da extensão, que tinha como objetivo permitir a geração de código de simulação em outras linguagens de programação além daquelas já suportadas pela ferramenta ASDA. Para isso, foi utilizado como base o padrão de geração de código especificado no módulo gerador do ASDA.

A nova ferramenta desenvolvida contou com a geração de código nas linguagens de

alto nível Python, Java e R, utilizando as bibliotecas de simulação Simpy, Javasim e Simmer, respectivamente. Além disso, a ferramenta implementada possui modelos implementados na linguagem .dot, que descrevem gráficos textualmente, e gabaritos, que são arquivos que contém a estrutura do código a ser criado. Para gerar o código final baseado nos modelos e gabaritos, a extensão utiliza os arquivos geradores.

Todo o processo de geração de código foi implementado na linguagem de programação Python e para testar a extensão de forma independente, foi criada uma tela simples que permitia aos usuários acessar as funcionalidades da ferramenta e realizar simulações em diferentes linguagens de programação.

Durante a fase de planejamento do projeto, foram descritos dois experimentos distintos, cada um com amostras diferentes e condições de aplicação específicas. O objetivo desses experimentos era avaliar a eficácia e a aceitação da ferramenta desenvolvida, bem como seu impacto na aprendizagem de avaliação de desempenho.

Os resultados obtidos dos experimentos foram extremamente positivos. Foi possível observar que a ferramenta desenvolvida teve uma grande aceitação por parte dos participantes, que a consideraram útil e relevante para o processo de ensino-aprendizagem. Além disso, os participantes destacaram que os códigos gerados pela ferramenta facilitaram o entendimento dos conceitos aprendidos, tornando a abordagem da disciplina mais acessível e menos complexa.

Esses resultados indicam que a ferramenta desenvolvida cumpriu seu propósito de melhorar a qualidade da aprendizagem de avaliação de desempenho, proporcionando aos alunos uma experiência mais enriquecedora e eficiente. A aceitação positiva por parte dos participantes reforça a relevância e o potencial da ferramenta como recurso educacional no contexto acadêmico.

6.3 Contribuições e Limitações

O presente projeto teve como principal contribuição o desenvolvimento de uma ferramenta de geração de código para auxiliar o ensino de Avaliação de Desempenho. Essa ferramenta possibilita aos alunos uma maior facilidade de compreensão e aprendizado dos conceitos da área, tornando o processo mais eficiente e menos complexo, principalmente para aqueles que estão tendo o primeiro contato com o tema.

A ferramenta desenvolvida permite a geração automática de códigos de simulação para a resolução de problemas e exercícios relacionados à avaliação de desempenho. Com ela, os alunos têm a oportunidade de visualizar e manipular o código gerado, facilitando a compreensão dos conceitos teóricos e a aplicação prática dos mesmos.

Dessa forma, a principal contribuição do projeto é a melhoria do processo de aprendizagem em Avaliação de Desempenho, proporcionando aos alunos uma ferramenta de apoio que simplifica o entendimento e torna o processo de aprendizado mais dinâmico e efetivo. Espera-se

que essa ferramenta possa ser utilizada por professores e alunos em outras instituições de ensino, contribuindo para a melhoria da qualidade do ensino e aprendizagem da área.

6.4 Publicações e Submissões

Durante o desenvolvimento deste trabalho o seguinte artigo foi submetido.

1. LOPES, N.; BRUSCHI, S. M. Ampliação das linguagens suportadas pelo Ambiente de Simulação Distribuída Automático (ASDA). EduComp'23, Abril 24-29, 2023, Recife, Pernambuco, Brasil (On-line).

6.5 Trabalhos Futuros

Este projeto deixa os seguintes tópicos como sugestão de trabalhos futuros:

- Integração da extensão ao Ambiente de Simulação Distribuída Automático (ASDA), atualizando a versão desktop do sistema;
- Analisar o impacto a longo prazo do uso da extensão desenvolvida em aulas;
- Desenvolvimento de uma interface gráfica web e dos demais módulos do ASDA na linguagem de programação Python, oferecendo deste modo uma versão web do ASDA possibilitando a comunidade externa acesso as funcionalidades do sistema;
- Desenvolvimento de um módulo de execução de código para a versão web que possibilite ao usuário além de gerar um código de simulação, executa-lo no sistema e colher os resultados para análise.

REFERÊNCIAS

ARNOLDUS, J.; BRAND, M. V. d.; SEREBRENIK, A.; BRUNEKREEF, J. J. **Code generation with templates**. [S.l.]: Springer Science & Business Media, 2012. v. 1. Citado na página 32.

BANKS, J.; CARSON, J. S.; BARRY, L. *et al.* **Discrete-event system simulation**. [S.l.]: Pearson, 2005. Citado na página 30.

BRUSCHI, S. M. **ASDA: um ambiente de simulação distribuída automático**. Tese (Doutorado) — Universidade de São Paulo, 2002. Citado nas páginas 26, 34, 37, 38, 49 e 50.

CARBONNELLE, P. **PYPL PopularitY of Programming Language**. 2020. <https://pypl.github.io/PYPL.html>. Acesso em: 01-06-2021. Citado na página 47.

CASACA, A. C. P. Simulation and the lean port environment. **Maritime Economics & Logistics**, Springer, v. 7, n. 3, p. 262–280, 2005. Citado nas páginas 25 e 29.

CZARNECKI, K.; EISENECKER, U. W. Components and generative programming. In: SPRINGER. **Software Engineering—ESEC/FSE'99**. [S.l.], 1999. p. 2–19. Citado na página 32.

DOLLARD, K. **Code Generation in Microsoft. NET**. [S.l.]: Apress, 2008. Citado nas páginas 30 e 31.

EISENECKER, U. W. Generative programming (gp) with c++. In: SPRINGER. **Joint Modular Languages Conference**. [S.l.], 1997. p. 351–365. Citado na página 31.

FURLANETTO, G. C. Geração de simuladores de filas para diferentes contextos com estudo de casos para redes de computadores. Universidade Estadual Paulista (UNESP), 2016. Citado nas páginas 44 e 45.

GITHUB. **GitHut 2.0 - A small place to discover languanges in GitHub**. 2021. <https://madnight.github.io/githut/#/pull_requests/2021/1>. Acesso em: 05-06-2021. Citado na página 47.

HERRINGTON, J. **Code generation in action**. [S.l.]: Manning Publications Co., 2003. Citado nas páginas 31 e 32.

JONES, B.; SALL, J. Jmp statistical discovery software. **WIREs Computational Statistics**, v. 3, n. 3, p. 188–194, 2011. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.162>. Citado na página 35.

KUROWSKI, P. **Engineering Analysis with SOLIDWORKS Simulation 2015**. [S.l.]: SDC Publications, 2015. Citado na página 35.

MARIA, A. Introduction to modeling and simulation. In: **Proceedings of the 29th conference on Winter simulation**. [S.l.: s.n.], 1997. p. 7–13. Citado na página 29.

MATLOFF, N. Introduction to discrete-event simulation and the simpy language. **Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August**, v. 2, n. 2009, p. 1–33, 2008. Citado na página 30.

NETWORK, P. T. **Packet tracer 8.0.0 available for download**. 2021. <https://www.packettracernetwork.com/packettracer-80-newfeatures.html>. Acesso em: 10-03-2021. Citado na página 37.

NETWORKING, C. A. **Teaching with Packet Tracer**. 2020. <https://www.netacad.com/pt-br/courses/packet-tracer/teaching>. Acesso em: 12-12-2020. Citado na página 36.

NOOR, N. M. M.; YAYAO, N.; SULAIMAN, S. Effectiveness of using cisco packet tracer as a learning tool: A case study of routing protocol. **Computer software**, v. 514, p. 689–9, 2018. Citado na página 36.

O'GRADY, S. **The RedMonk Programming Language Rankings: January 2021**. 2021. <https://redmonk.com/sogrady/2021/03/01/language-rankings-1-21/>. Acesso em: 05-06-2021. Citado na página 48.

OLIVEIRA, R.; CORREA, V.; NUNES, L. Mapeamento do fluxo de valor em um modelo de simulação computacional. **Revista Produção Online**, v. 14, p. 837, 08 2014. Citado na página 34.

PRADO, D. S. d. **Usando o Arena em simulação**. [S.l.]: Falconi Editora, 2014. v. 3. Citado na página 34.

ROBINSON, S. **Simulation: The practice of model development and use**. [S.l.]: John Wiley & Sons Ltd, 2004. Citado na página 30.

RUMPE, B. **Agile Modeling with UML: Code Generation, Testing, Refactoring**. [S.l.]: Springer, 2017. Citado na página 31.

SALL, J.; STEPHENS, M. L.; LEHMAN, A.; LORING, S. **JMP start statistics: a guide to statistics and data analysis using JMP**. [S.l.]: Sas Institute, 2017. Citado na página 36.

SAS. **Simulation Experiment**. 2020. <https://community.jmp.com/t5/JMP-On-Air/Simulation-Experiment/ta-p/261021>. Acesso em: 14-12-2020. Citado na página 36.

SCHLEE, M.; VANDERDONCKT, J. Generative programming of graphical user interfaces. In: **Proceedings of the working conference on Advanced visual interfaces**. [S.l.: s.n.], 2004. p. 403–406. Citado na página 31.

SHANNON, R. E. Introduction to the art and science of simulation. In: IEEE. **Proceedings of the winter simulation conference**. [S.l.], 1998. v. 1, p. 7–14. Citado na página 29.

SOGETILABS. **Code generation... Can it be simple and pragmatic?** 2013. <https://labs.sogeti.com/top-10-post-code-generation-can-simple-pragmatic/>. Acesso em: 14-12-2020. Citado na página 33.

SPOLON, R. **Um gerador de aplicação para um Ambiente de Simulação Automático**. Tese (Doutorado) — Universidade de São Paulo, 1994. Citado nas páginas 30 e 43.

SYSTÈMES, D. **Code generation... Can it be simple and pragmatic?** 2020. <https://www.solidworks.com/pt-br/product/3dexperience-works-simulation>. Acesso em: 14-12-2020. Citado na página 35.

TIOBE. **TIOBE Index for June 2021**. 2021. <https://www.tiobe.com/tiobe-index/>. Acesso em: 05-06-2021. Citado na página 47.

VAVILINA, E.; GAIGALS, G. Improved labview code generation. In: IEEE. **Workshop on Advances in Information, Electronic and Electrical Engineering**. [S.l.], 2015. p. 1–4. Citado na página 44.

WHITE, K. P.; INGALLS, R. G. The basics of simulation. In: **2018 Winter Simulation Conference (WSC)**. [S.l.: s.n.], 2018. p. 147–161. Citado na página 25.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. Experiment process illustration. In: ____. **Experimentation in Software Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 161–174. ISBN 978-3-642-29044-2. Disponível em: <https://doi.org/10.1007/978-3-642-29044-2_12>. Citado na página 63.

XIAO, J.; ANDELFINGER, P.; CAI, W.; RICHMOND, P.; KNOLL, A.; ECKHOFF, D. Advancing automatic code generation for agent-based simulations on heterogeneous hardware. In: SPRINGER. **European Conference on Parallel Processing**. [S.l.], 2019. p. 308–319. Citado na página 45.

APÊNDICE

A

QUESTIONÁRIO INICIAL - EXPERIMENTO I

E-mail: .....

Nome Completo: .....

1. **Idade:** .....

2. **Gênero:**
   ( ) Feminino
   ( ) Masculino
   ( ) Prefiro não dizer

3. **Você já cursou a disciplina de Avaliação de Desempenho de Sistemas?**
   ( ) Sim
   ( ) Não

4. **Sobre o tema de Avaliação de Desempenho, qual seu nível de conhecimento?**
   ( ) Nenhum
   ( ) Fraco
   ( ) Moderado
   ( ) Satisfatório
   ( ) Muito bom
   ( ) Excelente

5. **Sobre o tema de Simulação Discreta baseada em eventos, qual seu nível de conhecimento?**
   ( ) Nenhum
   ( ) Fraco
   ( ) Moderado
   ( ) Satisfatório

( ) Muito bom
( ) Excelente

6. **Sobre o tema de Bibliotecas de Simulação, qual seu nível de conhecimento?**
( ) Nenhum
( ) Fraco
( ) Moderado
( ) Satisfatório
( ) Muito bom
( ) Excelente

7. **Sobre o tema de Lógica de Programação, qual seu nível de conhecimento?**
( ) Nenhum
( ) Fraco
( ) Moderado
( ) Satisfatório
( ) Muito bom
( ) Excelente

8. **Qual seu nível de conhecimento sobre as Linguagens de Programação abaixo?**

- **Linguagem de programação Python:**
  ( ) Nenhum
  ( ) Fraco
  ( ) Moderado
  ( ) Satisfatório
  ( ) Muito bom
  ( ) Excelente
- **Linguagem de programação Java:**
  ( ) Nenhum
  ( ) Fraco
  ( ) Moderado
  ( ) Satisfatório
  ( ) Muito bom
  ( ) Excelente
- **Linguagem de programação R:**
  ( ) Nenhum
  ( ) Fraco
  ( ) Moderado
  ( ) Satisfatório

( ) Muito bom
( ) Excelente

- **Linguagem de programação C:**
  ( ) Nenhum
  ( ) Fraco
  ( ) Moderado
  ( ) Satisfatório
  ( ) Muito bom
  ( ) Excelente

APÊNDICE

B

QUESTIONÁRIO QUALITATIVO - EXPERIMENTO I

E-mail: .....

Nome Completo: .....

***De acordo com as afirmações a seguir escolha entre as opções, a resposta que mais te representa.**

1. **O instrutor conseguiu transmitir o conteúdo de forma clara e objetiva.**
   ( ) Concordo Totalmente
   ( ) Concordo
   ( ) Não concordo, nem discordo
   ( ) Discordo
   ( ) Discordo Totalmente

2. **Eu consegui compreender todo o conteúdo abordado na aula.**
   ( ) Concordo Totalmente
   ( ) Concordo
   ( ) Não concordo, nem discordo
   ( ) Discordo
   ( ) Discordo Totalmente

3. **O conteúdo aprendido nesta aula será útil pra mim.**
   ( ) Concordo Totalmente
   ( ) Concordo
   ( ) Não concordo, nem discordo
   ( ) Discordo
   ( ) Discordo Totalmente

4. **Eu gostei de aprender sobre o assunto desta aula.**

( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

5. **A atividade proposta era muito complexa.**

( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

6. **Seria mais fácil desenvolver a atividade *SEM* o uso da ferramenta.**

( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

7. **Ter um exemplo básico de um código de simulação gerado pela ferramenta, facilitou o entendimento sobre os códigos de simulação.**

( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

8. **Eu gostei do uso da ferramenta na aula.**

( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

9. **A ferramenta foi fácil de utilizar.**

( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

10. **Eu conseguiria replicar o exercício da aula sem o uso da ferramenta.**

( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

11. **Como você classifica o nível de dificuldade do assunto estudado nesta aula (Simulação de Sistemas)?**

( ) Muito Difícil
( ) Difícil
( ) Regular
( ) Fácil
( ) Muito fácil

12. **O método de aula utilizando uma ferramenta para auxiliar no ensino foi?**

( ) Muito bom
( ) Bom
( ) Regular
( ) Ruim
( ) Péssimo

13. **O uso da ferramenta foi relevante para a aula?**

( ) Muito relevante
( ) Relevante
( ) Mais ou menos relevante
( ) Pouco relevante
( ) Nada relevante

14. **O quanto você teria conseguido desenvolver da atividade proposta em aula se não tivesse utilizado a ferramenta?**

( ) A atividade toda e um pouco mais
( ) A atividade toda
( ) Metade da atividade
( ) Um pouco da atividade
( ) Não conseguiria desenvolver nada

15. **Quais foram suas maiores dificuldades sobre o conteúdo da aula?**

.....
.....
.....

16. **Quais as suas impressões sobre a ferramenta? Foi uma boa proposta utiliza-la em aula? É útil? Precisa de melhorias?**

.....
.....
.....

17. **Deixei um feedback sobre a aula e o uso ferramenta, destacando aspectos positivos e negativos que observou. Você também pode dar sugestões, fazer elogios e/ou críticas. Seus comentários são muito importantes para a melhoria das aulas e da ferramenta.**

.....
.....
.....

APÊNDICE

C

QUESTIONÁRIO INICIAL - EXPERIMENTO II

Você está convidado(a) a participar como voluntário da pesquisa "Ampliação das linguagens suportadas pelo Ambiente de Simulação Distribuída Automático (ASDA)", desenvolvida pela Mestranda Nathalia Lopes e orientada pela Profª Drª Sarita Mazzini Bruschi. Este questionário faz parte da pesquisa que tem como objetivo proporcionar uma melhora na qualidade da aprendizagem de avaliação de desempenho, com o auxílio da ferramenta ASDA. Você poderá deixar de participar da pesquisa a qualquer momento. Não aceitar participar não acarreta qualquer prejuízo. Mesmo que tenha iniciado sua participação, poderá desistir a qualquer momento, sem prejuízo algum. A estimativa de tempo para responder esse questionário é em torno de 5 minutos.

1. **Declaro, por meio deste termo, que concordei em participar desta pesquisa. Fui informado(a) que a pesquisa é desenvolvida pela Mestranda Nathalia Lopes e orientada pela Profª Drª Sarita Mazzini Bruschi, a quem poderei contatar a qualquer momento que julgar necessário, pelos e-mails nathalialp@usp.br ou sarita@icmc.usp.br. Afirmo que aceitei participar por minha própria vontade, sem receber qualquer incentivo financeiro ou ter qualquer ônus e com a finalidade exclusiva de colaborar para o sucesso da pesquisa. Fui informado(a) dos objetivos estritamente acadêmicos do estudo, que contribuirá para a melhora na qualidade de ensino de avaliação de desempenho. Fui também esclarecido(a) de que os usos das informações por mim oferecidas estão submetidos às normas éticas destinadas à pesquisa envolvendo seres humanos, da Comissão Nacional de Ética em Pesquisa (CONEP) do Conselho Nacional de Saúde, do Ministério da Saúde. Minha colaboração se fará de forma anônima, por meio de respostas ao questionário aqui retratado para futuras análises. O acesso e a análise dos dados coletados se farão apenas pelo(a) pesquisador(a) e/ou seu(s) orientador(es) / coordenador(es). Fui ainda informado(a) de que posso me retirar desse(a) estudo / pesquisa / programa a qualquer momento, sem prejuízo para meu acompanhamento ou sofrer quaisquer sanções ou constrangimentos.**

( ) Sim, concordo em participar.
( ) Não desejo participar

***Responder as seguintes perguntas apenas se tiver concordado em participar da pesquisa na questão anterior.**

2. **Idade:** .....

3. **Gênero:**
( ) Feminino
( ) Masculino
( ) Prefiro não dizer

4. **Você já cursou a disciplina de Avaliação de Desempenho de Sistemas?**
( ) Sim
( ) Não

5. **Sobre o tema de Avaliação de Desempenho, qual seu nível de conhecimento?**
( ) Nenhum
( ) Fraco
( ) Moderado
( ) Satisfatório
( ) Muito bom
( ) Excelente

6. **Sobre o tema de Simulação Discreta baseada em eventos, qual seu nível de conhecimento?**
( ) Nenhum
( ) Fraco
( ) Moderado
( ) Satisfatório
( ) Muito bom
( ) Excelente

7. **Sobre o tema de Bibliotecas de Simulação, qual seu nível de conhecimento?**
( ) Nenhum
( ) Fraco
( ) Moderado
( ) Satisfatório
( ) Muito bom
( ) Excelente

8. **Sobre o tema de Lógica de Programação, qual seu nível de conhecimento?**
   ( ) Nenhum
   ( ) Fraco
   ( ) Moderado
   ( ) Satisfatório
   ( ) Muito bom
   ( ) Excelente

9. **Qual seu nível de conhecimento sobre as Linguagens de Programação abaixo?**

   - **Linguagem de programação Python:**
     ( ) Nenhum
     ( ) Fraco
     ( ) Moderado
     ( ) Satisfatório
     ( ) Muito bom
     ( ) Excelente
   - **Linguagem de programação Java:**
     ( ) Nenhum
     ( ) Fraco
     ( ) Moderado
     ( ) Satisfatório
     ( ) Muito bom
     ( ) Excelente
   - **Linguagem de programação R:**
     ( ) Nenhum
     ( ) Fraco
     ( ) Moderado
     ( ) Satisfatório
     ( ) Muito bom
     ( ) Excelente
   - **Linguagem de programação C:**
     ( ) Nenhum
     ( ) Fraco
     ( ) Moderado
     ( ) Satisfatório
     ( ) Muito bom
     ( ) Excelente

APÊNDICE

D

QUESTIONÁRIO QUALITATIVO - EXPERIMENTO II

Você está convidado(a) a participar como voluntário da pesquisa "Ampliação das linguagens suportadas pelo Ambiente de Simulação Distribuída Automático (ASDA)", desenvolvida pela Mestranda Nathalia Lopes e orientada pela Profª Drª Sarita Mazzini Bruschi. Este questionário faz parte da pesquisa que tem como objetivo proporcionar uma melhora na qualidade da aprendizagem de avaliação de desempenho, com o auxílio da ferramenta ASDA. Você poderá deixar de participar da pesquisa a qualquer momento. Não aceitar participar não acarreta qualquer prejuízo. Mesmo que tenha iniciado sua participação, poderá desistir a qualquer momento, sem prejuízo algum. A estimativa de tempo para responder esse questionário é em torno de 10 minutos.

1. **Declaro, por meio deste termo, que concordei em participar desta pesquisa. Fui informado(a) que a pesquisa é desenvolvida pela Mestranda Nathalia Lopes e orientada pela Profª Drª Sarita Mazzini Bruschi, a quem poderei contatar a qualquer momento que julgar necessário, pelos e-mails nathalialp@usp.br ou sarita@icmc.usp.br. Afirmo que aceitei participar por minha própria vontade, sem receber qualquer incentivo financeiro ou ter qualquer ônus e com a finalidade exclusiva de colaborar para o sucesso da pesquisa. Fui informado(a) dos objetivos estritamente acadêmicos do estudo, que contribuirá para a melhora na qualidade de ensino de avaliação de desempenho. Fui também esclarecido(a) de que os usos das informações por mim oferecidas estão submetidos às normas éticas destinadas à pesquisa envolvendo seres humanos, da Comissão Nacional de Ética em Pesquisa (CONEP) do Conselho Nacional de Saúde, do Ministério da Saúde. Minha colaboração se fará de forma anônima, por meio de respostas ao questionário aqui retratado para futuras análises. O acesso e a análise dos dados coletados se farão apenas pelo(a) pesquisador(a)**

e/ou seu(s) orientador(es) / coordenador(es). Fui ainda informado(a) de que posso me retirar desse(a) estudo / pesquisa / programa a qualquer momento, sem prejuízo para meu acompanhamento ou sofrer quaisquer sanções ou constrangimentos.

( ) Sim, concordo em participar.

( ) Não desejo participar

*Responder as seguintes perguntas apenas se tiver concordado em participar da pesquisa na questão anterior.

De acordo com as afirmações a seguir escolha entre as opções, a resposta que mais te representa.

2. **O instrutor conseguiu transmitir o conteúdo de forma clara e objetiva.**

( ) Concordo Totalmente

( ) Concordo

( ) Não concordo, nem discordo

( ) Discordo

( ) Discordo Totalmente

3. **Eu consegui compreender todo o conteúdo abordado na aula.**

( ) Concordo Totalmente

( ) Concordo

( ) Não concordo, nem discordo

( ) Discordo

( ) Discordo Totalmente

4. **O conteúdo aprendido nesta aula será útil pra mim.**

( ) Concordo Totalmente

( ) Concordo

( ) Não concordo, nem discordo

( ) Discordo

( ) Discordo Totalmente

5. **Eu gostei de aprender sobre o assunto desta aula.**

( ) Concordo Totalmente

( ) Concordo

( ) Não concordo, nem discordo

( ) Discordo

( ) Discordo Totalmente

6. **Eu gostei do uso da ferramenta na aula.**

( ) Concordo Totalmente

( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

7. **A ferramenta foi fácil de utilizar.**
( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

8. **A atividade proposta era muito complexa.**
( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

9. **Seria mais fácil desenvolver a atividade *SEM* o uso da ferramenta.**
( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

10. **Ter um exemplo básico de um código de simulação gerado pela ferramenta, facilitou o entendimento sobre os códigos de simulação.**
( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

11. **Eu conseguiria replicar o exercício da aula sem o uso da ferramenta.**
( ) Concordo Totalmente
( ) Concordo
( ) Não concordo, nem discordo
( ) Discordo
( ) Discordo Totalmente

12. **Como você classifica o nível de dificuldade do assunto estudado nesta aula (Simulação de Sistemas)?**

( ) Muito Difícil
( ) Difícil
( ) Regular
( ) Fácil
( ) Muito fácil

13. O método de aula utilizando uma ferramenta para auxiliar no ensino foi?
( ) Muito bom
( ) Bom
( ) Regular
( ) Ruim
( ) Péssimo

14. O uso da ferramenta foi relevante para a aula?
( ) Muito relevante
( ) Relevante
( ) Mais ou menos relevante
( ) Pouco relevante
( ) Nada relevante

15. O quanto você teria conseguido desenvolver da atividade proposta em aula se não tivesse utilizado a ferramenta?
( ) A atividade toda e um pouco mais
( ) A atividade toda
( ) Metade da atividade
( ) Um pouco da atividade
( ) Não conseguiria desenvolver nada

16. Quais foram suas maiores dificuldades sobre o conteúdo da aula?
.....
.....
.....

17. Quais as suas impressões sobre a ferramenta? Foi uma boa proposta utiliza-la em aula? É útil? Precisa de melhorias?
.....
.....
.....

18. Deixei um feedback sobre a aula e o uso ferramenta, destacando aspectos positivos e negativos que observou. Você também pode dar sugestões, fazer elogios e/ou críticas. Seus comentários são muito importantes para a melhoria das aulas e da ferramenta.

.....

.....

.....

APÊNDICE

E

EXERCICIOS

E.1 Exercicio - Experimento I

Atividade: Simulação (Escalonamento de Processos).

Considere a tabela de processos abaixo:

Quadro 8 – Tempo de execução dos processos.

Processos	Tempo de Execução
A	10
B	11
C	05
D	15
E	10
F	12
G	09

Considerações:

- A ordem de chegada dos processos segue a tabela (o primeiro processo a chegar é o processo A e o último é o processo G) e todos os processos estão prontos para execução;
- Desconsidere o tempo de espera dos processos por entrada/saída;

Tarefa:

- Construa um código de simulação em python para simular o funcionamento do algoritmo de escalonamento FIFO com os processos acima e determine o tempo de retorno (turn around time - tempo até finalizar sua execução) de cada processo;
- Utilize o código de simulação gerado pela ferramenta na aula passada e faça as alterações necessárias para realizar a atividade;

- Utilize o código de simulação gerado pela ferramenta na aula passada e faça as alterações necessárias para realizar a atividade;
- Crie uma lista com com todos o processos e os tempos de execução;
- Altere a função **"def chegadaClientes(env, recursos)"** e coloque um for para percorrer apenas a lista de processos;
- Na função **"def processoCPU(env, recursos)"** faça o cálculo da chegada e saída do processo para obter o tempo de retorno;
- Também dentro da função **"def processoCPU(env, recursos)"** substitua o valor da distribuição exponencial da cpu pelo valor de cada processo na função **"yield env.timeout(distributions('cpu'))"**;
- No final imprima na tela cada processo, seu tempo de execução e o tempo de retorno.

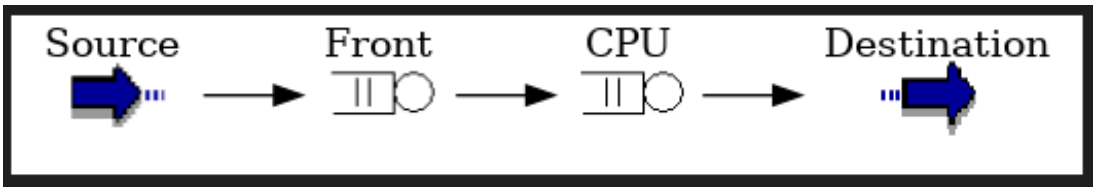
E.2 Exercicio - Experimento II

Curso: Avaliação de Desempenho de Sistemas Computacionais

Atividade: Servidor Web - Simulação.

Avaliar o desempenho de um servidor web que possui 1 front-end e 1 servidor, utilizando a técnica de Simulação.

Figura 42 – Modelo de sistema do exercício



Fonte – Elaborado pela autora.

Os objetivos da avaliação são:

- Verificar qual a utilização de cada centro de serviço em uma situação normal de carga de trabalho;
- Verificar qual o impacto no desempenho se for acrescentado um Banco de Dados - DB (centro de serviço) depois da CPU, com tempo de serviço de 2s;
- Verificar a utilização se a CPU for substituída por uma CPU duas vezes mais veloz;

- Verificar o impacto no desempenho se o DB for substituído por uma CPU (tempo de resposta).

Como discutido em sala de aula, considere que todas os valores são médias de distribuições exponenciais e utilize os seguintes valores como médias:

- Tempo médio entre as chegadas: **1s**;
- Tempo médio de serviço do FrontEnd: **0.1s**;
- Tempo médio de serviço da CPU: **1s**;

APÊNDICE

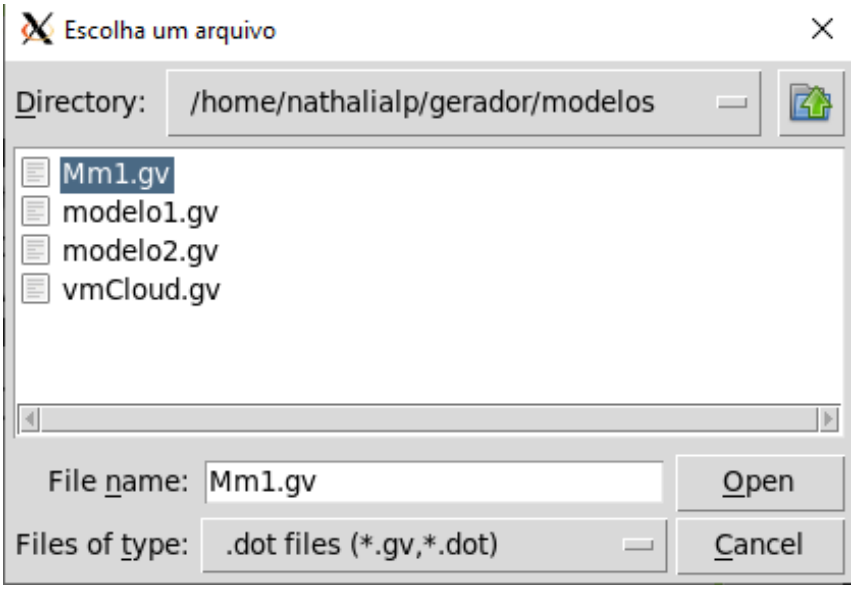
F

TELAS DO SISTEMA

Figura 43 – Tela Inicial

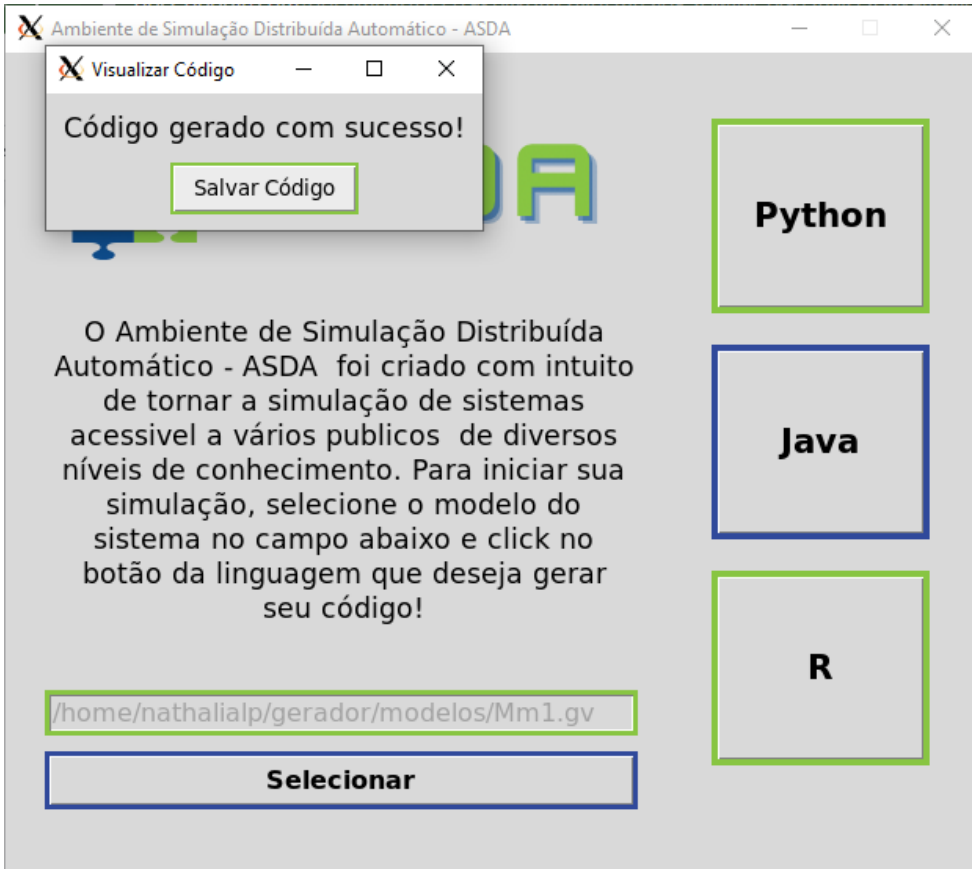
Fonte – Elaborado pela autora.

Figura 44 – Tela de seleção do modelo



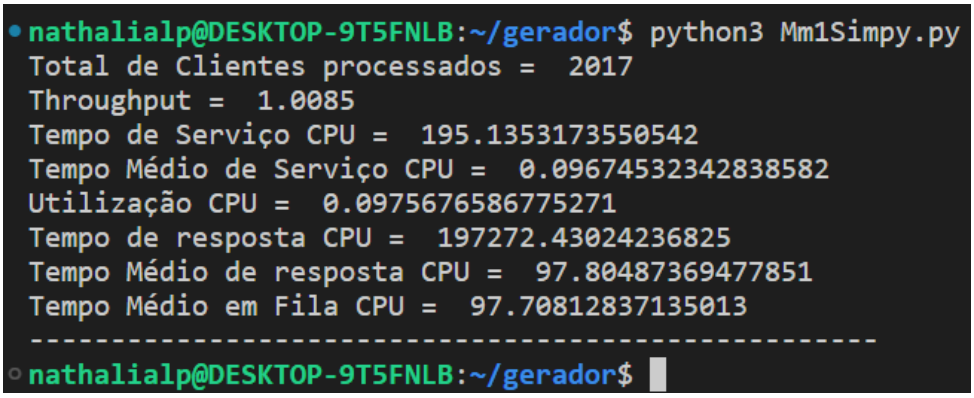
Fonte – Elaborado pela autora.

Figura 45 – Tela de geração de código



Fonte – Elaborado pela autora.

Figura 46 – Resultado da execução do código gerado



Fonte – Elaborado pela autora.

APÊNDICE

G

CÓDIGOS DOS PROGRAMAS DE SIMULAÇÃO

G.1 Python - Simpy

Código-fonte 1 – Código de Simulação M/M/1 Gerado em Python

```
1 # ------------------------------------------------------------
2 # Código gerado com o ASDA - Ambiente de Simulação Distribuída
3 # Automático
4 # ------------------------------------------------------------
5
6 import random
7 import simpy
8
9 contaChegada = 0
10 contaTerminos = 0
11 tempoServico = [0]*1
12 tempoResposta = [0]*1
13
14 # função que armazena as distribuições utilizadas no modelo
15 def distributions(tipo):
16   return {
17     'chegadas': random.expovariate(1.0/1.5),
18     'cpu': random.expovariate(1.0/0.9),
19   }.get(tipo, 0.0)
20
21 # função de chegada de clientes de acordo com os lugares que os
```

```
22 # clientes chegam
23 def chegadaClientes(env, recursos):
24   global contaChegada
25   while True:
26     contaChegada+=1
27     yield env.timeout(distributions('chegadas'))
28
29     env.process(processoCPU(env, recursos))
30
31
32 # funções que realizam o processamento dos demais nodes
33 def processoCPU(env, recursos):
34
35   global contaTerminos, tempoServico, tempoResposta
36
37   chegada = env.now
38   req = recursos[recursos.index(cpu)].request()
39   yield req
40   tempoFila = env.now - chegada
41
42   inicio = env.now
43   yield env.timeout(distributions('cpu'))
44
45   tempoServicoParcial = env.now - inicio
46
47   tempoServico[0] = tempoServico[0] + tempoServicoParcial
48
49   tempoResposta[0] = tempoResposta[0] + tempoFila +
      tempoServicoParcial
50
51   recursos[recursos.index(cpu)].release(req)
52
53   contaTerminos+=1
54
55 # define a semente ultizada para a geração aleatoria de numeros
56 random.seed(2)
57
58 # cria o ambiente de simulação
59 env = simpy.Environment()
```

```
60
61 # cria todos os recursos = facility
62 cpu = simpy.Resource(env, capacity = 1)
63
64 recursos = [
65   cpu,
66 ]
67
68 # iniciar os processos de chegada
69 env.process(chegadaClientes(env, recursos))
70
71 # define o tempo total de execução da simulação
72 env.run(until=600)
73
74 # gera os relatorios finais
75 print('Total de Clientes processados = ', contaTerminos)
76 print('Throughput = ', contaTerminos/600)
77
78 print('Tempo de Serviço CPU = ', tempoServico[0])
79 print('Tempo Médio de Serviço CPU = ', tempoServico[0]/
     contaTerminos)
80 print('Utilização CPU = ', tempoServico[0]/600)
81 print('Tempo de resposta CPU = ', tempoResposta[0])
82 print('Tempo Médio de resposta CPU = ', tempoResposta[0]/
     contaTerminos)
83 print('Tempo Médio em Fila CPU = ',(tempoResposta[0]/
     contaTerminos)-(tempoServico[0]/contaTerminos))
84 print('-------------------------------------------------')
```

G.2 Java - JavaSim

Código-fonte 2 – Código de Simulação M/M/1 Gerado Java Main.java

```
1 // ------------------------------------------------------------
2 // Código gerado com o ASDA - Ambiente de Simulação Distribuída
3 // Automático
4 // ------------------------------------------------------------
5
6 package com.javasim.teste.basic;
```

```
7
8  public class Main
9  {
10     public static void main (String[] args)
11     {
12
13         Controle m = new Controle();
14
15         // Para a thread principal e da controle do programa
       para a classe Controle
16         m.await();
17
18         System.exit(0);
19     }
20 }
```

Código-fonte 3 – Código de Simulação M/M/1 Gerado Java Controle.java

```
1  package com.javasim.teste.basic;
2
3  import org.javasim.RestartException;
4  import org.javasim.Simulation;
5  import org.javasim.SimulationException;
6  import org.javasim.SimulationProcess;
7
8  public class Controle extends SimulationProcess
9  {
10
11   public static CPU cpu = null;
12   public static Fila filaDoCPU = new Fila();
13
14
15     public static double tempoRespostaTotal = 0.0;
16     public static long totalClientes = 0;
17     public static long clientesProcessados = 0;
18     public static double totalServico = 0;
19
20
21     public Controle()
22     {
```

```
23
24      }
25
26      public void run ()
27      {
28          try
29          {
30        Chegadas chegadas = new Chegadas(1.5);
31        Controle.cpu = new CPU(0.9);
32                chegadas.activate();
33
34                Simulation.start();
35
36        hold(2000);
37
38                System.out.println("Tempo total = "+currentTime());
39                System.out.println("Total de clientes presentes no
    sistema = " + totalClientes);
40                System.out.println("Total de clientes processados =
     " + clientesProcessados);
41                System.out.println("Tempo de resposta total = " +
    tempoRespostaTotal);
42                System.out.println("Tempo médio de resposta = "
43                    + (tempoRespostaTotal / clientesProcessados));
44
45        System.out.println("Utilização do CPU = " + Controle.cpu.
    tempoDeServico);
46        System.out.println("Comprimento médio de filaCPU = "+ (
    Controle.filaDoCPU.clientesEmFila / Controle.filaDoCPU.
    checkFila));
47
48                Simulation.stop();
49
50                chegadas.terminate();
51        Controle.cpu.terminate();
52
53                SimulationProcess.mainResume();
54          }
55          catch (SimulationException e)
```

```
56          {
57          }
58          catch (RestartException e)
59          {
60          }
61      }
62      public void await ()
63      {
64          this.resumeProcess();
65          SimulationProcess.mainSuspend();
66      }
67
68  }
```

Código-fonte 4 – Código de Simulação M/M/1 Gerado Java Chegadas.java

```
1  package com.javasim.teste.basic;
2
3  import java.io.IOException;
4
5  import org.javasim.RestartException;
6  import org.javasim.SimulationException;
7  import org.javasim.SimulationProcess;
8  import org.javasim.streams.ExponentialStream;
9
10 public class Chegadas extends SimulationProcess
11 {
12     private ExponentialStream taxa;
13
14     public Chegadas(double media)
15     {
16         taxa = new ExponentialStream(media);
17     }
18
19     public void run ()
20     {
21         while (!terminated())
22         {
23             try
24             {
```

```
25                  hold(taxa.getNumber());
26              }
27              catch (SimulationException e)
28              {
29              }
30              catch (RestartException e)
31              {
32              }
33              catch (IOException e)
34              {
35              }
36
37              new Cliente();
38          }
39      }
40
41  }
```

Código-fonte 5 – Código de Simulação M/M/1 Gerado Java Cliente.java

```
1  package com.javasim.teste.basic;
2
3  import org.javasim.RestartException;
4  import org.javasim.Scheduler;
5  import org.javasim.SimulationException;
6
7  public class Cliente
8  {
9      private double tempoResposta;
10     private double tempoChegada;
11
12     public Cliente()
13     {
14         boolean vazio = false;
15
16         tempoResposta = 0.0;
17         tempoChegada = Scheduler.currentTime();
18     vazio = Controle.filaDoCPU.isEmpty();
19     Controle.filaDoCPU.enqueue(this);
20         Controle.totalClientes++;
```

```
21
22          if (vazio)
23          {
24              try
25              {
26          Controle.cpu.activate();
27              }
28              catch (SimulationException e)
29              {
30              }
31              catch (RestartException e)
32              {
33              }
34          }
35      }
36      public void finished ()
37      {
38          tempoResposta = Scheduler.currentTime() - tempoChegada;
39          Controle.tempoRespostaTotal += tempoResposta;
40      }
41
42 }
```

Código-fonte 6 – Código de Simulação M/M/1 Gerado Java CPU.java

```
1 package com.javasim.teste.basic;
2
3 import java.io.IOException;
4 import java.util.Random;
5
6 import org.javasim.RestartException;
7 import org.javasim.SimulationException;
8 import org.javasim.SimulationProcess;
9 import org.javasim.streams.ExponentialStream;
10
11 public class CPU extends SimulationProcess
12 {
13      private ExponentialStream taxa;
14      private Cliente cliente;
15      public double tempoDeServico = 0.0;
```

```
16
17   public CPU(double media)
18     {
19         taxa = new ExponentialStream(media);
20         cliente = null;
21     }
22
23     public void run ()
24     {
25         double inicioAtividade, fimAtividade;
26         boolean vazio = false;
27
28         while (!terminated())
29         {
30       while (!Controle.filaDoCPU.isEmpty())
31       {
32         inicioAtividade = currentTime();
33
34         Controle.filaDoCPU.checkFila++;
35         Controle.filaDoCPU.clientesEmFila += Controle.filaDoCPU
    .queueSize();
36         cliente = Controle.filaDoCPU.dequeue();
37
38                 try
39                 {
40                     hold(serviceTime());
41                 }
42                 catch (SimulationException e)
43                 {
44                 }
45                 catch (RestartException e)
46                 {
47                 }
48
49                 fimAtividade = currentTime();
50                 tempoDeServico += fimAtividade -
    inicioAtividade;
51                 Controle.totalServico += tempoDeServico;
52
```

```
53          Controle.clientesProcessados++;
54          cliente.finished();
55
56
57              }
58
59
60              try
61              {
62                  cancel();
63              }
64              catch (RestartException e)
65              {
66              }
67          }
68      }
69
70      public double serviceTime ()
71      {
72          try
73          {
74              return taxa.getNumber();
75          }
76          catch (IOException e)
77          {
78              return 0.0;
79          }
80      }
81
82  }
```

Código-fonte 7 – Código de Simulação M/M/1 Gerado Java Fila.java

```
1  package com.javasim.teste.basic;
2
3  import java.util.NoSuchElementException;
4
5  public class Fila
6  {
7      private Lista inicio;
```

```
8       private long tamanho;
9       public long clientesEmFila;
10      public long checkFila;
11
12      public Fila()
13      {
14          inicio = null;
15          tamanho = 0;
16          clientesEmFila = 0;
17          checkFila = 0;
18      }
19
20      public boolean isEmpty ()
21      {
22          if (tamanho == 0)
23              return true;
24          else
25              return false;
26      }
27
28      public long queueSize ()
29      {
30          return tamanho;
31      }
32
33      public Cliente dequeue () throws NoSuchElementException
34      {
35          if (isEmpty())
36              throw new NoSuchElementException();
37
38          Lista ptr = inicio;
39          inicio = inicio.proximo;
40
41          tamanho--;
42
43          return ptr.cliente;
44      }
45
46      public void enqueue (Cliente toadd)
```

```
47      {
48          if (toadd == null)
49              return;
50
51          Lista ptr = inicio;
52
53          if (isEmpty())
54          {
55              inicio = new Lista();
56              ptr = inicio;
57          }
58          else
59          {
60              while (ptr.proximo != null)
61                  ptr = ptr.proximo;
62
63              ptr.proximo = new Lista();
64              ptr = ptr.proximo;
65          }
66
67          ptr.proximo = null;
68          ptr.cliente = toadd;
69          tamanho++;
70      }
71
72
73  }
74
75  class Lista
76  {
77      public Lista()
78      {
79          cliente = null;
80          proximo = null;
81      }
82
83      public Cliente cliente;
84
85      public Lista proximo;
```

```
86 }
```

G.3 R - Simmer

Código-fonte 8 – Código de Simulação M/M/1 Gerado em R

```
1 # ------------------------------------------------------------
2 # Código gerado com o ASDA - Ambiente de Simulação Distribuída
3 # Automático
4 # ------------------------------------------------------------
5
6 library(simmer)
7
8 set.seed(20)
9
10 env <- simmer("Mm1")
11 env
12
13 # Configurar trajetória
14
15 cliente <- trajectory() %>%
16   seize("cpu", 1) %>%
17   timeout(function() rexp(1, 1/0.9)) %>%
18   release("cpu", 1) %>%
19   set_attribute("queue_cpu", function() get_queue_count(env, "
     cpu"))
20
21
22 # criando os recursos
23
24 env %>%
25 add_resource("cpu", 1) %>%
26 add_generator("cliente", cliente, function() rexp(1, 1/1.5),
     mon=2)
27
28 # tempo total de execução
29 env %>%
30   run(600) %>%
31   now()
```

```
32 # dados da simulação
33
34 chegadas <- get_mon_arrivals(env, TRUE)
35 recursos <- get_mon_resources(env)
36 fila <- get_mon_attributes(env)
37
38
39 sprintf("Total de Clientes Processados = %d", nrow(get_mon_
     arrivals(env)))
40 sprintf("Thoughput = %f", (nrow(get_mon_arrivals(env))/600))
41
42 sprintf("Tempo de Serviço CPU = %f", sum(chegadas[chegadas$
     resource == "cpu", c("activity_time")]))
43 sprintf("Tempo Médio de Serviço CPU = %f", sum(chegadas[
     chegadas$resource == "cpu", c("activity_time")])/nrow(
     chegadas[chegadas$resource == "cpu", c("resource", "name")])
     )
44 sprintf("Utilização CPU CPU = %f", sum(chegadas[chegadas$
     resource == "cpu", c("activity_time")])/600)
45 sprintf("Tempo de resposta CPU = %f", sum(chegadas[chegadas$
     resource == "cpu", c("end_time")])-sum(chegadas[chegadas$
     resource == "cpu", c("start_time")]))
46 sprintf("Tempo Médio de resposta CPU = %f", (sum(chegadas[
     chegadas$resource == "cpu", c("end_time")])-sum(chegadas[
     chegadas$resource == "cpu", c("start_time")]))/nrow(chegadas
     [chegadas$resource == "cpu", c("resource", "name")]) )
47 sprintf("Tempo Médio em Fila CPU = %f ", ((sum(chegadas[
     chegadas$resource == "cpu", c("end_time")])-sum(chegadas[
     chegadas$resource == "cpu", c("start_time")]))/nrow(chegadas
     [chegadas$resource == "cpu", c("resource", "name")]))-(sum(
     chegadas[chegadas$resource == "cpu", c("activity_time")])/
     nrow(chegadas[chegadas$resource == "cpu", c("resource", "
     name")])))
48 sprintf("Comprimento Médio de Fila CPU = %f", sum(fila[fila$
     key == "queue_cpu",c("value")])/nrow(fila[fila$key == "queue
     _cpu",c("value", "key")]))
```


ICMC
USP
SÃO CARLOS