UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

MCLYNDON SAINT-CHRISTIE DE LIMA XAVIER

UMLCOLLAB:
UMA ABORDAGEM HÍBRIDA PARA MODELAGEM COLABORATIVA DE MODELOS UML

São Leopoldo
2019

McLyndon Saint-Christie de Lima Xavier

UMLCOLLAB:
Uma Abordagem Híbrida para Modelagem Colaborativa de Modelos UML

Dissertação apresentada como requisito parcial
para a obtenção do título de Mestre pelo
Programa de Pós-Graduação em Computação
Aplicada da Universidade do Vale do Rio dos
Sinos — UNISINOS

Orientador:
Prof. Dr. Kleinner Silva Farias de Oliveira

Coorientador:
Prof. Dr. Jorge Luis Victória Barbosa

São Leopoldo
2019

X3u Xavier, McLyndon Saint-Christie de Lima.
UMLCollab : uma abordagem híbrida para modelagem colaborativa de modelos UML / por McLyndon Saint-Christie de Lima Xavier. – 2019.
92 f. : il. ; 30 cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2019.
Orientador: Dr. Kleinner Silva Farias de Oliveira.
Coorientador: Dr. Jorge Luis Victória Barbosa.

1. Engenharia de software. 2. Modelagem colaborativa. 3. Resolução de conflitos. 4. Colaboração assíncrona. 5. Colaboração síncrona. I. Título.

CDU: 004.41

Catalogação na Publicação (CIP):
Bibliotecário Alessandro Dietrich - CRB 10/2338

UNISINOS

UNIVERSIDADE DO VALE DO RIO DOS SINOS
Escola Politécnica
Programa de Pós-Graduação em Computação Aplicada - PPGCA

ATA DE BANCA EXAMINADORA DE DISSERTAÇÃO DE MESTRADO Nº 22/2019

Aluno: McLyndon Saint-Christie de Lima Xavier

Título da Dissertação: "UMLCOLLAB: UMA ABORDAGEM HÍBRIDA PARA MODELAGEM COLABORATIVA DE MODELOS UML"

Banca:
Prof. Dr. Kleinner Silva Farias de Oliveira – Orientador
Prof. Dr. Jorge Luis Victória Barbosa - Coorientador
Prof. Dr. Gustavo Pessin - UFOP
Prof. Dr. Sandro Rigo – UNISINOS

Aos vinte e nove dias do mês de agosto do ano de 2019, às 14h30 reuniu-se na sala C02 212, a Comissão Examinadora de Defesa de Dissertação composta pelos professores: Prof. Dr. Kleinner Silva Farias de Oliveira, Orientador – UNISINOS; Prof. Dr. Jorge Luis Victória Barbosa - Coorientador – UNISINOS; Prof. Dr. Gustavo Pessin, Membro da Banca – UFOP (por web conferência); Prof. Dr. Sandro José Rigo, Membro da Banca – UNISINOS; para analisar e avaliar a Dissertação apresentada pelo aluno McLyndon Saint-Christie de Lima Xavier.

Considerações da Banca:

A banca examinadora se reuniu e deliberou que a dissertação cumpriu com os requisitos, bem como recomendou melhorias.

Ocorreu alteração do título? (X)Não ( ) Sim
Indicar o novo título: .....

A Banca Examinadora, em cumprimento ao requisito exigido para a obtenção do Título de Mestre em Computação Aplicada, julga esta dissertação:

(X) APROVADA ( ) REPROVADA

Conforme Artigo 67 do Regimento do Programa o texto definitivo, com aprovação do Orientador, deverá ser entregue no prazo máximo de sessenta (60) dias após a defesa. O resultado da banca é de consenso entre os avaliadores. A emissão do Diploma está condicionada a entrega da versão final da Dissertação. A Ata de Defesa é assinada pelos membros que participaram da sessão de forma presencial.

São Leopoldo, 29 de agosto de 2019

Prof. Dr. Kleinner Silva Farias de Oliveira

Prof. Dr. Sandro José Rigo

Prof. Dr. Jorge Luis Victória Barbosa

Mestrando McLyndon Saint-Christie de L. Xavier

Dedico esse trabalho aos meus pais, Paulo Xavier e Rosilda Xavier que investiram na minha educação básica e não pouparam tempo e dinheiro para vencer as doenças crônicas da minha infância e permitiram tornar-me um adulto produtivo para a sociedade.

AGRADECIMENTOS

Agradeço primeiramente a Deus pela conclusão desse trabalho e sem o qual nenhuma conquista faz sentido.

Agradeço a minha esposa, Redvânia Xavier, a grande inspiração para cursar o mestrado e exemplo de esforço e persistência nos estudos.

Agradeço ao meu orientador e coorientador pela paciência e dedicação em orientar a elaboração desse trabalho.

Finalmente agradeço a Unisinos pela oportunidade de cursar o mestrado, aos bolsistas que auxiliaram na minha orientação, aos participantes do experimento e a todas as demais pessoas que direta ou indiretamente permitiram a conclusão desse trabalho.

"...Não é sobre chegar no topo do mundo e saber que venceu
É sobre escalar e sentir que o caminho te fortaleceu
É sobre ser abrigo e também ter morada em outros corações
E assim ter amigos contigo em todas as situações..."

Ana Carolina Vilela da Costa

RESUMO

A modelagem colaborativa de software é uma tendência para aumentar a produtividade das empresas e reduzir custos. Porém, os dois principais tipos de colaboração ainda apresentam problemas, por exemplo: a síncrona, apesar de evitar conflitos, em cenários com vários desenvolvedores atuando no modelo ao mesmo tempo, atrapalha o processo cognitivo dos desenvolvedores e a assíncrona leva a complicadas e custosas etapas de resolução de conflitos. Portanto, este trabalho propõe a UMLCollab, uma abordagem híbrida de modelagem colaborativa de modelos UML. A UMLCollab permite que cada usuário receba atualizações síncronas de outros usuários e envie as suas alterações de forma assíncrona, combinadas com a técnica de *merge* automático e manual. A abordagem proposta foi avaliada através de um experimento controlado, o qual permitiu compará-la com abordagens tradicionais (síncronas e assíncronas). Os resultados coletados mostram que a UMLCollab: (1) apresentou uma produtividade (esforço) intermediária em relação a formas de colaboração tradicionais; (2) apresentou um maior nível de corretude considerando o máximo das amostras de corretude coletadas em relação a colaboração síncrona; (3) recebeu uma melhor percepção dos participantes nos resultados de produtividade; e (4) foi confirmada pela maioria dos participantes como facilitadora na resolução de conflitos e redução da interferência na modelagem. Por fim, os resultados são encorajadores e mostram o potencial de usar a UMLCollab para suportar modelagem colaborativa em ambientes reais.

Palavras-chave: Engenharia de Software. Modelagem Colaborativa. Resolução de Conflitos. Colaboração Assíncrona. Colaboração Síncrona.

ABSTRACT

Collaborative software modeling is a trend to increase business productivity and reduce costs. However, the two main types of collaboration still present problems, for instance: synchronous, despite avoiding conflicts, in scenarios with several developers acting on the model at the same time, it disrupts the cognitive process of the developers and the asynchronous leads to complicated and costly steps of conflict resolution. Therefore, this work proposes the "UMLCollab", a hybrid approach of collaborative modeling of UML models. UMLCollab allows each user receive synchronous updates from other users and send their changes asynchronously, combined with the automatic and manual merge technique. The proposed approach was evaluated through a controlled experiment, which allowed comparing it with traditional approaches (synchronous and asynchronous). The results collected show that UMLCollab: (1) showed an intermediate productivity (effort) in relation to traditional forms of collaboration; (2) showed a higher level of correctness considering the maximum of the correctness samples collected in relation to the synchronous collaboration; (3) received a better perception of the participants in the results of productiveness; and (4) was confirmed by most participants as a facilitator in conflict resolution and modeling interference reduction. Finally, the results are encouraging and show the potential of using UMLCollab to support collaborative modeling in real environments.

Keywords: Software Engineering. Collaborative Modeling. Conflict Resolution. Asynchronous Collaboration. Synchronous collaboration.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

IBM	International Business Machines
PPGCA	Programa de Pós-Graduação em Computação Aplicada
MVC	Modelo-Visão-Controlador
RSAD	Rational Software Architect Designer
SCV	Sistema de Controle de Versão
SVN	Subversion
UNISINOS	Universidade do Vale do Rio dos Sinos
UML	Linguagem de Modelagem Unificada
UUID	Identificador Único Universal

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de software feito de forma colaborativa e distribuída geograficamente, como observa-se a bastante tempo, é uma tendência crescente (MUCCINI; BOSCH; VAN DER HOEK, 2018; FARIAS et al., 2015; KOLOVOS et al., 2013; BRAMBILLA; CABOT; WIMMER, 2012). Nesse ambiente, engenheiros trabalham de forma independente e simultânea nos mesmos artefatos do software (DAM; GHOSE, 2011). Um dos motivos para essa tendência, como observa-se a bastante tempo, é a escassez de profissionais com os conhecimentos e habilidades necessárias para projetos de grande porte em um único local (ROCHA, 2015; PRIKLADNICKI; CARMEL, 2013). A redução de custos e do tempo de desenvolvimento são outros motivos para o avanço dos softwares de forma colaborativa e distribuída (ROCHA, 2015).

Portillo-Rodríguez et al. (2012) informam que apesar das vantagens, o desenvolvimento de software de forma distribuída apresenta seus próprios desafios. Conforme estes autores, a distância geográfica reduz a qualidade da comunicação entre os membros da equipe e dificulta o gerenciamento das equipes. Além disso, as diferenças culturais causam mal-entendidos e a relação de confiança entre os membros fica prejudicada.

Através do desenvolvimento de software dirigido por modelos foi reconhecido a relevância dos modelos para o processo de desenvolvimento de software, por isso é importante a continuidade do suporte às alterações nos modelos (DAM; GHOSE, 2011). A técnica de modelagem de software é largamente utilizada durante a fase de levantamento de requisitos e a fase de análise em projetos de desenvolvimento de softwares. Brosch et al. (2009) e Serçe et al. (2011) citado por Polančič e Jošt (2016) afirmam que as organizações precisam de ferramentas efetivas e eficientes, com interações síncronas e assíncronas, para colaboração em ambientes distribuídos.

Na mesma linha, sobre a colaboração em ambientes distribuídos, Wateridge (1999) apud Polančič e Jošt (2016) afirma que cada membro da equipe precisa está ciente das mudanças realizadas por outros membros e ao mesmo tempo informar sobre as suas alterações. Corroborando Costa e Murta (2013) afirmam que há a necessidade de ferramentas para detectar conflitos em tempo real. Muito esforço foi colocado na resolução de conflitos de código, porém pouco foi investido na resolução de conflitos nos modelos (DAM; GHOSE, 2011). Como a manipulação dos modelos é feita em tempo real, a sincronização das mudanças é o principal desafio da modelagem colaborativa de aplicações (KURYAZOV; WINTER, 2015).

Neste contexto, possivelmente as colaborações síncronas e assíncronas necessitam de aprimoramentos para permitir que o desenvolvimento colaborativo ocorra de forma eficiente, evitando a grande quantidade de tempo gasto no processo de resolução de conflitos.

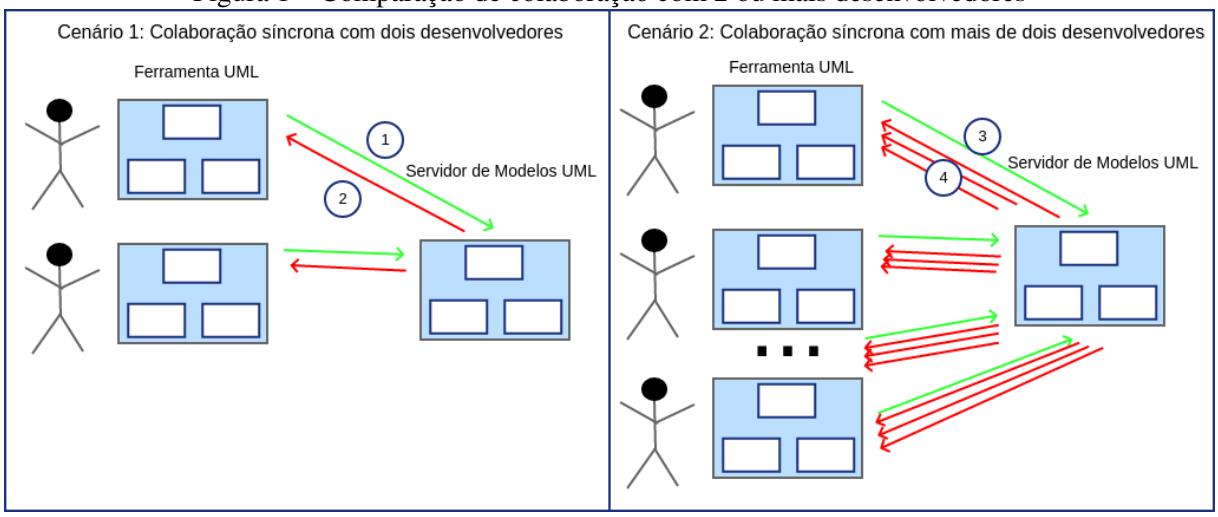
1.1 Problemática

O processo de desenvolvimento colaborativo apresenta o desafio de realizar o *merge* das alterações realizadas pelos colaboradores de forma automática, porém, apesar de várias técnicas propostas na literatura, nenhuma delas locrou êxito em automatizar completamente esse processo (GHIOTTO et al., 2018).

Como supracitado, a modelagem colaborativa de software é usada para aumentar a produtividade nas empresas e elaborar modelos completos e consistentes. A literatura cita dois tipos de colaboração, a síncrona e a assíncrona. Ambas as soluções apresentam suas vantagens, porém possuem deficiências com reflexos na geração de conflitos e na produtividade dos desenvolvedores.

Na síncrona cada desenvolvedor envia e recebe alterações dos demais colaboradores de forma quase instantânea. Esse tipo de colaboração é interessante, pois evita conflitos nos modelos, no entanto, o recebimento constante de alterações vindas dos demais colaboradores, especialmente em cenários com mais de dois colaboradores, pode causar interferência no trabalho cognitivo de cada desenvolvedor devido ao excesso de interrupções (BRIERTON et al., 2016).

Figura 1 – Comparação de colaboração com 2 ou mais desenvolvedores



Fonte: Elaborada pelo autor

Na Figura 1, exemplifica-se o problema da colaboração síncrona através de dois cenários. Em ambos os cenários, há um servidor de modelos UML (Unified Modeling Language) responsável por receber as alterações de cada desenvolvedor e enviá-las a cada desenvolvedor conectado colaborando no mesmo modelo. No cenário 1, com somente 2 desenvolvedores, ocorre o envio síncrono das alterações locais (passo 1) e o recebimento síncrono das alterações remotas do outro desenvolvedor (passo 2). Como somente existe outro desenvolvedor colaborando com o modelo simultaneamente, a interferência no trabalho de cada usuário pode ser considerada como razoável. No cenário 2, porém, fica evidente que a maior quantidade de desenvolvedores simultâneos pode prejudicar do trabalho de cada desenvolvedor. Nesse, semelhante ao cenário

1, o desenvolvedor envia suas alterações locais de forma síncrona (passo 3), porém o recebimento das alterações remotas (passo 4) é proporcional ao número de usuários simultâneos trabalhando no modelo podendo causar muita interferência no trabalho de cada usuário.

Na colaboração assíncrona, a situação ocorre diferente, ou seja, o envio e recebimento de alterações é feito manualmente pelo desenvolvedor, isto é, a sincronização ocorre quando o desenvolvedor acessa o repositório onde estão os arquivos dos modelos, normalmente através de um sistema de controle de versão como Subversion (SVN) ou Git. Dessa forma, o desenvolvedor consegue trabalhar nas suas alterações e só receberá as dos demais colaboradores quando desejar. Esse tipo de colaboração traz consigo a alta complexidade e esforço para a solução de conflitos que tendem se intensificar quando maior for o tempo que os arquivos dos modelos locais não forem sincronizados com o repositório.

Dado a problemática existente e contextualizada, a pesquisa propõe-se em responder à questão de pesquisa abordada no próximo tópico.

1.2 Questão de pesquisa

Esse trabalho apresenta uma abordagem híbrida que combina as vantagens de ambos os tipos de colaboração (síncrona e assíncrona), ao passo que procura minimizar as deficiências de cada uma delas. Dessa forma surge a seguinte questão de pesquisa:

Questão de Pesquisa: Como seria possível combinar as vantagens da colaboração síncrona com a colaboração assíncrona de forma a permitir a modelagem colaborativa com menor esforço e maior produtividade para o desenvolvedor?

1.3 Objetivos

O tempo gasto com a resolução de conflitos é um problema na modelagem colaborativa de software e é influenciado pela técnica de sincronização utilizada, como a técnica síncrona e a assíncrona, por conta disso o presente estudo tem como objetivo geral:

Propor *uma abordagem de sincronização e resolução de conflitos*
com o propósito de *melhorar a colaboração na elaboração dos modelos de software*
em respeito ao *esforço, conflitos e corretude*
a partir da *perspectiva de analistas e desenvolvedores*
no contexto de *modelagem colaborativa de software.*

Para obtenção da resposta perante a questão abordada, destacam-se os seguintes objetivos específicos:

- Analisar e identificar as técnicas de sincronização e resolução de conflitos disponíveis na literatura;

- Propor uma abordagem, nomeada de UMLCollab, com menor nível de interferência do trabalho do desenvolvedor e maior nível de colaboração, porém com menor esforço ou maior corretude dos modelos elaborados.

- Executar um experimento para avaliar, analisar e comparar as eficiências e deficiências das técnicas síncronas e assíncronas em comparação a UMLCollab.

1.4 Metodologia

Trata-se de uma pesquisa de cunho qualitativo e quantitativo e de caráter exploratório. O estudo é qualitativo, pois utilizou como estratégia de pesquisa a revisão da literatura e quantitativo pois além de se tratar de um experimento, são utilizados testes estatísticos para analisar os resultados e validar ou rejeitar as hipóteses do experimento. Este estudo é exploratório, pois esta pesquisa tem por objetivo examinar um tema ou uma questão de pesquisa que ainda foi pouco estudado (PAULA, 2010), ou seja, as técnicas de modelagem colaborativa de software, tema este que tem escassez de pesquisas empíricas (FARIAS, 2010) ainda pouco conhecimento e muitas dúvidas a respeito do assunto. Iniciativas de pesquisa tendem a se concentrar em propror a composição de modelos e técnicas ou mesmo criando linguagens de modelagem inovadoras (FARIAS et al., 2015).

O estudo foi dividido em três grandes etapas. A primeira etapa foi uma pesquisa qualitativa, onde se utilizou como método de pesquisa a revisão da literatura, com o objetivo de encontrar pesquisas empíricas sobre a modelagem colaborativa de software, com o intuito de identificar as técnicas de sincronização e resolução de conflitos.

A segunda etapa foi estruturada com base nos dados apurados da revisão da literatura e consistiu na elaboração de uma proposta de sincronização e resolução de conflitos com o objetivo de reduzir o esforço, melhorar a corretude e aumentar a eficiência dos modelos criados. Já a terceira e última etapa foi a realização de um experimento, com o uso de um instrumento (Checklist), para avaliar os resultados da proposta e comparar com as técnicas mais usadas de sincronização e resolução de conflitos pela indústria e pela academia.

1.5 Contribuições da pesquisa

Esta pesquisa apresenta duas grandes contribuições: para a teoria, contribui com uma nova abordagem híbrida empírica para melhorar a sincronização e resolução de conflitos na elaboração de modelos colaborativos de software, e na prática, para a sociedade, oferece a redução do esforço dos desenvolvedores na produção de projetos colaborativos, assim como o aumento da eficiência das próximas modelagens de software que serão criados, ou seja, este estudo empírico contribuirá para a literatura, pois, além de preencher a lacuna escassa, propõe um novo modelo teórico sobre a resolução dos conflitos da modelagem colaborativa de software. E em relação a sociedade, o estudo empírico contribuirá para permitir a redução do esforço dos desenvolvedo-

res e aumentar a eficiência na elaboração dos modelos colaborativos criados.

1.6 Organização do trabalho

Este trabalho dividido em seis capítulos, sendo esta introdução o primeiro capítulo e os demais descritos a seguir: o Capítulo 2 refere-se a fundamentação teórica necessária ao entendimento desse estudo, com conceitos e desafios relacionados a modelagem colaborativa de software; o Capítulo 3 trata-se dos trabalhos relacionados a modelagem colaborativa de software que buscou identificar os avanços realizados nessa área; no Capítulo 4 apresenta-se a proposta para tentar aumentar a eficiência na resolução de conflitos; o Capítulo 5 apresenta um experimento para avaliar a proposta; e o Capítulo 6 apresenta uma conclusão dos resultados obtidos e traça caminhos de pesquisa e desafios a serem vencidos na modelagem colaborativa de software.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo aborda a fundamentação teórica relacionada ao tema desta pesquisa. Na Seção 2.1 são apresentados os conceitos relacionados a modelagem colaborativa de software. A Seção 2.2 aborda resolução de conflitos. Finalmente, nas Seções 2.3 e 2.4 explanam sobre as colaborações síncrona e assíncrona, respectivamente.

2.1 Modelagem colaborativa de software

A modelagem de software colaborativa é uma tendência crescente na academia e na prática (MUCCINI; BOSCH; VAN DER HOEK, 2018; FARIAS et al., 2015; KOLOVOS et al., 2013; BRAMBILLA; CABOT; WIMMER, 2012). Os modelos no contexto do desenvolvimento de software são usados para representar, de forma simplificada, elementos de nossa realidade, com o intuito de permitir o entendimento dos conceitos e dinâmicas relacionados ao sistema sendo desenvolvido (OSMAN, 2013). Além disso, segundo Osman (2013), o desenvolvimento de software é atualmente uma atividade colaborativa entre vários desenvolvedores que vêm atraindo a atenção ao longo dos anos, logo a colaboração é essencial para os projetos de software.

Caso os desenvolvedores de modelagem de software atuem no mesmo local, eles podem interagirem diretamente conversando sobre os modelos sendo elaborados de forma informal, discutindo, resolvendo conflitos e tomando decisões juntos (OSMAN, 2013), porém, quando os desenvolvedores estão separados geograficamente, essa interação pode ser problemática (SARMA; VAN DER HOEK, 2006). A ferramenta de modelagem a distância ainda depende das vias de comunicação, como por exemplo, o chat, a videoconferência e os mecanismos de compartilhamento como Dropbox ou anexos de e-mail, que geralmente afetam negativamente a produtividade (OSMAN, 2013).

Na mesma linha, Lucas et al. (2017) afirmam que conduzir as atividades de reuso de software ainda é um problema complicado de resolver, particularmente quando se refere ao desenvolvimento de software colaborativo. Os autores explicam que os sistemas de software modernos são desenvolvidos em grupo, trabalhando em locais geograficamente separados e por isso, as técnicas para enriquecer a colaboração no desenvolvimento de software são relevantes para melhorar a qualidade e reduzir custos (LUCAS et al., 2017). Ferramentas para apoiar o trabalho colaborativo são crucialmente necessárias, pois os sistemas de software modernos são desenvolvidos por pessoas que trabalham juntos e a complexidade desses sistemas requer conhecimento relacionado em vários campos, logo a colaboração entre as pessoas surge como um fator importante para o sucesso do desenvolvimento de um projeto de software (LUCAS et al., 2017; BARTHELMESS; ANDERSON, 2002).

Já é reconhecida por pesquisadores e profissionais, a importância da produção de protótipos no desenvolvimento de software centralizado em modelos (FARIAS, 2010). Porém, existe a escassez de evidências empíricas sobre o impacto das técnicas de composição de modelos no

esforço dos desenvolvedores. Por isso, os engenheiros de software ainda permanecem sem qualquer orientação sobre como manusear corretamente certas técnicas de modelo de uma maneira que efetivamente reduza seus esforços de desenvolvimento (FARIAS, 2010).

2.2 Resolução de conflitos

No processo de desenvolvimento colaborativo é necessário integrar alterações de vários desenvolvedores. Nesse processo de integração ou *merge*, pode ocorre conflitos, logo é essencial que ocorra a comunicação dos desenvolvedores certos para chegar a solução desejada (SOUZA COSTA et al., 2019).

Altmanninger, Seidl e Wimmer citado por Bang e Popescu (2012) apontam dois métodos para os sistemas de controle de versão: o pessimista, com bloqueio em que cada artefato só possa ser alterado por um membro da equipe em um determinado momento; e a otimista, aonde cada membro têm sua cópia local dos artefatos e têm que fazer *merge* depois.

O método pessimista não evita conflitos de dependência entre artefatos e impede que mais de um membro trabalhe no modelo ao mesmo tempo. A abordagem otimista por outro lado, não informa as alterações dos outros engenheiros, logo mais decisões de projeto conflitantes ocorrem. Além disso, como a notificação sobre conflitos são adiadas até o *merge*, a resolução é dificultada e ocupa muito tempo. Apesar de geralmente ser possível fazer *merge* automático de alterações em diferentes partes de documentos, é muito mais complicado tratar documentos não textuais (De Lucia et al., 2007).

Bang e Popescu (2012) informam ainda que as ferramentas de modelagem devem tratar de dois tipos de conflitos: conflitos de sincronização e conflitos de alta ordem. Os conflitos de sincronização são aqueles que ocorrem na alteração de um mesmo artefato ou outro diretamente relacionado. Conflitos de alta ordem ocorrem por violações da sintaxe de modelagem ou semântica dos modelos e são mais difíceis e custosos para detectar.

Dam e Ghose (2011) chamam esses conflitos de conflitos diretos e indiretos, respectivamente. Além desses problemas, o ambiente de modelagem colaborativa não é confiável pois mudanças feitas por um engenheiro podem falhar devido a alterações concorrentes, como a renomeação de um método excluído por outro engenheiro sem o conhecimento do primeiro (DAM; GHOSE, 2011). Dam e Ghose (2011) acrescenta que, com relação as várias visões de um modelo UML, como diagramas de classes e diagramas de sequência, é importante manter a consistência sintática (conformidade com o metamodelo) e semântica (coerência entre as diferentes visões de um modelo).

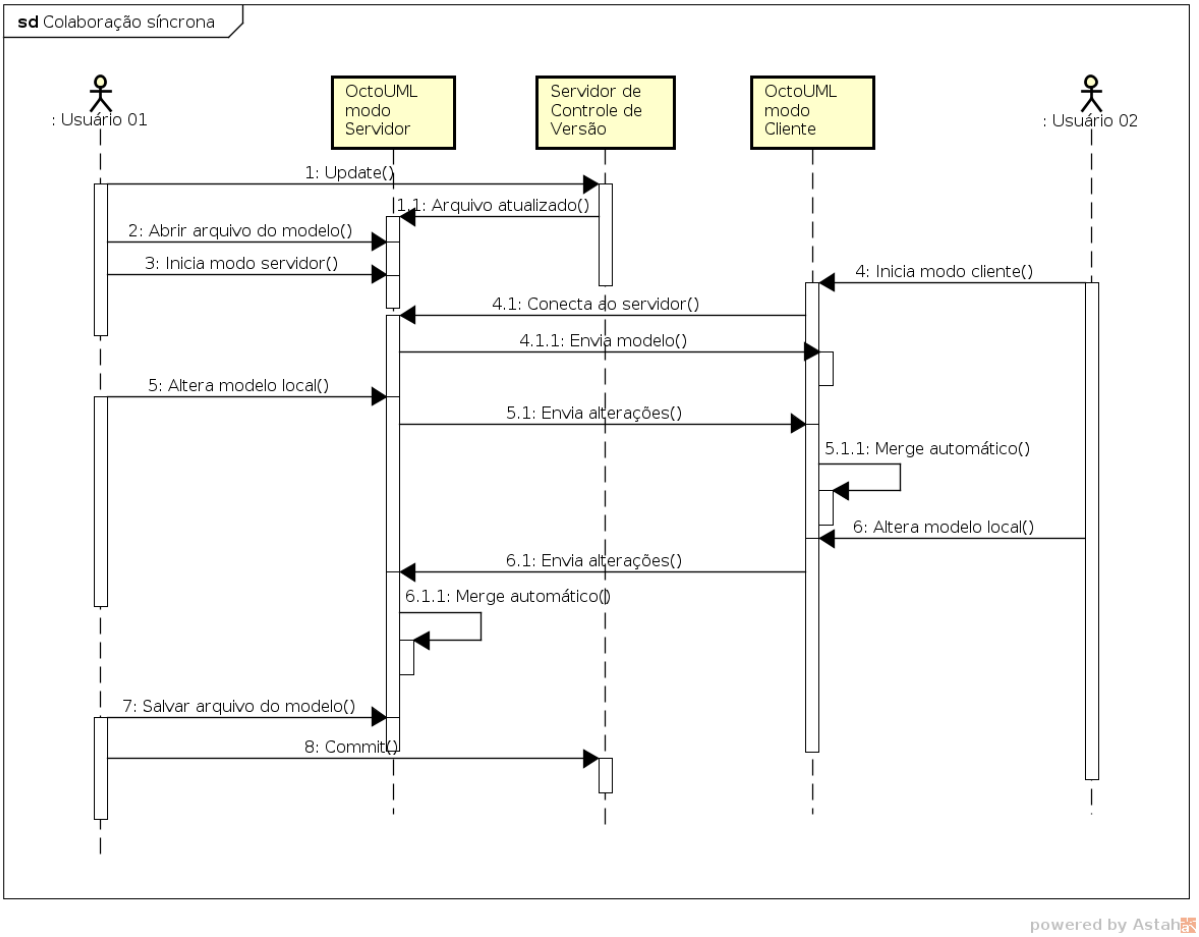
2.3 Colaboração síncrona

A colaboração síncrona ou colaboração em tempo real tem sido utilizada em ferramentas como GenMyModel (DIRIX; MULLER; ARANEGA, 2013) para permitir vários desen-

volvedores colaborarem simultaneamente no mesmo modelo evitando conflitos e permitindo modelos mais consistentes (OSMAN, 2013). Nessa colaboração, cada vez que um desenvolvedor faz uma alteração no modelo, esta alteração é enviada a todos os demais desenvolvedores que também estão com aquele modelo sendo visualizado ou alterado, dessa forma, na visão dos desenvolvedores, parece que estão desenhando juntos em um quadro branco compartilhado (OSMAN, 2013).

A Figura 2 apresenta um diagrama de sequência da UML para exibir o passo a passo típicos na colaboração síncrona, usando como exemplo a ferramenta desktop OctoUML (VESIN; JOLAK; CHAUDRON, 2017) e um sistema de controle de versão em separado, conforme detalhado a seguir. Salienta-se que o sistema de controle de versão não é obrigatório na colaboração síncrona, ele é apenas utilizado nesse exemplo para demostrar o uso prático das tecnologias em conjunto.

Figura 2 – Colaboração síncrona



Fonte: Elaborada pelo autor

- **Passos 1 e 1.1:** Nesse passo o desenvolvedor faz o recebimento (update) da versão mais recente do arquivo contendo os modelos a partir do sistema de controle de versão utilizado. Deve ser observado que nesse cenário de uso da colaboração síncrona com o

OctoUML, somente um usuário precisa fazer o update dos arquivos, pois todas as colaborações posteriores serão realizadas utilizando somente essa ferramenta.

- **Passo 2:** O desenvolvedor carrega o arquivo dos modelos, em formato xml, com a ferramenta OctoUML.

- **Passo 3:** O desenvolvedor inicia o modo servidor da ferramenta. Somente um usuário necessita iniciar o modo servidor. Todos os demais se conectam como clientes para receberem as atualizações em tempo real.

- **Passos 4. 4.1 e 4.1.1:** Cada desenvolvedor que desejar colaborar no modelo inicia o modo cliente do OctoUML ao ser conectado no servidor. Ao realizar esse procedimento, o OctoUML no modo servidor, automaticamente envia uma cópia completa e sincronizada do modelo para o cliente. Essa cópia local é carregada e manipulada pelo cliente e permite a comparação do modelo local com alterações recebidas de outros desenvolvedores.

- **Passos 5, 5.1 e 5.1.1:** Do passos 5 em diante, é apresentado o processo de colaboração síncrona. Nesse passos, o usuário altera o modelo local e de forma automática e imediata o sistema envia as alterações para o servidor, que faz *merge* automática na cópia local do modelo. Como as alterações são enviadas imediatamente, conflitos são evitados e caso ocorram, normalmente a técnica empregada é simplesmente sobrescrever com a última alteração.

- **Passos 6, 6.1 e 6.1.1:** O mesmo do passo 5, só que o cliente é quem fez uma alteração.

- **Passos 7 e 8:** Após toda a interação, qualquer desenvolvedor pode submeter a versão final do modelo alterada colaborativamente ao sistema de controle de versão. Conforme descrito na problemática desse trabalho, apesar da colaboração síncrona evitar conflitos, pode potencialmente atrapalhar o processo cognitivo de elaboração dos modelos (BRIERTON et al., 2016), principalmente quando mais do que 2 usuários trabalham colaborativamente no mesmo modelo.

2.4 Colaboração assíncrona

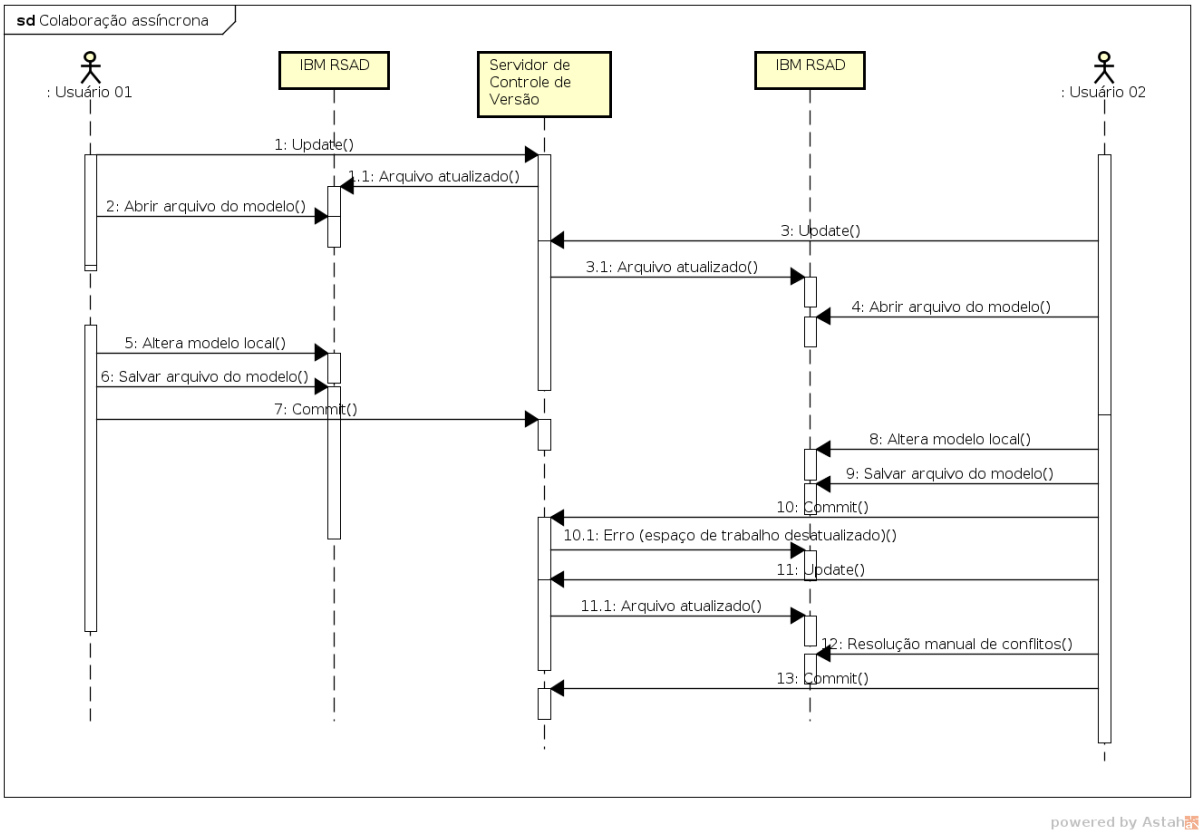
Na concorrência assíncrona, cada desenvolvedor mantém sua cópia dos arquivos dos modelos a qual é sincronizada manualmente com um repositório por cada desenvolvedor (FRANZAGO et al., 2018). Ainda segundo Franzago et al. (2018) é comum o uso de um sistema de controle de versão para esse tipo de colaboração para permitir identificar os conflitos ao tentar atualizar os modelos alterados.

Nesse tipo de colaboração, cada desenvolvedor sofre com a falta de informações sobre o que os outros desenvolvedores estão fazendo, pois somente quando o mesmo sincroniza com o repositório ele pode verificar as alterações dos demais desenvolvedores (OSMAN, 2013).

Osman (2013) também esclarece que como cada desenvolvedor não poderá ver as alterações dos demais até os mesmos as compartilhem, isso leva a muitos conflitos e alterações inconsistentes que precisam serem resolvidas.

Na Figura 3 é apresentado um exemplo através de um diagrama sequência de como ocorre a colaboração assíncrona. Nesse exemplo é utilizado o software desktop International Business Machines (IBM) Rational Software Arquitect Designer (RSAD) (LEROUX; NALLY; HUSSEY, 2006). O passo a passo da colaboração assíncrona que pode ser verificada abaixo.

Figura 3 – Colaboração assíncrona



Fonte: Elaborada pelo autor

- **Passos 1, 1.1 e 2:** Similar ao exemplificando na colaboração síncrona, nesse passo o desenvolvedor faz o recebimento (update) da versão mais recente do arquivo contendo os modelos a partir do sistema de controle de versão utilizado. Aqui o diferencial é que cada usuário precisa fazer o update dos arquivos, pois na colaboração síncrona, cada usuário realiza suas alterações e compartilha através do sistema de controle de versão.
- **Passos 3, 3.1 e 4:** Mesmos passos anteriores, porém realizado por outro usuário.
- **Passos 5, 6 e 7:** Quando o usuário quer enviar suas alterações relizadas no passo 5 e receber as alterações dos demais usuários, ele salva o arquivo do modelo (passo 6) e tenta efetuar o envio das alterações (commit). Nesse exemplo, a versão de base do modelo que o usuário utilizou estava atualizada e o sistema conseguirá realizar o commit.

- **Passos 8, 9, 10 e 10.1:** O segundo usuário, usando a mesma versão de base do modelo, realiza alterações e tenta realizar o commit. Nesse exemplo, a versão de base do modelo que o usuário utilizou estava desatualizada, pois o primeiro fez o commit primeiro. Nesse caso o sistema de controle de versão acusa o erro não permite o commit.

- **Passos 11, 11.1, 12 e 13:** Para resolver o conflito, o segundo usuário faz update (passo 11) para receber o arquivo do modelo (passo 11.1) mais atualizado do repositório, realiza a resolução manual dos conflitos (passo 12) utilizando a recurso de sincronização do IBM RSAD, sinaliza a resolução dos conflitos e efetua novamente o commit (passo 13).

O passo 12 de resolução manual dos conflitos é o que torna a colaboração assíncrona custosa, pois quanto mais desenvolvedores estivem atuando no modelo e quanto mais o tempo que o arquivo não foi atualizado, maior a quantidade e a complexidade dos conflitos que podem ocorrer, levando ao consumo de tempo significativo para resolução de tais conflitos.

3 TRABALHOS RELACIONADOS

Este capítulo tem por finalidade evidenciar o estudo realizado na literatura, ou seja, a busca das pesquisas empíricas sobre a modelagem colaborativa de software. Para isso, aplicou-se uma revisão da literatura, com o intuito de identificar as técnicas de sincronização e resolução de conflitos e outras questões de pesquisa.

3.1 Revisão da literatura

A revisão da literatura foi mapeada em etapas e critérios, visando encontrar estudos sobre o desenvolvimento de software com modelos de forma colaborativa. Para iniciar o processo de revisão da literatura desta pesquisa e encontrar os artigos, primeiramente foram selecionadas as bases de dados. Essas bases de dados foram escolhidas pois representam as bases que contém as pesquisas com alto fator de impacto na atualidade. As bases de dados selecionadas estão listadas na Tabela 1.

Tabela 1 – Base de dados científicas

Base de dados	**Link**
ACM Digital Library	http://dl.acm.org
CiteSeerX Library	http://citeseerx.ist.psu.edu
Google Scholar	https://scholar.google.com.br
IEEE Xplore Digital Library	http://ieeexplore.ieee.org
IET Digital Library	http://digital-library.theiet.org
Science Direct	http://www.sciencedirect.com
Springer Link	http://link.springer.com
Wiley Online Library	http://onlinelibrary.wiley.com

Fonte: Elaborada pelo autor.

Os artigos pesquisados em cada base e dados foram catalogados com o auxílio do aplicativo Mendeley, versão desktop 1.19.4. Para dar sequência a revisão da literatura foram elaboradas as questões de pesquisa enunciadas no item 3.1.1. Foi aplicado também um protocolo de revisão da literatura com critérios de inclusão e exclusão dos artigos, nas seções 3.1.2 e 3.1.3, respectivamente.

3.1.1 Questões de pesquisa

A revisão da literatura tentou responder as seguintes questões de pesquisa no cenário de modelagem colaborativa de software:

- **QP1:** Quais são as técnicas de resolução de conflitos mais usadas na modelagem colaborativa de software atualmente?

- **QP2:** Quais as linguagens de modelagem suportada, ex: Linguagem de Modelagem Unificada (UML) e Unified Method Framework (UMF).
- **QP3:** No caso da UML, quais são os diagramas suportados nos trabalhos de modelagens de software colaborativo?
- **QP4:** Qual a arquitetura que está sendo aplicada na modelagem colaborativa de software (Web or Desktop)?
- **QP5:** Quais recursos são suportados nos softwares de modelagem colaborativa (chat, video, notificações de mudanças, etc)?

3.1.2 Lista de critérios de inclusão (CI)

Para essa revisão da literatura foram elaborados os seguintes critérios de inclusão:

- **CI1:** Estudos publicados ou disseminados em inglês;
- **CI2:** Publicados ou disponíveis em revistas científicas, conferências ou workshops; e
- **CI3:** Estudos publicados até 2019.

3.1.3 Lista de critérios de exclusão (CE)

Para filtrar os estudos, foram elaborados os seguintes critérios de exclusão:

- **CE1:** Estudo que não foram publicados em inglês, patentes, chamadas para trabalhos, literatura cinzenta e materiais não revisados por pares;
- **CE2:** O título não está relacionado as palavras chaves da *string* de pesquisa;
- **CE3:** O resumo não está relacionado as palavras chaves da *string* de pesquisa;
- **CE4:** Estudo repetidos ou atualizados; e
- **CE5:** O conteúdo não está relacionado as questões de pesquisa ou objetivos desse estudo.

3.1.4 Protocolo da revisão da literatura

O protocolo da revisão da literatura foi divido em 6 fases, como descrito a seguir:

- **1. Pesquisa Inicial:** A pesquisa inicial consiste na pesquisa em cada base de dados com o uso dos termos principais e sinônimos conforme Tabela 2. Todos os resultados nessa fase foram registrados no aplicativo Mendeley. Nessa fase aplicam-se os critérios de inclusão CI1, CI2 e CI3.

- **2. Remoção de Impurezas:** A remoção de impurezas consiste na aplicação do critério de exclusão CE1.
- **3. Filtro pelo Título:** Nessa fase é aplicado os critérios de exclusão CE2 e CE5. De forma efetiva, foram eliminados todos os artigos cujo título não registrava nenhum dos termos de pesquisa.
- **4. Remoção de duplicidades:** Essa fase consiste na remoção de artigos duplicados não identificados nas fases anteriores, principalmente usando o recurso do Mendeley: "Checar por Duplicidades". É a efetiva aplicação do critério de exclusão CE4.
- **5. Filtro pelo Resumo:** O filtro pelo resumo consiste na aplicação dos critérios CE3 e CE5. Artigos sem as palavras Collaborative e Modeling no resumo foram eliminados.
- **6. Filtro pelo texto completo:** Fase final com leitura integral dos artigos e reaplicação, quando necessário, dos critérios EC1 e EC4 e EC5. Os resultados foram analisados usando estatística descritiva de frequência em conjunto com o software NVivo Pro®, versão 11.

Tabela 2 – Termos principais e sinônimos

Termos Principais	**Sinônimos**
Collaborative	Cooperative OR Team OR Contribute OR Distributed
Software	Application OR System OR Diagram
Modeling	Development

Fonte: Elaborada pelo autor.

3.1.5 Execução da revisão da literatura

A partir dos termos principais e sinônimos foi elaborado a *string* de pesquisa padrão:

(*Collaborative OR Cooperative OR Team OR Contribute OR Distributed*)
AND (*Software OR Application OR System OR Diagram*)
AND (*Modeling OR Development*)

Para contornar as peculiaridades e limitações de cada plataforma, a *string* acima sofreu alterações, as quais foram registradas no Apêndice B.

Conforme Tabela 3, observa-se que apesar da grande quantidade de resultados (2.272) retornados na pesquisa inicial, após a remoção de impurezas e filtragem pelo título restaram 391 artigos. 3 artigos foram identificados como duplicadas, restando, portanto, 388 artigos para as fases finais de filtragem pelo resumo e pelo texto completo. Dos 388 artigos, apenas 52 foram selecionados para esse trabalho sendo filtrados pelo resumo. Ao longo da leitura integral desses 52 artigos, foi verificado que apenas 18 mantinham relação com esse estudo de acordo com o critério de exclusão EC5, resultando na amostra final de 18 artigos.

Tabela 3 – Filtragem dos estudos

Plataforma	**Pesquisa inicial**	**Remoção de impurezas**	**Filtro pelo título**	**Remoção de duplicidades**	**Filtro pelo resumo**	**Filtro pelo texto completo**
ACM Digital Library	468	440	91			
CiteSeerX Library	51	31	29			
Google Scholar	632	600	49			
IEEE Explore	507	500	117			
IET Digital Library	39	36	6			
Science Direct	272	226	83			
Springer Link	209	197	20			
Wiley Online Library	118	108	18			
TOTAL	2272	2114	391	388	52	18

Fonte: Elaborada pelo autor.

3.1.6 Respostas as questões de pesquisa

3.1.6.1 QP1: Quais as técnicas de resolução de conflitos usadas na modelagem colaborativa de software?

Observa-se na Tabela 4, que 11,11% dos autores aplicaram uma abordagem pessimista, assíncrona, com bloqueio a nível de diagramas ou atributos de objetos sendo alterados. No trabalho de Lucia et al. (2007), foi implementado um sistema de bloqueio amplo, aonde quando um diagrama UML é usado, todos os elementos do diagrama são bloqueados, sendo somente leitura para outros colaboradores. Esse mecanismo de bloqueio a nível de diagrama, costuma causar períodos longos de bloqueio prejudicando o trabalho colaborativo (LIN et al., 2003). Lin at al. (2003) procurou minimizar esses problemas adotando um mecanismo de bloqueio mais refinado, movendo o bloqueio para nível de atributos de objetos, ao invés de objetos inteiros. Como esse mecanismo é possível, por exemplo, a criação de novos atributos por diferentes atores para o mesmo objeto simultaneamente (LIN et al., 2003).

Brosch et al. (2009) tenta evitar os problemas que podem ocorrer quando o *merge* dos modelos em conflito é realizado unilateralmente pelo último autor que fez a submissão das alterações. Na sua proposta, o sistema notifica os autores do conflito e disponibiliza uma janela com a versão local, outra com a versão remota e uma terceira na qual ambos os autores podem alterar o modelo final em tempo real, além de uma janela de chat e outras auxiliares para solução do conflito.

Dam e Ghose (2011) propuseram o uso de Agentes Inteligentes com base no modelo Belief-Desire-Intention (BDI) para solução dos conflitos. Nesse modelo, a implementação do agente reage a eventos (como a alteração do nome de um atributo) e executa planos que consistem na especificação do evento, condição a qual é aplicável e a sequência de ações a serem realizadas, inclusive podendo incluir o disparo de outros eventos e planos em cascada.

Tabela 4 – Métodos de resolução de conflitos

Método de Resolução de Conflitos	**Nº de Estudos**	**Perc.**	**Referência**
Abordagem Pessimista Assíncrona através Bloqueio	2	11,11%	Lin et al. (2003); De Lucia et al. (2007)
Abordagem Otimista Assíncrona através de *Merge*	1	5,56%	Brosch et al. (2009)
Agentes inteligentes modelo Belief-Desire-Intention	1	5,56%	Dam e Ghose (2011)
Abordagem Otimista Assíncrona através de Serialização	1	5,56%	Shen et al. (2008)
Abordagem Otimista Síncrona ou em Tempo Real	3	16,65%	Kuryazov e Winter (2015); Liu et al. (2006); Xu et al. (2009)
Ponto de Extensão	1	5,56%	Bang et al. (2010)
Não identificado	9	50,00%	Bang, Brun e Medvidović (2018); Jolak et al. (2018); Perez-Soler, Guerra e Lara (2018); Farias et al. (2015); Krusche e Bruegge (2014); Bang e Popescu (2012); Cicchetti et al. (2009); Zhu et al. (2007); Boulila (2004)

Fonte: Elaborada pelo autor.

Shen (2008) adota uma abordagem pessimista assíncrona com serialização na qual o o modelo é replicado em cada cliente e na ocorrência de conflitos ou violação de restrições, somente a operação com maior prioridade é mantida.

Verifica-se também na Tabela 4 que 16,65% dos autores adotaram uma abordagem otimista síncrona ou em tempo real para evitar conflitos. Kuryazov e Winter (2015) implementou pequenos deltas de modelagem para transferir as diferenças de um modelo para outro de forma eficiente. Cada cliente envia suas mudanças para o servidor que, por sua vez envia para ser aplicado nos outros clientes. É aplicado a técnica First Come First Serve, isto a primeira mudança recebida é aquela aplicada. Com esse modelo o autor alega que os conflitos são raros devido a rápida sincronização, não tento ele identificado nenhum conflito durante seus experimentos. Liu et al. (2006) usou a técnica de Transparent Adaptation (TA) para tornar o aplicativo Rational Software Architect colaborativo em tempo real, sem alteração do código do aplicativo. Xu et al. (2009) implantou um sistema de modelagem colaborativa síncrona com recursos de conversão associados aos objetos da modelagem.

Bang et al. (2010) ao invés de propor um método de resolução de conflitos, cria um arquitetura para permitir as equipes usar seus próprios mecanismos de detecção de conflitos através de pontos de extensão.

Nos demais trabalhos ou não é aplicável ou não foi identificado com clareza o método de resolução de conflito proposto.

3.1.6.2 QP2: Qual a linguagem de modelagem suportada?

Em relação a questão QP2, observa-se que na Tabela 5, 94,44% dos trabalhos usaram a UML para seus protótipos ou como exemplo de suas soluções para os problemas de modelagem de forma colaborativa. Em 5,56% nos trabalhos não era aplicável ou não foi possível identificar o uso de uma linguagem específica.

Tabela 5 – Linguagem de modelagem

Linguagem de Modelagem	**Nº de Estudos**	**Perc.**	**Referência**
UML	17	94,44%	Bang, Brun e Medvidović (2018); Jolak et al. (2018); Perez-Soler, Guerra e Lara (2018); Farias et al. (2015); Kuryazov e Winter (2015); Krusche e Bruegge (2014); Dam e Ghose (2011); Bang et al. (2010); Brosch et al. (2009); Cicchetti et al. (2009); Xu et al. (2009); Shen et al. (2008); De Lucia et al. (2007); Zhu et al. (2007); Liu et al. (2006); Boulila (2004); Lin et al. (2003)
Não identificada	1	5,56%	Bang e Popescu (2012)

Fonte: Elaborada pelo autor.

3.1.6.3 QP3: Quais diagramas da UML são suportados?

Tabela 6 – Diagrama UML

Diagrama UML	**Nº de Estudos**	**Perc.**	**Referência**
Casos de Uso	1	5,56%	De Lucia et al. (2007)
Classes	16	94,44%	Bang, Brun e Medvidović (2018); Jolak et al. (2018); Perez-Soler, Guerra e Lara (2018); Farias et al. (2015); Kuryazov e Winter (2015); Krusche e Bruegge (2014); Dam e Ghose (2011); Brosch et al. (2009); Cicchetti et al. (2009); Xu et al. (2009); Shen et al. (2008); De Lucia et al. (2007); Zhu et al. (2007); Liu et al. (2006); Boulila (2004); Lin et al. (2003)
Sequência	1	5,56%	Dam e Ghose (2011)
Não identificado	2	11,11%	Bang et al. (2010); Bang e Popescu (2012)

Fonte: Elaborada pelo autor.

No que tange a questão QP3, observa-se na Tabela 6 que 94,44% dos trabalhos exemplificaram suas soluções com o diagrama de classes, 5,56% com o diagrama de caso de uso e 5,56% com o diagrama de sequência. 11,11%, apesar da maioria usar a UML, não usaram nenhum de seus diagramas nos exemplos.

3.1.6.4 QP4: Qual a arquitetura das aplicações?

Observa-se na Tabela 7 que 50,00% dos trabalhos adotaram uma arquitetura cliente servidor, com o cliente sendo uma aplicação desktop. Apenas um trabalho adotou a arquitetura web com o cliente sendo o navegador. Nos demais trabalhos essa questão de pesquisa não era aplicável ou não foi possível identificar a arquitetura.

Tabela 7 – Arquitetura das aplicações na revisão da literatura

Arquitetura	**Nº de Estudos**	**Percentual**	**Referência**
Web	1	5,56%	De Lucia et al. (2007)
Desktop	9	50,00%	Kuryazov e Winter (2015); Dam e Ghose (2011); Bang et al. (2010); Brosch et al. (2009); Xu et al. (2009); Shen et al. (2008); Liu et al. (2006); Boulila (2004); Lin et al. (2003)
Não identificado	8	44,44%	Bang, Brun e Medvidović (2018); Jolak et al. (2018); Perez-Soler, Guerra e Lara (2018); Farias et al. (2015); Krusche e Bruegge (2014); Bang e Popescu (2012); Cicchetti et al. (2009); Zhu et al. (2007)

Fonte: Elaborada pelo autor.

3.1.6.5 QP5: Quais recursos são suportados nos softwares de modelagem colaborativa?

Tabela 8 – Recursos suportados das aplicações na revisão da literatura

Recursos	**Nº de Estudos**	**Perc.**	**Referência**
Notificações por E-mail, Mensagens ou Interface	5	27.78%	Bang et al. (2010); Dam e Ghose (2011); Kuryazov e Winter (2015); Lin et al. (2003); De Lucia et al. (2007)
Chat	5	27,78%	Perez-Soler, Guerra e Lara (2018); Boulila (2004); Brosch et al. (2009); Lin et al. (2003); Xu et al. (2009)
Vídeo	1	5,56%	Lin et al. (2003)

Fonte: Elaborada pelo autor.

Observa-se que na Tabela 8 que 27,78% dos trabalhos implementaram alguma forma de notificar cada usuário de alterações de outros usuários em seus modelos. 27,78% procuram melhorar a comunicação com recursos de conversação. Apenas 5,56% chegaram a usar recursos de vídeo para melhorar a comunicação.

3.2 Trabalhos relacionados com colaboração síncrona ou assíncrona

Nessa seção é apresentdo um resumo de cada trabalho que aplicaram técnicas híbridas usando colaboração síncrona e colaboração assíncrona ou técnicas relevantes para modelagem colaborativa de software e a análise comparativa dos trabalhos selecionados.

3.2.1 Análise dos artigos selecionados

Bang, Brun e Medvidović (2018) investigam o problema de conflitos que ocorrem quando vários desenvolvedores trabalham no mesmo modelo ao mesmo tempo. Nesta pesquisa os autores identificam requisitos para ferramentas de detecção de conflitos, focam na necessidade de detecção proativa e precoce de conflitos e introduzem o "FLAME"(Framework para registro e análise de eventos de modelagem).

Jolak et al. (2018) se baseia no artigo clássico de 2000 “Distance Matters” de Gary Olson e Judith Olson. O artigo tenta responder a seguinte questão de pesquisa: a distância ainda importa? Para responder essa questão de pesquisa, Jolak e os demais autores elaboraram um experimento que analisa como a distância afeta as decisões de design, a comunicação colaborativa e os desafios técnicos e sociais. Os principais resultados apontam que a distância ainda é importante, principalmente por causa da falta de consciência social e confiança.

Perez-Soler, Guerra e Lara (2018) realiza um experimento que explora uma forma diferente para os desenvolvedores colaborarem na construção de modelos. Os autores elaboraram a ferramenta chamada "SOCIO"que é um chatbot experimental que interpreta frases específicas, ouve as conversas dos desenvolvedores, constrói automaticamente uma representação em diagrama das definições que estão sendo discutidas e possibilita a chegada de consensos sobre o modelo debatido pelos desenvolvedores.

Farias et al. (2015) elaborou uma avaliação sobre a evolução de diagramas UML, com o intuito de inserir novos recursos ou modelos de reconciliação desenvolvidos de forma distribuída por diversas equipes de desenvolvimento de software. Os resultados principais sugerem que as técnicas heurísticas exigem menos esforço para produzir o modelo desejado. Em resumo, o artigo explana um experimento que investiga o esforço para aplicar técnicas de composição de modelos e para detectar e resolver inconsistências nos modelos compostos.

Krusche e Bruegge (2014) apresenta um abordagem baseada na colaboração síncrona e ponto a ponto (P2P), ou seja, com sincronização em tempo real de modelos colaborativos. Os autores realizaram uma extensão do *framework* "EMF-Store"com suporte a vários colaboradores conectados diretamente via mecanismos "peer-to-peer"para sincronização dos modelos entre os colaboradores para permitir o trabalho colaborativo em um único modelo.

A proposta de Brosch et al. (2009) ganha destaque por apresentar *merge* colaborativo para evitar que esta tarefa seja realizada por um único usuário, o quê normalmente é uma atividade demorada e sujeita a erros. A alteração dos modelos usa colaboração assíncrona, pois, cada

desenvolvedor trabalha na sua cópia dos modelos de forma isolada, alterando e as submetendo a um Sistema de Controle de Versão - SCV otimizado. Para resolver conflitos, esta abordagem adota uma colaboração síncrona, notificando os desenvolvedores envolvidos no conflito e permitindo que os mesmos trabalhem ao mesmo tempo no modelo com *merge* das alterações.

Cicchetti et al. (2009) apresenta um modelo com colaboração síncrona para compartilhamento do espaço de modelagem, modelos, documentação e configurações e assíncrona para *merge* de modelos alterados isoladamente por cada engenheiro de software. O estudo foca na descrição de um gerenciador de versões, um ótimizador para armazenamento e recuperação das versão e um gerenciador de conflitos.

Zhu et al. (2007) justifica sua pesquisa informando a necessidade de construção de ferramentas visuais de linguagem de domínio específico com várias visões de informações, suporte multiusuário, dentre outras. Os autores produziram o que foi chamado de meta-ferramenta a "Pounamu", para realizar ambientes de design visual. Nos principais resultados foram descritos a arquitetura e implementação de exemplos de linguagem visual específicas de domínio.

Na revisão da literatura, a maioria dos trabalhos aplicaram a abordagens de resolução de conflitos otimista, síncrona ou tempo real com recursos de notificações das alterações e chat. A resolução de conflitos indiretos ou semânticos e problemas de escalabilidade permanecem como desafios a serem superados.

3.2.2 Análise comparativa dos trabalhos

Nessa seção é apresentado uma comparação dos trabalhos relacionados para identificar os pontos em comum e divergentes da proposta apresentada e da literatura selecionada. Abaixo são apresentados os critérios de comparação dos trabalhos selecionados em relação a abordagem UMLCollab.

- **Estudo empírico (C1):** Estudos empíricos como experimentos controlados, casos de estudo, *survey*, entre outros.
- **Principais contribuições (C2):** O estudo têm como principal contribuição a modelagem colaborativa de software, utilizando linguagem UML.
- **Abordagem proposta (C3):** O estudo apresenta novas abordagens de *merge* ou resolução de conflitos ou redução do esforço ou corretude dos modelos gerados.
- **Interação (C4):** Existe interação na elaboração de modelos ou na resolução de conflitos.
- ***Merge* automático (C5):** O *merge* automático consiste no sistema aplicar automaticamente as mudanças remotas no modelo local quando não há conflitos.
- **Notificação dos conflitos (C6):** Estudos que identificaram que o conflito foi notificado para aos colaboradores direto no modelo sendo elaborado, na forma de mensagens no sistema, entre outros.

- **Esforço na elaboração dos modelos (C7):** Identifica se o esforço é medido na elaboração dos modelos na abordagem proposta.
- **Conflitos na elaboração dos modelos (C8):** Identifica se os conflitos são medidos na elaboração dos modelos na abordagem proposta.
- **Corretude dos modelos elaborados (C9):** Identifica se o nível de corretude dos modelos gerados é medido na elaboração dos modelos na abordagem proposta.

A Tabela 9 apresenta a comparação considerando todos os critérios citados acima. Nessa comparação, é possível verificar o atendimento pleno do trabalho proposto em todos os critérios definidos, evidenciando a contribuição deste trabalho.

Tabela 9 – Comparativo dos trabalhos relacionados

	Comparação dos critérios								
Trabalhos relacionados	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**
UMLCollab	●	●	●	●	●	●	●	●	●
Bang, Brun e Medvidović (2018)	○	○	●	○	○	●	○	○	○
Jolak et al. (2018)	●	●	●	●	○	○	○	○	○
Perez-Soler, Guerra e Lara (2018)	●	○	○	●	○	○	○	○	○
Farias et al. (2015)	●	●	●	○	○	○	●	○	●
Krusche e Bruegge (2014)	○	●	●	●	○	●	○	○	○
Brosch et al. (2009)	○	●	●	●	○	●	●	○	○
Cicchetti et al. (2009)	○	●	○	●	○	○	○	○	○
Zhu et al. (2007)	●	●	○	●	●	○	○	○	○

Legenda: ● = atende; ○ = não atende.

Fonte: Elaborada pelo autor.

Na UMLCollab o conflito é destacado direto no elemento na cor vermelha de forma discreta para não inteferir no trabalho do colaborador. Para realizar o merge, na UMLCollab a aceitação e rejeição de alterações é realizada direto no modelo. Nessa abordagem, todo o destaque é feito em cores, diminuindo a necessidade de interações por outras formas de comunicação. Na pesquisa de Brosch et al. (2009), para realizar o merge, exibe modelo consolidado em uma janela para aprovação e *commit*. Já Cicchetti et al. (2009) estabelece uma abordagem síncrona para evitar conflitos, com a aplicação da última alteração. Krusche e Bruegge (2014) evita os conflitos e *merge* através de bloqueio do elemento sendo alterado.

4 ABORDAGEM PROPOSTA

Neste capítulo é apresentada a UMLCollab, uma abordagem híbrida de modelagem colaborativa de modelos UML criada para melhorar o nível de colaboração evitando inteferências no processo cognitivo dos desenvolvedores na elaboração dos modelos e reduzindo as complicadas e custosas etapas de resolução de conflitos.

A UMLCollab permite que cada usuário receba atualizações síncronas de outros usuários e envie as suas alterações de forma assíncrona, combinadas com a técnica de *merge* automático e manual.

A estrutura deste capítulo foi organizada da seguinte forma: A Seção 4.1 lista requisitos para ferramentas de modelagem colaborativa de software que servirão de base para proposta apresentada. A Seção 4.2 oferece uma visão geral da UMLCollab. A Seção 4.3 compara o UMLCollab com os tipos tradicionais de colaboração, síncrona e assíncrona. A Seção 4.4 detalha o *merge* automático e resolução de conflitos da proposta. A Seção 4.5 resume as principais características do UMLCollab. A Seção 4.6 detalha arquitetura proposta e finalmente a Seção 4.7 comenta detalhes relacionados à implementação.

4.1 Requisitos para ferramentas de modelagem colaborativa de software

A modelagem colaborativa de software requer mais que somente compartilhar arquivos dos modelos em um repositório online. É necessário que as ferramentas providenciem recursos para que as equipes de desenvolvimento trabalhem de forma mais eficiente. Para isso, foi utilizado os 10 requerimentos para colaboração com base no estudo de Dullemond, Gameren e Solingen (2014) ilustrados na tabela A do referido estudo.

Está ciente do quê está ocorrendo em volta em um ambiente distribuído é uma tarefa difícil, porque toda informação precisar ser manualmente processada em comparação ao espaço de trabalho local. Pensando nisso, os três primeiros requerimentos foram elaborados com o foco em permitir a troca de informações sobre o ambiente de forma não intrusiva, isto é, sem distrair o desenvolvedor e sem atrapalhar suas atividades:

- **R01:** Rastrear alterações realizadas pelos desenvolvedores. Identificar o autor das alterações em um projeto e manter o histórico de cada alteração. Na prática, realizar este procedimento seria implementar um sistema automático de controle de versão, que é muito útil em um ambiente colaborativo, seja para comparar modelos, realizar *merge* e reverter alterações quando necessário. Na UMLCollab é contemplado o histórico das alterações para permitir a consulta dos *merges* automáticos realizados e da aceitação ou rejeição de alterações remotas conflitantes com alterações locais.

- **R02:** Notificar usuários sobre alterações em tempo real. Semelhante aos conflitos de código, conflitos nos modelos são difíceis de serem detectados e resolvidos. A literatura

corrente ainda considera que a detecção e resolução de tais conflitos seja uma tarefa sujeita a erros e que consomem muito tempo (BANG et al, 2010). A colaboração síncrona ou em tempo real ajuda a evitar conflitos, pois cada usuário é notificado e recebe as mudanças dos demais usuários em tempo real. Na UMLCollab é mantido a notificação dos usuários a cada alteração remota recebida de forma discreta através de destaque em cores.

- **R03:** Identificar se um membro do projeto pode ser interrompido. Saber o que o usuário está fazendo (R06) e seu humor (R07) ajuda o sistema a decidir, ou ser configurado para saber quando um membro do projeto pode ser interrompido, para que os usuários possam receber informações relevantes no tempo certo.

Receber atualizações relacionadas ao projeto que o usuário está trabalhando é importante, mas informações fora do contexto do projeto são valiosas. Os próximos dois requerimentos objetivam tornar dados básicos relacionados ao trabalho mais acessíveis:

- **R04:** Notificar usuários sobre mudanças enquanto se encontra fora do sistema. Este requerimento pode ajudar os usuários identificar as alterações que estão sendo realizadas no projeto enquanto eles estão fora do sistema, alterações que possivelmente podem impactar no trabalho deles. Essa aplicação pode ser feita através de e-mail, SMS ou notificações de smartphones, ou seja, o desenvolvedor iria receber um resumo das alterações ou as notificações consideradas importantes.

- **R05:** Notificar usuários sobre atualizações da organização. Apesar de atualizações sobre o projeto serem importantes para cada membro da equipe, atualizações da organização são úteis, ainda que não relacionadas ao projeto que o usuário participa (DULLEMOND; GAMEREN; SOLINGEN, 2014). A semelhança do requisito R04, isto pode ser alcançado por e-mail, SMS ou notificações de smartphones.

Os seguintes cinco requerimentos a seguir, expandem os recursos de colaboração par tornar os desenvolvedores de software mais efetivos.

- **R06:** Saber o que o usuário está fazendo. É útil que o sistema saiba se o usuário está revisando código, escrevendo e-mail ou atendendo ao telefone. Além disso, é importante saber se o usuário está fazendo está relacionado ao trabalho. Estas informações são úteis para saber quando o usuário pode ser interrompido.

- **R07:** Identificar o humor do usuário. Esse requisito pode ajudar a conectar pessoas em um nível social (ALHO; SULONEN, 1998) e ser definido manualmente pelo usuário, como é feito nas mídias sociais. Outra forma é o reconhecimento de imagens através de técnicas de inteligência artificial usando a webcam do usuário, o que apesar de possuir suas próprias limitações, pode ser mais precisa que a configuração manual pelo usuário.

- **R08:** Filtrar atualizações de informações do usuário. Com base nos requisitos R03, R06 e R07, o sistema pode filtrar as atualizações de informações por categoria e atividade atual do usuário e humor, evitando assim sobrecarregar o usuário com informações irrelevantes. As informações filtradas podem ser definidas automaticamente ou configuráveis pelo usuário como filtros presentes nos emails. Na UMLCollab se a alteração remota não é conflitante, ela é imediatamente aplicada, sem atrapalhar o usuário e apenas indicando na cor verde o *merge* automático. Se ocorre o conflito, o mesmo é destacado na cor vermelha. Se o conflito deixa de existir pela modificação do elemento remoto, a UMLCollab deixa de indicar o conflito. Dessa forma é realizado o filtro das informações que o usuário recebe.

- **R09:** Suporte a grupos abertos de conversas. O objetivo principal é permitir que qualquer um inicie um grupo, entre, pesquise por tópicos estáticos ou dinâmicos definidos, apenas escute, feche e convide outros. O sistema pode deixar automaticamente o usuário escutando uma conversa se o tópico lhe interessar ou puder contribuir com o projeto. Embora existam aplicativos de vídeo e bate-papo no mercado, como o Skype for Business, para atender a esse requisito, esses aplicativos não têm recursos automáticos para convidar o usuário a participar de uma conversa de seu interesse e pesquisa por tópicos estáticos ou dinâmicos.

- **R10:** Integração com outras ferramentas. Integração de sistemas com outras ferramentas como controle de versão de código, gerenciamento de mudanças (erros, melhorias, tarefas), gerenciamento de projetos, cronograma e wiki podem melhorar a colaboração e aumentar a produção da equipe.

4.2 Visão geral da proposta

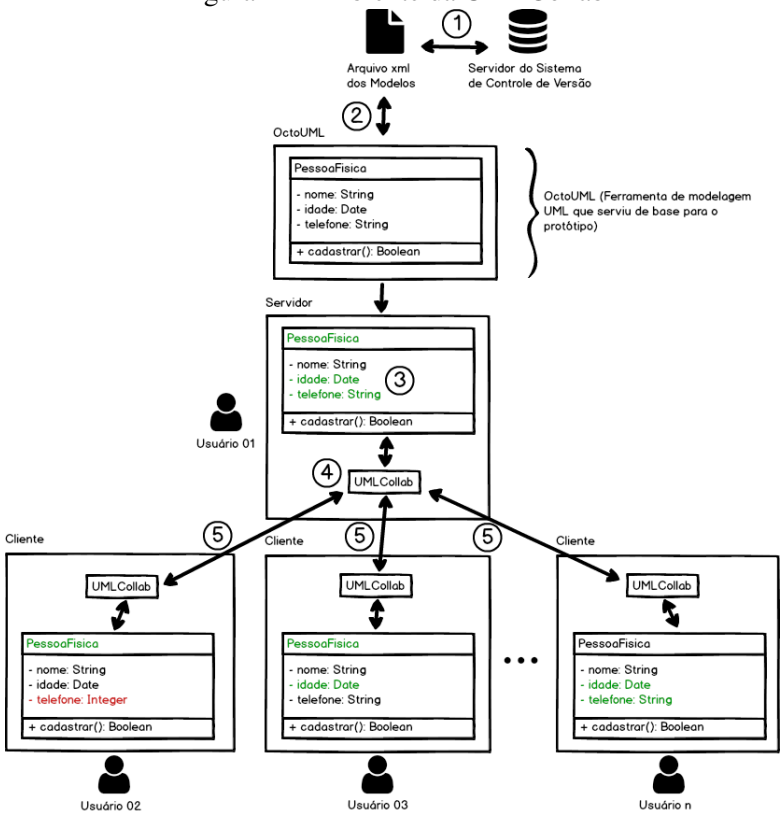
A Figura 4 apresenta uma visão global do ambiente de utilização da UMLCollab:

Para facilitar a compreensão da Figura 4, a UMLCollab foi ilustrada em 5 itens descritos a seguir:

- **1:** Inicialmente é exemplificado a atualização do arquivo xml dos modelos a partir de um SCV pelo usuário 01. Mais tarde, após as atividades de modelagem colaborativa, esse mesmo usuário pode realizar a submissão das alterações para o SCV. Os sistemas de controle de versão (SCV) assumem grande importância no registro das alterações dos códigos de sistemas, como também no registro das versões dos modelos. Na UMLCollab o foco está na diminuição do tempo para resolução de conflitos sendo para trabalhos futuros a integração com SCVs.

- **2:** O arquivo xml dos modelos é carregado pelo usuário 01 na ferramenta de modelagem UML OctoUML que serviu de base para o protótipo.

Figura 4 – Ambiente da UMLCollab



Fonte: Elaborada pelo autor

- **3:** Nesse momento o usuário ativa o modo servidor da ferramenta para que outros usuários colaborem na elaboração do modelo carregado na memória do servidor. Nesse exemplo, o diagrama de classes desse modelo contém as alterações recebidas de todos os clientes.

- **4:** Neste contexto, o componente UMLCollab intercepta os pedidos remotos de atualização do modelo local de forma síncrona e interage com a interface da ferramenta de modelagem UML para realizar *merge* automático ou indicar conflitos. Esse mesmo componente registra as alterações locais antes repassá-las aos demais colaboradores de forma assíncrona, isto é, quando indicado pelo usuário.

- **5:** Por fim, ocorre a troca das versões do modelo entre um usuário rodando a ferramenta de modelagem UML no modo servidor e vários usuários rodando no modo cliente. As alterações recebidas na ferramenta de modelagem no modo servidor são imediatamente enviadas a todos os clientes. Na próxima seção é detalhado a sincronização para o envio e recebimento das alterações locais e remotas pela UMLCollab.

4.3 UMLCollab e a colaboração síncrona e assíncrona

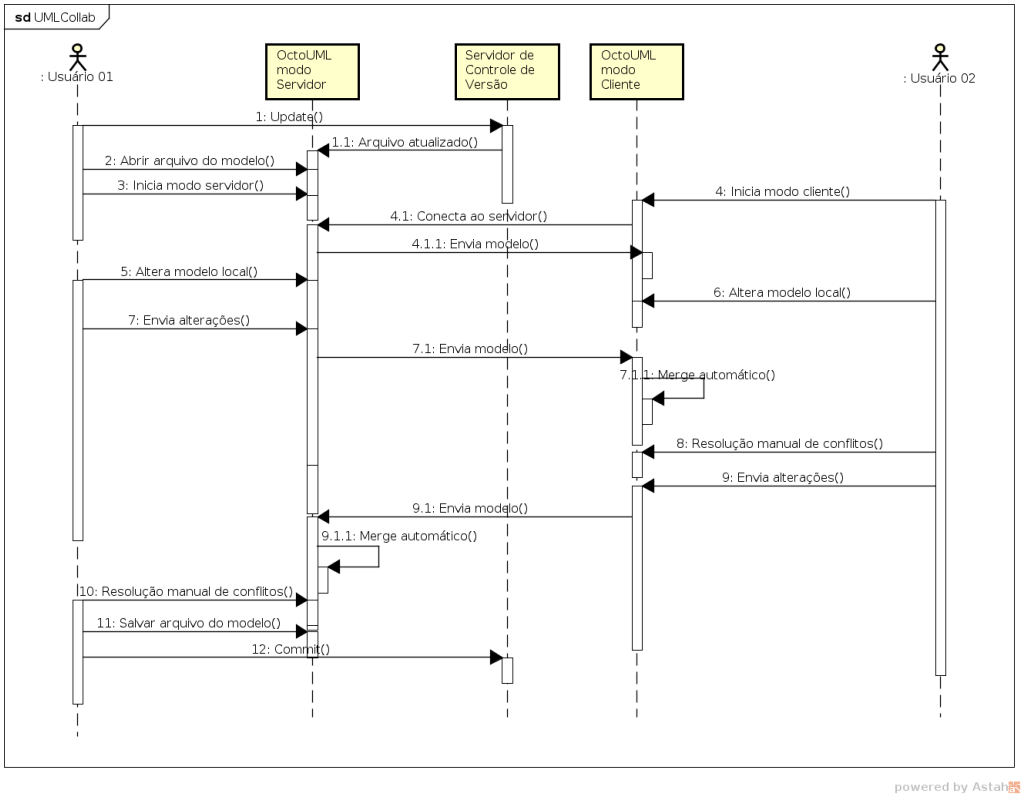
A colaboração síncrona apesar de evitar conflitos, não permite que o usuário possa livremente alterar o modelo enquanto completa o processo mental para modificar o modelo de forma

completa sem interferir no trabalho de outros desenvolvedores. De fato, apesar desse tipo de colaboração atender o requisito R02 e notificar usuários sobre alterações em tempo real, entra em conflito com o requisito R03 que buscar identificar se o desenvolvedor pode ser interrompido.

Em um cenário de apenas 2 ou 3 usuários, talvez isso não seja um problema, porém em ambientes de alta concorrência com mais de 3 usuários é provável que fique prejudicado qualquer interação produtiva para elaboração dos modelos.

A colaboração assíncrona permite que o desenvolvedor trabalhe de forma isolada, porém perde-se os benefícios da colaboração dos outros desenvolvedores pois, só será possível compartilhar as alterações nos modelos no momento que o usuário faz um update ou commit para o sistema de controle de versão. Este tipo de colaboração atende o requisito R03 pois o usuário tem garantido seu ambiente de trabalho sem interferências. Ele escolhe quando quer enviar suas alterações e receber alterações de outros usuários, porém como não atende o requisito R01 de receber notificações em tempo real, quando mais tempo demora para realizar update ou commit maior a probabilidade de ocorrência de conflitos e maior o esforço é gasto na solução dos mesmos.

Figura 5 – UMLCollab versus colaboração síncrona e assíncrona



Fonte: Elaborada pelo autor.

Fica claro que a colaboração síncrona e a colaboração assíncrona atendem ou o requisito R02 de notificar usuários sobre alterações em tempo real ou o requisito R03 de buscar identificar se o desenvolvedor pode ser interrompido, cada tipo apresentando suas vantagens e desvantagens. A Figura 5 apresenta uma visão da UMLCollab em comparação a colaboração síncrona

e a colaboração assíncrona. Os passos que vão de 1 ao 4 são idênticos aos descritos na colaboração síncrona da Figura 2. Logo, os passos restantes são descritos a seguir.

Na UMLCollab a alterações dos usuários, conforme os passos 5 e 6, não são imediatamente propagadas aos demais usuários como na colaboração síncrona. O comportamento padrão é que o usuário indique quando quer enviar suas alterações para evitar o problema de vários usuários enviando suas alterações parciais de forma síncrona e atrapalhando o processo cognitivo na elaboração dos modelos dos demais usuários.

O passo 7 demostra a indicação do usuário para o envio das alterações. Nesse momento, o sistema envia o modelo completo para todos os demais usuários utilizando o sistema. Cada sistema realiza *merge* automático das alterações que não resultarem em conflitos, indicando quem, o que foi alterado e aonde foi realizada as alterações para que o usuário possa revisar-las. Os detalhes sobre o *merge* automático são descritos na Seção 4.4. Havendo conflitos, os mesmos são destacados para que o usuário possa revisá-las, conforme descrito na Seção 4.4.

Tabela 10 – Comparativo dos tipos de colaboração

Características	**Síncrona**	**Assíncrona**	**Híbrido (UMLCollab)**
Envio de alterações	em tempo real	manual	manual
Recebimento de alterações	em tempo real	manual	em tempo real
Observações	Sem necessidade de intervenção do usuário.	Envio manual das alterações locais para outros usuários, ocasião na qual recebe as alterações remotas.	No cenário híbrido o sistema de comporta de forma assíncrona no envio de alterações e síncrona no recebimento de alterações.

Fonte: Elaborada pelo autor.

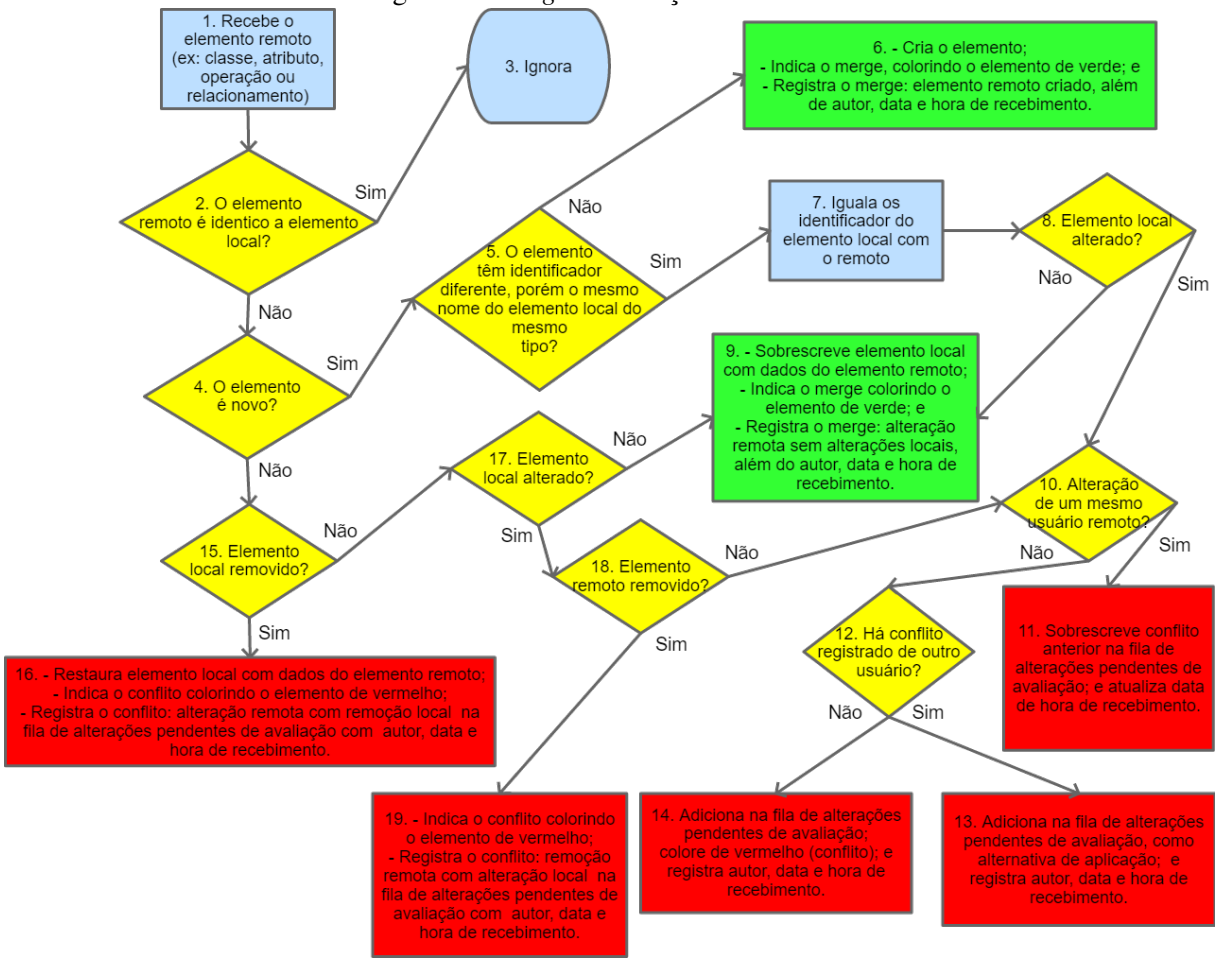
A Tabela 10 apresenta um comparativo dos tipos de colaboração demostrando como ocorre o envio e o recebimento das alterações.

4.4 *Merge* automático e resolução de conflitos

Quando uma instância da ferramenta de modelagem UML recebe o modelo alterado por outros usuários, a UMLCollab compara cada elemento remoto (classe, atributos, operações e relacionamentos) com o modelo local para identificar quais e que tipos de alterações foram realizadas. Para alterações que não conflitem, estas são imediatamente aplicadas e destacadas na cor verde para que o usuário se beneficie imediatamente do trabalho colaborativo com as alterações completas de outros usuários. Para as alterações que gerem conflitos, é feito destaque na cor vermelha e realizado o tratamento conforme Figura 6 com os passos descritos a seguir.

- **Passo 1:** A instância local da ferramenta de modelagem UML recebe as alterações realizadas pelos usuários remotos a partir de suas respectivas versões base do modelo. Cada

Figura 6 – *Merge* e resolução de conflitos



Legenda: Azul = Processo, Amarelo = Decisão; Verde = Merge automático; Vermelho = Tratamento de conflitos.

Fonte: Elaborada pelo autor.

instância local da ferramenta de modelagem UML contém uma versão distinta do modelo com as alterações locais, alterações remotas feitas por *merge* automático e alterações remotas em conflito com as alterações locais, já aprovadas, rejeitadas ou ainda aguardando avaliação. A UMLCollab inicia a análise de cada elemento remoto em comparação ao elemento local e depois continua no passo seguinte (passo 2).

- **Passos 2 e 3:** Caso o elemento remoto seja igual ao elemento local, o elemento remoto é ignorado (passo 3) e nenhuma ação é realizada. Caso seja uma nova versão, a UMLCollab continua no passo seguinte (passo 4).

- **Passo 4:** A UMLCollab verifica se o elemento é novo, isto é, não existe localmente. Caso sim, segue o passo 5. Caso contrário, é realizado o passo 15.

- **Passos 5, 6 e 7:** Mesmo um elemento remoto novo, pode ter o mesmo nome de um elemento local existente. Nesse contexto, ambos terão identificadores distintos. Caso os nomes sejam idênticos, a UMLCollab reconhece que se trata do mesmo elemento e iguala os identificadores (passo 7) e segue para o passo seguinte (passo 8). Caso contrário, trata-se de fato de um elemento totalmente novo. Elementos novos são considerados não conflitantes, logo é realizado um *merge* automático com integração do elemento ao modelo (passo 6). Para dar ciência ao usuário do *merge* realizado automaticamente, a UMLCollab destaca o elemento com a cor verde e ainda registra o nome do autor, data e hora de recebimento.

- **Passos 8 e 9:** Se o elemento local não sofreu alterações, o elemento remoto não é considerado conflitante com o elemento local, logo é realizado *merge* automático com sobrescrita do elemento local com os dados do elemento remoto (passo 9). Caso haja alteração local, existe conflito que é tratado a partir do passo 10.

- **Passos 10 e 11:** Ainda que o usuário local não tenha aprovado ou rejeitado uma alteração remota em conflito com alteração local, é perfeitamente possível que o usuário remoto autor da primeira alteração, em outro momento, altere novamente o mesmo elemento e envie essa nova alteração. Nesse caso, a UMLCollab considera somente a última alteração pois reflete a decisão mais recente do usuário remoto que a enviou. Nesse contexto, o conflito anteriormente registrado localmente é descartado e substituído pelo registro do novo conflito (passo 11). Devido a característica dinâmica do trabalho colaborativo, descartar conflitos antigos automaticamente é um comportamento interessante para diminuir a necessidade de intervenção do usuário para resolver conflitos. Caso do autor da alteração remota não tenha enviado anteriormente alterações que geraram um conflito, segue-se para o passo 12.

- **Passos 12, 13 e 14:** Enquanto um determinado elemento recebido continua pendente de avaliação, pode ocorrer que outro usuário remoto altere o mesmo elemento e envie suas

alterações. Nesse contexto, ocorre conflito da alteração local com n alterações recebidas de usuários remotos. Nesse caso o usuário que as recebeu deve poder decidir qual alteração vai efetivar em detrimento das outras. Portanto, se para o mesmo elemento local há conflitos provenientes de usuários remotos diferentes, a UMLCollab adiciona a nova alteração como alternativa de resolução do conflito na fila de alterações pendentes de aplicação do elemento (passo 13), caso contrário adiciona a alteração na fila de alterações pendentes de aplicação do elemento (passo 14).

- **Passos 15 e 16:** Se o elemento remoto não é novo e o elemento local foi excluído, a UMLCollab sinaliza esse caso especial de conflito restaurando o elemento local com dados do elemento remoto (passo 16). No caso de aprovação da alteração remota, o elemento local restaurado com dados do elemento remoto é mantido, caso rejeitado o elemento local é novamente excluído. Se o elemento local não foi excluído, o sistema continua no passo 17.

- **Passo 17:** Se o elemento local não foi alterado, não há conflito, logo é realizado o *merge* automático previsto no passo 9. Caso contrário, segue o passo 18.

- **Passos 18 e 19:** Se o elemento remoto foi excluído, há outro caso especial de conflito. Se houvesse a remoção imediata do elemento local, o usuário não poderia validar se a exclusão é devida, logo nesse caso o sistema indica o conflito (passo 19). Se aprovado a alteração remota, o elemento local é excluído, caso rejeitado o elemento é mantido. Caso o elemento remoto não tenha sido excluído, a UMLCollab continua a partir do passo 10 descrito anteriormente.

4.5 Principais características da UMLCollab

Dos dez requisitos listados na literatura (DULLEMOND; GAMEREN; SOLINGEN, 2014), três foram explorados com mais profundidade, a saber: R01 (Rastrear alterações realizadas pelos desenvolvedores), R02 (Notificar usuários sobre alterações) e R08 (filtrar atualizações de informações do usuário). A seguir observa-se como esses requisitos foram abordados pelo UMLCollab listando suas principais características:

4.5.1 Filtro de atualizações de informações para o usuário

O filtro de atualizações de informações para o usuário, requisito R08, pode ser avaliado como o principal benefício da UMLCollab, pois como o usuário só recebe alterações consideradas completas ou suficientemente prontas de outros usuários, possibilita um menor fluxo de alterações a serem recebidas, independentemente da quantidade de usuários simultâneos atuando no modelo. Desse fluxo reduzido, aquelas alterações que não conflitam, são imediatamente aplicadas, livrando o usuário da obrigação de avaliá-las, porém permitindo que as revise

se desejar.

4.5.2 Notificação dos usuários de alterações

Em conformidade com o requisito R02, na UMLCollab o usuário é notificado sobre todas as alterações recebidas sem que elas tirem a sua atenção em excesso dos modelos que está elaborando. Isso é realizado de forma discreta através de um esquema de cores. A cor verde é utilizada para indicar *merge* automático quando não ocorre conflitos e a cor vermelha é usada para indicar conflitos devido a alterações recebidas que necessitam de aprovação ou rejeição do usuário.

4.5.3 Rastreamento de alterações realizadas pelos usuários

Todas as alterações recebidas são rastreáveis conforme requisito R01 com o nome do autor, data e hora de recebimento. Todo esse histórico fica disponível tanto para alterações pendentes de avaliação com aqueles já aprovadas ou rejeitadas. Na UMLCollab não foi implementado o requisito R10 no que diz respeito e integração com sistemas de controle versão pois o foco foi delimitado para aumentar a produtividade da modelagem colaborativa. A recuperação e o salvamento posterior do arquivo contento os modelos é realizado manualmente pelo usuário que rodar o OctoUML em modo servidor.

4.5.4 *Merge* automático

O *merge* automático conforme descrito anteriormente visa minimizar a necessidade de interação do usuário nas mudanças recebidas. As alterações que não geram conflitos são imediatamente aplicadas. Com isso foca sua atenção nas alterações que está elaborando e naquelas enviadas pelos usuários que conflitam com suas alterações.

4.5.5 Aprovação e rejeição de alterações

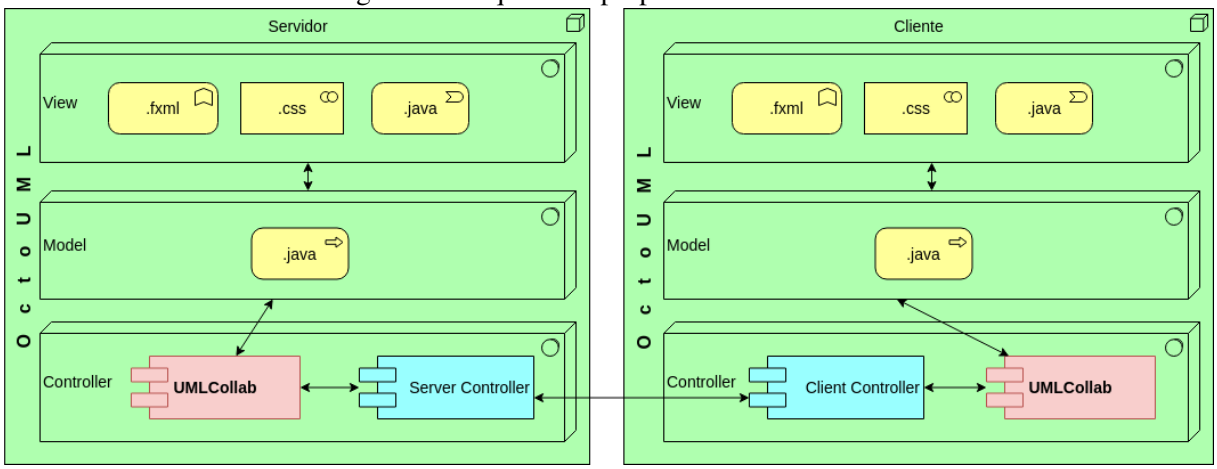
A UMLCollab procura maximizar a colaboração ao passo que evita o recebimento de alterações incompletas no modelo, com recebimento de alterações remotas de forma síncrona, porém com o envio das alterações locais de forma assíncrona. Nesse cenário, o usuário acaba por acumular diversas alterações recebidas que deve aprovar ou rejeitar. Independente de existir alterações recebidas pendentes de avaliação, o usuário pode a qualquer momento enviar suas alterações juntamente com as dos demais usuários já aprovados. Espera-se que esse sistema de *merge* automático com resolução manual de conflitos com aprovação e rejeição de alterações seja mais eficiente que as abordagens clássicas de colaboração síncrona e assíncrona.

4.6 Arquitetura proposta

A UMLCollab foi implementada como um componente na ferramenta de modelagem UML OctoUML fazendo parte de uma arquitetura Modelo-Visão-Controlador (MVC), conforme pode ser visualizado na Figura 7.

Essa arquitetura é baseada em camadas conforme descrito a seguir:

Figura 7 – Arquitetura proposta da UMLCollab



Fonte: Elaborada pelo autor.

Modelo: O modelo consiste nos dados da aplicação, podendo conter regras de negócios, lógica e funções referente a aplicação. Nesse caso é composta de arquivos de código fonte java (.java).

Visão: Camada responsável pela apresentação dos dados ao usuário. São usados arquivos .fxml para especificar os itens exibíveis que compõe a aplicação, .css (Cascading Style Sheets) para adicionar estilos aos elementos que exibíveis na tela e .java: código fonte a linguagem java, específicos para auxiliar na correta exibição dos elementos.

Controlador: O controlador manipula os elementos das demais camadas da aplicação a apartir das entradas do usuário. É nessa camada que se encontra o controlador UMLCollab responsável por registrar alterações locais e interceptar alterações remotas para detectar conflitos, realizar *merge* automático ou registrar e destacar confitos, atualizando os elementos da camada de modelo e visão. Dessa forma o desenvolvedor percebe as mudanças enviadas pelos demais usuários de forma colaborativa e pode enviar suas próprias alterações.

Observa-se na Figura 7 que o OctoUML está presente em todas as camadas da arquitetura MVC.

4.7 Aspectos de implementação

Essa seção descreve detalhes de implementação relevantes para UMLCollab.

4.7.1 Tecnologias utilizadas

Várias linguagens podem serem utilizadas para modelagem de software, porém, a UML permite a representação padronizada de sistemas (PREVEDELLO, 2018). Além disso, a UML é uma linguagem que serve para todas as etapas de modelagem de software (OSMAN, 2013).

As tecnicas de colaboração em modelagem de software com mensagens instantâneas, e-mails e compartilhamento de arquivos são improdutivas para modelos colaborativos, devido ao tempo para resolver conflitos e realizar merge de modelos (OSMAN, 2013), logo a linguagem UML foi escolhida como ferramenta de desenvolvimento dos modelos deste estudo.

Nessse trabalho foi utilizado o código da ferramenta OctoUML modificado, encadeado com o componente da abordagem UMLCollab. Os motivos da escolha dessa ferramenta como base são: ter suporte básico para colaboração síncrona, ser de código aberto, ter suporte para notações formais (diagrama de classes e sequência da UML) e informais, pois não têm regras restritivas de tipos, atributos e assinaturas de métodos, o que foi considerado como "ideal" segundo Osman (2013).

A linguagem de programação é Java, usando especificamente a tecnologia JavaFX e a IDE Eclipse.

4.7.2 Melhorias no código base do protótipo

Nessa seção são apresentados melhorias no código base do protótipo necessárias para realização do experimento de colaboração síncrona e da abordagem UMLCollab.

4.7.2.1 Sincronismo granular

A ferramenta originalmente implementava colaboração síncrona, porém o síncronismo não era granular. Só era possível modificar, por exemplo, do Diagrama de Classes da UML, uma classe inteira, como o nome da classe e todos seus atribudos ou operações. Com essa modificação a classe inteira era enviada para os demais colaboradores sobrescrevendo alterações ainda em andamento. Para permitir a implementação da abordagem UMLCollab e ainda realizar o experimento de colaboração síncrona, foi refatorado o código para permitir sincronismo mais granular, permitindo atualizar e sincronizar o nome da classe, atributo ou operação individualmente. Outras alterações foram necessárias, como registro do nome do usuário, suporte a colaboração simultânea em múltiplos modelos e alterações na interface para identificar o tipo de colaboração e enviar manualmente as alterações locais.

4.7.2.2 Identificador único universal

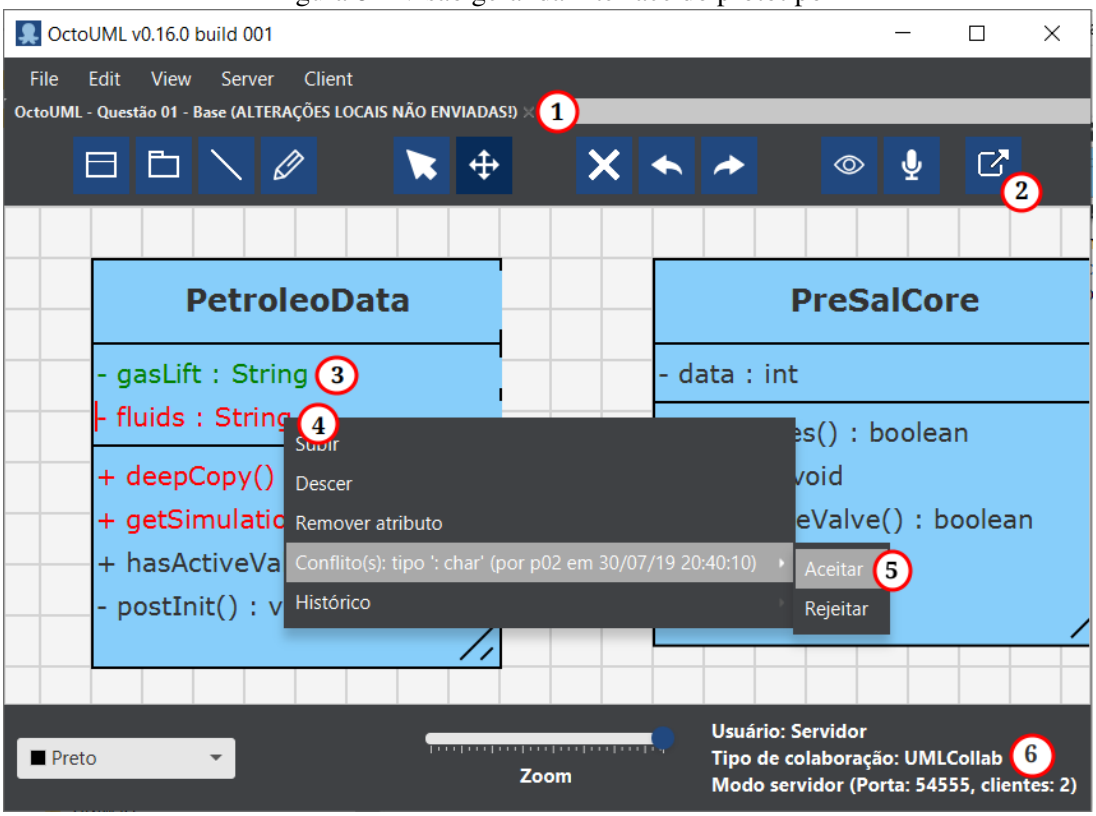
Para identificar ser um elemento é novo ou é um elemento antigo que foi alterado, se faz necessário identificar de forma única cada elemento. A implementação original do OctoUML não atende esse requisito pois gera uma numeração sequencial e um prefixo para identificar cada elemento, como Node_1, Attr_4, Oper_7, etc, a qual é reiniciada cada vez que o sistema é iniciado. Para permitir a identificação precisa em um cenário colaborativo, se fez necessário o uso de Identificador Único Universal, também conhecido como UUID (Universally Unique Identifier). Dessa forma cada elemento ao ser criado recebe um UUID.

Para verificar se a versão de um elemento já não foi enviada por um mesmo usuário, se faz necessário indicar se um elemento foi alterado. Para atender esse requisito, além do UUID usado para identificar um elemento, cada fez que o usuário altera um elemento, é gerado um novo UUID que é salvo em um novo atributo “version”.

4.7.3 Interface do protótipo

A abordagem UMLCollab permite a ciência das alteração dos demais colaboradores com menor interferência no trabalho do desenvoldor. Na Figura 8 é apresentada a tela inicial da abordagem proposta, conforme descrito a seguir.

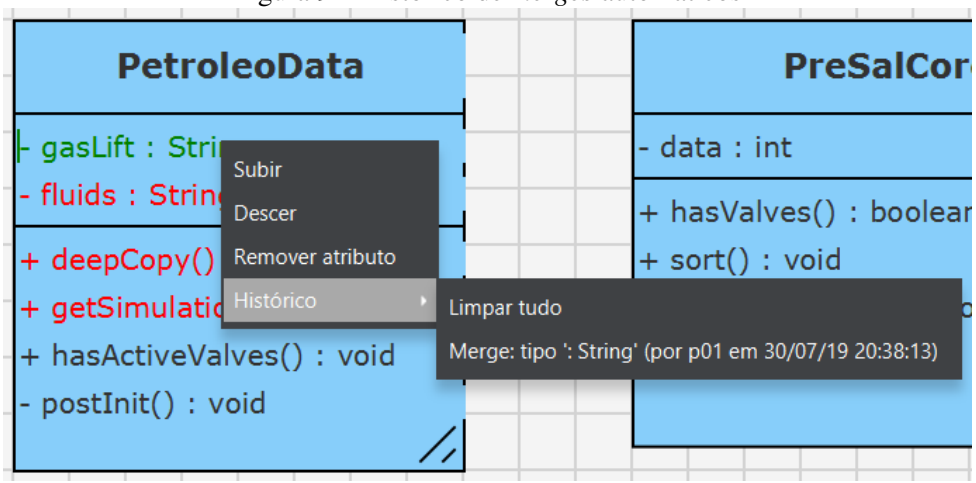
Figura 8 – Visão geral da interface do protótipo



Fonte: Elaborada pelo autor.

- **1:** Uma vez que o usuário efetua alterações locais, a abordagem UMLCollab notifica o colaborador que existem alterações locais para envio aos demais colaboradores.
- **2:** O envio das alterações locais para os demais colaboradores é realizado manualmente através do ícone apresentado. Provavelmente isso deve ocorrer quando o colaborador considerar um conjunto de alterações finalizadas.
- **3:** A UMLCollab realiza merge automático de alterações remotas recebidas quando não há conflitos, isto é, não houve alterações locais conflitantes com alterações remotas. O merge automático é destacado de forma discreta através da alteração da cor do elemento para verde.
- **4:** As alterações remotas que geraram conflitos com as alterações locais são destacados de forma discreta através da alteração da cor do elemento para vermelho.
- **5:** As alterações remotas que geraram conflitos podem serem aceitas ou rejeitadas pelo colaborador no momento que este considerar oportuno, ou seja, pode conviver com os conflitos enquando finaliza seu trabalho cognitivo e conclui um conjunto de alterações para serem enviados.
- **6:** Considerando a adaptação da ferramenta para aceitar colaboração síncrona e a abordagem UMLCollab é exibido no protótipo o tipo de colaboração sendo realizada e outras informações pertinentes.

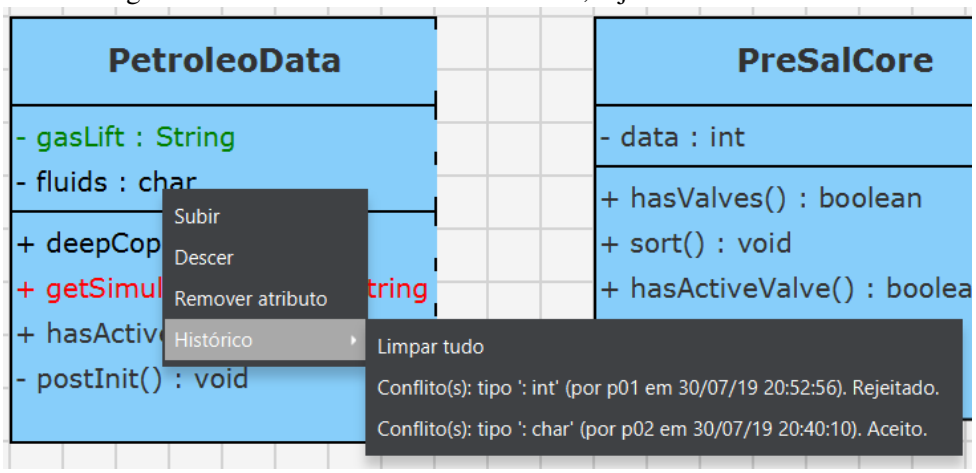
Figura 9 – Histórico de *merges* automáticos



Fonte: Elaborada pelo autor.

Na Figura 9 é apresentado com é exibido o histórico de *merges* automáticos realizados. Os merge automáticos são realizados sempre que não há conflitos para evitar interferências desnecessárias no trabalho do colaborador, não sendo necessário portanto, aceitá-las manualmente ou rejeitá-las como ocorre com as alterações que resultam em conflitos.

Figura 10 – Histórico de conflitos aceitos, rejeitados e descartados



Fonte: Elaborada pelo autor.

Na Figura 10 observa-se o histórico de conflitos aceitos, rejeitados e descartados. Os conflitos descartados por ocorrer quando o colaborador remoto autor de uma alteração previamente recebida pelo autor local, realiza uma nova alteração que coincide com a versão local. Nesse contexto, a UMLCollab resolve automaticamente o conflito fazendo o descarte e registrando no histórico.

4.7.4 Algoritmo do protótipo

Essa seção apresenta o principal algoritmo da abordagem UMLCollab. Na Tabela 11 é demostrado o algoritmo que detecta e trata conflitos e realiza *merge* automático a partir das alterações remotas do modelo recebidas dos demais colaboradores.

No algoritmo ilustado na Tabela 11 pode ser observado que a UMLCollab trata de três tipos de conflitos: 1. alterações locais com alterações remotas; 2. alterações locais com remoções remotas; e 3. remoção local com alteração remota. Também no mesmo algoritmo é observado quando é possível realizar o merge automático da alteração remota no elemento local.

Tabela 11 – Algoritmo de resolução de conflitos e *merge* automático

```
Se TipoColaboração = Síncrona
  Atualiza elemento;
Senão // UMLCollab
  Se elementoRemoto = elementoLocal
    Se existeConflitoAntigo(elementolocal,usuarioRemoto) = verdadeiro
      Remove conflito da lista de conflitos do elemento
      Se numeroConflitos = 0
        Remove indicação de conflito (elemento em vermelho)
        Registra no histórico a remoção automática do conflito
      Fim Se
    Fim Se
  Senão
    Se removido(elementoLocal) = verdadeiro
      Registra o conflito de exclusão local com alteração remota
      Adiciona indicação de conflito (elemento na cor vermelha)
    Senão Se alterado(elementoLocal) = falso
      Realiza merge automáticao
      Registra no histórico o merge automático
      Adiciona indicação de merge automático (elemento na cor verde)
    Senão Se removido(elementoRemoto) = verdadeiro
      Registra o conflito de alteração local com exclusão remota
      Adiciona indicação de conflito (elemento na cor vermelha)
    Senão
      Registra o conflito de alteração local e alteração remota
      Adiciona indicação de conflito (elemento na cor vermelha)
    Fim Se
  Fim Se
Fim Se
```

Fonte: Elaborada pelo autor.

5 AVALIAÇÃO

Para avaliar o impacto da UMLCollab foi realizado um experimento comparando a proposta com a colaboração síncrona e a colaboração assíncrona. Na colaboração síncrona foi usado a ferramenta OctoUML com melhorias no sincronismo do envio e recebimento das alterações em tempo real de forma a ter desempenho semelhante a ferramentas online conhecidas como o GenMyModel. Para a colaboração assíncrona foi usado o IBM RSAD. A UMLCollab foi implementada com adaptações na ferramenta UMLCollab de forma que esta operasse tanto no modo síncrono como no modo UMLCollab.

Este capítulo é organizado da seguinte forma: A Seção 5.1 lista o objetivo e a questão de pesquisa do experimento. Na Seção 5.2 são demonstradas as hipóteses da pesquisa e na Seção 5.3 são evidenciadas as variáveis independentes e dependentes identificadas para o experimento. Na Seção 5.4 é apresentada a população da pesquisa e são identificados os instrumentos que serão utilizados nesta pesquisa. Na Seção 5.5 apresenta-se o detalhamento do desenho do experimento. Na Seção 5.6 é apresentado os resultados do experimento.

5.1 Objetivo e questões de pesquisa

Através de um conjunto de atividades que os participantes devem realizar para alteração de um modelo base (M_b) composto de um Diagrama de Classes da UML em um modelo desejado (M_d), o experimento procura medir o esforço para detecção e resolução de conflitos e a exatidão dos modelos gerados. Dessa forma seguindo o modelo GQM (Goal, Questions, Metrics) (WOHLIN et al., 2000), abaixo é apresentado o objetivo do experimento:

Analisar *as técnicas de sincronização e resolução de conflitos*
para o propósito de *investigar seus efeitos*
com respeito ao *esforço, conflitos e corretude*
a partir da *perspectiva de analistas e desenvolvedores*
no contexto de *modelagem colaborativa de software.*

Nesse experimento foi medido o esforço, conflitos e a corretude dos modelos gerados pelo desenvolvedores usando os diferentes tipos de colaboração em comparação a UMLCollab, portanto foi elaborado as seguintes questões de pesquisa:

- **QP1:** Qual abordagem de colaboração demanda menor esforço para elaborações de modelos de software?
- **QP2:** Qual abordagem de colaboração gera menor quantidade de conflitos na elaboração de modelos de software?
- **QP3:** Qual abordagem de colaboração gera modelos com maior nível de corretude na elaboração de modelos de software?

5.2 Formulação das hipóteses

5.2.1 Hipóteses do experimento para os testes inferenciais

Partindo do pressuposto que a colaboração síncrona prejudica o processo cognitivo devido aos vários desenvolvedores atuando no mesmo modelo simultaneamente (BRIERTON et al., 2016) e a assíncrona sofre com o problema dos conflitos, a abordagem UMLCollab através da colaboração híbrida propõe a redução do esforço na elaboração dos modelos que seja mais colaborativo ou intervenha positivamente na resolução de conflitos, devido a menor interferência no trabalho dos demais desenvolvedores.

A Tabela 12 ilusta um resumo das hipóteses do experimento que visam mensurar a eficiência e eficácia da elaboração de modelos da UMLCollab em relação aos tipos de colaboração síncrona e assíncrona.

Tabela 12 – Resumo das hipóteses para análises dos testes inferenciais

Hipótese	**Tipo**	**Representação**
H1	Nula	H_{1-0}: esf(M_b,M_d)$_{assnc}$ = esf(M_b,M_d)$_{umlc}$ = esf(M_b,M_d)$_{sinc}$
	Alternativa	H_{1-1}: esf(M_b,M_d)$_{assnc}$ $\neq$ esf(M_b,M_d)$_{umlc}$ $\neq$ esf(M_b,M_d)$_{sinc}$
H2	Nula	H_{2-0}: conf(M_b,M_d)$_{assnc}$ = conf(M_b,M_d)$_{umlc}$ = conf(M_b,M_d)$_{sinc}$
	Alternativa	H_{2-1}: conf(M_b,M_d)$_{assnc}$ $\neq$ conf(M_b,M_d)$_{umlc}$ $\neq$ conf(M_b,M_d)$_{sinc}$
H3	Nula	H_{3-0}: cor(M_b,M_d)$_{assnc}$ = cor(M_b,M_d)$_{umlc}$ = cor(M_b,M_d)$_{sinc}$
	Alternativa	H_{3-1}: cor(M_b,M_d)$_{assnc}$ $\neq$ cor(M_b,M_d)$_{umlc}$ $\neq$ cor(M_b,M_d)$_{sinc}$

Legenda: M_b = Modelo base; M_d = Modelo desejado; Esf = Esfoço; Conf = Conflitos; Cor = Corretude; assínc = assíncrona; sínc = síncrona; umlc = UMLCollab

Fonte: Elaborada pelo autor.

A redução do esforço isoladamente não é o único fator a ser considerado. É necessário verificar a corretude do modelo gerado, considerando que cada desenvolvedor possa elaborar seus modelos com menor interferência do trabalho dos demais desenvolvedores.

Dessa forma, foram elaboradas as seguintes hipóteses nulas e alternativas:

- **Hipótese nula 1 (H_{1-0}):** Não existe diferença no esforço da UMLCollab em relação a colaboração assíncrona ou colaboração síncrona.
- **Hipótese alternativa 1 (H_{1-1}):** Existe diferença no esforço da UMLCollab em relação a colaboração assíncrona ou colaboração síncrona.
- **Hipótese nula 2 (H_{2-0}):** Não existe diferença no número de conflitos da UMLCollab em relação a colaboração assíncrona ou colaboração síncrona.
- **Hipótese alternativa 2 (H_{2-1}):** Existe diferença no número de conflitos da colaboração híbrida em relação a colaboração assíncrona ou colaboração síncrona.

- **Hipótese nula 3 (H_{3-0}):** Não existe diferença no aumento da corretude da UMLCollab em relação a colaboração assíncrona ou colaboração síncrona.

- **Hipótese alternativa 3 (H_{3-1}):** Existe diferença no aumento da corretude da colaboração híbrida em relação a colaboração assíncrona ou colaboração síncrona.

5.3 Variáveis do estudo

Foram identificadas como sendo as variáveis independentes que poderiam influenciar nos resultados do estudo, as seguintes: a) a educação dos participantes (graduação e pós-graduação), para aplicação dessa variável no experimento foi utilizada uma combinação binária, na qual foi atribuído 1 quando a característica apontada se referia aos participantes com nível de pós-graduação e 0 quando se referia aos participantes de nível de graduação; b) a experiência na área com o tema do estudo; e c) a idade dos participantes. Os dados das variáveis independentes foram adquiridos a partir de um questionário aplicado no final de cada experimento.

Como variáveis dependentes, foram avaliados o esforço, a corretude e os conflitos de cada tipo de técnica de modelagem colaborativa. A variável dependente da primeira hipótese são os esforços da modelagem, ou seja, os tempos representados em minutos para execução de uma lista de atividades elaboradas para a realização dos experimentos da pesquisa. A variável dependente da segunda hipótese são os conflitos do modelo tratados por cada participante. Estes conflitos foram contados a partir da observação das gravações dos vídeos de todos os participantes do experimento. A variável dependente da terceira hipótese é a corretude do modelo gerado em relação ao modelo desejado, isto é, se o modelo alterado a partir de um modelo base é igual ao modelo desejado planejado em cada questão do experimento. A Tabela 13 apresenta as variáveis independentes e dependentes do estudo.

Tabela 13 – Variáveis do experimento

Tipo de variável	**Variáveis**	**Escala ou valores**
Independente	- Abordagem de colaboração	- Nominal: Síncrona, assíncrona ou UMLCollab
	- Educação	- Ordinal: Graduação (0), Pós-graduação (1)
	- Experiência	- Ordinal: Número de anos de experiência do participante.
	- Idade	- Ordinal: Número de idade do participante.
Dependente	- Esforço	- Ordinal: Tempo em minutos
	- Conflitos	- Ordinal: Quantidade de conflitos por participante
	- Corretude	- Ordinal: Número de percentual de acertos por equipe

Fonte: Elaborada pelo autor.

A variável dependente esforço foi quantificada conforme o tempo gasto em minutos por cada equipe. Essa quantidade foi mensurada através dos vídeos de cada grupo. A variável conflito foi quantificada através da quantidade de conflitos gerados pelos participantes, seguindo a mesma

linha da variável esforço, essa quantidade foi mensurada através da observação da gravação dos vídeos. A variável corretude foi mensurada a partir da comparação dos modelos gerados por cada equipe, com o modelo esperado do estudo. A partir da comparação, foram gerados os percentuais de acerto por equipe.

5.3.1 Hipóteses do experimento para as análises de regressão linear múltiplas

O experimento buscou explicar as influências das variáveis independentes em relação as variáveis dependentes do estudo, logo foram elaboradas hipóteses para serem testadas nas análises das regressões:

- **Hipótese nula 4 ($\mathbf{H_{4-0}}$):** A educação, experiência ou idade dos participantes não afeta significativamente o uso da UMLCollab, colaboração assíncrona ou colaboração síncrona ao ponto de influenciar o esforço.

- **Hipótese alternativa 4 ($\mathbf{H_{4-1}}$):** A educação, experiência ou idade dos participantes afeta significativamente o uso da UMLCollab, colaboração assíncrona ou colaboração síncrona ao ponto de influenciar o esforço.

- **Hipótese nula 5 ($\mathbf{H_{5-0}}$):** A educação, experiência ou idade dos participantes não afeta significativamente o uso da UMLCollab, colaboração assíncrona ou colaboração síncrona ao ponto de influenciar o número de conflitos.

- **Hipótese alternativa 5 ($\mathbf{H_{5-1}}$):** A educação, experiência ou idade dos participantes afeta significativamente o uso da UMLCollab, colaboração assíncrona ou colaboração síncrona ao ponto de influenciar o número de conflitos.

- **Hipótese nula 6 ($\mathbf{H_{6-0}}$):** A educação, experiência ou idade dos participantes não afeta significativamente o uso da UMLCollab, colaboração assíncrona ou colaboração síncrona ao ponto de influenciar a corretude.

- **Hipótese alternativa 6 ($\mathbf{H_{6-1}}$):** A educação, experiência ou idade dos participantes afeta significativamente o uso da UMLCollab, colaboração assíncrona ou colaboração síncrona ao ponto de influenciar a corretude.

5.4 População e instrumentos de pesquisa

Neste trabalho foram realizados pré-testes e o experimento final com a participação dos alunos de graduação em informática e pós-graduação Stricto Sensu (mestrados e doutorados) do curso de computação aplicada de uma universidade privada localizada no Rio Grande do Sul, matriculados a partir de 2017 até a presente data.

Na Tabela 14 pode-se verificar o perfil da educação dos participantes, ou seja, do total 15 alunos que participaram do experimento, 5 são graduandos e 10 participantes são pós-graduandos (7 mestrandos e 3 doutorandos).

Tabela 14 – Educação dos participantes do experimento

Educação	**Nº de Participantes**	**Percentual**
Graduação	5	33,33
Pós-graduação	10	66,67

Fonte: Elaborada pelo autor.

Em relação a idade dos participantes, observa-se na Tabela 15 que dos 15 participantes, 7 alunos têm idade até 24 anos e 7 estão entre 25 e 40 anos. Apenas 1 participante possui idade acima de 40 anos.

Tabela 15 – Idade dos participantes

Faixa etária	**Nº de Participantes**	**Percentual**
Até 24 anos	7	46,67
De 25 até 40 anos	7	46,67
Acima de 40 anos	1	6,67

Fonte: Elaborada pelo autor.

No que tange a ocupação dos participantes, observa-se na Tabela 16 que, da população total de participantes, 2 atuam como Analista/Desenvolvedor de Sistemas, 2 auxiliares técnicos, 4 bolsistas de iniciação científica/CAPES, pesquisador ou assistente de pesquisa, 2 engenheiros eletricistas, 2 estudantes, 2 professores e 1 técnico de automação industrial.

Tabela 16 – Ocupação dos participantes

Ocupação	**Nº de Participantes**	**Percentual**
Analista/desenvolvedor de sistemas	2	13,33
Auxiliar técnico	2	13,33
Bolsistas de iniciação científica/CAPES, pesquisador ou assistente de pesquisa	4	26,67
Engenheiros eletricistas	2	13,33
Estudantes	2	13,33
Professores	2	13,33
Técnico de automação industrial	1	6,67

Fonte: Elaborada pelo autor.

Outro item avaliado foi a experiência em modelagem de software com UML. Nesse quesito, observa-se na Tabela 17 que, a maioria, ou seja, 10 alunos declararam ter até 2 anos de experiência, 3 alunos declaram ter entre 2 a 5 anos de experiência e somente dois alunos declararam mais de 5 anos de experiência.

Tabela 17 – Experiência em modelagem de software com UML

Anos de experiência	Nº de Participantes	Percentual
Até 2 anos	10	67,67
De 2 a 5 anos	3	20,00
Acima de 5 anos	2	13,33

Fonte: Elaborada pelo autor.

Para realização do experimento foram utilizados os seguintes instrumentos listados abaixo, com destaque para o aplicativo OctoUML que serviu de base para implementação da UMLCollab além de melhorias para os testes com colaboração síncrona.

- Aplicativo OctoUML para concorrência síncrona;
- Aplicativo IBM RSAD para concorrência assíncrona;
- Aplicativo Kazam para gravação de tela;
- Roteiro de atividades;
- Questionário do experimento; e
- 03 Computadores desktop, 01 notebook e 01 roteador.

O roteiro de atividades e o questionário do experimento foram utilizados no momento do experimento assim como, os computadores, o notebook e o roteador. Todos os instrumentos utilizados facilitaram a execução do experimento e a elaboração do desenho de pesquisa, item que será abordado no tópico a seguir.

5.5 Desenho do experimento da pesquisa

O desenho da pesquisa é um resumo de todas as etapas do experimento. O desenho foi elaborado com o objetivo de facilitar a compreenção das etapas do experimento. A Figura 11 ilustra as três etapas do experimento.

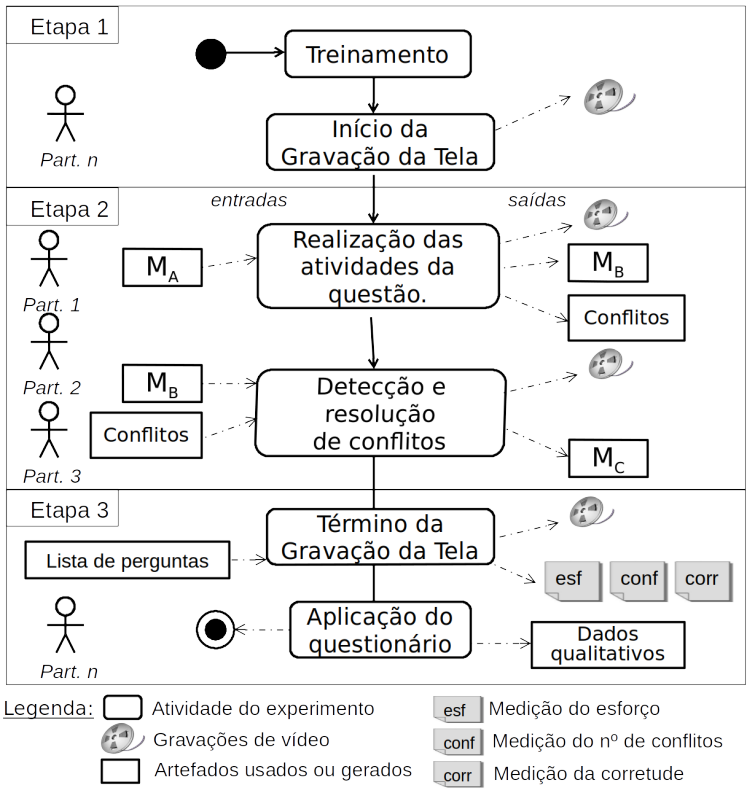
A Figura 11 foi baseada no processo experimental (Figura 3 - *The experimental Process*) do estudo experimental de Farias et al. (2015). Nas próximas seções, são apresentadas as descrições de cada etapa do experimento.

5.5.1 Primeira etapa do experimento

A primeira etapa compreende o treinamento nas ferramentas utilizadas e o início da gravação do experimento.

No treinameno é apresentado através de slides a forma de usar passo a passo do software OctoUML, versão adaptada com colaboração síncrona e UMLCollab e o software IBM RSAD.

Figura 11 – Desenho do experimento



Fonte: Adaptado de Farias et al. (2015).

Para gravação da tela é utilizado o software Kazam para documentar e validar o experimento, permitindo colher os dados da modelagem colaborativa, como o esforço (tempo em minutos) usado na resolução de cada questão por cada grupo e a quantidade de conflitos.

5.5.2 Segunda etapa do experimento

Na segunda etapa é iniciada o experimento propriamente dito, com a realização de atividades para evidenciar conflitos distribuídas em 6 questões que foram respondidas por grupos de 3 participantes utilizando a colaboração síncrona, assíncrona e UMLCollab. Tanto para a UMLCollab com para a colaboração síncrona é utilizada a ferramenta OctoUML no modo UMLCollab e no modo síncrono, respectivamente. Para a colaboração assíncrona é utilizado a ferramenta IBM RSAD.

A cada 2 questões a técnica é trocada, iniciando pela modelagem colaborativa síncrona, depois a modelagem colaborativa assíncrona e por último a modelagem colaborativa híbrida usando a UMLCollab. A cada grupo de participantes a ordem das ferramentas é alterada com o objetivo de evitar prejudicar a validade do experimento e possível influência de uma ferramenta ser utilizada antes da outra, conforme Tabela 18.

Cada questão trabalha em um modelo base único e contém 3 três atividades distintas dos demais participantes. As três atividades são projetadas para gerar conflitos com as atividades

Tabela 18 – Iterações da segunda etapa

Grupo	Questão 01	Questão 02	Questão 03	Questão 04	Questão 05	Questão 06
1	Síncrona	Síncrona	Assíncrona	Assíncrona	UMLCollab	UMLCollab
2	UMLCollab	UMLCollab	Síncrona	Síncrona	Assíncrona	Assíncrona
3	Assíncrona	Assíncrona	UMLCollab	UMLCollab	Síncrona	Síncrona
4	Síncrona	Síncrona	UMLCollab	UMLCollab	Assíncrona	Assíncrona
5	UMLCollab	UMLCollab	Assíncrona	Assíncrona	Síncrona	Síncrona

Fonte: Elaborada pelo autor.

dos demais participantes, porém uma delas também é o objetivo da versão final modelo, ou seja, deve ser mantido no modelo após a resolução de conflitos com os demais participantes. Dessa forma, cada participante contribui para gerar o modelo desejado ao passo que simula conflitos com os demais participantes. Só é possível passar para próxima questão quando todos os participantes finalizaram cada questão. Para verificar a corretude dos modelos gerados em cada questão, todos os modelos foram salvos no sistema de controle de versão Subversion e depois comparados ao modelo desejado.

5.5.3 Terceira etapa do experimento

Na etapa 3 é finalizada a gravação pelo software Kazam e aplicado questionário do experimento aos participantes conforme Apêndice E. O questionário do experimento foi elaborado para dados referentes ao perfil acadêmico e profissional dos participantes escolhidos aleatoriamente.

Além dos dados básicos do participante, para cada etapa do experimento usando a colaboração síncrona, assíncrona e híbrida é solicitado uma nota de 1 a 5 para a produtividade percebida pelos participantes, sendo a nota 1 a de menor produtividade e 5 a maior produtividade. Em busca das impressões dos usuários com cada ferramenta, foi adicionado também para cada tipo de colaboração, um campo de texto livre para que o participante justifique sua nota. Além dessas questões, foram adicionadas questões sobre percepção de facilidade de uso, percepção de utilidade, atitude e intenção de comportamento.

5.6 Análise dos resultados

Para verificar a validade das hipóteses do experimento, foram coletados os esforços (tempos em minutos), os conflitos e a corretude dos modelos gerados pelos participantes. As análises dos resultados foram realizadas através da estatística descritiva, teste de correlação de Pearson e da regressão linear múltipla usando o software estatístico RStudio[1].

Cada participante, dentro de cada uma das 5 equipes compostas de 3 membros, contribuiu

[1] https://www.rstudio.com

com duas medições de esforços para cada tipo de colaboração. Isso foi possível, pois haviam duas questões independentes para cada tipo de colaboração. Para identificar os esforços, foram coletados 3 métricas distintas: de forma objetiva, através da gravação de tela de cada participante foram verificados os esforços que cada equipe levou para terminar cada questão, desde o início de cada questão até que cada participante atingisse seu objetivo; de forma subjetiva, no questionário foi perguntado se a UMLCollab ajuda a aumentar a produtividade e também foi requisitado uma nota de 1 a 5 para produtividade de cada tipo de colaboração em relação as demais.

5.6.1 Análise descritiva dos dados

Nessa subseção é apresentado uma análise descritiva dos dados do experimento.

5.6.1.1 Variáveis independentes por grupo

A Tabela 19 evidencia os dados coletados das variáveis independentes "educação", "idade" e "experiência". Para a variável "educação", nos testes estatísticos foi considerado a combinação binária, na qual foi atribuído 1 quando a característica apontada se referia aos participantes com nível de pós-graduação e 0 quando se referia aos participantes de nível de graduação. Em relação a "idade" e a "experiência" foi verificado a média dos participantes de cada grupo.

Tabela 19 – Variáveis independentes por grupo

Grupo	**Educação**	**Idade Média**	**Experiência Média**
1	Pós-graduação	38	2
2	Graduação	25	3
3	Pós-graduação	28	4
4	Pós-graduação	24	2
5	Graduação	20	1

Fonte: Elaborada pelo autor.

5.6.1.2 Médias do esforço, conflitos e corretude por tipo de colaboração

Na Tabela 20 pode-se verificar as duas medições de esforço e conflitos para cada tipo de colaboração, juntamente com o índice de corretude do modelo gerado por cada equipe. Considerando as medições de esforço individuais, pode-se verificar na que a média do esforço para a colaboração assíncrona da primeira medição foi de 10:19 (dez minutos e dezenove segundos). Na segunda medição a média caiu para 06:05 (seis minutos e cinco segundos). Um dos motivos para essa queda pode ser explicado pela maior facilidade do uso da ferramenta, uma vez que o participante tenha adquirido experiência ao realizar a questão anterior. A corretude no

entanto, sofreu queda de 97% da primeira medição para 90% na segunda. A média de conflitos aumentou de 1,27 para 1,40 conflitos.

Tabela 20 – Médias do esforço, conflitos e corretude por tipo de colaboração

Colaboração	Esf1 (min)	Conf1	Cor1	Esf2 (min)	Conf2	Cor2
Assíncrona	10:19	1,27	0,97	06:05	1,40	0,90
Síncrona	02:31	2,67	0,96	01:34	1,33	0,90
UMLCollab	04:37	5,47	0,91	03:31	5,80	0,98

Legenda: G = Grupo; Esf = Esfoço; Conf = Conflitos; Cor = Corretude; M = Médias

Fonte: Elaborada pelo autor.

É interessante observar que a diferença nas medições de cada participante de cada equipe é influenciada pelo tipo de colaboração utilizada. Na colaboração assíncrona, por exemplo, o primeiro participante pode realizar o commit com sucesso de suas alterações, porém o segundo participante pode não conseguir fazer commit pois suas alterações foram baseadas em um modelo desatualizado. Dessa forma é necessário realizar *merge* das alterações remotas com as alterações locais resolvendo os conflitos e logo seu esforço será maior que o primeiro participante.

Em relação a colaboração síncrona, verifica-se na Tabela 20 que a média do esforço individuais da colaboração síncrona da primeira medição em relação a segunda, de forma semelhante a colaboração assíncrona, caiu de 02:31 (dois minutos e trinta e um segundos) para 01:34 (um minuto e trinta e quatro segundos). A corretude de forma similar, sofreu queda de 96% da primeira medição para 90% na segunda. A média de conflitos diminuiu de 2,67 para 1,33 conflitos.

No que tanje a UMLCollab, observa-se na Tabela 20 que a média do esforço individuais para a UMLCollab manteve o mesmo comportamento da primeira medição em relação a segunda dos demais tipos de colaboração, caindo de 04:37 (quatro minutose e trinta e sete segundos) para 03:31 (três minutos e trinta e um segundos). A corretude no entanto, ao invés da queda, obteve aumento de 91% da primeira medição para 98% na segunda. A média de conflitos aumentou de 5,47 para 5,80 conflitos.

5.6.1.3 Resposta da QP1: Comparação de esforços dos modelos colaborativos

A Tabela 21 apresenta as médias da variável dependente esforço e foi elaborada para comparar e perceber o desempenho das amostras das cinco equipes por modelo colaborativo.

Tabela 21 – Médias da variável esforço

Colaboração	Mediana	Média	Mínima	Máximo	Desvio Padrão
Assíncrona	10,10	10,90	8,96	15,10	2,45
Síncrona	2,38	2,49	1,85	3,87	0,81
UMLCollab	3,84	4,40	2,26	7,18	1,86

Fonte: Elaborada pelo autor.

Observa-se que na colaboração assíncrona, onde o envio e o recebimento dos dados alterados no projeto de experimento são manuais, o esforço médio foi de 10,90 minutos. Na UMLCollab, onde envio dos dados é manual e recebimento é automático, apresentou uma média de 4,40 minutos, ou seja, uma redução de esforços de 59,63% em relação a colaboração assíncrona, logo, um desempenho significativamente melhor em relação a colaboração anterior.

Na colaboração síncrona, onde tanto o envio quanto o recebimento das alterações são automáticos, o esforço médio foi de 2,49 minutos, isto é, houve uma redução de 77,16% do esforço em comparação com o modelo assíncrono e uma redução de 43,41% em relação a UMLCollab. Essa redução não é uma surpresa, pois, a colaboração síncrona não requer nenhum esforço para enviar ou receber qualquer alteração manualmente.

Para compreender e confirmar estes resultados, em todos os dados da variável "esforço" foi aplicada a técnica qualitativa de observação não-participante nas gravações dos vídeos dos participantes, onde percebeu-se e confirmou-se que realmente as alterações nos projetos são imediatamente enviadas e recebidas sem nenhuma interação ou colaboração entres os participantes. Portanto, essa técnica apresentou um melhor esforço, porque não existe interação e nem colaboração dos participantes nesta modelagem.

Esse procedimento de não consultar os participantes, possivelmente impactaria na qualidade final do projeto "colaborativo", ou seja, já que projetos não foram debatidos entre os participantes e ninguém necessitou entrar em consenso para saber qual é a melhor alternativa para um projeto de qualidade, possivelmente existe uma redução de qualidade na entrega de projeto "colaborativo" utilizando esta modelagem.

Apesar de apresentar sua produtividade intermediária em relação as demais técnicas, a UMLCollab apresenta a vantagem em relação a colaboração síncrona, pois não apresenta a restrição de quando alguém está alterando um elemento, como um atributo, outro colaborador não consegue efetivamente alterar o mesmo elemento sem atrapalhar o que o primeiro está tentando elaborar.

Outros dados da pesquisa que são interessantes para explanar é a mínima e a máxima de esforços. Observe-se na Tabela 21 que a UMLCollab mesmo enviando dados manualmente, apresentou uma mínima de esforço bem próxima da técnica síncrona que apresenta envios e recebimentos automáticos sem qualquer colaboração dos participantes. Em relação a máxima, cabe observar que a máxima da UMLCollab também apresentou dados relativamente razoáveis em comparação com os tipos de colaboração assíncrona e síncrona.

Portanto, respondendo a QP1, a colaboração síncrona demanda menor esforço para elaborações de modelos de software. Porém, houve um viés de pesquisa nesse resultado, pois verificou-se *in loco* que nesta abordagem, não existiu colaboração dos participantes, por causa das diversas interferências (interrupções), ou seja, a abordagem síncrona não permitiu que os usuários pudessem fechar um ciclo de raciocínio, desfavorecendo a execução do fluxo de atividades pré-estabelecido na inicial do experimento. Sendo assim, a abordagem UMLCollab apresentou o melhor esforço, pois, diferente da colaboração síncrona, os participantes interagi-

ram e colaboraram entre si. Estes resultados corroboram com o pensamento de Gray e Rumpe (2017), quando argumentaram que o fluxo de concentração dos desenvolvedores não deve ser interrompido, pois, isso prejudica a produtividade, logo os autores esperam que as próximas ferramentas de modelagem sejam melhoradas e aprimoradas. Sendo assim, a UMLCollab poderia ser a resposta a crítica destes autores em relação aos tipos de colaborações existentes.

5.6.1.4 Resposta da QP2: Comparação dos conflitos dos modelos colaborativos

Na Tabela 22 é apresentado as médias da variável dependente número de conflitos e pode ser verificado que a menor média de conflitos foi de 1,33 alcançada pelo colaboração assíncrona, o que é um dado prático interessante, visto que esperava-se que esse tipo de colaboração tivesse uma maior quantidade de conflitos devido a demora para o envio e recebimento de alterações.

Tabela 22 – Médias da variável número de conflitos

Colaboração	Mediana	Média	Mínima	Máximo	Desvio Padrão
Assíncrona	1,00	1,33	0,67	2,50	0,75
Síncrona	1,67	2,00	1,33	3,17	0,75
UMLCollab	4,33	5,63	3,00	11,17	3,20

Fonte: Elaborada pelo autor.

A UMLCollab aparece com a maior média de 5,63 conflitos em oposição a colaboração síncrona com média intermediária de 2,00 conflitos. É interessante destacar ainda que a UMLCollab têm o maior máximo de 11,17 conflitos.

Essa quantidade maior de conflitos pode ser explicada pela maior interatividade da UMLCollab na qual os usuários recebem automaticamente as alterações remotas, porém o usuário pode aceitá-las ou rejeitá-las. Com esse mecanismo foi observado a ocorrência de um usuário rejeitar um conflito, enviar sua alteração local para o usuário remoto e o usuário remoto insistir na sua versão do elemento enviando novamente e gerando novo conflito. Dessa forma, ocorria um mecanismo interativo aonde cada um tinha oportunidade de verificar as alterações remotas e negociar com os demais participantes aceitando ou rejeitando os conflitos. Apesar do número de conflitos ser maior, os esforço foi intermediário, demostrando que os conflitos eram rapidamente resolvidos.

Portanto, respondendo a QP2, a colaboração assíncrona gera menor quantidade de conflitos na elaboração de modelos de software. Este resultado pode ser explicado pois, nessa abordagem, observou-se um menor nível de interação entre os participantes. Talvez a demora do envio e recebimento das alterações com os demais participantes também possa explicar esse resultado.

5.6.1.5 Resposta da QP3: Comparação da corretude dos modelos colaborativos

Dado sequência a análise dos dados, para medir a corretude do modelo alterado pelos participantes em relação ao modelo esperado, foi elaborado uma pontuação máxima para cada

questão, sendo 4 pontos para operações (visibilidade, nome, argumentos e tipo de retorno) e 3 pontos para atributos (visibilidade, nome e tipo). Dessa forma, por exemplo, um modelo com classes contendo dois atributos e 8 operações tinha o máximo de 38 pontos. Cada subelemento (visibilidade, nome, etc) errado descontava um ponto da nota máxima. Cada atributo duplicado ou removido descontava 3 pontos e cada operação duplicada ou removida descontava 4 pontos.

Usando essa metodologia, foram avaliadas as corretudes pelas pontuações totais e evidenciadas em percentuais. Logo, verifica-se na Tabela 23 que apresenta a médias da variável dependente corretude, que a colaboração assíncrona obteve uma média de 94%, uma mínima de 86% e um máxima de 99% de corretude do modelo alterado.

Tabela 23 – Corretude

Colaboração	**Mediana**	**Média**	**Mínima**	**Máximo**	**Desvio Padrão**
Assíncrona	0,96	0,94	0,86	0,99	0,056
Síncrona	0,91	0,93	0,90	0,96	0,028
UMLCollab	0,95	0,94	0,89	0,99	0,034

Fonte: Elaborada pelo autor.

O modelo síncrono apresentou uma média menor de 93% de corretude e uma máxima menor de 96% de corretude. A UMLCollab por sua vez atingiu a média de 94% de corretude, similar a colaboração assíncrona e ligeiramente maior que a colaboração síncrona, porém apresentou o máximo de 99%, ou seja, percentual superior a colaboração síncrona e similar ao resultado percentual da colaboração assíncrona (99%).

Observa-se as semelhanças das médias das corretudes nos três tipos de colaboração. Apesar de serem técnicas de colaboração com índices de produtividade de esforços bastante diferenciados, na corretude as diferenças foram evidenciadas nas máximas favorecendo a colaboração assíncrona e a UMLCollab, com destaque para essa última técnica, devido ao melhor resultado na mínima da corretude observada em relação a colaboração assíncrona, ou seja, a UMLCollab apresentou o modelo final com melhor qualidade.

Portanto, respondendo a QP3, a abordagem UMLCollab gera modelos com maior nível de corretude na elaboração de modelos de software. Talvez um menor nível de inteferência no recebimento das alterações com os demais participantes possa explicar esse resultado.

5.6.1.6 Teste Shapiro Wilk para Esforço, Conflitos e Corretude

Para analisar a distribuição dos resultados das análises da estatística descritiva e para realizar os testes de hipóteses, foram aplicados testes estatísticos inferenciais. Para analisar a normalidade das amostras, optou-se em executar o teste de Shapiro-Wilk que apresenta um melhor desempenho em amostras reduzidas (n <30) (LEVINE; BERENSON; STEPHAN, 2005). Para demonstrar os resultados dos testes de normalidade foi elaborada a Tabela 24.

O resultado do teste Shapiro Wilk para o esforço e corretude foram positivos para distribuição normal. Logo, este estudo teve a oportunidade de aplicar vários testes estatísticos paramé-

Tabela 24 – Teste Shapiro Wilk para esforço, conflitos e corretude

Colaboração	W Esf	p-value Esf	W Conf	p-value Conf	W Cor	p-value Cor
Assíncrona	0,807	0,093	0,885	0,332	0,881	0,314
Síncrona	0,803	0,085	0,885	0,332	0,827	0,132
UMLCollab	0,962	0,819	0,783	0,059	0,986	0,964

Legenda: Esf = Esforço; Conf = Conflito; Cor = Corretude;

Fonte: Elaborada pelo autor.

tricos nos dados e dar mais robustez na pesquisa.

Os testes estatísticos paramétricos utilizados nesse estudo foram o teste Anova, teste de correlação de Pearson e regressão linear múltiplas. O teste Anova foi utilizado devido o estudo possuir múltiplas variáveis dependentes. O teste de correlação de Pearson foi aplicado para verificar as possíveis relações entre as variáveis do estudo (dependentes e independentes). A regressão linear múltipla foi aplicada após identificada as relações entre as variáveis pelo teste de correlação de Pearson para identificar as possíveis influências das variáveis independentes em relação as variáveis dependentes. Sendo que, para todos os resultados destes testes estatísticos deve ser levado em consideração o tamanho da amostra do estudo.

5.6.1.7 Teste Anova do esforço dos tipos de colaboração

Para verificar a significância da diferença das médias das três váriáveis dependentes (esforço, conflitos e corretude) dos três tipos de colaboração utilizadas, foi aplicado o teste Anova.

Na Tabela 25, observa-se que no teste do esforço o resultado do valor de F foi 28,81, sendo que valor crítico do esforço foi de 3,89.

Tabela 25 – Teste Anova do esforço dos tipos de colaboração

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
ivLabel	2	778,8	389,4	28,81	2,62e-05
Residuals	12	162,2	13,5		

Fonte: Elaborada pelo autor.

Portanto, o resultado apresenta um coeficiente de F maior que o valor crítico. Dessa forma pode-se confirmar que existe diferença significativa no esforço dos tipos de colaboração. Sendo assim, após a análise do esforço evidenciada na Tabela 25, pode-se rejeitar a hipótese nula 1 (H_{1-0}), ou seja, não existe diferença no esforço da UMLCollab em relação a colaboração assíncrona ou colaboração síncrona e aceitar a hipótese alternativa 1 (H_{1-1}): Existe diferença no esforço da UMLCollab em relação a colaboração assíncrona ou colaboração síncrona.

5.6.1.8 Teste Anova do número de conflitos dos tipos de colaboração

Na Tabela 26, observa-se que no teste do número de conflitos o resultado do valor de F foi 7,051, sendo que valor crítico dos conflitos foi de 3,89.

Tabela 26 – Teste Anova dos conflitos dos tipos de colaboração

	Df	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
ivLabel	2	53,56	26,780	7,051	0,00944**
Residuals	12	45,58	3,798		

Legenda: **Significante a nível de 1%

Fonte: Elaborada pelo autor.

Portanto, o resultado apresenta um coeficiente de F maior que o valor crítico. Dessa forma pode-se confirmar que existe diferença significativa no número de conflitos dos tipos de colaboração. Sendo assim, após a análise dos conflitos evidenciados na Tabela 26, pode-se rejeitar a hipótese nula 1 (H_{1-0}), ou seja, não existe diferença no número de conflitos da UMLCollab em relação a colaboração assíncrona ou colaboração síncrona e aceitar a hipótese alternativa 1 (H_{1-1}): Existe diferença no número de conflitos da UMLCollab em relação a colaboração assíncrona ou colaboração síncrona.

5.6.1.9 Teste Anova da corretude dos tipos de colaboração

De acordo com a Tabela 27, no teste da corretude, o valor do teste F foi 0,216. Portanto, o resultado de F foi menor que o valor crítico (3,89), dessa forma não foi encontrada diferença significativa das médias de corretude dos tipos de colaboração do experimento.

Tabela 27 – Teste Anova da corretude dos tipos de colaboração

	Df	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
ivLabel	2	0,000738	0,0003691	0,216	0,809
Residuals	12	0,020472	0,0017060		

Fonte: Elaborada pelo autor.

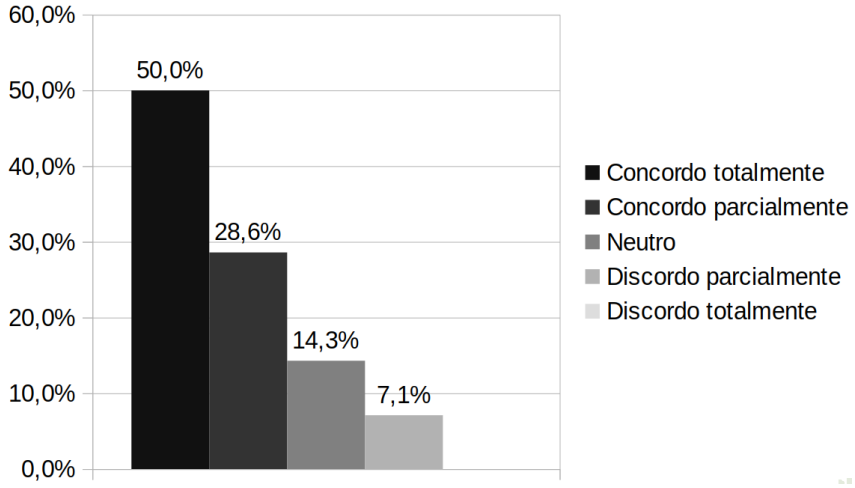
Porém, cabe recordar e ressaltar que o resultado das máximas das corretudes de cada tipo de colaboração evidenciaram que a UMLCollab e colaboração assíncrona apresentaram resultados melhores, ou seja, em relação ao desempenho da corretude, os resultados apontam que a UMLCollab (99%) e a colaboração assíncrona (99%) possuem nível de acertos superiores comparados com a colaboração síncrona (96%). Logo, a análise da corretude demonstrado na Tabela 27, rejeita a hipótese nula 3 (H_{3-0}) e aceita a hipótese alternativa 3 (H_{3-1}), ou seja, existe diferença na corretude da UMLCollab em relação a colaboração assíncrona ou na colaboração síncrona. No caso específico deste experimento, os resultados demonstraram que realmente existe diferença para as corretures do modelo UMLCollab em relação ao modelo síncrono.

5.6.1.10 Produtividade com a UMLCollab

Para avaliar qualitativamente os resultados do experimento, foram extraídos dados do levamento feito através dos questionários do Google Docs preenchidos no final do experimento.

Para complementar as avaliações do desempenho das técnicas do experimento, foi elaborada a Figura 12, que mostra dados sobre as percepções dos participantes em relação ao esforço.

Figura 12 – A UMLCollab ajuda a aumentar a produtividade



Fonte: Elaborada pelo autor.

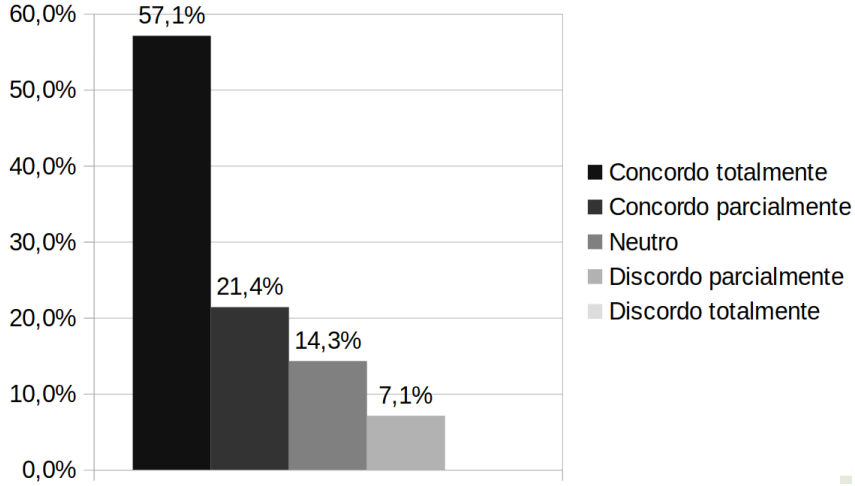
Ao analisar a Figura 12 verifica-se que 50,0% dos participantes concordam totalmente que a UMLCollab ajuda a aumentar a produtividade (esforço), com 28,6% concordando parcialmente, 14,3% neutros e 7,1% discordando parcialmente.

Quando requisitados para preencher uma escala linkert para fornecer uma nota de 1 a 5 para cada tipo de colaboração, sendo 1 para a pior e 5 para a melhor, a colaboração assíncrona obteve uma média 2,67 e a colaboração síncrona uma média de 3,87. Nesse contexto a UMLCollab, obteve uma média de 4,33, superando as outras técnicas de colaboração deste experimento. Possivelmente essa percepção pode ser oriunda ao menor nível de interrupções e maior nível interações colaborativas oferecidas pela UMLCollab.

5.6.1.11 Resolução de conflitos com a UMLCollab

A Figura 13 corrobora com o resultado anteriormente comentado, pois 57,1% dos participantes concordam totalmente que a UMLCollab facilita a resolução de conflitos, com 21,4% concordando parcialmente, 14,3% neutros e 7,1% discordando parcialmente.

Figura 13 – A UMLCollab facilita a resolução de conflitos

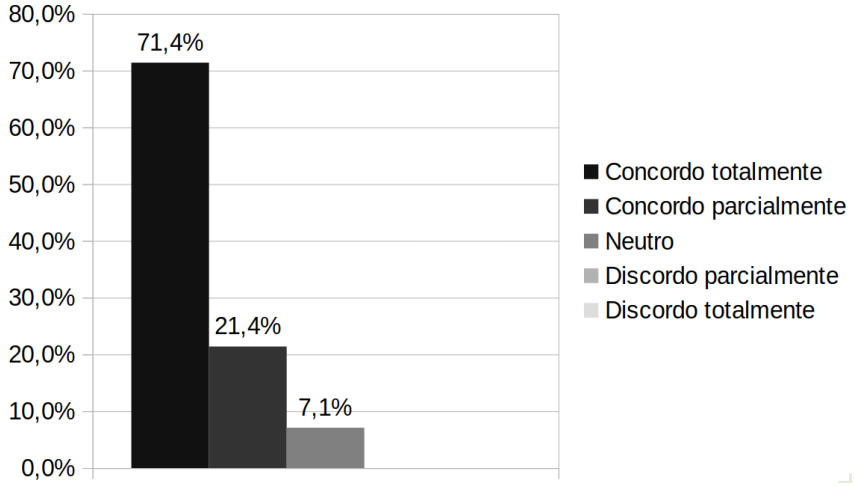


Fonte: Elaborada pelo autor

5.6.1.12 Ciência de alterações e interferências com a UMLCollab

Conforme Figura 14, na percepção dos participantes, 71,4% concordam totalmente que a UMLCollab proporcionaria ao desenvolvedor está ciente das alterações de outros colaboradores sem atrapalhar o trabalho de modelagem, sendo que apenas 21,4% concordam parcialmente e 7,1% neutros.

Figura 14 – Ciência de alterações coletivas no modelo sem interferência na modelagem individual



Fonte: Elaborada pelo autor

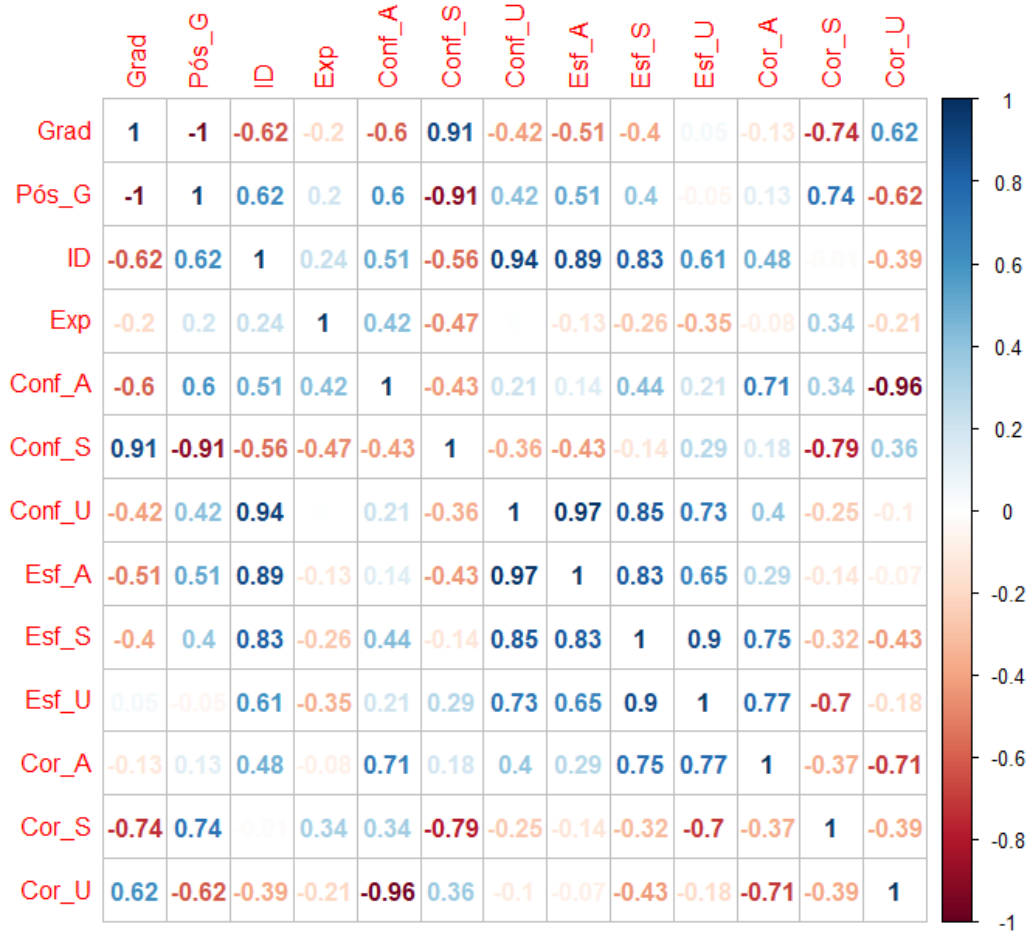
5.6.2 Teste de correlação de Pearson

A métrica mais comum de correlação na estatística é o coeficiente de correlação de Pearson. Essa métrica busca a relação linear entre duas variáveis aleatórias e vem sendo rotineiramente aplicada em vários estudos estatísticos para análise de dados e classificação (GL ESTEVES

et al., 2018).

Para verificar se existia alguma correlação entre as variáveis independentes e as variáveis dependentes foi realizado teste de correlação conforme por ser verificado na Figura 15.

Figura 15 – Correlação de Pearson das variáveis



Fonte: Elaborada pelo autor

Observa-se na Figura 15 que aparentemente existe uma forte correlação positiva entre as variáveis dependentes “número de conflitos” da UMLCollab (0,94), “esforço” da colaboração assíncrona (0,89) e “esforço” da colaboração síncrona (0,83) com a variável independente “idade” dos participantes (ID). Isto significa, de acordo a amostra do estudo, que quanto maior a idade do participante, maiores serão os números de conflitos na UMLCollab e maior será o esforço do tipo de colaboração assíncrona e da síncrona. Além disso, a Figura 15 também evidencia uma correlação moderada positiva entre a variável dependente "corretude"da UMLCollab (0,62) e uma correlação forte do "número de conflitos"da colaboração síncrona (0,91) quando correlacionadas com a variáveis de educação (graduação e pós-graduação), ou seja, percebe-se que possivelmente a educação pode influenciar nos desempenhos das técnicas de colaboração.

Para facilitar a identificação das significâncias dos coeficientes das variáveis do teste de correlação de Pearson, foi elaborada a Tabela 28. Após a análise da Tabela 28, pode-se afirmar existe uma correlação significante tanto positiva, quanto negativa, com significâncias ao nível

de 5% (p-value < 0,05), entre o número de conflitos da colaboração síncrona e variáveis de educação (Grad e Pós_G), ou seja, como um resultado anulou o outro, significa que independente do nível de educação, não foram aumentados ou diminuídos os números de conflitos desse tipo de colaboração.

Tabela 28 – Correlação de Pearson das variáveis

	Grad	Pós_G	ID	Exp	Conf_A	Conf_S	Conf_U	Esf_A	Esf_S	Esf_U	Cor_A	Cor_S	Cor_U
Grad	1.00												
Pós_G	-1.00	1.00											
ID	-0.62	0.62	1.00										
Exp	-0.20	0.20	0.24	1.00									
Conf_A	-0.60	0.60	0.51	0.42	1.00								
Conf_S	**0.91****	**-0.91****	-0.56	-0.47	-0.43	1.00							
Conf_U	-0.42	0.42	**0.94****	0.00	0.21	-0.36	1.00						
Esf_A	-0.51	0.51	**0.89****	-0.13	0.14	-0.43	0.97	1.00					
Esf_S	-0.40	0.40	**0.83***	-0.26	0.44	-0.14	0.85	0.83	1.00				
Esf_U	0.05	-0.05	0.61	-0.35	0.21	0.29	0.73	0.65	0.90	1.00			
Cor_A	-0.13	0.13	0.48	-0.08	0.71	0.18	0.40	0.29	0.75	0.77	1.00		
Cor_S	-0.74	0.74	-0.01	0.34	0.34	-0.79	-0.25	-0.14	-0.32	-0.70	-0.37	1.00	
Cor_U	0.62	-0.62	-0.39	-0.21	-0.96	0.36	-0.10	-0.07	-0.43	-0.18	-0.71	-0.39	1.00

Legenda:
Grad = Graduado; Pós_G = Pós-graduado; ID = Idade; Exp = Experiência;
Conf_A = Conflitos Assíncrono; Conf_S = Conflitos Síncrono; Conf_U = Conflitos UMLCollab;
Esf_A = Esforço Assíncrono; Esf_S = Esforço Síncrono; Esf_U = Esforço UMLCollab;
Cor_A = Corretude Assíncrona; Cor_S = Corretude Síncrona; Cor_U = Corretude UMLCollab;
*Significante a nível de 10% **Significante a nível de 5% ***Significante a nível de 1%
Fonte: Elaborada pelo autor

Outras correlações com significâncias ao nível de 5% e 10% foram as correlações entre as variáveis dependentes Conf_U (com p-value < 0,05), Esf_A (com p-value < 0,05) e Esf_S (p-value < 0,10) com a variável independente de idade. Logo, pode-se confirmar que existe relações entre o número de conflitos Conf_U e esforços das Esf_A e Esf_S e a idade dos participantes. Significando que, em relação a amostra deste estudo, quanto maior for a idade, possivelmente, maiores serão os números de conflitos da Conf_U e possivelmente, quanto maior for a idade, maiores serão os esforços da Esf_A e Esf_S.

5.6.3 Análise de regressões lineares múltiplas

Para realização dos testes de regressão linear múltipla, foram elaborados 6 modelos para com as variáveis dependentes das 3 tipos de colaboração e as independentes: educação, idade e experiência conforme listado abaixo. Esses modelos foram criados com o intuito das regressões explicarem o nível de influência de cada uma das variáveis independentes em relação as dependentes.

- **Modelo 1:** Esf_U = $\beta_0 + \beta_1 \text{Edu}_i + \beta_2 \text{ID}_i + \beta_3 \text{Exp}_i + \mu_i$
- **Modelo 2:** Esf_A = $\beta_0 + \beta_1 \text{Edu}_i + \beta_2 \text{ID}_i + \beta_3 \text{Exp}_i + \mu_i$
- **Modelo 3:** Esf_S = $\beta_0 + \beta_1 \text{Edu}_i + \beta_2 \text{ID}_i + \beta_3 \text{Exp}_i + \mu_i$
- **Modelo 4:** Conf_U = $\beta_0 + \beta_1 \text{Edu}_i + \beta_2 \text{ID}_i + \beta_3 \text{Exp}_i + \mu_i$
- **Modelo 5:** Conf_A = $\beta_0 + \beta_1 \text{Edu}_i + \beta_2 \text{ID}_i + \beta_3 \text{Exp}_i + \mu_i$
- **Modelo 6:** Conf_S = $\beta_0 + \beta_1 \text{Edu}_i + \beta_2 \text{ID}_i + \beta_3 \text{Exp}_i + \mu_i$
- **Modelo 7:** Cor_U = $\beta_0 + \beta_1 \text{Edu}_i + \beta_2 \text{ID}_i + \beta_3 \text{Exp}_i + \mu_i$
- **Modelo 8:** Cor_A = $\beta_0 + \beta_1 \text{Edu}_i + \beta_2 \text{ID}_i + \beta_3 \text{Exp}_i + \mu_i$
- **Modelo 9:** Cor_S = $\beta_0 + \beta_1 \text{Edu}_i + \beta_2 \text{ID}_i + \beta_3 \text{Exp}_i + \mu_i$

Nos modelos de 1 ao 3 foi utilizada a variável dependente **esforço** representada pelas proxies Esf_U, Esf_A e Esf_S. Nos modelos de 4 ao 6 foi utilizada a variável dependente **número de conflitos** representa pelas proxies Conf_U, Conf_A e Conf_S. E nos modelos de 7 ao 9 foi utilizada a variável dependente **corretude** representa pelas proxies Cor_U, Cor_A e Cor_S. As variáveis independentes foram representadas pelas proxies: Edu (educação dos participantes), ID (idade dos participantes) e Exp (experiências dos participantes) e o erro do modelo é ilustrado pelo símbolo μ_i.

Para validação da regressão linear múltipla foram realizados os testes de pressuposto; a normalidade dos dados foi analisada por meio do teste de Shapiro Wilk; na análise da multicolinearidade aplicou-se o teste de Variable Inflator Factor (VIF) - (Fator de Inflação da Variância); para a análise da homoscedasticidade realizou-se o Teste de Levene; foi determinada ainda a auto correlação dos resíduos por meio do teste de Durbin-Watson cujos resultados foram favoráveis para a utilização da técnica. Para a realização do teste foi utilizado o software Stata: Data Analysis and Statistical Software.

A primeira variável testada nas regressões foi a variável "esforço", conforme Tabela 29. Nota-se dos três modelos, apenas a variável idade apresentou significância para explicar os esforços da UMLCollab e colaboração síncrona, ou seja, de acordo com a amostra, a idade dos participantes influenciou em 31,23% dos resultados dos esforços da UMLCollab e influenciou em 12,41% dos esforços da técnica síncrona, isso quer dizer que estes resultados confirmam os resultados encontrados nas correlações, logo, quanto maior a idade, maiores serão o esforços. Em relação aos resultados dos modelos dos esforços testados, tanto o modelo 1 e o 3 apresentaram um bom coeficiente de determinação (46,18% e 42,36%, respectivamente), confirmando a influência da variável idade em relação ao esforço para UMLCollab e colaboração síncrona. Portanto, pode-se rejeitar a hipótese nula 4 (H_{4-0}) e aceitar a hipótese alternativa 4 (H_{4-1}) e afirmar a idade dos participantes afeta significativamente o uso da UMLCollab e da colaboração síncrona ao ponto de influenciar o esforço.

Tabela 29 – Teste de regressão linear múltipla para esforço

	Modelo 1 (Esf_U)	**Modelo 2 (Esf_A)**	**Modelo 3 (Esf_S)**
Edu	-2,2285	-0,1914	-0,2220
Sig. (p-value)	0,126	0,958	0,7190
ID	0,3123	0,3654	0,1245
Sig. (p-value)	0,023*	0,252	0,0416*
Exp	-0,8483	-0,8231	-0,3597
Sig. (p-value)	0,146	0,584	0,1819
Constant	-0,6372	3,1508	0,1268
Sig. (p-value)	0,808	0,670	0,9177
R-quadrado Ajustado	0,4618	-0,06428	0,4236
Estatística F	3,574	0,8188	3,205
Sig. (p-value)	0,08628	0,529	0,1046
Graus de Liberdade	6,000	6,000	6,000

Legenda: Edu = Educação; ID = Idade; Exp = Experiência; Esf = Esforço; A = Assíncrono; S = Síncrono; U = UMLCollab; **Significante a 5% **Significante a 1% ***Significante a 0,1%

Fonte: Elaborada pelo autor.

Para analisar a variável dependente "número de conflitos" no teste de regressão linear múltipla foi elaborada a Tabela 30.

Tabela 30 – Teste de regressão linear múltipla para número de conflitos

	Modelo 4 (Conf_U)	**Modelo 5 (Conf_A)**	**Modelo 6 (Conf_S)**
Edu	-1,47721	0,61532	-1,220538
Sig. (p-value)	0,21796	0,527	0,353
ID	0,54422	0,01804	0,007214
Sig. (p-value)	0,00082***	0,818	0,944
Exp	-0,64463	0,20504	-0,217983
Sig. (p-value)	0,18869	0,600	0,673
Constant	-6,59084	-0,01372	3,061180
Sig. (p-value)	0,02190*	0,994	0,254
R-quadrado Ajustado	0,8304	-0,1586	-0,134
Estatística F	15,69	0,5894	0,6454
Sig. (p-value)	0,003023	0,6441	0,6137
Graus de Liberdade	6,000	6,000	6,000

Legenda: Edu = Educação; ID = Idade; Exp = Experiência; Conf = Nº de conflitos; A = Assíncrono; S = Síncrono; U = UMLCollab; **Significante a 5% **Significante a 1% ***Significante a 0,1%

Fonte: Elaborada pelo autor.

Observa-se que dos três modelos (4, 5 e 6), apenas a variável idade novamente apresentou significância de 99% (p-value < 0,01) para explicar os conflitos da UMLCollab, ou seja, de acordo com a amostra desse estudo, a idade dos participantes influenciou em 54,42% dos resultados dos conflitos da UMLCollab, em outras palavras, quanto maior a idade maior os conflitos nesta proposta. Em relação aos resultados dos modelos testados dos esforços, somente o modelo 4 apresentou um excelente coeficiente de determinação (83,04%), confirmando a influência da

variável idade em relação aos conflitos na UMLCollab. Portanto, pode-se rejeitar a hipótese nula 5 (H_{5-0}) e aceitar a hipótese alternativa 5 (H_{5-0}) e afirmar que a idade dos participantes afeta significativamente o uso da UMLCollab ao ponto de influenciar o número de conflitos, ou seja, quanto menor a idade, menor serão os números de conflitos da UMLCollab.

Para a verificar o resultado da variável "corretude" foi realizado o teste de regressão linear múltipla (conforme o modelo 7, 8 e 9), porém não foi encontrada nenhuma relação significativa com qualquer variável independente deste estudo. Logo, não houve necessidade de ilustrar o resultado em formato de tabela.

6 CONCLUSÃO

Conforme a revisão da literatura desta pesquisa, a modelagem de software colaborativa teoricamente permite aumento da produtividade e qualidade dos modelos através do trabalho simultâneo de dois ou mais desenvolvedores em um mesmo modelo. As técnicas colaborativas atuais apresentam problemas crescentes em complexidade à medida que o número de desenvolvedores aumenta. A colaboração síncrona, na qual as alterações locais são imediatamente enviadas e as alterações remotas imediatamente recebidas, evita conflitos, porém, pode atrapalhar o processo cognitivo de elaboração dos modelos, pois quanto maior o número de usuários simultâneos maior quantidade de alterações remotas. Em contrapartida a colaboração assíncrona evita esses problemas, porém os conflitos são mais complexos e demorados para serem resolvidos.

Para tratar dessa problemática, foi proposta a UMLCollab com o objetivo de propor uma técnica de sincronização e resolução de conflitos para reduzir o esforço, melhorar a corretude e aumentar a eficiência dos modelos criados a partir da perspectiva de desenvolvedores no contexto de modelagem colaborativa de Software. As avaliações de inferências foram realizadas através de um experimento, que confirmou que existe uma produtividade (esforço) intermediária da UMLCollab usando colaboração híbrida em comparação a colaboração assíncrona e síncrona. Além disso, foi verificado ainda um maior nível de corretude da UMLCollab considerando o máximo das amostras de corretude coletadas em relação a colaboração síncrona. Ademais, a UMLCollab recebeu uma melhor percepção dos participantes nos resultados de produtivididade, resultados coletados a partir da análise das respostas do questionário aplicado. O questionário também evidenciou que a maioria dos participantes confirmaram a UMLCollab como facilitadora na resolução de conflitos e redução da interferência na modelagem.

Em relação ao teste de correlação de Pearson, foram encontradas relações significantes entre as variáveis idade e educação contrastando com as variáveis dependentes esforço e número de conflitos, significando que a idade possivelmente pode influenciar nos resultados do número de conflitos da UMLCollab e nos resultados dos esforços dos tipos de colaboração assíncrona e síncrona. E também confirmando que a educação não interfere nos resultados das técnicas experimentadas.

E por fim, nos resultados das regressões lineares múltiplas, foram encontrados resultados significantes que explicam que realmente a idade influenciou nos resultados dos esforços da UMLCollab e colaboração síncrona. Além disso, também foi encontrado que a idade influenciou o número de conflitos da UMLCollab, ou seja, quanto menor for a idade dos participantes, menos conflitos entre os participantes quando utilizarem esta técnica ou quanto maior a idade, maiores serão o número de conflitos.

Portanto, a partir dos resultados apurados pode-se afirmar que a UMLCollab é uma proposta inovadora que propõe expandir espaços para trabalhos futuros, pois é uma proposta intermediadora comparada com as demais técnicas existentes, mas, que pode trazer um nível de

desempenho melhor em relação as corretudes dos projetos.

6.1 Principais contribuições

Esta pesquisa apresentou duas principais contribuições: para a teoria, contribuiu com uma nova abordagem híbrida empírica para melhorar a sincronização e resolução dos conflitos e corretude na elaboração de modelos colaborativos de software, pois, a nova abordagem apresentou-se ser uma ferramenta intermediária em relação as demais ferramentas já existentes (síncrona e assíncrona). Além disso, este estudo empírico contribuiu para a literatura, pois, além de preencher a lacuna escassa, propõe um novo modelo teórico de modelagem colaborativa de software (UMLCollab).

Na prática, ou seja, para a sociedade a UMLCollab ofereceu a redução do esforço dos desenvolvedores na produção de projetos colaborativos, assim como o aumento da eficiência para as próximas modelagens de software que possivelmente serão criadas.

Portanto, a UMLCollab oferece uma abordagem de colaboração que evita a interferência do trabalho cognitivo do desenvolvedor na elaboração dos modelos ao mesmo tempo em que aumenta o nível de interação e colaboração entre os participantes. Além disso, o estudo identificou um falha na colaboração síncrona, ou seja, apesar da abordagem apresentar um menor esforço na elaboração dos modelos, este resultado foi prejudicado, pois os participantes descontinuavam a interação e colaboração entre eles ao perceber que outros participantes alteravam um elemento de forma concorrente, promovendo na prática um bloqueio indireto do elemento sendo disputado.

6.2 Limitações do trabalho e sugestões para trabalhos futuros

Uma das limitações do estudo deu-se a partir dos números de participantes e um possível viés de interação e colaboração ocorrido na técnica síncrona encontrado somente durante a observação dos vídeos dos participantes. Por isso, para trabalhos futuros, sugere-se como possível melhoria, a expansão do experimento para um maior número de equipes (por exemplo, 30 equipes) e realizar experimentos com equipes de tamanhos diferentes (por exemplo, de 4 a 6 participantes ou mais) para verificar se o tamanho da equipe pode influenciar nos resultados.

Além disso, sugere-se para os experimentos futuros a elaboração de um algoritmo para mensurar automaticamente o esforço, números de conflitos e corretude dos modelos gerados; A integração com sistemas de controle de versão permitindo reverter, comparar e realizar merge de versões anteriores dos modelos; e integração com sistemas de comunicação como chat e videoconferência, para melhorar a interação remota dos colaboradores.

REFERÊNCIAS

ALHO, K.; SULONEN, R. Supporting virtual software projects on the Web. In: SEVENTH IEEE INTERNATIONAL WORKSHOP ON ENABLING TECHNOLOGIES: INFRASTUCTURE FOR COLLABORATIVE ENTERPRISES (WET ICE'98), 1998. **Anais...** [S.l.: s.n.], 1998. p. 10–14.

ALTMANN, J. Cooperative software development: concepts, model and tools. In: TOOLS '99 PROCEEDINGS OF THE TECHNOLOGY OF OBJECT-ORIENTED LANGUAGES AND SYSTEMS, 1999. **Anais...** [S.l.: s.n.], 1999. p. 194.

BANG, J. Y. et al. Codesign: a highly extensible collaborative software modeling framework. **32nd International Conference on Software Engineering**, [S.l.], v. 2, p. 243–246, 2010.

BANG, J. young; BRUN, Y.; MEDVIDOVIĆ, N. Collaborative-design conflicts: costs and solutions. **IEEE Software**, [S.l.], v. 35, n. 6, p. 25–31, 2018.

BANG, J. young; POPESCU, D. Enabling workspace awareness for collaborative software modeling. **The Future of Collaborative Software Development at the ACM Conference on Computer Supported Cooperative Work (FutureCSD)**, [S.l.], 2012.

BARTHELMESS, P.; ANDERSON, K. M. A view of software development environments based on activity theory. **Computer Supported Cooperative Work (CSCW)**, [S.l.], v. 11, n. 1-2, p. 13–37, 2002.

BOULILA, N. Group support for distributed collaborative concurrent software modeling. **Automated Software Engineering, 2004. Proceedings. 19th International Conference on**, [S.l.], p. 422–425, 2004.

BRAMBILLA, M.; CABOT, J.; WIMMER, M. Model-driven software engineering in practice. **Synthesis Lectures on**, [S.l.], 2012.

BRIERTON, S. et al. A comparison of higher order thinking skills demonstrated in synchronous and asynchronous online college discussion posts. **NACTA Journal**, [S.l.], v. 60, n. 1, p. 14, 2016.

BROSCH, P. et al. We can work it out: collaborative conflict resolution in model versioning. **ECSCW 2009**, [S.l.], 2009.

CICCHETTI, A. et al. Towards a framework for distributed and collaborative modeling. In: IEEE INTERNATIONAL WORKSHOPS ON ENABLING TECHNOLOGIES: INFRASTRUCTURES FOR COLLABORATIVE ENTERPRISES, 18., 2009. **Anais...** [S.l.: s.n.], 2009. p. 149–154.

COSTA, C.; MURTA, L. Version control in distributed software development: a systematic mapping study. **Proceedings of IEEE 8th International Conference on Global Software Engineering - ICGSE'13**, [S.l.], p. 90–99, 2013.

DAM, H. K.; GHOSE, A. An agent-based framework for distributed collaborative model evolution. **12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution**, [S.l.], p. 121–130, 2011.

De Lucia, A. et al. Enhancing collaborative synchronous uml modelling with fine-grained versioning of software artefacts. **Journal of Visual Languages and Computing**, [S.l.], v. 18, n. 5, p. 492–503, 2007.

DIRIX, M.; MULLER, A.; ARANEGA, V. Genmymodel: an online uml case tool. In: ECOOP, 2013. **Anais...** [S.l.: s.n.], 2013.

DULLEMOND, K.; GAMEREN, B. van; SOLINGEN, R. van. Collaboration spaces for virtual software teams. **IEEE Software**, [S.l.], v. 31, n. 6, p. 47–53, 2014.

FARIAS, K. Empirical evaluation of effort on composing design models. In: ACM/IEEE 32ND INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2010., 2010. **Anais...** [S.l.: s.n.], 2010. v. 2, p. 405–408.

FARIAS, K. et al. Evaluating the effort of composing design models: a controlled experiment. **Software & Systems Modeling**, [S.l.], v. 14, n. 4, p. 1349–1365, 2015.

FRANZAGO, M. et al. Collaborative model-driven software engineering: a classification framework and a research map. **IEEE Transactions on Software Engineering**, [S.l.], v. 44, n. 12, p. 1146–1175, 2018.

GHIOTTO, G. et al. On the nature of merge conflicts: a study of 2,731 open source java projects hosted by github. **IEEE Transactions on Software Engineering. IEEE**, [S.l.], 2018.

GL ESTEVES, A. et al. Avaliação de modelos de preditivos de regressão para estimar esforço de software. **Revista de Sistemas e Computação-RSC**, [S.l.], v. 8, n. 2, 2018.

GRAY, J.; RUMPE, B. **The importance of flow in software development**. [S.l.]: Springer, 2017.

JIMéNEZ, M.; PIATTINI, M.; VIZCAíNO, A. Challenges and improvements in distributed software development: a systematic review. **Advances in Software Engineering**, [S.l.], v. 2009, p. 1–14, 2009.

JOLAK, R. et al. Does distance still matter? revisiting collaborative distributed software design. **IEEE Software**, [S.l.], v. 35, n. 6, p. 40–47, 2018.

KOLOVOS, D. S. et al. A research roadmap towards achieving scalability in model driven engineering. In: WORKSHOP ON SCALABILITY IN MODEL DRIVEN ENGINEERING, 2013. **Proceedings...** [S.l.: s.n.], 2013. p. 2.

KRUSCHE, S.; BRUEGGE, B. Model-based real-time synchronization. In: INTERNATIONAL WORKSHOP ON COMPARISON AND VERSIONING OF SOFTWARE MODELS (CVSM14), 2014. **Anais...** [S.l.: s.n.], 2014.

KURYAZOV, D.; WINTER, A. Collaborative modeling empowered by modeling deltas. In: INTERNATIONAL WORKSHOP ON (DOCUMENT) CHANGES: MODELING, DETECTION, STORAGE AND VISUALIZATION - DCHANGES 2015, 3., 2015. **Anais...** [S.l.: s.n.], 2015. p. 1–6. (DChanges 2015).

LEROUX, D.; NALLY, M.; HUSSEY, K. Rational software architect: a tool for domain-specific modeling. **IBM systems journal**, [S.l.], v. 45, n. 3, p. 555–568, 2006.

LEVINE, D. M.; BERENSON, M. L.; STEPHAN, D. **Estatística**: teoria e aplicações-usando microsoft excel português. [S.l.]: Ltc, 2005.

LIN, Q. et al. Multiuser collaborative work in virtual environment based case tool. **Information and Software Technology**, [S.l.], v. 45, n. 5, p. 253–267, 2003.

LIU, S. et al. Real-time collaborative software modeling using UML with rational software architect. **and Worksharing, 2006. . . .**, [S.l.], 2006.

LUCAS, E. M. et al. Collabrdl: a language to coordinate collaborative reuse. **Journal of Systems and Software**, [S.l.], v. 131, p. 505–527, 2017.

MUCCINI, H.; BOSCH, J.; VAN DER HOEK, A. Collaborative modeling in software engineering. **IEEE Software**, [S.l.], v. 35, n. 6, p. 20–24, 2018.

OSMAN, H. **Web-based collaborative software modeling**. 2013. Tese (Doutorado em Ciência da Computação) — Citeseer, 2013.

PAULA, G. C. C. d. Metodologia da pesquisa científica. **Goiânia, editora Vieira**, [S.l.], 2010.

PEREZ-SOLER, S.; GUERRA, E.; LARA, J. de. Collaborative modeling and group decision making using chatbots in social networks. **IEEE Software**, [S.l.], v. 35, n. 6, p. 48–54, 2018.

POLANČIČ, G.; JOŠT, G. The impact of the representatives of three types of process modeling tools on modeler's perceptions and performance. **Journal of Software: Evolution and Process**, [S.l.], v. 28, n. 1, p. 27–56, 2016.

PORTILLO-RODRÍGUEZ, J. et al. Tools used in global software engineering: a systematic mapping review. **Information and Software Technology**, [S.l.], v. 54, n. 7, p. 663–685, 2012.

PREVEDELLO, P. Ambiente virtual colaborativo para apoio ao ensino de circuitos integrados baseado em equipe. **Repositório Digital da UFSM**, [S.l.], 2018.

PRIKLADNICKI, R.; CARMEL, E. Is time-zone proximity an advantage for software development? the case of the brazilian it industry. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2013., 2013. **Proceedings...** [S.l.: s.n.], 2013. p. 973–981.

ROCHA, R. G. d. C. Uma abordagem baseada em ontologias e raciocínio baseado em casos para apoiar o desenvolvimento distribuído de software. **UNIVERSIDADE FEDERAL DE PERNAMBUCO**, [S.l.], 2015.

SARMA, A.; VAN DER HOEK, A. Towards awareness in the large. In: IEEE INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING (ICGSE'06), 2006. **Anais...** [S.l.: s.n.], 2006. p. 127–131.

SHEN, H. et al. Distributed Constraints Maintenance in Collaborative UML Modeling Environments. **Proceedings of the 2008 23rd IEEE/ACM**, [S.l.], 2008.

SOUZA COSTA, C. de et al. Recommending participants for collaborative merge sessions. **IEEE Transactions on Software Engineering**, [S.l.], 2019.

SUZUKI, J.; YAMAMOTO, Y. Softdock: a distributed collaborative platform for model-based software development. In: TENTH INTERNATIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS. DEXA 99, 1999. **Anais...** [S.l.: s.n.], 1999. p. 672–676.

VESIN, B.; JOLAK, R.; CHAUDRON, M. R. V. Octouml: an environment for exploratory and collaborative software design. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING COMPANION (ICSE-C), 39., 2017. **Anais...** [S.l.: s.n.], 2017. p. 7–10.

WATERIDGE, J. The role of configuration management in the development and management of information systems/technology (is/it) projects. **International Journal of Project Management**, [S.l.], v. 17, n. 4, p. 237–241, 1999.

XU, D. et al. Distributed collaborative modeling support system associating UML diagrams with chat messages. In: ANNUAL IEEE INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 33., 2009. **Anais...** [S.l.: s.n.], 2009. v. 1, p. 367–372.

ZHU, N. et al. Pounamu: a meta-tool for exploratory domain-specific visual language tool development. **Journal of Systems and Software**, [S.l.], v. 80, n. 8, p. 1390–1407, 2007.

APÊNDICE A – LISTA DE ESTUDOS PRIMÁRIOS SELECIONADOS NA REVISÃO DA LITERATURA

Tabela 31 – Lista de estudos primários

Ano	**Referência**
2018	Bang, Brun e Medvidović (2018)
2018	Jolak et al. (2018)
2018	Perez-Soler, Guerra e Lara (2018)
2015	Farias et al. (2015)
2015	Kuryazov e Winter (2015)
2014	Krusche e Bruegge (2014)
2012	Bang e Popescu (2012)
2012	Xu et al. (2009)
2011	Dam e Ghose (2011)
2010	Bang et al. (2010)
2009	Brosch et al. (2009)
2009	Cicchetti et al. (2009)
2009	Shen et al. (2008)
2007	De Lucia et al. (2007)
2007	Zhu et al. (2007)
2006	Liu et al. (2006)
2005	Boulila (2004)
2003	Lin et al. (2003)

Fonte: Elaborada pelo autor.

APÊNDICE B – AS *STRINGS* DE PESQUISA

Tabela 32 – *Strings* de pesquisa

Base de dados	***String* de Pesquisa**	**Outros Filtros**
ACM Digital Library	((Collaborative) AND (Software OR Diagram) AND (Modeling)) -agile -Bioconductor -Social -Simulation -Global -CCPN -online -shuttle -networks -comprehension -markov -estimating -versioning -secure -IBM	
CiteSeerX Library	text:collaborative AND title:diagrams AND abstract:(diagrams colaborative) AND keyword:(modeling uml collaborative)	
Google Scholar	*String* 1: "Collaborative Software Modeling"*String* 2: ((Collaborative) AND (Software AND "UML diagram") AND (Modeling)) -agile -Bioconductor -Social -Simulation -Global -CCPN -online -shuttle -networks -comprehension -markov -estimating -versioning -secure	Resultados filtrados pelo idioma inglês com exclusão de citações e patentes.
IEEE Xplore Digital Library	((Collaborative OR Cooperative OR Team OR Contribute OR Distributed) AND (Software OR Application OR System OR Diagram) AND (Modeling OR Development))	
IET Digital Library	Collaborative AND "Software Modeling"	Resultados filtrados para retornar somente Article e ConferencePaper.
Science Direct	(Collaborative AND "Software Modeling")[All Sources(Computer Science)].	Resultados filtrados para somente a área Computer Science .
Springer Link	'Collaborative AND Software AND Modeling' within Computer Science Software Engineering Computer Science, general Article	
Wiley Online Library	(Collaborative AND "Software Modeling")[All Sources(Computer Science)].	Resultados restrido somente para publication type journals.

Fonte: Elaborada pelo autor.

APÊNDICE C – CONFIGURAÇÃO DO AMBIENTE DO EXPERIMENTO

1 Lista de aplicativos

- Aplicativo Kazam v1.4.5
- IBM Rational Software Architect Designer V9.6 Evaluation Multilingual Multiplatform Set up (SAD_9.6_EVL_Setup.zip 800 MB)
- IBM Rational Software Architect Designer V9.6 Evaluation Multilingual Multiplatform Core Part 1 (SAD_9.6_EVL_1.zip 3743 MB)
- JavaHL (http://subclipse.tigris.org/wiki/JavaHL)
- OctoUML v0.6.2 (https://github.com/mlxavier/OctoUML/releases)
- Subclipse 4.2.4 (plugin para o eclipse, base do IBM RSAD)
- VirtualBox 6.0.4 (https://download.virtualbox.org/virtualbox/5.2.10)
- VirtualBox 6.0.4 Oracle VM VirtualBox Extension Pack (https://download.virtualbox.org/virtualbox/5
- Ubuntu Mate 18.04 Desktop 64-bits (https://ubuntu-mate.org/download)

2 Configurações da máquina hospedeira do virtuabox

- Instalar VirtualBox e VirtualBox Oracle VM VirtualBox Extension Pack;
- Instalar Ubuntu Mate dentro do VirtualBox com as configurações:
  - Disco rígido virtual do tipo VDI dinamicamente alocado em 140 gb;
  - Memória base 4096 MB;
  - 2 Processadores lógicos sem restrição de execução (100
  - Memória de vídeo de 128 MB;
  - Placa de rede em modo bridge.
  - No aplicativo de instalação do Ubuntu Mate informar: nome: vm, nome de usuário: vm e login automático habilitado.

3 Na máquina convidada do virtuabox (Ubuntu Mate)

- Atualizar o sistema operacional com apt-get update; apt-get dist-upgrade; init 6;

- Instalação do Aplicativo Kazam para gravação da tela

  sudo apt-get install kazam;

  Criar atalhos para o Kazam na área de trabalho (menu, digitar "kazam"e no menu de contexto em cima do ícone do kazam e escolher pin to desktop);

  Alterar as preferências do Kazam no menu Arquivo \ Preferências \ Aba Screencast: ativar “Automatic file saving“ e campo “Filename prefix“ alterar para o nome do usuário.

- Remover bloqueio da proteção de tela: Ir em Menu \ Preferências \ Proteção de Tela e desmarcar “Bloquear a tela quando a proteção de tela estiver ativa“;

- Oracle JAVA JDK

  sudo add-apt-repository ppa:webupd8team/java; sudo apt-get update;

  sudo apt-get install software-properties-common;

  sudo apt-get install oracle-java8-installer; sudo apt-get install oracle-java8-set-default;

- Subversion:

  sudo apt-get install subversion libsvn-java

- OctoUML

  mover para pasta /usr/local: sudo mv OctoUML-lastest-release /usr/local/

  sudo chmod o+rx /usr/local/OctoUML-latest-release

  criar atalho na área de trabalho: botão direito do mouse na área de trabalho, escolher “criar lançador“, informar nome “OctoUML“ e comando “java -jar /usr/local/OctoUML-latest-release.jar“;

- Download dos modelos base pequeno e grande do OctoUML e RSAD

  svn checkout http://mlxapps.com/svn/umlcollab/trunk umlcollab.svn

- Copiar scripts de manutenção para pasta home

- Opcionalmente, realizar snapshot da máquina virtual

- LTSP com os comandos:

  Método 1:

  sudo add-apt-repository –yes ppa:ts.sch.gr; sudo apt update;

  sudo apt install ltsp-manager

  Dependências: sudo apt-get install python-twisted python-dbus python-gi

  service nbd-server restart

No LTSP Manager, menu Server , clicar em Initial setup

Fazer logoff e login novamente ou reiniciar.

Atualizar imagem do LTSP a ser usada pelos clientes:

No LTSP Manager, menu Server \ LTSP image, clicar em Update

Método 2:

sudo nano /etc/apt/sources.list

adicionar: deb http://ftp.de.debian.org/debian experimental main

sudo add-apt-repository –yes ppa:ts.sch.gr; sudo apt update; apt install ltsp-manager

Dependências: sudo apt-get install python-twisted python-dbus python-gi

sudo apt purge –yes –auto-remove mate-hud snapd; sudo apt install –yes synaptic

sudo apt install –yes –install-recommends ltsp-server-standalone ltsp-client epoptes

sudo ltsp-update-image –cleanup /; sudo ltsp-config dnsmasq –enable-dns

Passos adicionais:

No ltsp manager, criar contas de acordo com a quantidade de participantes: File > Create accounts

Executar sudo ./createBranchAndSynchSvn.sh <novoramo>

- Opcionalmente, realizar snapshot da máquina virtual
- IBM RSAD

  Executar sudo ./imLauncherLinux.sh

  Desmarcar "IBM WebSphere Application Server Liberty Core".

  Criar atalhos para o IBM RSAD (clicar em menu, digitar ibm e no menu de contexto em cima do ícone do ibm rsad escolher pin to desktop)

  Configurar a dependência javahl do subversion:

  Encontrar pasta de instalação da dependência javahl:

  sudo find / -name libsvnjavahl-1.so

  sudo nano /opt/IBM/SDP/eclipse.ini

  Adicionar a linha abaixo após -vmargs:

  Criação de modelo para Treinamento no RSAD: Menu de contexto New \ Projeto de Modelo, manter selecionado "Criar a partir de modelo padrão", next, categoria Análise e Design, Pacote de Análise em Branco, next, Tipo de pacote "pacote", next, deselecionar customização, finish. Close Project.

Instalar Subclipse: Menu Help \ Eclipse Marketplace \ digitar subclipse, selecionar e clicar em "Install". Deixar marcado somente “Subclipse (required)” e Java HL.

Importar projeto do umlCollab

- Preparativos para as contas dos participantes do experimento

  Mover atalhos para a área de trabalho dos participantes

  cd "/home/vm/Área de Trabalho";

  sudo mkdir -p "/home/<participante>/Área de Trabalho"

  sudo cp *.desktop "/home/<participante>/Área de Trabalho"

- Atualizar imagem do LTSP a ser usada pelos clientes:

  No LTSP Manager ir no menu Server > LTSP image > Update menu ou sudo ltsp-update-image -c /

4 Máquina cliente LTSP de cada participante

- Alterar as preferências do Kazam no menu Arquivo \ Preferências \ Aba Screencast: ativar "Automatic file saving"e campo "Filename prefix"alterar para o nome do usuário

- Remover bloqueio da proteção de tela: Ir em Menu \ Preferências \ Proteção de Tela e desmarcar "Bloquear a tela quando a proteção de tela estiver ativa";

- IBM RSAD

  Criação de modelo para Treinamento no RSAD: Menu de contexto New \ Projeto de Modelo, manter selecionado "Criar a partir de modelo padrão", next, categoria Análise e Design, Pacote de Análise em Branco, next, Tipo de pacote "pacote", next, deselecionar customização, finish. Close Project.

  Instalar Subclipse: Menu Help \ Eclipse Marketplace \ digitar subclipse, selecionar e clicar em "Install". Deixar marcado somente “Subclipse (required)” e Java HL.

  Importar projeto do umlCollab.

  Trocar para o ambiente de sincronização de equipes indo em:

  Menu Window \ Perspective \ Open Perspective \ Other \ Team Syncronizing.

  Na aba Syncronize, clicar no botão Syncronize SVN.

  Volte para o ambiente de modelagem indo em:

  Menu Window \ Perspective \ Open Perspective \ Other \ Modelagem.

APÊNDICE D – LISTA DE ATIVIDADES DO EXPERIMENTO

Participante 01

Questão 01:

Em um sistema usado por uma empresa de extração de petróleo, execute as seguintes atividades:

( ) Na classe “PetroleoData”, trocar o tipo do atributo “gasLift” para “PreSalCore”;

( ) Na classe “PetroleoData”, trocar o tipo de retorno do método “getSimulation Data()” para “String”; e

( ) Na classe “PreSalCore”, trocar tipo do atributo “data” para “Date”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “-gasLift: PreSalCore” da classe “PetroleoData”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 02:

Em uma aplicação Web para uma concessionária de veículos, execute as seguintes atividades:

( ) Na classe “FactoryClient”, altere o nome do atributo “report” para “invoice”;

( ) Na classe “FactoryClient”, remova o método “exportData()”; e

( ) Na classe “Car”, altere o tipo do atributo “price” para “int”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “-.price: int” da classe “Car”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 03:

Em uma aplicação financeira para um banco, execute as seguintes atividades:

( ) Na classe “ATM”, crie o atributo “- location: char”;

( ) Na classe “ATM”, altere o tipo de retorno do método “providePIN()” para “String”; e

( ) Na classe “Account”, mantenha o atributo “- description: boolean”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “- description: boolean” da classe “Account”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 04:

Em um sistema de controle acadêmico, execute as seguintes atividades:

( ) Na classe “Faculty”, trocar o tipo do atributo “classType” para “String”;

( ) Na classe “Faculty”, trocar o tipo de retorno do método “addStudent ()” para “String”; e

( ) Na classe “Register”, trocar tipo do atributo “registerType” para “float”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “-classType: String” da classe “Faculty”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 05:

Em um sistema de controle financeiro, execute as seguintes atividades:

( ) Na classe “Bank”, altere o nome do atributo “cachAmount” para “amount”;

( ) Na classe “Bank”, remova o método “getBalance()”; e ( ) Na classe “Customer”, altere o tipo do atributo “payment” para “int”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “- payment: int” da classe “Customer”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 06:

Em uma aplicação financeira para um banco, execute as seguintes atividades:

( ) Na classe “FluidControlPanel”, crie o atributo “- selectedFluid: char”;

( ) Na classe “FluidControlPanel”, altere o tipo de retorno do método “detailAnalysis()” para “String”; e

( ) Na classe “Core”, mantenha o atributo “- outOfSync: int”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o método “+ detailAnalysis(): String” da classe FluidControlPanel. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Participante 02:

Questão 01:

Em um sistema usado por uma empresa de extração de petróleo, execute as seguintes atividades:

( ) Na classe “PetroleoData”, trocar o tipo do atributo “gasLift”para “char”;

( ) Na classe “PetroleoData”, trocar o tipo de retorno do método “getSimulation Data()” para “void”; e

( ) Na classe “PreSalCore”, trocar tipo do atributo “data” para “char”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o método “+ getSimulationData(): void” da classe “PetroleoData”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 02:

Em uma aplicação Web para uma concessionária de veículos, execute as seguintes atividades:

( ) Na classe “FactoryClient”, remova o atributo “report”;

( ) Na classe “FactoryClient”, altere o tipo de retorno do método “exportData()” para “String”; e

( ) Na classe “Car”, remova o atributo “price”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o método “+ exportData(): String” da classe “FactoryClient”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 03:

Em uma aplicação financeira para um banco, execute as seguintes atividades:

( ) Na classe “ATM”, altere o tipo do atributo “location” para “boolean”;

( ) Na classe “ATM”, mantenha o método “- providePIN(): int”; e

( ) Na classe “Account”, crie o atributo “- description: String”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “- location: boolean” da classe ATM. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 04:

Em um sistema de controle acadêmico, execute as seguintes atividades:

( ) Na classe “Faculty”, trocar o tipo do atributo “classType” para “char”;

( ) Na classe “Faculty”, trocar o tipo de retorno do método “addStudent()” para “void”; e

( ) Na classe “Register”, trocar tipo do atributo “registerType” para “char”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o método “+ addStudent(): void” da classe “Faculty”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 05:

Em uma aplicação Web para uma concessionária de veículos, execute as seguintes atividades:

( ) Na classe “Bank”, remova o atributo “cachAmount”;

( ) Na classe “Bank”, altere o tipo de retorno do método “getBalance()” para “double”; e

( ) Na classe “Customer”, remova o atributo “payment”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o método “+ getBalance(): double” da classe “Bank”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 06:

Em um sistema de controle de fluidos, execute as seguintes atividades:

( ) Na classe “FluidControlPanel”, altere o tipo do atributo “selectedFluid” para “double”;

( ) Na classe “FluidControlPanel”, mantenha o método “+ detailAnalysis(): int”; e

( ) Na classe “Core”, crie o atributo “- outOfSync: float”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “- outOfSync: float” da classe Core. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Participante 03

Questão 01:

Em um sistema usado por uma empresa de extração de petróleo, execute as seguintes atividades:

( ) Na classe “PetroleoData”, trocar o tipo do atributo “gasLift”para “String”;

( ) Na classe “PetroleoData”, trocar o tipo de retorno do método “getSimulation Data()” para “char”; e

( ) Na classe “PreSalCore”, trocar tipo do atributo “data” para “String”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “- data: String” da classe “PreSalCore”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 02:

Em uma aplicação Web para uma concessionária de veículos, execute as seguintes atividades:

( ) Na classe “FactoryClient”, altere o tipo do atributo “report” para “String”;

( ) Na classe “FactoryClient”, altere o nome do método “exportData()” para “exportReport()”; e

( ) Na classe “Car”, altere o nome do atributo “price” para “value”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “- report: String” da classe “FactoryClient”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 03:

Em uma aplicação financeira para um banco, execute as seguintes atividades:

( ) Na classe “ATM”, mantenha o atributo “- location: int”;

( ) Na classe “ATM”, crie o método “- providePIN: void”; e

( ) Na classe “Account”, altere o tipo do atributo “description” para “char”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o método “+ providePIN(): void” da classe “ATM”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 04:

Em um sistema de controle acadêmico, execute as seguintes atividades:

( ) Na classe “Faculty”, trocar o tipo do atributo “classType” para “flloat”;

( ) Na classe “Faculty”, trocar o tipo de retorno do método “addStudent()” para “char”; e

( ) Na classe “Register”, trocar tipo do atributo “registerType” para “int”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “- registerType: int” da classe “Register”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 05:

Em uma aplicação Web para uma concessionária de veículos, execute as seguintes atividades:

( ) Na classe “Bank”, altere o tipo do atributo “cachAmount” para “int”;

( ) Na classe “Bank”, altere o nome do método “getBalance()” para “exportBalance”; e

( ) Na classe “Customer”, altere o nome do atributo “payment” para “balance”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o atributo “- cachAmount: int” da classe “Bank”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

Questão 06:

Em uma aplicação financeira para um banco, execute as seguintes atividades:

( ) Na classe “FluidControlPanel”, mantenha o atributo “- selectedFluid: int”;

( ) Na classe “FluidControlPanel”, crie o método “+ detailAnalysis(): String”; e

( ) Na classe “Core”, altere o tipo do atributo “outOfSync” para “double”.

Objetivo: A versão final do modelo após a resolução de possíveis conflitos, deverá manter o método “+ selectedFluid (): int” da classe “FluidControlPanel”. Verifique se os demais participantes atingiram seus objetivos para passar a próxima questão.

APÊNDICE E – QUESTIONÁRIO

1 Dados gerais

1.1 Grupo:
1.2 Participante: [ ] p01 [ ] p02 [ ] p03
1.3 Nome:
1.4 E-mail:
1.5 Idade:
1.6 Nível acadêmico:
[ ] Graduação [ ] Doutorado [ ] Mestrado [ ] Pós-doutorado
[ ] Incompleto [ ] Completo
1.7 Profissão:
1.8 Experiencia em modelagem de software com UML (anos):

2 Informações do experimento

2.1 Forneça uma nota de como você avalia a produtividade da equipe usando a colaboração síncrona com o OctoUML modo síncrono (1 é pior e 5 é melhor):
[ ] 1 [ ] 2 [ ] 3 [ ] 4 [ ] 5
2.1.1 Justifique a nota fornecida em comparação as demais ferramentas.

2.2 Forneça uma nota de como você avalia a produtividade da equipe usando a colaboração assíncrona com o IBM RSAD (1 é pior e 5 é melhor):
[ ] 1 [ ] 2 [ ] 3 [ ] 4 [ ] 5
2.2.1 Justifique a nota fornecida em comparação as demais ferramentas.

2.3 Forneça uma nota de como você avalia a produtividade da equipe usando o UMLCollab com o OctoUML modo UMLCollab (1 é pior e 5 é melhor):
[ ] 1 [ ] 2 [ ] 3 [ ] 4 [ ] 5
2.3.1 Justifique a nota fornecida em comparação as demais ferramentas.

3 Percepção da facilidade de uso (PEU)

3.1 Eu achei a UMLCollab fácil de usar
[ ] Concordo Totalmente [ ] Concordo Parcialmente [ ] Neutro [ ] Discordo Parcialmente [ ] Discordo Totalmente

3.2 Eu achei a UMLCollab fácil de aprender

[ ] Concordo Totalmente [ ] Concordo Parcialmente [ ] Neutro [ ] Discordo Parcialmente [ ] Discordo Totalmente

3.3 Eu achei a UMLCollab fácil de dominar.

[ ] Concordo Totalmente [ ] Concordo Parcialmente [ ] Neutro [ ] Discordo Parcialmente [ ] Discordo Totalmente

4 Percepção de utilidade

4.1 A UMLCollab facilita a resolução de conflitos.

[ ] Concordo Totalmente [ ] Concordo Parcialmente [ ] Neutro [ ] Discordo Parcialmente [ ] Discordo Totalmente

4.2 A UMLCollab ajuda a aumentar a produtividade.

[ ] Concordo Totalmente [ ] Concordo Parcialmente [ ] Neutro [ ] Discordo Parcialmente [ ] Discordo Totalmente

4.3 A UMLCollab fornecerá ao desenvolvedor o conhecimento das mudanças de outros colaboradores sem prejudicar seu trabalho de modelagem.

[ ] Concordo Totalmente [ ] Concordo Parcialmente [ ] Neutro [ ] Discordo Parcialmente [ ] Discordo Totalmente

5. Atitude

5.1 Usar a UMLCollab é uma boa ideia.

[ ] Concordo Totalmente [ ] Concordo Parcialmente [ ] Neutro [ ] Discordo Parcialmente [ ] Discordo Totalmente

6.1 Estou confiante em relação à UMLCollab

[ ] Concordo Totalmente [ ] Concordo Parcialmente [ ] Neutro [ ] Discordo Parcialmente [ ] Discordo Totalmente

6 Intenção de Comportamento (BI)

6.1 Eu pretendo usar a UMLCollab para desenvolver modelos de forma colaborativa.

[ ] Concordo Totalmente [ ] Concordo Parcialmente [ ] Neutro [ ] Discordo Parcialmente [ ] Discordo Totalmente

7 Finalização

7.1 Coloque aqui suas observações gerais sobre o experimento, sugestões e outros: