RecorDS: Um Aplicativo para Extração de Diagramas UML

Douglas M. Giordano[1], **Arthur M. Becker**[1], **João P. S. da Silva**[1]

[1]Engenharia de Software – Universidade Federal do Pampa (UNIPAMPA)
97.546-550 – Alegrete – RS – Brazil

douglasmontanhagiordano@gmail.com

arthmalbeck@gmail.com, joaosilva@unipampa.edu.br

Abstract. *Teams that use agile modeling practices often use whiteboards, as they offer a dynamic and collaborative modeling environment. The problem occurs in the generation of software artifacts, due to the fact that normally the sketches made on whiteboards have to be recreated in a UML editor or else they are stored in images, causing the documentation to be out of date. Thus, in order to minimize this integration between sketches made in frames and UML editors, this article has the bias of presenting an application for recognizing sketches of class diagrams. For the purposes of developing this application, image processing and analysis techniques are used.*

Resumo. *Equipes que utilizam práticas de modelagem ágil muitas vezes fazem uso de quadros brancos, por oferecerem um ambiente dinâmico e colaborativo de modelagem. O problema ocorre na geração dos artefatos de software, devido ao fato de que normalmente os esboços feitos em quadros brancos são recriados em um editor UML ou então armazenados em imagens, ocasionando uma desatualização da documentação. Assim, com o objetivo de minimizar essa integração entre esboços feitos em quadros e editores UML, esse artigo tem o viés de apresentar um aplicativo de reconhecimento de esboços de diagramas de classe. Para fins de desenvolvimento desse aplicativo, técnicas de processamento e análise de imagens são utilizados.*

1. Introdução

Os métodos ágeis são adotados por muitas empresas de software durante o processo de desenvolvimento, sendo esses normalmente interativos, evolutivos, adaptativos e incrementais. Os métodos ágeis, segundo alguns autores, são flexíveis em relação aos métodos tradicionais. A modelagem ágil é uma prática ágil comumente usada por empresas para modelar o software, auxiliando na comunicação da equipe e geralmente utilizando a UML (*Unified Modeling Language*).

Dessa forma, equipes de desenvolvimento que seguem tais práticas ágeis, frequentemente recorrem a quadros brancos para a modelagem de diagramas. Devido ao fato de os quadros brancos não imporem regras [Costagliola et al. 2014] na hora de desenhar o modelo.

Embora a modelagem em quadros brancos seja simples e fácil, ela apresenta alguns problemas em relação à integração dos artefatos de software e a sua documentação em imagens [Wüest et al. 2013]. Tais problemas podem ser minimizados por meio de uma ferramenta que viabilize a integração do esboço com um editor UML.

Nesse sentido, esse presente trabalho tem como objetivo apresentar uma ferramenta para a extração de esboços de diagramas de classe desenhados em quadros brancos. A ferramenta, vai possibilitar a integração entre esboços feitos em quadros brancos e editores UML, viabilizar o reconhecimento de diagramas de classe em uma perspectiva conceitual, possibilitar a utilização de dispositivos móveis para capturar e reconhecer a imagem e gerar arquivos no formato XMI (*XML Metadata Interchange*) dos diagramas reconhecidos.

2. Fundamentação Teórica

Nessa seção será abordado os conceitos relacionados com os sistemas de reconhecimento de esboços e modelagem de software. Na seção 2.1 é apresentada a UML, uma linguagem de modelagem de software. Na seção 2.2 é descrito os principais conceitos relacionados ao processamento de imagens, responsável pela melhoria ou retirada de características indesejadas em imagens. Por fim, na seção 2.3 apresentamos a análise de imagens, que visa a extração e compreensão do conteúdo de uma imagem.

2.1. Linguagem Unificada de Modelagem

A UML (Linguagem Unificada de Modelagem) é uma linguagem de modelagem de software padrão que pode ser empregada na visualização, especificação, construção e documentação de artefatos de software [Booch et al. 2000].

A UML possui três tipos de blocos de construção denominados de itens, relacionamentos e diagramas. Os itens podem ser divididos em estruturais, comportamentais, de agrupamento e notacionais. Os diagramas da UML 2.5 podem ser divididos em duas categorias, os diagramas estruturais que tem como objetivo mostrar o sistema de uma perspectiva estática, e os diagramas comportamentais que tem como objetivo mostrar o sistema em uma perspectiva mais dinâmica [OMG 2015].

2.2. Processamento de Imagens Digitais

O processamento de imagens consiste em um conjunto de técnicas para capturar, representar e transformar imagens com auxílio de computador [Pedrini and Schwartz 2008]. Podemos definir uma imagem digital como uma função bidimensional onde em cada par de coordenadas (x,y) possui um valor representando a densidade, também chamado nível de cinza. Esse par de coordenada é o Pixel.

O sistema de registro é um dispositivo utilizado para impressão ou apresentação da imagem. O hardware especializado em processamento de imagens normalmente consiste em um digitalizador (conversor de estímulos elétricos em dados digitais) e um hardware para realizar operações que requerem um rápido processamento. O Software de processamento de imagens é um conjunto de módulos especializados para a realização de tarefas específicas. Os sensores são os dispositivos responsáveis pela aquisição da imagem. Por último temos a rede de comunicação que faz a transferência de dados entre os componentes.

Segundo [Gonzalez and Woods 2010] alguns passos são fundamentais no processamento digital de imagens. Podemos considerar como métodos de processamento de imagem a aquisição da imagem, realce de imagens, restauração de imagens, processamento de imagens coloridas e a compreensão de imagens. Os métodos de processa-

mento morfológico, segmentação, representação, descrição e reconhecimento de objetos são métodos de análise de imagens.

2.3. Análise de Imagens Digitais

A análise de imagens é, tradicionalmente, baseada na forma, textura, níveis de cinza ou nas cores dos objetos presentes em uma imagem. A análise de imagens tem como foco compreender o conteúdo da imagem. Algumas técnicas comuns na análise da imagem são a segmentação da imagem em regiões ou objetos de interesse, descrição dos objetos e reconhecimento ou classificação dos mesmos [Pedrini and Schwartz 2008].

A segmentação e suas técnicas buscam detectar descontinuidades e similaridades da imagem, assim sendo um processo importante para identificação de objetos. A imagem é subdividida em regiões ou objetos de interesse. A segmentação é uma das tarefas mais difíceis em um sistema de análise ou processamento de imagens onde a precisão da segmentação determina o sucesso, ou o fracasso final de um procedimento [Gonzalez and Woods 2010]. A divisão das regiões e objetos também depende do processamento da imagem onde, por exemplo, imagens com muito ruído podem distorcer a extração dos dados da imagem [Pedrini and Schwartz 2008].

As regiões e objetos gerados da segmentação necessitam ser representados e descritos para os próximos passos da análise da imagem. O objeto pode ser representado em termos de suas características externas (bordas) e características internas (pixels). A descrição depende da representação escolhida e é necessário a extração de características e medidas mínimas não permitindo ambiguidades.

A morfologia matemática é uma ferramenta para extrair componentes das imagens que são úteis na representação e na descrição da forma de uma região [Gonzalez and Woods 2010].

A classificação de padrões ou também chamada de reconhecimento de objetos visa mapear amostras extraídas com um conjunto de rótulos possuindo algumas restrições para mapear amostras com características semelhantes [Pedrini and Schwartz 2008].

3. Trabalhos Relacionados

Nessa seção será apresentada as questões e seus resultados obtidos por meio do mapeamento sistemático para investigar o estado da arte dos trabalhos relacionados aos sistemas de reconhecimento de diagramas UML ou estruturas semelhantes.

3.1. Quais técnicas de aquisição de imagem foram utilizadas?

No domínio de reconhecimento de fluxograma [Awal et al. 2011] utilizam telas sensíveis ao toque para aquisição de imagem. Outra ferramenta com proposta parecida é a FlexiSketch criada por [Wüest et al. 2013], tem como objetivo simular um quadro branco e receber esboços de diagramas UML desenhados diretamente na ferramenta.

[Buchmann 2012] propõe uma simulação de um quadro branco, oferecendo liberdade ao desenvolvedor. A aquisição da imagem é feito pelo desenho do desenvolvedor na tela do software. Uma ferramenta de modelagem denominada GMF (*Graphical Modeling Framework*) possibilitou ao usuário desenhar os esboços na própria tela do ambiente de desenvolvimento. Esse tipo de aquisição de imagens facilita a modelagem de diagramas em telas sensíveis ao toque, sem perder a liberdade de um quadro branco.

3.2. Quais filtros foram utilizados?

Todos os trabalhos relacionados encontrados recebiam como entrada um desenho feito pelo usuário diretamente no sistema. Logo, é eliminado grande parte do ruído ou qualquer outro tipo de distorção na imagem gerada muitas vezes pelo ambiente. Levando em consideração esse fator, poucos filtros foram utilizados nos artigos encontrados.

3.3. Quais técnicas de segmentação de imagem foram utilizadas?

A segmentação também foi pouco utilizada nos artigos encontrados, pois a maior parte dos desenhos eram analisados de forma isolada logo após o usuário desenhar. Contudo, foram encontradas algumas técnicas para separação do diagrama e do texto.

[Awal et al. 2011] tem como um dos principais problemas a separação do texto e dos gráficos, pois cada um deles é tratado de forma diferente, então para separá-los é apresentado uma técnica para medir a complexidade chamada de entropia. Por meio da entropia o grau de desordem é medido utilizando ângulo de pontos. Isso possibilita verificar a complexidade dos elementos, como o texto possui linhas mais complexas é possível separar a partir desta técnica. Outro método também utilizado para separar o diagrama do texto é a binarização linear absoluta que [Maggiori et al. 2014] aplicou com outras técnicas nos pixeis.

[Jiang et al. 2011] utilizaram algoritmo de partição de gráficos para separar em vários subgrafos cada mapa conceitual. A partir destes subgrafos é utilizada programação dinâmica para extrair os blocos de conceitos.

3.4. Quais técnicas de reconhecimento de padrões foram utilizadas?

Para o reconhecimento de esboço em um domínio geral [Liao and Duan 2012] utilizam uma sequência de caracteres que representam cada primitiva desenhada pelo usuário. São recebidos duas sequências de caracteres e é calculado a distância entre as mesmas utilizando métodos Kernel.

A falta de naturalidade das ferramentas CASE (*Computer-Aided Software Engineering*) e o retrabalho de transferir tais documentos para o computador levou [Costagliola et al. 2014] a apresentar uma ferramenta de reconhecimento de diagramas. A Metodologia foi dividida 4 em camadas: camada de separação de texto e gráficos, camada do reconhecimento de símbolos, camada de detecção do contexto local e por fim a camada de detecção do diagrama. O reconhecimento é feito em contexto local utilizando aprendizado de máquina.

Também em um domínio de reconhecimento de diagramas [Maggiori et al. 2014] utilizaram OpenCV para reconhecimento de diagramas arquiteturais. Para a detecção das relações foi utilizado Transformada de Hough e uma série de outras operações em cima dos pixeis da imagem. Também foi utilizado algoritmos de aprendizado de máquina para verificar a forma no final da linha que representa a relação. Os textos foram reconhecidos utilizando um OCR de código aberto chamado de Tesseract.

4. RecorDS: Recognizer Diagram Sketch

Nesta seção apresentamos o aplicativo denominado RecorDS (Recognizer Diagram Sketch). Na subseção 4.1 abordamos uma visão geral do trabalho, retomando os pro-

blemas e soluções ao qual motivarão a criação da ferramenta, além dos requisitos encontrados. Já na subseção 4.2 é tratado os detalhes da implementação do sistema.

4.1. Funcionalidades Gerais

A maioria dos editores UML não possibilitam a edição de diagramas UML por vários membros da equipe ao mesmo tempo, fazendo com que equipes prefiram quadros brancos [Wüest et al. 2013]. Assim, alguns problemas são encontrados na modelagem de software colaborativa utilizando um quadro branco ou mesmo papel e caneta. O primeiro é o retrabalho ocasionado pela reconstrução do diagrama manualmente em uma ferramenta CASE. Outro problema é que como muitas vezes a documentação é uma imagem não tem como gerar o código automaticamente.

Analisando os problemas descritos, é possível propor um produto que consiga minimizar o retrabalho de reconstruir todo diagrama. Uma ferramenta que capture a imagem com um aplicativo e possa extrair diagramas UML a partir da imagem, logo após gerar um arquivo para ser importado em um editor UML, resolveria o retrabalho de integrar o esboço de um diagrama com uma ferramenta CASE, consequentemente minimizando o trabalho das equipes.

A partir dos problemas e soluções encontramos os requisitos necessários ao sistema. Organizamos os requisitos por intermédio de uma lista apresentada abaixo.

- Detecção e reconhecimento de classes em um diagrama de classes.
- Detecção e reconhecimento de associações em um diagrama de classes.
- Detecção e reconhecimento de multiplicidades em um diagrama de classes.
- Reconhecimento de caracteres do diagrama de classes
- Geração do diagrama de classes no formato XMI seguindo a especificação UML.
- Utilização de um dispositivo Android para capturar e reconhecer o diagrama de classes.

4.2. Detalhes da Implementação

O processo de reconhecimento dos elemento para por um conjunto de passos. Utilizando um diagrama de atividades na Figura 1 mostramos cada passo necessário para reconhecer o diagrama. Adquirir a imagem por meio da câmera de um dispositivo Android é o primeiro passo. Após a imagem capturada pelo dispositivo, é realizada a detecção e reconhecimento dos elementos, na qual abrangem as seguintes etapas: extrair elementos, classificar elementos, segmentar nome da classe, interpretar nome da classe, interpretar conexão das relações, interpretar conexão das multiplicidades, interpretar texto da multiplicidade. Após o reconhecimento é gerado o arquivo XMI para ser utilizado em alguma ferramenta de edição de modelos UML.

4.2.1. Detecção dos Elementos

A detecção dos elementos é feita em duas etapas. A primeira etapa ocorre em tempo real quando o usuário abre o aplicativo. Neste momento nossa ferramenta processa cada imagem por segundo e localiza os elementos, como se pode observar na Figura 2. O Segundo passo é o momento de extração dos elementos, onde além de serem detectados os mesmos são classificados e interpretados. Os elementos das classes são extraídos por

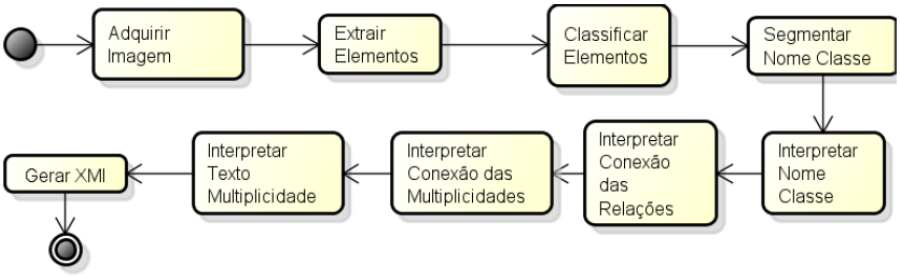

Figura 1. Processo de reconhecimento.

meio de uma série de processos bastante simples. Primeiramente captura-se a imagem por meio de um toque na tela no smartphone ou tablet. Este toque na tela inicia o processo de reconhecimento, pausando a detecção em tempo real e iniciando o reconhecimento completo da imagem.

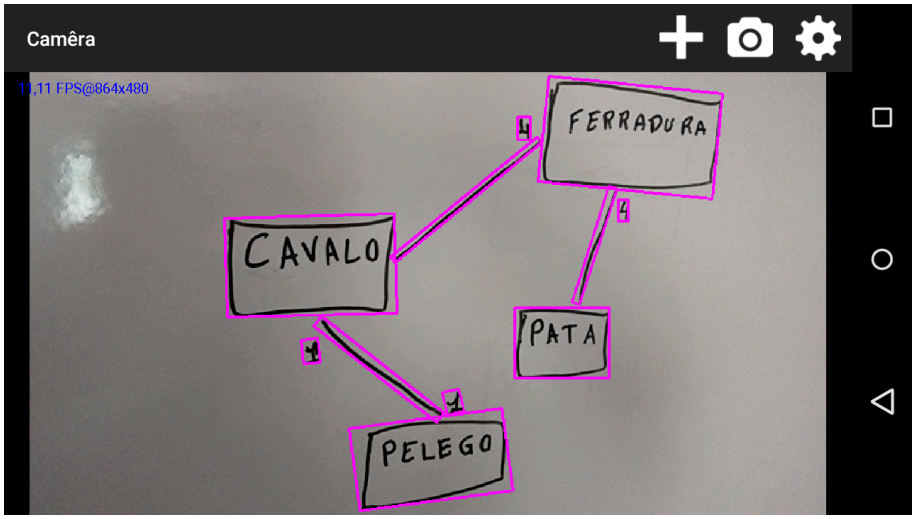

Figura 2. Detecção dos elementos em tempo real.

A imagem capturada é processada de modo a transformá-la para uma escala de cinza e facilitar a extração de características. Com a imagem em escala de cinza, utilizamos um operador de morfologia matemática denominada de *Morphological Gradient* que é a diferença entre a erosão e dilatação da imagem.

Com as bordas da imagem tratadas, utilizamos uma técnica para segmentação de bordas denominada de Limiarização (*Threshold*).

Com a imagem segmentada, utilizando o método "findContours" do Open CV todos os contornos são procurados. Extraímos apenas os contornos externos primeiramente, com o objetivo de saber onde está localizado cada elemento. Nesse momento ainda não se faz necessário pegar outras informações, como as letras da classe.

4.2.2. Reconhecimento de Classe

O reconhecimento de classe passa por duas etapas denominadas de classificação e de interpretação de nome. Na qual na primeira etapa vários elementos são extraídos da ima-

gem. Um elemento pode ser considerado uma classe quando o mesmo possuir o tamanho da área maior que 1000 pixeis e também possuir um nome.

Na segunda etapa o nome da classe é interpretado. Primeiramente é necessário pegar a imagem contendo apenas a classe. A partir dessa imagem utilizamos o método de limiarização para segmentar as bordas da imagem. Como se tem a conveniência de fazer o reconhecimento apenas do nome da classe e não de toda estrutura, teve-se que separar as bordas da classe do nome, para isso é utilizado a diferença de matrizes. Pegamos as bordas externas da classe e depois todas as bordas e subtraímos os valores, sobrando apenas o nome da classe.

Com o nome da classe já tratado, levamos a imagem para o reconhecimento por meio do OCR (*Optical Character Recognition*) chamado de Tesseract. Como estamos trabalhando e operando em um ambiente fechado, o OCR está treinado apenas para reconhecer letras de forma.

4.2.3. Reconhecimento das Associações

Atualmente nosso software suporta o reconhecimento de associações. As associações simples são diferenciadas dos demais elementos pela diferença entre sua largura e altura. O elemento é classificado como uma relação, caso a altura for cinco vezes maior que a largura ou vice-versa.

Depois de encontrada as relações do diagrama, deve-se verificar a quais classes elas estão relacionadas. As relações não devem encostar nas classes por que existe uma limitação no aplicativo (não foi encontrada nenhuma técnica viável para separação das linhas), ou seja, devem apenas estar perto. Por meio de um cálculo de proximidade, utilizando as extremidades da relação, é possível verificar quais classes estão conectadas.

Utilizando uma fórmula derivada do Teorema de Pitágoras, mostrada na Figura 3, medimos a distância entre o centro da classe e a extremidade da associação.

$$\sqrt{ditancia = (pontoBX - pontoAX)^2 + (pontoBY - pontoAY)^2}$$

Figura 3. Fórmula derivada do Teorema de Pitágoras

A melhor solução é utilizar a média de tamanho da classe, considerando altura e largura. Então pegamos o resultado da distância e verificamos com a média de tamanho da classe. Caso a distância for menor que a média de tamanho da classe, a associação está conectada.

4.2.4. Reconhecimento de Multiplicidade

São definidos como multiplicidade os elementos com área maior que 100 pixeis e menor que 1000 pixeis. As multiplicidades são ligeiramente menores que as classes, então nesse contexto foi suficiente para diferencia-las.

O cálculo de proximidade é o mesmo utilizado nas relações, com a diferença que nesse contexto as multiplicidades ficam mais próximas das relações. Então, apenas é ve-

rificado se cada multiplicidade está próxima da sua respectiva extremidade da associação. Com a proximidade verificada, os caracteres são reconhecidos com o OCR.

5. Verificação e Validação da Solução

Com o propósito de identificar se o software possui defeitos e está de acordo com o especificado, elaboraram-se alguns testes. Para a verificação da solução e a realização desses testes capturamos 10 diagramas de classe com seus respectivos XMI, a captura da imagem foi realizada por meio do smartphone Moto G2 com a resolução minima de 864 x 480. Os diagramas testados estão inseridos em um ambiente controlado no qual deveriam seguir as seguintes regras:

- As associações não devem encostar nas classes.
- As multiplicidades não devem encostar nas relações.
- As classes devem estar fechadas (aberturas mínimas são retiradas por meio da morfologia matemática, porém aberturas maiores não).
- O nome da classe deve ser escrito em letra de forma o mais legível possível.
- O nome da classe não devem encostar nas bordas da classe.

Comparamos cada classe com o seu respectivo arquivo XMI. Com os valores de reconhecimento fizemos uma média geral conseguindo como resultado a % de acerto na detecção e reconhecimento, como se pode observar na Tabela 1.

Tabela 1. Resultado dos testes

Elemento	**% Detecção**	**% Reconhecimento**
Classes	100%	80%
Associações	100%	90%
Multiplicidades	70%	30%

De modo a validar a ferramenta, foi marcado um horário onde se contou com a participação de 4 acadêmicos do curso de engenharia de software (7 Semestre e 8 Semestre), esses alunos possuem em sua grade curricular, disciplinas que fazem uso da UML. Cada aluno efetuou a instalação da ferramenta em seu dispositivo móvel, em seguida, desenhou alguns diagramas. Por fim, foi repassado um formulário para preenchimento, compondo-se por perguntas relacionadas a facilidade de instalação, facilidade de uso, tolerância a falhas, relevância da ferramenta e qual seria o próximo diagrama mais relevante para ser reconhecido pela ferramenta.

Vale ressaltar que a criação do diagrama de classe efetuado por cada graduando respeitou algumas regras de um ambiente controlado de validação de uso, limitando-se: criação de classes, multiplicidades, relações de associação simples não direcionais, elementos desenhados respeitando uma pequena distância de cada outro elemento, o uso de letras garrafais e o uso em ambientes com poucas sombras. Assim sendo, essa é uma validação do funcionamento do todo e não podendo ser usada para validar a assertividade do reconhecimento.

Na Figura 4 encontram-se as respostas dos alunos, onde se pode observar grande interesse por parte dos mesmos na ferramenta. Outro fator foi que alguns tiveram certas dificuldades na instalação, pois necessita-se baixar o Open CV inicialmente antes da

utilização do aplicativo. Ressalta-se também que, encontraram-se problemas com a conexão das relações, mas problema esse que já foi resolvido na versão atual do software.

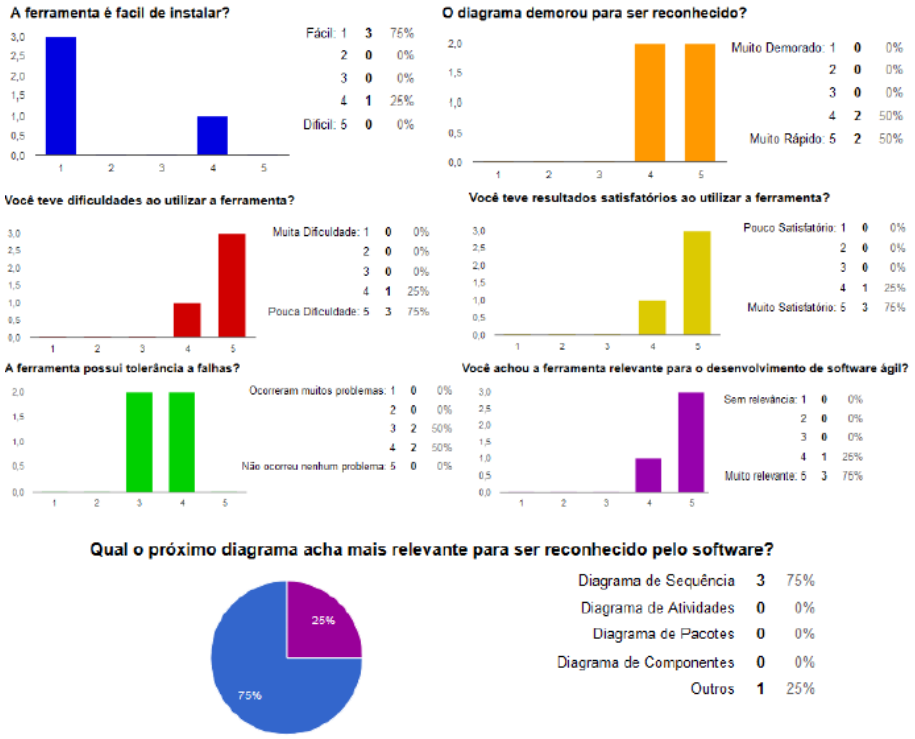

Figura 4. Respostas dos alunos que participaram da validação.

6. Conclusão

O principal objetivo do nosso trabalho foi criar uma ferramenta para extrair esboços de diagramas de classe da UML. Esta ferramenta busca minimizar o trabalho de equipes de software, no qual normalmente necessitam reconstruir os diagramas feitos em quadros brancos em uma ferramenta CASE apenas para geração do código ou atualização da arquitetura do software.

A construção da ferramenta de reconhecimento de esboço passou por alguns desafios. O primeiro desafio foi limitar o uso da ferramenta, no qual seu funcionamento encontra-se limitado a um ambiente controlado, tratando apenas de elementos simples de um diagrama de classes da UML. O segundo é a grande variação do esboço. Cada pessoa possui uma forma de desenhar ou escrever, o que dificulta o reconhecimento, porém não tornou a tarefa impossível.

A ferramenta causa um impacto muito grande em quem a usa (em ambiente controlado), pois apesar do ambiente controlado, ela automatiza um processo que muitas vezes é demorado e causa retrabalho. Outro ponto é que apenas com um smartphone ou tablet é possível reconhecer um diagrama e enviar direto para a nuvem, sem gerar retrabalhos.

Os trabalhos futuros em cima da ferramenta serão focados na utilização de inteligência artificial para realizar a segmentação dos elementos do quadro branco e a

classificação dos elementos do esboço. Logo, será possível aumentar o número de diagramas UML reconhecidos pela ferramenta, extrapolando o reconhecimento apenas do diagrama de classes.

Referências

Awal, A.-M., Feng, G., Mouchere, H., and Viard-Gaudin, C. (2011). First experiments on a new online handwritten flowchart database. In *IS&T/SPIE Electronic Imaging*, pages 78740A–78740A. International Society for Optics and Photonics.

Booch, G., Rumbaugh, J., and Jacobson, I. (2000). *UML-GUIA DO USUARIO: TRADUÇAO DA SEGUNDA EDIÇAO*. Elsevier Brasil.

Buchmann, T. (2012). Towards tool support for agile modeling: sketching equals modeling. In *Proceedings of the 2012 Extreme Modeling Workshop*, pages 9–14. ACM.

Costagliola, G., De Rosa, M., and Fuccella, V. (2014). Local context-based recognition of sketched diagrams. *Journal of Visual Languages & Computing*, 25(6):955–962.

Gonzalez, R. C. and Woods, R. E. (2010). Processamento digital de imagens. traduçao: Cristina yamagami e leonardo piamonte.

Jiang, Y., Tian, F., Zhang, X. L., Dai, G., and Wang, H. (2011). Understanding, manipulating and searching hand-drawn concept maps. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(1):11.

Liao, S. and Duan, M. (2012). Sketch recognition via string kernel. In *Natural Computation (ICNC), 2012 Eighth International Conference on*, pages 101–105. IEEE.

Maggiori, E., Gervasoni, L., Antúnez, M., Rago, A., and Díaz Pace, J. A. (2014). Towards recovering architectural information from images of architectural diagrams. In *XLIII Jornadas Argentinas de Informática e Investigación Operativa (43JAIIO)-XV Simposio Argentino de Ingeniería de Software (Buenos Aires, 2014)*.

OMG (2015). Omg unified modeling language.

Pedrini, H. and Schwartz, W. R. (2008). *Análise de imagens digitais: princípios, algoritmos e aplicações*. Thomson Learning.

Wüest, D., Seyff, N., and Glinz, M. (2013). Flexisketch: A mobile sketching tool for software modeling. In *Mobile Computing, Applications, and Services*, pages 225–244. Springer.