UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUAN LUIZ GONÇALVES

Mediação tecnológica em ambientes de desenvolvimento colaborativo de softwares : Apoiando Citizen Developers

São João del-Rei
2023

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUAN LUIZ GONÇALVES

Mediação tecnológica em ambientes de desenvolvimento colaborativo de softwares : Apoiando Citizen Developers

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação. Universidade Federal de São João del-Rei.
Área de Concentração: Sistemas Distribuídos e Computação de Alto Desempenho
Orientador: Flávio Luiz Schiavoni

São João del-Rei
2023

Ficha catalográfica elaborada pela Divisão de Biblioteca (DIBIB)
e Núcleo de Tecnologia da Informação (NTINF) da UFSJ,
com os dados fornecidos pelo(a) autor(a)

G635m

Gonçalves, Luan Luiz.
Mediação tecnológica em ambientes de desenvolvimento colaborativo de softwares : Apoiando Citizen Developers / Luan Luiz Gonçalves ; orientador Flávio Luiz Schiavoni. -- São João del-Rei, 2023.
72 p.

Dissertação (Mestrado - Ciência da Computação) -- Universidade Federal de São João del-Rei, 2023.

1. Trabalho Cooperativo Auxiliado por Computador. 2. Software Livre e de Código Aberto. 3. Redes de Computadores. 4. Engenharia de Software. 5. Arte Digital. I. Luiz Schiavoni, Flávio, orient. II. Título.

ATA Nº 3

ATA DE DEFESA PÚBLICA DE DISSERTAÇÃO DE MESTRADO

Ao trigésimo primeiro dia do mês de agosto do ano de dois mil e vinte três, às 9 horas, por meio da Plataforma Virtual Google Meet, realizou-se a sessão pública de defesa de dissertação, intitulada: Mediação tecnológica em ambientes de desenvolvimento colaborativo de softwares : Apoiando Citizen Developers, de autoria do candidato Luan Luiz Gonçalves, aluno do Programa de Pós-Graduação em Ciência da Computação, em nível de Mestrado. A Comissão examinadora esteve constituída pelos professores: Dr. Flávio Luiz Schiavoni (Universidade Federal de São João del-Rei), Dr. Elder José Reioli Cirilo (Universidade Federal de São João del-Rei) e Dr. Aluizio Barbosa de Oliveira Neto (Instituto Federal de Minas Gerais). Concluídos os trabalhos de apresentação e arguição, os membros da banca consideraram a dissertação:

(X) APROVADA
( ) APROVADA COM RESTRIÇÕES *
( ) NÃO APROVADA

* Regimento do PPGCC/UFSJ. Art. 49 No caso de aprovação com restrições, a Banca Examinadora deverá registrar as alterações solicitadas, o prazo para a sua correção (considerando um prazo máximo de 90 dias para entrega da versão final) e o(s) examinador(es) que ficará(ão) responsável(is) pela avaliação final no parecer em anexo.

Nada mais havendo a tratar, o(a) Senhor(a) Presidente declarou a sessão encerrada, sendo a ata lavrada pelo mesmo, que segue assinada pelos Senhores Membros da Comissão Examinadora, com ciência do(a) aluno(a).

Dr. ALUIZIO BARBOSA DE OLIVEIRA NETO, IFMG

Examinador Externo à Instituição

Dr. ELDER JOSE REIOLI CIRILO, UFSJ

Documento assinado digitalmente
gov.br
ELDER JOSE REIOLI CIRILO
Data: 31/08/2023 14:32:41-0300
Verifique em https://validar.iti.gov.br

Examinador Interno

Dr. FLAVIO LUIZ SCHIAVONI, UFSJ

Documento assinado digitalmente
gov.br
FLAVIO LUIZ SCHIAVONI
Data: 31/08/2023 11:20:58-0300
Verifique em https://validar.iti.gov.br

Presidente

LUAN LUIZ GONCALVES

Documento assinado digitalmente
gov.br
LUAN LUIZ GONCALVES
Data: 31/08/2023 14:52:55-0300
Verifique em https://validar.iti.gov.br

Mestrando

A minha família é a força por trás de cada conquista, por isso dedico este trabalho a eles. O apoio inabalável foi a luz que iluminou cada passo desta jornada. Vocês são minha inspiração constante e meu refúgio seguro. Com amor e gratidão eterna.

AGRADECIMENTOS

À minha família, expresso minha gratidão pelo apoio incondicional, amor constante e estímulo incansável. Vocês foram meu alicerce, fornecendo a base sólida necessária para superar desafios e alcançar este objetivo. Cada conquista é também de vocês.

Aos professores que cruzaram meu caminho, especialmente ao meu orientador, que não apenas compartilhou seus valiosos conhecimentos, mas também me inspirou e encorajou nos momentos mais desafiadores. Sua dedicação e orientação foram fundamentais para minha trajetória acadêmica e para a minha evolução como ser humano, sou profundamente grato por sua influência positiva.

Aos amigos, agradeço por estarem ao meu lado nos altos e baixos, compartilhando alegrias e incentivando-me nos momentos difíceis. Sua presença tornou esta jornada mais leve e significativa, e as lembranças que construímos juntos são tesouros que levarei para toda a vida.

À imensidão do universo, expresso minha gratidão pela conspiração de eventos que tornaram possível a realização deste trabalho.

Agradeço a UFSJ pela oportunidade de concluir o mestrado, o CNPq (151975/2019-1) e a FAPEMIG (APQ-02148-18) pelo apoio financeiro.

Que este agradecimento seja apenas um reflexo da profunda gratidão que sinto por todos que fizeram parte desta jornada. A dedicação e apoio recebidos foram fundamentais para o sucesso deste trabalho, e levo comigo não apenas o conhecimento adquirido, mas também as relações construídas ao longo do caminho. Obrigado a todos que tornaram este sonho uma realidade.

“Eu leio nos jornais
Novas notícias de guerras mortais
Eu vejo muita corrupção
Enquanto irmão mata irmão”
(Ponto de Equilíbrio, O que Eu Vejo)

RESUMO

O acesso à tecnologia pode ser não apenas como produto, mas como pensamento e código liberto a alteração e a adequação para cada necessidade. A tecnologia pode ser considerada como um item de necessidade básica humana, o que eleva a importância da liberdade de acesso ao código-fonte para ampliar a colaboração e a disseminação do conhecimento. Este trabalho aborda a colaboração mediada por tecnologias baseadas em software livre, discutindo a arte tradicional e a arte digital; os valores e as liberdades do software livre; e o compartilhamento de artefatos de software em diferentes granularidades. Foi realizado um estudo descritivo-analítico à luz dos fundamentos de Computer-Supported Cooperative Work (CSCW) e de plataformas Groupware e Low-Code. O estudo resulta em requisitos e features de Groupware que apoiam a colaboração em processos criativos presentes no desenvolvimento de software e conduz um estudo de casos da implementação das features definidas, que tem a ferramenta Mosaicode como objeto do estudo. Essa ferramenta é um ambiente de programação visual, gerador de código-fonte e software livre, com o foco na geração de aplicações de software para o domínio das artes digitais. Foi desenvolvida na Universidade Federal de São João del-Rei (UFSJ), no laboratório ALICE – Arts Lab in Interfaces, Computers, and Everything Else. O trabalho contribui para a compreensão das possibilidades e desafios do desenvolvimento colaborativo nesse contexto, abrindo caminho para um ambiente mais inclusivo, cooperativo e livre, permitindo que artistas digitais e não-programadores (citizen developers)) explorem a criatividade e a colaboração em novas dimensões tecnológicas.

Palavras-chaves: Trabalho Cooperativo Auxiliado por Computador, Software Livre e de Código Aberto, Redes de Computadores, Engenharia de Software, Arte Digital.

ABSTRACT

Title: Technologically Aided Collaborative Software Development Environments: Supporting Citizen Developers

Access to technology can be not only as a product, but also as thought and liberated code adaptable for each need. Technology can be considered as an item of basic human need, which raises the importance of freedom of access to source code to expand collaboration and dissemination of knowledge. This work addresses collaboration mediated by technology based on Free/Libre and Open Source Software (FLOSS), discussing traditional art and digital art; the values and freedoms of open-source software; and the sharing of software artifacts at different levels of granularity. A descriptive-analytical study is conducted in the light of the foundations of Computer-Supported Cooperative Work (CSCW) and Groupware and Low-Code platforms. The study yields requirements and features of Groupware that support collaboration in creative processes present in software development, and it conducts a case study of the implementation of the defined features, with the tool Mosaicode as the object of the study. This tool is a visual programming environment, source code generator, and open-source software, with a focus on generating software applications for the domain of digital arts. It was developed at the Federal University of São João del-Rei (UFSJ), in the ALICE laboratory — Arts Lab in Interfaces, Computers, and Everything Else. The work contributes to understanding the possibilities and challenges of collaborative development in this context, enabling the way for a more inclusive, cooperative, and open environment, allowing digital artists and non-programmers (citizen developers) to explore creativity and collaboration in new technological dimensions.

Keywords: Computer Supported Cooperative Work, Free and Open-Source Software, Computer Networks, Software Engineering, Digital Art.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

4GL	*Fourth Generation Programming*
ACID	*acrônimo de Atomicity, Consistency, Isolation, Durability*
CAPP	*Computer AidedProcess Planning*
CM	*Congestion Manager*
CSCD	*Computer-Supported Collaborative Design*
CP	*Coordination Protocol*
CPP	*Collaborative Process Planning*
CSCW	*Computer Supported Cooperative Work*
DML	*Data Manipulation Language*
DQL	*Data Query Language*
DSL	*Domain Specific Language*
FLOSS	*Free-Libre and Open Source software*
LAN	*Local Area Network*
LCDP	*Low-Code Development Platform*
MPS	*Meta Programming System*
OAG	*Online Application Generator*
OSPD	*Open Source Software Development*
PaaS	*Platform-as-a-Service*
RAD	*Rapid Application Deve-lopment*
RV	*Realidade Virtual*
SO	*Sistema Operacional*
SST	*Structured Stream Transport*
TCP	*Transmission Control Protocol*
TI	*Tecnologia da Informação*
VPL	*Visual Programming Language*
WAN	*Wide Area Network*

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais, a tecnologia está cada vez mais presente na vida humana em todas as áreas do conhecimento, o que permitiu uma mudança da relação entre a tecnologia e a arte. A proximidade dos processos industriais com a tecnologia resultou no estreitamento da tecnologia com os meios de comunicação, sendo este estreitamente um dos responsáveis por aproximar a arte e a tecnologia. As artes passaram a se aproximar cada vez mais dos processos de comunicação e a utilizar a tecnologia como parte de seu processo criativo.

Diante deste cenário, podemos considerar a tecnologia como um item de necessidade básico humano, mas que muitas vezes não se encontra de forma disponível e acessível. A regra de quem tem acesso ou quem não tem é ditada pelo capital, causando uma segregação social (ALMEIDA; SCHIAVONI, 2018). Almeida e Schiavoni apresentaram, no artigo "Aspectos da Sustentabilidade e Colaboração na Arte Digital", uma proposta de sustentabilidade tecnológica baseada nos princípios do Software Livre, defendendo o acesso à tecnologia como um item de necessidade básica humana; e o compartilhamento e distribuição, não apenas como produto, mas como pensamento e código liberto a alteração e à adequação para cada necessidade. Para isso, o acesso ao código-fonte está diretamente relacionado a colaboração, levando em consideração que o distribuído e colaborativo depende diretamente do compartilhamento do código-fonte – "*Se não mostrarmos o que estamos fazendo, dificilmente conseguiremos cooperar*" (ALMEIDA; SCHIAVONI, 2018).

A colaboração em arte nem sempre é algo simples de ser feito. Não é simples dois pintores pintarem a mesma tela quando esta pintura se trata de colocar tinta sobre tecido ou datilografar o mesmo texto em uma máquina de escrever. No entanto, a possibilidade de colaboração mediada pela tecnologia traz para a colaboração em arte a possibilidade da cópia de um artefato de software, arquivos que trazem em si obras de arte e que podem ser compartilhados durante a sua criação. O computador, enquanto máquina de cópia perfeita, permite que um artista possa distribuir sua obra durante a sua elaboração gerando novas cópias da mesma obra e sem que isso implique em alguém ter que abrir mão do seu trabalho para que outra pessoa possa trabalhar (SCHIAVONI, 2017).

Esta tecnologia permite que a colaboração na criação de um determinado artefato de software ocorra de forma síncrona ou assíncrona com pessoas em diferentes espaços e diferentes tempos e pode ser aplicada em qualquer área do conhecimento humano, não apenas na arte. A ciência, que por definição é uma atividade colaborativa onde o conhecimento é construído de forma incremental e cumulativa, também pode se valer desta mesma tecnologia para criação de obras colaborativas (JANDRE; DIIRR; BRAGANHOLO, 2019). Assim, a tecnologia vem sendo cada vez mais capaz de apoiar o ser humano em suas atividades, incluindo as atividades colaborativas, diferentes processos que envolvem o ensinar e o aprender (BAIDOO-ANU; ANSAH, 2023).

Durante a pandemia de COVID-19 surgiu o desafio da brusca adaptação ao trabalho e ao ensino remoto, onde a comunicação entre as pessoas aconteceu por meio da interação humano-computador. A tecnologia teve o papel muito importante de apoiar esse momento, permitindo uma parcela da população continuar trabalhando e estudando (SPALDING et al., 2020). No entanto, apesar de a tecnologia estar presente no dia-a-dia das pessoas para o compartilhamento de seus cotidianos e para o consumo de arte e entretenimento, a mesma não conseguiu oferecer o mesmo suporte para a criação artística ou científica colaborativa e podemos tentar entender as razões pelas quais tal colaboração ainda é tão difícil de ser feita.

O primeiro ponto é que muitas tecnologias não são livres e visam o benefício financeiro para pessoas específicas ou organizações. Além de restringir o uso, o software proprietário restringe o acesso ao código-fonte onde o software é uma "caixa preta" que impede que todos possam fazer o reuso do código e do conhecimento ali presente. O não uso do software livre diverge da ideia de colaboração e pode significar maior gasto de energia, desgaste humano e uma segregação social. Com o software livre todo conhecimento é compartilhado, podendo ser estudado, utilizado, adaptado e distribuído, colaborando não apenas com a ciência mas com todas as áreas do saber que dependem ou fazem uso da tecnologia (ALMEIDA; SCHIAVONI, 2018; ALMEIDA; SCHIAVONI, 2017).

As empresas fazem o uso da ciência e das tecnologias resultantes das pesquisas científicas, mas muitas vezes restringem o acesso a este conhecimento. Optar por não lucrar com software proprietários ou por abrir o código-fonte de uma aplicação pode parecer não ser vantajoso. No entanto, as liberdades do software livre provêm um cenário transparente, mais sustentável e propício a uma evolução não restrita, onde a colaboração é a essência. Não compartilhar um conhecimento implica em não colaborar para a evolução do estado da arte de tópicos os quais todos desfrutam. O software proprietário pode ser utilizado em um processo criativo, mas limita a colaboração ao abster-se das liberdades existentes no software livre.

O segundo ponto importante diz respeito à comunicação. Diferentes dispositivos de computação – denominados sistemas ou hospedeiros (hosts) finais – conseguem trocar mensagens via redes de computadores. Os sistemas finais podem estar em uma mesma rede (LAN/WLAN) ou não. A interconexão dessas redes resulta na internet, conectando sistemas finais localizados em diferentes locais do mundo. A internet apresenta uma estrutura complexa que é separada em camadas bem definidas e com responsabilidades específicas (arquitetura de camadas), cada uma com os seus protocolos de comunicação (camada de protocolo) formando uma pilha protocolos: "*Um* ***protocolo*** *define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no reconhecimento de uma mensagem ou outro evento*" (KUROSE, 2014).

Os protocolos de rede podem ser abertos, para que as entidades saibam como se comunicar, ou fechados de forma que apenas uma empresa saiba como comunicar. As entidades trabalham em conjuntos, fazendo com que as mensagens do sistema final de origem sejam

fragmentadas em pacotes, percorram o núcleo da rede (enlaces de comunicação e rede de comutadores de pacote) e cheguem no sistema final de destino. Esse cenário é um exemplo de como a transparência e a colaboração são fundamentais para que as redes de computadores e a internet funcionem eficientemente. Assim como o próprio significado de colaboração descreve um ato benéfico a todos envolvidos – se alguém não se envolve, pode ser que não queira o benefício de todos, mas apenas para si.

O terceiro ponto que elencamos diz respeito aos produtos desenvolvidos no processo colaborativo. A engenharia de software chama estes produtos de artefato de software e estes artefatos podem ter formatos abertos, que podem ser consumidos, produzidos e modificados por diversas ferramentas, ou formatos fechados, que apenas uma ferramenta é capaz de modificar seu conteúdo.

1.1 Premissas do trabalho

Este trabalho parte também das seguintes premissas:

- O cenário de colaboração pode ser limitado pelas licenças envolvidas na criação de um determinado artefato, já que nem todos os artefatos/softwares permitem o acesso e a cópia do seu código-fonte, sendo uma barreira na para a colaboração. Por essa razão, o software livre é adotado neste trabalho a fim de alcançar um ambiente colaborativo com todas as liberdades presentes no software, sem nenhuma restrição na colaboração mediada por tecnologias.

- Outro ponto considerado aqui são os atores presentes na criação de artefatos de software por leigos, em especial, na área de arte digital. Artistas digitais não possuem obrigatoriamente um conhecimento avançado em computação, conhecimento dominado por profissionais da área de TI (Tecnologia da Informação).

- O trabalho remoto tem sido mais compatível na área da computação, porque os profissionais da área já trabalham na frente do computador e contam com diferentes ferramentas para apoiar essa maneira de trabalhar. Com isso, existem diversas ferramentas para profissionais desta área para permitir e coordenar a colaboração de pares no desenvolvimento de algumas tarefas. Exemplos destas ferramentas são sistemas de controle de versão de código e ferramentas de gestão de projetos.

- Muitos requisitos não funcionais envolvidos na questão da colaboração e cooperação são problemas em abertos, como criptografia, encapsulamento e compactação de mensagens de redes, e precisam ser definidos para prover segurança de informações e otimizar/apoiar a comunicação por esse meio. Também é importante discutir possibilidades de conflitos, *locks* e transações *ACID* durante a colaboração síncrona. Além de permitir a edição de

diagramas de forma colaborativa é importante um modo de sugestão de alterações e implementação de uma interface para diff e debugging. No entanto, apesar de entender a importância de tais discussões, não será abordado tais requisitos neste trabalho.

1.2 Objetivos

Os processos criativos existentes nas artes, na computação e demais áreas podem ser apoiados pelo computador de forma a prover uma colaboração entre pessoas em locais físicos distintos. Diante do exposto, este trabalho disserta sobre a colaboração apoiada pelo computador, no contexto da arte digital, mas não se restringindo ao mesmo. Guiado por um estudo analítico descritivo à luz dos fundamentos de *Low-Code Development Platform*, este trabalho tem o objetivo geral de designar características (features) desejáveis a um ambiente de criação de arte digital para apoiar os usuários finais dessa ferramenta, em especial, os citizen developers.

Como resultado esperado, será apresentado uma série de funcionalidades desejáveis em sistemas que permitem a colaboração e também uma discussão sobre os recursos para a implementação das features desejáveis para tais sistemas de maneira que os mesmos possam oferecer suporte à colaboração de seus usuários finais, como: compartilhamento de bibliotecas de recursos, modelos, padrões de projeto, processos e ferramentas. Alguns requisitos de groupware foram escolhidos para oferecer um suporte inicial, sendo eles:

1. Compartilhamento de artefatos;
2. Cooperação em ambiente compartilhado;
3. Canais de comunicação.

1.3 Cenário atual

O Mosaicode[1] é um ambiente de programação visual e gerador de código-fonte, com o foco na geração de aplicações de software para o domínio de Arte Digital – envolvendo áreas da computação como: Inteligência Artificial, Computação Gráfica, Computação Musical, Redes de Computadores, Realidade Virtual e Visão Computacional (GONÇALVES; SCHIAVONI, 2020a).

Essa ferramenta foi desenvolvida pelo grupo de pesquisa ALICE[2] – acrônimo para Arts Lab in Interfaces, Computers, and (Education, Exceptions, Experiences, Entertainment, Environment, Entropy, Errors, Everything, Else and Etcetera...) – localizado no Departamento de Computação (DCOMP) da Universidade Federal de São João del-Rei (UFSJ) (SCHIAVONI et al., 2019). A Figura 1 apresenta um screenshot do Mosaicode.

[1] https://github.com/Alice-ArtsLab/mosaicode
[2] https://alice.dcomp.ufsj.edu.br/

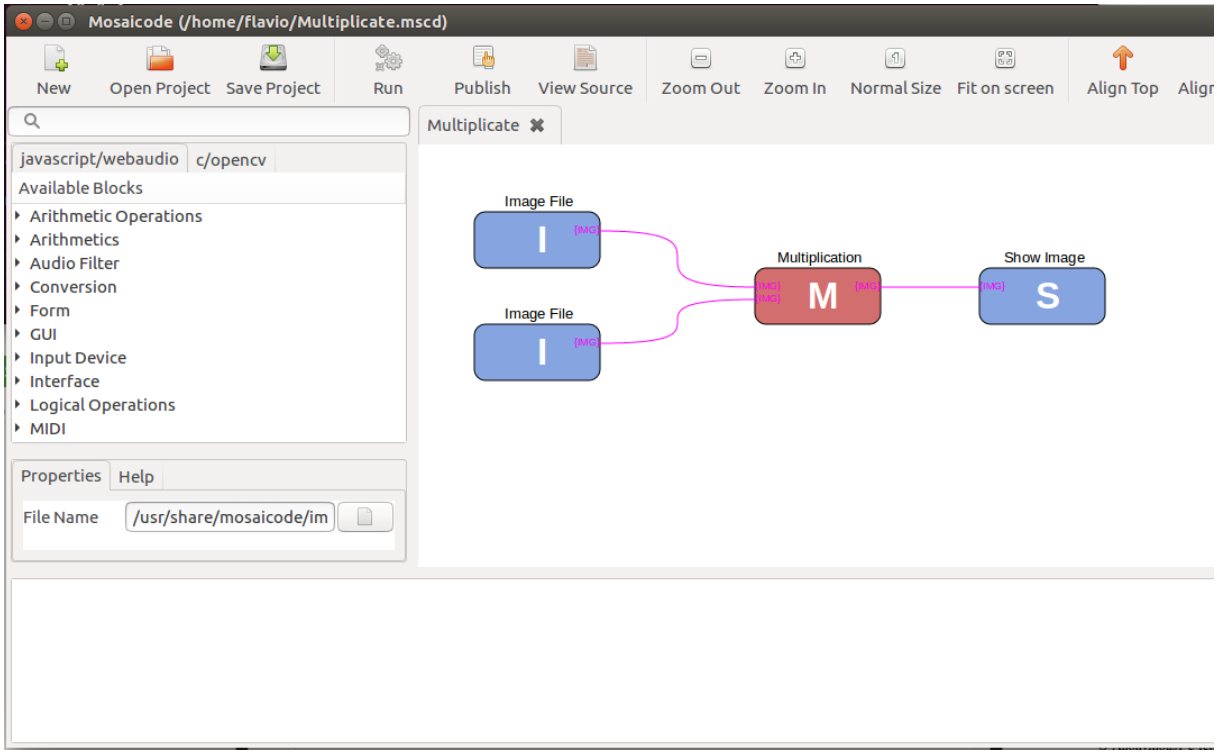

Figura 1 – Screenshot do Mosaicode.

Seguindo a proposta de apoio aos citizen developers no domínio de Arte Digital e de CSCW, presente no Capítulo 4. A ferramenta Mosaicode foi utilizada como objeto de estudo de casos (Capítulo 5) para auxiliar o levantamento de funcionalidades, features, ferramentas e outros artefatos que apoiam a colaboração remota. O mesmo estudo pode ser aplicado em outras ferramentas, para um mesmo ou outros domínios.

Features de colaboração são fundamentadas nos conceitos de LCDPs e abordam a colaboração/cooperação entre os usuários da ferramenta. Requisitos iniciais foram definidos a fim de atender os requisitos exigidos em plataformas Low-Code e Groupware.

É possível estender a ferramenta implementado nos artefatos que compõem uma VPL/DSL, gerando código para a linguagem e domínio definidos – esse conjunto de artefatos é chamado de extensão, pela comunidade do Mosaicode [3]. Dessa forma será definido um workspace colaborativo que permita o compartilhamento destes artefatos com as features levantadas.

1.4 Justificativa

No contexto deste trabalho, é apresentado as *Low-Code Development Platforms* (LCDPs), plataformas que adotam o paradigma de programação visual (VPL – Visual Programming Language) para desenvolvimento de aplicações de software. Podem ser utilizadas para desenvolver e implantar aplicações de software funcionais e completas, atendendo os requisitos e apoiando

[3] https://github.com/Alice-ArtsLab/mosaicode

os usuários finais. LCDPs/VPLs apoiam programadores, oferecendo um mecanismo rápido de prototipação, e não-programadores (citizen developers), permitindo-os desenvolver aplicações sem o conhecimento que profissionais da área de TI possuem (WASZKOWSKI, 2019).

Uma feature importante para LCDPs é o suporte ao desenvolvimento cooperativo (SAHAY et al., 2020). Softwares que apoiam o trabalho cooperativo são classificados como *groupware* – conceito da área de pesquisa Computer-Supported Cooperativo Work (CSCW), que estuda o uso de tecnologias de computação e comunicação para apoiar atividades em grupo (NABBEN, 2019).

A área de pesquisa CSCW fundamenta o estudo deste trabalho sobre a feature de suporte à colaboração em ambientes virtuais de desenvolvimento de softwares. Assim, LCDPs e o domínio CSCW se unem para apoiar os citizen developers.

1.5 Organização do trabalho

Este trabalho está organizado da seguinte forma. Os trabalhos relacionados estão presentes no Capítulo 2. No Capítulo 3 são apresentados os conceitos envolvidos neste trabalho, abordando formas de apoio aos citizen developers. O Mosaicode é apresentado como ferramenta de apoio a esses usuários finais.

O Capítulo 4 apresenta a definição da proposta deste trabalho. Foram descritos quatro cenários a fim de elucidar possibilidades de colaboração em arte em diferentes âmbitos: 1) a sala de aula de um curso de arte digital utilizando o Mosaicode (LAN); 2) colaboração em uma sala de aula remota de um curso online de arte digital (WAN); 3) uma equipe de 4 artistas programadores que eventualmente se encontram presencialmente está trabalhando no desenvolvimento de um projeto (WAN/LAN); 4) Colaboração remota em um ambiente de Realidade Virtual (WAN). Os cenários contribuíram com a análise de requisitos.

Também foi abordado a gestão de usuários, projetos e workspace, diferentes artefatos, comunicações e funcionalidades.

Em sequência, no Capítulo 5 é realizado um estudo de casos, onde o Mosaicode é o objeto. É aplicado a proposta apresentada no Capítulo 5, resultando em artefatos, funcionalidades, gerenciadores, modelo e fluxo de comunicação, ferramentas, definição do servidor de relay e mensagens de rede.

Os resultados e discussões são realizados no Capítulo 6. O trabalho é concluído no Capítulo 7 – incluindo trabalhos futuros. Por fim, é apresentado as referências e anexos.

2 TRABALHOS RELACIONADOS

Os seguintes trabalhos são relacionados ao apoio à citizen developers e preocupações existentes com a difusão de aplicações desenvolvidas por estes usuários.

Low-code platform for automating business processes in manufactuing

Neste trabalho, Waszkowski apresenta como resultado uma pesquisa uma plataforma low-code para automatizar processos de negócios na manufatura, chamada Aurea BPM. Essa plataforma fornece suporte para modelagem, automação, gerenciamento e otimização de processos de negócios. Entre os requisitos funcionais e não funcionais, dessa ferramenta, temos: Acesso remoto, segurança, interface gráfica do usuário, confiabilidade, relatórios, sistema administrativo, processos de produção de amostra, modelagem de processos de negócios, e coleta de estatísticas e relatórios. Esse tipo de plataforma low-code é uma abordagem nova e inovadora, sendo capaz de reduzir significativamente a redução de custo e tempo para o desenvolvimento e manutenção de processos (WASZKOWSKI, 2019).

Supporting the understanding and comparison of low-code development platforms

Sahay e seus coatores examinaram oito LCDPs, consideradas líderes no mercado relacionado, relatórios recentes do Gartner (VINCENT et al., 2019) e Forrester (RYMER et al., 2019). Os autores afirmam que essas ferramentas são representativas para o trabalho proposto, que tem como objetivo final facilitar o entendimento e a comparação de LCDP, que podem identificar os requisitos importantes dessas plataformas, atendendo os usuários finais. O diagrama de features presente na Figura 2, considera LCDP como feature abstrata e define as features concretas, obrigatórias e opcionais para atender a feature abstrata (SAHAY et al., 2020).

Challenges & Opportunities in Low-Code Testing

Khorram e seus coautores propõem uma lista de features e possíveis valores, que podem ser utilizadas como linha de base para comparar componentes de teste de Low-Code e como diretriz para construir novos. Resultado de uma análise inicial de componentes de teste de cinco LCDP comerciais e com base nos princípios de Low-Code, essa lista serviu para especificar o estado de componentes de teste das ferramentas investigadas, introduzir os desafios de teste para low-code e como ponto inicial para pesquisas futuras na área de teste de low-code. Três questões foram consideradas para introduzir os desafios: o papel do citizen developer no teste; a

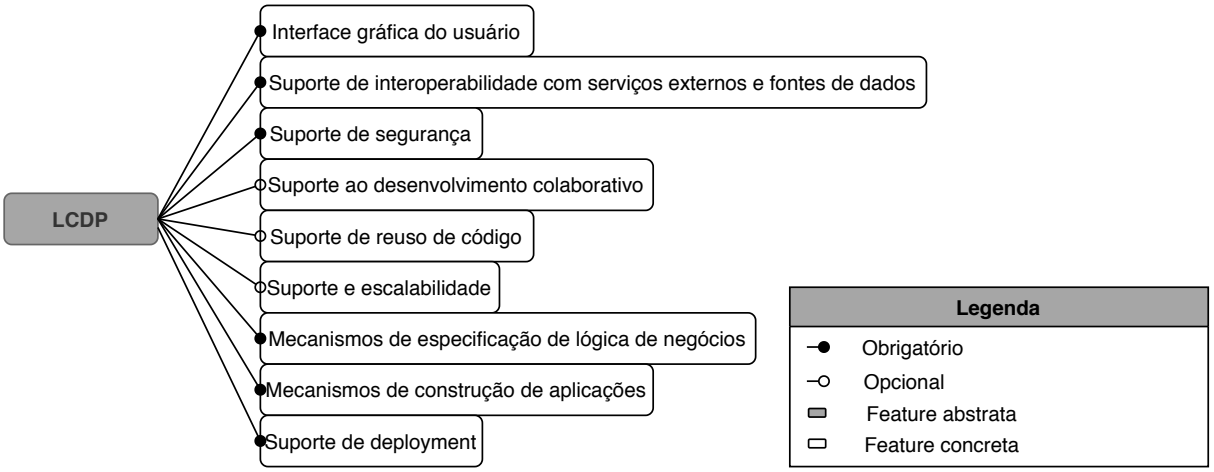

Figura 2 – Diagrama de features: oriundas da averiguação de requisitos presentes em oito LCDPs relevantes (SAHAY et al., 2020).

necessidade de automação de teste de alto nível; e teste de nuvem. O trabalho também proveu referências para o estado da arte para especificar as dificuldades e oportunidades do ponto de vista acadêmico (KHORRAM; MOTTU; SUNYé, 2020).

The Rise of the Citizen Developer: Assessing the Security Impact of Online App Generators

Os autores discutem e avaliam o impacto de segurança relacionados às aplicações móveis desenvolvidas por citizen developers, especificamente desenvolvida utilizando geradores de aplicativos online (online application generators – OAGs). Essas ferramentas automatizam o desenvolvimento, distribuição e manutenção de aplicativos, sendo atraentes para citizen developers. Falhas de segurança presentes nos geradores podem afetar milhares de aplicativos gerados pelo mesmo, influenciando na segurança geral do ecossistema móvel. Entre os 2.291.898 aplicativos Android gratuitos do Google Play que foram avaliados, pelo menos 11,1% foram gerados por OAGs e $7/13$ desse percentual são baseados em código clichê sujeito a ataques de reconfiguração. Também foi identificado vulnerabilidades de injeção de código e WebViews inseguros, problemas de segurança em sua infraestrutura. Caixas pretas presentes no ambiente de desenvolvimento podem violar a suposição de confiança do usuário final, colocando os dados confidenciais e a privacidade dos usuários finais em risco, devido a problemas ocultos. Os autores concluem que os OAGs de fato têm um fator de amplificação significativo para essas vulnerabilidades, prejudicando notavelmente a saúde do ecossistema geral de aplicativos móveis (Oltrogge et al., 2018).

Towards Automating the Construction of Recommender Systems for Low-Code Development Platforms

Para aliviar o alto investimento de tempo comumente demandado para o desenvolvimento de sistemas de recomendação, devido a necessidade de escolha do método de recomendação adequado, sua implementação e configuração para o problemas e domínio em questão, assim como a avaliação da precisão das recomendações. Almonte e seus coatores apresentam os primeiros passos em direção a um framework genérico dirigido por modelos (generic model-driven framework). Esse framework pode capacitar a geração de sistemas de recomendação orientados à tarefas ad-hoc para sua integração em plataformas low-code. Resultados preliminares, apresentados como prova de conceito, demonstram a capacidade de fornecer recomendações relevantes no contexto avaliado (ALMONTE et al., 2020).

UAISharing - Interface Universal de Acesso a Recursos Compartilhados

Sandy verifica a inexistência de uma padronização para o compartilhamento de recursos criados pelos mais diversos tipos de usuários de sistemas computacionais. Sistemas de informações envolvem particularidades que trazem a necessidade de definir uma camada de abstração para oferecer uma forma transparente de compartilhamento desses recursos. Nesse contexto, é proposto um modelo de empacotamento e distribuição de recursos produzidos por usuários finais. Métodos, modelos, protocolos, processos, técnicas e estruturas, descritos nos trabalhos a seguir, foram aderidos a este trabalho a fim de apoiar o desenvolvimento do mesmo (SANDY, 2019).

Data Communication Method in Collaborative Process Planning

Su e Han apresenta um método de comunicação de dados e o fluxo de sistemas *Collaborative Process Planning* (CPP). CPP é aplicação de CSCW na área *Computer Aided Process Planning* (CAPP) para desenvolvimento de aplicações CSCW, provendo um ambiente de trabalho colaborativo em computadores, redes e ferramentas de colaboração. O foco principal deste trabalho é garantir a troca de dados em tempo real e eficiente. A Figura 3 apresenta o modelo de comunicação para collaborative process planning, proposto por Su e Han. Esse modelo utiliza IP multicast para encaminhar mensagens para grupos específicos (SU; HAN, 2018).

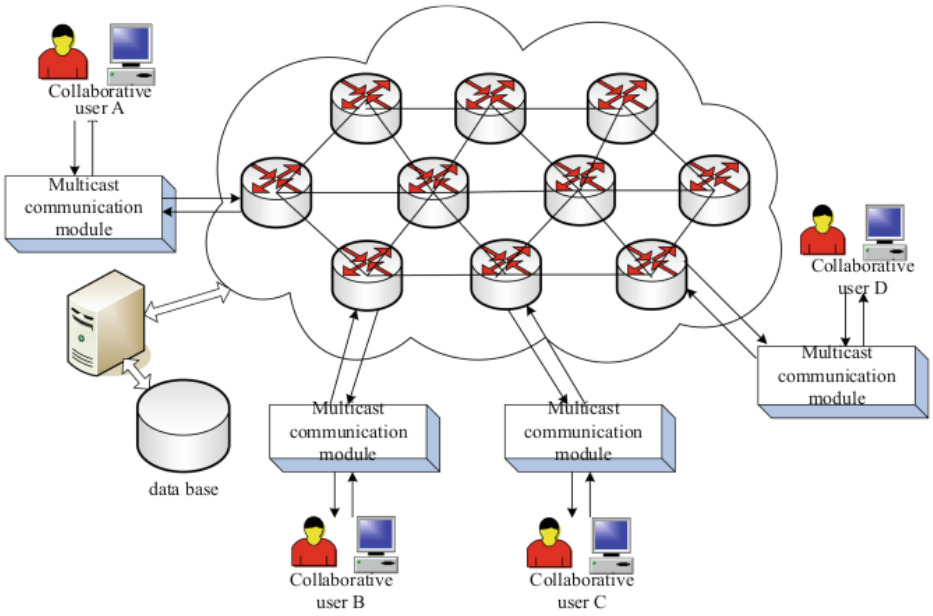

Figura 3 – Modelo de comunicação (SU; HAN, 2018).

MediaSynch Issues for Computer-Supported Cooperative Work

Mayer-Patel apresentam um overview de sistemas CSCW, incluindo arquiteturas de comunicação, desafios de sincronização, técnicas e protocolos em redes de computadores. Os autores descrevem e caracterizam problemas de sincronização de media e rever a evolução de protocolos multi-streaming e técnicas – incluindo: Structured Stream Transport (SST), Congestion Manager (CM), TCP trunking, e Coordination Protocol (CP) (MAYER-PATEL, 2018).

Enhancing collaboration efficiency through tailorability in synchronous groupware

O progresso vertiginoso na adoção da internet impulsionou notáveis avanços em outras esferas tecnológicas, como dispositivos móveis e a Internet das Coisas. Diante das crescentes exigências de conectividade e mobilidade, os fornecedores de software embarcam em uma acirrada competição para trazer aplicações inovadoras, algumas mais dependentes de conexões à internet com maior largura de banda e qualidade do sinal. A utilização de sistemas de conferência em vídeo, para fins de comunicação e colaboração em grandes instituições, cresceu consideravelmente. Desenvolvida no Instituto de Engenharia de Computação, a plataforma PASSENGER foi implementada para sistemas operacionais Windows, apresentando um mecanismo de controle de chão para viabilizar a cooperação síncrona distribuída. Essa plataforma é uma aplicação groupware síncrona destinada a pequenos grupos cooperando em atividades de design de software (HIRLEHEI, 2019).

Descriptive theory of awareness for groupware development

Sob a perspectiva da engenharia de software, o trabalho apresenta uma revisão de vários mecanismos, frameworks e usos de awareness propostos na literatura. O foco recai sobre os elementos a serem levados em conta durante o planejamento e a implementação de mecanismos de awareness em ferramentas de groupware, levando a uma teoria descritiva de consciência com o intuito de proporcionar suporte para o processo de desenvolvimento desse tipo de ferramenta. Também é realizado um estudo de caso detalhado, demonstrando a utilização do framework proposto (COLLAZOS et al., 2019).

A Survey of Development Strategies for Collaborative Systems

Estratégias de software que visam apoiar o desenvolvimento de sistemas colaborativos são levantadas, analisadas, classificadas e comparadas com base em diferentes critérios. Os resultados obtidos apontam para a inexistência de um método ou estratégia de desenvolvimento que envolve todo o ciclo de vida dessas aplicações. No entanto, nota-se que diversas abordagens se complementam, o que abre uma oportunidade promissora para o avanço do estado da arte neste domínio de estudo (CANCHÉ; OCHOA, 2019).

A novel systematic method to evaluate computer-supported

Declarações de requisitos do CSCD foram criadas partindo do mapeamento e categorização sistemática de 220 fatores que influenciam o sucesso desse domínio específico. O trabalho apresenta o primeiro método sistemático que permite que equipes de projeto de engenharia avaliem e selecionem as tecnologias CSCD mais adequadas. Essa avaliação é realizada comparando a funcionalidade das tecnologias com os requisitos do projeto, estabelecidos na literatura revisada por pares (BRISCO; WHITFIELD; GRIERSON, 2020).

Supporting the understanding and comparison of low-code development platforms

É apresentado uma exploração técnica sobre diferentes LCDPs, fundamentada em um framework conceitual comparativo proposto. De forma específica, ao analisar oito LCDPs representativas, um conjunto correspondente de características foi identificado para extrair as funcionalidades e os serviços que cada plataforma é capaz de suportar. O objetivo consiste em viabilizar a compreensão e a comparação das plataformas low-code para apoiar a escolha de qual melhor se adequa aos requisitos do usuário (SAHAY et al., 2020).

3 CONCEITOS ENVOLVIDOS

Neste Capítulo apresentaremos alguns conceitos envolvidos que serão utilizados ao longo do texto.

3.1 Programação sem código - Low-Code Programming

A programação sem código - *Low-Code Programmimg* (LCP) é um conceito na área da programação, que possui por intuito reduzir o esforço/tempo gasto no desenvolvimento de software e também permitir outras abordagens para o desenvolvimento e adequação de sistemas computacionais. Derivado da ideologia da Quarta geração de programação - *fourth generation programming* (4GL) e conceitos do modelo de processo de Desenvolvimento rápido de aplicações - *Rapid Application Development* (RAD), o Low-Code Programmig permite que programadores gastem menos tempo pensando na sintaxe do código, concentrando na implementação da funcionalidade e estética da aplicação. As abordagens *Model-driven software development*, *Automatic code generation* e o paradigma de programação visual também são bases na criação deste conceito (WASZKOWSKI, 2019). Lidar com a escassez de desenvolvedores profissionais qualificados é o objetivo principal para LCDPs. Para esse propósito, usuários finais, sem "nenhum" conhecimento de programação – *citizen developers* (no domínio LCDP), podem contribuir no processo de desenvolvimento de software utilizando essas plataformas (SAHAY et al., 2020).

Associado a este conceito, temos também o conceito da Plataforma de desenvolvimento de aplicações sem código - *Low-Code Development Platform* (LCDP), que adota o paradigma de programação visual (VPL – Visual Programming Language) para desenvolvimento de software usando este conceito. Com isso, temos ambientes que permitem a programação por meio de linguagens de programação visual - *Visual Programming languages* (VPL) que permitem que aplicações sejam desenvolvidas sem que o programador tenha que escrever seu código-fonte. Tais linguagens costumam ser linguagens específicas para um determinado domínio - *Domain Specific (Programming) Languages* (DSL) e não permitem o desenvolvimento de qualquer aplicação, como ocorre nas linguagens de programação de propósito geral - *General Purpose (Programming) Languages* (GPL).

LCPDs apoiam programadores, oferecendo um mecanismo rápido de prototipação, e não-programadores, permitindo-os desenvolver aplicações sem o conhecimento que programadores possuem (WASZKOWSKI, 2019) permitindo, assim, a popularização da programação entre leigos. LCDPs permitem ainda o desenvolvimento e implantação de aplicações de software, funcionais e completas, apoiando usuários finais (não desenvolvedores) e provido na nuvem (Cloud) aplicando o modelo Platform-as-a-Service (PaaS) (SAHAY et al., 2020).

3.2 Citizen Developers

A utilização de LCPD permite que usuários leigos se tornem programadores e possam participar da criação de aplicações de software. Esses desenvolvedores são chamados de *citizen developers*, pessoas sem experiências em programação que desenvolvem aplicações utilizando ferramentas para este fim, como o paradigma visual em plataformas *Low-Code* (WASZKOWSKI, 2019).

A inclusão de citizen developers no desenvolvimento de software resultou em aplicações desenvolvidas por esse grupo de desenvolvedores e que costumam ser usada apenas pelo próprio desenvolvedor ou por usuários próximos. Essa situação foi chamada de "Citizen Development", reportada por Joe McKendrick (analista na Unisphere Research) em 2017, no texto intitulado "The Rise of the Empowered Citizen Developer: Is IT Possible Without IT" (WASZKOWSKI, 2019).

Com o surgimento de citizen developers, ferramentas desenvolvidas por esses usuários finais – utilizando LCDPs e outros ambientes de programação visual – tornam-se cada vez mais existentes. Esse acontecimento gerou novas lacunas no desenvolvimento de software, como preocupações relacionadas à questões de segurança dos aplicativos desenvolvido por esses usuários (Oltrogge et al., 2018) e também no gerenciamento dos artefatos criados por este programador.

Os citizen developers não tem o conjunto de tecnologias (stack) que podem ser utilizados para o desenvolvimento de aplicações, especialmente se compararmos tais desenvolvedores com profissionais de TI. O stack de desenvolvimento inclui ferramentas de colaboração remota, de controle de versão, de empacotamento e de distribuição de software.

Expandindo este conceito, podemos assumir que diversas ferramentas permitem que seu usuário crie artefatos de software. Tais artefatos podem ser o objetivo final desta ferramenta, como um documento gerado por um editor de texto, ou artefatos que irão compor a ferramenta, como arquivos de fontes instalados em um editor de texto. Neste caso, a fonte a ser instalada em um editor de texto irá complementar o próprio editor e se tornar parte da ferramenta, expandindo suas possibilidades e tornando-a personalizada para seu usuário final.

Para entender melhor este contexto, considere que documentos da ferramenta Microsoft Word, por exemplo, são artefatos de software, sendo também um software, mas esses tipos de documentos gerados por usuários finais não são tratados da mesma forma que os artefatos gerados por programadores durante o processo de desenvolvimento. Estes artefatos não são armazenados com informações sobre versões, autores, modificações, não possuem mecanismos para atualização automática, caso ocorra uma mudança nos mesmos por outros usuários, não têm licenças ou estruturas para informar sobre sua distribuição e modificação. Ocorre o mesmo com arquivos de customizações, configurações e plugins desta mesma ferramenta.

3.3 3C e CSCW

O modelo 3C de colaboração foi pensado para apoiar o desenvolvimento de aplicações cooperativamente, também chamado de Trabalho Cooperativo Apoiado por Computador - *Computer Supported Cooperative Work* – CSCW. Juntamente com este conceito e modelo, surgiram aplicações que apoiam o trabalho cooperativo em computadores, classificados como *groupware* (NABBEN, 2019). Conforme ilustrado na Figura 4, um grupo de pessoas organizada para realizar uma tarefa coletivamente demanda de ferramentas que auxiliem no trabalho coletivo. Esta organização permite separar o desenvolvimento coletivo de tarefas em três tipos de serviços: comunicação (troca de mensagens), cooperação (registro e compartilhamento) e coordenação (políticas de acesso) (FUKS et al., 2008).

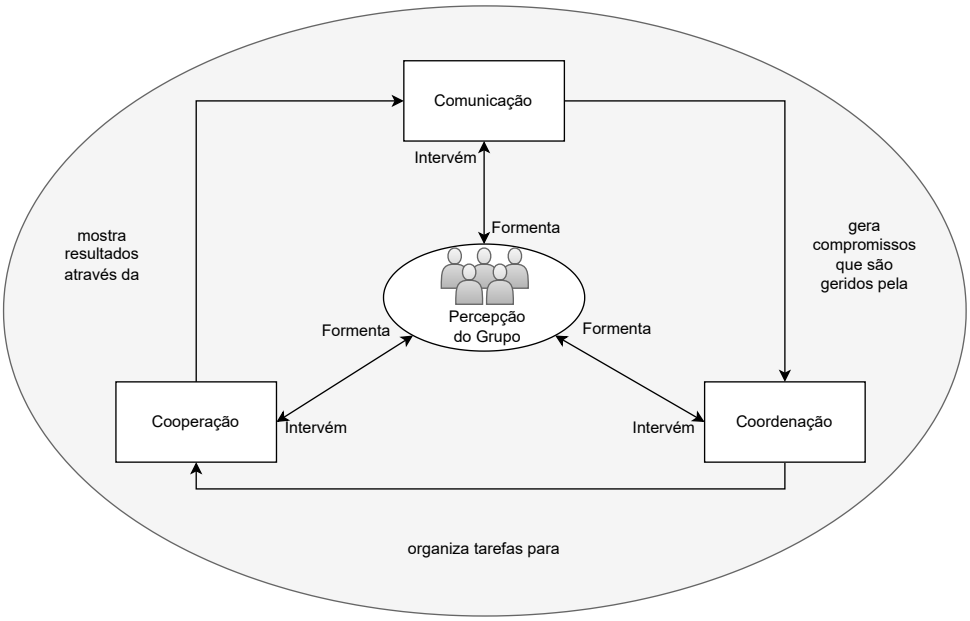

Figura 4 – Modelo 3C de colaboração (FUKS et al., 2008).

Entendendo que a colaboração pode ocorrer de maneira síncrona (ao mesmo tempo) ou assíncrona (em momentos separados) e no mesmo local ou de maneira remota, podemos também, classificar os sistemas CSCW por tempo e espaço, como ilustrado na Figura 5. As comunicações síncronas/synchronous e assíncronas/asynchronous determinam a classificação por tempo. O local pode ser classificado como co-located (mesmo local) ou remote (locais diferentes) (MAYER-PATEL, 2018).

3.4 Workspace

A colaboração mediada por tecnologia ocorre em um ambiente virtual, chamado de workspace ou espaço de trabalho. Com uma conta criada, o usuário tem a liberdade de criar ou participar de workspaces colaborativos.

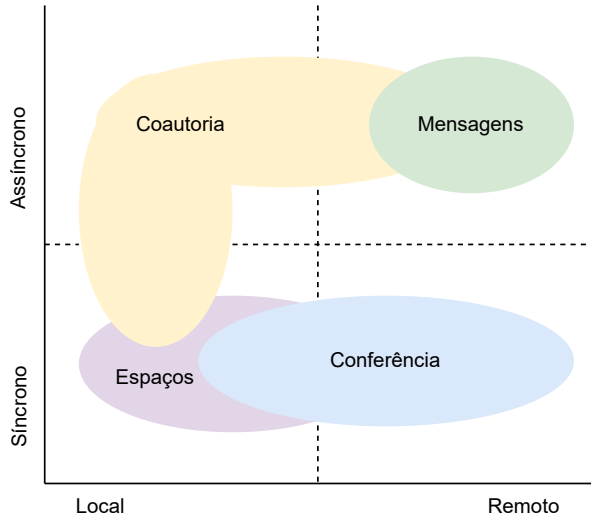

Figura 5 – Classificação dos sistemas CSCW (MAYER-PATEL, 2018).

Workspaces e projetos podem ser clonados, sendo mais um artefato possível de ser compartilhado, funcionando de forma semelhante à funcionalidade fork do github. Todos os artefatos compartilhados no workspace devem ser versionados, permitindo voltar em qualquer versão, ser auditados e também conter um valor hash como identificador único. A versão dos artefatos serão de acordo a última versão do Versionamento Semântico [1]

O workspace de um usuário poderá ter vários projetos e as permissões podem ser definidas por workspace ou projeto. Para implementar o workspace é necessário estudar como será feito atualizações para os usuários, definindo em qual granularidade deve ser feito o compartilhamento de artefato e quais informações devem ser amarradas aos artefatos. As informações adicionais (nome autor, data de criação, data de modificação e o número da versão) é importante para ter maior controle no empacotamento e distribuição dos artefatos.

3.5 Artefatos

Artefatos são subprodutos concretos de software gerados no desenvolvimento de uma aplicação por seus desenvolvedores. No desenvolvimento de uma aplicação, diversos artefatos são gerados como o código-fonte, e também documentações, relatórios, arquivos de log, imagens, arquivos de configuração, scripts, e demais subprodutos. Artefatos gerados no desenvolvimento de software podem ser armazenados em repositórios locais ou remotos, com o acesso público ou privado. Além dos artefatos gerados pelo próprio programador, outros artefatos podem ser necessários no workspace para que o desenvolvimento da aplicação seja possível como bibliotecas e outras dependências. Ao compartilhar um determinado artefato de software durante o seu processo de desenvolvimento, dependências amarradas precisam estar presentes e também

[1] https://semver.org/lang/pt-BR/

ser compartilhadas. Na ausência, o artefato dependente se torna inútil. Considerando essa dependência, uma biblioteca de software se faz necessária para que o artefato seja funcional.

Quando o software passa a ser utilizado por seu usuário final, o mesmo poderá também gerar artefatos por meio de sua utilização. Os artefatos gerados por citizen developers também possuem tais características, mesmo em um código com a estrutura de dados bem desacoplada é comum artefatos terem dependências de outros.

Um exemplo simples disso é uma diagramação feita em um editor de texto e que depende de algumas fontes tipográficas para que a mesma seja modificada. No caso do Mosaicode, que permite o usuário gerar artefatos criando novos recursos, compartilhar um diagrama com um outro usuário, apenas com a informação de quais blocos são utilizados e como foram conectados, não garante que o mesmo irá funcionar. Os artefatos utilizados na criação do diagramas podem existir apenas no computador do usuário que o criou ou adquiriu instalando uma extensão.

3.6 Mosaicode

O Mosaicode é um ambiente de programação visual e gerador de código, desenvolvido no laboratório ALICE. A programação visual no Mosaicode consiste na criação de um diagrama composto por blocos interconectados por meio de conexões entre portas de saída e de entrada. A partir do diagrama (algoritmo) é possível gerar o código-fonte escrito em uma linguagem de programação específica (não visual).

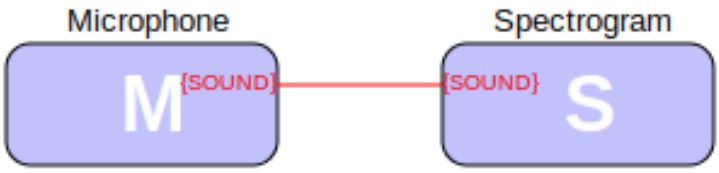

Figura 6 – Diagrama do Mosaicode: Algoritmo implementado utilizando VPL.

A Figura 6 apresenta a implementação de um aplicação (Figura 7) para o domínio de Computação Musical. Essa aplicação permite analisar o sinal de áudio do microfone (dispositivo de entrada) visualizando o espectrograma - gráfico representando o sinal de áudio no domínio de frequência.

Este algoritmo é composto por dois blocos (Microphone e Spectrogram) e uma conexão. A conexão é realizada entre duas portas do tipo *SOUND*, permitindo o fluxo de dados da porta de saída (do bloco Microfone) para a porta de entrada (do bloco Spectrogram).

É possível notar a simplicidade, praticidade e rapidez para implementar esta aplicação utilizando a VPL disponível no Mosaicode. Dois blocos e uma conexão foram suficientes para desenvolver a aplicação de áudio que permite visualizar o espectrograma do microfone.

Figura 7 – Aplicação gerada a partir do diagrama da Figura 6

O código-fonte gerado é mais complexo de entender e extenso, disponível no Anexo A, exigindo mais do programador em relação à habilidade e o conhecimento da linguagem de programação. A implementação utilizando a VPL do Mosaicode consiste apenas em escolher blocos, conectá-los e definir os valores dos seus parâmetros de entradas (propriedades dos blocos). Isso permite ao programador manter maior foco na lógica e conhecimento do domínio da aplicação desenvolvida, por não ter dificuldades no uso da linguagem de programação.

Os blocos podem ter propriedades estáticas e dinâmicas, que são parâmetros para atribuição de valores. As propriedades estáticas permitem a atribuição de valores durante a implementação do diagrama, preenchendo campos na janela de propriedades estáticas (interface gráfica do usuário). Os dados transmitidos pela conexão entre blocos são propriedades dinâmicas, atribuídas em tempo de execução da aplicação.

Por ser um ambiente de programação visual para domínios específicos, o Mosaicode herda o apoio oferecido por VPLs e DSLs. Os membros do ALICE preocupam-se com o suporte todos os usuários finais e de diversas formas: documentações, funcionalidades, escolha de licença do software, entre outras.

O plugin Library Manager[2] permite a extensão do Mosaicode utilizando o próprio ambiente, pelo preenchimento de campos disponível na interface gráfico do usuário, ao invés de escrever um arquivo XML ou Python, sendo outra forma de apoio ao citizen developers. A possibilidade de criar, na própria ferramenta, novos recursos para serem utilizados no processo de desenvolvimento do produto final faz do Mosaicode um *Meta Programming System* (MPS).

A lista a seguir apresenta os artefatos do Mosaicode a serem compartilhados durante o processo de colaboração, mediante as features de colaboração. É possível notar as dependências na composição de cada artefato:

[2] https://github.com/Alice-ArtsLab/mosaicode-plugin-libmanager

1. *Diagrama*: composto por blocos, portas e conexões, o diagrama é um algoritmo implementado pelo usuário final do Mosaicode, utilizando uma VPL. A partir do diagrama, é possível gerar o código-fonte da aplicação implementada;

2. *Bloco*: representa uma abstração de código que implementa um recurso, para uma finalidade;

3. *Porta*: permite receber e enviar fluxo de dados por meio da conexão entre os blocos pelas suas portas de entrada e saída;

4. *Conexão*: permite conectar uma porta de saída de um bloco à uma porta de entrada de outro bloco, do mesmo tipo.

5. *Code template*: define um template padrão para a geração de código-fonte;

6. *Extensão*: conjunto de artefatos – blocos, conexões, portas e code template – que compõe uma linguagem de programação visual para uma domínio específico;

7. *Plugin*: permite estender a ferramenta Mosaicode, adicionando novas funcionalidades;

8. *Code-gen*: código-fonte gerado a partir do diagrama.

3.7 FLOSS

Um dos objetivos do Mosaicode é dar suporte ao ensino de Arte Digital, alinhando com as ideias do software livre que tem como valores essenciais a liberdade e cooperação, disseminando o conhecimento humano (FLOSS e Educação [3]).

A utilização de licença de software livre e código aberto é também uma forma de apoio aos usuários finais (FLOSS[4] – Free/Libre and Open Source Software). Embora frequentemente esses softwares sejam distribuídos de forma gratuita, os benefícios desse tipo de licença vão muito além disso, permitindo a execução, cópia, redistribuição, estudo e melhoria do software – descrito em quatro liberdades essenciais. É necessário que os softwares desenvolvidos na academia sejam FLOSS para compartilhar todo o conhecimento envolvido e não apenas um produto. Criar uma caixa preta é criar barreiras na disseminação do conhecimento e na evolução da ciência, também pode comprometer a confiabilidade do sistema para os usuários finais (Oltrogge et al., 2018).

Além do conhecimento presente no código-fonte, trabalhos científicos de membros do laboratório ALICE compartilham conhecimento em formatos acadêmicos. Mesmos que os trabalhos publicados em periódicos e em conferências não sejam disponibilizados de forma livre e aberta, eles são disponibilizados dessa maneira no site do ALICE na página do menu publicações [5] (GONÇALVES; SCHIAVONI, 2020b; SCHIAVONI; GONÇALVES, 2017).

[3] https://www.gnu.org/education/education.pt-br.html
[4] https://www.gnu.org/philosophy/floss-and-foss.pt-br.html
[5] https://alice.dcomp.ufsj.edu.br/en/publications.html

4 PROPOSTA: CSCW E ARTE DIGITAL

As tecnologias existentes que são utilizadas por equipes de desenvolvimento de software para a colaboração podem também auxiliar a arte tradicional e digital a trabalhar de maneira colaborativa. Isso implica pensar em sistemas CSCW para criação de arte digital que possam ser síncronos ou assíncronos e que permitam trabalhar de maneira local ou remota.

Já existem linguagens e ambientes de programação visual para o domínio da Arte Digital, permitindo usuários leigos desenvolverem softwares. A programação visual é uma maneira de apoiar citizen developers, por ser um paradigma de programação que não exige o domínio de uma linguagem de programação. A programação é realizada de forma trivial, arrastando e conectando componentes. Assim, o desenvolvedor foca no conhecimento específico do domínio durante o processo de desenvolvimento, sem se preocupar em aprender e decorar comandos de uma linguagem de programação textual.

Além disso, tais ambientes de criação em arte digital e as ferramentas existentes para este domínio podem permitir colaboração em diversas situações como o ensino de arte ou a criação de objetos de arte.

Foi escolhido para este trabalho seguir com o apoio para os citizen developers no domínio da Arte Digital, de maneira que possa ser aplicado em outros domínios. O suporte envolve oferecer os recursos necessários para que citizen developers possam trabalhar de forma colaborativa.

4.1 Cenários de colaboração

Para elucidar as possibilidades de trabalhos colaborativos em Arte, apresentaremos alguns cenários de colaboração com intenção de listar os requisitos de usuários e de sistemas em cada um. Além disso, é possível, nestes cenários, classificar os mesmos por tipo de comunicação (síncrona e assíncrona) e por tipo de serviço (comunicação, colaboração/cooperação e coordenação).

Cenário 1 - A sala de aula de um curso de arte digital

Em uma sala de aula de um curso de arte digital, um professor e três alunos estão em uma aula presencial de criação de sintetizadores de som usando, para isso, uma ferramenta de programação visual voltada para a Arte. Os quatro usuários estão conectados em uma rede local de computadores, na qual se comunicam para trocar artefatos. Cada um está com o seu computador, mas conectados no mesmo workspace e compartilhando o mesmo ambiente de trabalho (**workarea**). Dessa forma, o professor e os alunos visualizam e editam o mesmo código, programando de maneira colaborativa. Nesse ambiente ocorre o processo de criação com a manipulação (edição/criação) e o compartilhamento de diferentes artefatos de software. A manipulação e o compartilhamento destes artefatos pode ser síncrono e/ou assíncrono.

Enquanto estão programando, caso algum usuário adicione na workarea um artefato que outro usuário não tem, como uma biblioteca ou plugin, esse artefato é compartilhado via rede. Assim, o usuário poderá obter todos os artefatos presentes na workarea para gerar o código-fonte a partir do diagrama e consequentemente executá-lo.

Nesse cenário, os usuários podem se comunicar sem o uso da tecnologia. Estão na mesma sala, próximos o suficiente para interagirem. Mesmo assim, notificações e mensagens podem auxiliar essa experiência presencial, exibindo mensagens de conflitos e de alterações, informações dos usuários integrantes do workspace e informações de alterações no código.

A coordenação do projeto poderá ser simples, envolvendo pequenos cronogramas de atividades e distribuições de tarefas.

Cooperação	síncrona
Comunicação	síncrona e descentralizada
Coordenação	simples

Tabela 1 – Análise do primeiro cenário em função dos requisitos selecionados

Cenário 2 - Uma sala de aula remota de um curso online de arte digital

O processo ocorrido entre os usuários no cenário anterior (cenário 1) acontece neste cenário, mas conectados remotamente, por meio de uma rede WAN. Todo o processo acontece com a interação humano computador, interagindo pelas interfaces de usuário que os conectam com a máquina, sem a interação direta do usuário com outro usuário.

Neste caso, é importante que, para além do compartilhamento de artefatos, as ferramentas envolvidas nesta atividade permitam também a comunicação, por texto, áudio ou vídeo, e até mesmo o compartilhamento de telas. Tanto o compartilhamento de artefatos quanto a comunicação deve ser possível de acontecer de maneira síncrona ou assíncrona.

Neste cenário podem surgir questões e preocupações sobre latência na comunicação em rede, situação esta que é pouco provável de acontecer no cenário anterior. Isso implica em pensar que transações e modificações não irão ocorrer em tempo real e que, por isso, é mais crítico gerenciar o histórico de modificação por artefato. Questões relacionadas à disponibilidade de um recurso remoto ou de tomada de decisões que devem acontecer caso ocorra uma interrupção na comunicação em rede também devem ser pensadas. Por esta questão de disponibilidade, este cenário sugere a utilização de um servidor central como servidor de replicação - *Relay server*, que pode servir para centralizar os dados a serem distribuídos entre todos os usuários e garantir a disponibilidade dos mesmos na WAN.

A coordenação poderá ser mais complexa pois deve ser possível que um participante que não conseguiu estar conectado o tempo todo no sistema possa interagir e receber tarefas em outro momento.

Cooperação	síncrona e assíncrona
Comunicação	síncrona e assíncrona e centralizada
Coordenação	mais complexa e assíncrona

Tabela 2 – Análise do segundo cenário em função dos requisitos selecionados

Cenário 3 - Um grupo de artistas programadores trabalhando no desenvolvimento de um projeto

Neste cenário os programadores desenvolvem um projeto de forma colaborativa, variando o local e a maneira de trabalhar. Parte do tempo estarão em locais separados conectados pela Internet ou no mesmo espaço conectados em uma rede local, trabalhando de forma síncrona em ambos. Em outro momento o trabalho é realizado de maneira assíncrona, onde cada programador segue o trabalho de forma individual e desconectado de redes de computadores, havendo a necessidade de mesclar o trabalho de cada um e resolver conflitos de versões caso existam.

A comunicação e a troca de artefatos também devem ocorrer tanto de maneira síncrona quanto assíncrona e deve ser possível escolher a maneira de colaborar com cada membro do grupo de maneira que a rede possa ser totalmente heterogênea. Deve ser possível ter conversas síncronas, ou em tempo real, caso estejam trabalhando ao mesmo tempo, ou conversas assíncrona no caso de um usuário não estar presente e precisar ser informado de alguma decisão do projeto. O mesmo irá acontecer em relação ao gerenciamento de artefatos. Em um momento pode ser importante a utilização síncrona de artefatos, com mais de um usuário trabalhando em um mesmo artefato ao mesmo tempo, e em outros momentos deve ser possível a colaboração assíncrona, com o usuário recebendo uma cópia dos artefatos que os outros usuários criaram / modificaram.

Este cenário também sugere a existência de um servidor de replicação que permita que artefatos e conversas estejam disponíveis sempre, mesmo que os demais usuários estejam desconectados em um determinado momento. O cenário também sugere que a coordenação do projeto será mais complexa e exigirá mais esforço de comunicação para gerenciar a equipe e as atividades.

Cooperação	síncrona e assíncrona
Comunicação	síncrona assíncrona / centralizada / descentralizada
Coordenação	mais complexa e assíncrona

Tabela 3 – Análise do terceiro cenário em função dos requisitos selecionados

4.1.1 Cenário 4 - Colaboração remota em um ambiente RV (WAN)

A Realidade Virtual (RV), é um conceito com origem em meados dos anos 80 que vem sendo cada vez mais explorado e que está caminhando para oferecer tecnologias/hardwares mais acessíveis e ubíquos(as). O ambiente colaborativo virtual permite a interação entre usuários geograficamente distantes ao se reunirem num local virtual, permitindo vivenciar experiências

semelhantes ao que acontece presencialmente e cenários que são possíveis apenas num ambiente de RV (COELHO, 2016).

Alunos e professores se reúnem em um ambiente de RV que virtualiza o âmbito educacional. Esse cenário apresenta uma interação diferente dos demais, que comumente tem como interface de interação humano-computador o teclado e mouse. O ambiente de RV apresenta novos hardwares que oferecem uma experiência imersiva, trabalhando com inúmeras percepções de realidades e possibilitando experiências em ambientes tridimensionais e multissensoriais, que provê novas formas de interação do estudante com o conhecimento (RESENDE; SANTOS, 2019).

No ambiente de RV os usuários podem ter a sensação de interagir diretamente com os artefatos, utilizando hardwares que proporcionam essa experiência. A RV tem muitos desafios, incluindo a acessibilidade às tecnologias. É importante partir do estado da arte para conhecer os hardwares (sensores e atuadores) e softwares adequados, quais tecnologias podem apoiar a proposta para alcançar um modelo que atenda e traga benefícios para o ensino/aprendizado.

Compartilhamento de artefato	síncrono e assíncrono
Cooperação	síncrona e assíncrona
Comunicação	síncrona assíncrona / centralizada / descentralizada
Coordenação	mais complexa e assíncrona

Tabela 4 – Análise do quarto cenário em função dos requisitos selecionados

4.2 Analisando requisitos dos cenários propostos

O trabalho "Descriptive theory of awareness for groupware development" (COLLAZOS et al., 2019), apresenta um framework com diretrizes e recomendações gerais adequadas para orientação do desenvolvimento de groupware. Partindo da feature de colaboração e com base nesse trabalho citado, foram inicialmente elencadas as seguinte features para o sistema proposto:

- Cooperação em ambiente compartilhado;
- Canais de comunicação;
- Coordenação.

Essas quatro features se relacionam com os três pilares do modelo 3C e com a abordagem denominada Engenharia de Groupware, baseada na Engenharia de Software e aprimorada com conceitos originários das áreas de CSCW e outras relacionadas (FUKS et al., 2004). Pensando nos três elementos desse modelo (cooperação/colaboração, comunicação e coordenação), o compartilhamento de artefatos permite que exista a colaboração entre os usuários, compartilhando artefatos de forma síncrona e assíncrona; a cooperação em ambiente compartilhado permitirá que

usuários desenvolvam um mesma aplicação, mesmo quando localizados em ambientes físicos distintos (em redes LAN ou WAN); e os canais de comunicação permitirá a interação entre os usuário durante esses desenvolvimento cooperativo.

A implementação dessas três features traz uma série de responsabilidades e cuidados necessários, uma vez lidando com usuários conectados em redes de computadores. Surgem novas questões de segurança, sistema de usuário, regras entre os usuários, entre outros detalhes que podem ser associados a coordenação, no modelo 3C.

4.3 Gestão de usuários e projetos

Gestão de usuário

Ao analisar os cenários propostos, é possível notar que pode ser necessário definir diferentes papéis para realizar as tarefas previstas. No primeiro e no segundo cenário, por exemplo, temos o papel de professor e de aluno, que poderiam ter acessos distintos aos recursos do ambiente. Já o terceiro cenário pode ter diversos usuários realizando tarefas distintas. Estes usuários com papéis distintos podem utilizar ferramentas distintas ou mesmo ter acessos distintos ao sistema como um todo.

Isso é bastante claro em equipes de desenvolvimento de software onde existem papéis como desenvolvedor de front end, back end, DBA, gerente de projeto, entre outros. Certamente estes papéis não implicam necessariamente em hierarquia e um mesmo indivíduo pode assumir mais de um papel. No entanto, a gestão de usuários pode ser importante para um sistema colaborativo.

O diagrama de caso de uso é comumente utilizado na modelagem de software para expressar o diálogo entre usuários e sistemas, e ajuda a identificar requisitos funcionais (BOURQUE; FAIRLEY; SOCIETY, 2014). A Figura 8 apresenta o diagrama de casos de uso para elicitação de requisitos deste trabalho. O diagrama está bem simplificado, comparando com complexidade do sistema referente a todos os detalhes de implementação que ainda devem ser investigados futuramente, de acordo com as referências deste trabalho e o conhecimento dos autores sobre o desenvolvimento de software. Esse diagrama é um importante passo inicial e sua simplicidade ajuda a ter uma visão geral das funcionalidades do sistema na perspectiva do usuário.

O usuário pode desenvolver uma aplicação individualmente ou optar por programar em cooperação com outros usuários. Para cooperar, é necessário identificar quais modificações foram feitas por quais usuários. Com a sessão de usuário ativa, o usuário poderá cooperar: compartilhar o seu ambiente de programação ou acessar um ambiente compartilhado; comunicar e compartilhar artefatos.

Todos as funcionalidades presentes no diagrama (Figura 8) devem ser coordenadas pelo

Figura 8 – Diagrama de caso de uso: diálogo entre o sistema e usuários final

sistema, aplicando regras na cooperação e sistema de usuários, assim como garantir a segurança do sistema/usuário.

Gestão de projetos

A gestão de usuários pode variar de um projeto para outro. Em um projeto um indivíduo pode ser o gerente enquanto em outro projeto o mesmo indivíduo pode ser o programador. Desta maneira, a gestão de usuários pode ser atrelada a gestão de projetos de forma que os papéis possam mudar dependendo da necessidade.

Nos cenários apresentados podemos imaginar que os alunos poderiam ser professores em outro contexto, no caso dos cenários um e dois, e que os envolvidos no terceiro e quarto cenário podem trabalhar em mais de um projeto.

Além disso, gerenciar um projeto pode incluir gerenciar seu cronograma e suas atividades, delegar responsabilidades para membros da equipe, de maneira que os participantes conheçam suas tarefas, os artefatos incluídos nas mesmas e também seus prazos.

4.4 Gestão de workspace

Cada projeto irá possuir um conjunto de artefatos específicos do próprio projeto, ao que denominamos . Imaginando que os indivíduos podem trabalhar em mais de um projeto e que os projetos podem ter características muito distintas, como diferentes **ferramentas** ou mesmo

diferentes linguagens de programação. Assim, podemos associar o papel do indivíduo ao projeto e definir seus atributos privados e públicos para cada artefato ou coleção de artefatos do projeto.

Os papéis e usuários definidos por projetos também são importante para auditar alterações em artefatos e identificar os participantes no ambiente colaborativo. Alguns artefatos são abertos, mas outros podem precisar de um armazenamento seguro e criptografado, exigindo a autenticação. Isso porque artefatos do sistema podem armazenar senhas e outros dados pessoais do usuário, que carecem de proteção e acesso restrito.

Para criar um workspace, primeiramente o usuário precisa ter uma conta registrada com os dados pessoais e um identificador único. Isso se faz importante para identificar cada usuário do sistema, permitindo ter um coordenação da comunidade de usuário. Nesse cenário, temos usuários com diferentes papéis em relação ao workspace. A princípio é possível pensar no usuário dono, administrador e participante do workspace. Os donos e os administradores têm total permissão para editar as configurações do workspace, a diferença é que o dono não pode ser excluído e nenhum outro usuário pode mudar o seu papel, apenas ele mesmo. Os usuários participantes, após ser aceito a solicitação de participar do workspace (pelo dono ou administrador), podem apenas utilizar os recursos de colaboração.

Além desses três papéis, seria interessante pensar em uma forma de definir novos papéis. Assim, seria possível adaptar o workspace em diferentes cenários de colaboração. Por exemplo, em uma sala de aula como ambiente colaborativo, podemos definir os papéis de professores e alunos de forma que o professor consiga coordenar a aula. Dentro do workspace os usuários poderão criar projetos para trabalhar, sendo dono do projeto criado e definindo as permissões da mesma forma que ocorre no workspace.

Durante a utilização do workspace artefatos são compartilhados entre usuários, de forma síncrona e assíncrona. Na comunicação síncrona o emissor e o receptor entram em sintonia, operando em um ritmo coordenado durante o compartilhamento de dados em tempo real. Isso exige que ambos estejam conectados para que os dados sejam enviados e recebidos de forma síncrona, trocando também mensagens de confirmação de recebimento. A colaboração síncrona acontece na **workarea** de um workspace específico, espaço compartilhado para cooperação, onde toda alteração realizada na máquina de usuário reflete em tempo real na máquina dos outros usuários.

Por outro lado, a comunicação assíncrona não depende dos usuários estarem sincronizados no tempo. Os dados são enviados independentemente, sem precisar de uma resposta imediata do usuário receptor. Dessa forma, os usuários podem trabalhar sozinhos e de modo offline, atualizando o workspace de todos os usuários ao ficar online novamente.

Ferramentas e extensões

Como ferramentas podem ser estendidas, é preciso considerar o fato de que usuários podem não ter as mesmas extensões da ferramenta disponíveis no computador. É necessário ter os artefatos bem definidos e desacoplados, assim como as suas dependências para que o artefato compartilhado funcione corretamente na máquina do receptor. A cooperação em ambiente compartilhado depende de os membros de uma equipe possuírem as mesmas ferramentas com os mesmos recursos para que mais de um membro possa realizar a mesma tarefa. Também é importante atribuir aos workspaces os plugins que tem permissão para serem utilizados, para que os usuários não possam burlar o sistema implementando uma feature que permite fazer alterações indevidas na aplicação.

Os usuários finais devem saber se o artefato que está sendo disponibilizado pela rede é a última versão, quem foi o autor, quais dependências este artefato possui e quais as modificações que a última versão teve, entre outras informações que podem ser imprescindíveis para que o artefato compartilhado seja efetivamente útil a outros usuários. É preciso definir uma forma de empacotar os artefatos para serem distribuídos, às informações e dependências necessárias.

Conflitos de versões podem existir na comunicação síncrona e assíncrona, sendo necessário definir a visualização e como devem ser realizadas a resolução dos conflitos e o que os artefatos precisam para que isso ocorra. Para que trabalhos não se percam, implementar um log de alterações permitiria voltar em qualquer versão para resolver os conflitos, exigindo a implementação de uma visualização do diff das versões.

Além de obter artefatos, os usuários também precisam distribuir os artefatos desenvolvidos. Assim, devido às considerações citadas anteriormente, é necessário que tais artefatos possuam um certo controle para garantir a efetividade de seu compartilhamento. Entre as possibilidades que temos para isso é utilizar uma ferramenta já existente e que traz tais garantias. A ferramenta *git*, por exemplo, pode ser integrada na implementação do modelo de empacotamento e distribuição, permitindo essa troca de artefatos – aproveitando as funcionalidades da ferramenta para controle e comparação de versões (diff), mantendo o log de alterações que possibilita o *undo* infinito (SANDY, 2019).

Os artefatos dos usuários podem ser disponibilizados uma biblioteca de artefatos, algo similar à lojas de aplicativos como o Play Store, plataforma da empresa Google para smartphones com o sistema operacional Android. Um lugar onde diferentes artefatos podem ser baixados, permitindo também avaliar e comentar. Um modelo de empacotamento de distribuição pode ser implementado, definindo protocolos para essa trocas de artefatos entres usuários finais. Esse modelo pode ser semelhante à ferramentas como *apt* e *pip*, que permitem usuários finais obterem softwares empacotados e distribuídos em repositórios.

4.5 Comentários, sugestões e issues

A coordenação pode depender do acompanhamento da evolução dos artefatos e a modificação de um artefato pode precisar de discussões sobre o mesmo. Assim, para acompanhar as modificações e coordenar estes trabalhos, é importante que os artefatos permitam comentários e sugestões de forma que a comunicação da evolução do mesmo possa acontecer. Os comentários e sugestões podem ter marcas de revisão, informando se já foi revisado ou não, e quais foram as alterações referentes à revisão.

O conceito de issues pode ser aplicado aqui da mesma maneira que é feito no GitHub, sendo um meio de comunicação para relatar problemas, discutir sobre melhorias e novas implementações (bugs, refatoração e features). Uma issue pode ter um responsável, ser categorizada pela criticidade e qualquer outra categoria relevante, ter status de resolução, ser vinculadas a diferentes artefatos e ter um espaço para discussão onde os colaboradores podem adicionar comentários.

Apesar de tais características servirem para a comunicação entre desenvolvedores, esta comunicação está amarrada diretamente ao artefato modificado. Por isso, estes comentários e sugestões podem ser considerados complementares a uma discussão feita por meio de chat ou email, por exemplo.

4.6 Comunicação

Esta seção aborda a comunicação entre os usuários e as comunicações do sistema, durante o processo colaborativo.

Comunicação entre usuários

Os usuários precisam se comunicar entre si e isso pode acontecer em um chat que permite trocar diferentes tipos de mídia como: texto, documentos, áudio, imagem, vídeo, compartilhamento de tela, entre outros. Este é um contexto com o chat e/ou conferência integrados ao ambiente de programação, mas ferramentas externas com o propósito de comunicação podem ser utilizadas para que esse processo aconteça.

Notificações do sistema

Além de comunicações entre pessoas, é possível que exista também comunicações por mensagens geradas pelos sistemas. Mensagens de **notificações** são úteis para diferentes funcionalidades e podem coordenar o usuário durante a utilização do sistema. Em um sistema com usuários autenticados, grupos de usuários e permissões de acessos diferentes, eventos podem ser notificados, como a solicitação para participar de um grupo ou a entrada de um novo

participante no ambiente. Essas solicitações e suas respostas são adequados para o modelo de comunicação assíncrono.

As alterações de artefatos de forma simultânea, e por usuários distintos, precisam ser coordenadas e auditadas para garantir a integridade dos dados. No cenário de conflito de versões, cabe aos usuários comparar as versões dos artefatos para resolvê-los. Notificações podem ser disparadas para informar os usuários sobre o conflito de versão que precisa ser resolvido para mesclar as versões conflitantes.

Mensagens de sincronização do sistema

Além das notificações, é possível haver outro tipo de mensagem para sincronizar tarefas compartilhadas. Estas mensagens permitem, por exemplo, que dois usuários distantes em suas geolocalização trabalhem de maneira síncrona no mesmo artefato de um projeto.

Histórico de comunicação

O histórico de comunicação do projeto pode ser armazenado em um log, mas traz a necessidade de avaliar a volumetria dessas informações. Para evitar problemas de limite de armazenamento, o log pode ter a configuração de expurgo que apaga os logs antigos considerando o tempo definido para isso. A opção de expurgo pode ser opcional, permitindo escolher manter todo o histórico. O log também permitirá que os usuários que estavam desconectados, ao se reconectarem, atualizem de forma assíncrona as mensagens que foram trocadas no canal de comunicação durante o período em que estiveram offline.

Localização da comunicação

A comunicação em WAN (Wide Area Network) e LAN (Local Area Network) apresenta diferentes abordagens para troca de mensagens. Na WAN, que abrange áreas geográficas extensas, várias arquiteturas e modelos de servidores podem ser implementados. No contexto deste projeto, uma opção é a utilização de um servidor de relay em WAN, centralizando a comunicação e coordenando a troca de mensagens entre os participantes. O relay permite que a comunicação ocorra de maneira eficiente e segura.

Por outro lado, em uma LAN, que cobre áreas mais restritas, como um espaço físico ou um edifício, um modelo sem servidor centralizado pode ser implementado. Nesse cenário, as mensagens podem ser disseminadas por meio de mensagens broadcast (enviadas a todos) ou multicast (enviadas a grupos específicos) dentro da rede local. Cada abordagem tem suas vantagens e considerações específicas em relação à escala e à eficiência da comunicação.

Mensagens síncronas e assíncronas

Dentro de um ambiente colaborativo virtual pode-se imaginar uma rede de computadores com diferentes grupos de usuários trabalhando em um mesmo workspace de forma síncrona e assíncrona.

Na cooperação síncrona os participantes mantêm-se sincronizados trabalhando em conjunto e em tempo real. As alterações na **workarea** são instantaneamente refletidas para todos, promovendo uma colaboração fluida e imediata. Caso a alteração envolva a adição de um artefatos que nem todos os usuários do grupo possuem, esses artefatos são compartilhados de maneira assíncrona. Além do compartilhamento de artefatos, mensagens nos canais de comunicações e notificações também são realizados de maneira assíncrona de forma que um usuário, ao se conectar, possa ser atualizado sobre as modificações que aconteceram enquanto ele não estava conectado.

Granularidade da comunicação

A persistência de notificações entre os usuários pode ser configurada em diferentes granularidades, notificando todos os usuários da plataforma ("hall"), grupo específico ou todos os participantes do workspace ou do projeto.

4.7 Artefatos, funcionalidades e o modelo 3C

Ao longo desta seção é discutido sobre os artefatos resultados da feature de colaboração, suas funcionalidades e a relação com o modelo 3C. A Tabela 5 lista esses artefatos, relacionando o papel de suas funcionalidades aos três pilares do modelo 3C.

Artefato	**Cooperação**	**Coordenação**	**Comunicação**
Usuário		x	
Workspace	x	x	x
Workarea	x		
Projeto	x	x	x
Issue	x	x	x
Sugestão	x		x
Comentário	x		x
Mensagens de usuário	x		x
Mensagens de sincronização		x	x
Notificações		x	x
Ferramentas	x	x	x
Plugins	x	x	x
Extensões	x		
Documentação	x		x

Tabela 5 – Artefatos mapeados e suas relações com o modelo 3C.

Os **usuários** possuem um identificador único, dados de autenticação e outras informações relevantes. São coordenados pelo sistema de usuários para garantir segurança, integridade dos dados e devidas permissões de acessos. Ao realizar o login na plataforma, o usuário pode conectar-se ao ambiente colaborativo chamado de workspace. O workspace dispõe de diferentes recursos para cooperar, comunicar e ser coordenado. Também é possível criar novos workspaces ou solicitar a participação em outros.

Dentro do **workspace** o usuário pode criar ou conectar em **projetos**. No escopo do projeto ocorre cooperação, coordenação e comunicação. A **workarea** é o espaço virtual dentro do projeto onde ocorre a colaboração síncrona no processo de desenvolvimento de software. Nesse espaço os usuários programam, resolvem conflitos de códigos comparando as suas versões, adicionam comentários e sugestões.

As **issues** dispõe de recursos e espaço para discutir sobre bugs e refatoração de features ou criação de novas, no escopo do projeto. Essa comunicação é uma maneira de cooperar e que também envolve coordenação do projeto e suporte aos usuários. As issues podem ter comentários, responsáveis, status e labels de categorizações.

Sugestões e **comentários** podem ser utilizados para comunicar e cooperar em diferentes granularidades, no escopo do projeto ou do workspace. A discussão que acontece na issue e é realizada utilizando **mensagens de usuário**. **Notificação** é outro artefato de comunicação que pode ser coordenativo ou não, e utilizado no processo de colaboração.

A cooperação síncrona demanda trocas de mensagens síncronas. Para evitar sobrescritas, artefatos podem ser bloqueados e liberados para alteração. As ações de bloquear e de liberar são **mensagens de sincronização** que acontecem na comunicação síncrona.

Ferramentas são softwares externos que podem ser utilizados em paralelo e terem a função de comunicação, coordenação e cooperação. Podem ser ferramentas de vídeo/áudio conferência, de gerenciamento de projetos e também de cooperação em tarefas específicas necessárias para a execução do projeto, mas podem ser realizadas isoladas ou integradas com o sistema via API. Tarefas isoladas podem ter a função de obter informações necessárias para um desenvolvimento do software produto da colaboração.

Plugins permitem adicionar novas features para a ferramenta de desenvolvimento de software. Essas features alteram o sistema em toda a sua granularidade, adicionando novos artefatos que trazem funcionalidades, e também modificar os já existentes. As funcionalidades podem ser de comunicação, coordenação ou cooperação.

Extensões permitem adicionar novos recursos de programação que podem ser utilizados para construir o produto do trabalho colaborativo ou trabalho individual. Os recursos são as funcionalidades da linguagem de programação utilizada para desenvolver. Em uma VPL que a programação é realizada escolhendo, adicionando e conectando blocos, os blocos são os recursos. Eles podem ter uma entrada dados que podem ser processados e ter as saídas que são dados

resultantes do processamento.

Documentações arquivos que contém conhecimento sobre o processos, regras de negócios, projetos, tecnologias, ferramentas e outros artefatos. Disponibilizar uma documentação é cooperar com a comunidade compartilhando conhecimento. O compartilhamento é feito de maneira assíncrona, trocando conhecimento sem exigir a interação direta entre os usuários.

Plugins/extensões da ferramenta em questão ou ferramentas externas foram considerados dependências do workspace, para que os participantes tenham os mesmos recursos e artefatos envolvidos nos projetos.

5 PROPOSTA DE CSCW NO MOSAICODE

Os artefatos e funcionalidades propostos no Capítulo anterior serviram de base para a discussão do presente Capítulo. O Mosaicode é utilizado como objeto de estudo de casos para aplicação da proposta, apresentando possibilidades de integração de tais funcionalidades a esta ferramenta. A Tabela 6 lista os artefatos que já existem no Mosaicode e também os que precisam ser desenvolvidos para implementar a proposta de colaboração deste trabalho.

Artefato	**Desenvolver**	**Reaproveitar**
Usuário	x	x
Workspace	x	x
Workarea	x	x
Projeto		x
Issue		x
Sugestão	x	x
Comentário	x	x
Mensagens de usuário	x	x
Mensagens de sincronização	x	
Notificações	x	x
Ferramentas	–	–
Plugins/Extensões	–	–
Documentações		x

Tabela 6 – Artefatos mapeados e sua relação com a necessidade de desenvolver e o que já existe e pode ser reaproveitado (x). Também o que já existe implementado na ferramenta ou externamente (–).

5.1 Gestão de usuário e projeto

Após a fase de planejamento de um projeto, com os requisitos definidos e stakeholders identificados, o projeto passa para a parte de execução. Tecnologias podem ser utilizadas para auxiliar o monitoramento e gerenciamento das tarefas durante a fase de execução. Já existem ferramentas FLOSS para este propósito de gerenciamento de projetos, entre elas: Open Project [1], Wekan [2], Kanboard [3], Taiga.io [4] e Redmine [5].

As ferramentas disponíveis apresentam diferentes visualizações para acompanhar o projeto. É possível acompanhar e controlar o cronograma do projeto utilizando o Gráfico de Gantt, visualizando o andamento das tarefas, as dependências de cada uma e os responsáveis, permitindo ter uma previsão de início e término das atividades e também identificar pontos

[1] https://www.openproject.org/
[2] https://wekan.github.io/
[3] https://kanboard.org/
[4] https://taiga.io/
[5] https://www.redmine.org/

críticos. O cronograma pode ser comparado com o calendário do projeto, para avaliar se o cronograma está cumprindo com as datas definidas.

Os stakeholders podem visualizar a lista de tarefas de todos os projetos ou aplicar o filtro para exibir apenas as tarefas as quais foi atribuído como responsável. A lista de tarefas são comumente visualizadas no board Kanban, onde cada atividade está na coluna referente a etapa de desenvolvimento que ela se encontra.

Também é possível gerar e visualizar relatórios com informações de desempenho do projeto, se está cumprindo o planejamento, qual a quantidade de bugs, entre outras. Essas informações podem ser utilizadas para identificar pontos de melhorias e de atenção no processo de desenvolvimento, mantendo o controle do projeto.

A implementação da gestão de usuário pode reaproveitar as funcionalidades do framework Flask [6]. Esse framework oferece recursos para o sistema de autenticação de usuário, com features de usuário/conta, grupo de papéis e suas permissões, login/logout, sessão e suporte a Sistemas Gerenciadores de Banco de Dados (SGBDs) para armazenamento dos dados, incluindo: Drizzle, Firebird, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, SQLite e Sybase. Também oferece features logging e debugging, com a possibilidade de integração com a ferramenta Sentry [7] ou outras ferramentas de logging com recursos de tracking de erros e monitoramento.

5.2 Documentações

O Mosaicode já contém diferentes documentações. É possível gerar a documentação em vários formatos, a partir de comentários escritos do código-fonte. A geração da documentação é realizada com a ferramenta Sphinx [8], que suporta comentários na linguagem de marcação *reStructuredText* e *MyST markdown*. Os novos artefatos precisam de comentários no código para também terem documentações geradas. Além dos comentários, é necessário garantir que o Sphinx considere as pastas desses artefatos para percorrê-las lendo os arquivos e seus comentários para gerar a documentação.

O repositório remoto do Mosaicode no GitHub contém arquivos markdown (README.md e página Wiki) com informações, instruções de uso e instalação, dependências, conceitos e tutoriais. Outros documentos são artigos científicos sobre o Mosaicode publicados em anais de conferências e em periódicos, todos disponibilizados na página do laboratório ALICE.

As documentações aqui são mais voltadas para desenvolvedores da ferramenta Mosaicode e suas extensões e plugins. As documentações do projeto implementado pelos citizen developers podem ser disponibilidades em ferramentas externas. As ferramentas de gestão de projetos

[6] https://pypi.org/project/Flask/
[7] https://github.com/getsentry/sentry
[8] https://www.sphinx-doc.org/en/master/

permitem anexar os documentos ou adicionar links para os mesmos. Os documentos também podem ser trocados em canais de comunicação de ferramentas externas.

5.3 Plugins e Extensões

Como o Mosaicode pode ser incrementado utilizando plugins e extensões, ou simplesmente pelo fato de ser FLOSS, é possível adaptar a ferramenta desenvolvendo novas features. As extensões definem linguagens de programação visual com recursos específicos para serem utilizados no desenvolvimento do projeto implementado pelos usuários finais.

Os plugins permitem adicionar novas funcionalidades à ferramenta. Existe um plugin desenvolvido para o Mosaicode chamado *Extensions Manager* [9], esse plugin permite criar, gerenciar/alterar os artefatos de uma extensão ou criar novas. A utilização do plugin consiste em um desenvolvimento Low-Code, apoiando os citizen developers na criação e alteração de recursos para serem utilizados na implementação dos projetos.

O Mosaicode carrega as extensões e os plugins instalados no momento da sua inicialização, que podem estar instalados no espaço de usuário ou de sistema.

5.4 Comentário

O Mosaicode contém comentários que podem ser adicionados no diagrama de um projeto, porém, eles podem ser evoluídos permitindo ter marcas de revisões ou sugestões.

5.5 Workspace

A cooperação requer um ambiente propício para sua efetivação, permitindo criar artefatos e manipular os existentes. Esse ambiente colaborativo é definido aqui como workspace, artefato resultante da feature de colaboração. No Mosaicode, esse workspace se destina à elaboração de projetos que englobam diagramas que são implementações de softwares associados a cada projeto.

Dentro desse workspace, é possível estabelecer comunicações por meio de notificações visuais que informam sobre mudanças e eventos durante tanto processos de colaboração síncrona quanto na sincronização de desenvolvimento assíncrono. Tais notificações têm a finalidade de prover informações coordenativas. Ações coordenativas desempenham um papel essencial no funcionamento do espaço de trabalho, assegurando a aderência às regras, limitando o acesso somente a informações autorizadas e oferecendo suporte ao desenvolvimento permitindo o controle de versões e a resolução de conflitos.

[9] https://github.com/Alice-ArtsLab/mosaicode/tree/main/mosaicode/plugins

O workspace pode apresentar diferentes modos de utilização, permitindo apenas leitura ou edição. O modo leitura remove a preocupação do usuário de acidentalmente realizar alterações na implementação. As manipulações permitidas incluem sugestões/comentários e lista de issues. Outra possibilidade é pode realizar uma alteração no desenvolvimento sem persistir na solução, sendo apenas uma sugestão que é uma versão da implementação que pode ser visualizada e testada pelos usuários permitidos. Essa versão é como uma branch da ferramenta git, ramificação do código com uma linha de desenvolvimento independente e que pode ser mesclada a outras branchs, seja a principal ou não.

O modo de edição permite alterar a versão principal do código diretamente. As sugestões, comentários e lista de issues podem ser exibidas ou ocultadas nesse modo também, sendo uma forma dessas informações sem trocar de modo. O conceito de branch também pode ser aplicado aqui, sendo não apenas uma sugestão, mas um desenvolvimento paralelo que será mesclado.

Para a quantidade de desenvolvimento dessa proposta, foi escolhido não desenvolver as issues mas utilizar uma ferramenta externa para isso. Seja a ferramenta de gestão de projetos, GitHub ou outra ferramenta/plataforma.

5.6 Workarea

A colaboração pode ocorrer de forma assíncrona, onde cada usuário altera uma versão de um artefato, gerando versões diferentes que consecutivamente são mescladas. A ferramenta *git* provê um controle de versão com recursos de log auditável, de acessar, de comparar e de mesclar versões.

O mesmo acontece com as alterações de outros artefatos manipulados no processo de colaboração síncrono. Imaginando a workarea do Mosaicode, espaço onde os usuário programam na linguagem de programação visual, alterando o diagrama de blocos e conexões. Ao conectar no workspace, deparando com uma versão diferente da versão local, seria útil mensagens assíncronas entre os usuários para notificar essa atualização e apresentar informações relevantes para cada caso. Assim cada usuário pode decidir obter ou não as alterações. No cenário de conflito, surge a necessidade de resolvê-los de forma manual, escolhendo a versão do trecho em questão.

O git pode ser utilizado para fazer esse controle, mas é necessário implementar os componentes de interface necessários para exibir essas versões de forma apropriada para os citizen developers. Caso o usuário queira visualizar o *diff* a nível de código-fonte dos artefatos, pode ser possível também, mas o principal seria essa camada sobe o git.

5.6.1 Artefatos do Mosaicode

Os artefatos precisam ter uma estrutura bem definida, de forma que o Mosaicode consiga reconhecê-los e carregá-los. Também é importante para a comparação de versão, empacotamento

e distribuição.

O Anexo B apresenta um exemplo de estrutura de pastas, seguindo o padrão atual implementado na ferramenta Mosaicode.

5.7 Cooperação e comunicação em ambiente compartilhado

A cooperação depende da comunicação, exigindo canais de comunicação para envio de mensagens de usuários. Diferentes tecnologias podem ser utilizadas para reutilizar funcionalidades já implementadas.

É possível utilizar API's para integrar soluções ou utilizar ferramentas em paralelo. O Etherpad [10] é um editor de texto colaborativo em tempo real que pode ser utilizado isolado ou integrado por meio da API HTTP disponível para esse propósito. Da mesma forma o Jitsi Meet [11], que pode ser utilizado para contemplar o recurso de vídeo conferência, também com chat, de forma integrada via API ou isolado. Vale destacar que ambas ferramentas são software livres.

Além das ferramentas externas, as mensagens de rede para comunicação síncrona definidas na Subseção 5.7.2 incluem a mensagem *Chat* que permite que membros do chat troquem mensagens privadas e públicas. Essa é a parte a ser desenvolvida para integrar a comunicação no Mosaicode.

5.7.1 Servidor de Relay

No worksparce colaborativo grupos de usuários podem manipular e editar artefatos, gerando um tráfego de mensagens pela rede para atualizar as alterações realizadas no ambiente. As mensagem definidas para a comunicação síncrona não apresentam o parâmetro para informar qual diagrama da aplicação pertence ao grupo informado na mensagem. Foi proposto uma tabela de repasse no lado da aplicação para resolver essa questão, listando o identificador do grupo e do diagrama correspondente (colunas: identificador do grupo; identificador do diagrama). Ao receber uma mensagem, a aplicação consulta na tabela de repasse o diagrama correspondente ao grupo informado.

Também é necessário implementar uma tabela de repasse na aplicação do servidor de relay para listar os grupos, o identificador dos membros de cada grupo, tipo dos membros (administrador ou não) e o tempo da sessão de cada membro. Essa tabela permitirá que as mensagens enviadas, de um cliente para o servidor, sejam encaminhadas para membros de um grupo específico – também para apenas um membro do grupo, no caso da mensagem Send Message, configurando os parâmetros.

[10] https://etherpad.org/
[11] https://jitsi.org/

A Figura 9 apresenta o modelo/arquitetura de comunicação, ilustrando o fluxo de mensagens. Quando o usuário (Host A) utiliza o ambiente de comunicação, essa ação pode resultar no envio de mensagem para o servidor. Quando a mensagem for do tipo Create Collaboration é adicionado o registro na tabela de repasse para que as mensagens trocadas possam ser direcionadas ao destino correto.

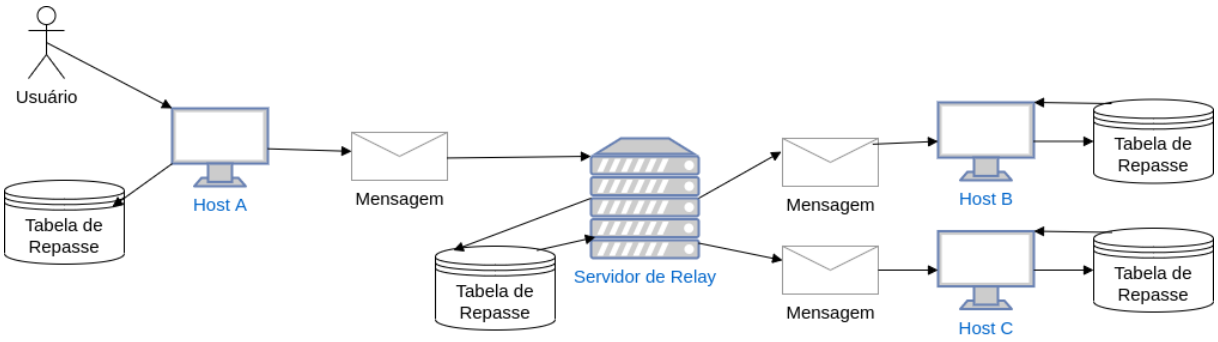

Figura 9 – Modelo e fluxo de comunicação.

Ao receber a mensagem, o servidor de relay consulta a tabela de repasse para identificar o destinatário da mensagem – podendo ser um membro de um grupo ou todos os membros desse grupo. No exemplo da Figura 9, a mensagem é replicada para todos os membros do grupo (Host B e C). A aplicação do Host B e do Host C, ao receber a mensagem, consulta a tabela de repasse para identificar o diagrama que deve receber a alteração. Em seguida, a alteração é realizada.

5.7.2 Mensagens de rede para comunicação síncrona

O compartilhamento de artefatos, cooperação em ambiente compartilhado e canais de comunicação, demandam trocas de mensagens em rede de computadores. São tipos diferentes de mensagens e conteúdo, entretanto, é possível definir uma estrutura básica genérica que permite encapsular essa variedade de mensagens existentes.

Para esse encapsulamento, utilizamos a estrutura básica de mensagens em rede definidas por Su e Han (SU; HAN, 2018). Essa estrutura é composta por três tipo de mensagens:

- Mensagem *Envelope*: encapsula as mensagens Header e Body;
- Mensagem *Header*: Define um cabeçalho com informações que descreve os dados dessa mensagem, como informações de propriedade da mensagem body;
- Mensagem *Body*: contém a mensagem (conteúdo) a ser entregue ao usuário – artefatos e outros tipos de dados/mensagens.

As mensagens de redes serão encapsuladas, contendo um cabeçalho padrão que inclui a mensagem para o host, além de outros parâmetros. A Tabela 9 apresenta os parâmetros desse cabeçalho, com descrição. E a Tabela 10 associa as mensagens síncronas ao modelo 3C.

Parâmetros	Descrição
To	Endereço do host que recebera a mensagem
From	Endereço do host que enviará a mensagem
Message ID	Identificador da mensagem
Timestamp	Define o timestamp da mensagem
Encoding	Informa a codificação de caracteres da mensagem body
Encrypted	Criptografia das mensagens header e body
Message	Mensagem header

Tabela 7 – Os parâmetros principais e descrição da mensagem envelope.

Parâmetros	Descrição
Sender	Identificador do remetente
Receiver	Identificador do destinatário
Type	Tipo de mensagem
Performative	Tipo de comportamento de comunicação
Ontology	Ontologia do corpo da mensagem
Language	Linguagem da expressão do corpo da mensagem
Content	Mensagem body

Tabela 8 – Os parâmetros principais e descrição da mensagem header.

Parâmetros	Descrição
Version	Versão do protocolo
Message Type	Tipo da mensagem
To type	Se a mensagem é para o servidor, usuário, grupo ou para todos
To	Endereço do host que recebera a mensagem
From	Endereço do host que enviará a mensagem
Message ID	Identificador da mensagem
Timestamp	Define o timestamp da mensagem
Encoding	Informa a codificação de caracteres da mensagem body
Encrypted	Criptografia das mensagens header e body
Compress	Se a mensagem é comprimida e qual o formato
Message	Mensagem em si

Tabela 9 – Padrão de cabeçalho para todas as mensagens

Hello Group

O usuário envia essa mensagem para o servidor, após receber a mensagem Replay Collaboration. O servidor encaminha a mensagem para os membros do grupo, informando a presença do novo usuário. A Tabela 11 contém a lista de parâmetros e descrição de cada um.

Heartbeat

Essa mensagem tem o propósito de informar ao servidor que o usuário continua conectado. Após entrar em um grupo, o usuário terá um tempo para enviar uma mensagem do tipo Heartbeat. Se o servidor não receber essa mensagem até o tempo pré-determinado, o usuário

Mensagem	Cooperação	Coordenação	Comunicação
Hello Group			x
Heartbeat		x	x
Goodbye Group			x
Pedido e resposta de colaboração		x	x
Início da colaboração			x
Permissão do grupo		x	
Fim da colaboração			x
Remover Membro			x
Adicionar Bloco	x		
Lock / Unlock de Artefatos		x	x
Modificar propriedades de blocos	x		
Remover Blocos	x		
Adicionar Conexão	x		
Remover conexão		x	
Chat			x

Tabela 10 – Mensagens síncronas propostas para o ambiente e suas relações com o modelo 3C

Parâmetros	**Descrição**
Message Type:	Hello Group
To type:	Identificador do grupo
Message:	Identificador do membro

Tabela 11 – Parâmetros da mensagem Hello Group

será desconectado do grupo.

O servidor terá uma thread ouvindo a mensagem de heartbeat, evitando que um membro permaneça cadastrado no grupo quando for desconectado/“cair"sem enviar a mensagem do tipo Goodbye – comum de acontecer devido problemas de conexão e bugs. A Tabela 12 contém a lista de parâmetros e descrição de cada um.

Parâmetros	**Descrição**
Message Type:	Heartbeat
To type:	Identificador do grupo
Message:	Identificador do membro

Tabela 12 – Parâmetros da mensagem Heartbeat

Goodbye Group

Permite o usuário sair de um grupo específico. Ao enviar essa mensagem para o servidor, a mesma é encaminhada para os membros do grupo, persistindo a informação. A Tabela 13 contém a lista de parâmetros e descrição de cada um.

Parâmetros	Descrição
Message Type:	Goodbye Group
To type:	Identificador do grupo
Message:	Identificador do membro

Tabela 13 – Parâmetros da mensagem Goodbye Group

Pedido e resposta de colaboração

Para participar de um grupo, o usuário envia a mensagem Request Collaboration para o servidor. A mensagem é encaminhada para os membros do grupo e o remetente fica esperando a mensagem de aceitação (Reply Connection), por um tempo pré-determinado. A Tabela 14 contém a lista de parâmetros e descrição de cada um.

Parâmetros	Descrição
Message Type:	Request Collaboration
To type:	Identificador do grupo
Message:	Identificador do usuário

Tabela 14 – Parâmetros da mensagem Request Collaboration

Ao receber um pedido de colaboração (Request Collaboration), o usuário é notificado. Essa notificação é exibida na interface gráfica da aplicação, com opção de aceitar ou recusar o pedido. Ao escolher uma das opções, é enviado a mensagem Reply Collaboration para o servidor, que é encaminhado para o usuário que realizou o pedido – informando se foi aceito ou não. Apenas administradores do grupo podem responder o pedido. A Tabela 15 contém a lista de parâmetros e descrição de cada um.

Parâmetros	Descrição
Message Type:	Reply Collaboration
To type:	Identificador do grupo
Message:	Aceitação ou rejeição

Tabela 15 – Parâmetros da mensagem Reply Collaboration

Início de colaboração

Para criar um ambiente compartilhado, o usuário deve solicitar a criação de um grupo. Com essa solicitação, é enviado para o servidor a mensagem Create Collaboration. O grupo é registrado no servidor, permitindo que outros usuário envie a solicitação para entrar no grupo (Request Collaboration). O administrador do grupo é o usuário que compartilhou o ambiente, também considerado como dono – impedindo que seja removido da lista de administradores. A Tabela 16 contém a lista de parâmetros e descrição de cada um.

Parâmetros	Descrição
Message Type:	Create Collaboration
To type:	Identificador do grupo
Message:	Identificador do usuário

Tabela 16 – Parâmetros da mensagem Create Collaboration

Permissão do grupo

Essa mensagem permite definir se o usuário é administrador do grupo ou não. Na lista de membros do grupo, exibida na interface gráfica da aplicação, terá, em cada membro da lista, a opção para definir a permissão. Essa opção aparece apenas para os membros administradores. A Tabela 17 contém a lista de parâmetros e descrição de cada um.

Parâmetros	Descrição
Message Type:	Group Permission
To type:	Identificador do grupo
User ID:	Identificador do usuário
Message:	Administrador ou não

Tabela 17 – Parâmetros da mensagem Group Permission

Fim de colaboração

Administradores do grupo podem finalizar a colaboração, enviando a mensagem Destroy Collaboration para o servidor. Ao receber essa mensagem, o ambiente compartilhado deixa de existir, implicando na exclusão do grupo e a persistência deste evento no servidor e para os membros do grupo. O servidor envia a mensagem Destroy para todos os membros do grupo, informado o fim da colaboração. A Tabela 18 contém a lista de parâmetros e descrição de cada um.

Parâmetros	Descrição
Message Type:	Destroy Collaboration
To type:	Identificador do grupo
Message:	Identificador do usuário

Tabela 18 – Parâmetros da mensagem Destroy Collaboration

Remover membro

Administrador de um grupo pode remover membros pertencentes ao mesmo. Para isso, é enviado a mensagem Remove Member para o servidor, resultando na remoção do membro. O usuário removido, recebe a replicação desta mensagem como informação do evento. A Tabela 19 contém a lista de parâmetros e descrição de cada um.

Parâmetros	Descrição
Message Type:	Remove Member
To type:	Identificador do grupo
Message:	Identificador do usuário

Tabela 19 – Parâmetros da mensagem Remove Member

Adicionar bloco

Quando um usuário adiciona um bloco no ambiente compartilhado, a mensagem Add Block é enviada para o servidor. Essa mensagem é encaminhada para todos os membros do grupo. A mensagem contém informações sobre o bloco, como: identificador e checksum do bloco, sua posição no diagrama, e valores das propriedades estáticas e dinâmicas. Essas informações são encapsuladas em uma estrutura de dados. A Tabela 20 contém a lista de parâmetros e descrição de cada um.

Parâmetros	Descrição
Message Type:	Add Block
To type:	Identificador do grupo
Message:	Informações do bloco.

Tabela 20 – Parâmetros da mensagem Add Block

Lock / Unlock de Artefatos

Ao editar o diagrama, alguns artefatos devem ser bloqueados para evitar problemas de concorrência. A mensagem Lock Artifact informa o bloqueio de um bloco ou uma conexão, permitindo que apenas o remetente edite o artefato bloqueado. Na interface gráfica do usuário (que receberam a mensagem encaminhada), é indicado de forma gráfica a parte do diagrama que foi bloqueado. A Tabela 21 contém a lista de parâmetros e descrição de cada um.

Após a persistência do bloqueio, a mensagem com a alteração do diagrama pode ser enviada para o servidor.

Parâmetros	Descrição
Message Type:	Lock Artifact
To type:	Identificador do grupo
Artifact type:	Bloco ou Conexão
Message:	Identificador do artefato e do usuário

Tabela 21 – Parâmetros da mensagem Lock Artifact

Após a persistência da alteração, a mensagem Unlock Artifact é enviada para o servidor e encaminhada para os demais membros, realizando o desbloquear do artefato do artefato informado. A Tabela 22 contém a lista de parâmetros e descrição de cada um.

Parâmetros	**Descrição**
Message Type:	Unlock Artifact
To type:	Identificador do grupo
Artifact type:	Bloco ou Conexão
Message:	Identificador do artefato

Tabela 22 – Parâmetros da mensagem Unlock Artifact

Modificar propriedades de blocos

Para enviar a alteração das propriedades dos blocos, é utilizado a mensagem Update Property. Essa mensagem é enviada para o servidor e encaminhada para os demais membros do grupo. A Tabela 23 contém a lista de parâmetros e descrição de cada um.

Parâmetros	**Descrição**
Message Type:	Update Property
To type:	Identificador do grupo
Block ID:	Identificador do bloco
Property ID:	Identificador da propriedade
Message:	Valor da propriedade

Tabela 23 – Parâmetros da mensagem Update Property

Remover Blocos

Para remover um bloco do diagrama, é utilizado a mensagem Remove Block. Essa mensagem é enviada para o servidor e encaminhada para os demais membros do grupo. A Tabela 24 contém a lista de parâmetros e descrição de cada um.

Parâmetros	**Descrição**
Message Type:	Remove Block
To type:	Identificador do grupo
Message:	Identificador do bloco

Tabela 24 – Parâmetros da mensagem Remove Block

Adicionar conexão

A mensagem Add Connection é utilizada para adicionar uma conexão. Essa mensagem é enviada para o servidor e encaminhada para os demais membros do grupo. A Tabela 25 contém a lista de parâmetros e descrição de cada um.

A conexão envolve dois blocos e suas portas (saída/entrada). O parâmetro Message encapsula o identificador da conexão, e dos blocos e portas envolvidos.

Parâmetros	**Descrição**
Message Type:	Add Connection
To type:	Identificador do grupo
Message:	Informações da conexão

Tabela 25 – Parâmetros da mensagem Add Connection

Remover Conexão

A mensagem Remove Connection é utilizada para remover uma conexão. Essa mensagem é enviada para o servidor e encaminhada para os demais membros do grupo. A Tabela 26 contém a lista de parâmetros e descrição de cada um.

A conexão envolve dois blocos e suas portas (saída/entrada). O parâmetro Message encapsula o identificador da conexão, e dos blocos e portas envolvidos.

Parâmetros	**Descrição**
Message Type:	Remove Connection
To type:	Identificador do grupo
Message:	Informações da conexão

Tabela 26 – Parâmetros da mensagem Remove Connection

Chat

Membros de um grupo podem conversar atrás de um chat. A mensagem Send Message permite comunicar com todos os membros do grupo (mensagem pública) ou com outro membro apenas (mensagem privada). Ao receber essa mensagem, o servidor encaminha a mensagem para o grupo ou membro específico. A Tabela 27 contém a lista de parâmetros e descrição de cada um.

Parâmetros	**Descrição**
Message Type:	Send Message
To type:	Identificador do grupo
User ID:	Null (chat público) ou identificador do usuário (mensagem privada)
Message:	Mensagem

Tabela 27 – Parâmetros da mensagem Send Message

6 RESULTADOS E DISCUSSÕES

Nesta seção, é apresentado e discutido os resultados obtidos a partir do estudo descritivo-analítico realizado sob a perspectiva dos fundamentos de Computer-Supported Cooperative Work (CSCW) e de plataformas Groupware e Low-Code. Através dessa análise, foram identificados requisitos e features que apoiam e permitem citzen developers colaborarem durante o processo de criação de software. O trabalho foca na criação de arte digital, mas pode ser aplicado em outros domínios.

A proposta de implementação apresenta a discussão sobre artefatos, arquitetura de comunicação e comportamento, funcionalidades e ferramentas de apoio. O Mosaicode é utilizado como objeto de estudo de caso, o qual permitiu listar os artefatos do Mosaicode a serem compartilhados e artefatos necessários para a feature de colaboração. A partir dos artefatos foi possível definir um estrutura de pastas para serem armazenados e manipulados por um gerenciados de artefatos. Também foi definido um protocolo para mensagens de rede para comunicação síncrona.

É crime comunicações que deprecie um usuário ou grupo pelas suas características. Os discursos de ódio trazem a necessidade do controle da comunidade. Dessa forma, é necessário o cadastramento do usuário com dados pessoais e de autenticação. A identificação do usuário traz a transparência de conhecer os membros do workspace, auditando as alterações de artefatos durante o processo de colaboração.

Assim é possível ter a visão de quem fez cada alteração, conhecendo os usuários que estão comunicando ou os autores de artefatos. Também como coordenação, a Lei Geral de Proteção de Dados Pessoais (LGPD) assegura os direitos fundamentais de liberdade e de privacidade do usuário (BARROS; FILHO; MACHADO, 2023). O Processamento de Linguagem Natural (PLN) contribui para a identificação de discurso de ódio (TRAJANO, 2023). A implementação de features com funcionalidades de reportar esses tipos de comportamentos é uma forma de cooperar e coordenar, através da comunicação.

6.1 Gerenciando artefatos

Ferramentas colaborativas podem ser capazes de gerar mais capacidade dinâmica de comunicação e oferecer melhor experiência nesse modelo remoto ou até mesmo no modelo tradicional, permitindo maior controle e cópias idênticas dos artefatos digitais, de forma “fácil e simples”, incluindo a sua distribuição.

Trabalhar sobre os artefatos, ter claramente quais são e como são, como as pessoas podem colaborar trocando arquivos. Permitir trabalhar com fuso horário diferente, locais diferentes. A comunicação assíncrona não é importante apenas pela diferença de comunicação, mas também por um modelo de trabalho que muitas vezes pode ser relacionado ao tipo de trabalho de

desenvolvimento de software e de criação de arte. As pessoas podem colaborar cada um no seu tempo, em cima de artefatos, então esses artefatos precisam estar disponíveis para que as pessoas possam colaborar.

Os Sistemas Operacionais (SO) Linux são baseados em um padrão de hierarquia de sistema de arquivos (Filesystem Hierarchy Standard [1]), que apresenta uma estrutura de arquivo clara e bem definida e atribui responsabilidades para os diretórios da sua hierarquia – ext4, JFS, Unix File System e ZFS são exemplos de sistemas de arquivos que seguem esse padrão. Diretórios podem ser para uso do usuário e/ou do sistema, também para armazenar diferentes informações como: arquivos de configurações (/etc), arquivos estáticos com informações necessárias para inicializar o SO (/boot), bibliotecas de softwares compartilhadas (/lib), mídias removíveis (/media), arquivos binários do usuário (/bin) e do sistema (/sbin), temporários (/tmp), manuais (/man) e logs (/log).

Essa estrutura de arquivo pode ser bem definida na criação de qualquer sistema, inclusive o Mosaicode. Não organizar os diretórios dos artefatos dificultaria gerenciar o compartilhamento dos mesmos, implicando numa maior complexidade de implementação e de tempo de execução para recuperar e compartilhar os artefatos. Pode ser feito uma analogia à estrutura de banco de dados relacional onde a definição dos dados é de extrema importância para simplificar e otimizar a manipulação (DML) e a consulta (DQL) de dados.

Dessa forma, foi definido uma workspace que representa um ambiente de colaboração que permite compartilhar artefatos. Esse workspace contém a sua estrutura de arquivos definida de forma a organizar os artefatos e beneficiar dessa organização provendo clareza dos artefatos existentes, favorecendo o gerenciamento.

6.2 Colaboração assíncrona e síncrona

O controle de versão dos artefatos é necessário para garantir que os usuários estejam trabalhando com a mesma versão, não gerando incompatibilidades. Também para permitir comparar as versões e voltar a pontos de modificações anteriores, além de realizar a auditoria salvando informações de quando, o que, por quem foi modificado. Os desenvolvedores já tem esse recurso disponível pela ferramenta de controle de versão *git*, porém é uma ferramenta comumente complexa para citizen developers. O controle de versão deve ser oferecido de forma adequada para esses usuário finais, sem exigir habilidades e conhecimentos de programadores avançados, mas oferecendo as mesmas funcionalidades como: *commit*, *merge*, *pull*, *push* e *cherry-pick*.

O software livre oferece liberdades para o programador ler, modificar e utilizar o código-fonte para qualquer propósito. O programador pode desenvolver artefatos e ter o controle através

[1] Descrição do padrão de hierarquia dos sistemas de arquivos Linux: https://man7.org/linux/man-pages/man7/hier.7.html

do git, auditar e versionar, sem problemas de acesso e com direito autoral. Os citizen developers, que são usuários finais que produzem artefatos, não tem o suporte para controle, auditoria e versionamento do seu produto. Mesmo utilizando um software livre, não tem acesso ao código-fonte do artefato produzido, isso priva o usuário das liberdades do software livre e o impede de ter um controle ao nível de código-fonte e de poder estudar o código caso tenha interesse.

É importante que o usuário final tenha esses recursos em um ambiente Low Code. De preferência, não fazer com que usuário precise lidar com códigos. Para adequar à forma do usuário trabalhar, os recursos de controle de versão devem oferecer uma visualização/manipulação semelhante ao que ele já está acostumado. Em vez de visualizar versões do código-fonte, visualizar as versões do artefatos.

Apesar do suporte ao desenvolvimento colaborativo não ser obrigatório, essa feature é muito importante, não é vão que todas as ferramentas examinadas por Sahay suportam a colaboração (offline) – apenas duas entre elas não suportam a colaboração online. A necessidade de ferramentas que suportam o trabalho colaborativo online tem surgido cada vez mais, sendo muito presente neste momento de pandemia causada pela COVID-19. Sem esse suporte, os usuários terão os recursos para a realização do trabalho, apenas não sendo possível que o mesmo seja realizado de forma colaborativa. Essa feature será importante para os usuários finais do Mosaicode e para as pesquisas atuais no ALICE, continuando servindo como objeto de estudo, desta vez, em um projeto de mestrado.

Atualmente a ferramenta Mosaicode dispõe de recursos para colaborar/cooperar durante a criação de diagramas. Para essa cooperação, é necessário o compartilhamento do ambiente de trabalho da ferramenta. A alteração do diagrama, por algum dos usuários, demanda a troca de mensagem em rede, de forma síncrona.

Um sistema de usuários deve ser integrado ao Mosaicode, pensando em questões de segurança e permitindo a criação de sessões e grupos/salas de colaboração. Alterações no diagrama resultará na transmissão síncrona de mensagens para um usuário ou mais usuário membros de um mesmo grupo.

Para a colaboração e cooperação é importante oferecer formas de comunicação entre usuários finais, por meio de canais de comunicação. Canais de comunicação como chat permitiriam a troca de mensagens entre usuários, de forma assíncrona. A mensagem poderá ser pública (de um membro para todos os membros do grupo) ou privada (de um membro para apenas um membro). O chat foi escolhido como um canal de comunicação a ser implementado. Para os demais canais de comunicação foi escolhido o uso de ferramentas externas, também para o conceito de issue e gerenciamento de projeto.

7 CONCLUSÃO E TRABALHOS FUTUROS

Para atingir a proposta deste trabalho, foi realizada uma análise descritiva à luz dos fundamentos de LCDP, assim como áreas que lidam com o trabalho cooperativo auxiliado por computador. A colaboração e cooperação no ambiente de desenvolvimento da ferramenta Mosaicode foi considerada a principal feature para a proposta. O apoio à citizen developers foi outro propósito. Esses usuários também são desenvolvedores/programadores – mesmo sem experiência e sem o pensamento computacional e habilidades de programação presentes em profissionais de TI, como cientistas da computação (GARCÍA-PEÑALVO, 2018; GARCÍA-PEÑALVO; MENDES, 2018; BOBSIN et al., 2020).

Foi elencado features necessárias para permitir a colaboração e cooperação no ambiente de desenvolvimento do Mosaicode, identificando processos, ferramentas, modelos e outros recursos de apoio à este trabalho e para oferecer suporte aos usuários finais. Os requisitos foram elencados utilizado o diagrama de caso de uso para descrever descrever o cenário, expressando o diálogo entre os usuários finais e o sistema.

A proposta de evolução da ferramenta foi discutida, apontando requisitos atendidos, não atendidos, assim como forma de avaliação e refinamento da proposta. A feature de colaboração traz a necessidade de novos artefatos que contemplam a comunicação, coordenação e cooperação, além da necessidade de encapsular os artefatos já existentes de forma que possibilite o compartilhamento via redes de computadores.

Também foi apresentado um protocolo de comunicação, descrevendo as mensagens para a colaboração síncrona. A existência de artefatos e a possibilidade de expandir o Mosaicode, traz a necessidade de implementar primeiramente a comunicação assíncrona para distribuição de artefatos empacotados. O diagrama pode conter artefatos que não existem para alguns membros do grupo. O mecanismo de empacotamento e distribuição permitirá a obtenção destes artefatos.

Requisitos não funcionais como criptografia, encapsulamento e compactação de mensagens precisam ser definidos para prover segurança de informações e otimizar/apoiar a comunicação. Também é importante discutir possibilidades de conflitos, locks e transações ACID.

As mensagens de ação podem ser utilizadas para salvar o arquivo e permitir undo infinito e diff de versões. Outras interfaces para o Mosaicode precisam ser pensadas, para exibir o resultados do diff e resolver conflitos entres as versões. Além do modo de edição de diagramas, seria interessante oferecer um modo de sugestão e apenas visualização.

A continuidade da discussão e implementação da proposta aplicada no Mosaicode fica como trabalho futuro. Também é possível pensar em livecode individual e colaborativo com o Mosaicode. Outra possibilidade seria a colaboração dentro de um ambiente de realidade virtual ou aumentada, oferecendo maior imersão.

REFERÊNCIAS

ALMEIDA, M. A.; SCHIAVONI, F. L. Do código à colaboração: sustentabilidade na arte digital. In: **Simpósio Internacional em Artes, Urbanidades e Sustentabilidade**. São João del-Rei - MG - Brazil: [s.n.], 2017. p. 156–164.

ALMEIDA, M. A.; SCHIAVONI, F. L. Aspectos da sustentabilidade e colaboração na arte digital. **Revista Interdisciplinar Internacional de Artes Visuais-Art&Sensorium**, v. 5, n. 1, p. 01–14, 2018.

ALMONTE, L. et al. Towards automating the construction of recommender systems for low-code development platforms. In: **Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings**. New York, NY, USA: Association for Computing Machinery, 2020. (MODELS '20). ISBN 9781450381352. Disponível em: <https://doi.org/10.1145/3417990.3420200>.

BAIDOO-ANU, D.; ANSAH, L. O. Education in the era of generative artificial intelligence (ai): Understanding the potential benefits of chatgpt in promoting teaching and learning. **Available at SSRN 4337484**, 2023.

BARROS, P. V. da S.; FILHO, J. M. da S. M.; MACHADO, J. de C. Estratégias para modelagem e avaliação da conformidade entre sistemas de informação e a lei geral de proteção de dados pessoais-lgpd. **Sociedade Brasileira de Computação**, 2023.

BOBSIN, R. et al. O pensamento computacional presente na resolução de problemas investigativos de matemática na escola básica. In: **Anais do XXXI Simpósio Brasileiro de Informática na Educação**. Porto Alegre, RS, Brasil: SBC, 2020. p. 1473–1482. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbie/article/view/12903>.

BOURQUE, P.; FAIRLEY, R. E.; SOCIETY, I. C. **Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0**. 3rd. ed. Washington, DC, USA: IEEE Computer Society Press, 2014. ISBN 0769551661.

BRISCO, R.; WHITFIELD, R.; GRIERSON, H. A novel systematic method to evaluate computer-supported collaborative design technologies. **Research in Engineering Design**, Springer, v. 31, n. 1, p. 53–81, 2020.

CANCHÉ, M.; OCHOA, S. F. A survey of development strategies for collaborative systems. In: **2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)**. Porto – Portugal: IEEE, 2019. p. 261–266.

COELHO, N. M. Realidade virtual "estado da arte". **ISLA–Instituto Politécnico de Gestão e Tecnologia, Vila Real**, 2016.

COLLAZOS, C. A. et al. Descriptive theory of awareness for groupware development. **Journal of Ambient Intelligence and Humanized Computing**, Springer, v. 10, n. 12, p. 4789–4818, 2019.

FUKS, H. et al. The 3c collaboration model. In: **Encyclopedia of E-collaboration**. [S.l.]: IGI Global, 2008. p. 637–644.

FUKS, H. et al. **Applying the 3C model to groupware engineering**. Rio de Janeiro – Brazil: PUC Rio de Janeiro, 2004.

GARCÍA-PEÑALVO, F. J. Editorial computational thinking. **IEEE Revista Iberoamericana de Tecnologias del Aprendizaje**, IEEE, v. 13, n. 1, p. 17–19, 2018.

GARCÍA-PEÑALVO, F. J.; MENDES, A. J. **Exploring the computational thinking effects in pre-university education**. [S.l.]: Elsevier, 2018.

GONÇALVES, L. L.; SCHIAVONI, F. L. Creating digital musical instruments with libmosaic-sound and mosaicode. **Revista de Informática Teórica e Aplicada**, v. 27, n. 4, p. 95–107, 2020.

GONÇALVES, L. L.; SCHIAVONI, F. L. **Do Harpia ao Mosaicode a Evolução de um Ambiente de Programação Visual**. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.4243180>.

HIRLEHEI, A. **Enhancing collaboration efficiency through tailorability in synchronous groupware.** Tese (Doutorado) — University of Duisburg-Essen, Germany, 2019.

JANDRE, E.; DIIRR, B.; BRAGANHOLO, V. Uma abordagem para viabilizar experimentos in silico colaborativos. In: **Anais do XV Simpósio Brasileiro de Sistemas Colaborativos**. Porto Alegre, RS, Brasil: SBC, 2019. p. 7–12. ISSN 2326-2842. Disponível em: <https://sol.sbc.org.br/index.php/sbsc/article/view/7798>.

KHORRAM, F.; MOTTU, J.-M.; SUNYé, G. Challenges &; opportunities in low-code testing. In: **Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings**. New York, NY, USA: Association for Computing Machinery, 2020. (MODELS '20). ISBN 9781450381352. Disponível em: <https://doi.org/10.1145/3417990.3420204>.

KUROSE, K. W. R. J. F. **Redes de computadores e internet: uma abordagem topdown**. [S.l.]: São Paulo: Pearson Education do Brasil, 2014.

MAYER-PATEL, K. Mediasynch issues for computer-supported cooperative work. In: **MediaSync**. [S.l.]: Springer, 2018. p. 191–208.

NABBEN, B. The impact of groupware and cscw on group collaboration. In: **International Workshop on Cultures of Participation in the Digital Age**. [S.l.: s.n.], 2019. v. 65, p. 71.

Oltrogge, M. et al. The rise of the citizen developer: Assessing the security impact of online app generators. In: **2018 IEEE Symposium on Security and Privacy (SP)**. San Francisco – CA – USA: IEEE, 2018. p. 634–647.

RESENDE, B.; SANTOS, M. G. dos. Virtualizaçao e educaçao: Desafios além da realidade. **Redin-Revista Educacional Interdisciplinar**, v. 8, n. 1, 2019.

RYMER, J. R. et al. **The Forrester Wave™: Low-Code Development Platforms For AD&D Professionals, Q1 2019**. [S.l.]: Forrester Research, 2019.

SAHAY, A. et al. Supporting the understanding and comparison of low-code development platforms. In: **2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. Portoroz – Slovenia: IEEE, 2020. p. 171–178.

SANDY, J. M. da S. **UAISharing - Interface Universal de Acesso a Recursos Compartilhados**. Tese (Dissertação (Mestrado em Ciência da Computação)) — Universidade Federal de São João del-Rei, 2019.

SCHIAVONI, F. L. Software livre e sustentabilidade. In: **Simpósio Internacional em Artes, Urbanidades e Sustentabilidade**. São João del-Rei - MG - Brazil: [s.n.], 2017. p. 175–184.

SCHIAVONI, F. L. et al. Alice: Arts lab in interfaces, computers, and everything else-research report (2019). In: **Anais do XVII Simpósio Brasileiro de Computação Musical**. São João del-Rei – MG – Brazil: SBC, 2019. p. 157–164.

SCHIAVONI, F. L.; GONÇALVES, L. L. From virtual reality to digital arts with mosaicode. In: **2017 19th Symposium on Virtual and Augmented Reality (SVR)**. [S.l.: s.n.], 2017. p. 200–206.

SPALDING, M. et al. Higher education challenges and possibilities: a brazilian experience in times of covid-19. **Research, Society and Development**, v. 9, n. 8, p. e534985970, Jul. 2020. Disponível em: <https://rsdjournal.org/index.php/rsd/article/view/5970>.

SU, Y.; HAN, L. Data communication method in collaborative process planning. In: SPRINGER. **International Conference on Applications and Techniques in Cyber Security and Intelligence**. [S.l.], 2018. p. 1185–1194.

TRAJANO, D. d. O. **Detecção de linguagem tóxica aplicada a textos em português**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, 2023.

VINCENT, P. et al. Magic quadrant for enterprise low-code application platforms. **Retrieved December**, v. 18, p. 2019, 2019.

WASZKOWSKI, R. Low-code platform for automating business processes in manufacturing. **IFAC-PapersOnLine**, v. 52, n. 10, p. 376 – 381, 2019. ISSN 2405-8963. 13th IFAC Workshop on Intelligent Manufacturing Systems IMS 2019. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2405896319309152>.

Anexos

ANEXO A – CÓDIGO GERADO PELO MOSAICODE

Este é o código gerado a partir do diagrama apresentado anteriormente (Figura 6):

```
<html>
    <head>
        <meta http-equiv="Cache-Control" content="no-store" />
        <!--  GPL 3.0 -->
        <title>Title</title>
        <link rel="stylesheet" type="text/css" href="theme.css">
        <script src="functions.js"></script>
        <script>

        function loadme(){

(function () {
  function SpectrogramVisualizer(audioContext, canvasElement) {
    this.analyserNode = audioContext.createAnalyser();
    this.analyserNode.fftSize = 8192;
    this.fftData = new Float32Array(this.analyserNode.frequencyBinCount)
    ;

    this.graphicWidth = parseInt(getComputedStyle(canvasElement).width,
    10);
    this.graphicHeight = parseInt(getComputedStyle(canvasElement).height
    , 10);

    var gc = this.graphicContext = canvasElement.getContext("2d");
    gc.fillStyle = '#000000';

    this.pixel = gc.createImageData(1,1);
    this.pixel.data[3] = 255;

    this.gain = 0;
    this.stopping = false;

    this.draw();
  }

  SpectrogramVisualizer.prototype.acceptConnection = function (
   connectedNode) {
    connectedNode.connect(this.analyserNode);
    this.connectedNode = connectedNode;
  };

  SpectrogramVisualizer.prototype.draw = function () {
    if (this.stopping) {
```

```
      this.stopping = false;
      return;
    }

    var gc = this.graphicContext;
    var gw = this.graphicWidth;
    var gh = this.graphicHeight;

    if (!this.connectedNode) {
      gc.fillRect(0, 0, gw, gh);
    }
    else {
      var slideImage = gc.getImageData(0, 0, gw - 1, gh);
      gc.putImageData(slideImage, 1, 0);

      var i, y, n;

      this.analyserNode.getFloatFrequencyData(this.fftData);

      for (i = 0; i < gh; ++i) {
        n = Math.min(Math.max((this.fftData[i] + this.gain + 80) * 4, 0)
    , 255);
        this.pixel.data[0] = n;
        this.pixel.data[1] = n;
        this.pixel.data[2] = n;
        gc.putImageData(this.pixel, 0, gh - i);
      }
    }

    this.animationHandle = requestAnimationFrame(function () { this.draw
    (); }.bind(this));
  };

  SpectrogramVisualizer.prototype.releaseConnection = function () {
    this.connectedNode.disconnect(this.analyserNode);
    delete this.connectedNode;
  };

  SpectrogramVisualizer.prototype.stop = function () {
    this.stopping = true;
  };

  window.App = window.App || {};
  window.App.SpectrogramVisualizer = SpectrogramVisualizer;
})();

(function () {
```

```
85      var visCanvas_2 = document.getElementById('spectrogram_2')
86      var visualizer_2 = new  App.SpectrogramVisualizer(context,
    visCanvas_2);
87      visualizer_2.gain = 1;
88      visualizer_2.acceptConnection(block_2);
89  })();
90
91          return;
92          }
93          var context = new (window.AudioContext || window.
    webkitAudioContext)();
94          //declaration block
95
96  </script>
97  <div>
98      <span class="title">Microphone</span>
99  </div>
100 <div>
101     <div>
102         <span class="label">Gain</span>
103         <input type="range" id="gain-slider_1" class="slider" min="-20"
    max="20" step="1" value="0" />
104         <span id="gain-display_1" class="label">0 db</span>
105     </div>
106 </div>
107
108 <script>
109     var gain = 0;
110     var audioSource;
111     var block_1;
112
113     block_1  = context.createBiquadFilter();
114     block_1.type = "lowshelf";
115     block_1.frequency.value = 1000;
116
117     var Microphone_1_output = block_1;
118
119     navigator.getUserMedia =
120         navigator.getUserMedia ||
121         navigator.webkitGetUserMedia ||
122         navigator.mozGetUserMedia ||
123         navigator.msGetUserMedia;
124
125 if (navigator.getUserMedia) {
126     navigator.getUserMedia (
127       {
128         audio: true,
```

```
        video: false
      },
      function (stream) {
        audioSource = context.createMediaStreamSource(stream);

         var gainDisplay = document.getElementById('gain-display_1');
         var gainSlider = document.getElementById('gain-slider_1');

         gainSlider.addEventListener('input', function () {
           gain = parseFloat(gainSlider.value)
           gainDisplay.textContent = gain + ' db';
           block_1.gain.value = gainSlider.value;
         });

         audioSource.connect(block_1);
      },
      function (err) {
         console.log('Error initializing user media stream: ' + err);
      }
    );
}

var block_2 = context.createGain();
block_2.gain = 1;
var Spectrogram_2_input = block_2;
        //execution

        //connections
        Microphone_1_output.connect(Spectrogram_2_input);
        </script>
    </head>

    <body onload='loadme();'>

<h2>SpectrogramVisualizer_Block2</h2>
<div>
  <canvas id="spectrogram_2" width="1024" height="300"></canvas>
</div>
    </body>
</html>
```

Listing A.1 – Código gerado a partir do diagrama a Figura 6

ANEXO B – ESTRUTURA DE PASTAS DOS ARTEFATOS

Estrutura de pasta dos artefatos e arquivos, incluindo os já existentes e os novos:

```
mosaicode/
├── code-gen/
│   ├── index.html/
│   ├── functions.js/
│   └── theme.css/
├── configuration.json
├── extensions/
│   ├── blocks/
│   │   └── javascript/
│   │       └── webaudio/
│   │           ├── Sound
│   │           │   ├── Playback.json
│   │           │   ├── White Noise.json
│   │           │   └── Speaker.json
│   │           ├── GUI/
│   │           │   └── Freq Bar.json
│   │           └── Outra categoria/
│   │               └── Outro bloco.json
│   ├── codetemplates/
│   │   └── webaudio.json
│   ├── examples/
│   │   └── Noise.mscd
│   └── ports/
│       ├── javascript.float.json
│       ├── javascript.sound.json
│       └── javascript.string.json
└── plugins/
    ├── __init__.py
    └── extensionmanager/
        ├── __init__.py
        ├── blockcodeeditor.py
        ├── blockcommoneditor.py
        ├── blockeditor.py
        ├── blockmanager.py
        └── blockporteditor.py
```

```
        ├── codetemplatecodeeditor.py
        ├── codetemplateeditor.py
        ├── codetemplatemanager.py
        ├── extensionsmanager.py
        ├── manager.py
        ├── porteditor.py
        ├── portmanager.py
        └── propertyeditor.py
    ├── users/
    │   └── configuration.json
    └── workspace/
        └── diagrams/
            └── javascript/
                └── webaudio/
                    └── additive-synth-project.mscd
```