# 1º Encontro Regional de Engenharia de Software do Centro-Sul do Ceará

eresc

ENCONTRO REGIONAL DE ENGENHARIA DE SOFTWARE

CENTRO-SUL

INSTITUTO FEDERAL

Ceará

Campus Cedro


# ANAIS DO I ENCONTRO REGIONAL DE ENGENHARIA DE SOFTWARE (ERESC)

ISBN: XXX-XX-XXXXX-XX-X

1º EDIÇÃO

**ORGANIZAÇÃO**

PEDRO LUIS SARAIVA BARBOSA

**REALIZAÇÃO**


CEDRO-CE
2018

Dados Internacionais de Catalogação na Publicação

Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo bibliotecário Robson Souza (CRB3/1438)

---

E56a Encontro Regional de Engenharia de Software do Centro-Sul do Ceará (1. : 2018 : Cedro, CE)

Anais [recurso eletrônico] / I Encontro Regional de Engenharia de Software do Centro Sul do Ceará, 21 de maio de 2018. – Cedro, CE: IFCE, 2018.

154 p. : il. color.

Disponível em: <xxxxxxxxxxxxxxx>.

1. Engenharia de Software – Eventos Científicos. I. Título.

CDD 005.1

# PREFÁCIO

No ano de 2018, o professor da disciplina de Engenharia de Software do Instituto Federal de Educação, Ciência e Tecnologia do Ceará – *Campus* Cedro, decidiu utilizar uma metodologia de ensino baseada em projetos, e passou a ministrar os conteúdos em sala de aula de uma forma que fosse possível transferir aquele conhecimento para a organização de um evento acadêmico. A proposta foi repassada aos alunos e todos aceitaram e fizeram com que o evento acontecesse.

O I Encontro Regional de Engenharia de Software do Centro Sul do Ceará, é o primeiro evento da região com características específicas para a área de Engenharia de Software, o evento além das atrações, como: palestras, minicursos, workshops, atrações culturais, teve sua vertente científica. Foram recebidos 38 resumos expandidos, dos quais foram aceitos 20. A edição de 2018, ocorreu no dia 21 de maio e teve duração de 8 horas, contou com um público de 317 pessoas, que representam alunos de cursos superiores e de escolas de ensino médio da região e do próprio *Campus*.

Os resumos expandidos dessa edição foram divididos em 07 (sete) áreas. Assim, nesta versão dos Anais do I ERESC, o leitor encontrará os resumos expandidos selecionados minunciosamente pelo Comissão Científica.

Por último, agradecemos a todos que se esforçaram para que o evento ocorresse na região. O nosso muito obrigado aos palestrantes, aos ministrantes dos minicursos e *workshops,* comissão científica e comissão organizadora. O trabalho voluntário realizado por vocês foi muito importante para o acontecimento e sucesso do I ERESC. Desejamos que ocorra outras versões do evento.

Até a próxima!

Comissão Organizadora

I ERESC 2018

# COMISSÃO ORGANIZADORA

## Coordenação Geral

Prof. M.Sc. Pedro Luis Saraiva Barbosa

## Comissão Científica

Coordenação: Prof. M.Sc. Pedro Luis Saraiva Barbosa

Integrantes:
Prof. Esp. Adriano Lima Cândido
Prof. Me. Anderson Passos de Aragão
Prof. Esp. Ednael Macedo Félix
Prof. Esp. Elias Paulino Medeiros
Prof. Me. Evandro Nogueira Oliveira
Prof. Dr. Francisco José de Lima
Esp. José Ailton Batista da Silva
Prof. Esp. Lucas Ferreira Mendes
Prof. Me. Luís Gustavo Coutinho do Rêgo
Prof. Esp. Lyrane Teixeira Brito Bezerra
Prof. Me. Marco André Machado
Prof. Me. Pedro Luis Saraiva Barbosa
Esp. Rangel Henrique Félix
Prof. Me. Renato William Rodrigues de Souza
Prof. Me. Rodrigo Gonçalves Menezes
Prof. Esp. Saulo de Lima Bezerra
Prof. Esp. Tulio Vidal Rolim
Prof. Me. Wellington Feitoza Gonçalves

## Comissão Organizadora

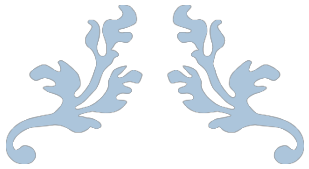
Coordenação: Prof. M.Sc. Pedro Luis Saraiva Barbosa

Discentes

Adalmária Diniz Ferreira
Alisson Rosa da Silva
Ana Valéria Bezerra
César Pinheiro Miranda
Eudes Germano Gonçalves
Fernanda Ferreira do Nascimento

Francisca Adriana Monteiro de Sousa
Francisco Soares da Silva Junior
Joeldo Olinda da Silva
José Danilo Torres Lima
Kamila Kelly Cardoso de Lima
Kevyn Bezerra Ladisla
Manoel Victor Cavalcante Inácio
Nayanne Carvalho Uchôa
Rafaella Alves de Sousa
Romero Bezerra Carvalho de Oliveira
Tainara Rocha Ferreira
Thomás Alves Cipriano de Oliveira

## Sumário



# APLICAÇÕES DA ENGENHARIA DE SOFTWARE NA INDÚSTRIA

# A INFLUÊNCIA DO SOFTWARE LIVRE NAS INSTITUIÇÕES PÚBLICAS

Antonio Airton Da Silva Neto,
Graduando em Bacharel no curso de Sistemas de Informação. IFCE.
airtonsilvan89@gmail.com.

Francisco Juvani Pereira da Silva,
Graduando em Bacharel no curso de Sistemas de Informação. IFCE.
juvaniifceiguatu@gmail.com .

Paulo Mailson Vieira da Mota,
Graduando em Bacharel no curso de Sistemas de Informação. IFCE.
pmailsonmota@gmail.com.

Ednael Macedo Felix,
Professor do curso de Bacharelado em Sistemas de Informação. IFCE.
ednael.macedo@ifce.edu.br.

## INTRODUÇÃO

Este artigo apresenta um estudo sobre as influências proporcionadas pelo uso do software livre nas organizações públicas. O intento do trabalho aqui apresentado vai além de simplesmente saber como o uso de softwares livres pode influenciar a administração dos setores das organizações públicas. Busca-se um aprofundamento, como é possível perceber no objetivo apresentado abaixo. A pesquisa foi realizada em duas organizações públicas: uma Instituição Federal de Ensino Técnico e Superior, e uma Autarquia Municipal de Serviços de Água e Esgoto. O público objeto da pesquisa, se constituiu de servidores/usuários que trabalham com software livre lotados nos setores de Tecnologia da Informação (TI) e Administrativo e Planejamento. Segundo Kliksberg, (1997) a intensidade e a pressão por resultados as quais esta submetida as organizações públicas torna a situação destas, cada vez mais complexas, tanto em aspectos relacionados a economia, o social e sobre tudo a tecnologia.

Segundo Silveira (2004), o software livre teve inicio em 1983 com o projeto GNU (*GNUs Not Unix*), como pesquisador *Richard Stallman* do *Intituto de Inteligência Artificial de Massachusets* Estados Unidos. Software livre não quer dizer software grátis, mais sim que há disponibilidade de seu código e consequentemente da possibilidade de poder modificar,

melhorar e redistribuir este código. Para o autor (id, 2004), os softwares têm contribuído muito com as organizações públicas, a ponto de serem amplamente utilizados nas mesmas para as mais diversas atribuições.

O software Livre oferece a possibilidade de modificação e adaptação a suas atividades, sem que seja necessária a comunicação ao seu desenvolvedor inicial. Além de possuir ferramentas prontas e muito completas para os diversos ambientes acadêmicos, administrativos, entre outros. Sem falar no custo benefício obtido pelas instituições, pois, não precisam usar de ferramentas proprietárias que necessitam de licenças pagas para o seu uso.

O objetivo deste trabalho é analisar a influência do software livre nas organizações públicas, de modo a identificar pontos positivos e negativos destes nos diferentes setores dessas organizações que se colocam frente a constante demanda por melhoria do ambiente.

É salutar considerar esta perspectiva, uma vez que a globalização, em constante avanço ao longo do tempo, impacta diretamente nas tecnologias da informação e possibilita mudanças na administração publica. Diante deste cenário, podemos destacar reformas e avanços nas ultimas décadas.

O fato é que os Softwares livres são ferramentas utilizadas em vários ambientes públicos. Segundo o Guia Livre (2005), a importância dessas distribuições e a comodidades do uso das mesmas são consideráveis, pois possibilitam uma adaptação em seu código-fonte, a fim de torná-lo mais eficiente. Contudo, podemos destacar as inúmeras possibilidades de ferramentas existentes na forma de softwares proprietários. Assim, pensando na redução de gastos os setores públicos podem optam pelo uso desta modalidade de software.

Não podemos deixar de ressaltar o precursor dos softwares livres, que foi o sistema operacional Linux. Este sistema auxilia diversas instituições públicas, possibilitando a utilização de diversas aplicações, sem gerar qualquer custo aos usuários, com grande segurança de dados. A possibilidade de adotar estes sistemas, que se adaptam ao negócio, ou ainda ser customizado na linguagem gerencial, quando aliada ao menor custo em relação aos sistemas proprietários, de fato seduz um novo numero cada vez maior de empresas (MARQUES. 2004).

Levando em consideração toda a questão do gerenciamento dos recursos recebidos pelas organizações públicas, e como esses devem ser utilizados sem desperdícios, podemos destacar o caso do *Total Value of Ownership* – TVO, que será aproveitado como guia para busca das informações necessárias, para inferir sobre a viabilidade do investimento na troca de software, pois este tem três componentes, ou seja, tanto a metodologia de custo/benefício para avaliar o incremento de valor criado pelo investimento em TI; os processos de gerenciamento robustos para integrá-la aos procedimentos da organização; bem como o julgamento maduro dos executivos para decidir entre opções com segurança (DEMPSEY *et al.*, 1998).

Dempsey (1998) ressalta ainda que se deve sempre avaliar a questão custo benefícios quando se deseja fazer um investimento nos setores de TI, para que essas decisões não afetem o gerenciamento dos recursos da Instituição.

Entre as características marcantes do software livre, esta o incentivo ao pensar, e com isso a possibilidade que as informações e tecnologias cheguem aos mais diversos setores da sociedade, assim como já citado por Michelazzo (2003).

Licenças como GPL (*General Public License*), permite copiar, alterar a obra original, sem nenhuma alteração prévia do autor. Esse registro foi o que equiparou o SL ao *software* proprietário, deixando o próprio governo optar por produzir *software* ou adquirir SL por meio de licitações (SILVA, 2014).

Alguns destes critérios foram citados por Roberto Hexsel (2002) em seu artigo "O que é software livre", segundo ele, os critérios são:

> "(a) a redistribuição deve ser livre; (b) o código fonte deve ser incluído e deve poder ser redistribuído; (c) trabalhos derivados devem poder ser redistribuídos sob a mesma licença do original; (d) pode haver restrições quanto à redistribuição do código fonte, se o original foi modificado; (e) a licença não pode discriminar contra qualquer pessoa ou grupo de pessoas, nem quanto a formas de utilização do software; (f) os direitos outorgados não podem depender da distribuição onde o software se encontra; e (g) a licença não pode 'contaminar' outro software. (HEXSEL, 2002 p. 10)."

Sendo assim, tendo em vista a vasta possibilidade de o software livre ser utilizado, bem como que oferecem as mesmas funcionalidades e desempenho que os softwares proprietários, com a questão do custo benefício sendo a mola central nesta perspectiva, são elementos que fazem Mello (2003) perceber nestes softwares, uma rara importância.

> [...] O software Livre tem um custo baixo e, toda via, é de alta qualidade tecnológica. Pois o trabalho conjunto de milhares de técnicos possui uma vantagem decisiva sobre pequenos grupos de técnicos trabalhando em um produto: tal produto evolui em uma velocidade enorme [...]. (PINHEIRO, 2003, p. 277).

Logo, o projeto software livre não possui o intuito financeiro, apesar de possui alguns pagos, mas na sua grande maioria são obtidos gratuitamente e as pessoas podem fazer contribuições para ajudar a manter a comunidade, afinal existem pessoas que empregam seu tempo neste projeto.

## METODOLOGIA

Para realização desta pesquisa utilizou-se do método descritivo, considerando que na pesquisa descritiva observa-se, registra-se e analisa-se os fenômenos sem que haja interferência do pesquisador (PRODANOVE; FREITAS, 2013). Como procedimento, utilizou-se a pesquisa de campo, tendo como objeto o uso de software livre por duas instituições públicas.

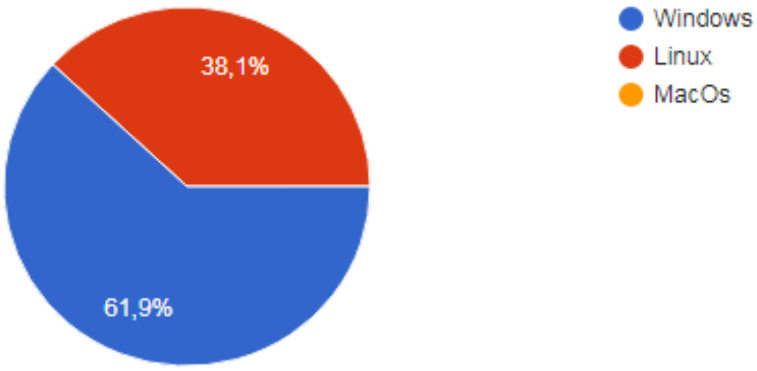
A coleta de dados se deu pela aplicação de questionário estruturado baseado no trabalho de Dalanhol e Silveira (2016). O questionário foi estruturado em cinco perguntas, aplicado a 42 servidores, com aplicação entre o mês o abril e maio de 2018.

Optou-se pela apresentação dos dados por meio de estatística descrita, uma vez que o trabalho aqui pleiteado trata-se de pesquisa quantitativa.

## RESULTADOS

As duas organizações objeto deste estudo, fazem o uso de Software Livre. Neste sentido foi indagado no questionário se os servidores conseguiam diferenciar Software Livre de Software Proprietário. Pode-se constatar que aproximadamente 86% dos pesquisados afirmaram que sim. Com tudo, 14% dos servidores não conseguem fazer essa diferenciação.

Gráfico 1 – Sistemas Operacionais em uso na instituição




Fonte: Dados da pesquisa (2018)

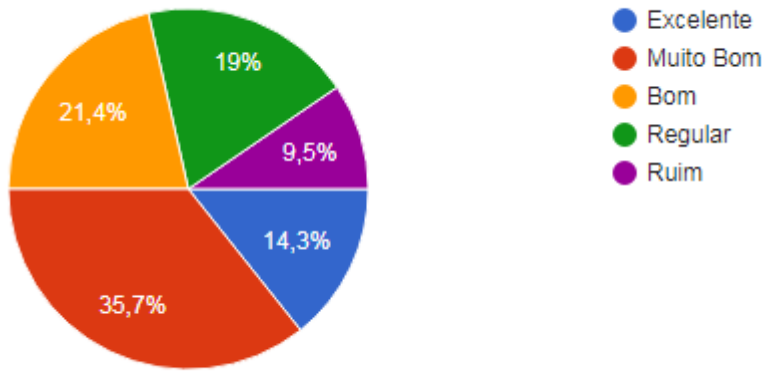
Assim como já constatada na pesquisa de Dalanhol e Silveira (2016), na qual aproximadamente 86% dos pesquisados por eles conseguiam fazer esta diferenciação, no trabalho aqui pleiteado, denotou-se que a grande maioria dos entrevistados também consegue diferenciar os softwares.

Pode-se constatar conforme indica o gráfico 1, que aproximadamente 62% dos servidores trabalham diariamente com sistema Windows, o que mostra ainda um predomínio de software proprietário nas instituições pesquisadas. É possível atribuir este índice ao fato de muitos sistemas utilizados nas instituições demandarem este sistema operacional para seu funcionamento.

A informação supracitada pode ser melhor esclarecida com o gráfico 2, o qual apresenta na perspectiva dos pesquisados o aproveitamento dos Software Livres em relação aos Software

Proprietário. Constatou-se que 50% dos pesquisados julgam ser o Software livre excelente ou muito bom.

Gráfico 2 - Aproveitamento do uso de Software Livre com relação ao software proprietário




Fonte: Dados da pesquisa (2018)

Essa constatação reforça a perspectiva de que além de apresentarem custos mais acessíveis, os softwares livres também possuem boa aceitação por parte dos usuários, indicando que investimentos nos mesmos, bem como na capacitação para seu uso pode ser um excelente mecanismo de redução de custos e aumento da eficiência nos processos burocráticos das instituições públicas.

Conforme destacado, vimos à importância de se realizar investimentos na utilização de softwares livres nas instituições públicas, principalmente na área de capacitação dos servidores. Conseguimos identificar essa necessidade no questionário, onde foi perguntando se os servidores já teriam participado de treinamentos sobre o uso de software livre e obtivemos o resultado de que aproximadamente 57% nunca tiveram esse tipo de treinamento e 47% já tiveram treinamentos relacionados ao uso de software livre. Com isso destacamos a falta de investimentos nessa tecnologia, não só nos servidores, como também nas equipes que compõem o setor de TI das instituições utilizadas como objeto de estudo.

Outro ponto de bastante relevância é a relação entre custos e benefícios no uso dessa tecnologia. Sabemos das Licenças, como GPL (General Public License), que permitem copiar e alterar a obra original, o que possibilita a utilização de softwares já prontos, ou customizarmos, para atender a execução de um determinado processo de uma área da instituição ou até mesmo os processos de toda instituição. Baseado nisso, indagamos em nossa pesquisa o custo beneficio que o software livre oferece a instituição, e tivemos um resultado expressivo: aproximadamente 84% acreditam que o Software Livre oferece um ótimo custo benefício.

## CONCLUSÃO

Com a conclusão deste trabalho, fica evidente a importância do Software Livre nas organizações públicas, uma vez que estes oferecem diversas ferramentas para auxiliar os servidores a desempenharem suas funções, de forma mais rápida e eficaz, sem a necessidade de um gasto excedente em softwares pagos, além da capacidade destes, de adaptarem-se a outra função além daquela para a qual foram criados.

Apesar das organizações ainda promoverem de forma absoluta o uso desta tecnologia, o incentivo do governo e as possibilidades proporcionadas por estes *softwares* tem contribuído muito para o crescimento destas ferramentas. Alguns projetos têm comprovado os valores e a qualidade dos *Softwares* Livres e os valores que mesmos tem empregados nas comunidades onde são implantados.

Portanto, o trabalho que empreendemos possibilitou à conclusão de que há uma importância significativa do objeto pesquisado, para as instituições, e também desafios que as instituições públicas enfrentam na utilização dos Softwares Livres, seja pela falta de conhecimento dos benefícios, seja pela ausência de treinamentos e informações tanto da sua equipe de TI e servidores das instituições que realizam os processos institucionais.

## REFERENCIAS


DEMPSEY,J., DVORAK, R. E., HOLEN, E., MARK, D., MEEHAN, W. F. 1998.A Hard and Soft Look at IT Investments**.** McKinsey Quarterly 1: 126-137.
Disponível em:<www.mckinseyquarterly.com>

GUIA LIVRE. Referência de Migração para Software Livre do Governo Federal / Organizado por Grupo de Trabalho Migração para Software Livre. Brasília, 2005. Disponível em : <http://www.softwarelivre.gov.br/publicacoes/guia-livre-referencia-de-migracao-para-software-livre>

HEXSEL, A. Roberto. O que é Software Livre?. In: Departamento de Informática da Universidade Federal do Paraná. Disponível em:
<http://www.softwarelivre.gov.br/SwLivre/>

HEXSEL, A. Roberto. O que é Software Livre?. In: Departamento de Informática da Universidade Federal do Paraná. Disponível em: <http://www.inf.ufpr.br/info/techrep/RT_DINF004_2002.pdf > .

KLIKSBERG, Bernardo. O Desafio da Exclusão – Para uma Gestão Social Eficiente. São Paulo: Edições FUNDAP, 1997

MARQUES, M. (2004) O Pinguim Avança: cresce o número de empresas privadas que adotam o Linux, e o governo federal resolve comprar briga com a Microsoft®. Carta Capital. São Paulo: Confiança, nº 282, Ed. Março 2004.

MELLO, Ricardo Andere. software livre e inclusão digital. Inpud.SILVEIRA, Sergio Amadeu e CASSINO, João. São Paulo: Conrad Editora do Brasil, 2003.

MICHELAZZO, Paulino. software livre e inclusão digital. Inpud.SILVEIRA, Sergio Amadeu e CASSINO, João. São Paulo: Conrad Editora do Brasil, 2003.

PINHEIRO, Walter. software livre e inclusão digital. Inpud.SILVEIRA, Sergio Amadeu e CASSINO, João. São Paulo: Conrad Editora do Brasil, 2003.

PRODANOV, Cleber. Cristiano; FREITAS, Ernaní. Cesar. Metodologia do Trabalho Cientifico: Métodos e Técnicas da pesquisa e do trabalho Acadêmico. 2 ed – Novo Hamburgo – Rio Grande do Sul - Brasil: Editora Feevale, 2013.

ROBREDO, Jaime. Da ciência da informação revisada aos sistemas humanos de informação. Brasilia: Thesaurus,2003.

SILVA, G. F. (2014) **Estudo de Caso do Projeto Expresso:** A Implementação de Políticas Públicas Voltadas ao *Software* Livre. Curso de Especialização em Gestão Pública – UNISERPRO (EaD). Porto Alegre, 2014. Disponível em: <http://www.lume.ufrgs.br/handle/10183/127295>

SILVEIRA, Sergio Amadeu. In: ENTREVISTA SOBRE PORQUE O SOFTWARE LIVRE NÃO PARA DE CRESCER. Disponível em: <http://samadeu.blogspot.com/2006/08/entrevista-sobre-porque-o-software.html>.

# ANÁLISE DO FLUXO DE ATENDIMENTO NAS UNIDADES BÁSICAS DE SAÚDE PARA DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DE DADOS

Kamila Kelly Cardoso de Lima,
Graduanda em Sistemas de Informação. IFCE. kamilalima1321@gmail.com.

José Danilo Torres de Lima,
Graduando em Sistemas de Informação. IFCE. danilotl21@gmail.com.

Vinicius Marques Silva,
Graduando em Sistemas de Informação. IFCE. viniciusmarques2602@hotmail.com.

Ednael Macedo Felix,
Mestrado em Administração de Empresas. Unifor; Professor EBTT. IFCE.
ednael.macedo@ifce.edu.br.

## INTRODUÇÃO

As Unidade Básica de Saúde (UBS), são definidas como um conjunto de condutas da saúde no contexto individual e coletivo, que tem por finalidade promover e proteger a saúde, prevenir possíveis agravos, realizar diagnóstico e tratamento, a reabilitação, que visa conservar o bem-estar, desenvolvendo uma atenção completa e proporcionando autonomia da coletividade dos usuários (RASIA, 2013).

A maioria das UBS's funcionam com pouca ou nenhuma tecnologia informatizada nas rotinas de trabalho, apesar de já existirem sistemas informatizados que possibilitam o armazenamento e manipulação dos dados. Na maioria dos postos de saúde há situações de que o preenchimento do prontuário do paciente ainda é feito manualmente.

O sistema de informação em enfermagem deve ser capaz de auxiliar o enfermeiro durante a avaliação, planejamento e execução dos cuidados. Os sistemas de informação hospitalar e de informação em enfermagem estão intrinsecamente relacionados, pois possuem objetivos em comum, como por exemplo, alcançar uma maior produtividade, tornar as informações mais disponíveis e facilitar o processo de comunicação (AMRIN, 1995, apud LIMA et al., 2011)

O Departamento de Atenção Básica (DAB) do Ministério da Saúde, reestruturou os Sistemas de Informação da Atenção Básica (SIAB), a fim de facilitar o trabalho e a gestão da atenção básica, e criou o Sistema de Informação em Saúde da Atenção Básica (SISAB), e o e-

SUS Atenção básica (e-SUS-AB), ambos com objetivos de melhorias na atenção à saúde (ANDRADE, 2014).

Os Sistemas informatizados são um conjunto de componentes inter-relacionados usados para coletar, processar, e porventura retornar informações que podem vir a ser usadas para inúmeros fins, coibindo possíveis perdas de informação, de recurso e tempodos usuários e beneficiários. Os mesmos normalmente sendo eficientes e confiáveis, garantem qualidade da informação, promovendo assim uma agilidade, segurança, redução de recursos e informatização, dessa forma, beneficiando os usuários.

O uso de sistemas de gerenciamento de dados estabeleceria uma nova forma de realizar um atendimento mais rápido e seguro para as unidades. Assim, a proposta de se estudar o fluxo de atendimento, vem como uma solução de melhorar a qualidade desse processo nessas unidades. Além disso, facilitar o armazenamento das informações e manter o ciclo de vida dos dados em total segurança.

Dessa forma o presente estudo tem por objetivo apresentar uma solução da utilização de um Sistemas de Gerenciamento de Banco de Dados (SGBD) para agilizar o fluxo de atendimento nas UBS's.

## METODOLOGIA

Visando alcançar o objetivo do presente projeto, foi necessário levantar informações sobre o funcionamento e fluxo de atendimento das Unidades Básicas de Saúde. A pesquisa aconteceu no município de Cedro no Ceará no período de março e abril de 2018.

Para a realização da coleta de dados, utilizou-se um questionário online desenvolvido no Google forms, contendo 6 perguntas sobre o atendimento nas unidades para pessoas que se beneficiam desse serviço, a fim de analisar quais as dificuldades de realizar um atendimento com rapidez e sem desperdícios de recursos materiais.

Os participantes da pesquisa tiveram acesso as informações por meio do compartilhamento do link do questionário nas redes sociais dos autores, durante o período de duas semanas.

O questionário contribuiu para um melhor entendimento sobre os problemas estudados, facilitando na construção de um meio para agilizar de forma rápida e segura o atendimento em tais unidades.

A pesquisa de Campo foi escolhida por ter objetivo de conseguir informações acerca de um determinado problema, para qual se procura solucionar, com o fato de comprovar e descobrir relações entre os mesmos.

> Consiste na observação de fatos e fenômenos tal como ocorrem espontaneamente, na coleta de dados a eles referentes e no registro de

> variáveis que se presume relevantes, para analisá-los. A pesquisa de campo propriamente dita "não deve ser confundida com a simples coleta de dados (este último corresponde à segunda fase de qualquer pesquisa); [...] (MARCONI; LAKATOS, 2010, p.169, apud TRUJILLO, 1982, p. 229).

A aplicação foi feita por meio de um formulário online, onde foi compartilhado entre diversos usuários que utilizam os serviços dessas unidades básicas de saúde. A duração da aplicação foi de 2 semanas, durante o mês de abril de 2018, a um público de 164 pessoas.

A utilização de formulários para coletar dados é considerado muito importante para a realização da pesquisa, pois ao decorrer do processo de coleta de dados que o investigador consegue as informações fundamentais para o andamento do estudo. Com relação aos questionários, pode-se afirmar que, o sucesso da pesquisa depende, em grande parte, da maneira como o pesquisador faz a coleta dos dados e, para coletar corretamente as informações necessárias para a realização de sua pesquisa, é desafio do pesquisador escolher corretamente os instrumentos de coleta de dados que atendam aos seus objetivos e que estejam de acordo com a técnica utilizada. (OLIVEIRA et al., 2016).

As respostas alcançadas no formulário durante as duas semanas de aplicação, foram representadas em gráficos que nos auxiliou na demonstração e compreensão dos dados. Percebeu-se que essas informações nos forneceu uma enorme importância para alcançarmos o nosso objetivo, proposto no presente artigo.

Mantemos o foco da pesquisa na UBS, visando a carência de informatização dessa unidade e os motivos da morosidade do atendimento, em busca de uma forma para melhorar e agilizar o fluxo. Levando em consideração o estudo das pesquisas encontradas, o uso de um sistema de gerenciamento para otimização das unidades se mostra uma ótima saída.

## RESULTADOS E DISCUSSÕES

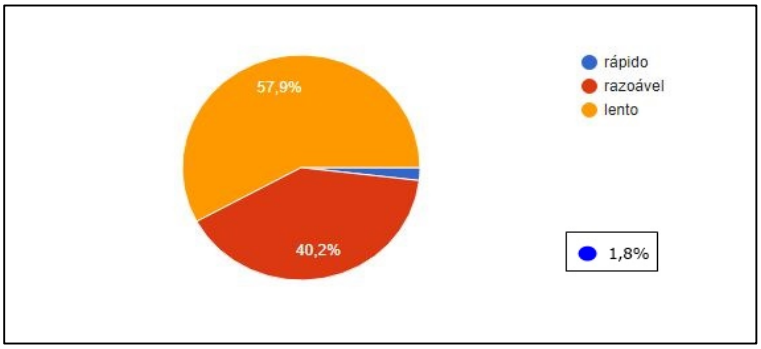
O questionário elaborado foi uma ferramenta importante para a coleta dos dados e nos proporcionou estabelecer uma solução viável para o problema do fluxo de atendimento das unidades. Teve-se dados de suma importância durante o processo de aplicação, onde as respostas foram significativas para a conclusão do presente trabalho. Vale ressaltar que a pesquisa contou com 164 participantes que responderam as questões contidas no questionário.

O questionário apresentou 6 perguntas para preenchimento com alternativas que facilitasse a compreensão no momento da analisar os dados. Essas perguntas foram elaboradas de modo que essas pessoas não tivessem dificuldade de entender e que marcasse a opção pretendida.

Pode-se perceber, observando no gráfico 1, que aproximadamente 60% dos pesquisados indicaram que os atuais procedimentos nas unidades básica de saúde é lento. Vendo este índice, é notório que é necessário que se faça uma modificação quanto a maneira como os processos

de atendimento ao público é realizada, uma vez que a maioria dos usuários caracterizam estas unidades como lentas em seu processo de atendimento.

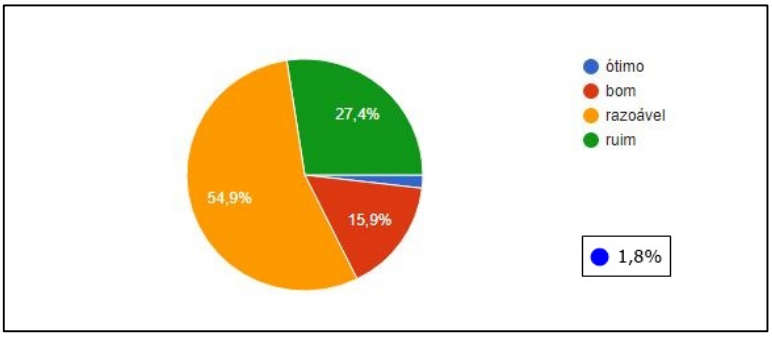
Gráfico 1 – Atendimento nas unidades básicas de saúde




Fonte: Dados da pesquisa (2018)

Atualmente, muitas pessoas consideram que o armazenamento de dados em papel ainda é seguro, possivelmente por insegurança de fornecer seus dados em um sistema e garantir que estejam seguros. Quando perguntado sobre a maneira com que os dados são armazenados atualmente nas unidades básicas de saúde, notou-se, como está apresentado no gráfico 2, que 50% dos participantes que preencheram o questionário respondeu que o armazenamento de dados é razoável.

Gráfico 2 – Segurança no modo de armazenamento de dados

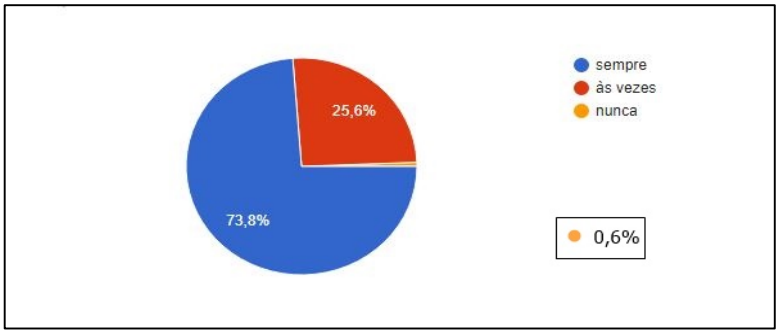



Fonte: Dados da pesquisa (2018)

Como pode-se observar no gráfico 3, cerca de 74% das pessoas afirmaram que uma possível utilização de um sistema computacional para armazenar os dados dos pacientes, seria uma ótima solução para o armazenamento de dados e automatização do processo de atendimento. Essa porcentagem relevante oferece uma sugestão de que a utilização de um sistema pode favorecer uma possível solução para o problema e como consequência, agilizar o atendimento.

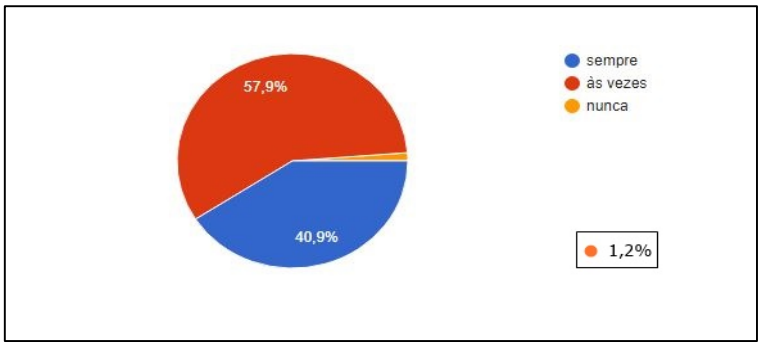
A pergunta número quatro foi a questão chave do nosso questionário, pois o nosso objetivo é agilizar o atendimento com o uso de um sistema, substituindo o uso de materiais de papel usados como fichas. Por meio desta constatou-se, como representado no gráfico 4, que 98,8% dos pesquisados atribuem a morosidade no atendimento, exatamente ao fato de as UBS ainda fazerem o cadastro em fichas de papel.

Gráfico 3 – Utilização de um sistema computacional para melhorar o atendimento




Fonte: Dados da pesquisa (2018)

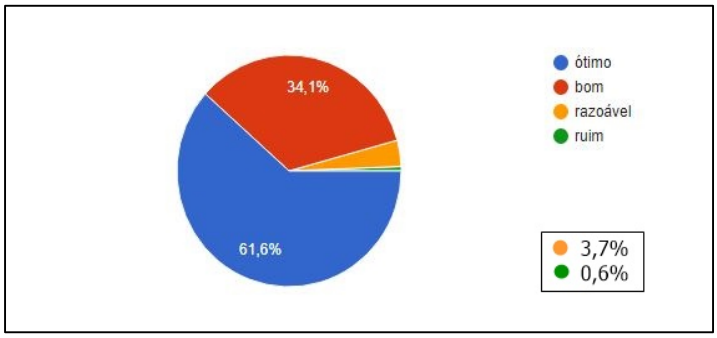
Gráfico 4 – Fichas que podem contribuir para demora no atendimento




Fonte: Dados da pesquisa (2018)

Pode-se notar, por meio do gráfico 5, que aproximadamente 62% dos pesquisados afirmaram que um cadastro pronto do paciente agilizaria o atendimento nessas unidades. O cadastro feito e completo do paciente, possivelmente, ajudaria no processo de atendimento no momento da captura de informações. A determinada sugestão surge como uma solução para acabar com a lentidão nessas unidades.

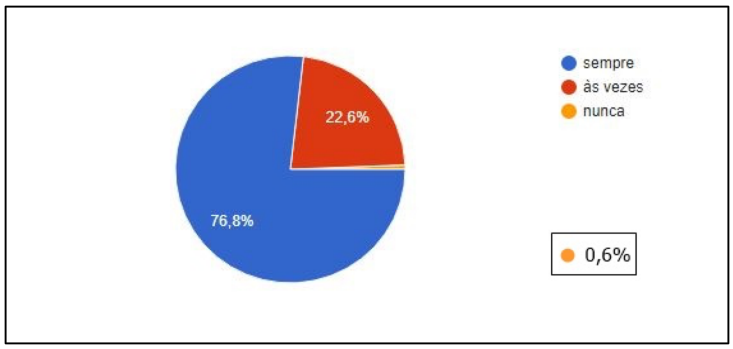
Gráfico 5 – Cadastro feito no processo de atendimento




Fonte: Dados da pesquisa (2018)

O sistema possibilitaria o preenchimento do prontuário e garantiria o retorno das informações do paciente, sem a necessidade de procurar em fichas de papel, que causam a demora durante o atendimento ao paciente. Como pode-se observar no gráfico 6, cerca de 80% dos pesquisados, declararam que facilitaria o processo de trabalho no retorno de informações e prontuário do paciente.

Gráfico 6 – Facilidade no processo de trabalho utilizando o sistema




Fonte: Dados da pesquisa (2018)

Com base nos dados dos gráficos apresentados anteriormente, os resultados foram satisfatórios. Por meio deles, conseguiu-se afirmar que o motivo da demora no atendimento nas unidades foram os esperados. Com isso, a nossa proposta mantém-se válida pelo fato de que a utilização de um suposto sistema facilitaria o trabalho das atendentes dessas unidades, como consequência, garantia a rapidez no processo de atendimento e segurança dos dados armazenados.

## CONCLUSÕES

Esse estudo teve como objetivo analisar o fluxo de atendimento nas Unidades Básicas de Saúde do município de Cedro e apresentar uma solução, por meio da utilização de um SGBD para agilizar o atendimento, assim como, otimizar as rotinas de trabalho.

Foi levado em consideração a satisfação e insatisfação dos pacientes em relação ao atendimento e o modo em os dados são armazenados. A aplicação do questionário se deu visando a coleta de dados de pessoas que se beneficiam das UBS's. Por conta disso, tivemos os dados apurados para propor a utilização de um sistema nessas unidades.

As informações extraídas do questionário nos garantiram que essas unidades necessitam de uma tecnologia para informatizar e agilizar o processo de atendimento. Conforme o nosso objetivo, a sugestão de um sistema de controle de dados para melhorar o atendimento seria sim uma solução para esse problema.

## REFERÊNCIAS

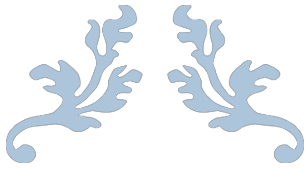

ANDRADE, R.C. Equipamentos de Informática nas Unidades de Atenção Básica do Brasil: Análise Baseada no Programa Nacional de Melhoria do Acesso e da Qualidade - PMAQ. Brasília-DF, Julho de 2014.

LIMA, Dayane et al. Sistema de informação em saúde: conceções e perspectivas dos enfermeiros sobre o prontuário eletrônico do paciente. **Revista de Enfermagem Referência**, [s.l.], v. , n. 5, p.113-119, 28 dez. 2011.

RASIA, I.B. Inovações na Atenção Primária à Saúde: Estudo de Casos de 5 UBS de Pelotas/RS. **Gestão e Planejamento**. Salvador, Janeiro 2013.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos de Metodologia Científica.** 7. ed. São Paulo: Atlas S.a., 2010. 297 p.

OLIVEIRA, José Clovis Pereira de et al. O QUESTIONÁRIO, O FORMULÁRIO E A ENTREVISTA COMO INSTRUMENTOS DE COLETA DE DADOS: VANTAGENS E DESVANTAGENS DO SEU USO NA PESQUISA DE CAMPO EM CIÊNCIAS HUMANAS. In: III CONEDU CONGRESSO NACIONAL DE EDUCAÇÃO, 17., 2016, Rio Grande do Norte. **Congresso Nacional de Educação.** Rio Grande do Norte: Realize, 2016. p. 1 - 13.



# EDUCAÇÃO E ENGENHARIA DE SOFTWARE

# ADVERSIDADES E INTERVENÇÕES NO ENSINO DA ENGENHARIA DE SOFTWARE: UMA REVISÃO SISTEMÁTICA NA LITERATURA

Adalmária Diniz Ferreira,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro

Túlio Vidal Rolim,
Discente de Mestrado em Ciência da Computação da Universidade Federal do Ceará.

Rangel Henrique Félix,
Graduado em Redes de Computadores pelo Instituto Federal de Educação, Ciência e Tecnologia do Ceará.

José Ailton Batista Da Silva,
Graduado de Tecnologia em Mecatrônica Industrial pelo Instituto Federal de Educação, Ciência e Tecnologia do Ceará.

Pedro Luís Saraiva Barbosa,
Docente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro. pedro.barbosa@ifce.edu.br.

## INTRODUÇÃO

Utilizar métodos e diversas opções de ferramentas dentro de um processo planejado para desenvolvimento de um software de qualidade é considerado Engenharia de Software (ES) (PRESSMAN, 2011). Já para Sommerville (2011) a ES é o ramo da engenharia envolvido nos aspectos da produção de softwares cujo o objetivo é apoiar o desenvolvimento profissional de software acima da programação individual, suas técnicas apoiam especificação, projeto e evolução de programas.

Devido ao hábito de desenvolverem suas atividades acadêmicas de forma descoordenada, sem registros e próximo ao vencimento dos prazos de entrega, os alunos possuem uma dificuldade inerente de adequar os conceitos da disciplina de ES a sua realidade. Metodologias que simulem a realidade do trabalho na área de estudo do discente podem ajudá-lo a mudar esses hábitos, aumentar seu interesse pela disciplina e ao mesmo tempo preparar para o mercado específico ao seu curso.

A aprendizagem tem sido transformada com o surgimento de novas tecnologias, onde modelos projetados apenas para mera acumulação de conhecimento tornaram-se obsoletos, visto que, com esta mudança a proposta é aprender praticando através do experimento, assim trazendo mudanças no processo educacional e impactando a forma como os alunos aprendem e vêem as aulas (MONSALVE; LEITE, 2015).

Baker, Navarro e Van Der Hoek (2005), corroboram com esse cenário ressaltando que a forma como a disciplina de ES é abordada em sala de aula não aborda a realidade envolvida

no processo de desenvolvimento e gestão dos projetos, pois o professor tentará da melhor maneira explicar e expor como funcionam as ações envolvidas nesse processo, porém os estudantes carecem de uma oportunidade de participar de um processo de desenvolvimento de software por inteiro.

O ensino em ES é em grande parte teórico, concentrado principalmente em aulas expositivas e alguns poucos trabalhos práticos com a intenção de usar a teoria aprendida em sala de aula. Porém, realizar apenas esses trabalhos teóricos geralmente não fornece um conhecimento específico ao aluno, o que por muitas vezes, acaba gerando uma situação desestimulante ou desinteressante ao discente.

Buscar novos métodos de ensino para uma melhor abordagem e aprendizado da ES, mostrando onde esta área está presente no cotidiano do indivíduo, torna-se a chave não só para que se compreenda melhor a matéria abordada em sala, como também desconstruir hábitos nocivos engendrados no meio acadêmicos que podem vir a atrapalhar a vida profissional do aluno. Portanto, objetivou-se com este estudo realizar uma revisão sistemática de literatura como meio de obter uma maior compreensão quanto aos principais problemas no processo bem como os métodos e abordagens de ensino em ES apresentados na literatura.

## METODOLOGIA

O procedimento técnico utilizado neste trabalho baseia-se em uma Revisão Sistemática da Literatura (RSL) que teve como objetivo realizar um levantamento bibliográfico a respeito de trabalhos abordando problemas, práticas e metodologias de ensino da disciplina de ES, de forma a descobrir quais as estratégias, métodos e técnicas são adotados nestes trabalhos.

Uma revisão sistemática sintetiza o trabalho existente de uma maneira justa e considerada justa. Por exemplo, as revisões sistemáticas devem ser realizadas de acordo com uma estratégia de pesquisa predefinida. A estratégia de busca deve permitir avaliar a completude da pesquisa (KITCHENHAM; CHARTERS, 2007).

O protocolo de revisão sistemática é um artefato metodológico que serve como elemento norteador no planejamento, execução e sumarização de uma revisão sistemática, especificando os métodos utilizados reduzindo a possibilidade de influência do pesquisador (KITCHENHAM, 2004).

## PROTOCOLO DE REVISÃO SISTEMÁTICA

### Descrição do Problema

O ensino tradicional dificulta com que o aluno assimile o aprendizado de disciplinas como ES devido às explanações a respeito do assunto serem abordadas de maneira descritiva e separadas do contexto da realidade dos mesmos. Metodologias que propõem uma estratégia de ensino mais interpretativa ou que imergem o discente dentro de cenários conectados a realidade do seu cotidiano ao mesmo tempo que se relacionam com a disciplina podem ser a chave para otimizar o aprendizado.

### 1) Questões de Pesquisa

*QP1. Quais são os problemas abordados na aprendizagem da ES pelos trabalhos?*

*QP2. Quais métodos e técnicas têm sido utilizados como estratégia de ensino na disciplina de ES?*

### 2) Processo de Busca utilizado para pesquisa de Fontes Primárias

As bases de dados utilizadas na coleta dos trabalhos acadêmicos foram: *Congresso da SBC (CSBC), Workshop sobre Educação em Computação (WEI), IEEE, BDBComp, SBGAMES, CSE, Australasian Conference on Computing Education, V Fórum de Educação em Engenharia de Software, ACM e Anais dos Workshops do Congresso Brasileiro de Informática na Educação (CBIE)*. As conferências e bases foram escolhidas por possuírem respaldo e notoriedade como veículos confiáveis e significativos no cenário de informática educativa.

**3) Critérios de Inclusão e Exclusão dos Estudos**

Os critérios de inclusão empregados na seleção dos estudos foram os seguintes: *i.a)* Ter sido escrito nos idiomas português ou inglês; *i.b)* Ter sido publicados nos últimos 10 anos; *i.c)* Os estudos devem apresentar a utilização de métodos e técnicas alternativas voltadas ao ensino da disciplina de ES. Já os critérios de exclusão são: *e.a)* Trabalhos inconclusos; *e.b)* Estudos que abordem outras disciplinas que não ES; *e.c)* Estudos que utilizavam discentes de outras modalidades de ensino como fundamental e médio.

**4) Critério de Qualidade dos Estudos**

O critério de qualidade define a condição necessária para considerar positivamente um estudo cogitado. Após a seleção dos estudos, foram somente considerados as estratégias de ensino aplicadas que tenham sido feitas de forma explícita, organizada e replicável, possuindo assim uma validade prática dirigidas a uma população amostra, o que garante uma maior integridade ao estudo.

**5) Extração dos Dados**

Os dados são extraídos da seguinte maneira: *a)* Os pesquisadores identificam os estudos primários; *b)* Os estudos são classificados e dispostos em uma Tabela de Revisão. *c)* Os artigos são averiguados pelos três pesquisadores. *d)* Após a averiguação, os artigos aprovados são aferidos de forma aprofundada, sendo coletadas informações como: *i)* Identificação do Estudo; *ii)* Veículo de Publicação; *iii)* Ano de publicação; *iv)* Autores e *v)* Título.

## RESULTADOS E DISCUSSÕES

Após a busca primária, 36 estudos foram catalogados, lidos e verificados conforme os critérios de inclusão e exclusão, gerando assim uma amostra de 10 estudos de acordo com os critérios definidos na (subseção 3) do protocolo. Os dados foram tabulados (ordenados numericamente por um identificador) em conjunto com os dados de extração de cada estudo apresentadas na seção *5) Extração dos Dados*. Na Tabela 1 estão exibidos os trabalhos analisados nesta RSL. A seguir, serão expostos os resultados obtidos para as questões *QP1* e *QP2* do estudo descritas na seção *1) Questões de Pesquisa* do protocolo.

*QP1 - Quais são os problemas abordados na aprendizagem da ES pelos trabalhos?*

Benitti e Molléri (2008) expõe através dos relatos de Navarro e Hoek (2002), Ohlsson e Johansson (1995), Silva, Leite e Breitman (2004) — que as aulas de processo e desenvolvimento de software juntamente com a gestão de projetos são apresentadas através de aulas teóricas juntamente com a resolução de pequenos projetos didáticos. Entretanto, o método por meio de projetos não fornece alguns aspectos importantes aos estudantes, tais como: (conhecimento do domínio, complexidade e tamanho), assim, em função do tempo e da natureza didática, os estudantes não conseguem atingir um nível de satisfação hábil e produtivo no que tange a aprendizagem e real compreensão do processo de desenvolvimento de software.

**Tabela 1** - Trabalhos Selecionados.

| ID | Veículo | Ano | Autores | Título |
|---|---|---|---|---|
| ID01 | SBGames | 2008 | BENITTI e MOLLÉRI | Utilização de um RPG no Ensino de Gerenciamento e Processo de Desenvolvimento de Software |
| ID02 | XVIII WEI | 2010 | DIGIAMPIETRI, KROPIWIEC e SILVA | O uso de jogos como fator motivacional em cursos de computação |
| ID03 | XX WEI | 2012 | DA SILVA et al. | Uma avaliação do emprego do jogo Modelando como apoio ao ensino |
| ID04 | BDBComp | 2012 | FARIAS et al. | iTest Learning: Um Jogo para o Ensino do Planejamento de Testes de Software |
| ID05 | SBGames | 2011 | KOHWALTER, CLUA e MURTA | SDM - An Educational Game for Software Engineering |
| ID06 | IEEE | 2015 | CHAVES et al. | Experimental Evaluation of a Serious Game for Teaching Software |
| ID07 | IEEE | 2015 | MONSALVE, DO PRADO LEITE e WERNECK | Transparently teaching in the context of game-based learning: the case of simulES-W |
| ID08 | IEEE | 2015 | LETRA, PAIVA e FLORES | Game Design Techniques for Software Engineering Management Education |
| ID09 | XXI WEI | 2013 | KUPSCH e RESENDE | SPIAL: Uma Ferramenta de apoio ao aprendizado de melhoria de processo de software |
| ID10 | CBIE | 2016 | SOUZA e FRANÇA | O Sucesso dos Jogos para Ensino de Disciplinas de Engenharia de Software sob a Ótica de uma Teoria Motivacional |

Digiampietri, Kropiwiec e Silva (2010) exibem com problemática o fator motivacional dos alunos de cursos como sistemas de informação descrevendo como as aulas teóricas acabam tornando-se maçantes e difíceis de correlacionar com a realidade do aluno e mercado de trabalho, assim como a dificuldade de linkar o conteúdo as outras disciplinas. Sendo assim, exaltando a importância de inserir aulas práticas tanto para consolidar a interdisciplinaridade, como motivar o aluno e prepará-lo para a realidade de sua profissão. Kohwalter, Clua e Murta (2011) descrevem problema semelhante direcionado ao ensino da disciplina de ES. Chaves et al. (2015) relata dilema semelhante em relação ao ensino de modelagem de processos de software.

Da Silva et al. (2012) enfrentam a dificuldade de fazerem estudantes de cursos de TI associarem conceitos de abstração característicos do processo de análise de requisitos a problemas reais, sugerindo o uso do jogo de estratégia “Modelando” como alternativa metodológica de ensino.

Farias et al. (2012) destaca a importância do ensino de teste de software para o curso de ciência da computação como forma de avaliar se os requisitos do usuário são de fato atendidos e como, apesar disso, essa disciplina recebe pouco destaque no decorrer do curso, não tendo

sido abordada de maneira adequada nos programas curriculares, além de ser vista ainda como um dos tópicos cuja aprendizagem na graduação ainda deixa muito a desejar.

Para Souza e França (2016) o infortúnio apresentado é não só a questão motivacional relacionada a evasão de alunos em cursos na área de TI, mas também formas de atrair interesse do público a área em questão. O que levou os autores analisar outras metodologias de ensino para sanar essa questão.

Monsalve, do Prado Leite e Werneck (2015) apresentam como as novas tecnologias influenciam no atual comportamento da sociedade atual e mudou as formas de interação pessoal levando a reflexão de maneiras de usar os avanços tecnológicos disponíveis como metodologias de ensino buscando inteirar a realidade social tecnológica ao aprendizado.

Ainda realçando a ideia de que há complexidade no ensino mais realista e prático, Letra, Paiva e Flores (2015) abordam a respeito da vasta gama de conteúdos inerente a área de ES e a também ampla variedade de técnicas de ensino associadas a esse conteúdo, levando-os a buscar uma metodologia de ensino que possibilite trabalhar esse vasto conteúdo de maneira interdisciplinar e prática.

Kupsch e Resende (2013) observam que o treinamento dos estudantes em situações próximas do ambiente de empresas que desenvolvem software é muito difícil. O principal fator relatado consiste na diversidade cultural das organizações e na natureza das aplicações de software. Contudo, a ES vê a simulação por meio de jogos como um campo de pesquisa com grande potencial e possibilidade de expansão.

*QP2. Quais métodos e técnicas têm sido utilizados como estratégia de ensino voltados a disciplina de ES?*

Após a extração das informações relevantes dos trabalhos selecionados no critério de inclusão e passados nos critérios de exclusão, observou-se o uso recorrente de jogos digitais de simulação que utilizam cenários voltados a realidade de trabalho da área de TIC, principalmente centrados em ES ou conteúdo dessa disciplina, como observados em (BENNITI; MOLLÉRI, 2008), (DA SILVA et al., 2012), (FARIAS et al., 2012), (KOHWALTER; CLUA; MURTA, 2011), (CHAVES et al, 2015), (MONSALVE; DO PRADO LEITE; WERNECK, 2015) e (KUPSCH; RESENDE, 2013).

O diferencial entre estas obras são os estratagemas utilizados nos jogos e o conteúdo da área de ES abordado, por exemplo: Benniti e Moléri (2008) utiliza-se de um jogo baseado em RPG, o SE*RPG (Software Engineering Role-Playing Game), para sua simulação e aborda o aspecto de gerenciamento de processos.

Da Silva et al (2012) faz uso do aspecto de estratégia dos jogos de Card Games (jogos de cartas) para abordar a temática de engenharia de requisitos utilizando o jogo modelando e seu trabalho avalia o uso dessa técnica de ensino. Modelando é um jogo competitivo de estratégia voltados para alunos da área de computação onde os jogadores são desafiados a elaborarem um modelo conceitual através das jogadas que apresentam situações que simulam a realidade desse tipo de atividade no mercado de trabalho. A avaliação foi feita através de questionário elencando as características positivas e negativas do jogo pelos alunos.

Farias et al (2012) utiliza um jogo interativo de estratégia voltado para apenas um jogador (single player) para trabalhar conceitos de planejamento de testes. Sua pesquisa avalia a eficácia dessa prática pedagógica. Para avaliação foi utilizado o modelo de avaliação de treinamentos (KIRKPATRICK, 1994) para elencar a qualidade do jogo e ARCS (KELLER, 1987) para averiguar as estratégias emocionais.

Kohwalter, Clua e Murta (2011) utilizam um jogo de estratégia single player estilo RPG onde o usuário é dono de uma empresa de software, gerencia funcionários e atribui escores nos seus atributos para melhorar características necessárias às tarefas que serão designadas, dessas formas vários conceitos de ES são abordados uma vez que o objetivo do jogo é entregar o software encomendado ao cliente.

Chaves et al (2015) avalia em seu trabalho o uso de um Serious Game (Jogo de simulação) como apoio ao ensino de modelagem de processos de Software (MPS), no jogo DesignMPS, o aluno modela um processo de software a partir de uma perspectiva SPI, baseada no modelo SPI brasileiro (MPS.BR).

Monsalve, Do Prado Leite e Werneck (2015) descreve a utilização do SimulES-W que é a versão digital do SimulES, um jogo de tabuleiro (*board game*) educativo e jogo de cartas. SimulES é uma evolução das idéias do jogo Problems and Programmers (Problemas e Programadores). Porém, Diferente do PnP, o SimulES-W não possui processo de desenvolvimento específico e o processo de desenvolvimento pode ser explorado pedagogicamente durante o jogo. O objetivo é fazer o aluno ter papel participante na construção de seu próprio conhecimento.

Kupsch e Resende (2013) apresenta o SPIAL (*Software Process Improvement Animated Learning Environment*) ou *Software* Ambiente de Aprendizagem Animado para Melhoria de Processos, como metodologia complementar no ensino de conceitos de ES, principalmente voltados para melhoria de processos de software. O jogo é um simulador interativo e personalizável que simula a realidade de uma empresa de software. A avaliação foi feita por Método de Inspeção Semiótica.

Houve um trabalho avaliado que não faz uso de jogos digitais como atividade didática, mas faz uso como tema e defendem a utilização de aulas práticas no ensino de ES e/ou disciplinas da área TI. Os autores Digiampietri, Kropiwiec e Silva (2010) descrevem como práticas voltadas ao desenvolvimento de jogos podem ser utilizadas como fator motivacional e de interdisciplinaridade envolvendo diversas disciplinas do curso de Sistemas de Informação como: Introdução a Ciência da Computação, Algoritmos e Estruturas de Dados, Inteligência Artificial, Interface Humano-Computador e ES.

Letra, Paiva e Flores (2015) fazem em sua tese uma revisão sistemática sobre padrão de design de jogos e funções de ensino e aprendizagem, utilizam-se questionário para levantar dados com professores da área para relacionar ensino e aprendizado a gerenciamento de projetos de *software*. Ao final, estabelecem a relação entre padrões de design de jogos e educação em gerenciamento de projetos de *software* por meio de um estudo empírico.

Sousa e França (2016) conduzem um levantamento bibliográfico a respeito do uso de jogos no ensino de ES sob o ponto de vista da Teoria da Motivação e Satisfação do Engenheiro de *Software* (TMS-ES), sendo este o principal conceito utilizado para elencar os estudos primários do trabalho e assim mensurar a taxa de sucesso dessa prática pedagógica.

## CONCLUSÕES

O ensino da disciplina de ES, muitas vezes, é direcionado no sentido da aprendizagem teórica, fazendo com que muitos discentes a vejam como pouco aplicada, bem como distante da realidade encontrada em empresas ou no mercado. Com base nisso, a pesquisa apresentada almejou identificar quais são as principais problemáticas relatadas no ensino da disciplina em questão, bem como quais eram as metodologias, técnicas e estratégias utilizadas como alternativas ao ensino por meio uma revisão sistemática na literatura. Como trabalhos futuros é pretendido a construção de um ambiente online que condense parte das metodologias utilizadas como forma de resolução à problemas específicos no ensino da Engenharia de Software.

## REFERÊNCIAS

BAKER, A.; NAVARRO, E. O.; VAN DER HOEK, A. An experimental card game for teaching software engineering processes. **Journal of Systems and Software**, v. 75, n. 1-2, p. 3-16, 2005.

BENITTI, F. B. V.; MOLLÉRI, J. S. Utilização de um RPG no ensino de gerenciamento e processo de desenvolvimento de software. In: **WEI-Workshop sobre Educação em Computação**. 2008. p. 258-267.

CHAVES, R. O. et al. Experimental evaluation of a serious game for teaching software process modeling. In: **Anais...**, v. 58, n. 4, p. 289-296, 2015.

DA SILVA, J. C. et al. Uma avaliação do emprego do jogo Modelando como apoio ao ensino de Engenharia de Requisitos. In: **Anais…** 2012.

DIGIAMPIETRI, L. A.; KROPIWIEC, D. D.; SILVA, R. A. C. O uso de jogos como fator motivacional em cursos de computação. In: **Anais…**,p.768 – 777, 2010.

FARIAS, V. et al. Itest learning: Um jogo para o ensino do planejamento de testes de software. In: **Anais…,** 2012.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1-26, 2004.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing Systematic Literature Reviews. In **Software Engineering, Technical Report EBSE- 2007-01**, Departament of Computer Science Keele University, Keele. 2007.

KOHWALTER, T, C.; CLUA, E. W. G.; MURTA, L. G. P. SDM-An educational game for software engineering. In: **Anais...**, 2011. p. 222-231.

KUPSCH, D. C. C.; RESENDE, R. S. F. SPIAL: Uma Ferramenta de Apoio ao Aprendizado de Melhoria de Processos de Software. In: **Anais…**, 2013.

LETRA, P.; PAIVA, A. C. R.; FLORES, N. Game Design Techniques for Software Engineering Management Education. In: **Anais...**, 2015. p. 192-199.

MONSALVE, E. S.; DO PRADO LEITE, J. C. S.; WERNECK, V. M. B. Transparently teaching in the context of game-based learning: the case of simulES-W. In: **Anais...**, 2015. p. 343-352.

PÁDUA, W. **Engenharia de Software: Fundamentos, Métodos e Padrões**. 1. ed. Rio de Janeiro: LTC, 2001. v. 1.

PRENSKY, Marc. Digital natives, digital immigrants part 1. **On the horizon**, v. 9, n. 5, p. 1-6, 2001.

PRENSKY, Marc. The motivation of gameplay: The real twenty-first century learning revolution. **On the horizon**, v. 10, n. 1, p. 5-11, 2002.

PRESSMAN, R. S. **Engenharia de Software: Uma abordagem profissional**. 7. ed. São Paulo: Pearson Makron Books, 2011.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, M.; FRANÇA, C. O Sucesso dos Jogos para Ensino de Disciplinas de Engenharia de Software sob a Ótica de uma Teoria Motivacional. In: **Anais...,** 2016. p. 450.

# CONTRIBUIÇÕES DA ENGENHARIA DE SOFTWARE PARA A EDUCAÇÃO

Bianca Souza Augusto,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

Felipe Ferreira de Almeida,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

Klebiana Moreira de Souza,
Discente do curso de Matemática do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

## INTRODUÇÃO

A ascensão tecnológica dos últimos anos e a crescente adesão aos novos dispositivos de informação, cria uma demanda por novas metodologias de ensino que se adaptem a realidade dos jovens, principalmente no que se refere aos estudantes das Instituições de Ensino Superior - IES (VARELA; BARBOSA; MACHADO, 2016).

Os aplicativos móveis e as redes sociais se tornaram ferramentas bastante difundidas entre os jovens e uma grande oportunidade para alunos e professores no que diz respeito ao desenvolvimento do ensino-aprendizagem. Para Garcia *et al.* (2008), ambos têm a oportunidade de desenvolver suas competências através destes meios.

Aplicativos voltados para a área da educação provocam melhores resultados em termos de aprendizagem devido ao fato de propiciar o acesso a novas metodologias condizentes a uma realidade na qual a grande maioria dos professores e alunos está cada vez mais conectada as redes virtuais.

Sabe-se que o processo de desenvolvimento de aplicativos se torna mais eficiente quando se utilizam métodos trazidos pela engenharia de *software*. No âmbito educacional necessita-se de tais processos para conseguir produzir um sistema de qualidade, que atenda as demandas envolvidas nos vários contextos de sala de aula (BITTENCOURT, 2012).

Este trabalho consta de uma pesquisa bibliográfica que analisa o papel das novas tecnologias no processo de ensino, mais especificamente no desenvolvimento de novas metodologias de ensino, nas das IES e investiga como a engenharia de software pode vir a contribuir da no desenvolvimento de aplicativos.

## METODOLOGIA

O trabalho proposto trata-se de um levantamento bibliográfico que objetiva investigar a inserção das novas tecnologias como ferramenta de ensino, bem como o papel da engenharia de *software* no desenvolvimento destas. Gil (2008), caracteriza a pesquisa bibliográfica como o levantamento de dados, em meios físicos ou virtuais, para conhecer um determinado assunto.

A pesquisa bibliográfica é o primeiro passo na concepção de qualquer trabalho, pois é ela quem irá nortear o desenvolvimento do mesmo, portanto é necessária uma atenção especial a esta parte do trabalho (FONSECA, 2002).

A referida pesquisa foi feita através das bases de dados: Capes (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), TISE (Conferência Internacional sobre Informática na Educação), CBIE (Conferência Brasileira de Informática na Educação) e Google Acadêmico. Foram pesquisados trabalhos publicados nos últimos dez anos que contivessem os seguintes descritores: redes sociais na educação, ensino e tecnologia, dispositivos móveis na educação e engenharia de *software*.

Dentre os trabalhos encontrados na pesquisa foram selecionados cinco para cada descritor, tendo como critério de inclusão serem artigos completos e em língua portuguesa. Após uma análise destes, foram dois para compor cada tópico abordado neste resumo, com exceção do tópico "o uso de dispositivos móveis na educação" para o qual escolheu-se apenas um, pois um dos artigos escolhidos para o descritor "ensino e tecnologia" aborda também o uso de celulares e tablets como ferramenta educativa. Usou-se como critério para esta última seleção o trabalho ter sido publicado nos últimos quinze anos.

Em relação aos objetivos, a pesquisa é definida como exploratória. A pesquisa exploratória procura investigar e analisar determinado fenômeno, procurando compreender e modificar ideias, e elaborar hipóteses sobre o mesmo. Utiliza para esse fim materiais bibliográficos e de levantamento de dados (GIL, 2008). A abordagem dar-se-á de forma qualitativa, na qual busca-se entender dinâmicas sociais (FONSECA, 2002). Neste caso, busca-se investigar o atual papel da tecnologia no desenvolvimento de novas metodologias de ensino.

## RESULTADOS E DISCUSSÕES

Nesta seção serão apresentados os resultados do levantamento bibliográfico. Abaixo apresenta-se uma tabela mostrando informações sobre os artigos utilizados e em seguida as ideias defendidas nos mesmos.

Tabela 01 – Artigos utilizados

| ID | Autor | Título | Ano | Síntese |
|---|---|---|---|---|
| A1 | VARELA, Danielle Santiago da Silva; BARBOSA, Maria Udete Facundo; MACHADO, Maria de Fátima Antero S | Desafios da Prática Pedagógica no Ensino Superior | 2016 | Este trabalho trata-se de um estudo de caso realizado com professores com o objetivo de investigar as práticas pedagógicas nas IES. |
| A2 | ALVES, Liduina Lopes *et al.* | Avaliação dos Desafios Relacionados às Práticas Docentes em Instituições do Ensino Superior | 2012 | Os autores buscam conhecer e analisar, através de um levantamento bibliográfico, os desafios enfrentados na atualidade pelos professores das IES. |
| A3 | ANDRADE, Carla; FERRETE, Anne Alilma Silva Souza | Formação docente: Percepções dos professores sobre o uso das tecnologias móveis digitais no processo de ensino e aprendizagem | 2017 | O artigo busca analisar a visão dos professores frente ao papel das novas tecnologias móveis no ensino. |
| A4 | SOUZA, Adriana Alves Novais; SCHNEIDER, Henrique Nou | Aprendizagem Nas Redes Sociais: Colaboração Online Na Prática De Ensino Presencial | 2012 | Neste trabalho é promovida uma discussão sobre o uso de redes sociais como ambiente de aprendizagem colaborativa. |
| A5 | BEZERRA, Júlio César Cavalcante; BRITO, Sydneia de Oliveira | Redes Sociais como ferramenta pedagógica: O caso do projeto e-Jovem. | 2013 | O artigo traz um estudo sobre o Projeto e-Jovem, que utiliza redes sociais para o ensino em tecnologia da informação. |
| A6 | BITTENCOURT, Ig Ibert *et al.* | Desafios da Engenharia de Software na Educação: Variabilidade de Sistemas Educacionais Inteligentes e | 2012 | Os autores apresentam uma discussão que busca relacionar os desafios envolvidos no contexto educacional com os desafio encontrados no |

| | | | | |
|---|---|---|---|---|
| | | Instanciação em Larga Escala | | processo de desenvolvimento de software |
| A7 | BENITTI, Fabiane Barreto Vassori; SEARA, Everton Flavio Rufino; SCHLINDWEIN, Luciane Maria | Processo de Desenvolvimento de Software Educacional: proposta e experimentação | 2005 | O artigo traz uma proposta de metodologia de desenvolvimento de um software educacional |

A1 defende que o papel do professor no Ensino Superior tem sofrido diversas mudanças nos últimos anos devido a fatores como a globalização e o fácil acesso às novas tecnologias. O autor afirma que atualmente, e cada vez mais, é exigido que o docente esteja preparado para lidar com tais mudanças, se aperfeiçoando no que diz respeito a sua metodologia de ensino e modelo de avaliação.

A2 concorda ao apontar que a maioria das aulas nas Instituições de Ensino Superior (IES) segue um padrão de exposição teórica do conteúdo seguida de uma avaliação dos mesmos. O problema apontado na referida obra é que os docentes tendem a transmitir conhecimentos da mesma maneira com que estas lhe foram passadas no momento de sua formação, fato que dificulta o aprendizado dos jovens de hoje.

A3 fala sobre o uso de celulares e tablets no contexto educacional e afirma que esses dispositivos têm sido utilizados como ferramenta de aprendizagem, por ser atrativa aos alunos e proporcionar uma melhor compreensão do conteúdo. A1 justifica o uso dessas ferramentas ao afirmar que as mesmas são atrativas aos alunos e proporcionar uma melhor compreensão do conteúdo.

A4 também defende a tecnologia como auxiliar no processo de ensino, em especial as redes sociais ao afirmar que a possibilidade de trocar informações, opinar e classificar conteúdos torna o meio proativo e propício a disseminação do conhecimento. A5 exemplifica este processo ao dizer que neste meio é comum a criação de grupos que falem, através de imagens, textos e vídeos, a respeito de um dado assunto.

A6 defende as contribuições da engenharia de *software* na educação e afirma que os softwares beneficiam o processo de ensino devido ao fato de adequar-se as necessidades das situações encontradas no contexto educacional. Segundo o autor, a variabilidade de modalidades de ensino é um desafio para os desenvolvedores, mas afirma, em contrapartida, que se conta com diversos recursos da engenharia de software para lidar com tais situações.

A7 concorda e complementa o pensamento de A6 quando diz que utilizar a informática em processos educativos pode tornar o processo de aprendizagem mais rápido ao mesmo tempo que incentiva o processo de compartilhamento de conhecimentos. O autor defende os processos de engenharia de software como essenciais no desenvolvimento de aplicativos com fins educacionais afirmando que este tipo de software exige efetiva participação dos futuros usuários, pois as funcionalidades de um sistema educacional precisam ser facilmente compreendidas.

## CONCLUSÕES

Foi possível constatar que os autores concordam que as novas tecnologias podem auxiliar o professor no que diz respeito a compreender e desenvolver metodologias modernas, capazes de facilitar o processo de aprendizagem de indivíduos. Logo, existe uma necessidade por aplicativos e redes sociais que incentivem o ensino e mostrem que a educação não precisa nem deve ser vista como algo antiquado.

O desenvolvimento dos referidos aplicativos não é uma tarefa fácil, visto que cada instituição de ensino traz uma realidade diferente em relação aos alunos e professores. As técnicas da engenharia de software permitem aos desenvolvedores uma maior familiarização com o público-alvo e, consequentemente, com os problemas e necessidades que motivam a construção desse tipo de aplicativo, maximizando sua qualidade e eficiência.

Pode-se concluir então que o desenvolvimento de aplicações orientadas a processos da engenharia de *software* pode vir a contribuir positivamente no contexto educacional, auxiliando no processo de aprendizagem.

Para trabalhos futuros, planeja-se desenvolver um software que irá criar uma rede colaborativa de metodologias de ensino.

## REFERÊNCIAS

ALVES, Liduina Lopes *et al*. AVALIAÇÃO DOS DESAFIOS RELACIONADOS ÀS PRÁTICAS DOCENTES EM INSTITUIÇÕES DO ENSINO SUPERIOR. **Revista Expressão Católica**, Ceará, 2012. Disponível em: < http://publicacoesacademicas.unicatolicaquixada.edu.br/index.php/rec/article/view/1283/1046 >. Acesso em: 07 abr. 2018.

ANDRADE, Carla; FERRETE, Anne Alilma Silva Souza. Formação docente: Percepções dos professores sobre o uso das tecnologias móveis digitais no processo de ensino e aprendizagem. **Anais dos Workshops do Vi Congresso Brasileiro de Informática na Educação (cbie 2017)**, [s.l.], 27 out. 2017. Brazilian Computer Society (Sociedade Brasileira de Computação - SBC). Disponível em: <http://dx.doi.org/10.5753/cbie.wcbie.2017.515>. Acesso em: 08 abr. 2018.

BENITTI, Fabiane Barreto Vassori; SEARA, Everton Flavio Rufino; SCHLINDWEIN, Luciane Maria. Processo de Desenvolvimento de Software Educacional: proposta e experimentação. **RENOTE**, v. 3, n. 1, 2005. Disponível em: < http://www.seer.ufrgs.br/renote/article/viewFile/13849/8025>. Acesso em: 20 abr. 2018.

BEZERRA, Júlio César Cavalcante; BRITO, Sydneia de Oliveira**. Redes Sociais como ferramenta pedagógica**: O caso do projeto e-Jovem. In: Congresso Internacional Abed de Educação a Distância. 2013. Disponível em: < http://www.abed.org.br/congresso2013/trabalhos/277.pdf >. Acesso em: 07 abr. 2018.

BITTENCOURT, Ig Ibert et al. Desafios da Engenharia de Software na Educação: Variabilidade de Sistemas Educacionais Inteligentes e Instanciação em Larga Escala. In: **Anais do Workshop de Desafios da Computação Aplicada à Educação**. 2012. p. 70-79. Disponível em: < http://br-ie.org/pub/index.php/desafie/article/viewFile/2777/2430>. Acesso em: 06 abr. 2018.

FONSECA, José saraiva da. **Metodologia da Pesquisa Científica**. Universidade Estadual do Ceará, 2002, 127p.

GARCIA, Marta Fernandes et al. NOVAS COMPETÊNCIAS DOCENTES FRENTE ÀS TECNOLOGIAS DIGITAIS INTERATIVAS. **Teoria e Prática da Educação**, v. 14, n. 1, p.79-87, 2011.. Disponível em: < http://ojs.uem.br/ojs/index.php/TeorPratEduc/article/view/16108 >. Acesso em: 07 abr. 2018.

GIL, Antonio Carlos. **Métodos e Técnicas de Pequisa Social**. 6ª ed. Editora Atlas S.A. São Paulo, 2008, 220p.

SOUZA, Adriana Alves Novais; SCHNEIDER, Henrique Nou. APRENDIZAGEM NAS REDES SOCIAIS: COLABORAÇÃO ONLINE NA PRÁTICA DE ENSINO PRESENCIAL. In: **SIMPÓSIO EDUCACIONAL DE EDUCAÇÃO À DSTÂNCIA**. São Carlos, 2012. Disponível em: < http://sistemas3.sead.ufscar.br/ojs/index.php/sied/article/viewFile/43/17 Acesso em: 06 abr. 2018.

VARELA, Danielle Santiago da Silva; BARBOSA, Maria Udete Facundo; MACHADO, Maria de Fátima Antero S.. DESAFIOS DA PRÁTICA PEDAGÓGICA NO ENSINO SUPERIOR. **Revista Expressão Católica**, Ceará, 2016. Disponível em: < http://publicacoesacademicas.unicatolicaquixada.edu.br/index.php/rec/article/view/1474/1207 >. Acesso em: 07 abr. 2018.

# O COMPUTADOR COMO INSTRUMENTO DE APRENDIZAGEM DO INDIVÍDUO COM ESPECTRO AUTISMO

*Jéssica Oliveira Ferreira,*
Graduanda em Bacharelado em Sistemas de Informação do IFCE Campus Cedro.

*Jéssica de Oliveira Maia,*
Graduanda em Bacharelado em Sistemas de Informação do IFCE Campus Cedro.

*Ednael Macêdo Félix,*
Professor do Curso de Bacharelado em Sistemas de Informação do IFCE Campus Cedro.

## INTRODUÇÃO

O Transtorno do Espectro Autista (TEA) atualmente é definido como um distúrbio do desenvolvimento neurológico, que deve estar presente desde a infância, apresentando déficits nas dimensões sociocomunicativa e comportamental (APA, 2013). Devido a existência de diferentes níveis do transtorno, proporciona-se a compreensão dos meios de mediação com tecnologias, tendo-se assim distinção nos processos de desenvolvimento tradicionais, agraciando suas necessidades, expectativas e experiências associados as singularidades e especificidades de diversificados agentes em interação. O cerne está nas intervenções que efetiva na interação, nas práticas culturais e nos equipamentos tecnológicos que estão envolvidos (PASSERINO, et al, 2013).

Segundo BORBA (2018) é de extrema importância, propor a reflexão e adesão de ambientes educacionais estruturados de forma a torná-los agradáveis a criança, rejeitando punições e "premiando" o comportamento desejado. Conforme ANJOS (2015) a problemática aqui abordada afeta aproximadamente 1 em cada 200 indivíduos, estão também entre os com maior carga genética entre os transtornos de desenvolvimento, há riscos de recorrência entre familiares entre 2 a 15% se for adotada uma definição mais ampla de critério diagnóstico. Os indivíduos com necessidades especiais tendem a serem mantidos as margens da sociedade, assim torna-se imprescindível o desenvolvimento de pesquisas e métodos que possam vir a auxiliar no desenvolvimento da criança.

Em conformidade com GLAT (2005) para oferecer uma educação de qualidade para todos os educandos, inclusive os portadores de necessidades especiais, a escola precisa capacitar seus professores, preparar-se, organizar-se, enfim, adaptar-se. "Inclusão não significa, simplesmente, matricular os educandos com necessidades especiais na classe comum,

ignorando suas necessidades especificas, mas significa dar ao professor e à escola o suporte necessário à sua ação pedagógica" (MEC-SEESP, 1998).

Levando em consideração a realidade social, torna-se controverso, apesar de todas as adversidades, negar a utilidade do computador como instrumento de aprendizado. Barbosa (2002, p.76), enfatiza a dificuldade dos professores em aceita-lo:

> [...] a resistência às mudanças surge no processo como uma forma de expressão, por parte das pessoas, a respeito de suas dificuldades em abrir mão de alguns dos seus significados de base de sua personalidade, ou da construção de seus esquemas cognitivos. É uma atitude de defesa de não verem destruídos os modelos que aprenderam no passado e que a seus olhos, deram certo. As pessoas se vêm diante de um dilema: se aceitar significa destruir algo em que acredita. Se rejeitar significa atrair represálias.

**1.1 Teoria do Construcionismo**

Na concepção de Papert (1994), o Construcionismo seria uma extensão do Instrucionismo, pois os esquemas ou estruturas cognitivas seriam construídos de modo especialmente venturoso, quando apoiados em algo tangível. A Teoria do Construcionismo não é uma concepção puramente mentalista, uma vez que ela reúne o trabalho intelectual do aluno e sua externalização por meio de diversos recursos disponíveis.

A informatização dos meios tradicionais de ensino se faz necessária. É nessa perspectiva que o computador surge como ferramenta educacional. Valente (1993, p.12) explica que segundo esta modalidade, o computador não é mais o instrumento que ensina o aprendiz, mas a ferramenta com a qual o aluno desenvolve algo, e, portanto, a aprendizagem ocorre pelo fato de estar executando uma tarefa por meio do computador.

Segundo Costa (2010) através do "computador ferramenta" o aluno será o sujeito promotor de uma ação, ou seja, seu lugar deixa de ser o de espectador e passa a ser o de agente. O aluno passa a ter uma postura ativa em relação ao conhecimento, e não mais passiva como antes.

De acordo com Papert (1994) as novas tecnologias abrem um portal para uma nova era, a era da informação. E as novas tecnologias conduzem-nos para um ambiente propício para a aprendizagem e, consequentemente, para o ensino por excelência.

**Plano de Ensino Individualizado - PEI**

O método utilizado no processo de ensino aprendizagem tornou-se um assunto bastante debatido entre familiares, educadores e médicos. A ABA (Análise do Comportamento Aplicada) baseia-se em princípios científicos, a fim de possibilitar à melhor intervenção a criança diagnosticada com autismo. Mediante tal análise é elaborado um plano de ensino individualizado, buscando desenvolver e/ou consolidar os potenciais do aluno. A estimulação

visual viabilizada pelo computador favorece o aprendizado uma vez que funciona como reforçador, onde a criança sente-se atraída a realizar a atividade a qual foi submetida.

## METODOLOGIA

A pesquisa em questão trata-se de um estudo descritivo, que se propõe a estudar os benefícios da adesão de uma tecnologia, o computador, já que pesquisas indicam que a utilização da mesma pode trazer melhorias, singularmente, enquanto ferramenta de aprendizagem. Considera-se para tanto, que a pesquisa descritiva é aquela na qual o pesquisador busca caracterizar determinada população ou fenômeno (FREITAS; PRODANOV, 2013).

Como procedimento optou-se por realizar o estudo com um grupo focal, uma vez que este é um tipo de estudo com grupos, alicerçado na comunicação e na interação. Sua meta é agrupar dados detalhados sobre um tema específico, a partir de um grupo de participantes selecionados. Este tem como objetivo coletar informações que possam proporcionar o entendimento de conceitos, crenças, atitudes sobre um tema, produto ou serviços (SILVA et al., 2017).

Esta define-se ainda como pesquisa de caráter quantitativo, pois é um método de estudos científicos que utiliza diferentes meios estatísticos para mensurar conceitos e informações para um determinado estudo (SILVA; LOPES; JÚNIOR, 2014).

Os participantes do estudo foram monitores e pais de indivíduos autistas, que atuam dando suporte na educação destes. Considerou-se como objeto de estudo o Modelo de Aprendizagem empregado pelos monitores, tendo como foco a utilização do computador como ferramenta de ajuda para crianças com autismo.

A coleta de dados ocorreu entre os meses de abril a maio de 2018, através da aplicação de questionário online, cujo objetivo foi proporcionar a compreensão de entes mais instruídos ou adjuntos ao sujeito que se encontra correlacionado ao objeto analisado e investigar os métodos educacionais tecnológicos usados na educação de crianças autistas.

## RESULTADOS E DISCUSSÕES

Posteriormente a aplicação do questionário, foram recolhidos os discernimentos dos pais e instrutores da criança sobre a utilização de tecnologias na aprendizagem dos mesmos.

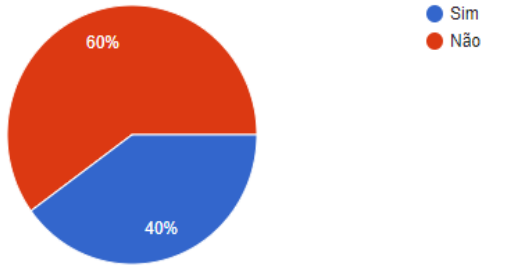
A pesquisa abrangeu macrorregiões do estado do Ceará, especificamente, Centro-Sul, Cariri e na Grande Fortaleza, sendo distribuída propriamente nos municípios de Cedro, Juazeiro do Norte, Crato, Barbalha, Farias Brito e Fortaleza. Considerando os dados sociodemográficos

disponibilizados pelos 25 participantes, constatou-se que 44% das crianças possuem idade de 1 a 5 anos, com uma taxa relativamente maior as que possuem de 6 a 10 anos com 48% e apenas 8% acima de 10 anos. Além de haver uma relevância considerável em relação ao sexo da criança, onde 84% são do gênero masculino e apenas e 16% do gênero feminino.

De acordo com as indagações, foi perceptível que todos os envolvidos já notaram algum interesse da criança com Espectro Autista por equipamentos eletrônicos, significativamente pelo computador.

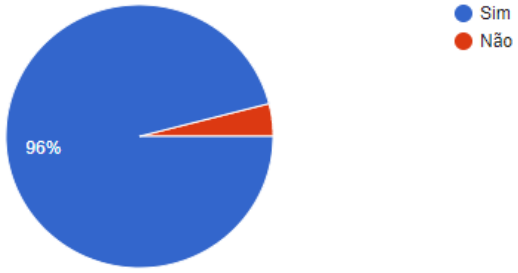
Conforme os indivíduos foram questionados sobre os métodos educacionais instituídos na escola, no qual os 60% indicaram que a metodologia utilizada pela escola não seria conveniente para crianças autistas, de acordo com o gráfico 1. O que é relevante pontuar, é que 96% dos pesquisados, consentem com a admissão de novas técnicas de aprendizagem, como revela o gráfico 2.

Gráfico 1 - Aceitamento dos métodos utilizados pelas escolas como algo adequado para autistas.




Fonte: Dados da Pesquisa (2018)

Gráfico 2 - Concordância na adoção de novos métodos de aprendizagem

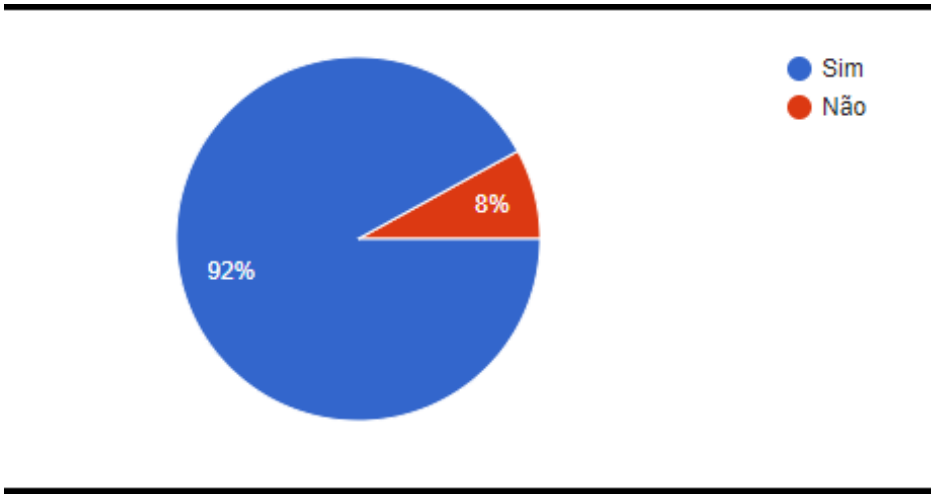



Fonte: Dados da Pesquisa (2018)

No decorrer dos questionamentos foram apresentados aos participantes fatores que poderiam influenciar de forma negativa na aprendizagem do filho com TEA, segundo 36% dos pesquisados, o método instituído pela escola é o principal fator de influência negativa. A falta de interesse da escola em apresentar inovações, foi indicado como fator, por 20%. O preconceito e a falta de comunicação com os colegas, tiveram respectivamente 16% e 8% de indicação. Logo, os métodos aparecem como fatores mais determinantes, na percepção dos pais, do que fatores como preconceito e falta de comunicação.

Quando indagado aos participantes sobre as tecnologias, como: o computador, para auxiliar na melhoria da aprendizagem, ficou explicito que muitos acreditam que o computador fomenta avanços expressivos, ao ponto que 92% indicaram essa perspectiva conforme o gráfico 3. Desta maneira, o computador seria então uma ferramenta de auxílio a dinamicidade e entusiasmo das crianças,

Gráfico 3 - Concordância que os recursos tecnológicos possam ajudar de uma forma mais dinâmica e satisfatória no aprendizado.




Fonte: Dados da Pesquisa (2018)

## CONCLUSÕES

O estudo constatou que o uso do computador como ferramenta de ensino pode colaborar consideravelmente no meio de aprendizagem de crianças autistas. Estes, demonstram interesse e afeições pelo aparelho, visto que o mesmo atrai a concentração dos entes por meio de seus recursos, desta maneira tornando a metodologia aplicada mais aprazível e acessível.

No entanto, analisando os resultados, observou-se não somente dificuldades em aprender, mas também na aceitação e atração de instituições por melhorias de ensino e inovações com auxílio tecnológicos, retardando cada vez mais a extinção de algumas limitações causadas pelo transtorno.

Portanto, o computador enquanto instrumento que envolve diferentes tipos de aprendizagem, acomoda-se como uma ajuda técnica, propiciando progressos com habilidades na cognição, interação e autonomia pessoal ou até mesmo total de pessoas que possuem tais funcionalidades reduzidas (LIMA, 2007).

## REFERÊNCIAS

ANJOS, Reinaldo Tavares dos; MARTINES, Elizabeth Antônia Leonel de Moraes**. O computador como instrumento mediador na educação de alunos autistas.** 2015.
American Psychiatric Association. 2013. **Diagnostic and statistical manual of mental Disorders.** 5 ed. Washington, DC: American Psychiatric Publishing.

BARBOSA, D. A. **Utilizando o computador como ferramenta pedagógica para vencer a resistência do professor** - O caso da 38ª superintendência Regional de Ensino de Ubá-MG.

2002, 104 f.: Dissertação (Mestrado) - Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis-SC, 2002.

BORBA, M. M. C.; BARROS, R. S**. Ele é autista: como posso ajudar na intervenção? Um guia para profissionais e pais com crianças sob intervenção analítico-comportamental ao autismo.** Cartilha da Associação Brasileira de Psicologia e Medicina Comportamental (ABPMC), 2018.

BRASIL. Ministério da Educação. **Secretaria de Educação Especial. Diretrizes Curriculares Nacionais para a Educação Especial**, 1998.
COSTA, Thais Cristina Alves. **Uma abordagem construcionista da utilização dos computadores na educação**. Universidade Federal de Pernambuco, p. 32, 2010.

GLAT, Rosana; FERNANDES, Edicléa Mascarenhas**. Da educação segregada à educação inclusiva: uma breve reflexão sobre os paradigmas educacionais no contexto da educação especial brasileira.** Revista Inclusão, v. 1, n. 1, p. 35-39, 2005.

GOMES, Marina. **Biologia do Autismo**. Ciência e Cultura. Vol. 66 n. 1. São Paulo, 2014. Disponível em: http://cienciaecultura.bvs.br/scielo.

LIMA, Niusarete Margarida de**. Legislação Federal Básica na área da pessoa portadora de Deficiência.** Brasília: Secretaria Especial dos Direitos Humanos, Coordenadoria Nacional para Integração da Pessoa Portadora de Deficiência, 2007.

PAPERT, Seymour. **A máquina das crianças**. Porto Alegre: Artmed, 1994.

PASSERINO, Liliana Maria, BEZ, Maria Rosangela. ***Building an Alternative Communication System for Literacy of Children with Autism (SCALA) with Context Centered Design of Usage.*** Recent Advances in Autism Spectrum Disorders- Volume I, Prof. Michael Fitzgerald (Ed.), ISBN: 978-953-51-1021-7, InTech, DOI:10.5772/54547. (2013).

PRODANOV, C.C.; FREITAS, E.C. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico-2ª Edição**. Editora Feevale, 2013.

SILVA, D.; LOPES, E.L.; JUNIOR, S.S.B. **Pesquisa Quantitativa: Elementos, Paradigmas e Definições.** Revista de Gestão e Secretariado, v. 5, n. 1, p. 01, 2014.

SILVA, S.C. et al. **Desafios na operacionalização da técnica de grupo focal para coleta de dados em pesquisa qualitativa.** Semana de Enfermagem (28.: 2017: Porto Alegre, RS). Enfermagem e suas dimensões: a gestão do cuidado e o impacto na saúde; anais; [recurso eletrônico]. Porto Alegre: HCPA, 2017. 1 CD-ROM, 2017.

VALENTE, José Armando. **Formação de profissionais na área de informática em educação. Computadores e Conhecimento: Repensando a Educação.** Primeira edição, Campinas: NIED–Unicamp, p. 114-134, 1993.

# OS CONCEITOS DE ENGENHARIA DE SOFTWARE APLICADOS NA ORGANIZAÇÃO DE UM EVENTO ACADÊMICO

Adalmária Diniz Ferreira,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro. sapehdra2@gmail.com.

Rafaella Alves de Sousa,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

Francisco Soares da Silva Júnior,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

Manoel Victor Cavalcante Inácio,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

Pedro Luís Saraiva Barbosa,
Docente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro. pedro.barbosa@ifce.edu.br.

## INTRODUÇÃO

A área de Engenharia de Software possui uma vasta gama de conteúdos, de forma que uma base de conhecimento online o Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) foi elaborada para auxiliar a compreensão dessa variedade (LETRA, 2015). Os conceitos desses conteúdos em sua maioria são repassados ostensivamente de maneira teórica dificultando a inserção do conteúdo a realidade do aluno e impedindo-os de enxergar sua conexão com as demais disciplinas.

O objetivo deste trabalho é relatar como alunos da disciplina de engenharia de software do curso de Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) - Campus Cedro utilizaram os conceitos aprendidos durante a disciplina como alicerce na elaboração de um evento acadêmico real de nível regional.

Entre os conteúdos aprendidos e implementados estão: engenharia de requisitos, processo de desenvolvimento de software, gerência de projetos de software, métricas de software e teste de software que foram conceituados e apresentados pelo professor nas aulas da

disciplina de Engenharia de Software. O docente da disciplina utilizou-se de metodologias ativas durante as aulas para uma melhor qualidade no processo de ensino-aprendizagem. Entre essas metodologias podem se destacar a Metodologia Baseada em Problemas que oportunizou os alunos a compreender e aplicar os assuntos em contextos e a Metodologia Baseada em Projetos que foi essencial na construção do evento.

É importante salientar que o evento desenvolvido nesse processo é pioneiro na região e se faz necessário para favorecer discussões construtivas acerca da sua área temática, no caso, Engenharia de Software. Além de possibilitar relações dos discentes do IFCE Campus Cedro com profissionais e alunos de outros contextos e vivências.

## METODOLOGIA

Esta pesquisa possui natureza aplicada pois segundo Prodanov e Freitas (2013) envolve interesse locais e envolve geração de conhecimento com aplicabilidade prática, no caso o uso de conhecimentos de Engenharia de Software voltado a organização de um evento acadêmico na região. Quanto aos objetivos ela é descritiva explicativa pois de acordo com Gil (2007), pesquisa descritiva é aquela que visa a descrição de aspectos de uma população ou fenômeno, e explicativa, pois de acordo com Gerhardt e Silveira (2009), a mesma ocupa-se em elencar os fatores que estabelecem ou influenciam na incidência do fenômeno.

A abordagem é qualitativa e o procedimento técnico utilizado é estudo de caso que consiste em coletar e analisar informações sobre determinado indivíduo, uma família, um grupo ou uma comunidade, a fim de estudar aspectos variados de sua vida, de acordo com o assunto da pesquisa (PRODANOV e FREITAS, 2013).

O docente utilizou-se da metodologia baseada em projetos que segundo Fernandes et al. (2010), consiste em centralizar o processo de aprendizagem em volta do aluno, fazendo ir além da memorização e favorecendo a assimilação de novas competências, destacando assim a relevância de trabalhos de projeto, levando-os a desenvolver não só as competências técnicas da sua área de atuação como também competências transversais encontradas em um cotidiano normal de ambiente de trabalho, como: gestão de conflitos, assumir responsabilidades, interlocução, dirigência e gestão de tempo.

No caso o projeto desenvolvido foi a elaboração de um evento acadêmico da área de Engenharia de Software cuja organização foi feita no decorrer da disciplina homônima, as etapas de evolução do evento deram-se concomitantemente ao conteúdo da disciplina e serviram para estruturar o desenrolar das atividades feitas pelos alunos.

O conceito básico envolvido no desenvolvimento do evento foi bastante simples: o docente da disciplina seria o cliente; o serviço, a organização de um evento na área de

Engenharia de Software; e os alunos seriam membros da empresa responsável pela realização do evento.

Dessa forma, durante o decorrer da disciplina foi discutido entre cliente e empresa (professor/alunos) como iriam proceder as ações para realização do evento, desde da separação das atividades entre as equipes envolvidas e os métodos que seriam utilizados para realizar os objetivos. Para tal, ocorreu a divisão dos alunos em 4 grupos específicos:

- **Equipe de divulgação:** Responsável por realizar todos os procedimentos relativos à divulgação do evento, desde a criação de redes sociais até a articulação com possíveis apoiadores externos, como também a busca por patrocinadores;
- **Equipe de organização:** Com a atribuição de estruturar a sistemática interna, como materiais necessários, controle do fluxo do público no dia do evento;
- **Equipe de logística:** logística tinha como designação tornar possível as atrações, entrando em contato com palestrantes, reservando auditório, salas e laboratórios para oficinas e workshops;
- **Equipe de produção audiovisual:** Responsável pela criação da arte do evento, como o desenvolvimento de um site oficial, logotipo, banners e as imagens que foram utilizadas para divulgação do evento em geral.

## RESULTADOS E DISCUSSÕES

Analisando as observações feitas durante o desenvolvimento do evento, pode-se nitidamente perceber a utilização dos conceitos da Engenharia de Software na realização do evento. Os conceitos abordados pelo professor em sala de aula eram fixados pelo uso prático do conhecimento adquirido aplicados na administração das atividades das equipes do evento para atingir as metas e prazos estabelecidos, juntando assim experiência profissional e acadêmica.

Na **Tabela 1** observa-se o conceito aprendido, breve explanação de sua aplicação e em que momento do planejamento foi utilizado:

**Tabela 1 - Correlacionamento da aplicação dos conceitos de Engenharia de Software com as etapas de evolução do evento.**

| Conceito | Aplicação | Etapa do Evento |
|---|---|---|
| **Engenharia de Requisitos** | Foram realizadas reuniões com o "cliente" para coletar os dados necessário para a elaboração do evento. | Fase inicial - definição das atrações do evento: palestras, workshops, oficinas, apresentação de trabalhos acadêmicos. |
| **Processo de desenvolvimento de software** | Definiu-se as atividades a serem executadas, quando e por quem. Assim como padrões de | Fase de captação - início das atividades das equipes de trabalho visando a captação e elaboração |

| | | |
|---|---|---|
| | desenvolvimento. Utilizou-se a ferramenta online *Trello* para organizar as atividades do *SCRUM* online do produto *backlog*. O *Trello* simula a utilização da técnica do quadro *Kanban*. | dos recursos a serem utilizados tanto físicos: espaço, mesas, cadeiras; quanto humanos: palestrantes, professores, voluntários...Como financeiro: patrocinadores e anunciantes. E até mesmo tecnológicos: mídias sociais, site, sistema de submissão, equipamentos de áudio e vídeo. |
| **Gerência de Projetos** | Utilizou-se como base para administração de recursos de maneira colaborativa, visando cumprir os prazos e garantir que os requisitos estivessem de acordo com o que foi negociado com o cliente. | Fase de estruturação - organização das equipes a fim de desenvolver de maneira plena suas atividades de maneira a otimizar o tempo e a utilização de recursos de forma a evitar desperdícios. |
| **Métricas de software** | Foi utilizado esse conceito na elaboração do orçamento do evento, o custo para trazer participantes, compra de materiais, investimento em divulgações, prazos de fechamento de patrocínios. Um cronograma de atividade foi elaborado com as responsabilidades de cada equipe e seus prazos e metas. Assim como para atividades colaborativas. | Fase de planejamento - Cálculo de custos e definição de cronogramas de atividades com os prazos de cada equipe e coordenação de atividades conjuntas. Determinação de vagas em minicursos e oficinas, previsão de público, determinação de estratégias para organização de fluxo de participantes. |
| **Teste de software** | O teste é utilizado para validar o projeto/software desenvolvido antes do uso, de forma a evitar possíveis erros e tomar medidas necessárias para prevenção caso ocorra algum empecilho. | Fase de Teste - Foram feitas simulações de diversas atividades chaves para evitar possíveis falhas de execução durante sua realização. Assim como elaboração de planos de contingência para possíveis erros, observados e previstos pelas equipes. |

**Fonte:** Autores da pesquisa, 2018.

As equipes tiveram como alicerce a utilização de processos de desenvolvimentos de softwares, que segundo Sommerville (2011), é um processo que diz respeito a um conjunto de atividades relacionadas que levam a produção de software. Tendo como base tal definição, o professor e os alunos, trouxeram-na para o desenvolvimento do evento.

Iniciando com o estabelecimento do contato e desenvolvimento do relacionamento com o cliente, estabelecendo reuniões rotineiras e iniciando a coleta dos requisitos para o evento. Um *brainstorming* foi feito juntamente com o cliente não só visando coleta de ideias, mas também incrementar ideias já determinadas baseadas nos desejos e expectativas do mesmo. A base dessa atitude vem do conteúdo aprendido em engenharia de requisitos, que é a base da coleta e análise dos dados necessário para a elaboração de um software que atenda às necessidades fomentadas pelo cliente ou público alvo.

Para organizar o evento, foi utilizado conhecimento de algumas metodologias ágeis, como é o exemplo da metodologia *SCRUM*. O *SCRUM* tenta auxiliar o desenvolvimento de forma a controlar os processos de criação do produto/software e a equipe de forma a minimizar possíveis falhas, isto é, é definido qual produto será criado e a equipe dividirá o produto em *sprints, que* são parcelas pequenas do produto, cada *sprint* deve ter um tempo determinado a ser concluída, evitando assim possíveis atrasos do projeto. Além disso, melhora de forma considerável a organização do projeto e a dinâmica da equipe. A aplicação dessa técnica auxiliada pelo uso de quadros Kanbans por algumas equipes proporcionou uma visão ampla do projeto e ajudou a antecipar falhas, cumprir metas e auxiliou na distribuição de tarefas. Essa metodologia ajudou na gestão de conflitos a respeito do acúmulo excessivo de funções na divisão de trabalho.

O conteúdo de métricas de software influenciou na gestão de custos, desenvolvimento do cronograma de atividades dos grupo e programação do evento. Mas, além disso, ajudou a equipe de logística a organizar previsão do público, organizar o número de vagas a serem disponibilizadas nas oficinas e workshops, enfim organizar as atividades de forma a atender a demanda, dados esses que também serão de grande valor para a equipe de organização durante o evento na hora de controlar o fluxo de participantes. Métricas de software ensina a importância de mensurar características do que está sendo administrado de forma a avaliar a melhor forma de gerenciar e tomar decisões a respeito das mesmas.

Os conceitos de gerência de projetos foram amplamente utilizados dentro da questão de atender aos requisitos. Modificações de estruturas e tomadas de decisões complexas tiveram que ser feitas, pois no decorrer do tempo o trabalho da equipe de divulgação e equipe de produção audiovisual renderam, elevando o status do evento a outro patamar, a turma de Engenharia de Software de Sistemas de Informação não possuía mais apenas um cliente para suprir os requisitos, eles passaram a ser responsáveis por todo um público de mais de 300 pessoas e tinham que corresponder às expectativas desse público. Novos requisitos tiveram que ser elaborados visando o público do evento e novas estratégias tiveram que ser desenvolvidas buscando-se harmonizar o que o cliente tinha planejado inicialmente com o que o público esperava encontrar.

Dentro dos conceitos de teste de software, as equipes debateram problemas que poderiam surgir, imaginando possíveis cenários dessas situações, como: público maior que o esperado; palestrante faltando; poucos locais para refeições; atrasos na programação.... Tudo isso visando traçar planos de contingência para essas possibilidades. Em alguns casos foram feitas simulações, como no caso do sistema de submissão implementado para facilitar a avaliação dos trabalhos acadêmicos que seriam apresentados durante o evento. Nove sistemas

diferentes foram testados, até que se encontrasse um que atendesse as necessidades do evento e as especificações do servidor do instituto. Depois diversos cenários de usabilidade foram testados para se elaborar um manual para auxiliar os autores a se inscreverem. Além disso, um dos planos de contingência traçados ajudou na recuperação da maioria dos dados do sistema de submissão após um mal funcionamento do isolamento elétrico da rede do IFCE - Campus Cedro danificar o servidor onde estes dados estavam armazenados.

## CONCLUSÕES

Segundo a análise dos fatos observados, os conceitos de engenharia de software podem ser aplicados em atividades que não envolvem necessariamente o desenvolvimento de um software e mesmo assim serem compreendidos de forma plena por ter feito parte da elaboração de algo diretamente ligado à realidade dos alunos.

Até a escrita final do resumo expandido, o evento acadêmico já contava com diversas atrações confirmadas: 4 workshops, 4 minicursos, 3 palestras, duas atividades games para distrair o público nos intervalos e atrações culturais para o encerramento do evento. Além disso, já haviam mais de 300 inscritos no site oficial com formação de caravanas de diversas cidades da macrorregião como: Iguatu, Icó, Crato e Jaguaribe.

Alunos e professores de engenharia de software e TIC haviam inscritos seus trabalhos acadêmicos para apresentar durante o evento, visando a ampliação do debate sobre o conhecimento da área. Dezenas de professores haviam se inscrito no sistema de submissão no intuito de colaborar com o evento fosse como avaliadores, editores, revisores ou autores.

Como pesquisa futura pretende-se relatar os resultados obtidos durante e após a execução do evento a fim de mensurar seu impacto na macrorregião e sua relevância dentro da área de Engenharia de Software.

## REFERÊNCIAS

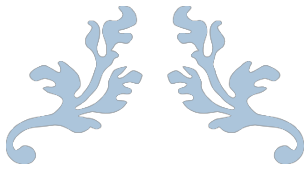

FERNANDES S R, FLORES M A, LIMA R M. **A Aprendizagem Baseada em Projectos Interdisciplinares: Avaliação do Impacto de uma Experiência no Ensino de Engenharia**. Avaliação, Campinas; Sorocaba, SP, v. 15, n. 3, p. 59-86, nov. 2010. Disponível em: <http://www.scielo.br/pdf/aval/v15n3/04.pdf>, acesso em 29 de abr. de 2018.

GERHARDT T. E., SILVEIRA D. T. **Métodos de pesquisa**. 1. ed. Rio Grande do Sul: UFRGS, 2009.

GIL, A. C. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2007.

LETRA, P.; PAIVA, A. C. R.; FLORES, N. Game Design Techniques for Software Engineering Management Education. In: **Anais...**, 2015. p. 192-199.

PRODANOV C C. FREITAS E. C. **Metodologia do trabalho científico: Métodos e técnicas da pesquisa e do trabalho acadêmico**. 2. ed. Rio Grande do Sul: Feevale, 2013
SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.



# ENGENHARIA DE REQUISITOS

# APLICAÇÃO DA TÉCNICA DE PROTOTIPAÇÃO NO PROCESSO DE AMADURECIMENTO DE REQUISITOS PARA DESENVOLVIMENTO DE UM SOFTWARE NO AMBIENTE ACADÊMICO

Fernanda Ferreira do Nascimento,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

Cesar Pinheiro Miranda,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

Francisca Adriana Monteiro de Sousa,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

Kamila Kelly Cardoso de Lima,
Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro.

## INTRODUÇÃO

Com o avanço da tecnologia, o software se tornou um instrumento de suma importância no ambiente acadêmico, na indústria, no comércio e até mesmo em casa, elevando a demanda eletrônica e gerando assim produção diária de novos dispositivos. A Engenharia de Software área dedicada a tal fim, pode ser definida como o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais (PRESSMAN, 2006), ou também como a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, operação e manutenção de software (IEEE STANDARD 610.12).

Na Engenharia de Software são concebidas Metodologias para desenvolvimento do *software*, coleta de necessidades do cliente, estudo de prazos e custos analisados pela equipe responsável garantindo assim, um produto eficiente. O processo de desenvolvimento de software é composto pelas etapas de levantamento, validação e gerenciamento de requisitos, onde o sucesso do desenvolvimento depende da interação entre o cliente e a equipe no decorrer dessas etapas. Na etapa de levantamento de requisitos, o cliente exibe quais as suas necessidades e seus desejos de maneira que o desenvolvedor converte isso em funcionalidades para o sistema.

Uma técnica indicada para amadurecer os requisitos é a prototipação. “A prototipação é uma representação visual do produto de software, através da simulação das telas da aplicação a qual permite ao usuário visualizar a aplicação que ainda não foi produzida” (SANTOS Fábio, SILVA Custódio, 2003).

O objetivo da prototipação é buscar a similaridade à aparência e funcionalidades do sistema, assim, permitindo os *stakeholders*[1], avaliar, sugerir alterações, além de interagir com o protótipo, influenciando diretamente na produtividade da equipe, ajudando a entender o propósito do software que será desenvolvido, assim, minimizando riscos e propondo melhorias. Esse processo ajuda a entender facilmente os requisitos do software, com isto, pode-se propor uma solução adequada para o problema do cliente. Com a utilização desse processo evita-se gastos dispendiosos de tempo e monetário.

Diante do exposto o presente estudo tem como objetivo apresentar a técnica de prototipação para amadurecimento dos requisitos de um software de acompanhamento das atividades acadêmicas dos alunos do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) - *Campus* Cedro. O referido software dará suporte aos departamentos Da Direção de Ensino, Assistência Estudantil, Coordenações de Curso, Serviço Social e Setores de Saúde no que se refere a ocorrências de atendimento a estudantes. Atualmente os registros dessas ocorrências acontecem de forma manual, fazendo assim com que algumas informações se percam ou se dupliquem quando encaminhadas a diferentes setores.

## METODOLOGIA

Para que se atinja os objetivos desta pesquisa, seguiu-se as classificações de pesquisa de Gil (1987). O projeto é de natureza aplicada, “a pesquisa que gera novos conhecimentos para aplicação prática, dirigidos à solução de problemas específicos. Envolve verdades e interesses locais.” (GIL, 1987, p.41).

A abordagem é qualitativa, visando utilização de protótipos para aperfeiçoar os requisitos do cliente. Também se caracteriza como uma pesquisa exploratória – a investigação de algum objeto de estudo que possui poucas informações. Para os procedimentos técnicos utilizou-se pesquisa no ambiente onde o software será implementado, assim como reuniões com usuários que farão a utilização do software em questão. Esse processo ocorreu nas seguintes etapas:

**Etapa 1:** O primeiro contato com o cliente, se deu em uma reunião com uma a coordenadora de ensino. A mesma relatou a problemática existente e o que almejava para solucionar.

[1] Público interessado.

Depois dessa reunião com o cliente, as equipes de desenvolvimento se reuniram com o professor da disciplina de Engenharia de Software, analisaram o problema traçando estratégias para sua resolução. Em análise percebeu-se que a cliente não tinha clara convicção do que realmente precisava e diante disso se concluiu que de todas as técnicas de desenvolvimento estudadas em sala de aula, a mais eficiente seria a de prototipação, para que quando a cliente observasse sua ideia consolidada pudesse daí mostrar os requisitos desejados. Seguindo na reunião, a equipe desenvolveu no papel o primeiro desenho de protótipo e através dele, foi gerado o primeiro Documento de Requisitos funcionais para o sistema.

**Etapa 2:** De posse dos requisitos Iniciais e do primeiro protótipo a equipe seguiu em busca de mais informações pois foi percebido que o sistema deveria conter informações específicas de cada setor e não poderia ser genérico entre todos departamento haja visto que cada um deles lidam com diferentes informações. Com novas informações coletadas, percebeu-se a necessidade de reenquadramento do protótipo já construído, visando a maior interação dos setores envolvidos sem que ocorressem exposições indesejadas ou omissão de informações importantes.

**Etapa 3:** Com a equipe reunida novamente e de posse dos novos dados coletados a criação de novo protótipo são colocados em pauta, para que seja discutido uma melhor forma de reorganização da documentação inicial e de verificação com o ideal do projeto. Com o novo protótipo pronto, foi levado até a cliente que o aprovou, pedindo apenas para adicionar algumas funcionalidades.

Encerrando este ciclo de planejamento, construção de protótipos e feedback do cliente, a equipe seguiu com o projeto, de modo a preparar toda a documentação de versões, documento de requisitos e métricas do sistema.

## RESULTADOS E DISCUSSÕES

Os protótipos de alta fidelidade equivale a apresentações mais aproximadas do sistema a ser criado, em certas situações, é possível simular as funcionalidades do sistema, como os aspectos visuais, os fluxos de navegação, permitindo ao usuário interagir com o ambiente em desenvolvido(NASCIMENTO, 2013).

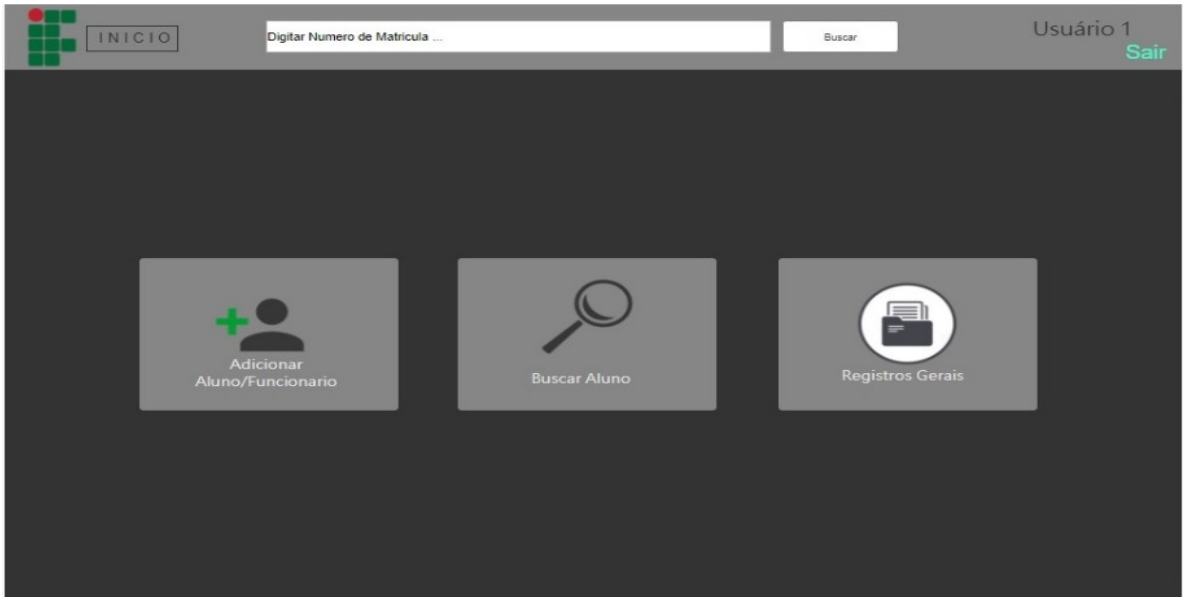
O sistema em questão auxilia a direção de ensino no controle e manutenção de ocorrências ligadas aos discentes da instituição, agilizando assim o processo de decisão das mesmas, gerando relatórios, agilizando os trabalhos e ligando os garfos da direção.

A Figura 1 apresenta a tela de login do sistema, todo usuário válido poderá acessar o sistema, mas com algumas restrições de sistema para um ambiente hierárquico. A Figura 2 apresenta a tela principal do sistema, a partir desta interface serão iniciadas todas as operações de cadastros de alunos, funcionários e registro de ocorrências. No menu de movimentação serão efetuados cadastros bem como consultas, geração de relatórios e modificação de cadastros.

**Figura 1** - Tela de login


**Fonte:** Equipe do Projeto(2018)

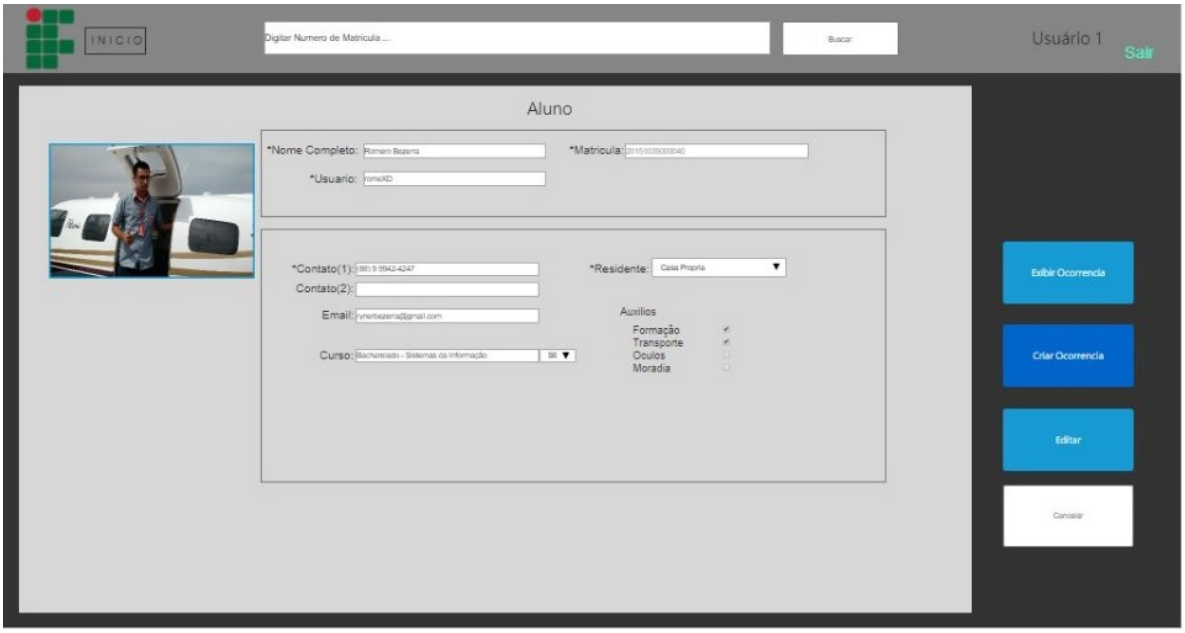
**Figura 2** - Tela Principal do Sistema




**Fonte:** Equipe do Projeto(2018)

A Figura 3 e 4 apresenta uma tela para o cadastro de alunos e funcionários, dentro desta tela o usuário apto poderá incluir ou alterar cadastros, bem como atualizar informações referentes aos dados salvos. A Figura 5 apresenta a tela cadastro de ocorrências onde as assistências poderão cadastrar eventualidade do discente, podendo também visualizar um fato já registrado no formulário do mesmo.

**Figura 3** - Tela de Cadastro do Aluno

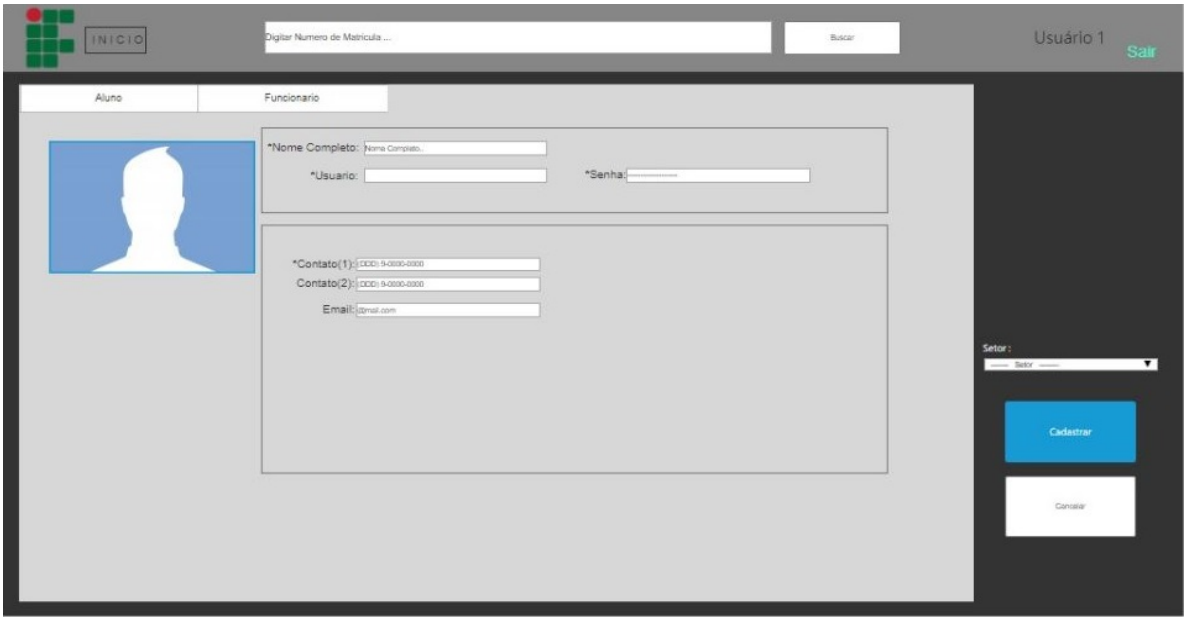



**Fonte:** Equipe do Projeto(2018)

Como foi solicitada pela direção de ensino nenhuma informação pode ser apagada ou alterada, todo registro deve ser mantido intacto, mesmo alterações devem manter a origem do dado e ser gerado um novo, para que não ocorram eventuais perdas dessas informações.

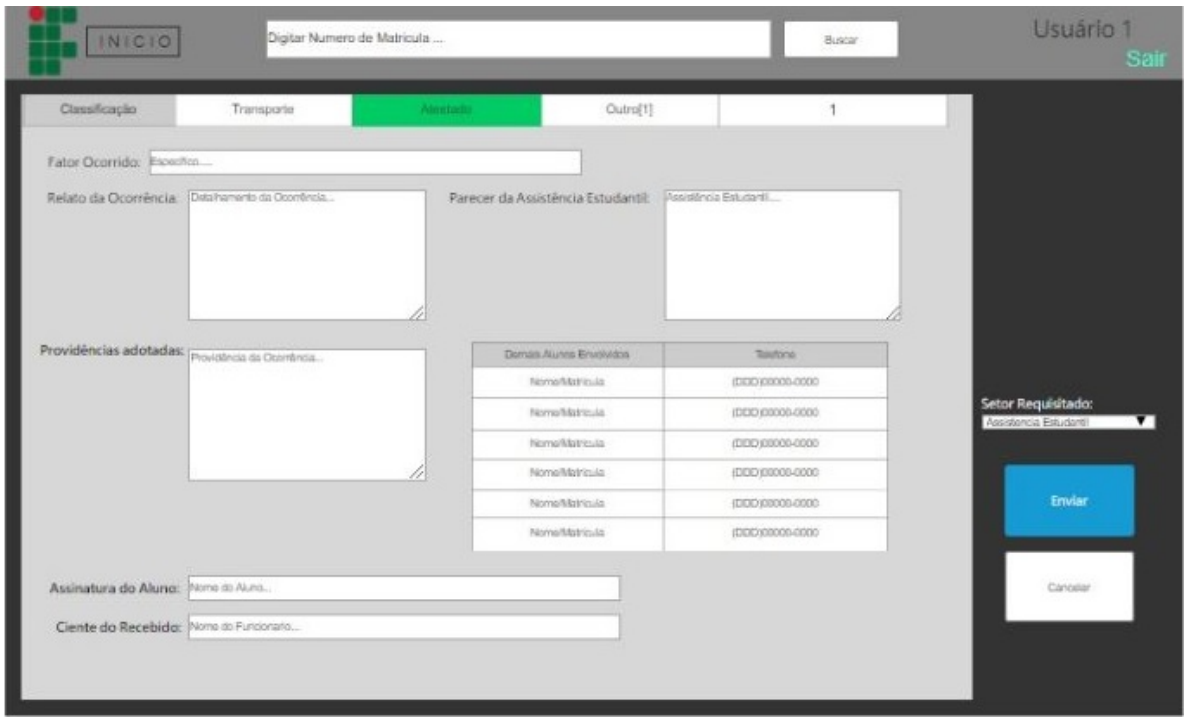
Para uma definição mais clara dos requisitos, é importante destacar a série de entrevistas feitas com a Instituição de Ensino onde buscou-se o maior número de informações possíveis para que o sistema resultante atingisse os objetivos desejados. Além disto, a análise dos protótipos em conjunto com a direção ajudou a esclarecer pontos que não estavam bem claros após as entrevistas e assim validar os protótipos, certificando-se que o sistema atenda as necessidades e expectativas do cliente.

**Figura 4** - Tela de Cadastro de Funcionários




**Fonte:** Equipe do Projeto(2018)

**Figura 5** - Tela de Ocorrências




**Fonte:** Equipe do Projeto(2018)

## CONCLUSÕES

Ao final do presente estudo pode-se concluir que a técnica de Prototipação é fortemente eficaz para validação de requisitos, principalmente quando o cliente não tem clareza na definição dos requisitos do sistema. Por meio de apresentações de telas pode-se oferecer ao cliente um esboço do que será o sistema e fornecer um feedback para assim, fazer alterações

conforme o pretendido., até que se chegue ao protótipo ideal, almejado pelo cliente. Essa técnica aliada à outras metodologias, facilita notoriamente o processo de construção de Software.

## REFERÊNCIAS


GIL, Antonio Carlos. **Como Elaborar um Projeto de Pesquisa**. 6 ed.São Paulo: Atlas, 2008. 200 p.

IMASTERS. **A importância da prototipagem**. Disponível em: <https://imasters.com.br/tecnologia/importancia-da-prototipagem/?trace=1519021197&source=single>. Acesso em: 13 abr. 2018.

LINKEDIN SLIDESHARE. **Prototipagem**. Disponível em: <https://pt.slideshare.net/jwainer/prototipagem>.Acesso em: 16 abr. 2018.

MJV BLOG. **O papel da prototipagem no design thinking**. Disponível em: <http://blog.mjv.com.br/papel-da-prototipagem-no-design-thinking>. Acesso em: 16 abr. 2018.

NASCIMENTO, Thiago. **A importância dos protótipos no desenvolvimento de sistemas.** 2013. Disponível em: <http://thiagonasc.com/desenvolvimento-web/a-importancia-dos-prototipos-no-desenvolvimento-de-sistemas>. Acesso em: 03 maio 2018.

PRESSMAN, Roger S. Software Engineering: a Practitioner's Approach. 6. ed. McGraw-Hill, 2005

SANTOS, Fabio S.; JUNIOR, Custodio Gastão Da Silva. Validação de requisitos através da prototipação de software.. **Connectioline,** [S.L.], v. 1, n. 9, p. 13-14, jan. 2012.

# DESENVOLVIMENTO DE UM SISTEMA PARA PIZZARIAS UTILIZANDO O MÉTODO DE PROTOTIPAÇÃO

Nayanne Carvalho Uchoa,

Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro. nayanneuchoaig@gmail.com.

Joeldo Olinda da Silva,

Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro. joeldo.olinda@gmail.com.

José Danilo Torres Lima,

Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro. danilotl21@gmail.com.

## INTRODUÇÃO

A evolução da tecnologia dos aparelhos celulares permitiu oferecer ao usuário recursos que vão muito além da realização de uma chamada ou envio de mensagem. As melhorias de hardware dos aparelhos celulares permitiram o desenvolvimento de sistemas operacionais mais avançados. Com sistemas operacionais mais avançados, foi possível desenvolver aplicativos melhores, com cada vez mais recursos e serviços ao usuário(SILVA; SANTOS, 2014).

Visando melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de Software, cuja principal função é a de tratar dos aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de software. (FALBO, 2005)

Dentro desta perspectiva da Engenharia de Software, insere-se a prototipagem que apresenta-se como uma técnica que pode ser utilizada na construção de software. O protótipo pode ser construído de diversas formas, como por exemplo, na construção civil onde os engenheiros e arquitetos constroem maquetes com o intuito de reproduzir conceitos visuais e funcionais do projeto. De forma análoga, temos o protótipo de software que pode ser construído usando esboços simples em papel ou ferramentas especializadas.

Seguindo essa tendência, o mercado dessa área está cada vez mais acirrado e, desta forma, necessitando cada vez mais de estudos na área de prototipagem, que é um dos responsáveis pelo desenvolvimento ágil.

Segundo Sommerville (2011) um protótipo é uma versão inicial de um sistema de software, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais informações sobre o problema e suas possíveis soluções. O desenvolvimento rápido e interativo do protótipo é essencial para que os custos sejam controlados e os *stakeholders* do sistema

possam experimentá-lo no início do processo de software, de forma interativa-contributiva (*feedback)*.

A partir destas proposições, foi desenvolvido um protótipo para pizzarias usando o processo de engenharia de software, chamado prototipação, e que tem como objetivo facilitar o entendimento dos requisitos e fazer a simulação das telas do sistema.

## METODOLOGIA

Para o desenvolvimento do presente projeto, foi criada uma equipe de 3(três) pessoas na disciplina de Interface Homem-Computador, cujas atividades utilizaram o processo iterativo denominado prototipagem, com o auxílio da ferramenta Moqups, um aplicativo feito em HTML5, disponível online e com planos gratuitos. Neste aplicativo, foram desenvolvidas telas para a simulação do sistema que será desenvolvido, com o fito de apresentar, ao cliente, como será o sistema, de forma a possibilitar que o mesmo possa validar os requisitos então apresentados.

Para validação dos protótipos foi utilizada uma *checklist*, cujo objetivo é o de promover a interação com o nosso cliente, permitindo, assim, que o mesmo pudesse atestar que todas as etapas e/ou itens da lista foram devidamente cumpridas de acordo com o programado.

Após a criação de todas as telas foi feita uma reunião com os membros da equipe para analisarem se o aplicativo estava certo ou se ainda precisava de alterações. Finalizado a parte de análise começou a fase de teste do aplicativo.

O método de prototipação foi escolhido por ser um método que nos permite aperfeiçoar todas as etapas do processo, garantindo uma boa exploração dos conceitos antes da criação do aplicativo. Os benefícios desse processo são vários entre eles a questão de que podemos testar a ideia para ver se ela realmente atende as necessidades e as expectativas esperadas e por fim seu baixo custo faz com que o projeto seja apresentado ao mercado com um preço acessível.

## RESULTADOS E DISCUSSÕES

Percebeu- se que depois dos resultado obtidos e com o *feedback* do nosso cliente o método escolhido de prototipação mostrou-se eficaz no quesito de entendimento do aplicativo e para a simulação das telas do projeto que será desenvolvido. Durante a fase de criação das telas foi mantido o contato com o cliente para que ele pudesse acompanhar a evolução do projeto e fazer alterações caso fosse necessário, nos primeiros protótipos mostrado ao nosso cliente o mesmo viu a necessidade de adicionar mais uma tela a qual tivesse um carrinho de compras para que todas a pizzas escolhidas pelo cliente fossem adicionadas nesse carrinho, e

só depois ser encerrada a compra, facilitando a compra de várias pizzas ao mesmo tempo e salvando os pedidos já feitos antes de finalizar a compra.

Depois das alterações no protótipo e da nova tela implementada, houve um teste de verificação e validação dos requisitos esse teste foi feito por meio de uma checklist para saber se todos os itens atendia o esperado, após essa etapa houve a conversa final com o cliente a qual foi apresentado os seguintes protótipos:

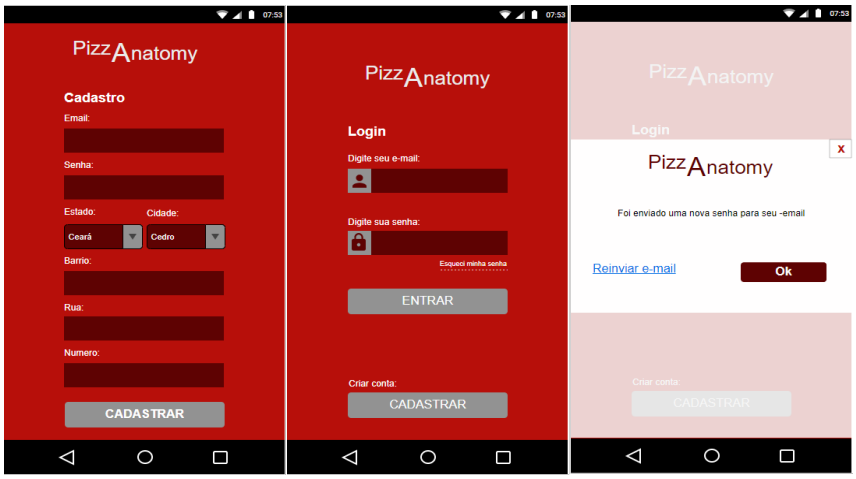



Figura 1 – Simulação das telas

Fonte: Autor

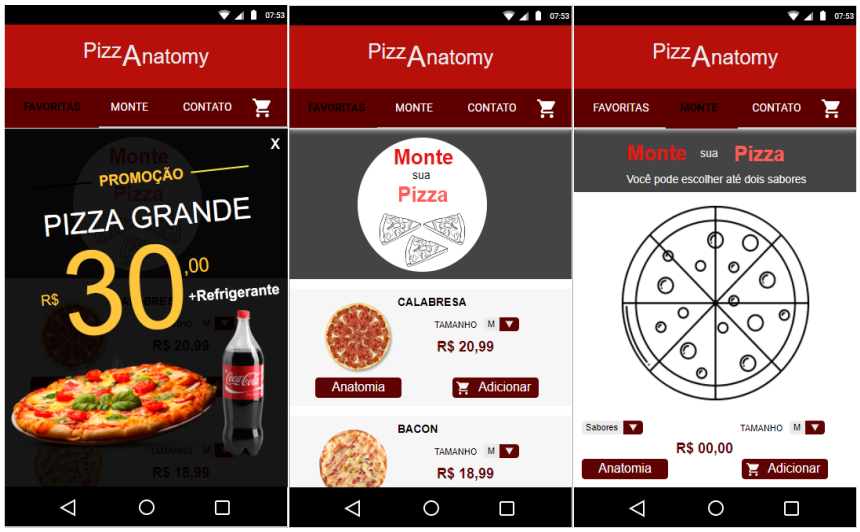



Figura 2 -Simulação das telas

Fonte: Autor

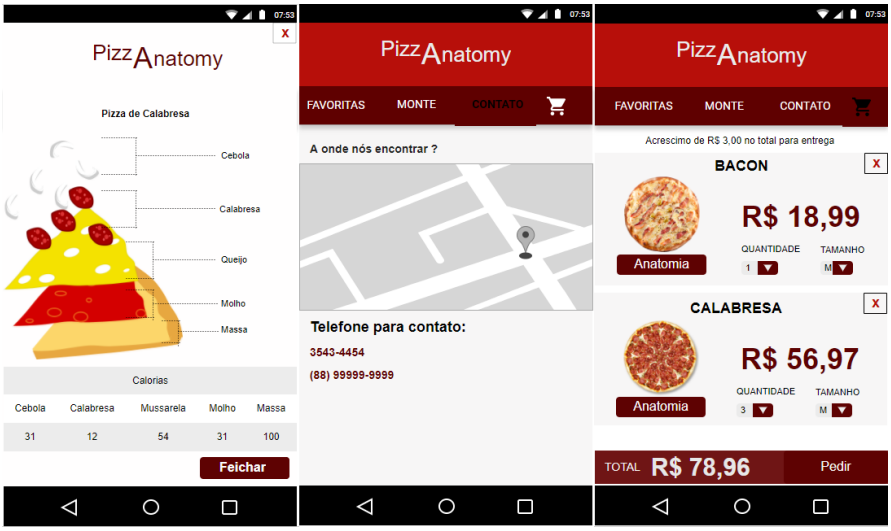



Figura 3 – Simulação das telas

Fonte:Autor

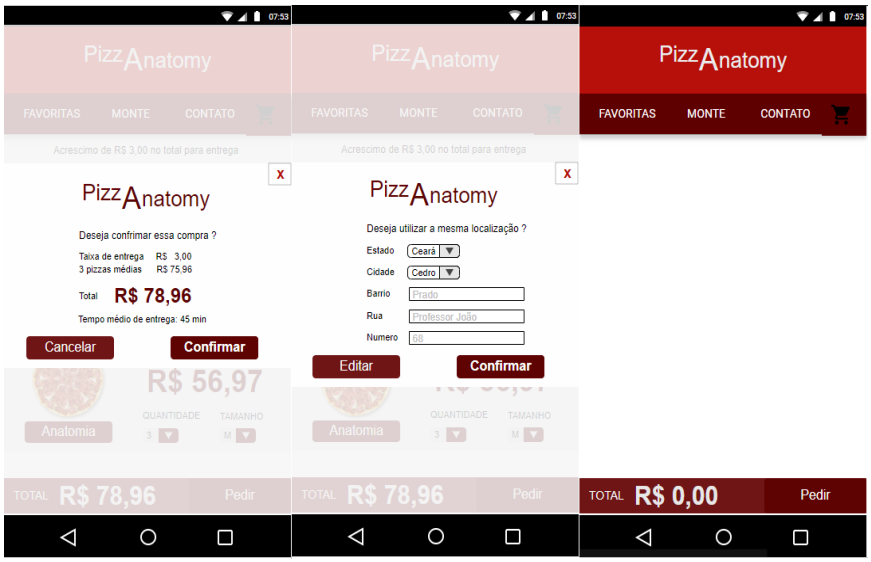



Figura 4 -Simulação das telas

Fonte: Autor

Todas as telas acima foram produzidas a partir do resultado obtido através da análise do *feedback*, obtido da apresentação dos protótipos ao cliente. Com todas as alterações feitas o usuário afirmou, todos os requisitos solicitados foram cumpridos de forma clara e objetiva.

Nesse caso os protótipos não foram uma má influência para o cliente, já que é comum que o mesmo possa a achar que as telas apresentadas seja o sistema completo, apressando a entrega tornando a interação com a equipe negativa.

## CONCLUSÕES

A engenharia de software é um campo que vem ganhando grande destaque no mundo da tecnologia são várias técnicas e processos que tem como propósito de auxiliar o

desenvolvimento da criação do software, essas técnicas visa a organização, produtividade e qualidade do software, ela também traz benefícios para a equipe fazendo com que tenha maior rendimento para produzir e organização no trabalho que está sendo gerado.

Conclui-se que o processo de engenharia de software denominado prototipação tem como objetivo principal a interação do usuário com o projeto dando a ele a visão de como será a versão final, isso faz com que o usufruidor esteja mais presente no desenvolvimento tornando o projeto mais usual.

A prototipação bem efetuada faz com que o sistema tenha maior qualidade e sejam aceito facilmente pelos usuários, já que nesse método a corroboração vem de quem mais entende do sistema, o cliente**.**

Para trabalhos futuros, no intuito de fazer um aplicativo mais completo a equipe está a sugerir ao cliente algumas funções que venham trazer melhorias para o seu sistema como por exemplo a criação de uma tela para bebidas, uma função para retirada de ingredientes da pizza e outra para que o usuário possa escolher caso queira bordas recheadas tornando assim uma aplicação mais completa

## REFERÊNCIAS

FALBO, Ricardo Engenharia de Software, 2005. Disponivel em: <https://inf.ufes.br/~falbo/download/aulas/es-g/2005-1/NotasDeAula.pdf>. Acesso em 25 abr 2018.

SILVA, Marcelo, SANTOS, Marilde . Os Paradigmas para desenvolvimentos de Aplicativos para Aparelhos Celulares, 2014. Disponivel em: <http://revistatis.dc.ufscar.br/index.php/revista/article/view/86/80> Acesso em: 25 abr 2018.

SOMMERVILLE, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson, 2011.

# ENGENHARIA DE SOFTWARE APLICADA A VALIDAÇÃO DE REQUISITOS NO DESENVOLVIMENTO DE UM SOFTWARE PARA UM FRIGORÍFICO

Francisco Alves Pereira,
Discente do curso de Sistemas de Informação. IFCE- Campus Cedro.
francisco7ifce@gmail.com.

José Wellington da Costa Bezerra,
Discente do curso de Sistemas de Informação. IFCE- Campus Cedro.
Wellingtoncedro2011@gmail.com.

Igor Lima Facundo Xavier,
Discente do curso de Sistemas de Informação. IFCE- Campus Cedro.
mail.igorlima@gmail.com.

Luiz Felipe Bezerra Souza Luz,
Discente do curso de Sistemas de Informação. IFCE- Campus Cedro.
luizktn29@gmail.com.

## INTRODUÇÃO

É notório o crescimento do desenvolvimento de software, pelo papel importante que o mesmo exerce na sociedade contemporânea. Esse crescimento é decorrente do uso cada vez mais contínuo de computadores pessoais e das inúmeras áreas do conhecimento humano, o que tem propiciado uma procura constante por soluções que dinamizam as atividades diárias. Neste meio, surge a engenharia de software que nos oferece mecanismos sistemáticos embasados na utilização correta de técnicas e ferramentas, tomando como base o problema a ser solucionado, os recursos que se tem à disposição para suprir as necessidades dos clientes.

Falbo (2005), afirma que a Engenharia de Software possui o objetivo de melhorar continuamente a qualidade dos softwares em termos gerais e intensificar a produtividade no desenvolvimento de soluções. Pressman (2011) também afirma que a engenharia de software traz um conjunto de programas, feito para atender a outros programas. Para tanto, este universo surge com a incumbência de designar práticas e técnicas para a implementação de softwares abarcando um grande leque de aplicações e diversos tipos de dispositivos, como por exemplo sistemas de informações corporativos, portais, Web e aplicações mobile.

Diante do exposto, pode-se perceber alguns problemas que podem ser solucionados com a engenharia de software no frigorífico X, localizado na cidade de Iguatu-CE, que atua com vendas de carnes, cereais, verduras e bebidas, apresentando um grande fluxo de clientes. Com a necessidade de automatizar os processos de vendas e ficar de acordo com as leis governamentais da região o frigorífico adquiriu um sistema de gerenciamento de caixa, o qual futuramente percebeu-se que não atendia completamente as suas necessidades, pois além da dificuldade de utilização por parte dos funcionários locais, devido a interfaces complicadas, o sistema não oferecia funcionalidades importantes, como o gerenciamento de contas a pagar e a falta de um portfólio para a divulgação.

Pela insatisfação com o sistema, o frigorífico continuou a realizar todos os processos rotineiros e pertinentes ao estabelecimento, manualmente. O que ocasionava a demora no atendimento dos clientes, falta de controle no estoque, o recorrente desatendimento com as contas a serem pagas e a insciência do quanto foi vendido.

Este trabalho tem como objetivo apresentar a validação de requisitos de um sistema WEB, que gerencie os processos comerciais e internos, inerentes ao frigorífico, através do método ágil da engenharia de software, a prototipação de telas, *wireframe*.

## METODOLOGIA

A proposta de solução para os problemas do frigorífico X, foi uma iniciativa dos alunos do 7º semestre do curso de sistemas de informação do Instituto Federal de Educação, Ciência e tecnologia do Ceará-IFCE, com a finalidade de obter aprovação nas disciplinas de projeto integrador II e laboratório de desenvolvimento de sistemas.

Para tal, a equipe portou-se como uma fábrica de software denominada CajuWeb. Com cargos de gerente, analista de sistemas e desenvolvedores, cada integrante da equipe recebeu um ou mais papéis por meio de eleição.

De acordo com Sommerville (2012), é de extrema importância obter o conhecimento e gerenciar corretamente a especificação e os requisitos de software, é preciso entender e compreender qual a real demanda dos usuários e fazer o gerenciamento de suas expectativas para que o sistema entregue seja útil e esteja dentro do orçamento e cronograma estipulado. Dessa forma a efetivação dos trabalhos deu-se com a realização de uma entrevista ao proprietário do frigorífico X, com o intuito de conhecer a realidade da empresa, bem como todos os processos que ocorrem internamente. Após a entrevista a equipe de desenvolvimento reuniu-se, com o intuito de extrair requisitos para implementação do sistema. Com os requisitos elaborados, necessitou-se realizar uma reunião para revisão dos mesmos, entre a equipe da CajuWeb, e decidir como esses requisitos seriam validados, visto que a distância entre o cliente

e a fábrica de software, dificultava a comunicação. Como decisão unânime, a validação foi realizada por meio da prototipação de telas, utilizando o método ágil *wireframe* com o auxílio da ferramenta Balsamiq Mockups3. Cada requisito elencado foi representado por meio de elementos de uma página Web, definindo a hierarquia entre eles, agrupando-os de acordo com suas importâncias relativas, onde o objetivo central foi de fazer a comunicação do conteúdo e disponibilizar em cada página orientações de uso e base para o desenvolvimento da aplicação, como apresentado na tela de vendas na figura 1:

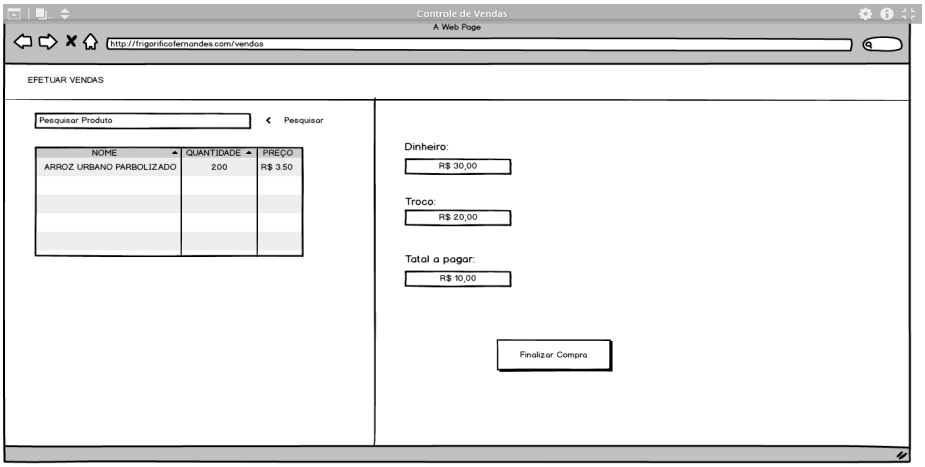



**Figura 1:** protótipo do requisito de vendas, Autor, 2018.

Todos os protótipos de tela foram enviados para o cliente através do aplicativo de mensagens, Whatsapp, onde cada imagem possui sua respectiva descrição. Posteriormente, foi recebido um feedback por parte do cliente, onde o mesmo pôde melhor compreender como seria o funcionamento do sistema, e por sua compreensão, sugeriu melhorias com relação às funções que seriam disponibilizadas no sistema de informação, como por exemplo, a criação de novos botões ou a posição onde algumas informações seriam disponibilizadas.

## RESULTADOS E DISCUSSÕES

Nota-se que adoção da prototipação de telas no processo de validação dos requisitos mostrou-se eficiente, pois o mesmo teve uma visão aprimorada de como o sistema ficaria ao ser concluído e compreendendo as dimensões do seu problema propôs soluções que outrora não tinha sido identificada na entrevista. Santos (2014) também corrobora com essa ideia ao afirmar que o protótipo é a primeira versão desenvolvida de um sistema a qual tem a finalidade de abordar a questão de interface com o usuário, validar requisitos e apresentar a viabilidade do sistema ou seja é uma ferramenta indispensável na comunicação entre desenvolvedores e clientes. Durante a criação do protótipo de software, pode perceber a eficácia interação entre clientes e desenvolvedores, o que facilita o levantamento, validação de requisitos e funcionalidades do sistema. Santos(2014) ainda afirma que a prototipação é uma representação

visual do produto de software, através da simulação das telas da aplicação a qual permite ao usuário visualizar a aplicação que ainda não foi produzida e entender o que será feito pela equipe de desenvolvimento. As melhorias foram implementadas e são mostradas a seguir a tela final de vendas do sistema proposto:

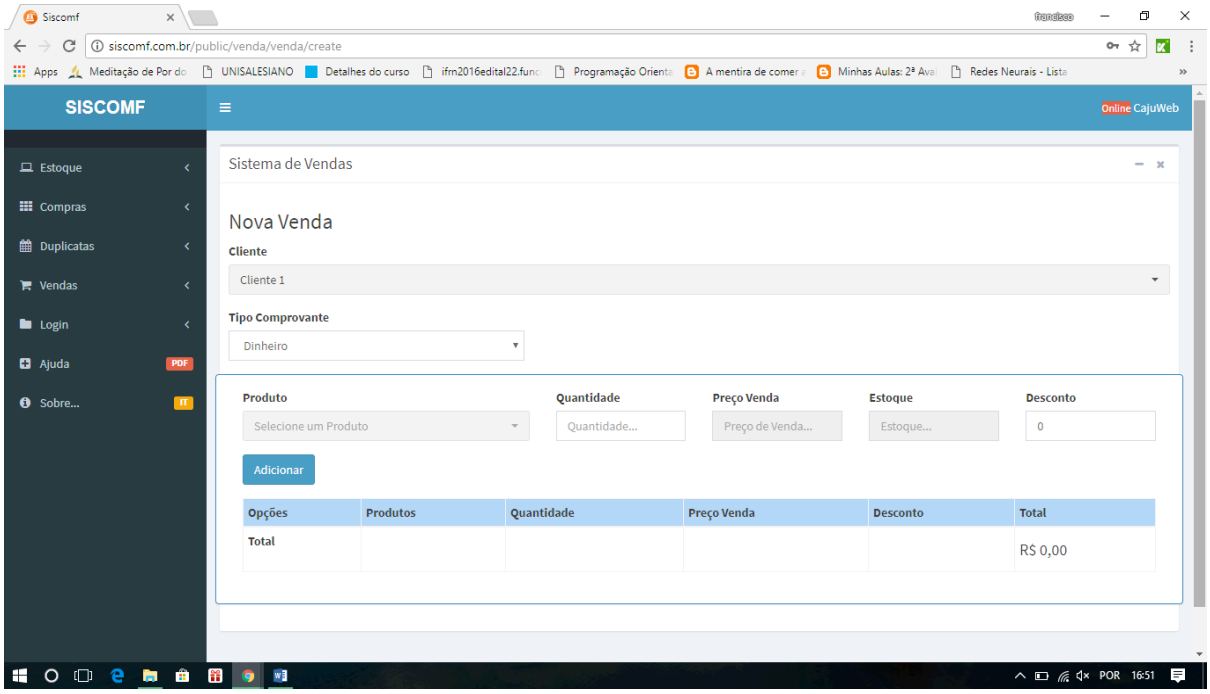



**Figura 2:** protótipo dos requisitos de vendas com alterações solicitadas pelo cliente, Autor, 2018.

A prototipação de telas facilitou o entendimento do propósito do software que se desenvolveu, por ser um grande aliado dos processos metodológicos ágeis, proporcionado uma sinergia entre a equipe de desenvolvimento e o cliente. Engholm (2010), afirma que a prototipação é uma solução extremamente útil e importante para validação de requisitos, pois servem para demonstrar o sistema, como será a navegação entre as páginas, reproduzindo o comportamento do futuro sistema a ser implementado. Assim, conseguiu-se propor uma solução apropriada, agregando valor a aplicação.

## CONCLUSÕES

Em meio aos benefícios já apresentados, é importante destacar que para haver sinergia entre uma equipe de desenvolvimento e seus clientes no processo de desenvolvimento de software é imprescindível, a comunicação. Não é tarefa fácil, pois grande parte dos projetos de software falham no ato da comunicação. Assim, unir todos os envolvidos de um projeto de software, torna-se fator chave para conseguir melhorar continuamente e, em consequência disso, lograr êxito no desenvolvimento das aplicações.

A prototipação *wireframe* contribuiu efetivamente para o propósito central da comunicação de clientes e empresa, conforme apresentado nos resultados, conclui-se que o método de validação e aprovação dos requisitos por meio de protótipos é eficiente. Principalmente nos casos, onde os únicos contatos possíveis com o cliente são por meios eletrônicos.

Recomenda-se, como estudos futuros, a utilização da engenharia de software em todo processo de desenvolvimento da aplicação, desde a entrevista para o levantamento de requisitos até a entrega do sistema ao cliente.

## REFERÊNCIAS

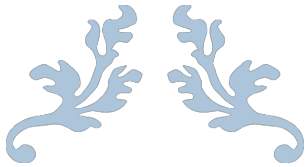

ENGHOLM Júnior, Hélio. **Engenharia de software na prática**. São Paulo. 2010.

FALBO, Ricardo de Almeida. **Engenharia de Software.** 2005. TCC

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional**. 7ª Edição. Ed: McGraw Hill, 2011.

SANTOS, Fabio S. **Validação de requisitos através da prototipação de software**. connection line, n. 9, 2014.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. 2012.



# MÉTRICAS E MEDIÇÕES DA ENGENHARIA DE SOFTWARE

# ANÁLISE ESTÁTICA EM UM SISTEMA DE ROTEIRIZAÇÃO DE VEÍCULOS

*Matheus de Souza Oliveira[2]; [2]Matheus Diógenes Andrade[3].*

## INTRODUÇÃO

Considerando a impossibilidade de garantir a qualidade de um produto de *software* sem técnicas que ajudem a prevenir, encontrar e corrigir defeitos, não se pode inserir qualidade em um produto mal construído apenas com teste, mas também não é possível construir um produto de qualidade sem testes (PEZZÈ; YOUNG, 2008). Contudo, as técnicas para mitigar a quantidade de defeitos de um sistema vão muito além de testar, e para isso é importante destacar conceitos como o de verificação estática e dinâmica que fazem parte do processo de V&V de uma empresa de desenvolvimento de *software*.

Segundo Delamaro et al.(2017), técnicas estáticas são aquelas que não requerem a execução ou mesmo a existência de um programa ou modelo executável para serem conduzidas. Já a verificação dinâmica para Delamaro et al.(2017), técnicas dinâmicas são aquelas que se baseiam na execução de um programa ou de um modelo.

Ainda nesta mesma linha de considerações, vale ressaltar que, vários estudos e experimentos mostram que as inspeções são mais eficientes na descoberta de defeitos do que os testes, segundo Sommerville (2010), inspeções conseguem considerar atributos de qualidade de um programa que nem mesmos são requisitos funcionais, como a conformidade com padrões, portabilidade e manutenibilidade, encontrando ineficiências, algoritmos inadequados e um estilo de programação pobre que poderiam tornar o código de difícil manutenção e atualização.

De acordo com Medeiros (2017):

> A análise estática permite que diversos erros sejam encontrados antes mesmo que o programa tenha que ser compilado (somente válido para as ferramentas que analisam somente o código fonte). [...] A análise estática pode encontrar problemas independentemente da entrada e da saída do programa, enquanto que a análise dinâmica pode encontrar problemas de codificação que não foram ainda informados como um padrão de erro nas ferramentas de análise estática.

O escopo do estudo foca na técnica de verificação estática com apoio de ferramentas automatizadas de inspeção do código-fonte. Esse trabalho teve como finalidade realizar um estudo de caso das inspeções automatizadas de código-fonte em um Sistema de Roteirização de

[2]Discente do curso de Engenharia de Software. UFC Campus Russas. matheusoliveira2496@gmail.com
[3]Discente do curso de Engenharia de Software. UFC Campus Russas. matheusdiogenesandrade@gmail.com

Veículos (SRV), onde não se utiliza a maioria das técnicas de V&V no seu desenvolvimento. Para atingir esse objetivo o estudo buscou:

Aplicar ferramentas de verificação estática automatizada de código-fonte;

Realizar refatorações indicadas pelas ferramentas;

Propor e utilizar métricas para avaliar a eficiência das refatorações;

Analisar os resultados.

A escolha de um estudo de caso é de suma importância para qualquer análise empírica em engenharia de *software* (ROJAS et al., 2016). Segundo Wiklund et al.(2017), após 2004, houve um aumento de pesquisas nas áreas de engenharia e testes de *software* que provavelmente pode ser explicado por um interesse crescente em estudos empíricos nessas áreas. Portanto, julga-se necessário difundir princípios e conceitos relacionados à aplicação de técnicas e critérios de teste, a fim de possibilitar o desenvolvimento de *softwares* de qualidade (COSTA JÚNIOR et al., 2016).

## METODOLOGIA

A metodologia utilizada foi o estudo de caso fundamentando-se em pesquisas bibliográficas e seguindo estratégias de investigação direta. Tem caráter descritivo e natureza qualitativa e quantitativa. A partir dos dados quantitativos gerados pela aplicação das ferramentas, os aspectos particulares foram identificados e correlacionados para sintetizar uma relação entre eles e a melhoria na qualidade do *software*.

Essa pesquisa conduz um estudo de caso em um SRV observando a aplicação de quatro ferramentas de análise estática e quantificando os resultados após as refatorações indicadas por elas.

O processo metodológico iniciou-se com as pesquisas bibliográficas e definição dos objetivos. Com isso, as ferramentas adequadas para o sistema foram selecionadas. Tomando como base os objetivos e as pesquisas bibliográficas, foram definidas as métricas que seriam utilizadas, a aplicação das ferramentas, refatorações e coleta de dados. Posteriormente, foram realizadas as análises e comparações das métricas verificando os indicadores no início e fim do estudo.

As verificações feitas pelas ferramentas selecionadas se concentram em identificar principalmente estilos, boas práticas, *bugs* e código lixo. Abaixo, estão listadas essas verificações e qual ferramenta será responsável por identificar cada tipo de problema que pode ser melhorados no SRV.

- Verificação de estilos: Checkstyle;
- Verificação de boas práticas: PMD;

- Verificação de *bugs* e vulnerabilidades: Findbugs.
- Identificação de métodos e classes não utilizadas: UCDetector.

## RESULTADOS E DISCUSSÕES

Segundo Wilkerson et al. (2012), embora qualquer artefato de software possa ser inspecionado, a maior parte das pesquisas sobre inspeção de software aborda como foco a inspeção do código-fonte. Neste estudo, limitamos as inspeções à análise estática com apoio de ferramentas como base para refatorações e melhorias no *software* e nos referimos a elas como inspeções de código.

A escolha das ferramentas se deu tanto pela linguagem utilizada pelo sistema, como também por elas poderem ser utilizadas como *plugin* da plataforma de desenvolvimento Eclipse. Por mais que apresentem algumas verificações em comum, as ferramentas fornecem muitas outras que são distintas, dispondo da flexibilidade de configurar o que vai ser checado, e podendo até criar regras de verificações de acordo as necessidades do desenvolvedor.

Através dos dados que são gerados pelas ferramentas, foi possível definir métricas para medir a qualidade do *software*, como também identificar indicadores de melhoria, através destas que estão listadas abaixo:

Quantidade de Defeitos Encontradas no Produto (QDP): A quantidade de defeitos de um sistema em produção é uma métrica muito importante para mostrar a efetividade das tarefas de teste e correções de erros.

Defect Reduction Efficiency ou Eficiência de Redução de Defeitos (DRDE): A eficiência de redução de defeitos é uma métrica orientada a função, proposta neste trabalho para fornecer uma medida da capacidade da equipe de reduzir defeitos. É calculado a partir da quantidade de defeitos reduzidos pela equipe dividido pelo número de defeitos encontrados na primeira medição, resultando em um percentual onde quanto mais alto maior a eficiência. A fórmula é dada por:

$QDP_0$: Valor da métrica QDP na primeira medição, quantidade inicial de erros encontrados no produto;

QDP: Valor da métrica QDP na medição atual, quantidade de erros encontrados no produto no momento da medição.

$$DRDE = \frac{QDP_0 - QDP}{QDP_0}$$

O objetivo da medição de *software* é quantificar atributos de um processo ou produto de *software.* Assim com a comparação dos dados é possível analisar a qualidade do *software*. Possibilita ainda mensurar se adoção de novas ferramentas e/ou metodologias estão sendo

positivas ou não para os processos de desenvolvimento do sistema. Com isso, as medições são realizadas antes e depois da mudança a fim de verificar se a mesma foi positiva ou não para a organização (SOMMERVILLE, 2008). Alguns exemplos são a medição do tamanho pela quantidade de linhas de código, grau de facilidade de leitura de trechos de código, número de pessoas-dia necessárias para desenvolver um *software* e número de defeitos reportados (SOMMERVILLE, 2008).

Para utilização dessas métricas, foi criado um processo para coleta de dados quantitativos gerados pelas ferramentas. Esse processo tem o objetivo de registrar todas as informações geradas pelas ferramentas, capturando o estado do sistema antes e depois de cada conjunto de refatorações/correções indicadas por elas. O processo de coleta de dados é apresentado na Figura 1.

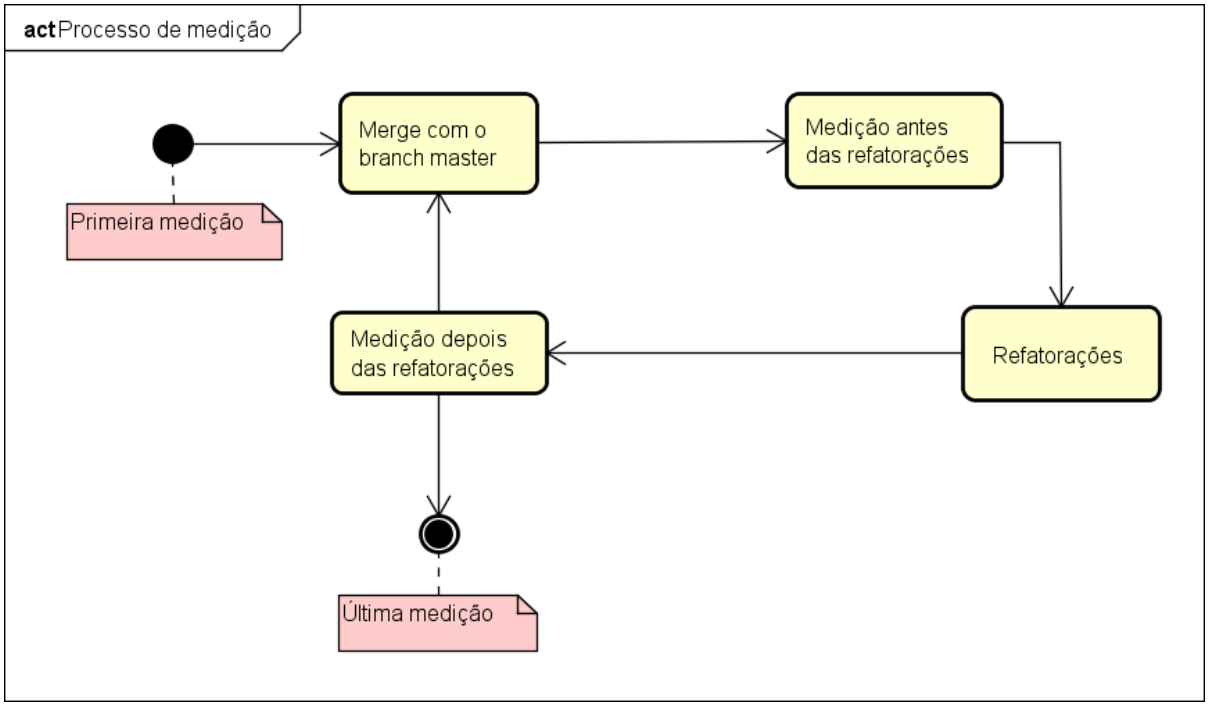



Figura 1: Processo de medição dos indicadores.

Fonte: Elaborada pelo autor.

Antes da primeira refatoração no código-fonte, as ferramentas foram aplicadas e registrados todos os dados resultantes dessa aplicação sobre os erros que foram identificados no sistema. Após isso, foram feitas as refatorações e registros do progresso em mitigar os erros identificados pelas ferramentas, sempre antes e depois de cada ciclo de refatorações. Todas as refatorações foram realizadas em um *branch* criado especificamente para isso, sendo esse uma cópia do repositório principal onde as alterações desenvolvidas pela equipe são incorporadas. Assim antes de cada conjunto de refatorações era feito um *merge* do repositório principal para o *branch* das refatorações, incorporando as novas mudanças e logo após realizando um novo registro das métricas. Assim foi possível comparar o sistema no início e fim do projeto,

relacionando a diminuição desses erros através das métricas definidas, com o salto na qualidade do *software*.

Para analisar os dados quantitativos coletados a partir dos erros identificados pelas ferramentas de análise estática, é preciso levar em consideração o declínio de alguns indicadores e o aumento de outros que estão relacionados com as métricas definidas, a cada medição realizada ao longo das refatorações feitas no SRV.

Realizando uma análise sobre as métricas utilizadas no estudo, é possível ver uma forte relação no comportamento dos indicadores de qualidade, a DRDE e a QDP. Assim, é possível ver essa relação no gráfico apresentado na Figura 2.

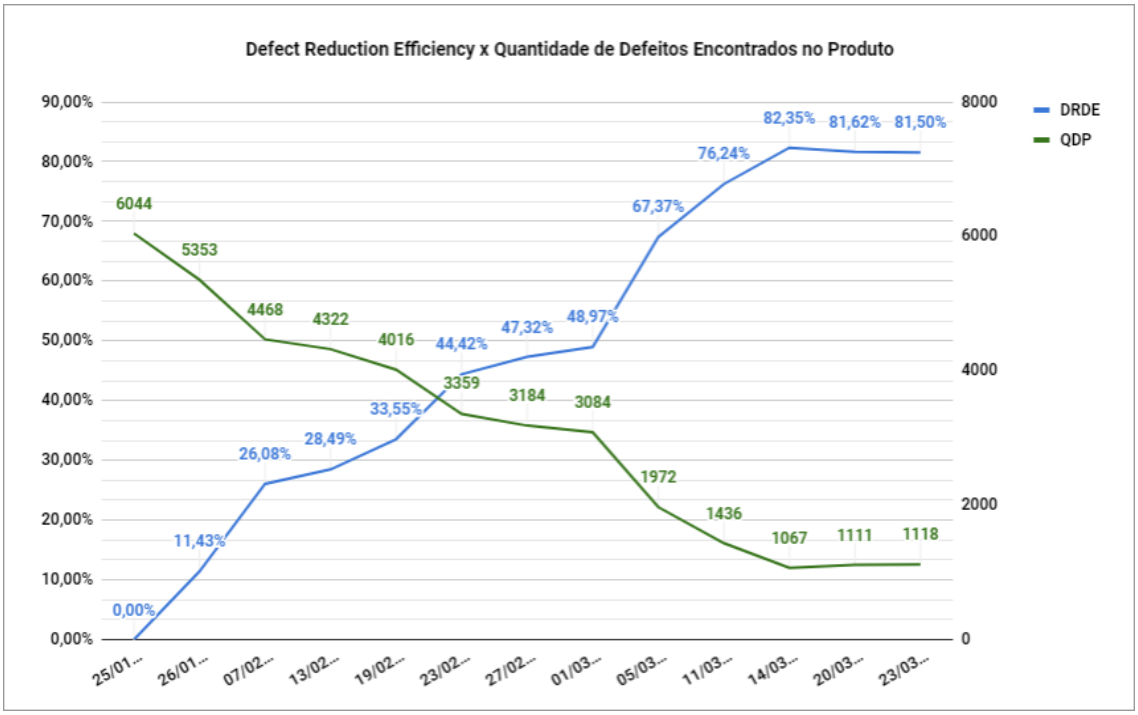



Figura 2: Eficiência de redução de defeitos x Quantidade de defeitos encontrados no produto.

Fonte: Elaborada pelo autor.

A partir do gráfico é fácil ver que quanto mais a quantidade de defeitos diminui, maior é a DRDE, de forma análoga, quanto mais eficiente for o esforço da equipe para remover os defeitos identificados, menos erros vão ser encontrados no produto. Contudo, mesmo que na teoria a QDP ideal seja zero, na prática é muito difícil isso acontecer, principalmente quando não existe um processo para garantir que esses defeitos sejam corrigidos antes da alteração ser consolidada e incorporada ao repositório do sistema.

Após visualizar as últimas datas de refatorações, percebe-se que quanto mais a QDP diminui, há uma tendência da quantidade de defeitos manter-se baixa, mas sem grandes variações. Isso acontece pelo fato de que os erros restantes possuem uma complexidade maior de resolução, tornando mais difícil eliminar todos os defeitos com novos sendo inseridos constantemente. Para isso, seria necessário que os erros fossem eliminados antes mesmo do

código que estão inseridos ser incorporado ao repositório do *software*. Assim, tanto nas correções de problemas no sistema quanto no desenvolvimento de novas funcionalidades, pode ser utilizado um processo para que toda alteração no código-fonte tenha que ser verificada e validada antes de se consolidar. Com isso, toda alteração feita no sistema já seria incorporada sem os erros que foram verificados pelo próprio desenvolvedor, diminuindo o esforço de refatorações com o propósito de remover defeitos que já estão inseridos no produto.

## CONCLUSÕES

Considerando as leituras realizadas para elaboração deste trabalho, pode-se tomar conhecimento que a preocupação com a qualidade de *software* tem se tornado cada vez maior em função do aumento na utilização de *softwares* e das exigências dos usuários que buscam antes de tudo sistemas de confiança, eficazes, eficientes e de boa qualidade.

Este trabalho apresentou diferentes ferramentas de análise estática de código-fonte que podem ser utilizadas de forma complementar para se obter a melhoria de qualidade de *software*. Como também, foram propostas métricas de teste de *software* que podem ser utilizadas para controle de processo de qualidade das correções de erros.

Durante o estudo foi utilizada como técnica de inspeção de *software* a análise estática de código com apoio de quatro ferramentas. As ferramentas de análise estática mostraram-se eficientes na remoção de defeitos para melhoria do entendimento do código-fonte, atendendo bem as expectativas para esse propósito.

Através das análises realizadas sobre os dados quantitativos foi possível perceber a melhoria dos indicadores de qualidade com o declínio dos defeitos identificados pelas ferramentas. Contudo, também foi possível perceber através de observações empíricas a melhoria do código principalmente no que se diz respeito a manutenibilidade do sistema, mas também na confiabilidade com o uso de boas práticas de codificação que tornam o código menos suscetível a erros.

## REFERÊNCIAS


COSTA JÚNIOR, M.; ANDRADE, S.; DELAMARO, M. Automatização de teste de software com ênfase em teste de unidade. In: AIRES, K. R. T.; MOURA, R. S. (Ed.). **II Escola Regional de Informática do Piauí.** Teresina: Sociedade Brasileira de Computação – SBC, 2016. cap. 6,p. 121–143.

DELAMARO, M.; JINO, M.; MALDONADO, J. **Introdução ao teste de software**. 2. ed. São Paulo: Elsevier Brasil, 2017.

MEDEIROS, J. E. R. de. **Estudo Comparativo de Ferramentas de Análise Estática de Código**. Dissertação (Graduação em Engenharia de Software) — Departamento de Informática

e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Natal, nov 2017.

PEZZÈ, M.; YOUNG, M. **Teste e análise de software: processos, princípios e técnicas**. 1. ed. Porto Alegre: Bookman Editora, 2008.

ROJAS, J. M.; FRASER, G.; ARCURI, A. Seeding strategies in search-based unit test generation. **Software Testing, Verification and Reliability**, v. 26, n. 5, p. 366–401, 2016.

SOMMERVILLE, I. **Engenharia de Software**. 8 ed. Rio de Janeiro: Prentice-Hall, 2008.

SOMMERVILLE, I. **Software Engineering**. 9. ed. New York: Addison Wesley, 2010.

WILKERSON, J. W.; NUNAMAKER, J. F.; MERCER, R. Comparing the defect reduction benefits of code inspection and test-driven development. **IEEE Transactions on Software Engineering**, v. 38, n. 3, p. 547–560, May 2012.

WIKLUND, K.; ELDH, S.; SUNDMARK, D.; LUNDQVIST, K. Impediments for software test automation: A systematic literature review. **Software Testing, Verification and Reliability**, v. 27, n. 8, 2017.

YIN, R. K. **Estudo de Caso: Planejamento e Métodos.** 2. ed. São Paulo: Bookman editora,2001.

# APLICAÇÃO DE FERRAMENTAS E TÉCNICAS DA ENGENHARIA DE SOFTWARE NO DESENVOLVIMENTO DE UM SISTEMA ESCOLAR PARA WEB

*Gabriela Nayara Duarte Oliveira Damazio*[4]*; Daniel Santos da Silva*[5]*;*
*Expedito Alves de Lima*[6]*.*

## INTRODUÇÃO

Com a internet cada vez mais ao alcance de todos, as formas de comunicação se tornaram mais acessíveis e o desenvolvimento de atividades diárias, sem locais e horários fixos, de forma automática, rápida e também acessível são primordiais. Com isso, o desenvolvimento de sistemas para Web vem crescendo constantemente, tornando-se um meio promissor para a prestação de serviços computacionais, atingindo um número cada vez maior e diversificado de usuários e/ou clientes, eles sendo ou não empresariais (DIAS e *et al.*, 2010).

Como cada cliente ou usuário possui uma necessidade com complexidade diferente a ser atendida, o processo de desenvolvimento de software deve ser implementado de maneira distinta. Desta forma, a engenharia de software é responsável por caracterizar um conjunto de atividades, a partir de 3 elementos fundamentais (métodos, ferramentas e procedimentos), de forma a desenvolver de maneira prática, ordenada e medida, softwares que respeitem prazos, orçamentos e necessidades especificas, com aspectos que possam ser facilmente gerenciados (REZENDE, 2005).

Em outros termos, segundo Sommerville (2011) a engenharia de software tem por objetivo incluir técnicas que apoiam especificação, projeto e evolução de programas tratando-se de toda a documentação associada e dados de configurações necessários para fazer com que o software opere corretamente, incluindo também atividades de gerenciamento de projeto de software e desenvolvimento de ferramentas, métodos e teorias para apoiar a produção de software, não limitando-se apenas com os processos técnicos do desenvolvimento.

Uma das ferramentas da engenharia de software é a métrica de software, que segundo Carvalho, Chiossi e Drach (2006) é uma poderosa ferramenta gerencial quantitativa de medida de vários aspectos de um sistema de software, contribuindo para estimativa de custos de maneira precisa através da elaboração de estimativas de prazo para o estabelecimento de metas plausíveis, facilitando assim o processo de tomada de decisões e a posterior obtenção de

[4] Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro. E-mail: gabrielanayara10@gmail.com
[5] Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro. E-mail: daniel.ifce2@gmail.com
[6] Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Cedro. E-mail: expeditoalves2016@gmail.com

medidas de produtividade e qualidade. Ainda segundo o autor a métrica mais utilizada é a análise de por pontos de função.

Apesar de softwares oferecerem facilidade e praticidade, muitas empresas ou instituições (particulares ou públicas) não fazem uso de tais tecnologias. Como é o caso da Escola Amiguinhos de Infância, escola do setor privado, que atende alunos, desde o Maternal até o 9º ano do ensino fundamental, e encontra-se localizada na cidade de Cedro - Ceará. Durante a entrevista feita a empresa, foram identificados problemas com relação a elaboração de histórico escolar, ficha individual e boletim do aluno. Todas essas atividades são feitas manualmente pela secretária, coordenadora pedagógica e diretor(a).

Atualmente o histórico escolar não segue um modelo de documento satisfatório as necessidades da escola, pois há muitos campos desatualizados, e as informações estão organizadas de forma que dificultam o preenchimento. Já as fichas individuais (que são documentos enviados anualmente a CREDE 17) e boletins dos alunos (que são documentos entregues aos pais dos alunos a cada bimestre), são preenchidos de forma manuscrita pelos gestores da escola, assim os cálculos das médias são elaborados com o auxílio de uma calculadora, e em caso de erro ou rasura o documento deve ser refeito.

Diante de tal fato, está sendo implementado um sistema com utilização de técnicas e ferramentas de engenharia de software, e neste trabalho pretende-se apresentar em destaque as derivações que as métricas por pontos de função proporcionou a uma fábrica de software acadêmica fundada no curso de sistemas de informação do Instituto Federal de Educação Ciência e Tecnologia Campus Cedro.

## METODOLOGIA

Para desenvolvimento do projeto proposto, foi criada uma empresa de desenvolvimento de software na disciplina de Laboratório de software, orientado pelo professor responsável pela disciplina. Empresa essa composta por uma equipe com três integrantes, cada qual com dois papéis específicos organizados da seguinte maneira, conforme a Tabela 1.

**Tabela 1:** Integrantes e seus respectivos papéis

| Equipe | Papéis |
|---|---|
| Integrante 1 | Gerente de desenvolvimento e analista de requisitos |
| Integrante 2 | Analista de requisitos e Analista de testes |
| Integrante 3 | Arquiteto e Desenvolvedor |

Fonte: Desenvolvido pelos autores

Dessa forma, para gerenciamento do processo do software implementado, utilizaram-se as metodologias ágeis, sendo o framework *Scrum* para gerenciamento do projeto, e para o processo de desenvolvimensegunto o XP (*Extreme Programming*).

Métodos ágeis tem por objetivo desenvolver softwares com qualidade, focando nos indivíduos e interações, funcionamento dos softwares, comunicação constante com o cliente, e adaptação a mudanças (PRIKLADNICKI; WILLI; MILANI, 2014). Segundo Sato (2007), os métodos ágeis permitem diminuir custos, pois em comparação com os métodos tradicionais, diminuem o excesso de documentação e burocracias, focando em atividades que trazem valor imediato na criação de software com qualidade.

Com a utilização dos métodos ágeis pretendeu-se desenvolver o sistema dentro de um curto espaço de tempo, de forma a atender as necessidades do cliente de acordo com a disponibilidade da equipe nesse período.

Como supracitado, *Scrum* é um método ágil para gerenciamento de projetos. De acordo com Da Silva e Lovato (2016) o *Scrum* baseia-se no princípio da objetividade, papeis bem definidos e facilidade de aprendizado, dando visibilidade aos problemas e servindo como um guia para construção das soluções.

No desenvolvimento de um projeto *Scrum*, deve-se obedecer as seguintes etapas (LEITE e LUCREDIO, 2014):

- *Product Owner*: identificar as necessidades dos clientes, com a criação de uma lista única e ordenada do trabalho a ser feito (escopo) para garantir a entrega do produto do projeto, denominada *Product Backlog*. Nessa lista são registrados todos os requisitos, funcionalidade e características que irão compor o produto a ser entregue.
- Essa lista é gerida pelo *Product Owner* e são avaliadas em prol de entregar o maior valor ao cliente do projeto no menor espaço de tempo possível.
- *Sprint*: A partir do escopo do *Product Backlog*, priorizados pelo *Product Owner* a *Sprint* deve ser definida por duração fixa e constante, onde o incremento do produto é gerado pela equipe de desenvolvimento. As *Sprints* devem haver reuniões de planejamento (*Sprint planning*), objetivo claro (quais os Product Backlog serão entregues), processo de desenvolvimento a ser utilizado (ciclo de vida) entregas frequentes de funcionalidades e requisitos desenvolvidos no período, e uma reunião ao final de cada Sprint para identificação dos pontos positivos e planejamento das estratégias para melhorar os pontos negativos.

A primeira atividade da equipe, após definir o framework de gerenciamento, foi fazer o levantamento de requisitos, tratado na Sprint como produto *backlog*, com objetivo de identificar

as necessidades do cliente. Para isso, realizou-se uma entrevista com diretor(a) e secretários(as) com participação de todos os membros da equipe. Apesar da participação de todos, a entrevista foi conduzida por apenas dois integrantes, ambos com papeis de analistas de requisitos.

Após o levantamento de requisitos e descrição de todas as funções necessárias para entrega do produto final, definiu-se a métrica de pontos por função para dividir as funcionalidades entre as *Sprints*.

Medição por análise de Pontos por Função propõe a partir da especificação de requisitos, baseada em casos de uso, antecipar o trabalho de medição das funcionalidades, sob a visão do usuário, diminuindo assim o esforço dos líderes de projeto, obtendo uma compreensão intuitiva do porte do projeto em termos de Pontos por Função a partir dos requisitos elicitados (TAVARES; CARVALHO; CASTRO, 2002).

Dado continuidade as *Sprints* utilizaram-se a metodologia XP para gerenciamento do ciclo de vida. XP (*Extreme Programming*) é um sistema de desenvolvimento ágil implementado por equipes pequenas (no máximo 12 integrantes), com desenvolvimento incremental (implementação das funcionalidades de maneira gradual, com entregas frequentes), focado na comunicação com o cliente (TELES, 2017).

No processo XP o ciclo de vida do projeto foi divido em cinco etapas, onde em cada etapa foram desenvolvidas as seguintes atividades:

- Planejamento: Etapa para planejar e descrever as atividades desenvolvidas durante cada Sprint de maneira detalhada com: nome da função, data de início e data de término.
- Projeto: Elaboração da modelagem lógica e conceitual do banco de dados, e desenvolvimento dos diagramas UML de classe e casos de uso.
- Implementação: Codificação das funções definidas durante o planejamento.
- Testes: avaliação das funcionalidades implementadas e validação com o cliente.
- Implantação: release.

## RESULTADOS E DISCUSSÕES

A métrica de ponto por função foi utilizada no desenvolvimento do projeto para analisar os esforços necessários para o desenvolvimento, e dessa forma verificar se o projeto poderia ser implementado dentro do prazo que foi disponibilizado à equipe, no caso aproximadamente 3 meses.

Dada as funcionalidades do sistema, a aplicação da métrica de pontos por função resultou em 185 pontos de função ajustáveis. A equipe possuía 3 pessoas e foi definido que cada uma trabalharia 2 horas por dia, e cada ponto de função teria um esforço de 2 horas que resultou em um esforço estimado de 62 dias para entrega do produto final, de acordo com a seguinte equação

$$EPEH = \frac{(PFA*EF)}{QME}$$

EPEH = (PFA * EF) / QME (1)
EPED = EPEH / HTD (2)

Onde:

- EPEH: Esforço Por Equipe em Horas;
- PFA: Ponto de Função Ajustáveis;
- EF: Esforço por Função;
- QME: Quantidade de Membros da Equipe;
- EPED: Esforço Por Equipe em Dias;
- HTD: Horas Trabalhadas por Dia.

A partir desse cálculo, em dias, a equipe pode verificar que o projeto poderia ser implementado dentro do prazo, e desta forma definir o escopo do projeto.

Após a análise dos pontos por função, foram definidas 3 *Sprints*. Cada *Sprint* foi dividida de maneira a entregar parte do projeto final solicitado pelo cliente, obedecendo o número de pontos por função que poderia ser desenvolvido de acordo com cada prazo, conforme observado na tabela 2.

O projeto encontra-se na implementação da última *Sprint*, entretanto, até o momento, a métrica de pontos por função foi implementada corretamente, tendo em vista que as duas primeiras *Sprints* foram implementadas em menos de 20 dias.

**Tabela 2**: Divisão das *Sprints* com pontos por função, total de dias disponíveis e dias utilizados.

| Sprint | Número de pontos por função | Total de dias disponíveis | Total de dias utilizados |
|---|---|---|---|
| 1° Sprint | 60 | 20 | 15 |
| 2° Sprint | 60 | 20 | 16 |
| 3° Sprint | 65 | 22 | - |

Fonte: Desenvolvido pelos autores

## CONCLUSÕES

As técnicas e ferramentas da engenharia de software, descritas no trabalho, são de suma importância para o desenvolvimento de softwares por diminuir as burocracias, facilitando a comunicação de todos os envolvidos no sistema, permitindo também um melhor entendimento e acompanhamento do projeto.

A partir da análise dos resultados, até o momento pode-se perceber que dada a lista de todas as funcionalidades, a partir do levantamento de requisitos, e a disponibilidade e esforço

empregado da equipe, a métrica de análise de pontos por função é uma ferramenta eficaz por estimar maneira correta o esforço, em dias, necessárias para implementação de um software.

Pretende-se concluir em trabalhos futuros a análise sobre a utilização da métrica após a implementação da terceira *Sprint*. Como também apresentar a importância da prototipação para validação dos requisitos com o cliente.

## REFERÊNCIAS


CARVALHO, Ariadne; CHIOSSI, Thelma; DRACH, Marcos. Aplicabilidade de Métricas por Ponto de Função a Sistemas Baseados em Web. In: **WER**. 2006. p. 109-115.

DA SILVA, Edson Coutinho; LOVATO, Leandro Alvarez. Framework Scrum: eficiência em projetos de software. **Revista de Gestão e Projetos-GeP**, v. 7, n. 2, p. 01-15, 2016.

DIAS, Ana Luiza et al. Uma Revisão Sistemática sobre a inserção de Acessibilidade nas fases de desenvolvimento da Engenharia de Software em sistemas Web. **In IHC**, p. 39-48, 2010.

LEITE, Larissa Menune; LUCRÉDIO, Daniel. Desenvolvimento de software utilizando o framework scrum: um estudo de caso. **Revista TIS**, v. 3, n. 2, 2014.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. (Org.). **Metodos ágeis para desenvolvimento de software**. Porto Alegre: Bookman, 2014. 312p.

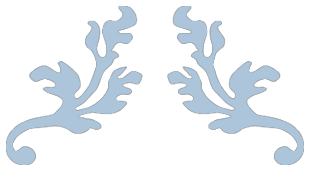
REZENDE, Denis Alcides. **Engenharia de software e sistemas de informação**. Brasport, 2005.

SATO, Danilo Toshiaki. Uso eficaz de métricas em métodos ágeis de desenvolvimento de software. Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, v. 139, 2007.

SOMMERVILLE, I. Arquitetura orientada a serviços. **Engenharia de Software**, p. 355-368, 2011.

TAVARES, H. C. A. B.; DE CARVALHO, Ana Elizabete Souza; CASTRO, Jaelson. Medição de pontos de Função a Partor da Especificação de Requisitos. In: **WER**. 2002. p. 278-298.

TELES, Vinícius Manhães. **Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade**. Novatec Editora, 2017.



# PROCESSO DE SOFTWARE

# METODOLOGIA DE DESENVOLVIMENTO DE UM SISTEMA WEB COM PROCESSOS DA ENGENHARIA DE SOFTWARE: O CASO DA ÓTICA X DA CIDADE DE CEDRO-CE

*Alyne Alves da Costa*[7]*; Bianca Souza Augusto*[8]*; Victoria Ricarte Bispo Beserra*[9]*; Bruno Rocha da Silva*[10].

## INTRODUÇÃO

Os procedimentos e métodos trazidos pelo estudo da engenharia de software melhoraram significativamente o processo de desenvolvimento de sistemas. Tais técnicas não só auxiliam na compreensão e análise das necessidades do cliente como também escolha do modelo de ciclo de vida do software e finalmente guiam o desenvolvedor em relação aos testes das aplicações. Segundo Filho (2011), a engenharia de software é essencial para o desenvolvimento e manutenção de um sistema e deve ser norteada pelos fatores tempo, custo e qualidade.

O uso de sistemas informatizados nas organizações propicia maior eficiência em determinadas atividades, além de reduzir o tempo necessário para a realização das mesmas (BAPTISTA; VARAJÃO; MOREIRA, 2013). Porém, algumas empresas ainda realizam atividades como cadastros ou busca de informações de clientes e serviços de forma manual e, consequentemente, lenta, como é o caso da ótica X localizada na cidade de Cedro-Ceará.

O objetivo deste trabalho é descrever o processo de desenvolvimento de um website, que irá aperfeiçoar o processo de gerenciamento de dados de clientes e fluxo de caixa da ótica X, enfatizando os processos e técnicas de engenharia de software utilizados.

O trabalho está estruturado da seguinte forma: na próxima seção apresenta-se a metodologia, onde são descritos os processos utilizados no desenvolvimento do projeto. Na seção seguinte são apontados os resultados obtidos em cada etapa da construção da aplicação, bem como discussões sobre os mesmos. Por fim, apresentam-se as conclusões, *feedbacks* da cliente e ideias de trabalhos futuros.

## METODOLOGIA

Este trabalho é fruto de um projeto realizado para as disciplinas de Laboratório de Desenvolvimento de Sistemas e Projeto Integrador II do sétimo semestre do Bacharelado em

[7] Graduando em Sistemas de Informação. IFCE - Campus Cedro. alineamelia2015@gmail.com.
[8] Graduando em Sistemas de Informação. IFCE - Campus Cedro. biankasouza.012@gmail.com.
[9] Graduando em Sistemas de Informação. IFCE - Campus Cedro. victoriaricartebispo@gmail.com.
[10] Graduando em Sistemas de Informação. IFCE - Campus Cedro. mail.brunorocha@gmail.com.

Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), campus Cedro. Foi solicitado aos discentes que se dividissem em equipes, onde cada uma destas representaria uma empresa e se dedicaria a desenvolver uma aplicação Web, utilizando técnicas da engenharia de software, para um determinado segmento de mercado. A empresa Equipe Rocket, formada por quatro dos discentes, teve como cliente a ótica X que atua nas cidades de Cedro e Aurora no interior do estado do Ceará.

Os professores das disciplinas mencionadas deixaram as equipes livres para escolher os processos e métodos que melhor avaliassem para desenvolver o site. Os alunos então pesquisaram sobre engenharia de software e cada equipe definiu sua metodologia de trabalho.

A ótica em questão fazia uso de um sistema comercial, que continha registros de vendas e informações de clientes do estabelecimento. A mesma relatou que atualmente não faz mais uso do software, porque não estava satisfeita com o valor da licença nem com as funcionalidades do produto. A administradora do estabelecimento apontou dificuldades no gerenciamento de informações e afirmou estar realizando tais processos de forma manuscrita, o que demanda muito tempo.

Foi então proposto o desenvolvimento colaborativo de um sistema Web para dar o suporte requerido pela empresa. Foi então realizado o levantamento de requisitos do projeto através de uma entrevista com a cliente.

## 1. Levantamento de requisitos

A primeira etapa no processo de desenvolvimento do software em questão foi o levantamento de requisitos, onde buscou-se conhecer e analisar as reais necessidades do cliente. Realizou-se então uma entrevista com a administradora da ótica X, na qual a cliente descreveu todos os requisitos/funcionalidades que deveriam fazer parte no sistema.

Nas tabelas abaixo estão listados, respectivamente, os requisitos funcionais e não funcionais que foram solicitados e desenvolvidos para o sistema.

Quadro 1 – Requisitos Funcionais

| **Id** | **Nome** | **Descrição** |
|---|---|---|
| RF001 | Cadastrar Clientes | O sistema deverá manter os dados dos clientes |
| RF002 | Cadastrar Produtos | O sistema deverá cadastrar os dados dos produtos: |
| RF003 | Registrar Venda | O sistema deverá registrar as vendas |

| | | |
|---|---|---|
| RF004 | Emitir nota promissória | O sistema deverá emitir uma promissória quando solicitado. |
| RF005 | Cadastrar Refratometria | O sistema deverá registrar a refratometria resultante de uma consulta oftalmológica |
| RF006 | Emitir Carnê de Pagamento | O sistema deverá emitir os carnês de pagamento. |
| RF007 | Cadastrar Funcionários | O administrador poderá manter funcionários para que os mesmos tenham acesso ao sistema. |
| RF008 | Exibir Relatórios Financeiros | O sistema deverá exibir quando solicitado um relatório sobre as vendas mensais. |

Quadro 2 – Requisitos Não Funcionais

| **Id** | **Nome** | **Descrição** |
|---|---|---|
| RNF001 | Linguagem de programação back-end | O sistema será desenvolvido com PHP na versão 7. |
| RNF002 | Usabilidade do sistema | A usabilidade do sistema deve atender as dez heurísticas de Nielsen. |
| RNF003 | Cores base | A interface do sistema deve ter como base as cores da logomarca da empresa. |
| RNF004 | Adaptabilidade do sistema | O sistema deve ter uma interface responsiva para se adaptar a telas com as seguintes dimensões: 320x480, 640x960 e 1024x768. |
| RNF005 | Login no sistema | O sistema deverá ter um login multiusuário com dois níveis de acesso (Administrador e Funcionário). Para a autenticação deverão ser solicitados nome de usuário e senha. |

Os softwares utilizados para auxiliar na documentação do projeto foram o GitHub, uma plataforma de hospedagem para código fonte que ajudou na organização e controle do desenvolvimento, e o Trello, uma ferramenta de colaboração para organizar quadros tipo *Kanban*, que auxiliou no controle do desenvolvimento das *sprints*. A prototipação das telas foi feita utilizando o Photoshop CS6. Para o desenvolvimento do site foi utilizada a linguagem de programação PHP7, a linguagem de marcação HTML5 e CSS3 para a estilização das telas. Também para a estilização foi usado o Bootstrap4, um framework front-end nas linguagens CSS e Javascript. Para o suporte do banco de dados foi utilizando o MariaDB 5.2, baseado no MySQL.

## RESULTADOS E DISCUSSÕES

Foram escolhidas para este projeto as linguagens: PHP, na versão 7, para a construção do site e SQL para a implementação do banco de dados da aplicação, tendo como sistema gerenciador de banco de dados o MySQL. Utilizou-se também o *framework front-end* Bootstrap, versão 4. Em relação aos ambientes de desenvolvimento, foram utilizados os softwares Sublime Text (para programação *back* e *front end*), e XAMPP (para executar o sistema).

### Divisão das sprints

O prazo máximo para a entrega do sistema foi de três meses, então optou-se pelo desenvolvimento baseado em métodos ágeis. A metodologia ágil tem alta adaptabilidade a mudanças e efetiva participação do cliente através de entregas parciais das funcionalidades (LIBARDI; BARBOSA, 2010).

Dentre os métodos ágeis conhecidos escolheu-se para este projeto o SCRUM, que é dividido em *sprints*. Durante cada *sprint* são desenvolvidas e testadas parte das funcionalidades do sistema, o que garante maior controle sobre possíveis mudanças ou falhas (LIBARDI; BARBOSA, 2010).

O projeto foi então dividido em três *sprints*, tendo um intervalo de aproximadamente 20 dias entre estas. Para dividir quais funcionalidades seriam desenvolvidas em cada *sprint* utilizou-se a técnica de ajuste por pontos de função, que leva em consideração a relevância de cada funcionalidade, o esforço e o tempo necessários para seu desenvolvimento (LOPES, 2011).

As *sprints* foram divididas da seguinte forma:

- *Sprint* 01: Implementar funções cadastrar, atualizar, excluir das entidades cliente, funcionário e produto;
- *Sprint* 02: Implementar funções cadastrar, atualizar, excluir da entidade venda e cadastrar e atualizar refratometria, desenvolver o login do sistema;
- *Sprint* 03: Implementar funções consulta de refratometria, excluir venda, consultar cliente por nome, emitir carnê e promissória, emitir relatório financeiro.

Foram feitos testes unitários a cada *sprint* e ao final do desenvolvimento o software foi hospedado e testado pela cliente para validação de requisitos e determinação de possíveis ajustes.

### Prototipagem de telas

Deu-se então início ao desenvolvimento do fluxo de telas pertinente a toda parte gráfica e visual do software. Utilizando a ferramenta Photoshop CS6 as telas foram prototipadas e apresentadas para a cliente para validação.

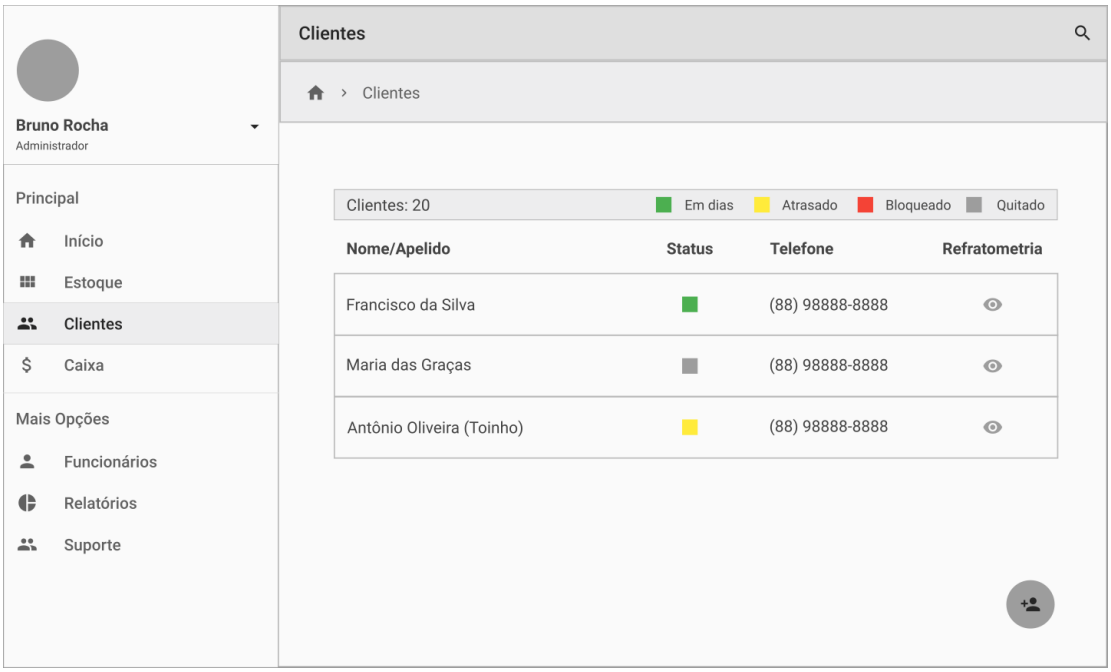



Figura 1 – Protótipo da Tela de Clientes

Os protótipos foram desenvolvidos de forma responsiva, ou seja, o layout se adapta a tela do dispositivo em que o site é requisitado. Abaixo é mostrada a tela de consulta de clientes, na qual é possível observar refratometrias antigas e a situação de pagamento dos clientes.

## CONCLUSÕES

Os resultados obtidos foram satisfatórios e condizentes com aquilo que foi solicitado pela cliente. As funcionalidades foram desenvolvidas conforme o planejamento inicial das *sprints* sem necessidade de reajustes, fato que foi possível devido ao planejamento detalhado das ações. Pode-se concluir então que o uso das técnicas de engenharia de software melhora consideravelmente o processo de construção de software e, consequentemente, a qualidade do mesmo.

O software foi avaliado pela administradora da ótica X que afirmou estar satisfeita com o produto final. Planeja-se dar continuidade ao sistema e implementar funções para o consultório oftalmológico do estabelecimento.

## REFERÊNCIAS

BAPTISTA, João Paulo; VARAJÃO, João; MOREIRA, Fernando. Função Sistemas de Informação nas organizações: realidade, desafios e oportunidades do uso de arquiteturas

empresariais. **Novas tendências e marketing intelligence**, p. 155-159, 2013. Disponível em: < https://scholar.google.com.br>. Acesso em: 18 abr. 2018.

FILHO, Antonio Mendes da Silva. Engenharia de Software – Essencial para próximas décadas. **Revista Espaço Acadêmico**, n. 121, p.60-66, jun. 2011. Mensal. Disponível em: < http://www.periodicos.capes.gov.br>. Acesso em: 18 abr. 2018.

LIBARDI, Paula LO; BARBOSA, Vladimir. Métodos ágeis. **Monografia (Graduação em ciência da computação)**, 2010. Disponível em: < https://scholar.google.com.br >. Acesso em: 20 abr. 2018.

LOPES, Jhoney da Silva. **Guia Prático em Análise de Ponto de Função**, 2011. Disponível em: < fattocs.com/files/pt/livro-apf/citacao/JhoneySLopes-JoseLBraga-2011.pdf>. Acesso em: 19 abr. 2018.

# PROPOSTA DE UM PROCESSO ÁGIL PARA PROJETOS DE SOFTWARE PARA EQUIPES COM POUCA OU NENHUMA EXPERIÊNCIA

*Luan Dharlin Lemos da Silva[11]; Deyvison Nogueira Rodrigues[12]; Matheus de Souza Oliveira[3].*

## INTRODUÇÃO

De acordo com a ABES (2017), considerando-se as 4.872 empresas que atuam no desenvolvimento e produção de *software* no Brasil, cerca de 95% podem ser classificadas como micro e pequenas empresas. De acordo com (BERNI, 2010 *apud* O'CONNOR; COLEMAN, 2007), essas pequenas empresas de *software* precisam ser flexíveis e altamente dinâmicas, sempre buscando melhorar o *time-to-market*, que é o tempo que leva da demanda de um produto até sua entrega ao cliente. Por essa questão, estas empresas adotam um comportamento *ad hoc*, ou seja, deixam de lado o gerenciamento do projeto e alocam praticamente todos os seus recursos apenas no desenvolvimento, com o receio de burocratizar e engessar demais as suas rotinas. Este gerenciamento deve ser realizado através da aplicação e integração de processos e práticas de gerenciamento definidos pela organização ou pelo responsável pelo projeto, permitindo que o mesmo seja executado de forma eficiente e eficaz.

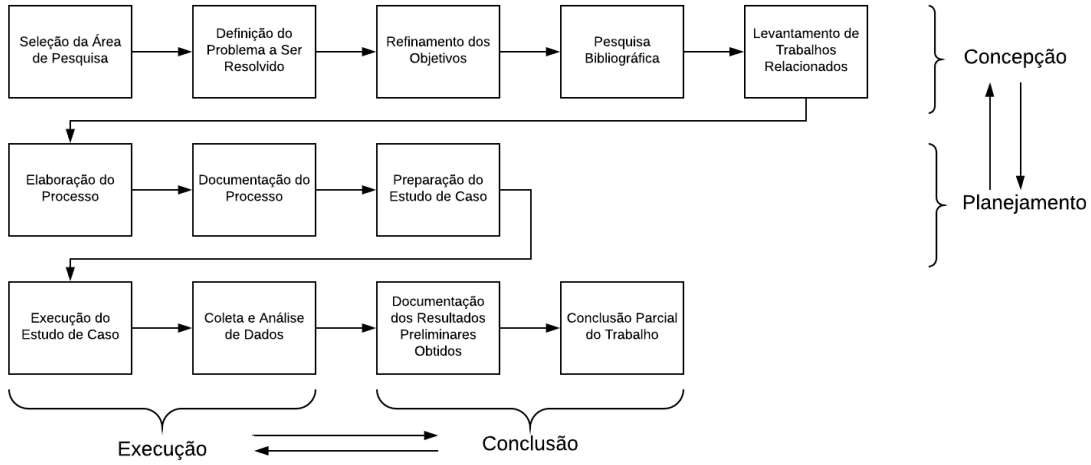
"Em projetos complexos como o desenvolvimento de *software*, a incerteza e imprevisibilidade nos direcionam a utilizar uma abordagem orientada a valor, visto que a entrega final tem grande probabilidade de não ser a gerada nos primeiros momentos do projeto"(RIBEIRO; RIBEIRO, 2015). No entanto, para Abrahamsson *et al.* (2003), apesar das metodologias ágeis darem suporte à gerência, elas deixam lacunas que podem colocar o projeto em risco, fazendo-se necessário a adoção de práticas dos métodos tradicionais. Deste modo, fica claro a necessidade de utilizar-se uma abordagem mista, usando métodos tradicionais e ágeis um como complemento do outro.

O objetivo geral deste trabalho é apresentar um processo ágil para gerenciamento de projetos de *software* que auxilie equipes com pouca ou nenhuma experiência na execução e controle das atividades do projeto.

## METODOLOGIA

[11] Discente do curso de Engenharia de Software da Universidade Federal do Ceará Campus Russas, atuando em pesquisas na área de Gerência de Projetos e Processos de Software. E-mail: luandharlinls@gmail.com
[12] Discente do curso de Engenharia de Software da Universidade Federal do Ceará Campus Russas. E-mail: deyvison_s@hotmail.com
[3] Discente do curso de Engenharia de Software da Universidade Federal do Ceará Campus Russas. E-mail: matheusoliveira2496@gmail.com

A metodologia utilizada neste trabalho foi o Estudo de Caso, cujo objetivo é o estudo empírico e aprofundado de um contexto específico, tendo carácter exploratório e prospectivo. A metodologia utilizada é ilustrada na Figura 1.




Fonte: Elaborado pelo Autor

Figura 1 - Metodologia de Trabalho

Iniciou-se pela seleção da área de pesquisa e pela definição do problema a ser resolvido, seguido posteriormente pelo estabelecimento dos objetivos do trabalho. A partir disso foi realizada uma revisão bibliográfica. Tomando como base a problemática, os objetivos e esta revisão bibliográfica, foram elencadas as práticas e as metodologias que melhor se enquadraram ao problema em questão. A partir daí houve a experimentação de práticas que resultou na elaboração e documentação do processo para gerenciamento e execução de projetos. Com o processo documentado, um estudo de caso foi executado, havendo em paralelo a coleta e análise dos dados gerados pelo mesmo. A partir desses dados que já foram analisados, obteve-se resultados que foram documentados e usados para fazer a conclusão sobre este trabalho.

Vale ressaltar a natureza iterativa entre as fases de Concepção e Planejamento e entre as fases de Execução e Conclusão. Isso se faz necessário pelo empirismo deste trabalho, que é baseado na melhoria por experimentação.

## RESULTADOS E DISCUSSÕES

A maior contribuição deste trabalho foi a elaboração de um processo que define um passo a passo confiável para dar suporte a equipes com pouca ou nenhuma experiência em desenvolvimento profissional de *software*. O processo proposto segue o ciclo de vida *Scrum* e utiliza-se dos papéis, eventos e artefatos do mesmo. É dividido em cinco fases com seus

respectivos subprocessos, e estes, por sua vez, contêm seus artefatos de entrada e saída, ferramentas, responsáveis e participantes. Essas fases são:

- Fase de Comunicação: contém os processos e atividades relacionados em providenciar o início do projeto, criando uma visão inicial do produto a ser desenvolvido, identificando as partes interessadas, os requisitos básicos e as restrições impostas, alinhando o projeto com toda a organização. Seus subprocessos são Identificar Partes Interessadas, Criar Termo de Abertura do Projeto e Criar *Backlog* do Produto ;
- Fase de Planejamento: contém os processos e atividades relacionados em definir como o projeto deve ser tratado, para guiar a equipe na abordagem do cronograma, escopo, partes interessadas, riscos, qualidade, recursos e outros aspectos relevantes ao projeto. Seus subprocessos são Coletar Requisitos, Criar Histórias de Usuário, Priorizar Histórias de Usuário, Criar Estrutura Analítica do Projeto (EAP), Estimar Recursos das Atividades, Criar *Backlog* da *Sprint* e Identificar Riscos;
- Fase de Construção: contém os processos e atividades relacionados a colocar em prática o que foi planejado. A partir de artefatos oriundos da fase de planejamento gera outros artefatos, principalmente versões funcionais do produto. Seus subprocessos são Implementar Entregáveis, Orientar e Gerenciar Trabalho do Projeto e Gerenciar o Conhecimento do Projeto;
- Fase de Controle: contém os processos e atividades relacionados em monitorar e controlar o andamento do projeto, aferindo desempenho da do time de desenvolvimento e outros indicadores para alinhar o trabalho real com o trabalho planejado, monitorando riscos e possíveis mudanças Seus subprocessos são Monitorar e Controlar Trabalho do Projeto, Monitorar e Controlar Riscos e Validar e Controlar Escopo;
- Fase de Entrega: Contém os processos e atividades relacionadas com a transição de artefatos para os usuários finais e clientes e avaliação processual, da equipe e individual. Seus subprocessos são Realizar Revisão da *Sprint*, Realizar Retrospectiva da *Sprint* e Formalizar Entrega.

Outras contribuições, oriundas da implantação do processo, podem ser mencionadas, como o possível ganho de produtividade quando se aplicam práticas e princípios ágeis e a prova que há a possibilidade de se adotar práticas mundialmente reconhecidas em um contexto menos grandioso, como a aplicação de processos do Guia Project Management Body of Knowledge (PMBOK).

O processo foi implantado dentro de um contexto real de desenvolvimento de *software*, contando com uma equipe de cinco pessoas, no segundo semestre de 2017, mesmo período em que ocorreu a coleta e análise de seus dados. Observa-se como resultado real da aplicação do processo proposto um aumento gradativo no desempenho do time, considerando a pontuação das atividades. Isso é melhor ilustrado na Figura 2, comparando os gráficos A, B e C.

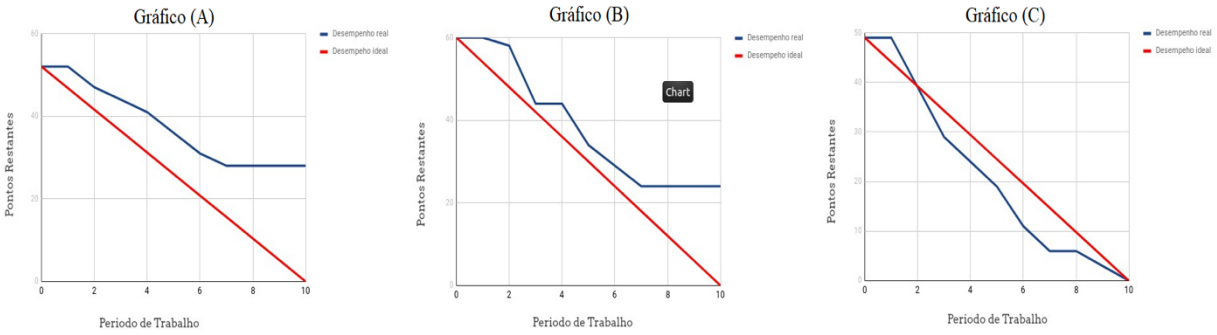



Figura 2 - Gráficos de Desempenho das Sprints
Fonte: Elaborado pelo Autor

Nos gráficos, o eixo horizontal representa os períodos de trabalho no projeto usado como experimentação, que são formados por duas horas cada um. O eixo vertical representa os pontos restantes para conclusão da *Sprint* como planejado. A linha vermelha indica o trabalho ideal a ser realizado. Dividindo-se a quantidade de pontos pela quantidade de períodos de trabalhos, podemos obter o trabalho diário necessário, e apenas o necessário para a conclusão do trabalho destinado à *Sprint*. A linha azul indica o trabalho realizado pelo time de desenvolvimento.

Percebe-se na Figura 2, Gráfico (A), que ilustra o desempenho na primeira *Sprint,* que em momento algum o time conseguiu atingir a linha de trabalho ideal, estando sempre com o trabalho em atraso, e, em certo momento, ficando estagnada no desenvolver da *Sprint*. Isso ocorre por dois motivos principais, a alocação demasiada de trabalho e má pontuação das atividades do *Sprint Backlog*. É comum que equipes inexperientes a prática de superestimar ou subestimar o trabalho a ser realizado, acarretando nos problemas supracitados. Ao comparar o Gráfico (A) com o Gráfico (B), que ilustra o desempenho da equipe na segunda *Sprint,* é possível notar uma pequena melhora de desempenho do time de desenvolvimento, havendo uma maior aproximação entre as linhas de trabalho ideal e trabalho realizado, porém, em nenhum dos dois casos o *Sprint Backlog* foi concluído. Essa melhoria gradativa a pequenos passos se repetiu ao decorrer das *Sprints*.

Ainda na Figura 2, o Gráfico (C), que ilustra o desempenho da equipe na terceira *Sprint,* mostra o aprimoramento do time, que manteve o trabalho realizado quase sempre a frente do trabalho ideal. Com a experiência adquirida nas *Sprints* anteriores, a equipe pode conhecer

melhor suas limitações e pontuar melhor as atividades, evitando os problemas encontrados nos ciclos de trabalhos anteriores. Além deste ganho de produtividade outros resultados foram mencionados na reunião de retrospectiva pelas pessoas envolvidas no estudo de caso, como:

- Melhor divisão do trabalho, evitando sobrecarga em uns membros e ociosidade em outros;
- Maior clareza do trabalho a ser realizado devido ao passo a passo estabelecido;
- Aumento da coesão dos papéis e suas funções;
- Maior controle do andamento do projeto, todavia maior facilidade de demonstração de resultados.

No quesito dificuldades encontradas, a equipe do projeto mencionou por meio de reunião de retrospectiva, a falta de experiência com o *Framework Scrum,* principalmente quem atuou no papel de *Scrum Master*.

## CONCLUSÕES

Com base no problema em questão, nos objetivos levantados e nos resultados alcançados, pode-se concluir uma melhoria significativa sobre o planejamento, controle e execução do projeto. Como corolário, temos que houve uma melhora no desempenho do time de desenvolvimento e no ganho de conhecimento do mesmo. Fatores estes ocasionados pela definição de um passo a passo a ser seguido, utilizando de práticas ágeis de gerenciamento e execução de projetos.

## REFERÊNCIAS


ABES. **ABES SOFTWARE  Mercado Brasileiro de Software - Panorama e Tendências.** 2017. Disponível em: <http://central.abessoftware.com.br/Content/UploadedFiles/Arquivos/Dados\%202011/ABES-Publicacao-Mercado-2017.pdf>.

ABRAHAMSSON, P.; WARSTA, J.; SIPONEN, M. T.; RONKAINEN, J. New directions on agile methods: a comparative analysis. In: IEEE. **Software Engineering, 2003. Proceedings. 25th International Conference on** . [S.l.], 2003. p. 244–254.

BERNI, J. C. A. **Gestão para o processo de desenvolvimento de software cientÍfico, utilizando uma abordagem Ágil e adaptativa na microempresa**. Tese(Mestrado em Engenharia de Produção) - Universidade Federal de Santa Maria, Santa Maria, Rio Grande do Sul, 2010. Disponível em: <http://repositorio.ufsm.br/bitstream/handle/1/8132/BERNI,\%20JEAN\

%20CARLO\%20ALBIERO.pdf>.

O'CONNOR, R. V.; COLEMAN, G. An investigation of barriers to the adoption of software process best practice models. **Proceedings of the Australasian Conference on Information Systems**, Toowoomba, Australian, n.18, p. 780 – 789, December 2007. Disponível em: <https://www.researchgate.net/publication/228641056_An_Investigation_of_Barriers_to_the_Adoption_of_Software_Process_Best_Practice_Models>

RIBEIRO, R. D.; RIBEIRO, H. da Cunha e S. **Gerenciamento de Projetos Orientados a Planos**. [S.l.]: SPIN, 2015. ISBN 978-85-919102-2-9.

# SPIN: UM PROCESSO ÁGIL PARA DESENVOLVIMENTO DEPROJETOS INTEGRADORES

*Adriel Vieira Santos*[13]

## INTRODUÇÃO

Durante a graduação, os alunos dos cursos de computação são instruídos acerca de técnicas e metodologias para o desenvolvimento profissional de Software. Entretanto levando em consideração que geralmente os projetos são realizados por pequenas equipes e com grande limitação de tempo, custo e experiência, alguns processos acabam não sendo adequados para utilização. Com base nesse problema, este projeto visa propor e aplicar um novo modelo de processo de software chamado de SPIN (Simple Process for INtegrated Projects), específico para grupos acadêmicos pequenos com restrições de tempo e recursos.

De acordo com o que foi analisado de outras instituições como o IFCE Campus Cedro, as principais dificuldades na utilização de processos já disponíveis no mercado vêm principalmente da falta de experiência dos alunos tanto em desenvolvimento de software quanto em utilização de processos para gerenciar o projeto. Essa carência reflete-se em pontos como a falta de conhecimento do próprio ritmo de trabalho individual, do ritmo de trabalho em equipe, consequentemente nas estimativas quanto a prazo de entrega. Portanto um novo processo focado no público acadêmico que evita a produção de grande quantidade de artefatos e caracterizasse por sua simplicidade de aprendizagem é de grande importância para academia.

## METODOLOGIA

Como esse projeto objetiva-se desenvolver e avaliar os resultados da aplicação de um novo modelo de processo de software. A primeira ação tomada foi o levantamento bibliográfico sobre métodos ágeis e modelos clássicos (PRESSMAN, 2011) para o desenvolvimento de software, analisando as necessidades da produção dos mesmos e suas características.

Após o levantamento teórico, foi iniciado o estudo sobre o ambiente alvo do projeto. O principal objetivo era entender como os projetos são desenvolvidos, quais metodologias são aplicadas e quais são os resultados obtidos.

[13] Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Crato

Toda essa análise serviu de base para o novo processo de software, abordando todas as etapas identificadas como necessárias, bem como papéis e artefatos gerados pelo processo. Além disso o SPIN se baseia no estudo bibliográfico, tomando como modelo principal a metodologia ágil (BECK, Kent et al, 2017).

## RESULTADOS E DISCUSSÕES

O processo em desenvolvimento está sendo aplicado no IFCE Campus Crato na cadeira: Análise e projetos de Sistemas II. A disciplina possui 29 alunos divididos em 5 grupos diferentes, cada grupo possui um nome e um projeto real associado a clientes internos da instituição de ensino, como também empresas da região.

Atualmente o novo processo possui definido os seguintes papeis: Gerente de projetos, Analista, Arquiteto, Desenvolvedor e Testador. O seu ciclo de vida é iterativo e incremental (PRESSMAN, 2011), dividido em quatro fases básicas: Concepção, Definição, Desenvolvimento (SABBAGH, 2013) (TELES, 2018) e entrega. A fase de desenvolvimento é subdividida em três partes: planejamento de iteração, desenvolvimento de iteração e validação de iteração. As fases de desenvolvimento e entrega podem ocorrer diversas vezes de acordo com a quantidade de releases necessárias. O processo ainda sugere a utilização de padrões de projeto, refatoração de código, propriedade coletiva do código e produção de testes antes do desenvolvimento.

As equipes da disciplina se encontram no final da fase de definição em que é feita a arquitetura do sistema e o início da produção dos testes, já que, assim como o eXtreme Programming (XP) (TELES, 2018), o SPIN sugere a produção de testes antes do desenvolvimento. A fase anterior foi a de concepção, etapa de planejamento, obtenção das histórias de usuários utilizando-se do modelo (QUEM, O QUE, PORQUE) (BERNARDO, 2018), definição de tempo, custo e escopo, como também elicitação dos requisitos. Os períodos foram marcados por apresentações sobre temas como: coleta de requisitos, gerenciamento de projetos, qualidade de software, métricas de software, teste de software, papéis e suas funções.

Durante essas duas etapas foi visível a influência do processo sobre as equipes, isso porque todas estão com os objetivos básicos solicitados nas duas fases completos. A constatação desse fato foi perceptível durante as apresentações de todos os artefatos solicitados durante os dois períodos sendo eles: histórias de usuário, requisitos, diagrama de caso de uso, diagrama de classe, modelo lógico de dados (quando necessário), planejamento de teste, planejamento de tempo, escopo, riscos e atas de reunião.

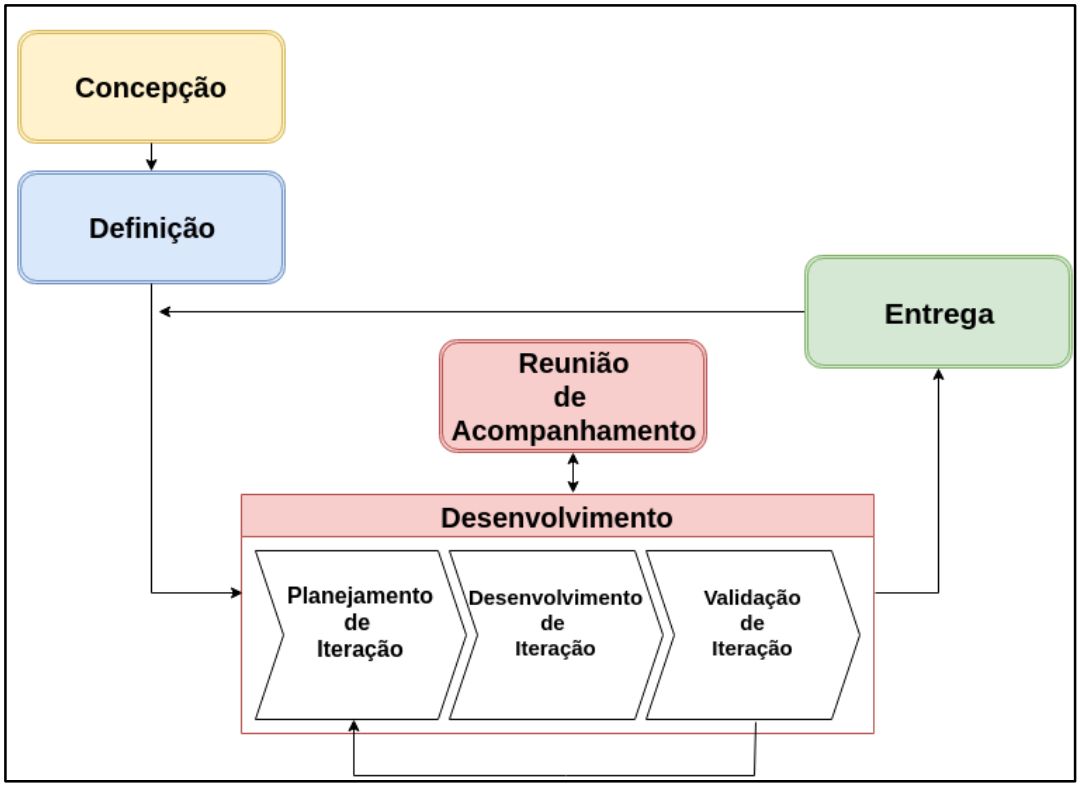



Figura 01. Ciclo de vida do processo SPIN.

Fonte: Elaborado pelo autor (2018).

## CONCLUSÕES

É evidente que o processo de software é algo de extrema importância para a produção de aplicações com qualidade. Embora esse conceito vindo da engenharia de software seja de difícil aplicação, a produção de processos com uma redução na carga de artefatos, papéis e atividades baseado em uma metodologia ágil fazendo com que as equipes sejam flexíveis a mudança, acaba facilitando a aprendizagem dentro da academia e com isso preparando os alunos para os mais variados tipos de processo aplicados no mercado de trabalho.

## REFERÊNCIAS

PRESSMAN, Roger. Engenharia de Software: Uma abordagem profissional. 7. Ed. [S.I.]: Bookman, 2011. 780 p.

BECK, Kent et al. Manifesto para o desenvolvimento ágil de software. Disponível em: <http://www.manifestoagil.com.br/>. Acesso em: 18 fev. 2017.

SABBAGH, Rafael. Scrum: Gestão Ágil para Projetos de Sucesso. Ed. 1. Casa do Código, 2013.

Teles, Vinícius Manhães. Scrum. Disponível em <http://www.desenvolvimentoagil.com.br/scrum/>. Acesso em 15 de março de 2018.

Teles, Vinícius Manhães. Papéis do XP. Disponível em <http://www.desenvolvimentoagil.com.br/xp/praticas/>.Acesso em 15 de março de 2018.

Bernardo, Kleber. Estória de usuário você saberia contar. Disponível em <https://www.culturaagil.com.br/estoria-de-usuario-voce-saberia-contar/>. Acesso em 29 de março de 2018.

# UMA ANÁLISE COMPARATIVA ENTRE METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE

*[14]Sara Macêdo Lima; [15]Talles Brito Viana*

## INTRODUÇÃO

Nos últimos anos vem crescendo constantemente a demanda em desenvolvimento de software, logo surge a necessidade para um mais alto nível de gerenciamento, organização e maturidade no desenvolvimento de software.

Muitas são as metodologias de desenvolvimento, porém por se tratar de um assunto novo, muitas empresas têm dúvidas em qual usar. A escolha de uma metodologia errada pode ser um fator crucial para o fracasso de projetos de software. Pensando nisso, nesse trabalho são investigadas quatro metodologias que nos últimos anos têm sido usadas em grande número pelas empresas, na busca por melhorias no desenvolvimento de software: *Scrum*, *XP (Extreme Programming)*, *FDD* (*Feature Driven Development*) e *Kanban*. Neste contexto, esse trabalho tem como objetivo apresentar uma discussão comparativa sobre tais metodologias ágeis para o desenvolvimento de software.

De acordo com Souza (2007), o XP é uma metodologia ágil para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente. Já a metodologia FDD, ou simplesmente, desenvolvimento guiado por funcionalidades, tem como principal objetivo alcançar resultados frequentes, tangíveis e funcionais (Cunha e Filipakis, 2012). Por fim, o Scrum é uma metodologia ágil que tem por objetivo gerenciar os processos de desenvolvimento de software. O Scrum é focado nas pessoas e indicados para ambientes em que os requisitos surgem e mudam rapidamente (Silva e Hoentsch, 2009).

É preciso observar que o Kanban não é necessariamente uma metodologia de desenvolvimento, mas um *framework* de organização de tarefas que pode ser aplicado em diversas metodologias de desenvolvimento. Apesar disso, neste trabalho também consideramos e analisamos a situação em que os desenvolvedores utilizam somente o Kanban para organizar as atividades de um projeto sem a aplicação de uma metodologia de desenvolvimento específica concorrentemente.

Neste contexto, esse trabalho tem como objetivo apresentar uma discussão comparativa sobre tais metodologias ágeis para o desenvolvimento de software.

[14] Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Crato.
[15] Docente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Crato.

## METODOLOGIA

A metodologia empregada neste trabalho é a pesquisa exploratória em que a fonte de pesquisa foi o Google Acadêmico[1]. Este mesmo dispõe de um vasto conteúdo relacionado à engenharia de software. A forma de pesquisa bibliográfica foi baseada em material já elaborado, constituído de livros e artigos científicos. Tais trabalhos foram coletados e julgados para escolha a partir da leitura dos mesmos. Após a leitura, foi feita uma comparação entre os trabalhos, para com isso chegar aos resultados alcançados com esta pesquisa. A quantidade de material escolhido foi ao todo de doze (12) trabalhos.

Um dos critérios de inclusão utilizados para a esta pesquisa incluiu somente selecionar trabalhos elaborados após o ano 2001, pois foi neste ano que surgiu o manifesto ágil para desenvolvimento de software. Além disso, somente as metodologias que atendem aos princípios deste manifesto foram estudadas neste trabalho. Especificamente, foi considerada de extrema importância verificar como que as metodologias estudadas lidam com o tempo e custo de desenvolvimento, a variação de escopo do projeto, e como é feita a alocação de pessoas em um projeto de desenvolvimento de software.

## RESULTADOS E DISCUSSÕES

Com intuito de sumarizar os estudos realizados, foi criado um quadro comparativo das metodologias analisadas em relação a alguns parâmetros como: custo, tempo, escopo variável e alocação de pessoas. Foram realizadas pesquisas em como essas metodologias se propõe para resolver os referidos parâmetros, os quais foram categorizados em uma escala: Alto, Média e Baixa aderência. Um resumo desta análise é mostrado na Tabela 1. Espera-se que a análise das metodologias ágeis em relação a estes parâmetros possa servir de insumos para escolha da melhor metodologia ágil para as empresas.

***i) Scrum***: *Tempo*: A metodologia Scrum estima o tempo das atividades no projeto através de um método chamado *Planning Poker*, o mesmo estabelece o tempo de desenvolvimento das atividades descritas no *Product Backlog* através de uma sequência de

[1] https://scholar.google.com.br/ 3

Poker para estimar o tempo necessita-se do Product Backlog e de um baralho de cartas, em que as cartas devem ter os seguintes números 0, 1, 2, 3, 5, 8, 13, 20, 40 e 100, semelhante à sequência de Fibonacci.

*Tabela 1- Quadro Comparativo das Metodologias*

| PARÂMETROS ANALISADOS | SCRUM | XP | FDD | KANBAN |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| CUSTO | Alta Aderência | Alta Aderência | Alta Aderência | Baixa Aderência |
| TEMPO | Média Aderência | Média Aderência | Alta Aderência | Baixa Aderência |
| ESCOPO VARIÁVEL | Alta Aderência | Alta Aderência | Alta Aderência | Média Aderência |
| ALOCAÇÃO DE PESSOAS | Alta Aderência | Alta Aderência | Média Aderência | Baixa Aderência |

*Custo***:** Uma vez que as estimativas de tempo foram realizadas, é possível fazer o controle de custo. O orçamento de cada *Sprint* será calculado de acordo com as horas despendidas. Sabendo o valor bruto da Sprint chega-se ao total do custo do projeto, se, por exemplo, tem-se um projeto estimado com 4 Sprints, então o custo total será, o valor bruto da Sprint multiplicado pelo número total de Sprints do projeto, chegando ao valor do custo total.

*Escopo variável*: O Scrum é classificado como alta aderência ao quesito escopo, pois toda alteração necessária ocorre no levantamento do backlog do produto. Se for percebido que é necessário implementar modificações no produto, na iteração seguinte essas mudanças já podem ser implementadas de acordo com a prioridade definida pelo dono do produto.

*Alocação de Pessoas*: O Scrum define uma divisão de trabalho bastante dinâmica, alocando cada integrante ao seu papel, que pode ser *Scrum Master, Product Owner ou Scrum Team*. Cada papel é separado no time Scrum, logo cada membro da equipe fica responsável por sua tarefa, evitando com isso sobrecarga de trabalho.

***ii) XP:*** *Tempo*: O XP assim como o FDD utiliza o método Análise de Pontos de Função para auxiliar no cálculo das estimativas. Para estimar uma história tem-se que definir o tamanho da mesma através de pontos de função.

*Custo:* Para estimar o custo no projeto, o desenvolvedor precisa saber o tamanho do projeto, o valor da hora do profissional, o número de pontos de função e o preço de cada ponto de função, com esses dados é possível estimar o custo.

*Escopo variável*: No quesito escopo variável o XP atende bem as necessidades deste requisito. Alterações podem ser feitas a qualquer momento, já que o cliente pode sentir a necessidade de atribuir ou retirar funcionalidades. Pressman define bem essa questão da seguinte forma: conforme o trabalho de desenvolvimento prossegue, o cliente pode acrescentar histórias, mudar o valor de uma história existente, dividir algumas ou eliminá-las. (Pressman, 2011).

*Alocação de pessoas*: O processo de planejamento do XP confia claramente na separação de papéis de pessoas do negócio e pessoas de software. Isto assegura que as pessoas do negócio (analistas de negócios) sejam responsáveis pelas decisões de negócio e as pessoas técnicas (programadores) tomem as decisões técnicas (Hazan, 2005).

***iii) FDD*****:** *Tempo:* O FDD não possui uma prática pré-estabelecida para calcular o tempo de duração do projeto, por este motivo essa metodologia emprega com um método de estimativas por Análise de Pontos de Função. Análise de Pontos de Função é uma técnica de medição do tamanho funcional de um software. Essas funções são operações extraídas dos requisitos funcionais gerados a partir da visão do usuário. (LOPES, 2011).

*Custo*: O custo é calculado mediante dados que são fornecidos pela empresa, como valor da hora de trabalho do desenvolvedor e o valor do ponto de função, tendo conhecimento destes dois dados é possível calcular o custo de desenvolvimento.

*Escopo variável*: No processo de desenvolvimento, uma vez definido o escopo também podem ser realizadas alterações no mesmo, conforme citado em (Barbosa, Azevedo e Pereira, 2007): "O cliente indica quais os requisitos que quer ver cumpridos pelo sistema. Cabe às equipes de desenvolvimento analisar o sistema anterior e verificar se todos os requisitos foram especificados pelo cliente, sugerir novos requisitos e colocar todas as questões que possam ainda não terem sido respondidas, ou que tenham sido esquecidas".

*Alocação de pessoas*: No FDD é elaborado um plano para cada funcionalidade e são atribuídas responsabilidades. As responsabilidades de uma equipe no FDD são distribuídas por papéis, onde cada membro da equipe pode assumir mais de um papel simultaneamente e um papel pode ser assumido por mais de um membro da equipe (CUNHA e FILIPAKIS, 2012). Diante disso, será classificado como média aderência em nossa tabela, considera-se como ponto negativo o fato de um desenvolvedor poder ficar com mais de uma função atribuída para si no projeto, pois isso pode gerar sobrecarga de trabalho.

***iv) Kanban:*** *Tempo*: O Kanban não utiliza um método específico para estimar o tempo como as outras metodologias abordadas neste trabalho.

*Custo:* Quando se refere ao custo, o Kanban também não define uma métrica específica para medir o custo do trabalho, bem como não prescreve nenhum método, com isso não é possível estimar o esforço que uma equipe irá empregar para desenvolver um projeto.

*Escopo*: Classificado na tabela com média aderência ao escopo, caso surja alguma necessidade de alterar requisitos, o Kanban altera-os diante das histórias a serem colocadas em cartões e tornando-as visíveis. Assim a equipe define o escopo e o altera quantas vezes forem necessárias. Além disso, como afirma Silva e Santos (2012): no Kanban o escopo pode ser alterado constantemente.

*Alocação de pessoas:* Kanban não prescreve papel nenhum (KNIBER e SKARIN, 2009). Não ocorre divisão de trabalho, não prescreve papéis, não define uma equipe responsável por cada parte projeto. Sendo assim quem utilizar somente o Kanban no projeto pode gerar sobrecarga de trabalho ocorrendo com isso perda de produtividade da equipe.

## CONCLUSÃO

Neste trabalho foi apresentada uma análise comparativa entre as metodologias ágeis Scrum, XP, FDD e Kanban. Cada metodologia foi analisada em relação aos quesitos: tempo, custo, alocação de pessoas e escopo variável. Assim, foi apresentado como cada metodologia comporta-se diante dos quesitos estudados, para com isso chegar-se a conclusão se a metodologia atende cada parâmetro analisado.

De acordo com estudos e o resumo da análise apresentado na Tabela 1 chega-se a conclusão de qual é metodologia mais adequada de acordo com cada parâmetro.

O FDD e XP atendem muito bem ao quesito *custo*, utilizando-se do método análise de pontos de função, que coleta dados do projeto para que com isso possa chegar ao valor do custo. O Scrum e XP atende ao quesito *alocação de pessoas*, pois cada integrante da equipe é definido com um papel específico no projeto. Por outro lado, isto não acontece no FDD, o que é um ponto negativo do FDD, pois o desenvolvedor pode ficar com mais de uma tarefa. Isto que para muitos é considerado como agilidade no processo também pode gerar sobrecarga de trabalho, implicando em atrasos no projeto.

Todas as metodologias atenderam bem ao quesito *escopo variável*, sendo assim fica a critério do desenvolvedor escolher a que se adapta melhor ao projeto.

Por fim chega-se ao Kanban que é a menos indicada para os quesitos analisados. Esta foi analisada com baixa aderência em praticamente todos os quesitos. Nesta pesquisa o Kanban atendeu apenas o quesito *escopo variável*, diante disso seria indicada apenas para auxiliar na organização do projeto.

## REFERÊNCIAS BIBLIOGRÁFICAS

BARBOSA, Antonio; AZEVEDO, Bruno; PEREIRA, Bruno, CAMPOES, Pedro; SANTOS, Pedro. **Metodologia Ágil: Feature Driven Development**, 2007.

CUNHA, Cristiane Ribeiro; FILIPAKIS, Cristina D'Ornellas. **Proposta de Utilização de FDD e APF para melhoria do Processo de Software**: Encontro de Computação e Informática do Tocantins, p. 142-151,2012.

GAMBA, Mateus Luiz; BARBOSA, Ana Cláudia Garcia. **Engenharia de Software - Aplicação de Métricas de Software com Scrum**. Anais SULCOMP, v. 5, 2010.

HAZAN, Claudia; VON STAA, Arndt. **Análise e Melhoria de um Processo de Estimativas de Tamanho de Projetos de Software**. Trabalho de Conclusão de Curso; Ciência da Computação; Pontífica Universidade Católica do Rio de Janeiro, 2005.

KNIBERG, Henrik; SKARIN Matias. **Kanban e Scrum obtendo o melhor de ambos**. Estados Unidos da América, 2009.

PRESSMAN, Roger. **Engenharia de Software uma Abordagem Profissional**. 7ª ed. Porto Alegre, 2011.

SILVA LOPES, Jhoney. **Guia Prático em Analise de Ponto de Função**, 2011.

SILVA, Diogo Vinícius de S.; SANTOS, F. Alan de O.; NETO, Pedro Santos. **Os benefícios do uso de Kanban na gerência de projetos de manutenção de software**. VII Simpósio Brasileiro de Sistemas de Informação, 2012.

SILVA,F.G;HOENTSCH,L.Silva. **Uma análise das Metodologias Ágeis FDD e Scrum sob a perspectiva do modelo de qualidade MPS.BR**, vol 5, num 12, 2009.

SOUZA, Luciano M. **Método ágil XP (extreme programming)**. Revista Eletrônica da FIA, v. 3, n. 3, p. 3, 2007.

# UM RELATO DE EXPERIÊNCIA UTILIZANDO O *OPENUP* NO DESENVOLVIMENTO DE UM APLICATIVO MÓVEL

*[16]José Risoaldo Nóbrega da Silva Filho; [2]Tulio Vidal Rolim; [3]Adriano Cândido Lima, [4]Vitória Regina Nicolau Silvestre, [5]Hugo Silva Nascimento*

## INTRODUÇÃO

Com o passar dos anos o mundo está ficando cada vez mais informatizado, consequentemente a demanda por *softwares* tem aumentado; junto a isso problemas referentes ao desenvolvimento de *software* são mais visíveis. Dentre estes, apresenta-se a falta de processos que supram as demandas de tempo e formalização.

O *RUP* (*Rational Unified Process*) é um processo bem definido e validado tanto na literatura como no mercado, tornando-se o principal processo em muitas organizações. Entretanto, o *RUP* é também complexo, podendo gerar dificuldades na implementação de uma organização em razão do seu tamanho, dificuldade e curva de aprendizagem (KRUCHTEN, 2004).

Diante disso, cada vez mais as empresas do mercado atual estão aderindo ao uso de metodologias ágeis, visto que além da melhora no processo de fabricação, são perceptíveis inúmeras outras contribuições às empresas (GATTO, 2015). Dentre inúmeras metodologias ágeis existentes no mundo, as mais bem avaliadas segundo Carreira (2017) são: *Scrum*, *XP* (*eXtremeProgramming*) e *Lean*.

As metodologias ágeis, contudo, não são a cura para todas as dores de um processo de *software*, e assim, as equipes de *software* ainda continuam falhando. Ser ágil exige uma mudança não só na mentalidade, mas também nas atitudes em toda uma organização. Porém, nem todas as organizações estão prontas para essa mudança cultural. Alguns problemas comuns podem ser observados em projetos: *a)* A equipe do projeto não compartilha uma visão clara de como o sistema será visto por seus usuários; *b)* Os requisitos não conduzem o trabalho de desenvolvimento; *c)* A arquitetura do sistema não foi articulada; *d)* Os planos não estão conectados à realidade da engenharia. *e)* Os riscos são ignorados. Com base nisso, a IBM projetou o *OpenUP* (*Open Unified Process*), como forma de resolver as principais problemáticas apresentadas nos projetos de *software*.

[16] Graduado em Análise e Desenvolvimento de Sistemas (FVS)
[2] Discente de Mestrado em Ciência da Computação (UFC)
[3] Docente na Faculdade Vale do Salgado (FVS) - Discente em Ciência da Computação (UFC)
[4] Discente de Mestrado em Ciência da Computação (UFC)
[5] Discente em Análise e Desenvolvimento de Sistemas (FVS)

A intenção do *OpenUP* consiste em colher o que há de melhor nas metodologias ágeis e no *RUP*. Seu processo de desenvolvimento é fundamentado justamente no princípio ágil, constituindo-se de um processo de desenvolvimento de *software* iterativo mínimo, contendo apenas o que nele é fundamental. É também completo e extensível, sendo fundamentado na estrutura do *RUP*, que por sua vez atende todas as necessidades para o desenvolvimento de um *software* (KROLL; MACISAAC, 2006).

Em frente às possibilidades oportunizadas, o objetivo do estudo consiste em descrever um relato de experiência durante a construção de um aplicativo chamado "Guia Médico" com base na utilização do *OpenUP*.

As demais seções presentes no trabalho estão organizadas da seguinte maneira: a seção 2 apresenta com mais detalhes o processo *OpenUP*; a seção 3 expõe a definição do escopo do estudo; a seção 4 contém a metodologia empregada e seus detalhes; a seção 5 explana e discute acerca dos principais resultados; e, por fim, a seção 6 apresenta a conclusão final acerca do trabalho.

## *OPENUP*

O *OpenUP* é uma adição bem-vinda ao panorama do processo de *software*: é ágil, tanto em sua direção como em "espírito", estando também ao mesmo tempo documentado. O objetivo principal do *OpenUP* era criar uma abordagem ágil ao *RUP*, usufruindo ao mesmo tempo de todas as vantagens e valores que todos gostam em outros processos e metodologias ágeis (KROLL; MACISAAC, 2006). Em sua composição, o *OpenUP* define um conjunto de regras, papéis, artefatos e tarefas que devem ser seguidos pelo time de desenvolvimento

O modelo de Ciclo de Vida do *OpenUP* visto na Figura 1 fornece o processo de governança para iterações e micro incrementos. Sua divisão é feita em 3 etapas: *a) Concepção:* responsável pela visão geral do projeto, isto é, definir o escopo e os objetivos; *b) Elaboração:* consiste em dirigir a maior parte dos esforços ao entendimento do projeto, levantamento de requisitos e as modelagens conceituais e comportamentais; e *c) Construção:* responsável por transformar as etapas a) e b) no produto (*software*).

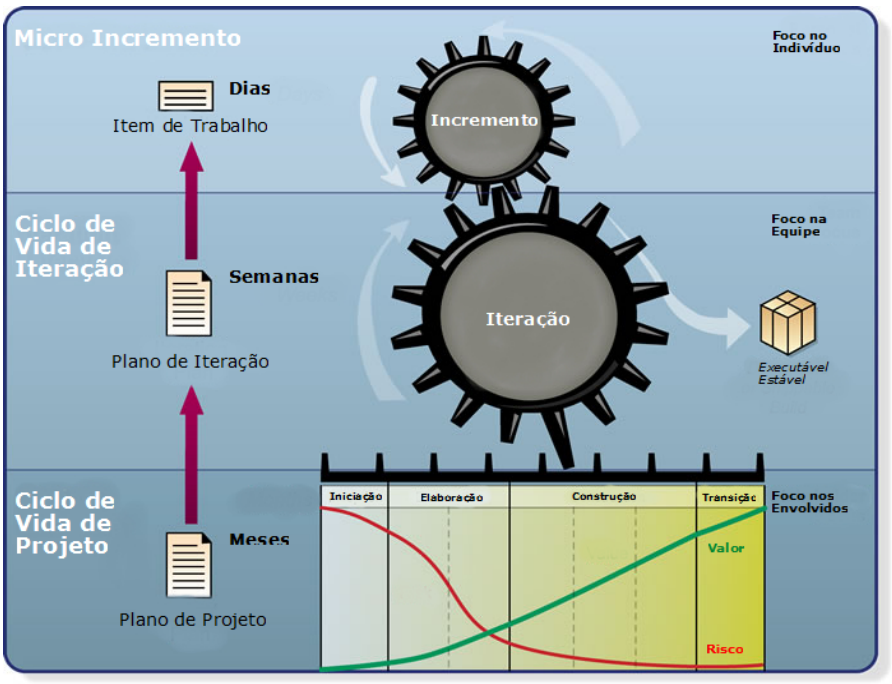



**Figura 1** - Ciclo de vida OpenUP.
**Fonte:** ECLIPSE, 2007.

## APLICATIVO GUIA MÉDICO

Em análise na busca de algumas problemáticas na região centro-sul do Ceará, observou-se que uma das áreas mais precárias é a saúde. Com base nisso, cogitou-se a possibilidade de analisar o acesso aos profissionais médicos, visto que muitos deles não possuíam contato público, fazendo com que houvesse o contato mediado por uma clínica ou hospital particular.

Assim, um método para facilitar os atendimentos pessoais por parte dos médicos, seria através de um ambiente tecnológico que fornecesse os meios para busca e contato. Por conseguinte, apresentou-se o aplicativo *"Guia Médico"*.

O *Guia Médico* é um aplicativo para dispositivos móveis que permite a localização de médicos em um dado município, disponibilizando posteriormente a marcação de consultas para usuários habitantes na região Centro-Sul do Ceará. A Figura 2 apresenta a tela inicial do *Guia Médico*, contudo, apresentar a interface, bem como os artefatos do aplicativo, não são o foco deste trabalho.


**Figura 2** - Interface Inicial do Aplicativo Guia Médico.
**Fonte:** Elaborado pelos autores, 2018.

## METODOLOGIA

A metodologia empregada neste trabalho consiste em um relato de experiência utilizando o *OpenUP* na construção e desenvolvimento do aplicativo *Guia Médico* abordado na seção anterior.

Os relatos de experiências apresentam-se como uma descrição de uma determinada vivência ou experiência, seja ela bem sucedida ou não. Em suma, o relato é um texto descritivo que expõe pontos precisos de uma determinada experiência, apresentando as impressões ou considerações que aquela experiência trouxe para os envolvidos, sendo este um importante método para a divulgação do conhecimento adquirido em um determinado processo (GANCHO, 2004).

## RESULTADOS E DISCUSSÕES

Os resultados identificados durante a etapa de construção do aplicativo *Guia Médico* serão expostos de forma organizada em 3 subtópicos, sendo: *1) Concepção; 2) Elaboração* e *3) Construção,* conforme explicadas na seção *OpenUP*. A implantação e coleta de dados com algum usuário fim não foram apresentadas no trabalho por não fazerem parte do propósito deste estudo.

### Concepção

O processo descrito nesta etapa foi mínimo e completo. Os analistas formularam a intenção do projeto com base em uma problemática levantada (previamente discutida na seção Aplicativo *Guia Médico*), gerando posteriormente a intenção e visão do sistema. Dentre os principais achados nesta etapa estão: *1)* Identificou-se que através da adoção do *OpenUP* foi possível uma maior proximidade com os *stakeholders* (pacientes e médicos), maximizando assim valores a estes interessados. *2)* Foi possível estabelecer grande parte das estruturas fundamentais do projeto a curto prazo.

Analisando os itens 1 e 2, notou-se que houve uma relação entre a proximidade dos *stakeholders* e a agilidade apresentada na definição dos objetivos bem como do escopo. Corroborando com isto, trazer os *stakeholders* para próximo do processo faz com que haja uma maior colaboração com a equipe de desenvolvimento, garantindo que sua necessidade, problemas e potenciais detalhes sejam repassados (KROLL; MACISAAC, 2006).

### Elaboração

A fase de elaboração consistiu principalmente no levantamento, especificação e aprimoramento dos requisitos. Para tal, as funcionalidades foram definidas através de

entrevistas com 15 pessoas. A amostra se deu com base em Virzi (1992) e Nielsen (1993), que afirmam dizendo que 100% dos problemas de usabilidade podem ser encontrados com 15 participantes, tendo sido 10 usuários e 5 médicos, instigando assim que posteriormente esses participantes pudessem validar aspectos de usabilidade e de requisitos por meio da prototipagem. Inicialmente almejava-se também realizar a validação *in loco* após a construção, entretanto, os autores resolveram não elevar o trabalho a tal ponto em razão de sua inviabilidade quanto ao tempo e alguns recursos necessários.

Após levantados, os requisitos foram classificados de acordo com o grau de importância aos *stakeholders*, dividindo-se em *(1)* Desejável - *(2)* Importante e *(3)* Extremamente Importante. Esta etapa foi desenvolvida conforme orientações do *OpenUP*, demonstrado por Eclipse (2007) — o levantamento de requisitos deve compreender os principais requisitos dos *stakeholders,* comunicando-os ao time de desenvolvimento. Nesta etapa, o *OpenUP* pode contribuir de forma significativa na definição da arquitetura, tendo sido definida como uma atividade simples, razão justificável por adotar um processo iterativo e suas atividades incrementais que objetivaram apenas os requisitos, regras e restrições do projeto, não necessitando de etapas e atividades que demandam um alto custo de tempo.

**Construção**

Em razão do gerenciamento ter sido iterativo, muitas das atividades da equipe eram direcionadas com base nas iterações. Na fase de Construção, a equipe se comprometeu com a entrega de partes do código como forma de satisfazer incrementalmente os requisitos. As entregas aos *stakeholders* foram feitas como forma de validação das funcionalidades, sendo divididas de acordo com as prioridades estabelecidas na etapa de Elaboração. A equipe de desenvolvimento inclui arquitetos, desenvolvedores e testadores.

Como problemas, foi identificado uma maior dificuldade por parte dos desenvolvedores no que tange à adoção do *OpenUP*. Este fator pode ser justificado pelo fato de que o *OpenUP* demonstrou-se como ágil porém formalizado, e assim ocorreram algumas inadequações ao caráter híbrido do processo. Coube aos desenvolvedores primeiramente selecionar uma tarefa. Posteriormente, os mesmos desenvolvedores davam início à programação em par, ambos utilizando um mesmo computador contendo o ambiente e as ferramentas necessárias para o desenvolvimento, seguindo a prática presente no *XP*.

Cabe ressaltar que os desenvolvedores do time também não possuíam experiência quanto à real adoção de processos ágeis, o que demandou tempo até atingirem um nível de familiarização aceitável por parte de todos.

## CONCLUSÕES

Neste trabalho foram abordados conceitos voltados para o desenvolvimento de um aplicativo com base no processo e no modelo de ciclo de vida do *OpenUP*. Primeiramente, o time buscou estudar de forma teórica o *OpenUP* a fim de compreender suas etapas e funcionamento. Com base nisso, realizou-se uma análise das principais problemáticas da região centro-sul do Ceará, e com isso foi estabelecido o desenvolvimento de um projeto específico: o *Guia Médico*, um aplicativo voltado para a localização e agendamento de profissionais médicos.

Foram relatados os principais aspectos positivos e negativos encontrados durante a etapa de construção do aplicativo. Assim, mediante análise, sua adequação se mostrou eficaz, tendo como resultados uma experiência satisfatória no emprego das suas etapas e organização do processo.

## REFERÊNCIAS


ARAÚJO, Alex de. **Manifesto Ágil e as TOP 3 Metodologias Ágeis de Desenvolvimento digital**. Disponível em:<https://becode.com.br/manifesto-agil-e-top-metodologias-ageis/>. Acesso em: 01 maio. 2018.

ECLIPSE. **EPF Published Websites Downloads**. Disponível em: **<**http://www.eclipse.org/downloads/download.php?file=/technology/epf/OpenUP/published/OpenUP_published_PT_1.0_20070801.zip. Acesso em: 07 maio. 2018.

GANCHO, Cândida Vilares. **Como analisar narrativas**. Editora Ática, 2004.

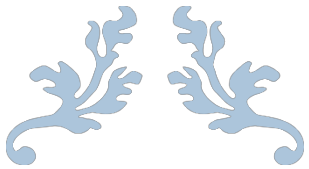
GATTO, Alves. **A importância da metodologia ágil na era da transformação digital**. Disponível em:<http://www.administradores.com.br/noticias/negocios/a-importancia-da-metodologia-agil-na-era-da-transformacao-digital/102467/>. Acesso em: 01 maio. 2018.

KROLL, Per; MACISAAC, Bruce. **Agility and Discipline Made Easy: Practices from OpenUP and RUP**. Pearson Education, 2006.

KRUCHTEN, Philippe. **The rational unified process: an introduction**. Addison-Wesley Professional, 2004.

NIELSEN, Jakob; LANDAUER, Thomas K. A mathematical model of the finding of usability problems. In: **Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems**. ACM, 1993. p. 206-213.

VIRZI, Robert A. Refining the test phase of usability evaluation: How many subjects is enough?. **Human factors**, v. 34, n. 4, p. 457-468, 1992.



# TECNOLOGIAS EMERGENTES E ENGENHARIA DE SOFTWARE

# INTERNET DAS COISAS: UMA REVISÃO DA LITERATURA

*Maria Thaís Alves Barros[17]; Hugo Silva Nascimento[18]; Mailson Pinheiro Alves[19]; Tulio Vidal Rolim[20]; João Carlos da Cruz de Lima[21].*

## INTRODUÇÃO

Ao se falar do acesso à internet nos dias de hoje, já imaginamos algo que seria simplesmente indispensável para o nosso cotidiano pode-se afirmar que uma simples manhã sem estarmos conectados, passa certa sensação de omissão de informações, o que seria um verdadeiro caos na vida de milhões de pessoas ao redor do mundo. Onde é justificável, pois a principal fonte de conhecimento destas pessoas sobre tudo que acontece à sua volta, depende exclusivamente do que é enviado para rede.

A internet contribui para expandir o acesso à informação atualizada, permite estabelecer novas relações com o saber que ultrapassam os limites dos materiais instrucionais tradicionais, favorece a criação de comunidades colaborativas que privilegiam a comunicação e permite eliminar os muros que separam a instituição da sociedade (ALMEIDA, M. E. B. 2003). Em todas as definições conceituais que se busca sobre a internet é definida como um conjunto de redes mundial, originária de dois termos: *inter* (internacional) e *net* (rede). Essas redes estão interligadas e possibilitam o acesso a um mundo de informações, serviços, vendas online, divulgação de serviços e produtos, notícias, documentários e uma infinidade de lugares onde se pode ir, sem sair do conforto de casa onde sua utilidade é em tal intensidade e em demasia, que pessoas se conhecem ao redor do mundo, sem sequer precisar sair até a esquina, quebrando assim, tanto as fronteiras geográficas, como as culturais, as políticas e econômicas.

Segundo Carr (2011) ela é um instrumento que possibilita expressar ideias, mostrando a personalidade individual, além de criar vínculos com os demais através das chamadas redes sociais tanto no lado pessoal bem como no lado profissional, estreitando as relações de trabalho, ampliados através de milhões de redes particulares, tornando-se assim fenômeno da comunicação em empresas, universidades, órgãos executivos e inúmeros outros. Desde o seu surgimento, a internet vem alcançando o maior número de usuários e acessos, tornado a vida mais rápida, fácil e dinâmica. Nela as informações fluem tanto quanto as notícias se propagam em tempo real armazenando dados sobre qualquer pessoa, assunto e lugar deixando assim disponível e no alcance sempre que for preciso.

[17] Graduada em Análise e Desenvolvimento de Sistemas. FVS. thaislk12@gmail.com.
[18] Graduando em Análise e Desenvolvimento de Sistemas. FVS. hugosilva05512@gmail.com.
[19] Graduando em Análise e Desenvolvimento de Sistemas. FVS. mailsonpalves@gmail.com.
[20] Mestrando em Ciência da Computação. UFC. tulio.xcrtf@gmail.com.
[21] Mestrando em Ciência da Computação. UFC. carlos@fvs.edu.br.

Devido à evolução da internet, surge a Internet das Coisas (*IoT - Internet Of Things*), um novo conceito tecnológico que alia computação e comunicação a objetos físicos e virtuais usados no nosso cotidiano, conectando ainda mais o homem e a rede (LACERDA e LIMA-MARQUES, 2015). O que tem colaborado com esse crescimento é o grande número de equipamentos dos mais variados tipos capazes de usar a rede, como smartphones, geladeiras, fogões, computadores entre outros, acentuando a agilidade, a flexibilidade e rapidez do uso da internet no cotidiano das pessoas, das fábricas, empresas e de ambientes, inclusive, ambientes domésticos e públicos em geral. Esse novo fenômeno que se utiliza da nanotecnologia para produzir equipamentos eletrônicos e portáteis com uso de chips, tornando-os autômatos inteligentes tem como principal objetivo atender e oferecer serviço à sociedade em geral.

Como salienta Atzori, Iera e Morabito (2010), são muitas as possibilidades de aplicações da internet das coisas, sendo assim Rodrigues e Kleinschmidt (2014) afirmam que, no transporte bem como a logística a IoT é utilizada na identificação, rastreamento e controle de produtos transportados e em estoque. Já se tratando da grande área da saúde, se pode apresentar possibilidades de identificação de pacientes e controle de dosagens em medicamentos, como por exemplo, uma proposta de sistema de IoT para monitoramento de ambiente hospitalar observado em detalhes na obra de Cantanhede e Da Silva (2014). Em aspectos comuns da sociedade também informam sobre as casas e prédios inteligentes, tendo como funções o controle de iluminação, economia de energia, monitoramento inteligentes e etc. Rodrigues e Kleinschmidt (2014) ainda apresentam um trabalho contendo um sistema completo de IoT utilizado em alguns cenários para ambientes acadêmicos contando com toda uma arquitetura baseada em dispositivos conectados a redes sem fio.

Muito se fala em *IoT* e como essa tecnologia irá revolucionar o modo como as pessoas interagem com a sociedade, porém, essas mudanças podem ser muito abstratas para o real entendimento de seu impacto, principalmente para pessoas que estão fora do contexto da tecnologia.

O presente trabalho traz uma análise de forma objetiva através da introdução de conhecimentos básicos sobre a tecnologia de *IoT,* e sua utilização em diversos contextos e cenários, a fim de difundir, de forma clara e objetiva, a compreensão do impacto e das possibilidades dessa tecnologia.

Isto faz com que pessoas leigas acerca do tema possam adquirir uma base teórica, conhecendo como a *IoT* está aos poucos mudando a sociedade. Sendo assim, neste trabalho tem-se como principal objetivo analisar as principais características, motivações e desafios na utilização da Internet das Coisas, através de uma revisão da literatura.

## METODOLOGIA

Este trabalho discorre acerca de um estudo exploratório, para isto, foi utilizada uma pesquisa bibliográfica. Conforme Gil (2017), é possível entender como uma pesquisa exploratória uma busca para obter uma perspectiva geral, de perspectiva aproximada, de uma situação delimitada.

No que diz respeito a pesquisa bibliográfica, ela propõe-se a identificar princípios norteadores e trabalhos científicos recentes que exploram o tema proposto em determinada pesquisa, com isso, apoiar as considerações desta pesquisa em autores já destacados e que possuem autoridade sobre o tema delimitado (MOTTA-ROTH e HENDGES, 2010).

Na fundamentação da pesquisa bibliográfica foram investigados trabalhos de autores que disseminam as temáticas de redes de computador e IoT, tendo foco principal nos trabalhos que abordaram a temática de forma conceitual e aplicada, procurando detectar as principais convicções, que orientam o restante da pesquisa.

Para a inclusão dos trabalhos, foram escolhidos aqueles que mostraram textos coerentes e pesquisas na área de redes de computadores voltadas para *IoT*, artigos completos e publicados em língua portuguesa e inglesa. Foram utilizadas bases de dados como: Revista de Informática Aplicada, *IEEE* e *SBC Journal on Interactive Systems*. Tendo a escolha dos autores baseado no trabalho realizado condizente com o tema da pesquisa, autoridade na área e tipo de trabalho realizado concedendo preferência para artigos completos.

Como critérios de exclusão foram estabelecidos: artigos incompletos e trabalhos que fogem da temática da pesquisa ou não agregam em conteúdo teórico para a formação dos conceitos básicos. Os descritores de buscas foram: Internet das Coisas, *Internet of Things* e *IoT*, sendo que somente foram selecionados para leitura, artigos com temática semelhante ou exata na área da pesquisa, evitando assim redundância de tempo devido a demasia de artigos encontrados para a construção do trabalho.

Grande parte dos artigos foram excluídos da leitura e pesquisa, sendo estes artigos que fogem da área ou tema proposto, temáticas redundantes ou já abordadas no trabalho, trabalhos que ultrapassem 10 anos de publicação com exceção de referências de autoridades na área e principalmente trabalhos com falta expressiva de informações de caráter teórico à respeito dos conceitos básicos sobre Internet das Coisas.

## RESULTADOS E DISCUSSÕES

Utilizando as bases de dados *IEEE*, *SBC Journal on Interactive* e Revista de Informática Aplicada foram encontrados diversos resultados utilizando as *Strings* de busca especificadas, e

a partir dos resultados coletados é notória a opinião de diversos autores sobre os temas contidos nas *Strings* de busca, onde se segue descrevendo as opiniões resultantes da mesma.

Lacerda e Lima-Marques (2015) apresenta que o termo "Internet das coisas" tem sua origem no inglês *"Internet of Things*", referindo-se a uma Revolução tecnológica cuja finalidade é conectar dispositivos eletrônicos em equipamentos e espaços usuais no cotidiano das pessoas, cujo desenvolvimento e funcionamento depende de técnicas inovadoras. Desde os eletroportáteis, eletrodomésticos, máquinas industriais, transportes e outros meios de serviços, poderão receber sensores *wireless,* denominados como equipamentos com inteligência artificial e com nanotecnologia.

Dessa forma, os dispositivos eletrônicos têm como mola propulsora, estabelecer a comunicação entre si, como se fosse uma conversa, para isso, usam a internet como mecanismo de comunicação.

Para efetivação de uso da Internet das Coisas, Lazarescu (2013) salienta que alguns protocolos foram adotados, dentre eles: O Protocolo *IP* e o *TCP* são os principais dentre um conjunto de outros que direcionam o funcionamento da internet neste campo.

Mediante a evolução da internet das coisas, Zhao e Ge (2013) afirma que haverá também alguns problemas, como o risco da falta de segurança aos usuários. Com o papel de não apenas revolucionar o modo de vida das pessoas, a Internet das coisas de acordo com Xiaohui (2013), poderá também trazer insegurança quando estes dispositivos forem invadidos por pessoas de má fé ou *hackers*, pois muito embora haja investimentos para essa proteção, estes também evoluirão para fugir ao controle das empresas.

A imensa demanda de uso simultâneo da internet das coisas ao redor do mundo incorrerá numa avalanche de informações a partir de dados pessoais dos usuários, o que já se faz previsões de problemas, pois com a vida cada vez mais informatizada, demandará disponibilidade na internet do maior número de informações sobre suas vidas pessoais, hábitos, rotinas e interesses, terminando por subtrair sua privacidade e expor os usuários em situações de riscos.

Entretanto, os avanços presenciados na atualidade em relação à internet das coisas (*IoT*), conforme afirma Da Xu, He e Le (2014), proporcionam uma imensa oportunidade para resolver os grandes desafios que são enfrentados pelos indivíduos da sociedade, onde o autor ressalta a problemática do transporte veicular e as possíveis soluções utilizando os recursos da internet das coisas.

A partir destes avanços, Feki et al. (2013) ressalta que os pesquisadores da área seguem com parceria aos empresários, agências governamentais e até mesmo cidadãos explorando tais tecnologias que cercam a internet das coisas se fundamentando em três perspectivas principais:

a teoria científica, projeto de engenharia e a experiência do usuário final a fim de construir o que o próprio autor denomina como a próxima revolução tecnológica.

## CONCLUSÕES

O presente trabalho aborda de forma sintética a presença da internet das coisas na vida das pessoas comuns, empresários, órgãos públicos e em diversos setores da economia e da política, enquanto ferramenta inovadora capaz de tornar fácil, acessível, prática e inteligente o uso de objetos e ferramentas a partir de conexão com a rede mundial de computadores, a internet.

As discussões, afirmações e conceitos técnicos apresentados decorrem de uma pesquisa teórica e bibliográfica obtida nos sites apresentados, bem como são amplas, atuais e fomentam o aprofundamento do estudo para leitores interessados e demais pesquisadores do assunto. Não é um tema fácil de compreender, mas, sobretudo, requer interesse e buscas contínuas acerca do mesmo, uma vez que, por ser algo dinâmico e atual, é crescente o número de publicações que fomentam a vontade de compreendê-lo. Isto a partir da ótica da produção de softwares e hardwares que possam ensinar como usar e produzir a internet das coisas.

Nessa lógica, uma tecnologia inteligente estará acessível no cotidiano das pessoas de todos os níveis e classes sociais quando dispositivos eletrônicos terão a finalidade de estabelecer a comunicação entre si, como se fosse uma conversa, para isso, usam a internet como mecanismo de comunicação. Assim a Internet das Coisas envolverá a indústria, órgãos reguladores e de padronização e entidades acadêmicas e de pesquisa de todo o mundo, facilitando as consultas, pesquisas, transações comerciais, ações corriqueiras do cotidiano, com a mesma facilidade com que se conecta e desconecta-se um *plugin* da tomada convencional.

Como trabalhos futuros, espera-se realizar uma revisão mais ampla na literatura a fim de estipular projeções da *IoT* para os próximos anos, bem como realizar um aparato de trabalhos que expressem resultados satisfatórios nos âmbitos da educação, saúde e cidadania.

## REFERÊNCIAS


ALMEIDA, Maria Elizabeth Bianconcini de. Tecnologias e gestão do conhecimento na escola. **Gestão Educacional e Tecnologia. São Paulo: Avercamp**, p. 113-130, 2003.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The internet of things: A survey. **Computer networks**, v. 54, n. 15, p. 2787-2805, 2010.

CANTANHEDE, Romulo Fagundes; DA SILVA, Carlos Eduardo. Uma Proposta de Sistema de IoT para Monitoramento de Ambiente Hospitalar. **Anais da VII Escola de Computação e suas Aplicações-Época**, 2014.

CARR, Nicholas. A geração superficial: o que a internet está fazendo com os nossos cérebros. **Rio de Janeiro: Agir**, 2011.

DA XU, Li; HE, Wu; LI, Shancang. Internet of things in industries: A survey. **IEEE Transactions on industrial informatics**, v. 10, n. 4, p. 2233-2243, 2014.

FEKI, Mohamed Ali et al. The internet of things: the next technological revolution. **Computer**, v. 46, n. 2, p. 24-25, 2013.

GIL, A. C. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: Atlas, 2017.

LACERDA, Flavia; LIMA-MARQUES, Mamede. Da necessidade de princípios de Arquitetura da Informação para a Internet das Coisas. **Perspectivas em Ciência da Informação**, v. 20, n. 2, p. 158-171, 2015.

LAZARESCU, Mihai T. Design of a WSN platform for long-term environmental monitoring for IoT applications. **IEEE Journal on emerging and selected topics in circuits and systems**, v. 3, n. 1, p. 45-54, 2013.

MOTTA-ROTH, Désirée; HENDGES, Graciela H. Produção textual na universidade. São Paulo: **Parábola Editorial**, 2010.

RODRIGUES, Fábio Figueiredo; KLEINSCHMIDT, João Henrique. Estudo de Aplicações da Internet das Coisas em um Ambiente Acadêmico. **Revista ESPACIOS**| Vol. 35 (Nº 13), 2014.

XIAOHUI, Xu. Study on security problems and key technologies of the internet of things. In: **Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on**. IEEE, 2013. p. 407-410.

ZHAO, Kai; GE, Lina. A survey on the internet of things security. In: **Computational Intelligence and Security (CIS), 2013 9th International Conference on.** IEEE, 2013. p. 663-667.

# ONTOLOGIAS DE DOMÍNIO EM LINHAS DE PRODUTO DE SOFTWARE: UM MAPEAMENTO SISTEMÁTICO

*[22]Tulio Vidal Rolim; [2]Vitória Regina Nicolau Silvestre; [3] Caio Viktor da Silva Ávila; [4]Matheus Mayron Lima da Cruz*

## INTRODUÇÃO

O termo *"Ontologia"* é utilizado na filosofia como uma disciplina filosófica, sendo definida como uma teoria particular acerca da natureza do ser ou da existência das coisas. Na computação, entretanto, o termo é referido como *"ontologia"* (com *"o"* minúsculo), e sua definição atualmente aceita na literatura foi proposta por Studer et al. (1998) ao dizer que a ontologia é "uma especificação formal e explícita de uma conceituação compartilhada". As ontologias na computação são organizadas em 4 tipos: fundamentação, domínio, tarefa e aplicação (GUARINO, 1998).

A utilização das ontologias de domínio em Linhas de Produto de Software (LPS) podem promover a reutilização de artefatos através da validação pelo domínio. Este fato auxilia na resolução de problemas relacionados ao desenvolvimento de softwares em domínios altamente variáveis, o que por sua vez resulta em dificuldades de entrega e manutenção (FERREIRA; MACHADO; GASEVIC, 2009). Contudo, a utilização de ontologias na representação de domínios ainda é pouco vista associada à LPS quando se comparada com outras representações de processos específicos, inclusive, de forma errônea alguns times de desenvolvimento se quer distinguem esta etapa da análise e levantamento de requisitos.

O objetivo deste estudo consiste em realizar um mapeamento de trabalhos na literatura como forma de identificar e compreender o estado atual do uso de ontologias de domínio no contexto de linhas de produto de software.

## METODOLOGIA

A metodologia empregada consiste em um Mapeamento Sistemático (MS). O MS tem como propósito realizar uma pesquisa com abrangência em largura e não adota como objetivo fazer uma pesquisa em profundidade na literatura, tendo como resultados fornecer uma visão ampla de determinada área pesquisada, buscando identificar características tais como: tipo da pesquisa, constância em período nas publicações e resultados apresentados (KITCHENHAM; CHARTES, 2007; PETERSEN et al., 2008).

[22] Discente de Mestrado em Ciência da Computação (UFC)
[2] Discente de Mestrado em Ciência da Computação (UFC)
[3] Discente de Mestrado em Ciência da Computação (UFC)
[4] Discente de Mestrado em Ciência da Computação (UFC)

O MS é organizado de acordo com o Protocolo de Mapeamento Sistemático (PMS). Neste protocolo são apresentados a descrição da condução do trabalho, processo avaliativo e de extração dos resultados. O PMS desta pesquisa é dividido em duas subseções: Planejamento *{Questões de pesquisa, Seleção de Fontes e String de Busca, Critérios de Inclusão e de Exclusão}* e Condução.

**Planejamento**

*Questões de Pesquisa*

As questões de pesquisa são identificadas respectivamente por *RQ1*, *RQ2*, *RQ3* e *RQ4* e foram definidas da seguinte maneira: *(RQ1)* Qual foi a frequência de publicação dos trabalhos envolvendo ontologias de domínio para linhas de produto de software? *(RQ2)* Dentre as conferências, quais foram as que contiveram a maioria das publicações? *(RQ3)* Quais foram os autores que mais efetuaram publicações de trabalhos relativos ao tema? *(RQ4)* Dentre as Instituições dos pesquisadores, quais apresentaram maiores quantidades de publicação?

*Seleção de Fontes e String de Busca*

A estratégia adotada no processo de busca foi organizada da seguinte maneira: *i)* Busca com Strings experimentais em uma das bases de busca; *ii)* Análise dos resultados retornados; *iii)* Aprimoramento da String; *iv)* Realização de novas Buscas; *v)* Definição da String Oficial; *vi)* Realização de Busca com base em *v)*; *vii)* Exportação dos Resultados em formato *.tex*. As Fontes de Busca definidas foram: *IEEE Xplore, Scopus, WebOfScience e ACM Digital Library*.

A *string* de busca inicial foi montada com base no PICO (*Population (P) – Intervention(I) – Comparator(C - Opcional) – Outcome(O)*), sendo definida como *P = {Software Project <or> Software Development Process <or> Software Product Line <or> Model-Driven Software Product Lines <or> Software Engineering Process <or> Software Design}, I = {Ontology Domain Model <or> Ontology Modeling <or> Ontology Modeling Requirements}, C {Ø}* e *O = {Ontology Diagrams <or> Ontology Artifacts <or> Ontology Architecture <or> Ontology-Based Product Lines <or> Ontology-Based Approach to Software Product Lines <or> Ontology Approachs <or> Ontology Framework <or> Ontology Create Tool <or> Ontology Tool}*. Entretanto, ao executar a busca na base *IEEE Xplore* foi retornado um erro em razão da quantidade de termos usados ter excedido o limite permissível pela base que é de 15 em uma mesma consulta.

Com base nesse aspecto e no *X* inicial de trabalhos encontrados ter se apresentado relativamente grande para análise, optou-se então pela definição de uma *string* que trouxesse

uma quantidade Y de estudos que pudessem ser analisados. Assim, a *string* oficial adotada foi: ((*Software Product Line) AND Domain Ontology)*.

*Critérios de Inclusão e de Exclusão*

Os critérios de inclusão definidos foram respectivamente: *a)* Período de publicação entre 2008 e 2018; *b)* Estudos sobre ontologias para linhas de *software*; já os de exclusão foram: *a)* Publicações diferentes de inglês e português; *b)* Livros, teses, dissertações, *workshops*, slides de apresentação, dentre outros que não artigos completos; e *c)* Estudos que não possuam pelo menos duas palavras da *string* de busca.

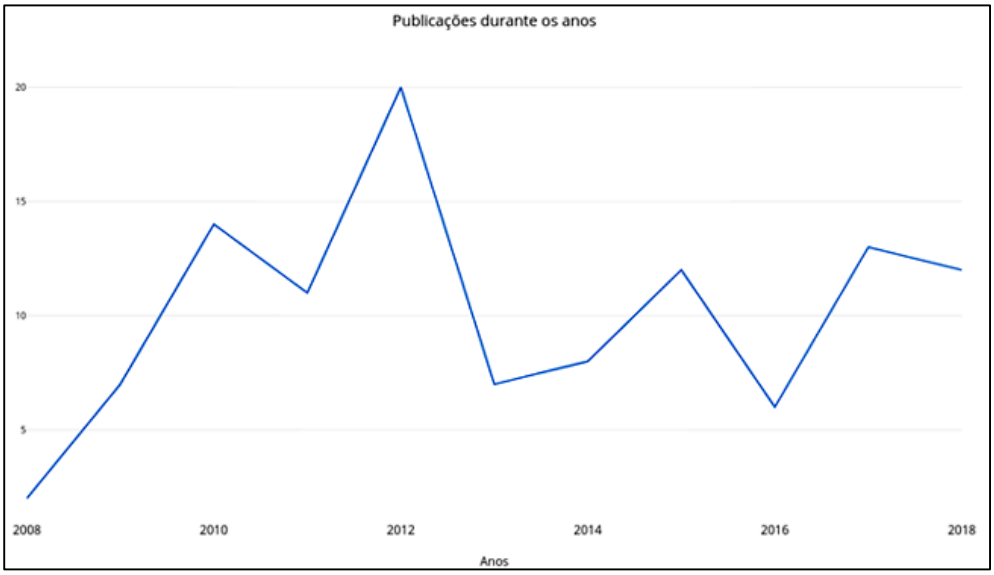
**Condução**

A realização da pesquisa ocorreu no dia 22 de abril de 2018. A busca na base do *WebOfScience* retornou 39 trabalhos, dentre os quais 6 foram excluídos pelos critérios de exclusão por meio das ferramentas de filtro, resultando em 33 artigos. No *Scopus* foram retornados 81 estudos, dentre os quais 39 artigos foram removidos, restando 42. Já na *IEEE Xplore* foram retornados 31 trabalhos, resultando em 27 após a remoção de 4 artigos. Por fim, a base que retornou menos resultados foi a *ACM Digital Library* trazendo apenas 11 artigos, do qual 1 foi removido com base nos critérios de exclusão, resultando em 10 artigos relacionados ao tema. A junção dos resultados totalizou uma quantidade de 112 trabalhos. Para cada um desses analisou-se o resumo, palavras-chaves e o *abstract*, em caso de dúvida, o trabalho era analisado de forma completa.

## RESULTADOS E DISCUSSÕES

Após execução de todas as etapas anteriores, os resultados encontrados foram organizados e representados de forma gráfica. Os achados visam apresentar as respostas para as questões *RQ1, RQ2, RQ3* e *RQ4*. Foram removidos das análises resultados menores, principalmente quando houve uma alta diversidade de dados, tais como nos Gráficos 2, 3 e 4. Para a RQ1 a busca consistia em identificar a frequência dos trabalhos relacionados ao uso das ontologias de domínio durante os anos conforme pode ser observado no Gráfico 1.

**Gráfico 1 -** Quantidade de Publicações por Conferência.



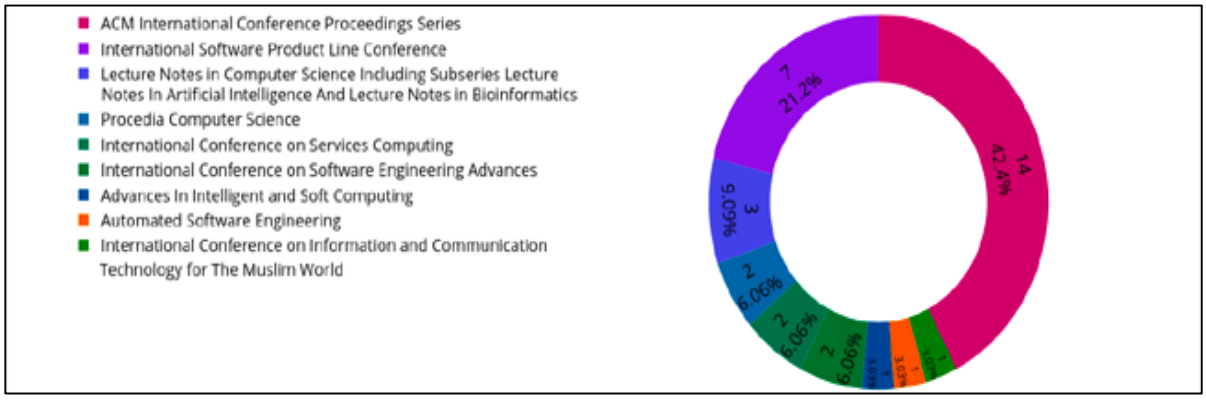

**Fonte:** Dados da pesquisa, 2018.

Os resultados apresentados no Gráfico 1 são respectivamente: 2008 *(2),* 2009 *(7),* 2010 *(14),* 2011 *(11),* 2012 *(20),* 2013 *(7),* 2014 *(8),* 2015 *(12),* 2016 *(6),* 2017 *(13),* 2018 *(12)*. Com base nisso, observa-se que há uma boa quantidade de estudos envolvendo o uso de ontologias de domínio no contexto de LPS em comparação aos últimos 10 anos. Tal afirmação pode ser justificada por Cheng et al. (2016) em razão da alta aplicabilidade das ontologias, conforme cita ao dizer que as ontologias são valorosas para outras áreas por desempenharem um papel significativo na análise, modelagem e implementação do conhecimento acerca do domínio.

O ano de 2012 foi o ano que mais estudos foram publicados, já os anos de 2011, 2013 e 2016 especificamente foram anos que tiveram decaídas em relação a anos anteriores. O ano de 2018 não pode ser projetado como um ano crítico, pois, apesar de ainda apresentar uma menor quantidade de trabalhos em relação a outros anos, até o presente momento da pesquisa (abril de 2018) já foram publicados 12 trabalhos. Observa-se assim, que provavelmente o ano de 2018 possa vir a ter uma maior quantidade que os anos de 2017 e 2010, vislumbrando alcançar ou até mesmo superar o ano de 2012.

As respostas da *RQ2*, observadas no Gráfico 2 mostram que as maiores publicações ocorreram em conferências internacionais, sendo distribuída principalmente em conferências de conceito A1 e A2. Dentre elas podem se destacar as conferências promovidas pela *Association for Computing Machinery (ACM)* e a *International Software Product Line Conference*. Outro fator perceptível é alta distribuição de conferências durante os anos, justificativa que pode ser apresentada com base na união de dois temas Ontologias e Linhas de Produto de Software, ambas áreas filhas de grandes áreas diferentes, Banco de Dados e Engenharia de Software, assim, há uma boa diversidade conferências disponíveis para submissão de trabalhos.

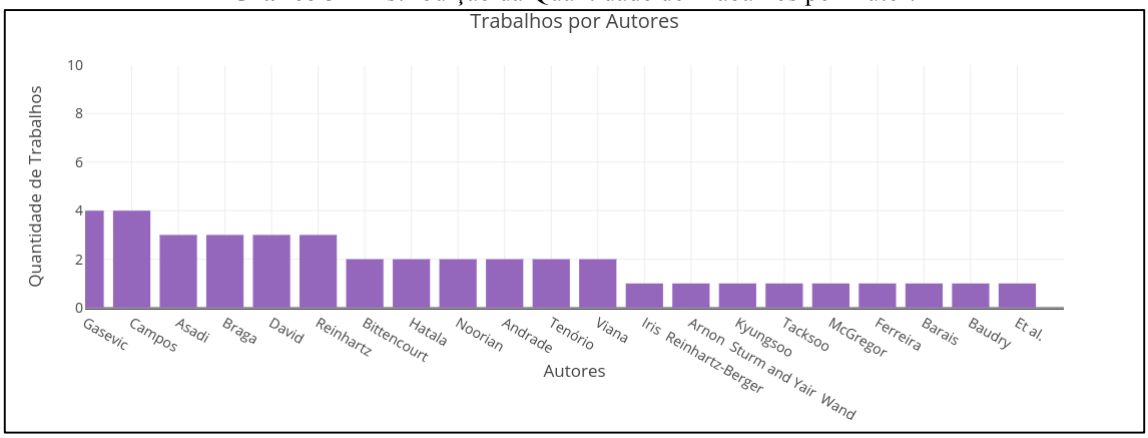
**Gráfico 2 -** Quantidade de Publicações por Conferência.




**Fonte:** Dados da pesquisa, 2018.

Para encontrar a resposta da *RQ3*, buscou-se extrair a quantidade de publicações envolvendo cada um dos autores encontrados nos trabalhos apresentando seu *research name* (Nome em Pesquisas). O total de autores distintos encontrados foi de 162, destes foram selecionados 21 autores conforme apresentados no Gráfico 3. Os autores foram organizados da seguinte maneira (3 autores com 4 trabalhos, 4 autores com 3 trabalhos, 6 autores com 2 trabalhos e 8 autores com 1 trabalho), a seleção se deu em razão de buscar explorar perspectivas distintas entre os autores, bem como oportunizar a variação dos dados encontrados.

**Gráfico 3 -** Distribuição da Quantidade de Trabalhos por Autor.

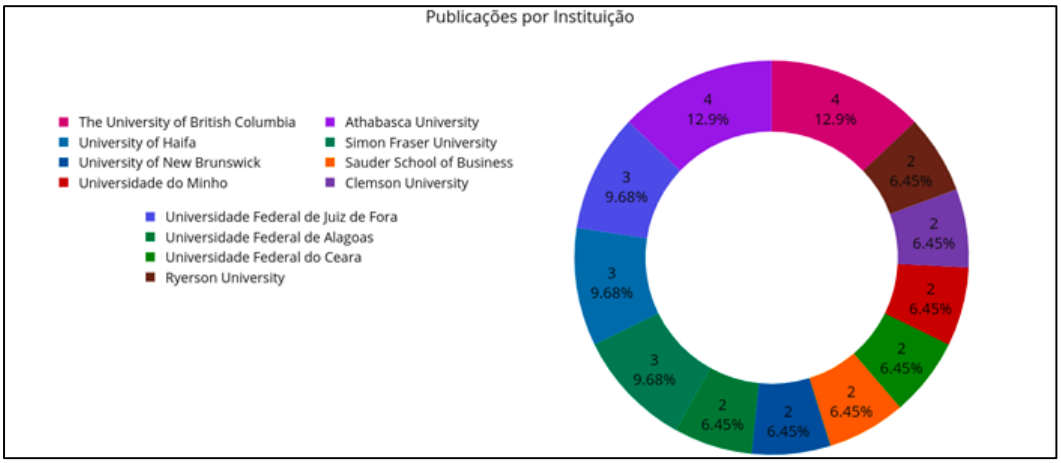



**Fonte:** Dados da pesquisa, 2018.

Os autores encontrados mostraram boa margem de publicação com base nos anos, na análise, foi observado que uma boa parte dos autores realizaram parcerias na escrita de seus trabalhos, Bagheri por exemplo, participou como autor e coautor de 4 publicações: Boskovic, Gasevic, Mohabbati, (...), Rusk e Bagheri (2011), Bagheri, Ensan e Gasevic (2012), Noorian, Bagheri e Du (2014), Noorian, Bagheri e Du (2014). É observado que das 2 publicações Gasevic participou nas duas primeiras publicações. Noorian também foi um outro autor que realizou

parcerias com Bagheri. Com isso, dos 162 autores, foi percebido que ocorreram diversas parcerias, e que em grande parte dos trabalhos houve pelo menos a presença de 2 autores.

**Gráfico 4 -** Quantidade de Publicações por Instituição.




**Fonte:** Dados da Pesquisa.

Diversas instituições foram encontradas nos achados envolvendo a *RQ4* conforme visto no Gráfico 4, dentre essas, foi possível identificar a boa distribuição geográfica entre as instituições, sendo localizadas em países como Estados Unidos, Canadá, Portugal, Brasil, Israel e entre outros. A assertiva apresenta correlação com a alta distribuição entre conferências e autores, associando-se diretamente com os resultados de *RQ2* e *RQ3*

## CONCLUSÕES

O estudo visou realizar um mapeamento sistemático na literatura atual a fim de identificar como se encontra o estado da arte em aspectos de abrangência. Os resultados mostraram que a utilização de ontologias de domínio vem oportunizando uma boa aplicabilidade e adequação ao contexto de LPS. Assim, a perspectiva é que a adoção de ontologias venha a estar cada vez mais presente buscando amenizar problemáticas relativas à falta de definição e manutenção nas especificações do domínio.

## REFERÊNCIAS

BAGHERI, Ebrahim; ENSAN, Faezeh; GASEVIC, Dragan. Decision support for the software product line domain engineering lifecycle. **Automated Software Engineering**, v. 19, n. 3, p. 335-377, 2012.

BOSKOVIC, Marko et al. Developing families of software services: a semantic web approach. **Journal of Research and Practice in Information Technology**, v. 43, n. 3, p. 179, 2011.

CHENG, Haibo et al. Manufacturing Ontology Development based on Industry 4.0 Demonstration Production Line. In: **Trustworthy Systems and their Applications (TSA), 2016 Third International Conference on**. IEEE, 2016. p. 42-47.

FERREIRA, Nuno; MACHADO, Ricardo J.; GASEVIC, Dragan. An ontology-based approach to model-driven software product lines. In: **Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on**. IEEE, 2009. p. 559-564.

GUARINO, Nicola. **Formal ontology in information systems:** Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy. IOS press, 1998.

KITCHENHAM, Barbara; CHARTES, Stuart. **Guidelines for performing systematic literature reviews in software engineering.** Technical Report EBSE, 2007.

NOORIAN, Mahdi et al. Addressing non-functional properties in feature models: a goal-oriented approach. **International Journal of Software Engineering and Knowledge Engineering**, v. 24, n. 10, p. 1439-1487, 2014.

NOORIAN, Mahdi; BAGHERI, Ebrahim; DU, Weichang. Capturing non-functional properties through model interlinking. In: **Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on**. IEEE, 2014. p. 1-6.

PETERSEN, K. et al. Systematic Mapping Studies in Software Engineering. In: **EASE**. 2008. p. 68-77.

STUDER, Rudi; BENJAMINS, V. Richard; FENSEL, Dieter. Knowledge engineering: principles and methods. **Data & knowledge engineering**, v. 25, n. 1-2, p. 161-197, 1998.

# UTILIZAÇÃO DO *DOMAIN DRIVEN DESIGN* NA DEFINIÇÃO DE MICROSSERVIÇOS

*[23]Katyeudo Karlos de Sousa Oliveira; [24]Adriano Lima Cândido;*
*[25]Carlos Williamy Lourenço Andrade; [26]Leonardo Bezerra Franco de Sá; [27]Túlio Vidal Rolim*

## INTRODUÇÃO

Atualmente, as arquiteturas de microsserviço evoluíram para um método bastante difundido em relação a criação de aplicativos multiplataforma. Um exemplo com grande notoriedade é a Netflix, que fornece seus serviços para várias plataformas, envolvendo dispositivos móveis, televisores *smart* e consoles de jogos. Um microsserviço é autônomo e proporciona um conjunto limitado de funções, ou seja, de negócios (NEWMAN, 2015).

Ao se tratar de microsserviços, pode-se considerar que é uma expressão parcialmente nova em meio aos padrões de arquitetura de software, a mesma é vista como uma forma de envolver o desenvolvimento de software como um conjunto de pequenos serviços independentes (THÖNES, 2015).

A tendência atual de criar aplicativos da *Web* usando arquiteturas de microsserviços baseia-se no conceito de *Domain-Driven Design* (DDD). Com isso, entre os desenvolvedores de softwares, o DDD é uma abordagem amplamente aceita e utilizada para a criação de aplicativos que utilizam microsserviços. Essa abordagem de design está presente em diversos softwares atualmente e, como o nome insinua, refere-se a desenvolver aplicações direcionadas ao domínio de negócios. (NEWMAN, 2015).

Em arquiteturas orientadas a serviços, projetar serviços e selecionar limites é um problema complexo (NEWMAN, 2015). Na utilização das abordagens tradicionais para projeção de serviços, segundo Erl (2008), é sugerida uma separação técnica e funcional dos serviços. Porém, de acordo com Evans (2004), o DDD fornece as principais concepções necessárias para dividir os microsserviços. Essas duas características levam ao aperfeiçoamento da qualidade da arquitetura de software (LANDRE, 2006). Nas arquiteturas de microsserviço, tais contextos limitados são utilizados para organizar e identificar microsserviços. O uso do DDD é um fator fundamental para o sucesso na criação de softwares que têm como base os microsserviços (NEWMAN, 2015).

[23] Graduando em Análise e Desenvolvimento de Sistemas da Faculdade Vale do Salgado – FVS
[24] Mestrando em Ciências da Computação da Universidade Federal do Ceará – UFC e docente do curso de Análise e Desenvolvimento de Sistemas da Faculdade Vale do Salgado – FVS
[25] Graduando em Análise e Desenvolvimento de Sistemas da Faculdade Vale do Salgado – FVS
[26] Graduando em Análise e Desenvolvimento de Sistemas da Faculdade Vale do Salgado – FVS
[27] Mestrando em Ciências da Computação da Universidade Federal do Ceará – UFC

De acordo com Steinegger (2017), a abordagem DDD oferece uma forma de simbolizar o mundo real dentro da arquitetura, por exemplo, utilizando contextos limitados representando unidades organizacionais, ainda, identifica e enfoca o domínio central.

Conforme Evans (2004), o DDD é uma abordagem de design de software que evoluiu gradualmente dentro da comunidade de desenvolvedores que usam a tecnologia de objetos. Tal abordagem, visa simplificar o desenvolvimento de projetos de software que tratam de domínios complexos.

Objetivou-se neste estudo, analisar as principais características, motivações e desafios na utilização de *Domain-Driven Design* (DDD) na definição de microsserviços.

## METODOLOGIA

Trata-se de um estudo exploratório, para tanto, utilizou-se de uma revisão de literatura. De acordo com Gil (2008), pode-se entender que pesquisas exploratórias buscam alcançar uma perspectiva geral, de forma aproximada, de um fato delimitado.

No que tange a pesquisa bibliográfica, a mesma visa detectar princípios norteadores e trabalhos científicos atuais que abordam o tema proposto, para, desta forma, respaldar as considerações desta pesquisa em autores já renomados e que têm domínio sobre o tema tratado (MOTTA-ROTH, 2010).

Nesta pesquisa, foram investigados trabalhos de autores que disseminam as temáticas de Arquitetura de Software, Arquitetura de Microserviços e DDD, procurando detectar as principais convicções, que orientaram o restante da pesquisa.

Para a inclusão dos trabalhos, foram selecionados artigos completos e pesquisas publicadas em língua portuguesa e inglesa na área da Engenharia de Software voltadas a Arquiteturas de Microsserviços. As bases de dados utilizadas para seleção dos trabalhos foram: Revista de Informática Aplicada, RITA- Revista de Informática Teórica e Aplicada, Revista de Computação e Tecnologia, IEEE e SBC *Journal on Interactive Systems*, assim como, livros de autores que disseminam as temáticas citadas. Os descritores para as buscas foram: microsserviços, arquitetura de microsserviços, *microservices and* DDD, *architecture microservices or* DDD, *Domain-Driven Design* (DDD) e *architecture microservices or* DDD. Quanto aos critérios de exclusão foram: trabalhos que não tenham sido publicados nos últimos 15 anos, que não sejam escritos no idioma inglês ou português, além de trabalhos não completos. O período especificado se dá pelo fato do trabalho de Evans, a principal obra sobre DDD, ser datada de 2004.

## RESULTADOS E DISCUSSÕES

Foram encontrados 18 trabalhos relacionados a temática da Arquitetura de Microsserviços e DDD, após aplicação dos critérios de inclusão e exclusão, restaram 6 trabalhos, foram analisados os aspectos de projeção de serviços, seleção de limites e o uso de DDD para descobrir serviços. Abaixo, é realizada uma discussão acerca da aplicação, características e benefícios da utilização do DDD para descobrir serviços em construção de softwares com a arquitetura de microsserviços.

Para uma melhor organização e visualização dos resultados do estudo, a seguir é apresentado um quadro com o autor, título e uma síntese sobre o trabalho e em seguida dado início a discussão.

Ao analisar os trabalhos selecionados, foi visto que em suma, a arquitetura de Microserviços é uma abordagem para a construção de uma aplicação única como uma associação de pequenos serviços, cada um executando em seu próprio processo e se comunicando com mecanismos leves. Existe um pequeno número de administração centralizada de tais serviços, que pode ser construído em diversas linguagens de programação e usar diversas tecnologias para persistência dos dados (FOWLER, 2014).

No que está relacionado ao DDD, é visto que tal abordagem disponibiliza um conjunto de princípios e métodos para gerenciar a complexidade para projetar serviços e selecionar limites, identificando domínios centrais e auxiliares e abordando arquitetura, desenvolvimento, operações, equipes, etc. dentro do contexto limitado de cada domínio. A importância da comunicação é acentuada no DDD através de um princípio de linguagem onipresente, ou seja, uma linguagem estruturada ao redor de um modelo de domínio e utilizada por todos os membros da equipe de desenvolvimento e para conectar todas as tarefas da equipe ao software. (BELL et al., 2016).

Ao aplicar o DDD no desenvolvimento de softwares baseados em microsserviços, conforme o nível de experiência da equipe de desenvolvimento, diversos problemas poderão ocorrer, por exemplo, estruturar a linguagem onipresente e escolher em qual momento do projeto utilizar o DDD. Dessa forma, o DDD fornece padrões, princípios, procedimentos e exemplos de como desenvolver um modelo de domínio, fazendo com que a equipe de desenvolvimento possa aplicar o que funciona de maneira mais adequada para cada situação. (STEINEGGER, 2017).

**Quadro 01 –** Casos notificados sobre violência sexual da região do Vale do Salgado

| Trabalho | Síntese |
|---|---|
| (FOWLER, 2014). Microservices a definition of this new | O principal objetivo deste trabalho é explicar as principais ideias e princípios de microsserviços, trazendo uma definição sobre a arquitetura, bem como, características, organização e |

| | |
|---|---|
| architectural term. | gerenciamento dos dados. Por fim, o autor oferece uma reflexão acerca do futuro da arquitetura. |
| (BELL et al, 2016). Microservices Architecture. | Este trabalho apresenta a perspectiva da equipe de trabalho do Open Group SOA, desenvolvido através de pesquisas aplicadas de pontos de vista atuais de profissionais que utilizam a arquitetura de microsserviços. Ele fornece uma definição clara e específica da arquitetura, aborda os principais princípios e principais características. |
| (STEINEGGER, 2017). Overview of a Domain-Driven Design Approach to Build Microservice-Based Applications | Neste trabalho, é fornecido uma breve visão de um processo de desenvolvimento de software baseado em DDD para criar microsserviços levando em consideração os requisitos do aplicativo que se deseja criar. Além disso, é classificado o DDD e depois da descrição do processo, ele é aplicado a um estudo de caso para demonstrar suas possíveis aplicações e limitações. |
| (ŠEVČÍK, 2009). Domain-Driven Design. | O objetivo deste trabalho é apresentar os aspectos mais importantes do DDD e demonstrá-lo em um estudo de caso de um sistema simples de CRM. |
| (VERNON, 2013). Implementing domain-driven design. | Este trabalho consiste em uma abordagem top-down sobre DDD de forma a conectar padrões estratégicos às principais ferramentas de programação. É feita a junção de abordagens guiadas à implementação com arquiteturas modernas, destacando a importância e o valor de se concentrar no domínio do negócio, equilibrando considerações técnicas. |
| (EVANS, 2004). Domain-driven design: tackling complexity in the heart of software. | O objetivo deste trabalho é oferecer uma abordagem sistemática para DDD, apresentando um conjunto abrangente de práticas recomendadas de design, técnicas baseadas em experiência e princípios fundamentais que facilitam o desenvolvimento de projetos de software que enfrentam domínios complexos. |

Fonte: Autores

No contexto das fases de desenvolvimento de software, o foco principal do DDD é por parte da análise e do design, mas ao mesmo tempo interfere nas etapas de coleta e implementação dos requisitos. Vincular essas fases é uma necessidade para o DDD conseguir um processo de desenvolvimento efetivo, porque cada um pode revelar detalhes do domínio de uma perspectiva diferente (ŠEVČÍK, 2009).

Evans (2004), diz que o DDD requer uma arquitetura em camadas para separar o domínio de outras preocupações. Ainda, sugere uma arquitetura de quatro camadas, consistindo nas camadas de interface do usuário, aplicativo, domínio e infraestrutura.

Ao usar DDD, os microsserviços são estruturados de acordo com as unidades organizacionais, usando contextos limitados do modelo de domínio. Os objetos de domínio dentro de um contexto limitado especificam a arquitetura principal de um microsserviço (VERNON, 2013).

Dessa forma, na abordagem de Evans ao DDD, o princípio central é alinhar o aplicativo pretendido com o modelo de domínio. O modelo de domínio molda a linguagem onipresente usada entre os membros da equipe e funciona como uma ferramenta usada para atingir essa meta (EVANS, 2004).

## CONCLUSÕES

A partir do estudo realizado foi visto que, o objetivo do DDD é auxiliar as equipes de desenvolvimento a compreender claramente o contexto e a ideia central envolvida nos projetos, possibilitando o uso de tais conhecimentos para produzir um produto final com maior qualidade e satisfação ao cliente. No DDD, a colaboração com os clientes é essencial para explorar e modelar detalhadamente o domínio. Dessa forma, a primeira e recorrente etapa do DDD é a de conhecimento junto aos clientes. Simultaneamente, a equipe de desenvolvimento implementa a atividade de modelagem e produz o modelo de domínio passo a passo.

Ainda, o DDD introduz o termo "linguagem onipresente" como forma de conectar concepções de modelos individuais, possibilitando que o modelo se comunique entre todos os grupos. Uma linguagem onipresente pode ser compreendida como um dicionário compartilhado acomodando todos os termos que possuem relação com o domínio para o qual o software é desenvolvido. O DDD não diz nada acerca da forma em que este dicionário deve ser elaborado, deixa a cargo dos desenvolvedores se o mesmo deve ser escrito formalmente, ou, se apenas um acordo verbal é o suficiente.

Pode ser observado como vantagens no DDD a facilidade de comunicação, melhora a flexibilidade visto que DDD baseado nos conceitos de orientação a objetos e, assim sendo, se torna bastante modular e encapsulado. Porém, pode ser visto que possui algumas desvantagens, por exemplo, a necessidade de profissionais com boa experiência e que saiba precisamente as particularidades do domínio em que o sistema é destinado. Ainda, para alguns projetos, pode ser visto como problema o fator da iteração contínua e da integração constante, principalmente se projetos anteriores estiverem vinculados a modelos de desenvolvimento menos flexíveis.

Portanto, O DDD oferece os principais conceitos e atividades para construir aplicativos baseados na arquitetura de microsserviços, assim, microsserviços e DDD se encaixam bem. Porém, caso seja necessário implementar um *CRUD* ou criar algum sistema para executar uma tarefa simples com poucas regras, não é necessário utilizar DDD. Aplicar o DDD em tais contextos, somente irá causar complexidade desnecessária no projeto. Assim, caso exista uma certa complexidade no domínio, o DDD foi criado para este fim, ele irá auxiliar na compreensão, facilitar a estruturação dos objetos e reduzir a complicação em manuseá-los.

## REFERÊNCIAS


BELL, J. AJONTECH, L. L. C. CURRIER, B. HARRINGTON, E. E. HELSTROM, C. B. MARTINS, M. **Microservices Architecture**. 2016.

ERL, T. **Soa: principles of service design.** Upper Saddle River: Prentice Hall, 2008.

EVANS, E. **Domain-driven design: tackling complexity in the heart of software.** Addison-Wesley Professional, 2004.

FOWLER, M.; LEWIS, J.; **Microservices a definition of this new architectural term**, 2014. Disponível em: <http://martinfowler.com/articles/microservices.html>. Acesso em: 30 abr. 2018.

GIL, A.C. **Métodos e técnicas de pesquisa social.** São Paulo: Atlas, 2008.

LANDRE, E.; WESENBERG, H.; RØNNEBERG, H. Architectural improvement by use of strategic level domain-driven design. In: **Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications**. ACM, 2006. p. 809-814.

MOTTA-ROTH, D.; HENDGES, G. H. **Produção textual na universidade**. São Paulo: Parábola Editorial, v. 15, p. 16, 2010.

NEWMAN, S. **Building microservices: designing fine-grained systems.** " O'Reilly Media, Inc.", 2015.

ŠEVČÍK, D. **Domain-Driven Design**. 2009. Tese de Doutorado. Masarykova univerzita, Fakulta informatiky.

STEINEGGER, R. H. GIESSLER, P. HIPPCHEN, B. ABECK, S. **Overview of a Domain-Driven Design Approach to Build Microservice-Based Applications**, 2017.

THÖNES, J. Microservices. **IEEE Software**, v. 32, n. 1, p. 116-116, 2015.

VERNON, V. **Implementing domain-driven design**. Addison-Wesley, 2013.

# UTILIZAÇÃO DO PROTOCOLO MQTT NO DESENVOLVIMENTO DE UM SIMULADOR IOT

*[28]Katyeudo Karlos de Sousa Oliveira; [29]Adriano Lima Cândido; [30]Carlos Williamy Lourenço Andrade; [31]Leonardo Bezerra Franco de Sá; [32]João Carlos da Cruz de Lima*

## INTRODUÇÃO

A *Internet of Things* (IoT) é um paradigma que está adquirindo espaço continuamente no cenário das *Wireless Sensor Networks* (WSN) - Redes de Sensores sem Fio. O conceito baseia-se em conectar vários objetos (sensores, computadores, *smartphones*, e objetos de utilidade comum no cotidiano) entre si. Tais objetos, produzem uma corrente de dados e a conexão possibilita com que eles propaguem os dados para os demais objetos no ambiente, estabelecendo uma internet de coisas (ATZORI, 2010).

A principal força do conceito da IoT é o grande impacto causado nas várias vertentes da vida cotidiana e no comportamento dos potenciais usuários. Da perspectiva de um usuário simples, os efeitos mais notáveis da inclusão da IoT se tornam visíveis nas áreas de trabalho e doméstica. Neste âmbito, a saúde, a domótica, a vida assistida, a aprendizagem avançada são apenas alguns exemplos de situações de aplicação em que a IoT representa uma função de influência no cotidiano das pessoas. Da mesma maneira, da perspectiva dos usuários de negócios, os impactos mais visíveis serão igualmente evidentes nas áreas de automação e manufatura industrial, logística, gerenciamento de negócios e processos, transporte inteligente de pessoas e mercadorias (ATZORI, 2010).

Segundo Freund et al. (2016), este é o princípio de um levante tecnológico que ligará objetos que cercam o cotidiano das pessoas à rede mundial de computadores e causará o entendimento entre eles de maneira independente, sem a interferência humana e possibilitando o advento de novos negócios e ações baseadas em IoT.

Para tanto, é necessária a aplicação de diferentes protocolos de comunicação para estabelecer um padrão de comunicação, haja vista que existem vários tipos de usuários e de aplicações. Torna-se essencial um protocolo comum que, independentemente da tecnologia de rede aplicada, possibilite uma comunicação transparente e segura (STANFORD-CLARK, 2013).

[28] Graduando em Análise e Desenvolvimento de Sistemas da Faculdade Vale do Salgado – FVS
[29] Mestrando em Ciências da Computação da Universidade Federal do Ceará – UFC e docente do curso de Análise e Desenvolvimento de Sistemas da Faculdade Vale do Salgado – FVS
[30] Graduando em Análise e Desenvolvimento de Sistemas da Faculdade Vale do Salgado – FVS
[31] Graduando em Análise e Desenvolvimento de Sistemas da Faculdade Vale do Salgado – FVS
[32] Mestrando em Ciências da Computação da Universidade Federal do Ceará – UFC e docente do curso de Análise e Desenvolvimento de Sistemas da Faculdade Vale do Salgado – FVS

No presente trabalho, foi utilizado o protocolo aberto MQTT (*Message Queuing Telemetry Transport*) para analisar o desenvolvimento de um simulador IoT. Dessa forma, Jaffey (2014) estabelece que o protocolo MQTT obedece ao modelo cliente/servidor. Os sistemas sensoriais representam os clientes que fazem conexão a um servidor (*Message Broker*) usando o TCP. As mensagens que serão propagadas, são publicadas para um determinado endereço (tópico). Já os clientes, são capazes de se subscrever para muitos tópicos, ficando dessa forma, possibilitados de obter as mensagens que demais clientes venham a publicar no tópico.

Assim sendo, o MQTT é um protocolo de publicação/assinatura aberto e leve, projetado especificamente para aplicações de máquina para máquina (M2M) e dispositivos móveis. É desenvolvido para comunicações em redes onde a largura de banda é baixa e possui alta latência ou onde a conexão de rede pode ser inconstante. Entretanto, o MQTT requer uma rede subjacente, como o TCP/IP, que viabiliza uma capacidade de conexão sem perdas (STANFORD-CLARK, 2013).

Neste protocolo, existe uma negociação em cada nível de serviço, ocorrendo entre o cliente e um servidor (*Message Broker*), e não entre o publicador e o subscritor. O *Message Broker* poderá ser aplicado na criação de um servidor de mensagens instantâneas altamente eficiente (TANG et al., 2013).

No MQTT são previstos três níveis de qualidade de serviço: o lançamento da mensagem "*most once*" é completamente dependente da rede TCP/IP; Mensagens perdidas ou duplicadas ocorrerão "*least once*" garante que a mensagem chegou, porém, a repetição da mensagem poderá ocorrer; "*only once*" garante que as mensagens cheguem somente uma única vez (SANTOS et al., 2016).

Ainda, o protocolo pode diminuir o custo da eletricidade e reduzir o consumo de tráfego de dados. O servidor publica mensagens ativamente e o cliente recebe a mensagem em tempo real, assim o impulso da mensagem pode ser otimizado e inovado (TANG et al., 2013). Com isso, a utilização dos conceitos do MQTT, poderia ser uma forma de melhorar a comunicação para dispositivos de baixo índice de processamento?

Objetivou-se neste estudo, analisar a aplicação do protocolo MQTT no desenvolvimento de um simulador de IoT.

**METODOLOGIA**

Trata-se de um estudo de natureza básica e exploratória, com abordagem qualitativa, para tanto, utilizou-se de uma pesquisa bibliográfica, assim como, foi desenvolvido um sistema utilizando o protocolo MQTT.

A pesquisa bibliográfica visa detectar princípios norteadores e trabalhos científicos atuais que abordam o tema proposto, para, desta forma, respaldar as considerações desta pesquisa em autores já renomados e que têm domínio sobre o tema tratado (MOTTA-ROTH, 2010).

A pesquisa aplicada se faz com o objetivo de gerar novos conhecimentos relacionando a atividade prática dirigidos à solução de problema específicos (GERHARDT e SILVEIRA, 2009). A abordagem qualitativa tem a fundamentação teórica que contribui para a citação de novos conceitos e progressão do conhecimento, pois visa atender e interpretar comportamentos, atitudes e motivação que influenciam ou determinam a escolha de algo e novas hipóteses (MINAYO, 2014). Quanto ao objetivo, à pesquisa é exploratória, onde oportuniza maior parentesco com o assunto, que pode envolver o estudo bibliográfico e entrevistas com as pessoas que estão diariamente naquele vínculo (GIL, 2008).

A aplicação utilizando o MQTT foi codificada na linguagem de programação Java, utilizando as bibliotecas *paho-mqtt-client*, mantidas pela comunidade Eclipse através do projeto *Paho*. O protocolo é baseado no TCP/IP e ambos, cliente e broker, necessitam da pilha TCP/IP para o seu funcionamento.

Como framework para implementação do MQTT foi utilizado o servidor de código aberto *Mosquitto*. Para a execução dos comandos de ativação e desativação dos aparelhos eletrônicos da aplicação, foi utilizado um dispositivo móvel (*Smartphone*) Motorola, modelo Moto G (2ª geração) e que continha o Sistema Operacional Android na versão 6.0, visto que a ele era atribuída a função similar a um controle remoto. Ainda, foi utilizado um *notebook* Samsung, modelo ATIV *Book* 5, com um processador Intel *Core* i5-5200U e possuía o Sistema Operacional Windows 10.

## RESULTADOS E DISCUSSÕES

Para a fundamentação e embasamento do presente estudo, foi desenvolvida uma aplicação que simula um controlador de dispositivos eletrônicos, o mesmo, adotou o padrão de mensagens do MQTT *publisher/subscriber* (publicador/assinante). Na aplicação não foi implementada qualquer tipo de criptografia (SSL pode ser utilizado independentemente), existindo apenas um método de autenticação com nome de usuário e senha. Havia também os campos de *Broker* (Servidor que armazena todas as informações advindas dos clientes) e *Port*, os mesmos deveriam ser preenchidos para que fosse possível fazer uso do aplicativo. Ainda sobre o campo *Port*, trata-se da porta de conexão com o servidor, por padrão é utilizada a porta 1883, mas pode ser alterada de acordo com a necessidade.

No desenvolvimento do simulador, foi utilizada a biblioteca *Paho Java Client* que forneceu duas APIs: o *MqttAsyncClient* e o *MqttClient*. Para a publicação das mensagens foi utilizado o dispositivo citado anteriormente.

O sistema ainda utilizou uma *thread*, a mesma fazia com que o método *Subscribe* permanecesse, constantemente, ouvindo os tópicos em linhas de execução paralelas. Este método foi utilizado para que fosse permitido que a aplicação executasse múltiplas atividades fazendo com que as demais funções não ficassem indisponíveis, como também, não ficava esperando a finalização de outras aplicações.

**Figura 01 –** Tela do sistema utilizado para publicação de assinatura de tópicos


Fonte: Autores

As mensagens foram enviadas diretamente para o broker e então distribuídas baseadas na escolha dos tópicos do cliente do tipo *Subsbriber*. Cada uma das informações presentes no broker foram designadas por um caminho muito bem determinado, os tópicos. Na aplicação aqui analisada, o tópico se chamava *SmartHome*. Os clientes do tipo *Publisher* eram responsáveis pelo encaminhamento das mensagens para os tópicos, e, também, sua classificação correta das mensagens entre os tópicos. Quando o cliente tipo *Subscriber* lia a mensagem, ele tomava a decisão de acordo com mensagem enviada. Se a informação fosse ligar uma lâmpada, a mesma era ligada.

Na figura 01, é possível verificar a estrutura e as formas de comportamento do sistema desenvolvido. Nesta é possível observar o *Publish* que publicava em um tópico nomeado de *SmartHome* após o cliente *Subscribe* assinava todos os tópicos e a partir de uma publicação ele acionava uma determinada função no dispositivo, como por exemplo: ligar uma lâmpada. Na aplicação analisada, somente era possível a ativação de botões como amostra, visto que a

proposta era a de analisar a aplicação do protocolo MQTT porém, o mesmo poderia ser aplicado em dispositivos reais.

## CONCLUSÕES

Neste trabalho, foi estudado a aplicação do protocolo MQTT no desenvolvimento de um sistema que simulava o controle de aparelhos eletrônicos. Com isso, foi possível a implementação de uma comunicação básica utilizando um dispositivo móvel.

Desta forma, pôde-se concluir que o serviço de envio de informações utilizando o protocolo MQTT é efetivo e ativo. Ainda, é possível identificar que o uso de tal protocolo, em aplicações eletrônicas, pode melhorar a eficiência do acesso à informação e melhorar a comunicação de objetos relativos a IoT.

Também, é possível afirmar que a aplicação desenvolvida, até o presente momento, obteve resultados satisfatórios, uma vez que os objetivos da aplicação foram convincentes no que se diz respeito ao funcionamento do *broker*, funcionamento da comunicação via MQTT e fluxo de informações entre clientes MQTT através de um servidor.

Como trabalhos futuros, fica a proposta de criação de um sistema onde possa ser aplicado o protocolo através do Arduino, para que seja possível enviar mensagens que exerçam as funcionalidades de um controlador de equipamentos real, a fim de realizar uma melhor análise prática sobre o tema.

## REFERÊNCIAS


ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, v. 54, n. 15, p. 2787-2805, 2010.

FREUND, F. F.; STEENBOCK, F. A.; MARANGONI, G. A. C.; VIEIRA, J. D.; DEUS, S. L.; ANGONESE, R. M. Novos negócios baseados em internet das coisas. **Revista da FAE**, v. 1, p. 7-25, 2016.

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de pesquisa**. Plageder, 2009.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 6. ed. Editora Atlas SA, 2008.

JAFFEY, T. **MQTT and CoAP, IoT protocols**. 2014. Disponível em: <http://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php>. Acesso em 30 abr. de 2018.

MINAYO, M. C. S. O desafio do conhecimento: pesquisa qualitativa em saúde. 14. ed. São Paulo. **Hucitec**, 2014. 408p.

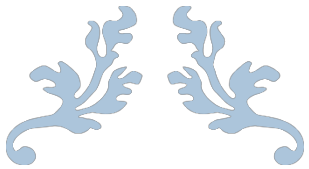
MOTTA-ROTH, D.; HENDGES, G. H. **Produção textual na universidade**. São Paulo: Parábola Editorial, v. 15, p. 16, 2010.

SANTOS, B. P.; SILVA, L. A.; CELES, C. S.; BORGES, J. B.; NETO, B. S. P.; VIEIRA, M. A. M.; LOUREIRO, A. A. Internet das coisas: da teoria a prática. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos**, 2016.

STANFORD-CLARK, A.; TRUONG, H. L. Mqtt for sensor networks (mqtt-sn) protocol specification. **International business machines (IBM) Corporation version**, v. 1, 2013.

TANG, K.; WANG, Y.; LIU, H.; SHENG, Y.; WANG, X.; WEI, Z. Design and implementation of push notification system based on the MQTT protocol. In: **International Conference on Information Science and Computer Applications (ISCA 2013).** 2013. p. 116-119.

MQTT protocol. In: **International Conference on Information Science and Computer Applications (ISCA 2013)**. 2013. p. 116-119.



# VERIFICAÇÃO, VALIDAÇÃO E TESTE DE SOFTWARE

# ANÁLISE DE TEMPO GASTO REALIZANDO TESTES MANUAIS VERSUS OTIMIZADOS COM *SELENIUM IDE*

*Robson Lemos Antonino*[33]*; Andressa Bezerra Ferreira*[34]

## INTRODUÇÃO

Segundo Sommerville (2007, p. 6) "engenharia de *software* é uma abordagem sistemática para a produção de *software*; ela analisa questões práticas de custo, prazo e confiança, assim como as necessidades dos clientes e produtores do *software*.".

O processo de engenharia de *software* compreende diversas etapas: a) Análise e levantamento de requisitos; b) Arquitetura de *software*; c) Implementação; d) Testes; e) Documentação; f) Manutenção

Neste trabalho estarão informações relevantes para a área de engenharia de *software* na parte específica de testes. Segundo Pereira (2012, p. iv) o teste é uma etapa importante dentro do processo de desenvolvimento de *software*, pois "[...] objetiva detectar eventuais falhas antes que o produto chegue ao cliente, sendo um elemento crucial para que o produto atenda suas expectativas"

Antigamente o teste era executado manualmente, o que tornava uma atividade bastante trabalhosa. Testes são exaustivos, porque têm de ser executados repetidamente a cada nova implementação, a cada *Sprint*. Sendo assim, surgiram várias ferramentas para otimizar a elaboração e execução deles.

O presente trabalho aborda o uso da *IDE Selenium* para testes automatizados. Ela funciona junto ao *Firefox* como uma extensão, e não está disponível para outros navegadores. É gravado o passo a passo dos testes executados, para que sejam replicados exatamente como foram gravados quando necessário. Dessa forma, o utilizador precisa realizar os testes manualmente apenas uma vez, ficando a cargo da *IDE Selenium* toda a carga de testes repetitivos (CARVALHO, 2010).

Todos os testes relatados nesta pesquisa foram feitos no site CopaTur (http://copatur-com-br.umbler.net/), que foi desenvolvido durante a disciplina engenharia de software 2, no IFCE Campus Crato. A escolha pelo *Selenium IDE* se deu pela sua facilidade em se trabalhar com testes funcionais. Mesmo sendo um projeto de disciplina, a intenção foi desenvolver um site seguindo os princípios e padrões da engenharia de *software*, contemplando todas as etapas de um projeto de software real.

[33] Discente do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Crato. Email: robsonlemos2011@gmail.com.
[34] Prof.ª Msc. do curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará Campus Crato. Email: andra04@gmail.com.

O *website* CopaTur atua na questão da publicidade e divulgação das cidades sede da copa do mundo na Rússia, que acontecerá em junho/2018. Há 7 categorias com conteúdo exclusivo: História da cidade sede; Restaurantes da cidade sede; Comidas típicas da cidade sede; Artesanato da cidade sede; Estádio da cidade sede; pontos turísticos da cidade sede e Hotéis da cidade sede. O usuário pode se cadastrar no site, realizar *login* e comentar publicações, alterar seu perfil. Os administradores podem criar publicações, tornar um usuário comum em administrador e realizar todas as ações que um usuário comum faz.

O presente trabalho irá fazer uma análise de tempo gasto realizando testes manuais e otimizados. O objetivo é buscar diminuir o tempo gasto com testes durante e após o desenvolvimento de um produto de *software.*

## METODOLOGIA

Este trabalho foi conduzido utilizando metodologia descritiva. Segundo Gil (2008, p. 131) trata de "descrever as características de determinadas populações ou fenômenos. Uma de suas peculiaridades está na utilização de técnicas padronizadas de coleta de dados tais como o questionário e a observação sistemática.".

Dois tipos de testes foram realizados no *website* CopaTur: manuais e testes com a ferramenta de automação *Selenium IDE*, que funciona juntamente com o navegador *Mozilla Firefox*. Tudo foi categorizado e organizado de forma a revelar o quão eficaz essa ferramenta digital é ao auxiliar nos testes de software.

Neste artigo se sobressai a pesquisa quantitativa. Segundo Fonseca (2002, p. 20 apud SILVEIRA, 2009, p. 33) "A pesquisa quantitativa se centra na objetividade. Influenciada pelo positivismo, considera que a realidade só pode ser compreendida com base na análise de dados brutos, recolhidos com o auxílio de instrumentos padronizados e neutros.".

Vários testes foram feitos e o tempo necessário para cumprir cada um deles cronometrado. A partir desses dados, foi possível analisar e apresentar o quão vantajoso é utilizar a *IDE Selenium* para testes.

## RESULTADOS E DISCUSSÕES

Os testes foram agrupados de acordo com o tipo, e foram cronometrados dessa forma, em conjunto. Alguns testes foram bastante extensos: no gerenciamento de temáticas foi criada uma publicação, alterado algum dado e depois excluída a nova página em cada uma das 11 cidades sede. Nos testes relacionados a comentários, todas as 77 publicações do site tiveram comentários inseridos tanto manualmente quanto de forma otimizada com o *Selenium IDE*.

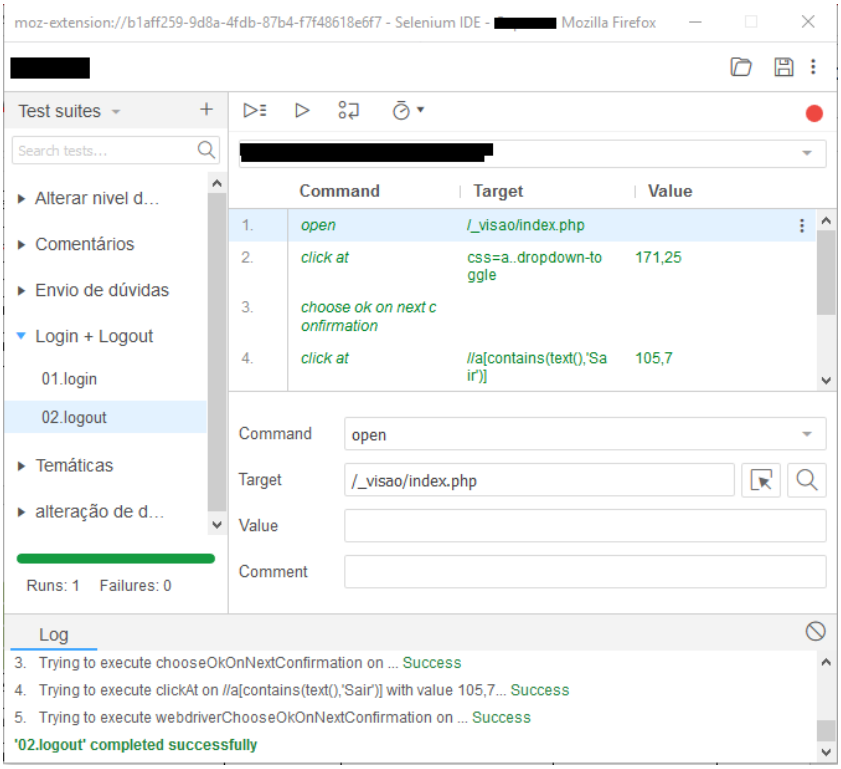



**Figura 01** – Projeto de testes do *website* CopaTur no *Selenium IDE*

FONTE: O autor (2018)

No campo "teste manual" da tabela 01 (abaixo) são realizados testes normais, com o usuário inserindo dados manualmente no website; no campo "Programação do teste no *Selenium IDE*" está o tempo gasto inserindo dados manualmente e gravando todos eles com o *Selenium IDE*, além de solução de *bugs* simples; o "Teste automatizado" é referente a testes executados especificamente com o auxílio do *Selenium IDE*, no qual todos os testes são automatizados. A partir deste ponto o testador não precisa mais inserir os dados manualmente, pois a ferramenta auxilia nessa tarefa.

**Tabela 01** – Testes manuais e automatizados por categoria

| **Tipo do teste** | **Teste manual** | **Programação do teste no *Selenium IDE*** | **Teste automatizado** |
|---|---|---|---|
| Cadastro | 00:00:25 | 00:00:52 | 00:00:14 |
| *Login* e *Logout* | 00:00:29 | 00:00:59 | 00:00:17 |
| Envio de *email* sobre dúvidas | 00:00:28 | 00:00:48 | 00:00:12 |
| Alteração de dados do usuário | 00:00:35 | 00:01:25 | 00:00:23 |
| Alteração de nível de usuário | 00:01:14 | 00:02:49 | 00:00:51 |
| Gerenciamento de Temáticas | 00:05:37 | 00:11:30 | 00:04:52 |
| Comentários nas categorias | 00:16:52 | 00:29:06 | 00:11:15 |

| | | | |
|---|---|---|---|
| **Total** | **00:25:40** | **00:47:29** | **00:18:04** |

FONTE: O autor (2018)

Foi constatado que a maioria do tempo gasto é realizando a programação dos testes no *Selenium IDE*. Embora seja uma tarefa simples até para pessoas que não entendem de testes de software, se torna algo trabalhoso de acordo com a quantidade de testes.

A tabela 02 abaixo mostra o acumulado do tempo gasto com testes manuais e testes automatizados com *Selenium IDE*. Os dados foram retirados da tabela 01, no qual há o somatório de tempo gasto realizando todo o conjunto de testes.

**Tabela 02** – Total acumulado dos testes manuais e automatizados

| **Tempo acumulado/Testes** | **Teste manual** | **Teste com o *Selenium* IDE** |
|---|---|---|
| Tempo acumulado 1º teste | 00:25:40 | 01:05:33 |
| Tempo acumulado 2º teste | 00:51:20 | 01:23:37 |
| Tempo acumulado 3º teste | 01:16:00 | 01:41:41 |
| Tempo acumulado 4º teste | 01:41:40 | 01:59:45 |
| Tempo acumulado 5º teste | 02:07:20 | 02:17:49 |
| Tempo acumulado 6º teste | 02:33:00 | 02:35:51 |
| **Tempo acumulado 7º teste** | **02:58:40** | **02:53:55** |

FONTE: O autor (2018)

É possível observar que a medida que a quantidade de testes avança, se torna mais vantajoso realizar testes com a ferramenta *Selenium IDE*. Além de não ter mais esforços com os testes, não ter mais que digitar todos os dados novamente no momento de testar completamente o sistema, o usuário terá um padrão a ser seguido.

## CONCLUSÕES

Foi comprovado que o uso do *Selenium IDE* é vantajoso para o auxílio nos testes de grandes projetos web, visto que os testes sempre serão necessários durante todo o período de implementação de código e após a codificação para encontrar *bugs* e problemas do projeto.

Quanto mais testes realizados, mais útil será ter feito a programação deles no *Selenium IDE*, visto que testes manuais sempre vão ser mais trabalhosos e demorados à longo prazo. Caso o projeto a ser testado não precise de testes recorrentes, apenas poucos deles, então não haverá vantagem em utilizar a ferramenta para automação dos testes.

## REFERÊNCIAS

CARVALHO, Márcio Filipe Alves. **Automação de testes de software.** ***Dashboard QMSanalyser.*** Tese de mestrado. Documentos ISEC. Coimbra, 2010. Disponível em: <http://files.isec.pt/DOCUMENTOS/SERVICOS/BIBLIO/teses/Tese_Mest_Marcio-Carvalho.pdf>. Acesso em 03 mai. 2018

GERHARDT, Tatiana Egel; SILVEIRA, Denise Tolfo. **Métodos de pesquisa.** Curso PGDR. Editora UFRGS, 2009. 1ª ed. ISBN 978-85-386-0071-8

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2008.

PEREIRA, Diego Varusa. **Estudo da Ferramenta *Selenium IDE* para Testes Automatizados de Aplicações *Web***. Monografia de especialização. Site da UEM. Maringá – Paraná, 2012. Disponível em: <https://bit.ly/1lY0IHz>. Acesso em: 01 mai. 2018

SOMMERVILLE, Ian. Teste de software. In: SOMMERVILLE, Ian, **Engenharia de *Software***. São Paulo: Pearson Addison Wesley, 2007. 8ª Ed.

# UM ESTUDO COMPARATIVO DE FERRAMENTAS DE ANÁLISE ESTÁTICA

*Matheus de Souza Oliveira[35]; Luan Dharlin Lemos da Silva[36]; Anderson Alexandre Paz Cardoso[37]; Emerson Diego Ribeiro Martins[38].*

## INTRODUÇÃO

Desde os princípios do desenvolvimento de software, grandes esforços são aplicados pelos programadores para obter um código-fonte de alta qualidade. Porém, uma definição de código-fonte de qualidade é muito subjetiva, pois esta depende de diversos fatores relacionados a sua implementação. É muito comum nesta fase do ciclo de vida do software a utilização de más práticas de programação que degradam a qualidade do código, tais como a não conformidade com padrões, a utilização de algoritmos e estilos de programação inadequados que tornam o código-fonte de difícil manutenção.

Dentre as técnicas de Verificação & Validação (V&V) para mitigar tais problemas, destacam-se as inspeções de programa, que são análises estáticas realizadas de forma manual com apoio de *checklists* ou automatizadas com o suporte de ferramentas que buscam através de verificadores de qualidade possíveis problemas de implementação no código-fonte. Segundo Terra e Bigonha (2008), a análise estática de código é um dos instrumentos conhecidos pela Engenharia de Software para a mitigação de erros, seja por sua utilização para a verificação de estilos, para a verificação de erros ou ambos.

Embora qualquer artefato de software possa ser inspecionado, a maior parte das pesquisas sobre inspeção de software aborda como foco a inspeção do código-fonte (WILKERSON et al., 2012). Neste estudo, limitamos as inspeções á análise estática com apoio de ferramentas e nos referimos a elas como inspeções de código.

De acordo com Medeiros (2017), análise estática é uma técnica muito importante para garantir a qualidade dos *softwares.* Ainda para o autor, não existem muitos trabalhos relacionados a comparação entre ferramentas de inspeção automatizada de código.

Inicialmente, o estudo foi conduzido com o intuito de identificar os verificadores de qualidade comumente utilizados. Após esse estudo, três ferramentas de análise estática específicas para código Java foram selecionadas: Findbugs, PMD, e Checkstyle.

O objetivo geral é avaliar a cobertura de três ferramentas de análise estática com o intuito de identificar os verificadores de qualidade utilizados por essas e realizar um

[35] Discente do curso de Engenharia de Software. UFC Campus Russas. matheusoliveira2496@gmail.com.
[36] Discente do curso de Engenharia de Software. UFC Campus Russas. luandharlinls@gmail.com.
[37] Discente do curso de Engenharia de Software. UFC Campus Russas. andersonsoftware@alu.ufc.br.
[38] Discente do curso de Engenharia de Software. UFC Campus Russas. emersondiego.engenharia@alu.ufc.br.

comparativo entre elas. O trabalho concentra-se nas ferramentas Findbugs, Checkstyle e PMD. A escolha do Java para o estudo em questão justifica-se pela ampla utilização no ambiente acadêmico e pela disponibilidade do sistema LSCA(Lar Santa Clara de Assis) para aplicação das ferramentas. Os objetivos específicos deste trabalho estão listados abaixo:

- Aplicar as ferramentas de análise estática;
- Categorizar os tipos de erros identificados;
- Analisar e comparar os resultados das ferramentas;

## METODOLOGIA

A metodologia utilizada foi a Pesquisa Aplicada, cujo objetivo é, a partir de conceitos, contribuir para fins práticos. Tem caráter descritivo e natureza qualitativa e quantitativa.

A pesquisa foi conduzida através de um estudo de caso onde foram aplicadas as ferramentas em um sistema Java para "Gerenciamento de Lar de Idosos" desenvolvido no ambiente acadêmico.

O processo metodológico iniciou-se com as pesquisas bibliográficas e a definição dos objetivos. Com isso, as ferramentas que foram foco do estudo foram selecionadas, como também o sistema a ser verificado. Tomando como base os objetivos e as pesquisas bibliográficas, foram aplicadas as ferramentas e coletados os dados resultantes. Posteriormente, foram realizadas as análises, categorização de erros e comparações.

## RESULTADOS E DISCUSSÕES

A maior contribuição deste trabalho foi apresentar resultados de cada ferramenta, bem como uma discussão sobre a cobertura de cada uma e suas diferenças. Abaixo estão descritas as ferramentas e os seus resultados.

Findbugs (2017) considera a ferramenta como um patrocinador na fortificação da qualidade do software. Atualmente, ela pode analisar programas compilados em qualquer versão da linguagem Java. A ferramenta FindBugs provê suporte a mais de 250 padrões de erros sendo mais de uma centena deles classificados como erros de correção. Além disso, permite que programadores escrevam seus próprios padrões de erro.

O Findbugs categoriza e separa os erros de forma detalhada e minuciosa, utilizando separações por classificação de risco(de preocupação, preocupante, assustador e muito assustador) e categoria de erro. Dentre as categorias de erros estão:

- Más práticas;
- Códigos maliciosos;

- Correção;
- Performance;
- Segurança.

Neste trabalho, a ferramenta foi configurada para identificar todos os erros que fazem parte do escopo de verificação da mesma. Nesta análise foram encontrados 5 erros, dentro de duas das quatro classificações do rank de riscos da ferramenta. Os erros encontrados pela ferramenta estão listados no Quadro 1.

Quadro 1: Erros encontrados pelo Findbugs no projeto.

| Erro | Categoria | Rank | Qtd. |
|---|---|---|---|
| Senha do banco de dados é uma constante. | Security | Scary (7) | 1 |
| Vulnerabilidade de divisão de resposta. | Security | Scary (7) | 3 |
| Armazenamento de objeto não serializáveis no HttpSession. | Bad practice | Troubling (14) | 1 |

Fonte: Elaborado pelo autor.

Segundo Checkstyle (2017), é uma ferramenta que ajuda os desenvolvedores a escrever código Java que adere a um padrão de código. Ela automatiza o processo de verificação de código Java dispensando que pessoas realizem esta chata, mas importante, tarefa. É ideal para projetos que querem forçar os desenvolvedores a seguir um estilo de programação. É, ainda, altamente configurável e pode suportar quase qualquer padrão de código que a permite verificar muitos aspectos de seu código-fonte (CHECKSTYLE, 2017).

A ferramenta Checkstyle pode ajudar apoiar a adoção da convenção dos padrões de codificação, ela pode ser configurada de acordo com os padrões estabelecidos, porém está não realiza correções de código, ela apenas indica onde e quais alterações precisam ser realizadas para que a convenção adotada seja obedecida.

Neste trabalho, a ferramenta Checkstyle foi utilizada com as configurações padrões da mesma, dessa forma vários aspectos que não foram considerados tão críticos ou mesmo importantes também foram analisados. Nesta análise foram encontrados 5309 erros, categorizados em 20 tipos listados no Quadro 2.

Segundo o PMD 5.7 (2017) é um analisador de código-fonte que encontra falhas de programação comuns, como variáveis não utilizadas, blocos de captura vazios, criação de objetos desnecessários e assim por diante.

Quadro 2: Categorias de erros encontrados pelo CheckStyle no projeto

| Categoria | Qtd. |
|---|---|
| ‘+’ deve estar em uma nova linha. | 7 |
| Erro na ordem lexicográfica das importações. | 18 |
| As estruturas sintáticas devem utilizar '{}'s. | 51 |
| Falta de comentário Javadoc. | 112 |
| 'import' deve ser separada da declaração anterior. | 3 |
| A forma de importação '.*' deve ser evitada. | 2 |
| Indentação incorreta. | 2078 |
| Linha contém um caractere de tabulação. | 2472 |
| Linha contém mais do que 100 caracteres. | 184 |
| Métodos de sobrecarga não deve ser dividida. | 5 |
| Padrão de escrita de variável. | 42 |
| Padrão de escrita dos nomes das classes. | 28 |
| Parênteses é precedido por espaço em branco. | 15 |
| Primeira frase do Javadoc está incompleta. | 5 |
| 'METHOD_DEF' deve ser separada da declaração anterior. | 8 |
| Modificador 'static' fora da ordem sugerida pela JLS. | 1 |
| Token não está rodeado por espaços em branco. | 247 |
| Cada declaração de variável deve estar em sua própria declaração. | 11 |
| ',' deve ser separada da declaração anterior. | 20 |

Fonte: Elaborado pelo autor.

O PMD também verifica o código-fonte Java e procura possíveis problemas como:

- Possíveis erros – tentativas vazias / catch / finally / switch statements;
- Código morto – variáveis locais, parâmetros e métodos privados não utilizados;
- Código Suboptimal – desperdício String / StringBuffer uso;

- Expressões excessivamente complicadas – instruções desnecessárias se, para loops que poderiam ser enquanto loops;
- Código duplicado – código copiado / colado.

A ferramenta realiza varreduras automáticas em bases de código, gerando relatórios de possíveis problemas encontrados. Seu uso permite que possamos garantir mais qualidade para os códigos produzidos. Certamente, não é possível assegurar a qualidade de um código apenas pelo uso de uma ferramenta como o PMD, no entanto todos os erros encontrados possibilitaram as melhores possíveis soluções.

O PMD separa os erros em cinco classificações de risco(atenção, importante, urgente, crítico e bloqueador) e separa os erros pelas classes e seus respectivos pacotes, como também conta com um relatório preciso com utilização de métricas de teste de software.

Neste trabalho, a ferramenta foi configurada para que sejam verificados problemas considerados mais relevantes para o projeto, que se concentram nas duas classificações de risco mais graves(crítico e bloqueador). Nesta análise foram encontrados 148 erros, categorizados em 5 tipos, dentro de duas das cinco classificações de riscos da ferramenta. Os erros encontrados pela ferramenta estão listados no Quadro 3.

Não existe nenhuma ferramenta de análise estática que seja completa, porém, podem ser consideradas de grande utilidade já que principalmente detectam defeitos que são relacionados à codificação, o que corresponde a expectativa que um desenvolvedor experiente teria.

Dentre as diversas ferramentas para análise estática de códigos Java disponíveis, neste trabalho foram avaliadas as ferramentas Checkstyle, FindBugs e PMD. Essas ferramentas possuem plugin para a versão mais atual da IDE Eclipse e também são de código aberto. De acordo com o estudo e análises o Quadro 4 mostra o comparativo entre as ferramentas.

Quadro 3: Categorias de erros encontrados pelo PMD no projeto.

| Categoria | Qtd. |
|---|---|
| Uma classe que só tem construtores privados deve ser final. | 3 |
| Evite usar tipos de exceção bruta. | 70 |
| Padrão de escrita de variáveis. | 34 |
| Métodos superáveis chamado durante a construção do objeto. | 20 |
| System.out.println é usado. | 21 |

Fonte: Elaborado pelo autor.

Quadro 4: Comparativo das ferramentas e resultados.

| ANÁLISE DAS FERRAMENTAS E RESULTADOS | | | |
|---|---|---|---|
| Ferramenta | Checkstyle | PMD | Findbugs |
| Objetivo | Verificar regras de codificação | Identificar possíveis problemas | Encontrar erros |
| Qtd. de erros identificados | 5309 | 148 | 5 |
| Classificações de riscos | Não possui | 5 níveis (Atenção, Importante, Urgente, Crítico e Bloqueador) | 4 Níveis (De Preocupação, Preocupante, Assustador, Muito Assustador) |

Fonte: Elaborado pelo autor.

## CONCLUSÕES

Este trabalho mostrou que as equipes de desenvolvimento em Java dispõe de diversas ferramentas livres destinadas à melhoria da qualidade de código em seus projetos de software. As três ferramentas aqui apresentadas mostraram que esses utilitários são capazes de verificar o código-fonte produzido e detectar os possíveis problemas no código,

Os resultados obtidos neste estudo mostraram que as três ferramentas possuem coberturas diferentes, por exemplo, a ferramenta FindBugs identificou 5 erros ao passo que a ferramenta Checkstyle identificou 5309 problemas. Já a ferramenta PMD identificou 148 erros e cobriu 2 critérios que as outras não cobriram. Assim, foi possível observar vários erros comuns de programação que passam despercebidos pelo programador. Além disso, observou-se que as ferramentas podem ser utilizadas em conjunto para uma maior cobertura.

## REFERÊNCIAS

CHECKSTYLE. **Checkstyle 7.6**. Disponível em <http://checkstyle.sourceforge.net> Acesso em: 01 jun. 2017.

FINDBUGS. **Findbugs 3.0.1**. Disponível em <http://findbugs.sourceforge.net> Acesso em: 02 jun. 2017.

MEDEIROS, J. E. R. de. **Estudo Comparativo de Ferramentas de Análise Estática de Código**. Dissertação (Graduação em Engenharia de Software) — Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Natal, nov

2017.

PMD. **PMD 5.7**. Disponivel em <http://pmd.sourceforge.net/pmd-4.3.0/index.html> Acesso em 04 jun 2017.

TERRA, R.; BIGONHA, R. S. Ferramentas para análise estática de códigos java. In: **III Encontro Brasileiro de Teste de Software.** Recife: EBTS, 2008. p. 1–5.

WILKERSON, J. W.; NUNAMAKER, J. F.; MERCER, R. Comparing the defect reduction benefits of code inspection and test-driven development. **IEEE Transactions on Software Engineering**, v. 38, n. 3, p. 547–560, May 2012.