VI EIGEDIN
Encontro Internacional de Gestão, Desenvolvimento e Inovação
16 a 19 de novembro de 2022
Edição Online
ISSN: 2594-8083
NOVAS DINÂMICAS DA SOCIEDADE:
desafios e soluções

IMPLEMENTAÇÃO DE RECURSOS E TÉCNICAS DE TRABALHO COLABORATIVO E UMA PLATAFORMA DE INTERVENÇÕES BASEADA NO MÉTODO ESM

Laurentino Augusto Dantas
USP - Universidade de São Paulo / IFMS-Inst. Fed. do Mato Grosso do Sul
laurentino.dantas@usp.br

Joab Cavalcante da Silva
UEMS - Universidade Estadual de Mato Grosso do Sul
joabms@gmail.com

Maria da Graça Campos Pimentel
USP - Universidade de São Paulo
mgp@icmc.usp.br

RESUMO

A plataforma ESPIM permite que especialistas implementem programas de intervenções programadas através de dispositivos móveis, os fundamentos da plataforma são baseados no método ESM, além de utilizar uma interface de desenvolvimento de programas por usuário final. No presente trabalho é descrita a atividade de implementar na plataforma ESPIM, recursos e técnicas de suporte ao trabalho colaborativo. São descritas em forma de um roteiro todas as técnicas, algoritmos e ferramentas de TI empregadas para que o objetivo do trabalho fosse alcançado. Os testes realizados demonstraram que a implementação das técnicas descritas, dotam o ESPIM da capacidade de edição colaborativa. A relevância do trabalho reside no fato de que ele descreve um roteiro simples, que se corretamente adaptado, pode ser utilizado por desenvolvedores para dotarem sistemas computacionais da capacidade de edição colaborativa e consequentemente, suporte ao trabalho colaborativo.

Palavras-chave: Método ESM; Edição Colaborativa; Colaboração.

ISSN: 2594-8083

1 INTRODUÇÃO

O Experience Sampling Method (ESM) é um procedimento de pesquisa para estudar o que as pessoas fazem, sentem e pensam durante sua vida diária. Consiste em pedir aos indivíduos que forneçam auto-relatos sistemáticos em ocasiões aleatórias durante as horas de vigília de uma semana normal (LARSON e CSIKSZENTMIHALYI, 2014). Conjuntos desses auto relatos de uma amostra de indivíduos criam um arquivo de experiência diária. Do ponto de vista de TI, um programa de acompanhamento ESM demanda em criar uma série de situações (eventos) no quais os dados do indivíduo observado serão coletados. Em um evento de coleta uma série de dados podem ser coletados por sensores, ou o indivíduo pode responder a perguntas (intervenções). Os eventos podem ser programados para ocorrerem de forma regular, em determinado dia e horário, ou podem ser disparados quando determinada situação específica ocorrer, como por exemplo o indivíduo ir a determinado local, ou realizar determinada atividade que possa ser detectada por algum tipo de sensor, como a localização do telefone, por exemplo.

A plataforma Espim (ZAINE et al., 2016), visa apoiar pesquisadores ou profissionais na criação e acompanhamento de programas de intervenção interativa, que lhes permitam interagir de forma assíncrona com seu público-alvo remotamente em seus ambientes naturais. O método pode ser utilizado com o suporte da plataforma ESPIM, que é baseada na web para autoria e baseada no Android para consumo (RODRIGUES et al., 2018). Os programas de intervenção ESPIM são projetados por especialistas no domínio e reproduzidos como um documento multimídia interativo em smartphones usados por seus usuários-alvo. Neste contexto, os especialistas de domínio são os autores do documento multimídia interativo e seus usuários-alvo são os consumidores.

Do ponto de vista da engenharia de documentos, os programas de intervenção ESPIM são documentos declarativos que são interpretados e processados por um aplicativo baseado em reprodutor quando acessados pelos usuários. Além disso, os programas de intervenção são documentos não textuais, uma vez que podem apresentar arquivos de texto, imagens, vídeos e áudio. Os programas de intervenção podem também conter formulários preenchidos e usados pelos especialistas do domínio para fazer perguntas e coletar informações dos usuários. Além do mais, os programas de intervenção também são documentos dinâmicos, pois sua apresentação pode depender da seleção de alternativas fornecidas pelo usuário. Além disso, os programas de intervenção são documentos centrais: eles podem iniciar aplicativos externos e receber dados correspondentes à interação do usuário com tais aplicativos. As intervenções são

ISSN: 2594-8083

construídas na forma de fluxos não lineares: um determinado fluxo é executado como resultado da interação do usuário com o programa de intervenção, o que é possível porque diferentes respostas a uma questão de múltipla escolha de alternativa pode levar a diferentes fluxos subsequentes da intervenção.

Um dos grandes problemas enfrentados pelos especialistas que utilizam o ESPIM é que ele é uma ferramenta que não apresenta suporte à edição colaborativa síncrona, sendo assim, neste trabalho descrevemos o processo e os recursos de TI seguidos, adotados e desenvolvidos de modo a permitir que os profissionais possam trabalhar de forma colaborativa na plataforma ESPIM. Apesar do trabalho estar voltado à plataforma ESPIM, ele se torna relevante devido ao fato de que as soluções adotadas, e descritas, podem ser utilizadas como um roteiro para converter sistemas não colaborativos em sistemas que dão suporte ao trabalho colaborativo.

O restante deste artigo está organizado da seguinte forma. Na Seção 2, apresentamos trabalhos relacionados. Na seção 3, mostramos a utilização de programas de intervenção por meio de um cenário aplicado ao domínio da saúde. Na Seção 4, descrevemos os elementos utilizados de modo a dar suporte à edição colaborativa na plataforma ESPIM. Na Seção 5, descrevemos os testes realizados bem como os resultados obtidos . Na seção 6, apresentamos nossas considerações finais e apontamos para trabalhos futuros.

2 REVISÃO DE LITERATURA

Rough e Quigle (2020) partindo do princípio que a adoção da EUD por usuários finais depende de vários fatores contextuais que não são bem compreendidos. Realizaram uma pesquisa visando determinar quais os fatores que influenciariam pesquisadores a adotar o desenvolvimento de aplicações do método ESM a partir de aplicativos para dispositivos móveis. Como resultado da pesquisa estabeleceram uma série de recomendações para o design de ferramentas EUD, de modo a permitir que não-programadores desenvolvam aplicativos para coletar dados de participantes em suas vidas cotidianas.

Barricelli e Valtolina (2017) apresentaram uma linguagem visual e sua implementação, além de um sistema visual para o gerenciamento colaborativo de sensores da Internet das Coisas (IoT). O sistema, denominado SmartFit Rule Editor, foi projetado para ser usado por treinadores para monitorar e analisar a rotina de atletas. Seraj, Serge e Janssen (2018) descreveram o BEESM, um Aplicativo de programação de usuário final baseado em blocos que permitia a usuários inexperientes e programadores novatos desenvolver de forma rápida

ISSN: 2594-8083

aplicativos para dispositivos e ambientes inteligentes.

A edição colaborativa quase em tempo real é suportada, entre outros, quando os usuários empregam linguagem de marcação (MACSWEEN et al., 2018) e primitivos baseados em forma. Ferramentas especializadas suportam edição e interação com elementos com semântica e comportamentos pré-definidos, como ferramentas de geometria dinâmica (ISOTANI e BRANDÃO,2008) ou simulações. Também existem ferramentas que empregam linguagem de marcação em vez de primitivas gráficas (MACSWEEN et al., 2018).

O suporte para edição colaborativa síncrona exige tanto micro versão quanto macro versão, conforme destacado por Kuryazov et al. (2015). O primeiro identifica pequenas mudanças que, por sua vez, são trocadas entre colaboradores remotos e usadas na edição do histórico para suportar desfazer / refazer; no entanto, eles não são armazenados para fazer parte do histórico de versões. O último, por outro lado, registra explicitamente as mudanças para fornecer controle de versão.

Fraser (2009) apresentou o método de Sincronização Diferencial (DS) para manter documentos sincronizados - é um algoritmo simétrico que emprega um ciclo interminável de diferença de fundo (diff) e operação de patch (difusa). O autor argumenta que o DS pode lidar com qualquer conteúdo, incluindo texto simples, rich text, bitmaps, gráficos vetoriais: o requisito é a disponibilidade de um algoritmo de diferença e um algoritmo de patch difuso para o conteúdo específico. A escalabilidade e a capacidade de resposta dependem da eficiência e precisão desses algoritmos.

No contexto da edição colaborativa de modelos, Kuryazov et al. (2014, 2015, 2018), detalham sua arquitetura para suportar a modelagem colaborativa, que inclui componentes para macro versão (controle de versão) e micro versão. Este último é um componente sincronizador que recebe os *deltas* computados pelos clientes e os transmite aos colaboradores remotos.

3 CENÁRIO DE APLICAÇÃO DE UM PROGRAMA ESM NO DOMÍNIO DA SAÚDE

Os profissionais podem aproveitar a ampla utilização de smartphones pela população para realizar o atendimento remoto que complementa o atendimento presencial (VAN BERKEL, FERREIRA e KOSTAKOS, 2017). Isso pode ser útil para apoiar, por exemplo, ações voltadas para a solidão, uma vez que é crescente o número de indivíduos que sofrem de solidão (BEUTEL et al., 2017; CACIOPPO et al, 2015), principalmente entre os idosos (ONG, UCHINO e WETHINGTON, 2016) . Porém, é importante que o atendimento remoto seja personalizado para o paciente e, ao mesmo tempo, não resulte em sobrecarga para o

ISSN: 2594-8083

profissional. A plataforma ESPIM (ESPIM, 2022), visa apoiar os profissionais (ou investigadores) a utilizarem programas de intervenção baseados no método ESM, de forma assíncrona, visando permitir aos profissionais interagirem à distância com os seus pacientes (ou participantes da investigação), nos seus ambientes naturais.

A aplicação de programas de intervenção é ilustrada em um cenário hipotético em que um profissional de saúde monitora um paciente durante um período de tempo. O profissional solicita um conjunto de tarefas a serem realizadas em horários pré-definidos do dia. Os storyboards das Figuras 1 e 2 ilustram dois cenários alternativos programados pelo terapeuta, dependendo se o paciente está ou não sozinho: o conjunto de perguntas é diferente para o caso em que o paciente está sozinho ou não. No storyboard mostrado na Figura 1, o paciente está sozinho quando o programa de intervenção é acionado: o terapeuta pede uma mensagem de áudio e os sentimentos atuais do paciente; a seguir, o paciente é convidado a jogar um jogo (que é um programa externo); uma vez que o jogo termina, uma última questão, pergunta novamente como o paciente se sente.

No storyboard mostrado na Figura 2, o paciente não está sozinho quando o programa de intervenção é acionado: neste caso, o programa de intervenção pede ao participante que envie uma selfie e informe quem é a outra pessoa; a última pergunta pede ao participante que informe como ele está se sentindo.

3.1 WORKFLOW DE UM PROGRAMA DE INTERVENÇÕES DO ESPIM

Os especialistas em domínio programam intervenções na forma de fluxos não lineares: um fluxo específico é executado como resultado da interação do usuário com o programa de intervenção.

De agora em diante, usaremos de forma intercambiável os termos especialistas de domínio e observadores. Da mesma forma, usamos os termos usuários-alvo e participantes de forma intercambiável.

Uma intervenção é uma tarefa que o observador solicita ao participante para realizar. As intervenções são organizadas em um fluxo de trabalho de intervenção não linear (um grafo direcionado) em que:

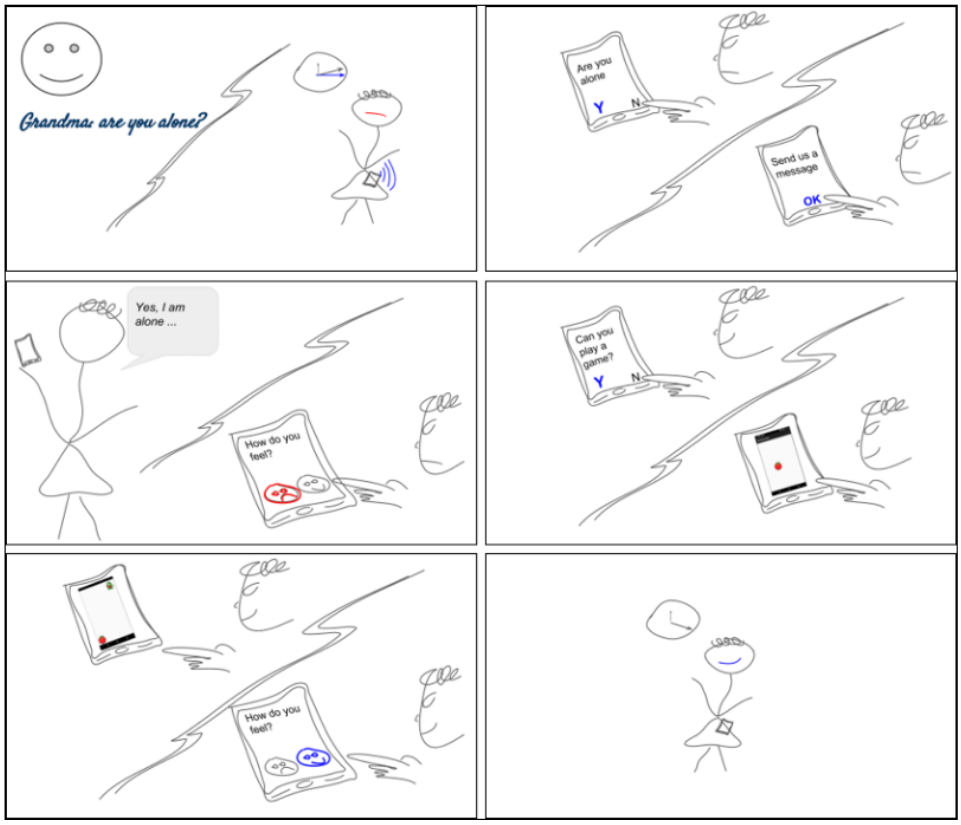
- uma única intervenção foi definida como a intervenção inicial;
- cada intervenção aponta para outra (próxima) intervenção ou é definida como uma intervenção final; e
- pelo menos uma intervenção é definida como uma intervenção final. As tarefas

ISSN: 2594-8083

são solicitadas aos usuários por meio de perguntas ou mensagens.

As tarefas de intervenção podem ser de diferentes tipos: questões abertas, questões de múltipla escolha, questões de múltipla opção, solicitação de mídia, ativação de uma atividade externa e mensagem. Embora, dependendo do tipo, possam ter atributos diferentes, todas as intervenções compartilham os seguintes atributos: id, texto (ou declaração para perguntas), próxima intervenção, mídias que serão apresentadas ao participante, se a intervenção é ou não a primeira intervenção e se a intervenção é ou não uma intervenção final.

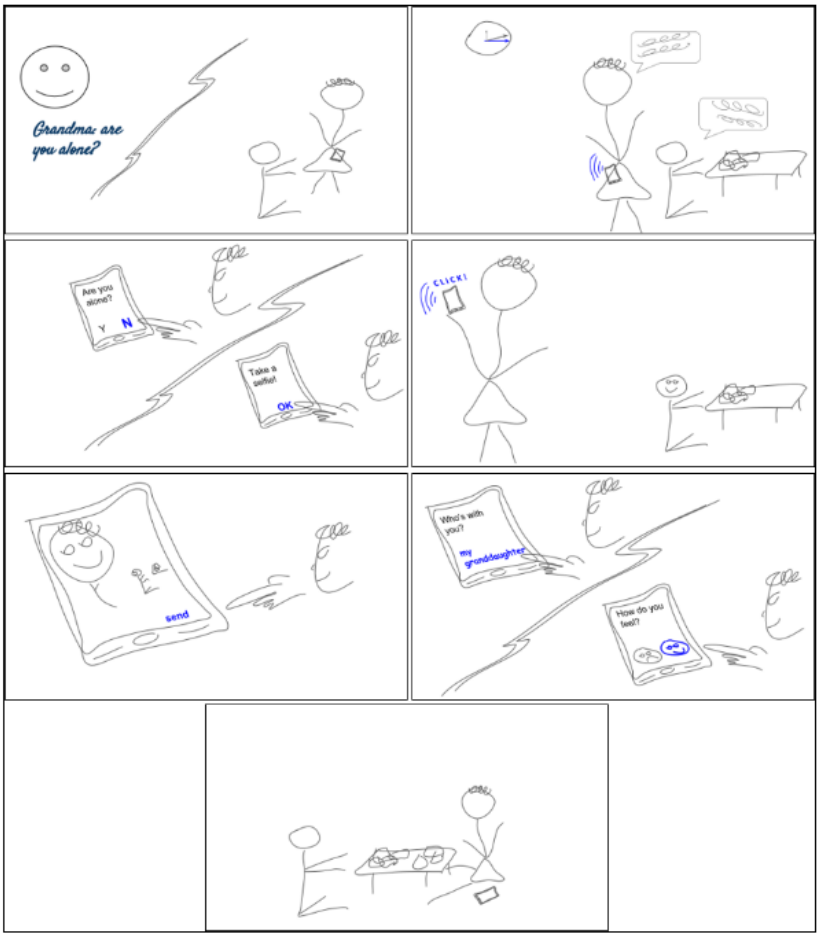
Figura 1: Storyboard (de cima para baixo e da esquerda para a direita): O participante está sozinho quando a intervenção é acionada; O participante responde a uma pergunta informando que seu status é "sozinha"; o programa de intervenção pede ao participante para enviar uma mensagem de áudio e informar seu estado de sentimento atual; em seguida, o Participante é convidado a jogar; O participante joga o jogo (com um sorriso); terminado o jogo, a última pergunta pede ao participante que informe seu estado de sentimento; o participante fica sozinho (e feliz) após a interação remota programada com o terapeuta.



Fonte: Material de divulgação ESPIM.

ISSN: 2594-8083

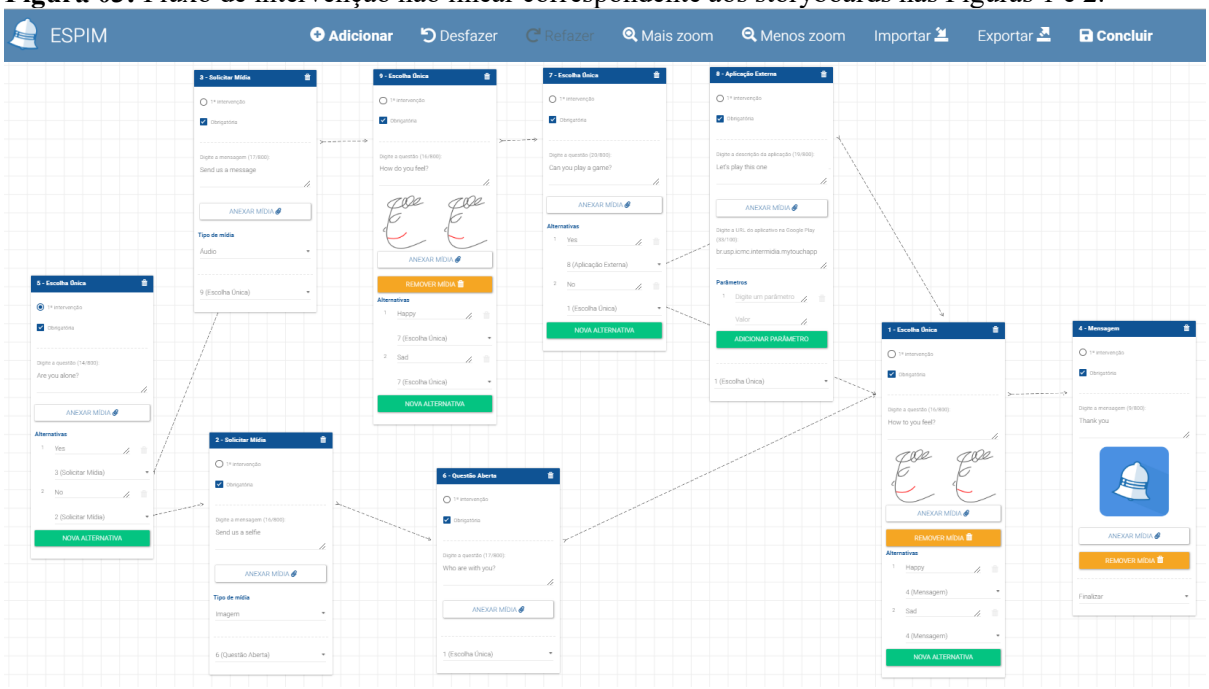
Figura 02: Storyboard (de cima para baixo e da esquerda para a direita): O participante não está sozinho quando a intervenção é acionada; O participante responde que seu status não é o único; neste caso, a próxima pergunta pede ao participante que envie uma foto de selfie e informe quem é sua empresa; a última pergunta pede ao participante que informe seu estado de sentimento atual. A Participante volta ao seu acompanhante após a interação remota programada com a terapeuta.



Fonte: Material de divulgação ESPIM.

Um tipo particular de pergunta - múltipla escolha com uma única resposta - permite que cada alternativa (escolha) leve a uma intervenção diferente: isso resulta na criação de fluxos não lineares com vários caminhos (ou seja, um grafo direcionado). Por exemplo, as tarefas implementadas nos programas de intervenção ilustrados nos dois storyboards mostrados nas Figuras 1 e 2 são definidas no fluxo de intervenção não linear ilustrado na Figura 3.

Figura 03: Fluxo de intervenção não linear correspondente aos storyboards nas Figuras 1 e 2.



Fonte: Material de divulgação ESPIM.

Os eventos podem ser programados para ocorrerem de forma regular, em determinado dia e horário, ou podem ser disparados quando determinada situação específica ocorrer, como por exemplo o participante ir a um determinado local, ou realizar determinada atividade que possa ser detectada por algum tipo de sensor. Como exemplo podemos citar se um participante for a academia, pela localização do smartphone isso pode ser detectado e pode gerar uma intervenção perguntando se ele gostou da atividade.

4 ATUALIZAÇÃO DO ESPIM PARA EDIÇÃO COLABORATIVA

Com base em sua arquitetura, o ESPIM pode ser definido com um sistema cliente-servidor de 3 camadas (OLUWATOSIN,2014), seguindo os padrões atuais para desenvolvimento de sistemas de internet, possui um frontend e um backend. Os sistemas baseados na arquitetura cliente-servidor dividem o processamento das aplicações em várias máquinas (KAMBALYAL,20103), permite um compartilhamento mais fácil de recursos e reduz a replicação de dados (OLUWATOSIN,2014). Por ser o ESPIM um sistema web baseado em protocolo HTTP e sistema de banco de dados relacional, as operações realizadas pelos usuários são enviadas ao servidor constantemente. O servidor armazena os dados e cria identificadores para cada objeto armazenado.

A arquitetura ESPIM WEB permite que vários autores editem a um mesmo programa,

ISSN: 2594-8083

porém adota um modelo pessimista, onde apenas um autor pode editar o programa por vez. A edição de um programa ESPIM é dividida em duas partes:

1. A manutenção dos eventos segue o fluxo de um cadastro simples, que segue o modelo padrão de sistemas cliente-servidor com banco de dados; e
2. O editor de intervenções, que possui uma interface gráfica e mostra os dados na forma de um gráfico direcionado.

O grande desafio de tornar o ESPIM colaborativo se deve ao fato de já existirem conceitos implícitos nas interfaces que são desenvolvidas, o ESPIM aplica o conceito de desenvolvimento de programas por usuários finais, em inglês End User Development (EUD), (LIEBERMAN,2006), onde toda arquitetura proposta tem o objetivo de permitir que especialistas com pouco, ou nenhum conhecimento, em programação para computadores, possam criar os programas de monitoramento que serão interpretados e processados por um aplicativo acessado pelos usuários.

No caso específico do ESPIM, mas comum a todos editores colaborativos, o grande desafio é fazer com as alterações realizadas por diversos usuários trabalhando que editam um mesmo documento ao mesmo tempo, tenham versões atualizadas dos documentos e que as atualizações sejam distribuídas em *broadcast* para todos o usuários em um tempo razoável. Além disso é importante manter um estrutura que aplique aos documentos as alterações na ordem em que elas foram realizadas, e quando isso não for possível que exista um mecanismo de *merge* adequado.

No presente trabalho, para apoiar a edição colaborativa e manter a sincronização entre diversos documentos sendo editados por diverso usuários ao mesmo tempo, foram propostos os conceitos de delta, macro-versão e micro-versão de Kuryazov et al. (2014, 2018, 2019), além disso foram desenvolvidos mecanismo para:

- Manter um histórico das alterações realizadas pelos usuários;
- Sincronizar os documentos dos usuários;
- Gerenciar as alterações realizadas;
- Controlar a concorrência e o *merge* das alterações realizadas.

4.1 HISTÓRICO DE MUDANÇAS

A arquitetura do ESPIM, não guarda nenhum registro das operações que foram realizadas pelos autores. Embora os programas ESPIM tendam a ser pequenos e a oferta atual

ISSN: 2594-8083

de desfazer / refazer e salvar automaticamente tenha sido apreciada pelos usuários, o suporte para edição síncrona quase em tempo real e acesso ao histórico da edição é essencial.

Da mesma forma que Kuryazov et al. (2014, 2015, 2018), que também apóia a edição de modelos gráficos com semântica associada, estamos trabalhando para oferecer suporte a uma abordagem baseada em edição para a sincronização das mudanças feitas por usuários remotos. Para oferecer suporte a micro-versões e macro-versões, primeiro é necessário implementar no sistema funções que permitam que todas as operações de edição possam ser desfeitas, para que possamos empregar a codificação delta com base nas operações de adição, exclusão e atualização.

Por ser o ESPIM um sistema web baseado em protocolo HTTP e sistema de banco de dados relacional, as operações realizadas pelos usuários são enviadas ao servidor constantemente. O servidor armazena os dados e cria identificadores para cada objeto armazenado. Os sistemas baseados em delta precisam manter o histórico de mudanças central, denominado macro-versão, este registro de mudanças central permite que todos os usuários fiquem em sincronia.

Se a diferença for armazenada em um banco de dados, o banco de dados se torna muito complexo e enorme e os deltas de modelagem podem não ser identificados e reutilizados. Para evitar o problema que pode ocorrer com o histórico de alterações, por ser armazenado em um banco de dados, foi levantado um histórico onde são registradas todas as operações em ordem de execução, cada registro possui data, hora, dados do objeto, editor e um número sequencial único .

O histórico representa de forma ordenada todas as operações que foram realizadas, se alguém executar todas as operações relacionadas a um programa desde as primeiras operações que estão na história chegará à versão atual do programa. Na direção oposta, se alguém aplicar os comandos de desfazer da última operação na ordem inversa do que foi executado, o programa será completamente excluído.

4.2 SINCRONIZAÇÃO DE USUÁRIOS

Para manter os usuários em sincronia, juntamente com o histórico de alterações, é necessário implementar um canal que notifique os usuários quando ocorrerem alterações, o ESPIM utiliza o protocolo HTTP, porém o HTTP não foi projetado para fornecer comunicação full-duplex em tempo real (PIMENTEL e NICKERSON,2012), ele só pode simular real-comunicação de tempo, mas com uma latência aumentada e alto tráfego de rede

ISSN: 2594-8083

(KAMBALYAL,20103). O HTTP é baseado em um modelo de envio / recebimento , onde as partes não mantêm uma conexão ativa entre si.

Visando suprimir as deficiências apresentadas pelo HTTP no sentido de criar conexões, no projeto foi utilizado websocket a fim de fornecer um canal de comunicação que permitisse a sincronização entre diferentes usuários. O protocolo WebSocket fornece um canal de comunicação bidirecional full-duplex que opera através de um único soquete na Web (LUBBERS et al., 2010). Desta forma, utilizando o WebSocket, foi criado um canal para notificar todos os usuários, conectados em um determinado programa, quando ocorre alterações no programa no qual o usuário está conectado. O frontend verifica através do histórico o que foi modificado, se necessário, o frontend solicita as alterações no backend e aplica as modificações nos dados que estão sendo exibidos.

4.3 GERENCIAMENTO DE ALTERAÇÕES

Para que um editor colaborativo pareça tão responsivo quanto um editor de usuário único, conforme preconizado por Ellis e Gibbs (1989) nas primórdios dos trabalhos sobre edição colaborativa, é necessário a utilização de um algoritmo de controle de simultaneidade, para manter a consistência entre os diferentes usuários que editam um documento simultaneamente (GADEA,2021).

Visando atender a essa necessidade, foi adotada uma solução baseada nos algoritmos de controle da competição Jupiter (NICHOLS et al.,1995) e SOCT4 (VIDOT et al., 2000) adaptada para aproveitar as características de um banco de dados relacional . Cada operação realizada recebe um número sequencial, um *ticket*, que identifica aquela operação. Com base nos números dos *tickets*, o servidor é capaz de identificar as versões dos documentos e dos elementos que o compõem.

Quando um usuário vai editar um programa de intervenção, no momento em que solicita a edição do programa recebe a versão mais atual que se encontra no servidor, junto com o ticket que identifica essa versão. Quando o usuário faz uma alteração, a alteração é enviada ao servidor junto com o ticket, o servidor baseado no ticket é capaz de realizar a operação mantendo a consistência do documento que está sendo editado. Após fazer a alteração, o servidor notifica todos os usuários que estão editando aquele documento, transmitindo a operação por meio do websocket. Deve-se notar que toda operação enviada em broadcast pelo servidor leva consigo o número do *ticket*.

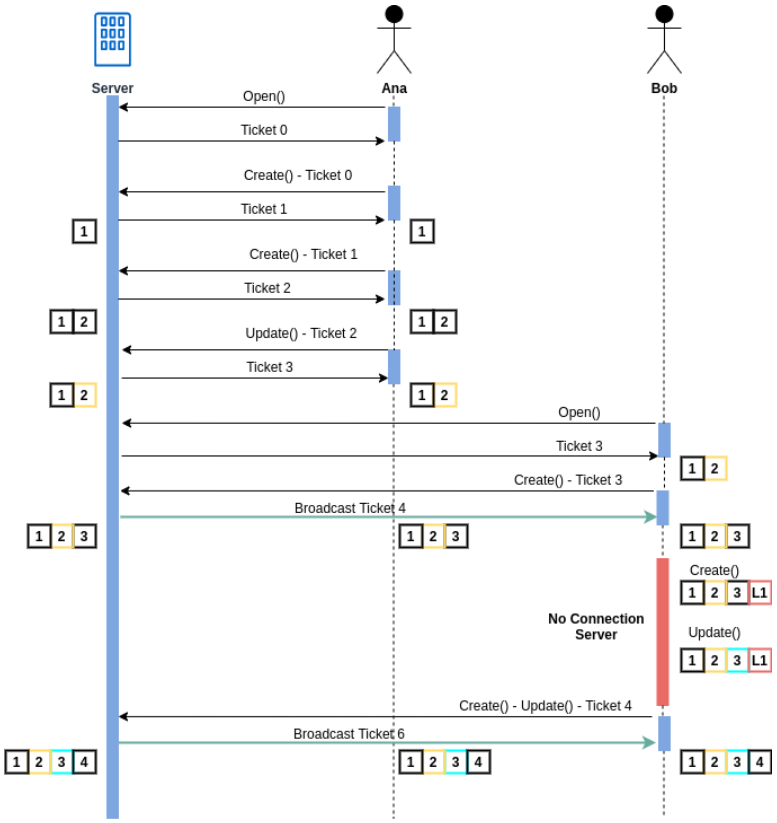
A figura 4 ilustra o processo de criação das operações e geração do *ticket*, quando Ana

abre o editor de intervenção ele está vazio e ela recebe ticket 0 (zero) em resposta. Ana cria a intervenção 1 e recebe como resposta o *ticket* 1, quem define o id para todas as intervenções é também o servidor. Ana cria a intervenção 2, recebe o *ticket* 2 do servidor, finalmente Ana altera a intervenção 2 e recebe o *ticket* 3 em resposta.

Quando Bob abre o editor de intervenção, ele recebe uma cópia do modelo de intervenção e o *ticket* 3. Bob cria uma intervenção, o servidor envia por broadcast a todos os usuários (Ana e Bob) as informações da nova intervenção criada e o valor do novo *ticket*. Todos os usuários são atualizados e sincronizados.

Em algum ponto, Bob perde sua conexão com o servidor, ele continua a trabalhar no modelo e cria uma nova intervenção e faz uma atualização. A intervenção criada receberá apenas uma identificação local (L1). Quando Bob retoma a conexão com o servidor, as alterações são enviadas ao servidor, a intervenção criada recebe um identificador do servidor e as operações realizadas são transmitidas por todos os usuários. Assim, todos os usuários e o servidor terão o mesmo modelo e tíquete.

Figura 04: Processo de gerenciamento de alterações.



Fonte: Autor

4.4 CONTROLE DE CONCORRÊNCIA E MERGE

ISSN: 2594-8083

No exemplo da figura 4 enquanto Bob estava desconectado do servidor, Ana não executou nenhuma operação, no entanto, se Ana tivesse feito alguma alteração no modelo, o servidor deveria agir de forma que não houvesse perda de dados ou sobreposição ocorre. Por meio de regras é possível implementar o controle de simultaneidade, tais regras definirão o que o servidor deve fazer ao receber uma lista de operações de um usuário que realizou operações sem estar conectado ao servidor. Em nossa implementação, quando o servidor receber uma lista de alterações e perceber que o tíquete está desatualizado, ele obedecerá às seguintes regras:

1. Se a intervenção foi atualizada após o ticket, ela não pode ser excluída. A operação de exclusão será descartada e um ticket não será gerado;
2. As atualizações recebidas serão realizadas e um ticket será gerado;
3. Se a intervenção foi deletada e outra operação vem para deletá-la, a operação será descartada e não será gerado um ticket;
4. Se for recebida uma atualização para uma intervenção que foi deletada, ela será recriada com o conteúdo que tinha na deleção e a atualização será aplicada, serão gerados dois tickets, um para a criação e outro para a atualização;
5. Todas as criações serão realizadas no final do arquivo de intervenção.

No exemplo mostrado na figura 5, Ana abre o editor de intervenção e cria uma nova intervenção. Quando Bob abre o editor de intervenção, ele recebe o mesmo modelo e ticket que Ana. Bob cria uma nova intervenção que é atualizada no servidor e distribuída pelo servidor. Bob perde a conexão com o servidor e executa 3 operações: uma atualização, uma criação e uma exclusão. Como ele não está conectado ao servidor, suas operações são realizadas apenas localmente.

Enquanto Bob não está conectado ao Servidor Ana, ele também executa três operações, uma exclusão, uma criação e uma atualização. Por estar em conexão com o servidor, as operações de Ana são sincronizadas com o Servidor, portanto, o servidor e Ana têm o mesmo modelo e valor de ticket.

Quando Bob consegue se reconectar ao servidor, ele envia todas as operações que realizou. Ele verifica o valor do tíquete e é capaz de determinar como o modelo estava quando Bob realizou suas operações e o que aconteceu depois que Bob perdeu sua conexão. O servidor recebe as operações de Bob e as executa da seguinte maneira:

1. Atualização (1) - A intervenção foi eliminada por Ana no Bilhete 3, seguindo as regras do controlo da competição, o servidor irá recriar a intervenção 1 com os dados que ela tinha no Bilhete 2, a regra diz que todas as intervenções são criadas

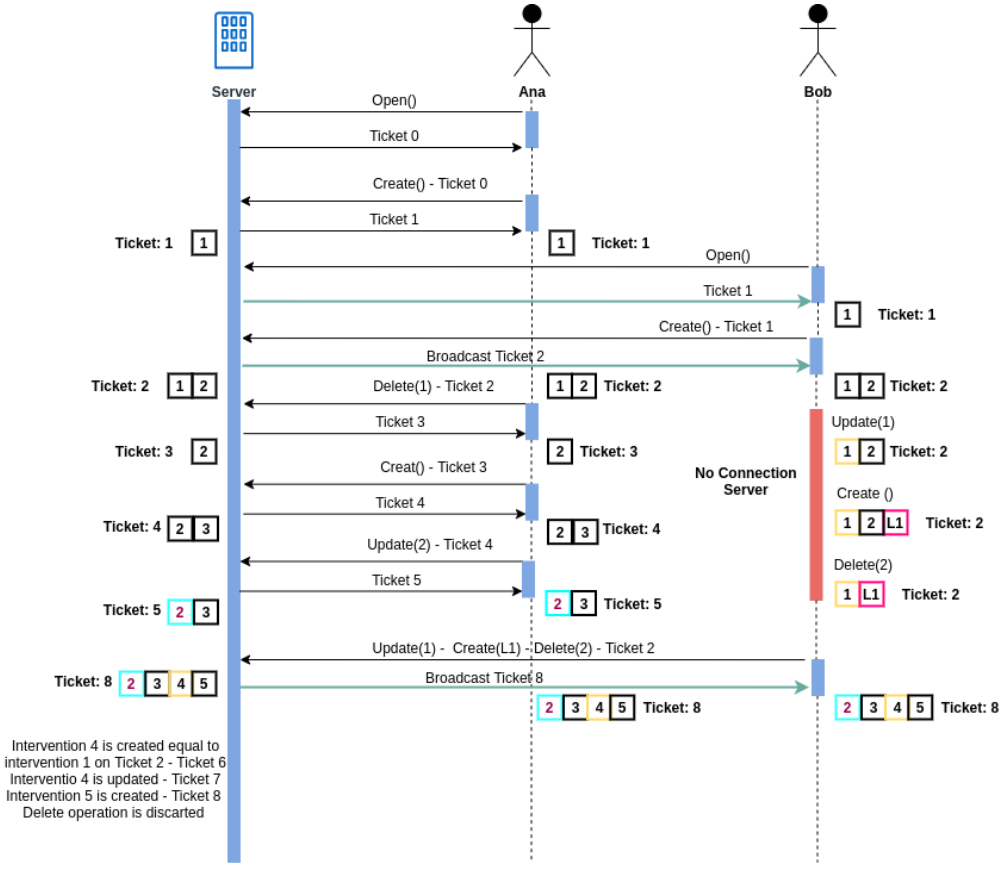
ISSN: 2594-8083

no final da fila, após recriar a intervenção, o servidor fará a atualização, conforme solicitado por Bob. Dois tickets serão gerados, um para a criação e outro para a atualização;

2. Criar (L1) - A intervenção L1 foi criada localmente por Bob, então ela precisa ser enviada ao servidor, a regra de criação de intervenção diz que todas as intervenções devem ser criadas no final da fila. Desta forma, é criada a intervenção 5 e gerado um novo ticket;
3. Eliminar (2) - A intervenção 2 foi alterada por Ana no bilhete 5, a regra do controle da competição diz que quando uma intervenção foi alterada não pode ser eliminada por operações com um bilhete anterior. Portanto, a operação de exclusão de intervenção 2 será descartada sem gerar um tíquete.

Depois que todas as alterações enviadas por Bob forem realizadas, o servidor irá transmitir todas as operações realizadas. E todos os usuários serão sincronizados novamente.

Figura 05: Controle de Concorrência



Fonte: Autor

Embora o servidor tenha controle de simultaneidade que permite aos usuários editar

ISSN: 2594-8083

intervenções sem estarem conectados ao servidor, esta é uma situação que deve ser entendida como exceção. O controle de simultaneidade visa evitar a perda de dados, entretanto, o resultado pode não ser o esperado pelos usuários que estão editando as intervenções. O Editor apresentará resultados mais satisfatórios quando todos os usuários permanecerem conectados durante todo o processo de edição colaborativa.

5 TESTES

Para testar a funcionalidade do modelo colaborativo implementado, o backend foi disponibilizado em um servidor, e seis instâncias do frontend foram executadas em três computadores simultaneamente, em cada computador dois navegadores foram abertos com o frontend. Os objetos foram criados, atualizados e excluídos em todas as instâncias abertas. Todas as operações realizadas em uma instância foram replicadas corretamente em todas as outras.

Para testar a validade do histórico, todas as operações foram realizadas em ordem, como uma fila, no final os registros existentes eram exatamente os mesmos que os criados no primeiro teste. Por fim, todas as operações foram desfeitas em ordem reversa, como uma pilha, como resultado, todos os registros criados para o teste foram excluídos.

Com a utilização apenas das três operações delta, a construção de um histórico de operações e o uso da comunicação via websocket, implementamos recursos de colaboração na ferramenta de autoria ESPIM. Nos testes foi possível: (a) coletar alterações (adicionar, excluir, atualizar) feitas pelos usuários, (b) divulgar essas alterações entre os usuários remotos, (c) aplicar as alterações a cada editor cliente. Como resultado de nossos esforços para fornecer à ferramenta de autoria do ESPIM recursos de trabalho colaborativo, desenvolvemos uma solução simples que usa apenas operações delta, um registro de histórico de operações e um websocket. Nosso modelo, assim como foi feito no ESPIM, pode ser implementado sem grandes esforços em sistemas cliente-servidor que precisam oferecer recursos de colaboração.

6 CONSIDERAÇÕES FINAIS

O método ESPIM atualmente é utilizado por especialistas em áreas que incluem a interação humano-computador (IHC), pesquisadores de computação ubíqua, educadores, psicólogos, terapeutas ocupacionais e fonoaudiólogos. Um dos seus principais feedbacks positivos é que os programas de intervenção permitem não apenas o contato com seus usuários,

mas também que a interação pode ser planejada e ocorrer de forma sistemática que auxilie na coleta e análise de dados controlados.

No presente trabalho desenvolvemos e descrevemos em forma de roteiro uma sequência de técnicas e recursos de TI que possibilitaram à plataforma ESPIM ofertar a edição colaborativa e por consequência o suporte ao trabalho colaborativo por parte dos especialistas.

A grande relevância do trabalho reside no fato de que ele descreve uma forma simples de prover sistemas computacionais da capacidade de ofertar o recurso de edição colaborativa e suporte ao trabalho colaborativo.

Na sequência deste trabalho pretende-se estender a plataforma ESPIM de forma que a autoria de workflows de intervenção não linear empregue codificação delta baseada em operações de adição, exclusão e atualização modeladas neste artigo. Também projetamos um sincronizador que oferecerá suporte à colaboração quase em tempo real com base na troca de fluxos das operações de adição, exclusão e atualização realizadas pelos usuários. A plataforma resultante será então estendida para oferecer suporte aos usuários para ter acesso ao histórico de autoria do documento, identificar versões por tempo e autor e reverter para versões anteriores.

7 REFERÊNCIAS BIBLIOGRÁFICAS

BARRICELLI, Barbara Rita e VALTOLINA, Stefano. 2017. A visual language and interactive system for end-user development of internet of things ecosystems. **Journal of Visual Languages & Computing 40** (2017), 1–19.

BERNERS-LEE, T., FIELDING, R., and FRYSTYK, H. (1996). Rfc1945: **Hypertext transfer protocol –** http/1.0.

BEUTEL, M. E., et al. (2017). Loneliness in the general population: prevalence, determinants and relations to mental health. **BMC psychiatry**, 17(1):97.

CACIOPPO, S., et al. (2015). Loneliness: Clinical import and interventions. P**erspectives on Psychological Science,** 10(2):238–249.

ELLIS, C. A. e GIBBS, S. J. (1989). **Concurrency control in groupware systems**. In Proceedings of the **1989 ACM SIGMOD international conference on Management of data**, pages 399–407.

ESPIM, Ferramenta para Intervenção Programada, Disponível em: www.espim.com.br, Acessado em: 30/08/2022.

FRASER, N. (2009). **Differential synchronization.** In Proceedings of the **9th ACM Symposium on Document Engineering**, DocEng ’09, page 13–20, New York, NY, USA. Association for Computing Machinery.

GADEA, C. (2021). **Architectures and Algorithms for Real-Time Web-Based Collaboration**. PhD thesis, Universit´e d'Ottawa/University of Ottawa.

ISOTANI, S. e BRANDÃO, L. d. O. (2008). An algorithm for automatic checking of exercises in a dynamic geometry system: Igeom. **Comput. Educ.**, 51(3):1283–1303.

KAMBALYAL, C. (2010). **3-tier architecture**. Retrieved On, 2:34.

KURYAZOV, D. e WINTER, A. (2014). **Representing model differences by delta operations**. In Proceedings of the 2014 **IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, EDOCW '14,** page 211–220, USA. IEEE Computer Society.

KURYAZOV, D. e WINTER, A.(2015). **Collaborative modeling empowered by modeling deltas**. In Proceedings of the 3**rd International Workshop on (Document) Changes: Modeling, Detection, Storage and Visualization, DChanges 2015**, page 1–6, New York, NY, USA. Association for Computing Machinery.

KURYAZOV, D. e WINTER, A. e REUSSNER, R. (2018). **Collaborative modeling enabled by version control.** Modellierung 2018.

LARSON, R e CSIKSZENTMIHALYI, M. (2014). **The experience sampling method. In Flow and the foundations of positive psychology**, pages 21–34. Springer.

Lieberman, H., et al. (2006). **End-user development: An emerging paradigm. In End user development**, pages 1–8. Springer.

LUBBERS, P., et al. (2010). Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development. Apress, USA, 1st edition.

MACSWEEN, A.. et al.(2018). Private document editing with some trust. In Proceedings of the ACM Symposium on Document Engineering 2018, DocEng '18, New York, NY, USA. Association for Computing Machinery.

NICHOLS, D. A., et al. (1995). High-latency, lowbandwidth windowing in the jupiter collaboration system. In Proceedings of the 8th annual ACM symposium on User interface and software technology, pages 111–120.

OLUWATOSIN, H. S. (2014). Client-server model. IOSRJ Comput. Eng, 16(1):2278–8727.

ONG, A. D., UCHINO, B. N., AND WETHINGTON, E. (2016). Loneliness and health in older adults: A mini-review and synthesis. Gerontology, 62(4):443–449.

PIMENTEL, V. AND NICKERSON, B. G. (2012). Communicating and displaying real-time data with websocket. IEEE Internet Computing, 16(4):45–53.

RAEMAEKERS, S., VAN DEURSEN, A., AND VISSER, J. (2017). Semantic versioning and impact of breaking changes in the maven repository. J. System Software, 129(C):140–158.

RODRIGUES, Kamila RH, et al. "Espim system: interface evolution to enable authoring and interaction with multimedia intervention programs." *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*. 2018.

ISSN: 2594-8083

Daniel J Rough and Aaron Quigley. 2020. End-User Development of Experience Sampling Smartphone Apps-Recommendations and Requirements. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 4, 2 (2020), 1–19.

MAZYAR Seraj, SERGE, Autexier e JANSSEN, Jan. 2018. BEESM, a block-based educational programming tool for end users. In Proceedings of the 10th Nordic Conference on Human-Computer Interaction. 886–891

VAN BERKEL, N., FERREIRA, D., AND KOSTAKOS, V. (2017). The experience sampling method on mobile devices. ACM Computing Surveys (CSUR), 50(6):1–40.

VIDOT, N., et al. (2000). Copies convergence in a distributed real-time collaborative environment. In Proceedings of the 2000 ACM conference on Computer supported cooperative work, pages 171–180.

ZAINE, Isabela, et al. "ESPIM: An ubiquitous data collection and programmed intervention system using esm and mobile devices." *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. 2016.