ADOÇÃO DE FERRAMENTAS *CASE* GRATUITAS PARA UML, EM UMA EMPRESA NIBESS: UM ESTUDO DE CASO

Alessandro João Brassanini[1], Luiz Camargo[2]

Resumo: *Com a publicação crescente de livros e artigos sobre o desenvolvimento de software, os profissionais da área têm alcançado mais sucesso em seus projetos, pois há um compartilhamento maior de experiências. As ferramentas CASE (Computer Aided Software Engineering) a disposição destes profissionais está ganhando cada vez mais adeptos na área de desenvolvimento de software, além de várias comunidades que trabalham em projetos open source. O presente artigo tem como objetivo descrever e ilustrar algumas ferramentas sem custo de licenças para desenvolvimento de diagramas da notação UML (Unified Modeling Language), em detrimento as ferramentas desenho como Microsoft Visio. Ao adotar essas ferramentas, avaliar as implicações para o processo de desenvolvimento de software e as razões para não adoção dessas ferramentas. Quando adotadas certificar-se da aplicação do uso adequado ou se é somente para desenho. E por fim, descrever quais os benefícios em usá-las. O método adotado para avaliar essas ferramentas são pesquisas, e o uso das ferramentas de forma exploratória em um ambiente de testes a fim de descrever um conjunto de resultados e funcionalidades. Baseado nesses resultados, concluir se as ferramentas CASE gratuitas podem ou não ajudar de alguma forma a melhorar o processo e qualidade de desenvolvimento de sistemas.*

***Palavras-chave*:** *Ferramentas cases livres. UML. Engenharia de software.*

1 INTRODUÇÃO

Utilizar software faz parte do cotidiano das pessoas em diversas situações, como por exemplo, em indústrias, hospitais, robôs, casas, segurança, entre inúmeras possibilidades que estão surgindo a todo o momento.

Entretanto, desenvolver sistemas computacionais ainda precisa melhorar em muito a qualidade do produto final do software. É comum ver noticiários ou relatos sobre problemas e falhas em software (ENGHOLM, 2010).

Por esse motivo existe um processo de desenvolvimento de software com normas e modelos de qualidade. Um dos principais modelos, cita-se o *CMMI* (*Capability Matury Model Integration*) e o modelo brasileiro chamado de MPS.BR (Melhoria do Processo de Software Brasileiro). Cada um têm suas características especificas, porém com os mesmos propósitos (RIOS & MOREIRA, 2013).

Para se alcançar um nível de qualidade de software dentro dos padrões esperados, a Engenharia de Software assim como a Engenharia Civil ou Mecânica têm suas técnicas e modelos bem consolidados na indústria.

Um dos itens para se chegar a qualidade é realizar a modelagem dos sistemas através da *UML*, ao qual permiti especificar, construir, visualizar e documentar os artefatos envolvidos no Sistema de Informação através do paradigma da orientação a objetos (SILVA, 2007).

O presente artigo tem como objetivo principal descrever o cenário atual de uma empresa de grande porte no setor logístico localizada na cidade de Navegantes, Santa Catarina, que utiliza a TI para apoiar fortemente seus negócios. Sendo que no desenvolvimento do seu software interno, um dos softwares utilizado é o *Microsoft Visio* para produção de diagramas da *UML*.

Os objetivos específicos deste artigo contemplam em abordar um estudo de caso e experimentar algumas ferramentas *CASES* de licença livre especificas para *UML*. Com isso espera-

1 Pós-Graduando em Engenharia de Software - SOCIESC. e-mail: alessandrobrassanini@gmail.com
2 Professor de Pós-Graduação e Graduação - SOCIESC. e-mail: camargho@gmail.com

se apresentar quais as implicações para o processo de desenvolvimento de software, a razão da não adoção de uma ferramenta *CASE* e os benefícios que a adoção pode trazer para o desenvolvimento e analise dos sistemas.

Portanto, este artigo esta organizado da seguinte forma: Na primeira seção é apresentada uma introdução dos temas que serão abordados no artigo. Na segunda seção é abordada uma introdução a Engenharia de Software. Na terceira seção serão analisadas as principais funcionalidades de algumas ferramentas *CASES* para *UML* com licenças sem custo. Na quarta seção será apresentado o cenário atual no desenvolvimento de software desta empresa de logística e o cenário proposto para melhoria deste cenário através do estudo de caso e adoção de uma ferramenta *CASE* para *UML* de licença livre. E na última parte apresenta as conclusões do artigo.

2 UML

A *UML* (*Unified Modeling Language*) é uma linguagem para especificação, documentação, visualização e desenvolvimento de softwares orientados a objetos. É uma linguagem muito bem difundida para desenvolvimento de sistemas que reúne melhores práticas de metodologia.

É uma notação aberta e mantida pela *OMG (Object Management Group)*, um consórcio de empresas que definem padrões relacionados à orientação a objetos. A *UML* é adaptável conforme a necessidade, não é uma notação rígida, ou seja, elementos de um diagrama podem participar em outros diagramas. Neste modelo, diversos diagramas ajudam na visualização do problema e a concepção da solução, permitindo uma visão dos objetos e seus relacionamentos. Propõe-se um visual para especificação (modelagem) de sistemas orientados a objetos, fornece representação gráfica para os elementos essenciais do paradigma de objetos como classes, atributos, objetos, troca de mensagens, entre outros.

Um dos principais intuitos da *UML* é permitir que as pessoas ou equipe entendam os principais aspectos do sistema antes a ser codificado ou produzido. Para ser mais esclarecedor é dizer que a *UML*, vem para facilitar a comunicação entre as equipes de desenvolvimento de software. A *UML* foi criada com o foco no paradigma orientado a objetos, porém, não é um padrão restrito somente a esse paradigma. Pode ser usado até mesmo no tradicional paradigma procedural.

Conforme citado por Pressman (2011, p.727), assim como os arquitetos criam plantas e projetos para ser usados por uma empresa de construção, os arquitetos de software criam diagramas *UML* para ajudar os desenvolvedores de software a construir o software.

Segundo LIMA (2014, p. 26), a *UML* é uma linguagem completa e cheia de recursos, que permiti não apenas capturar as informações, mas também expressá-las com uma sintaxe clara e objetiva. Os modelos são documentados visualmente e possibilitam a produção de artefatos que podem ser publicados e mantidos com facilidade, qualidade e produtividade.

Dentre os diversos diagramas da *UML*, o de classes é o mais importante do paradigma orientada a objetos e pode ter uma definição iniciada já na fase de análise, apenas identificando as classes candidatas (ENGHOLM, 2013).

A *UML* pode ser utilizada no apoio ás fases de análise, projeto e implementação, pois nos permiti pensar antes de codificar. A *UML* trabalha com um conjunto de diagramas que permitem uma notação clara e consistente. Facilita a comunicação entre a equipe de desenvolvimento, bem com dela com usuários do sistema, aumentando assim a participação do time do projeto (GÓES, 2014).

3 ENGENHARIA DE SOFTWARE

Segundo PRESSMAN (2011, p.31) o software distribui o produto mais importante da nossa era – a informação. Ele transforma dados pessoais (por exemplo, transações financeiras de um

indivíduo) de modo que possa ser mais úteis num determinado contexto; gerencia informações comerciais para aumentar a competitividade; fornece um portal para redes mundiais de informação (internet) e aos meios para obter informações sob todas as suas formas.

Através da Engenharia de Software, foram desenvolvidas diversas técnicas e métodos para ajudar a resolver problemas na codificação dos softwares. Neste contexto pode-se afirmar que a Engenharia de Software é composta por quatro camadas importantes como: ferramentas, métodos, processo e foco na qualidade (PRESSMAN, 2011).

Considera-se que, "software de sucesso" significa "muitas pessoas usam e amam seu software", o profissional de tecnologia da informação precisa prestar muita atenção aos seus usuários. O Google cita um lema famoso: "Concentre-se no usuário e tudo o mais será consequência" (FITZPATRICK & SUSSMAN, 2013).

Para SILVA (2009, p.17), um software de alta complexidade não é algo que se produza simplesmente sentando diante de um ambiente de suporte a programação e gerando código. Como qualquer desenvolvimento de alta complexidade, a produção em si precisa ser precedida de algum planejamento. Esse planejamento que o autor cita pode ser feito através dos conceitos e técnicas da Engenharia de Software.

Outro item importante na Engenharia de Software é a engenharia de requisitos, o qual ajuda por meio de técnicas definir a melhor forma para coletar os requisitos. A engenharia de requisitos é responsável pelo planejamento, pela execução e pelo controle dos requisitos do projeto, servindo de base para a identificação, a documentação, a validação e o rastreamento, além de fornecer informações necessárias para a realização de testes e elaboração do manual do usuário (ENGHOLM, 2013).

4 MODELO DA ÁREA DE TI DO ESTUDO DE CASO

O modelo da área de Tecnologia da informação deste estudo de caso é de uma empresa de grande porte na área de logística que atende vários seguimentos multimodais[3]. A tecnologia da informação é formada por desenvolvedores, analistas de sistemas e equipe de infraestrutura. As demandas das implementações do sistema são passadas aos analistas de sistemas através de reuniões. Depois deste processo é descrito em um documento texto com diagramas as necessidades a serem desenvolvidas. As demandas solicitadas são geralmente implementações e melhorias no sistema de controle de multimodais, para atender regras de negócio dos clientes internos e externos. O sistema é de código legado e foi desenvolvido em linguagens de programação *Delphi 7 e Visual FoxPro9.*

Neste documento texto é inserido as regras de negócio e diagramas da *UML*. E a ferramenta utilizada para confeccionar os diagramas é o *Microsoft Visio* da versão 2010. Os diagramas desenvolvidos são na maioria das vezes o diagrama de caso de uso, diagrama de atividades, diagramas de classe e diagramas de componentes. A metodologia para o desenvolvimento é o interativo incremental.

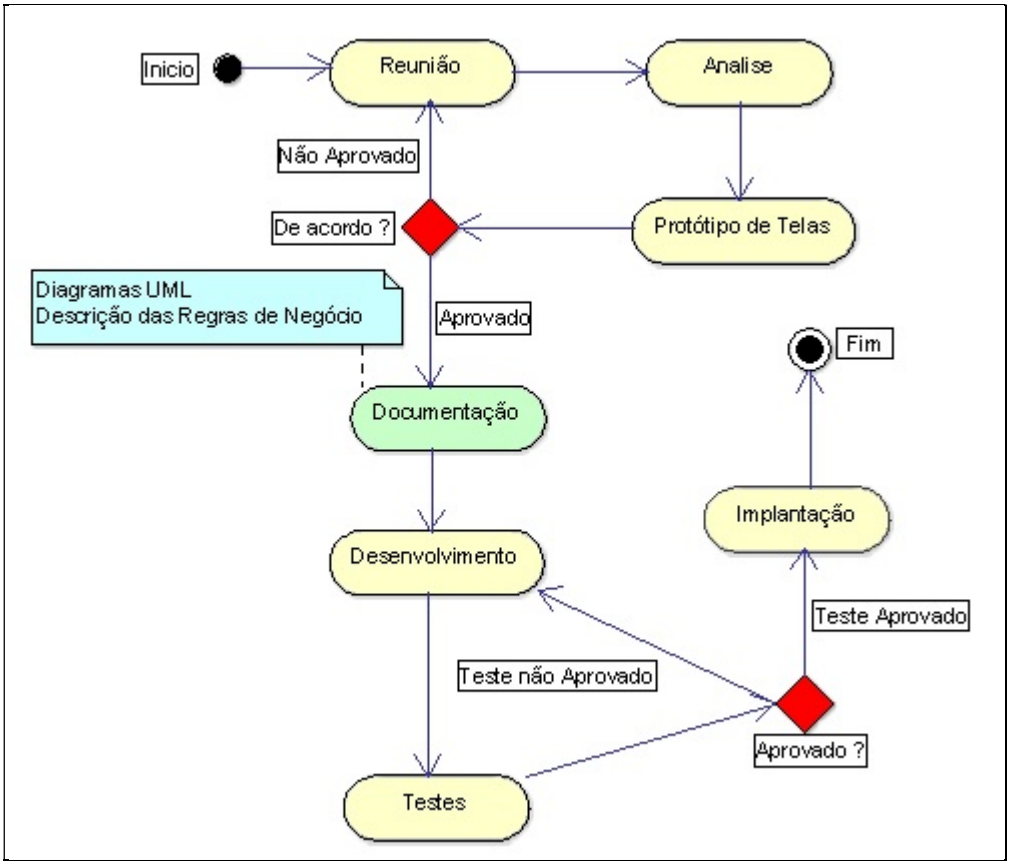
4.1 Contextualizações do problema

Por ser uma ferramenta de desenho a mesma não tem as facilidades e funcionalidades que uma ferramenta *CASE* possui. Isso acaba gerando mais tempo para ser diagramado no sentido de não permitir fazer um aproveitamento de um diagrama para outro.

[3] Multimodal: Utilizam-se duas ou mais modalidades de transporte, ou seja, pode ser terrestre, marítimo, ferroviário ou aéreo.

Na figura 1 encontra-se o processo para desenvolvimento e melhorias no software. Inicialmente os gerentes operacionais de diversos modais se reúnem com os analistas de sistemas para informar as demandas. Depois de discutido em reunião as implementações ou melhorias, os analistas realizam uma análise contemplando com protótipos de telas feitas na própria linguagem de programação. Tendo protótipo de tela feito, é realizada uma nova reunião com os gerentes operacionais para alinhar o acordo. Se os envolvidos estiverem de acordo, os analistas desenvolvem a documentação que seria os diagramas feitos no *Microsoft Visio*, com a notação da *UML*. Também é redigido em documento texto às regras de negócio. O próximo passo são os testes e implantação do novo recurso.

Figura 1 – Processo atual da TI representado através de um diagrama de atividades



Fonte: O autor, adaptado do processo empresa Unicalog (2016)

Existe um processo desenhado para o desenvolvimento do software conforme figura 1, mas a atividade de documentação que consiste na criação de diagramas pode ser otimizada utilizando ferramentas livres a fim de criar os diagramas da *UML*.

5 FERRAMENTAS UTILIZADAS PARA O ESTUDO DE CASO

Nesta seção são descritos as ferramentas utilizadas para o experimento na diagramação dos artefatos produzidos utilizando a notação da *UML*.

5.1 ARGOUML

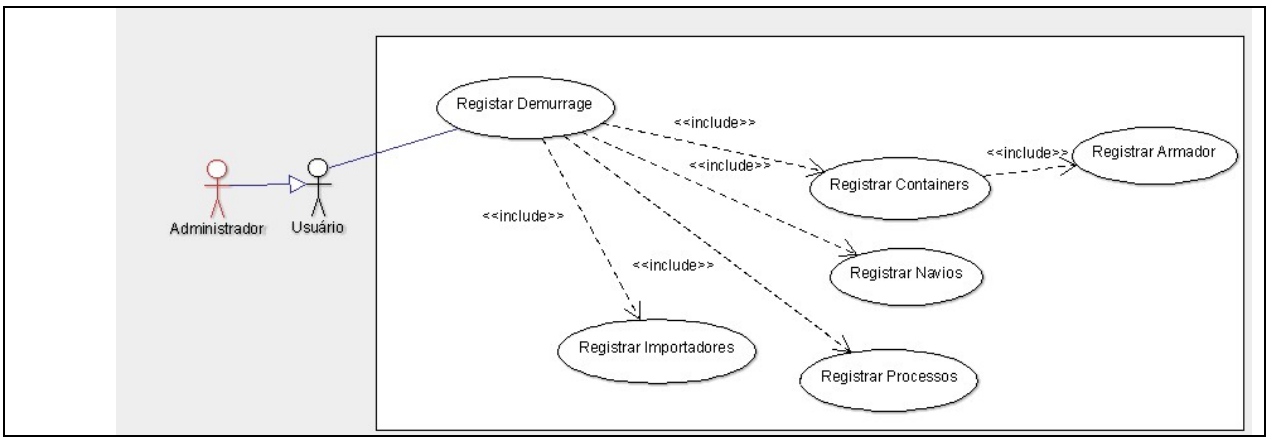
Segundo (AMBIENTE LIVRE, 2016) a ferramenta *CASE*, ArgoUML que pode ser encontrada no seguinte endereço eletrônico http://www.argoUML.tigris.org. É uma ferramenta para diagramação *UML*, fornecendo a geração básica do código fonte. Exporta diagramas em *PostScript,* RPS, GIF ou SVG. Funciona em qualquer versão do Java 1.2. (Usa a licença BSD). Essa ferramenta para diagramação foi desenvolvida na Universidade da Califórnia, Berkeley.

Na versão 0.34 a ferramenta tem seus menus e telas traduzidos inclusive para o Português do Brasil. Uma das principais capacidades ou funcionalidades do ArgoUML é criar as declarações de classe em Java além de exportar para páginas web os diagramas desenvolvidos. Outro recurso importante é que o ArgoUML organiza os diagramas em forma de projetos. Para cada tipo de diagrama produzido no projeto, o analista de sistemas pode também definir propriedades, textos livres além de poder alterar cores dos objetos da *UML*.

Apesar de o padrão *UML* estar na versão 2.5 o ArgoUML utiliza a versão do padrão 1.4. Mesmo assim, segundo manual do software não se torna um problema, pois são mantidos os conceitos mais importantes e tradicionais da *UML*.

Conforme figura 2, pode-se observar um diagrama de caso de uso desenvolvido no ArgoUML. Para cada caso de uso abaixo o analista de sistemas, pode descrever os fluxos alternativos e secundários dentro da própria ferramenta, sem necessidade de ter que escrever isso em um arquivo texto separado do diagrama de caso de uso. Centralizando assim melhor a documentação.

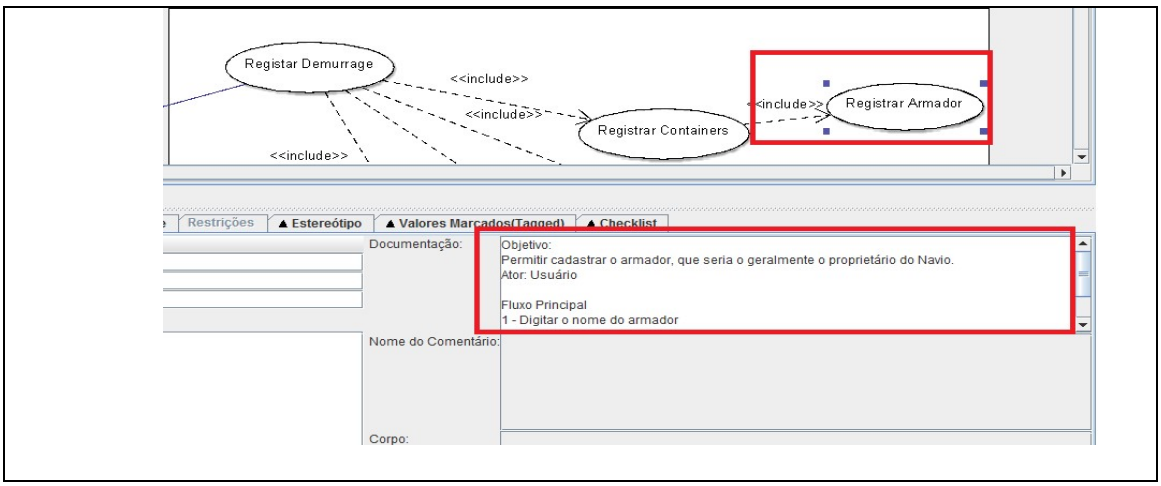
Figura 2 – Diagrama de caso de uso desenvolvido no ArgoUML



Fonte: O autor (2016)

Na figura 3 pode-se verificar o caso de uso “Registrar Armador”, onde pode ser especificado todo o fluxo principal e alternativo de forma textual com a ferramenta.

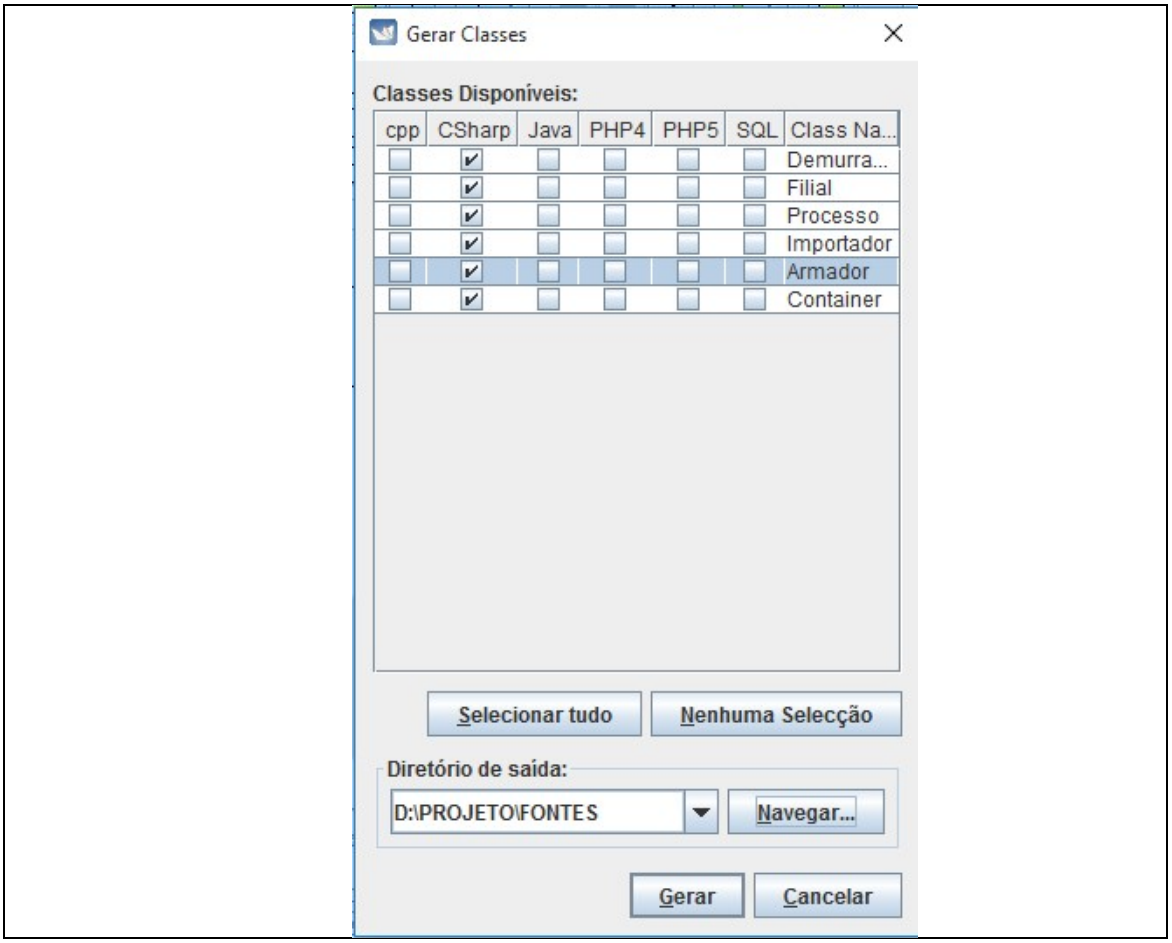
Figura 3 – Tela com a descrição de um caso de uso na forma de textos



Fonte: O autor (2016)

É possível também gerar o código fonte a partir do diagrama de classes conforme pode ser visualizado na figura 4. O código fonte pode ser gerado em diferentes linguagens de programação como C#, Java, C++ e PHP. Quanto mais detalhado o diagrama de classes, mas completo vai ser o código fonte gerado.

Figura 4 – Tela do gerador de código fonte do ArgoUML



Fonte: O autor (2016)

5.2 ASTAH

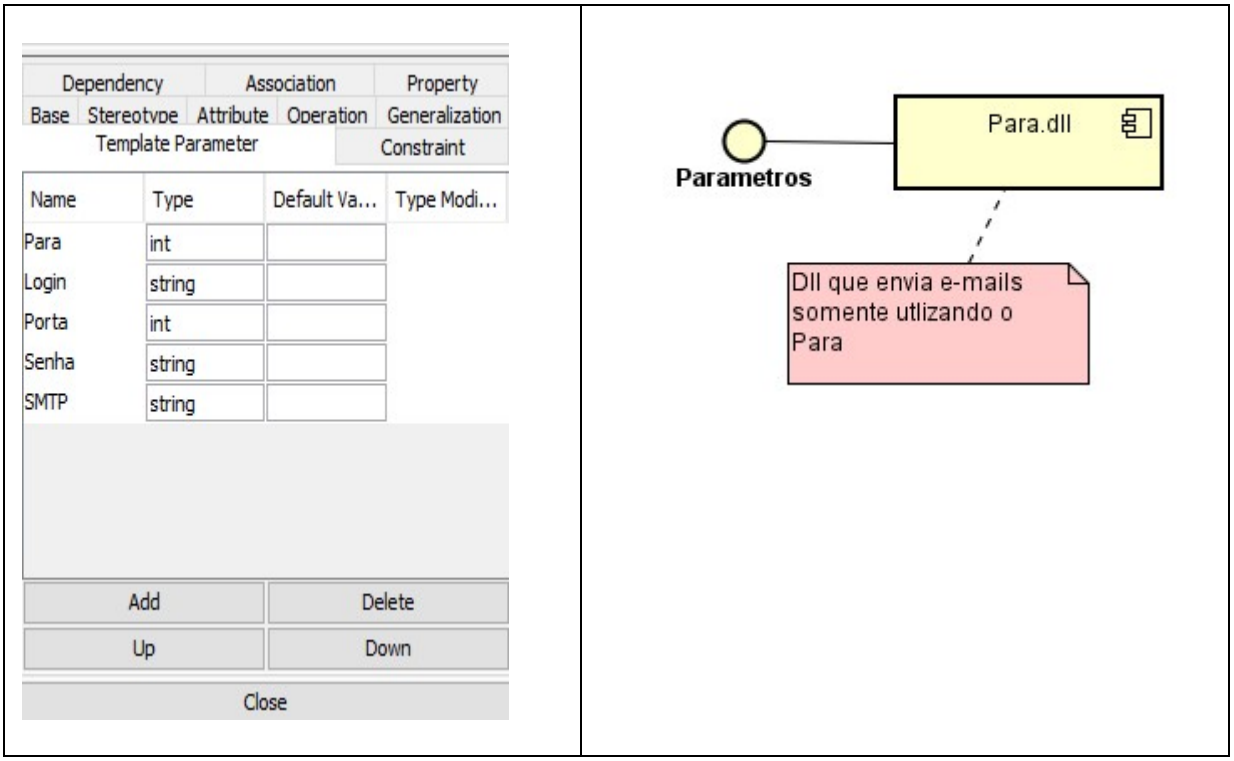
Esta ferramenta também tem uma versão para comunidade chamada Astah UML Community, a versão para comunidade não suporta todos os serviços e opções que estão disponível somente nas versões *standard* ou *professional*. No entanto, para quem deseja praticar a *UML*, a edição para a comunidade é uma boa alternativa, apresentando um ambiente amigável e de fácil compreensão conforme descrito no site www.ambientelivre.com.br (AMBIENTE LIVRE, 2016).

Os principais diagramas suportados pelo Astah são o diagrama de caso de uso, classes, sequência, atividades, comunicação e componentes. O Astah utiliza a *UML* padrão 2.5 da *UML*. A ferramenta também possui versão para Mac OS e Linux. Infelizmente a versão do Astah para comunidade não permiti gerar o código fonte a partir do diagrama de classes. Somente a versão profissional tem esse recurso. Recurso esse que esta presente na ultima versão da ferramenta ArgoUML. Assim como o ArgoUML o Astah também pode armazenar no mesmo arquivo diversos diagramas produzidos.

No desenvolvimento dos diagramas de caso de uso dentro do Astah, também é possível descrever os fluxos principais e secundários dentro da ferramenta. Centralizando assim a documentação no mesmo artefato que é o diagrama de caso de uso. Uma das desvantagens é que a ferramenta para comunidade não permiti a impressão desse texto.

Dentre as diversas funcionalidades da ferramenta Astah, pode-se destacar o *Template Parameter*, onde, por exemplo, podem ser especificados os parâmetros que o componente fornece conforme figura 5.

Figura 5 – Exemplo de um diagrama de componentes utilizando os parâmetros



Fonte: O autor (2016)

5.3 DIA

A ferramenta para diagramação chamada DIA, encontrada no site http://www.lysator.liu.se/~alla/dia, realiza muitas das mesmas funções que o *Microsoft Visio* com suporte específico para *UML*. (Usa a licença GNU).

Segundo site da ferramenta, o software pode ser usado para desenhar muitos tipos diferentes de diagramas. Atualmente tem objetos especiais para ajudar a desenhar diagramas entidade relacionamento, diagramas *UML*, fluxogramas, diagramas de rede e muitos outros diagramas. A ferramenta pode carregar e salvar diagramas para um formato XML personalizado (compactado por padrão, para economizar espaço), pode exportar diagramas para um número de formatos, incluindo EPS, SVG, XFIG, WMF e PNG, e pode imprimir diagramas (incluindo os que abrangem várias páginas).

Não é possível realização a geração de código fonte em nenhuma linguagem de programação além de não permitir gerar diagramas a partir de outro. Uma das principais funcionalidades desta ferramenta é a geração de scripts de banco de dados (DIA, 2016).

5.4 UMBRELLO

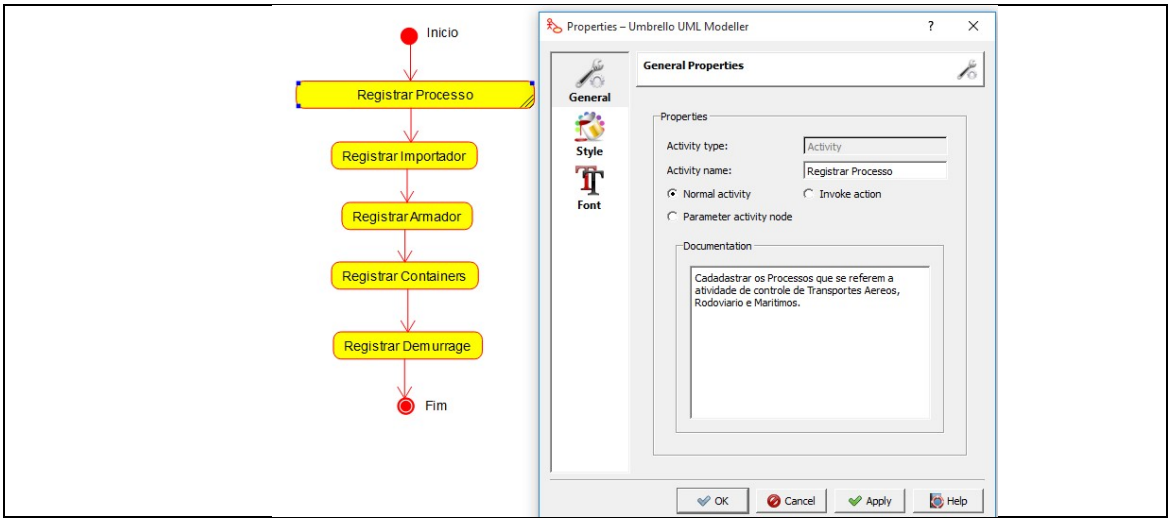
Esta ferramenta possui versão para Windows além de Linux. Umbrello vem geralmente contido nas distribuições Linux e está disponível através de seu gerenciador de pacotes. É um projeto *open source*.

A Umbrello UML Modeller pode gerar código fonte a partir de várias linguagens de programação baseadas em diagramas de classes. O código gerado consiste em declarações de classe, com seus métodos e atributos, fornecendo a funcionalidade das suas operações de classes. A Umbrello UML Modeller da versão 4.14 vem com suporte à geração de código para *Action Script*, Ada, C++, C#, D, IDL, Java, JavaScript, MySQL e Pascal (UMBRELLO, 2016).

Para melhor organizar o seu modelo, especialmente para grandes projetos, pode se criar pastas lógicas na visão de árvore. Apenas selecione a opção Nova Pasta no menu de contexto das pastas padrão na visão de árvore para criá-las. Pastas podem ser aninhadas, e você pode mover objetos arrastando-os de uma pasta e soltando-os em outra.

Para ilustrar melhor a ferramenta, segue a figura 6 onde é mostrado um diagrama de atividades juntamente com as propriedades de uma determinada atividade. Inclusive é possível visualizar a descrição de uma determinada atividade. Para cada atividade o profissional pode descrever um texto explicativo.

Figura 6 – Exemplo de um diagrama de atividades desenvolvido no Umbrello



Fonte: O autor (2016)

5.5 VISUAL PARADIGM COMMUNITY

Visual Paradigm é uma ferramenta *CASE* que suporta a elaboração dos diversos diagramas da *UML*, diagramas de requisitos SysML e os diagramas ER (Entidade Relacionamento). Também permiti a elaboração e manipulação do projeto de modelagem. A ferramenta oferece suporte a transformações especificas para código fonte de algumas linguagens de programação tradicional, como C++ e Java (FURGERI, 2013).

Permiti a criação de outras notações como a BPMP. A ferramenta atualmente utiliza o padrão 2.0 da *UML*. Existem versões para Mac OS, Linux e Windows.

Apesar de ser uma ferramenta comercial a mesma tem uma versão livre para comunidade. E mesmo sendo uma versão livre o Visual Paradigm tem a possibilidade de gerar o código fonte a partir, por exemplo, de um diagrama de classes (DEVMEDIA, 2013).

O Visual Paradigm também permiti a importação e exportação de modelos no formato *XMI (XML Metadata Interchange).* É possível importar e exportar em formato XMI. O projetista pode exportar o modelo atual para o formato XMI ou importar um modelo já criado anteriormente.

Outra funcionalidade que vai contribuir e beneficiar na produtividade são a geração de *script* para o banco de dados, isso beneficia muito a produtividade no processo de desenvolvimento de software (VISUAL PARADIGM, 2016).

6 ADOÇÃO DE FERRAMENTAS CASES LIVRES PARA O ESTUDO DE CASO

Na apresentação das ferramentas *CASES*, foram ilustradas algumas de suas capacidades e funcionalidades que estão à disposição dos profissionais de desenvolvimento de software. O estudo de caso visa ilustrar e descrever sobre a adoção de ferramentas *CASE* para a diagramação em detrimento a ferramentas desenho. Para esse estudo de caso, utiliza-se o *Microsoft Visio*, que é uma software que mistura desenho vetorial com desenho artístico.

O estudo de caso visa descrever as implicações, os benefícios e não adoção dessas ferramentas no processo de desenvolvimento de software na atual estrutura de TI de uma empresa NIBESS.

Pode-se citar algumas implicações caso fosse adotado uma ferramenta *CASE* livre para criação dos diagramas da *UML*, como a ferramenta deixar de ser produzida. Esse é um risco pequeno, mas considerável, mesmo sendo uma ferramenta *open source*. Outra implicação seria se houvesse necessidade de importar os diagramas atuais para a ferramenta adotada. Pois há necessidade de constante manutenção nos diagramas produzidos. Numa empresa NIBESS deste estudo de caso não seria um problema, pois na grande maioria são produzidos novos diagramas. Mas certamente em uma empresa que desenvolve sistemas para diversos clientes seria um problema a ser estudado. Independente da ferramenta *CASE* adotada, vai também exigir tempo de adaptação dos profissionais, claro que isso não chega a ser um problema, pois é um processo natural de qualquer aprendizado.

No experimento da ferramenta Umbrello foi gerado o código fonte em C# através de três classes com intuito de visualizar se o código será gerado de forma simples ou complexa. Constatou se através deste experimento que o código gerado é simples, não adicionado muito detalhes como acontece geralmente em ferramentas de frameworks.

Apesar das implicações, não há razões para não adoção de uma ferramenta *CASE*. Este contexto esta fundamentado no experimento realizado com algumas ferramentas apresentadas neste artigo. Uma questão a ser considerada, é que as ferramentas *CASE* que foram testadas nesse estudo de caso, não permitiram a transformação ou aproveitamento de um diagrama para outro. Recurso esse presente em ferramentas *CASE* comerciais pagas. Mas as ferramentas testadas, como o ArgoUML, permiti a geração de código fonte em linguagens de programação bem tradicional, como

por exemplo o C#. Não somente a geração de código fonte, mas a questão de usabilidade da ferramenta *CASE* em produzir os diagramas utilizando a notação correta da *UML*. Na *Microsoft Viso* permiti produzir até mesmo um diagrama com notação em não conformidade com as regras da *UML*.

Foi aplicado nesse estudo de caso o uso dessas ferramentas *CASES* em paralelo com o processo de desenvolvimento atual descrito na contextualização do problema. Constatou-se através da modelagem de problemas reais, que o nível de detalhamento de uma ferramenta *CASE* proporciona, é mais eficiente que um software como o *Microsoft Visio*. Pode-se citar como exemplo, um diagrama de classes desenvolvido pela da ferramenta Astah e adicionar atributos e métodos o qual já traz como padrão pela ferramenta a visibilidade (- ou +) e o tipo (atributos). Além disso, a possibilidade de adicionar *stereotype* nas classes de forma automatizada e padronizada o que não acontece utilizando ferramentas de desenho como o *Visio* da Microsoft. Os relacionamentos entre as classes é um dos recursos mais importantes que as ferramentas *CASES* ajudam o profissional de desenvolvimento ou analista, as mesmas permitem através de poucos *clicks* a ligação entre as classes pela regra *UML*. No *Visio* o profissional de desenvolvimento tem toda liberdade de realizar os relacionamentos entre as classes, mas pode cometer erros. Fato que acontece na prática esse problema, não só no diagrama de classes, mas no diagrama de caso de uso também.

De acordo com o quadro 1 abaixo, pode-se visualizar melhor as características entre as ferramentas *CASE* do experimento, evidenciando as funcionalidades que pode contribuir para obter um melhor resultado no processo de desenvolvimento de software:

Quadro 1 – Resultados obtidos após o experimento das ferramentas *CASE* para *UML*

Características	FERRAMENTAS				
	ArgoUML	**Astah Community**	**DIA**	**Umbrello**	**Visual Paradigm Community**
Diagramas Suportados	Casos de uso, classes, sequência, colaboração, estados, atividades e distribuição.	Casos de uso, classes, sequencia, atividades, comunicação e componentes.	Casos de uso, classes, atividades e componentes.	Casos de usos, classes, sequencia, comunicação, estado, atividade e componentes.	Casos de usos, classes, atividades, comunicação, componentes e sequencia. Suporta também, diagramas de requisitos SysML e os diagramas ER
Plataforma Suportada	Win/Linux/Mac OS	Win/Linux/MacOS	Win/Linux/Mac OS	Win/Linux	Win/Linux/Mac OS
Gera Código Fonte	C#, Java entre outros.	Não	Não	C++, C# entre outros.	C++ e Java
Gera Scripts Banco de Dados	Sim	Não	Sim	Sim	Sim
Padrão da UML utilizado	1.4	2.5	2.x	2.x	2.0
Imprime os Diagramas	Sim	Não	Sim	Sim	Não
Licença	Open Source	Livre (não para uso comercial)	Open Source	Open Source	Livre (não para uso comercial)
Gera diagramas a partir de outro diagrama (classe x Objeto)	Não	Não	Não	Não	Sim
Permite o uso de stereotype nas classes	Sim	Sim	Não	Sim	Sim
Idiomas	Português entre outros	Inglês	Português entre outros	Inglês	Inglês
Importa arquivos do Microsoft Visio	Não	Não	Não	Não	Não

Fica evidente apenas com exemplo citado que as ferramentas *CASE* testadas no estudo de caso são realmente utilizadas adequadamente para desenvolvimento dos diagramas. Apesar de os diagramas desenvolvimentos no *Visio* também ajudarem no processo de desenvolvimento da atual estrutura, as ferramentas *CASES* são muito mais que apenas desenhos e sim um avanço para melhor modelar os sistemas, além do ganho de tempo na produtividade.

7 CONCLUSÕES

Esse trabalho descreveu e ilustrou algumas ferramentas *CASES open source e* livres para desenvolvimento de diagramas da *UML*, além de avaliar a adoção dessas ferramentas *CASES* em detrimento a ferramentas de desenho como *Microsoft Visio*.

Conforme experimentos realizados com algumas ferramentas *CASE*, constatou-se que é possível melhorar o processo de desenvolvimento de software através da utilização de uma ferramenta *CASE* especifica para *UML*. Em se tratando de desenvolvimento de software é inevitável que as mudanças ocorram. Pois frequentemente, linguagens de programação são atualizadas e até mesmo descontinuadas. No processo de desenvolvimento de software não é diferente. Novos processos são criados, novas ferramentas *CASE* são lançadas no intuito de melhorar a qualidade tanto do processo de criação de software como na qualidade do software produzido.

Os benefícios que essas ferramentas *CASE* proporcionam ao processo de desenvolvimento é superior que ao processo de TI utilizado neste estudo de caso com o *Viso*. Apesar de nenhuma ferramenta testada gerar código fonte para as linguagens de programação utilizadas pela empresa do estudo de caso, as ferramentas *CASES* permitem melhor experiência na criação dos diagramas dentro das regras da *UML*, sem perder a possibilidade de usar seu próprio formato usando a criatividade. Pois a *UML* é uma notação bem flexível de se utilizar.

Vale ressaltar a questão de geração de código fonte e scripts dos bancos de dados a partir dos diagramas de classes, pois das ferramentas *CASES* testadas, algumas permitem a geração de código fonte de algumas linguagens de programação mais populares no mundo como Java e C# segundo site do Tiobe (http://www.tiobe.com/tiobe-index).

Alguns pontos negativos das ferramentas livres, especificamente as que foram testadas neste trabalho, é que não é possível gerar todos os diagramas da *UML.* Outro recurso importante e ausente é a possibilidade de aproveitamento ou geração de um diagrama para outro. Recurso este presente em ferramentas comerciais como o *Enterprise Architect*. E por fim, a limitação na impressão dos diagramas a partir da ferramenta.

Neste artigo foram questionadas e respondidas algumas perguntas como as implicações da adoção dessas ferramentas, quais a razões para não adoção dessas ferramentas, quando adotadas são usadas corretamente ou somente para desenho e por ultimo os benefícios que elas trazem.

E por fim, como trabalho futuro, pretende-se utilizar algumas das ferramentas *CASE,* para diagramação, principalmente com mais relevância na geração dos *scripts* de banco de dados, que se mostrou muito prático e funcional nos testes realizados. Nessa parte de geração de scripts de banco de dados seria um ponto que poderia agregar valor ao negócio, além de aumentar a produtividade no processo de desenvolvimento de software.

Agradecimentos

Agradeço profundamente a minha família pelo amor dado durante a jornada nesta Especialização. A Unica Logística e Transportes Ltda, por ter fornecido todo apoio e estrutura necessária, para o desenvolvimento deste artigo. Ao Professor Luiz Camargo, pelo seu apoio dado no transcorrer do artigo. A Deus, por me proporcionar saúde, a fim de atingir meus objetivos.

REFERÊNCIAS

AMBIENTE LIVRE. Ferramentas livres para UML. Disponível em: < http://www.ambientelivre.com.br/> Acesso em: 21 de Ago. de 2016.

ANTT. Agência Nacional de Transportes Terrestres. Disponível em: < http://www.antt.gov.br/index.php/content/view/4963/Multimodal.html> Acesso em: 29 de Ago. de 2016.

DEVMEDIA. Modelagem de dados com Visual Paradigm. Disponível em: <http://www.devmedia.com.br/artigo-sql-magazine-42-modelagem-de-dados-com-a-visual-paradigm-do-modelo-de-classes-a-criacao-do-banco-de-dados/7019> Acesso em: 21 de Ago. de 2016

DIA. Site da Ferramenta. Disponível em: <http://www.lysator.liu.se/~alla/dia> Acesso em: 08 de Ago. de 2016

ENGHOLM, H.JR. Análise e Design Orientado a Objetos. São Paulo: Novatec, 2013

ENGHOLM, H.JR. Engenharia de Software na Prática. São Paulo: Novatec, 2010

ARGOUML. Site da Ferramenta. Disponível em: <http://www.argoUML.tigris.org> Acessado em: 31 de Jul. de 2016.

UMBRELLO UML. Site da Ferramenta. Disponível em: < https://umbrello.kde.org/> Acessado em: 15 de Ago. de 2016.

VISUAL PARADIGM. Site da Ferramenta. Disponível em: < https://www.visual-paradigm.com> Acessado em: 28 de Ago. de 2016.

FURGERI, Sergio. Modelagem de Sistemas Orientados a Objetos. São Paulo: Erica, 2013

FITZPATICK, B.W. SUSSMAN, B.C. Equipes de Software: Um guia para o desenvolvedor de software se relacionar melhor com outras pessoas. São Paulo: Novatec, 2013

GÓES, W.M. Aprenda UML por meio de estudos de caso. Novatec, 2014

LIMA, A.S. UML 2.5 Do Requisito a Solução. São Paulo: Erica, 2014

PRESSAMAN, R.S. Engenharia de Software: Uma Abordagem Profissional. São Paulo: Bookman, 2011

RIOS,E. MOREIRA, TRAYAHÚ. Teste de Software. Alta Books: Rio de Janeiro, 2013 – 03 Edição

SILVA, R e P. Como modelar com UML2. Florianópolis. Visual Books, 2009

THE ADOPTION OF FREE CASE TOOLS TO UML IN A NIBESS COMPANY: A CASE STUDY

Abstract: *With the increasing publishing of books and articles on software development, the professionals in this area have achieved more success in their projects due to a greater sharing of experiences. The CASE (Computer Aided Software Engineering) tools at the disposal of these professionals are gaining more supporters in the software development area, and in several communities working on open source projects. This article aims at describing and illustrating some tools without the cost of licenses for development of UML (Unified Modeling Language) diagrams, rather than drawing tools like Microsoft Visio. By adopting these tools, it aims at evaluating the implications for the software development process and the reasons for not adopting these tools. When adopted, it´s important to ensure the application of their proper use or whether they are only for drawing. Finally, it describes the benefits of using such tools. The method adopted to evaluate these tools will be research and the use of tools in an exploratory way in a test environment in order to describe a set of results and features. Based on these results, it leads to the conclusion whether the free case tools may or may not help in any way to improve the process and the quality of systems development.*

Keywords: *Free case tools. UML. Software engineering.*