FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

GABRIEL EMÍDIO MOREIRA

CONTROLE DE FALHAS E RISCOS EM APLICAÇÕES CRÍTICAS COM AUXÍLIO DE WRAPPER

MARÍLIA
2015

GABRIEL EMÍDIO MOREIRA

CONTROLE DE FALHAS E RISCOS EM APLICAÇÕES CRÍTICAS COM AUXÍLIO DE WRAPPER

Trabalho de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador
Profº Ms. Allan Cesar Moreira de Oliveira

Coorientador
Profº: Dr. Paulo Augusto Nardi

MARÍLIA
2015

Moreira, Gabriel Emídio

Controle de Falhas e Riscos em Aplicações Críticas com Auxílio de Wrapper /Gabriel Emídio Moreira; orientador: Profº. Ms. Allan Cesar Moreira de Oliveira. Marília, SP: [s.n.], 2015.

76 folhas

Monografia (Bacharelado em Sistemas de Informação): Centro Universitário Eurípides de Marília.

1. Sistemas Críticos 2. Wrapper 3. Falhas.

CDD: 005.1

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
MANTIDO PELA FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Gabriel Emídio Moreira

TÍTULO: Controle de Falhas e Riscos em Aplicações Críticas com Auxílio de Wrapper.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Sistemas de Informação.

Nota: 9,0 (Nove)

Orientador:Allan Cesar Moreira de Oliveira Allan

1º.Examinador:Paulo Rogerio de Mello Cardoso Paulo Rogério de Mello Cardoso

2º.Examinador:Ricardo José Sabatine Ricardo Sabatine

Marília, 01 de dezembro de 2015.

AGRADECIMENTOS

Agradeço primeiramente a minha família por sempre proporcionar bases para seguir em frente com meus estudos e por priorizarem a educação antes de tudo.

Ao Profº. Dr. Paulo Augusto Nardi, por suas orientações e mesmo após sua saída da instituição de ensino comprometeu-se a continuar seu trabalho para a construção deste trabalho. Muito obrigado.

Ao Profº. Ms. Allan Cesar Moreira de Oliveira, pelo seu auxílio e comprometimento mesmo após sua entrada repentina em um trabalho já em andamento. Muito obrigado.

Ao Profº. Ms. Ricardo José Sabatine, que mesmo entre suas aulas disponibilizava-se de alguns minutos para saciar dúvidas e breve orientações referente a formulação deste trabalho. Muito obrigado.

Agradeço também a empresa Vertical3W, por ter acredito em meu potencial e fazer parte de minha formação profissional. Obrigado Fernando Mioto, Guilherme Pinheiro, Rafael Moraes e ao Wellington Sono por me aguentarem e serem pacientes. A equipe de desenvolvedores João Paulo de Oliveira, Marina Beretta e Vitor Carnello pela convivência e zueira. E meu agradecimento em especial ao Anderson Meira (vulgo Perna), pois nada disso seria possível sem o seu auxílio, obrigado cara isso ficará guardado eternamente comigo.

Agradeço ao Alexandre Chiararia (Lele), Anderson Meira (Perna), Diogo Henrique, Evandro Garcia, Everton Lima, Felipe Marques, Fernando Castro (Fernandinho), Guilherme Plaza, Leonardo Lima (Léo), Luís Augusto Schiavon (Guto), Marcos Alexandre (Marquinhos), Peterson Ricardo, Rafael Pessotti, Renato Augusto (Renatão), Rodrigo Mateus e ao Thiago Corredo pela convivência desses quatro anos e dos momentos de descontração. Lembre-se sempre desta sábia frase *“Aluno de Sistemas de Informação vai passar VERGONHA!!!” Mr. Buga*.

Agradeço também ao funcionário da biblioteca Izaias pela educação que sempre demonstrou diante de seu atendimento.

“Se você largou tudo pra trás
Se tem o sonho de vencer
Então não é pra se arrepender
Então não é pra se arrepender
A vida é uma teia de escolhas
E escolhas sempre são feitas por você
Então não é pra se arrepender
Então não é pra se arrepender
Mesmo que a vida desenrole
Pra te surpreender
Mas não é pra se arrepender, não é pra se arrepender
A vida é uma teia de escolhas
E escolhas são feitas por você, nunca é pra esquecer
E não é pra se arrepender[...]”

Um Dia Lindo
O Rappa (part. Edi Rock).

MOREIRA, Gabriel Emídio. **Controle de Falhas e Riscos Em Aplicações Críticas com Auxílio de Wrapper**. 2015. 76 f. Trabalho de curso. (Bacharelado em Sistemas de Informação) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2015.

RESUMO

Sistemas críticos são aplicações cujas falhas podem proporcionar sérias consequências, sejam elas ao ambiente de aplicação, aos usuários ou ao próprio sistema. Tais sistemas devem possuir um rígido controle de seus processos para ser precaver contra possíveis imprevistos que possibilitem o surgimento de falhas em seu fluxo operacional. Caso falhas surjam em determinadas aplicações, medidas ágeis devem ser tomadas para minimizar os impactos causados e evitar sua difusão e agravamento diante do sistema. Esse trabalho busca apresentar uma possível ferramenta que auxilie no controle de falhas e riscos em aplicações críticas que venham a falharem em tempo de execução utilizando-se de *Wrappers* para contenção do problema por meio de soluções temporárias até que o sistema seja corrigidos definitivamente pela equipe de desenvolvimento.

Palavras-Chave: Sistemas Críticos, Wrapper, Falhas.

MOREIRA, Gabriel Emídio. **Controle de Falhas e Riscos Em Aplicações Críticas com Auxílio de Wrapper**. 2015. 76 f. Trabalho de curso. (Bacharelado em Sistemas de Informação) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2015.

ABSTRACT

Critical systems are applications whose failure can bring serious consequences for the application environment, users or the system itself. Such systems should have tight control of its processes protecting against possible unforeseen events that enable the emergence of flaws in its operating flow. If faults arise in certain applications, agile steps should be taken to minimize these impacts and prevent its spread and aggravation on the system. This monography aims to present a possible assist that tool in control failures and risks in critical that may fail at runtime using *Wrappers* for problem containment through temporary solutions applications until the system is definitely gone by the development team.

Keywords: Critical Systems, Wrapper, Failures.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

IEEE	*The Institute of Electrical and Eletronics Engineers*
ISTBQ	*International Software Testing Qualifications Board*
MVC	*Model; View; Controller*
RSS	*Really Simple Syndication*
VV&T	*Validação, Verificação e Teste*

SUMÁRIO

INTRODUÇÃO

Com o avanço cientifico e tecnológico das últimas décadas, atividades antes atribuídas a práticas manuais ou mecânicas tornaram-se automatizadas, e assim possibilitaram que seus processos fossem informatizados ao longo do tempo. Sendo assim, sistemas capazes de gerenciar tais operações foram projetados e desenvolvidos para fins de atender tais demandas e suprir as necessidades atribuídas a tais atividades.

Dentre esses sistemas, destaca-se as aplicações voltadas ao controle de operações de alta periculosidade, onde irregularidades em seus processos pode proporcionar situações de risco para a aplicação, aos seus usuários e ao ambiente no qual esteja implantada. A tais tipos de sistemas denomina-se Sistemas Críticos.

Sistemas Críticos são sistemas cuja irregularidades em seus processos podem acarretar em situações catastróficas, onde suas consequência podem levar a perca de vida, financeira ou a desastres ambientais (Capítulo 1). Sendo assim, é de grande importância que tais sistemas apresentem alto grau de confiança e segurança de seus processos, na qual impeça que possíveis falhas venham a surgir e proporcione irregularidades que levem a tais consequências.

Entre as aplicações críticas estão relacionados os serviços de pagamento em comércio eletrônico. Esse modelo de serviço possibilita que usuários realizem operações transacionais financeiras por meio da *web*, o que torna mais ágil os processos de compra e venda através do comércio online.

Um dos modelos de negócio inspirado neste tipo de aplicação crítica são conhecidos como Intermediadores de Pagamentos. Esse modelo de serviço possibilita que seus contratantes ofereçam aos seus clientes variadas formas de pagamentos para serem utilizadas em compras realizadas em suas plataformas de comércio online.

Os serviços de intermediação de pagamentos envolve a comunicação com gateway de bancos e bandeiras de cartões de crédito/débito, além de possuir um infraestrutura e uma equipe capacitada que proporcione maior segurança na realização de transações financeiras por meio da internet. Desta forma, é retirada de seus contratantes a responsabilidade burocrática com tais serviços e com seus devidos clientes em relação a processos financeiros.

Por possuírem acesso as informações financeiras de um vasto volume de usuários, aplicações atreladas a esse modelo de negócio passam a ser classificados como sistemas críticos. Sendo assim, qualquer falha pode causar grandes impactos tanto para o sistema como para as pessoas envolvidas com o negócio (neste caso prejuízos financeiros).

Como falhas são inevitáveis, qualquer tipo de sistema está sujeito a apresentar irregularidades em sua operação. Caso tal fatalidade venha ocorrer à sistemas críticos, procedimentos capazes de interromper e eliminar determinados problemas devem ser realizados, e assim possibilitar a estabilização do sistema em curto prazo de tempo.

Identificar a origem do problema pode não ser uma tarefa de rápida execução, no entanto torna-se necessário que o sistema mantenha a sua operabilidade para que outros serviços não sejam afetados podendo levar a um total colapso da aplicação.

Assim sendo, dois modelos de solução devem ser aplicados: A interrupção imediata da falha crítica; E a identificação e eliminação do defeito. Para que a interrupção da falha crítica não implique na paralisação do serviço, é proposta a criação de um mecanismo que encapsule a funcionalidade que ofereça riscos para a aplicação, sobrescrevendo trechos que possibilitem o mal funcionamento do sistema.

No processo de identificação e eliminação do defeito realiza-se o estudo da arquitetura do sistema em busca da origem do problema apresentado pela aplicação. Desta forma, utiliza-se de processos e ferramentas que auxiliem no encontro do defeito e em sua exclusão, além da validação e verificação da aplicação conforme os seus requisitos.

A fim de não prejudicar a interoperabilidade entre os componentes da aplicação no processo de interrupção da falha crítica, é proposta a utilização de componentes Wrappers (Capítulo 2) pelo mecanismo. A implantação de Wrappers para correção da falha, possibilita que soluções paliativas sejam implantadas no sistema sem que a estrutura do mesmo sofra modificações diretas. Desta forma, o sistema pode operar sem a necessidade de redefinições de interfaces de comunicação entre seus componentes, e assim atendendo as devidas soluções implantadas pelos componentes Wrappers.

Esse trabalho busca solucionar possíveis irregularidades que venham a ocorrer por motivos de falhas em sistemas críticos, cujo seu domínio esteja voltado ao de aplicações de pagamentos em comércio eletrônico. A proposta deste trabalho, é o desenvolvimento de uma arquitetura que proporcione a implantação de soluções por meio de componentes Wrappers em situações que apresente irregularidade em aplicações críticas.

Objetivo Geral

Esse trabalho tem como objetivo desenvolver e implantar um mecanismo ágil de resolução de problemas em aplicações de grau crítico. Como descrito na seção anterior, esse

projeto foca na análise de aplicações voltadas para o mercado eletrônico envolvidas com processos transacionais financeiros via *web*.

Organização do Trabalho

No Capítulo 1 é apresentada uma breve teoria referente a Sistemas Críticos, onde realiza-se a definição de seus conceitos básicos, a sua propriedade de confiança e suas dimensões, uma abordagem sobre a dimensão de segurança e a descrição de relatos sobre sistemas críticos que vieram a apresentar falhas em seus processos.

No Capítulo 2 é feita uma conceitualização referente a Wrapper, na qual é feita uma especificação genérica e alguns pontos específicos da computação em que é aplicado. Também é realizada uma breve descrição referente a sua aplicação e sistemas críticos.

No Capítulo 3 é apresentado o problema em que esse trabalho propôs a resolver, onde é feita uma análise referente ao cenário e as possíveis consequências que o surgimento de falhas podem proporciona para a aplicação e seus envolvidos.

No Capítulo 4 é descrita a solução para a problemática abordada por essa pesquisa, onde são levantadas as ferramentas utilizadas para o desenvolvimento do mecanismo, o seu o embasamento teórico e a sua aplicação em um ambiente simulado.

1. CAPÍTULO 1 - SISTEMAS CRÍTICOS

Neste capítulo, é apresentada revisão bibliográfica referente a sistemas críticos, onde são abordados os conceitos básicos, a propriedade de confiança, relatos de aplicações críticas que apresentaram falhas em sua operação e procedimentos que auxiliam na identificação de estados inseguros no software.

1.1. Conceitos e princípios

De acordo com Lawrence (1995), um sistema crítico é um sistema cuja falha pode levar a consequências irreparáveis. Seus resultados falhos podem afetar no desenvolvimento do sistema, seus usuários, o público em geral ou o meio em que esteja aplicado. As consequências podem implicar na perda de vida, financeira ou danos ambientais que são de extrema importância para sociedade.

Sommerville (2003), define três principais tipos de sistemas críticos mostrados a seguir:

- Sistema crítico de segurança: Um sistema cuja falha pode resultar em acidentes fatais, na perda de vida ou em grande dano ambiental.
- Sistema crítico de missão: Um sistema cuja falha pode resultar no mal funcionamento, onde não consiga atingir seu objetivo pré-determinado (atividades orientadas a metas).
- Sistema crítico de negócio: Um sistema cuja falha pode resultar no fracasso do negócios que utilizam o sistema (sistemas bancários, bolsa de valores, e-commerce e etc).

Com os avanços científicos e tecnológicos, a informatização de operações antes atribuídas a práticas manuais ou mecânicas passou a oferecer maiores perspectivas a respeito da estabilidade das atividades providas de tais processos. A atribuição da gestão por meio de sistemas computacionais proporcionou a redução de irregularidades referente a contextura dos processos, passando a fornecer maior eficiência e precisão na execução de suas atividades.

Segundo Buja et al (2004), a evolução tecnológica leva a sociedade a aceitar a informatização de suas operações cotidianas. Porém, como consequência, as atividades humanas se tornaram dependentes da capacidade de tais sistemas em fornecer os serviços esperados por seus usuários. Assim sendo, tal concepção denominou-se como sendo

dependência.

Em nosso dia-a-dia nos deparamos com a operabilidade de tais sistemas desenvolvidos para saciar práticas comuns aos seres humanos. Por exemplo, o processamento de transações bancárias, a operação de maquinário em organizações(exemplos: indústria metalúrgica, automotiva e etc.), a utilização de equipamentos hospitalares, meios de transporte, dentre outros cenários que atualmente dependem da empregabilidade de sistemas crítico para realização de suas tarefas.

É de extrema importância que tais sistemas sejam confiáveis a ponto de vista de seus usuários, pois serão eles que irão determinar e avaliar o quão seguro é empregar tal sistema em um cenário ou em sua prática cotidiana.

1.2. Confiança em sistemas críticos

A confiança (*Dependability*) aplicada em sistemas define-se como uma propriedade de um determinado sistema computacional, em que passa a ser utilizada como requisito de avaliação para determinar se um sistema corresponde de maneira integra ou não a execução de seus serviços ofertados.

Segundo Sommerville (2003), possuir um sistema crítico com alto grau de confiança é a garantia que o sistema proporciona ao seu meio de aplicação que suas funcionalidades correspondem de maneira eficaz as suas requisições.

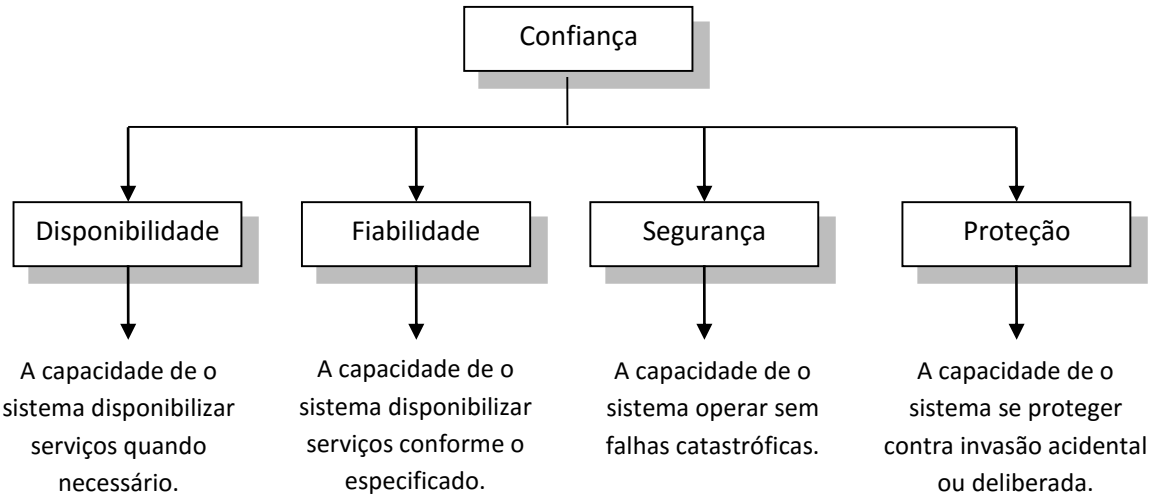
Assim sendo, são estabelecidas quatro dimensões que integram a propriedade de confiança de um sistema computacional, sendo elas: disponibilidade, fiabilidade, segurança e proteção.

Segundo Sommerville (2003), podemos definir essas dimensões da seguinte forma:

1. Disponibilidade (Availability): É a probabilidade de que estará rodando e em condições de disponibilizar serviços úteis a qualquer momento.
2. Fiabilidade (Reliability): É a probabilidade, por determinado período de tempo, de que o sistema disponibilizará serviços de maneira correta, conforme o usuário espera.
3. Segurança (Safety): É um julgamento da probabilidade de o sistema causar danos às pessoas ou a seu ambiente.
4. Proteção (Security): É o julgamento da probabilidade de o sistema poder resistir à invasão acidental ou deliberada.

A Figura 1 ilustra a relação entre a propriedade de confiança com o seus aspectos descritos anteriormente.

Figura 1 - Dimensões da Confiança.



Fonte: SOMMERVILLE, 2003, p. 300.

Entre as quatros dimensões abordadas anteriormente, o objetivo deste projeto busca a exploração do aspecto de segurança relacionado a propriedade de confiança em sistemas críticos. A **segurança** é um importante atributo para sistemas, principalmente os que possuam características de criticidade. Esse aspecto busca avaliar e verificar tais sistema em busca de complicações que possam acarretar em falhas que possibilitem o mal funcionamento do sistema.

De acordo com Hecht (2011), falhas em sistemas digitais críticos define-se como um fator que possibilita o surgimento de irregularidades propicias a causarem danos em que envolvam pessoas, meio ambiente e a prejuízos financeiros.

Kandasamy et al (1999), explica que falhas podem ser classificadas de acordo com a sua duração, ou seja, permanentes ou transitórias. As falhas permanentes possibilita a indução a danos catastróficos, pois são distúrbios que o programa não consegue superar e assim dar continuidade a execução de suas operações. Já as falhas transitórias surgem eventualmente e são toleráveis pelo sistema, não proporcionando risco ao usuários, ao meio aplicado e também ao próprio software.

As falhas propiciadas por sistemas críticos acarretam em dois tipos de custos os diretos e indiretos. Os custos diretos são aqueles que envolvem o sistema e os componentes que auxiliam em sua operação. Por exemplo, a substituição de um determinado hardware falho que impossibilita o funcionamento adequado do sistema. Os custos indiretos afetam a

estratégia do negócio. Por exemplo, uma determinada pesquisa que usufruiria das informações processadas pelo sistema caso ele não tivesse apresentado falha.

"[...] Os altos custos em potencial, relacionados a falhas, significam que a integridade das técnicas e dos processos de desenvolvimento, em geral, é mais importante do que os custos relativos à aplicação dessas técnicas" (SOMMERVILLE, 2003).

> Os Sistemas Críticos são compostos por diversos módulos potencialmente falíveis, nos quais diversos módulos funcionam entre si em numerosas relações de interdependência e que demandam, constantemente, ações preditivas, preventivas e corretivas para manterem-se protegidos de perigos, e operarem com níveis de riscos aceitáveis (OLIVEIRA, 2009).

Sommerville (2003), define três fatores em sistemas críticos que estão sujeitos a proporcionar falhas:

1. Hardware de sistemas, que pode falhar devido a enganos em seu projeto, porque os componentes falham como resultado de erros de fabricação ou por que os componentes de hardware chegaram ao final de sua vida útil;
2. Software de sistemas, que pode falhar devido a enganos em sua especificação, seu projeto ou sua implementação;
3. Usuários do sistema, que podem cometer falhas ao operar o sistema.

Neste projeto será explorado o módulo referente a **software**, onde será analisada sua especificação e implementação em busca de falhas que possam comprometer o funcionamento do programa e que possibilite a indução a graves danos.

> Denomina-se software crítico, aquele utilizado em computadores componentes de Sistemas Críticos. Este tipo de software exige um desenvolvimento cuidadoso para garantir a segurança de software necessária, pois, uma falha na sua execução pode originar um acidente com grandes perdas humanas, econômicas e ambientais (LOVISI FILHO e CUNHA, 2001).

No tópico subjacente serão descritas algumas situações que envolveram aplicações críticas e que apresentaram falhas em seus processos proporcionando perdas significativas.

1.2.1. Relatos de falhas em sistemas críticos

Neste tópico são abordados alguns relatados a respeito de sistemas críticos que apresentaram falhas em sua execução proporcionando perdas relevantes tanto humanas, como ambientais ou financeiras a sociedade.

Missão Progress M-27M

No dia 28 do mês de Abril do ano de 2015, a Roscosmos (Agência Espacial Federal Russa) lançou a nave Progress M-27M com destino a Estação Espacial Internacional (ISS) com objetivo de repor os mantimentos dos astronautas que se encontram em missão no espaço. Porém após chegar ao espaço, o foguete que conduzia a nave ficou inoperante por alguns segundos proporcionando a perca de dados do sistema telemetria (responsável por enviar informações a Terra).

Com o surgimento de tal falha, a nave espacial Progress M-27M parou de corresponder aos comandos enviados pelos seus operadores da Terra e sendo considerada perdida pela Roscosmos. Após tal declaração, foi informada que a nave-cargueira se direcionava a Terra e que estava girando em alta velocidade entorno do seu próprio eixo e que não havia nenhuma maneira de conseguir recuperar a nave pelo fato apresentado.

Com a notícia que a nave estava se aproximando da Terra, as pessoas foram alertadas que os destroços poderiam cair em qualquer parte do planeta, mas que possuirá maior probabilidade de cair nos oceanos que corresponde a 71% da superfície terrestre. Passados alguns dias a nave entrou na atmosfera terrestre, mas antes de atingir a superfície terrestre ela se desintegrou em fragmentos que atingiram o oceano Pacífico.

Neste caso o dano proporcionado por tal falha levou a perca de mantimentos, investimento de US$ 90 milhões e danos ambientais (não graves) ao atingir o oceano Pacífico em sua decida a superfície da Terra.

Outro caso importante em que sua falha proporcionou a divulgação de informações sigilosas de seus usuários está relacionado ao aplicativo mobile do Banco do Brasil, onde se caracteriza como um sistema crítico de negócio.

Fonte:http://g1.globo.com/ciencia-e-saude/noticia/2015/04/nave-russa-descontrolada-comeca-cair-em-direcao-terra-diz-agencia.html <Acessado: 06 de Dezembro de 2015>

Aplicativo do Banco do Brasil expõem contas de clientes.

Em 2013 uma falha no aplicativo mobile de acesso a contas do Banco do Brasil apresentou problemas expondo informações bancárias de clientes do banco. O problema ocorria quando usuários do aplicativos acessavam suas contas, mas eram redirecionados para contas contendo informações de outros clientes do banco.

Essa falha de segurança gerou muitas especulações em redes sócias, pois seus usuários temiam que suas informações também poderiam estar sendo expostas para terceiros que utilizam o mesmo serviço oferecido pelo banco.

O Banco do Brasil informou que o problema estava relacionado aos processos realizados para atualizações do aplicativo, onde causou uma intermitência e inconsistência de dados cadastrais. Sendo assim, o banco relatou que tais informações expostas poderiam ser fictícias e que não houve nenhum risco relacionado ao comprometimento de dados ou transações realizadas nesse período que a falha ocorreu.

Baseando-se nos relatos descritos anteriormente, podemos concluir que sistemas críticos estão cada vez mais presente em nosso cotidiano. Seja para operações habituais ou inovações cientificas, necessitamos que a sua execução corresponda ao seus princípios definidos caso isso não ocorra podemos nos deparar com falhas semelhantes as abordas ou agravantes.

Fonte: http://www1.folha.uol.com.br/tec/2013/12/1383430-falha-em-aplicativos-do-banco-do-brasil-expoe-contas-de-clientes.shtml <Acessado: 06 de Dezembro de 2015>

1.3. Segurança de Software

Segurança de software(*Software Safety*) busca aplicar metodologias que possam assegurar a execução de um programa sem que cause risco à segurança de seus usuários, ao ambiente em que esteja inserido ou ao próprio fluxo de negócio do software.

Segundo Moura (1996), “O fator segurança de software, por sua vez, preocupa-se em garantir que o funcionamento não leve a falhas cuja consequência, para o sistema e o ambiente, seja o ocorrer de sérios danos materiais ou para a vida. O foco, neste caso, é o funcionamento livre de acidentes”.

> Como, à medida em que aumenta a complexidade dos sistemas, cresce também sua susceptibilidade a defeitos de projeto, o aumento do uso de software tende a agravar ainda mais esse problema: a diversidade de estados que um software pode assumir, em função dos dados e caminhos de execução, dificultam o domínio da complexidade. Ficam, assim, comprometidos tanto a compreensão de seu funcionamento, quanto a demonstração de ter-se atingindo os níveis de qualidade previstos (MOURA, 1996).

Para que tais casos não ocorram como descrito anteriormente por Moura (1996), é

necessário o auxílio de métodos que ofereçam formas de identificação de supostas falhas que possam promover sérios risco ao fluxo da aplicação.

Lawrence (1995), define a análise de risco como um método que tem como objetivo: (1) Incentivar mudanças do design do projeto, onde possa reduzir ou eliminar algum tipo de perigo; Ou (2) Efetuar análises e testes em parcelas vulneráveis do sistema proporcionando maior confiança.

A utilização desse método proporciona ao sistema que Estados Inseguros (*hazards*) sejam evitados, e suas operações em geral sejam executadas com poucas possibilidades de surgimento de risco no fluxo da aplicação.

Segundo Lovisi Filho e Cunha (2001), ao se estudar a Segurança de Software é importante que algumas definições sejam esclarecidas:

- Acidentes (*Mishap*) é um evento não planejado que causa perdas humanas, danos ao meio ambiente ou danos ao equipamento;
- Estados Inseguros ou Perigo (*Hazards*) são os estados do sistema que, combinados a certas condições externas, podem levar a acidentes;
- Risco (*Risk*) para esta área do conhecimento é definido em função dos seguintes fatores:
    - Da probabilidade de ocorrência de Estados Inseguros;
    - Da probabilidade dos Estados Inseguros acarretarem acidentes;
    - Pior perda possível associada a esse acidente;

Moura (1996), apresenta duas considerações envolvendo os termos de *defeito*, *falha* e *erro* em um cenário de Segurança de Software:

- Considerações Estáticas referem-se a *defeitos*: o tipo ou número de defeitos que exprimem características estáticas do software, não guardando relação direta com o seu funcionamento.
- Considerações Dinâmicas referem-se a *erros* e *falhas*: os erros só ocorrem quando um defeito é encontrado num dos caminhos de execução do software. A falha acontecerá se, por causa de um ou mais erros encadeados, o software deixar de cumprir um de seus requisitos. Logo, as considerações sobre erro e falha vinculam-se estreitamente às condições de execução do software.

Lovisi Filho e Cunha (2001), também sugerem a aplicação do método da Análise de Segurança de Software por meio da aplicação de uma sistemática que auxilia a identificar

situações que podem levar o sistema a Estados Inseguros.

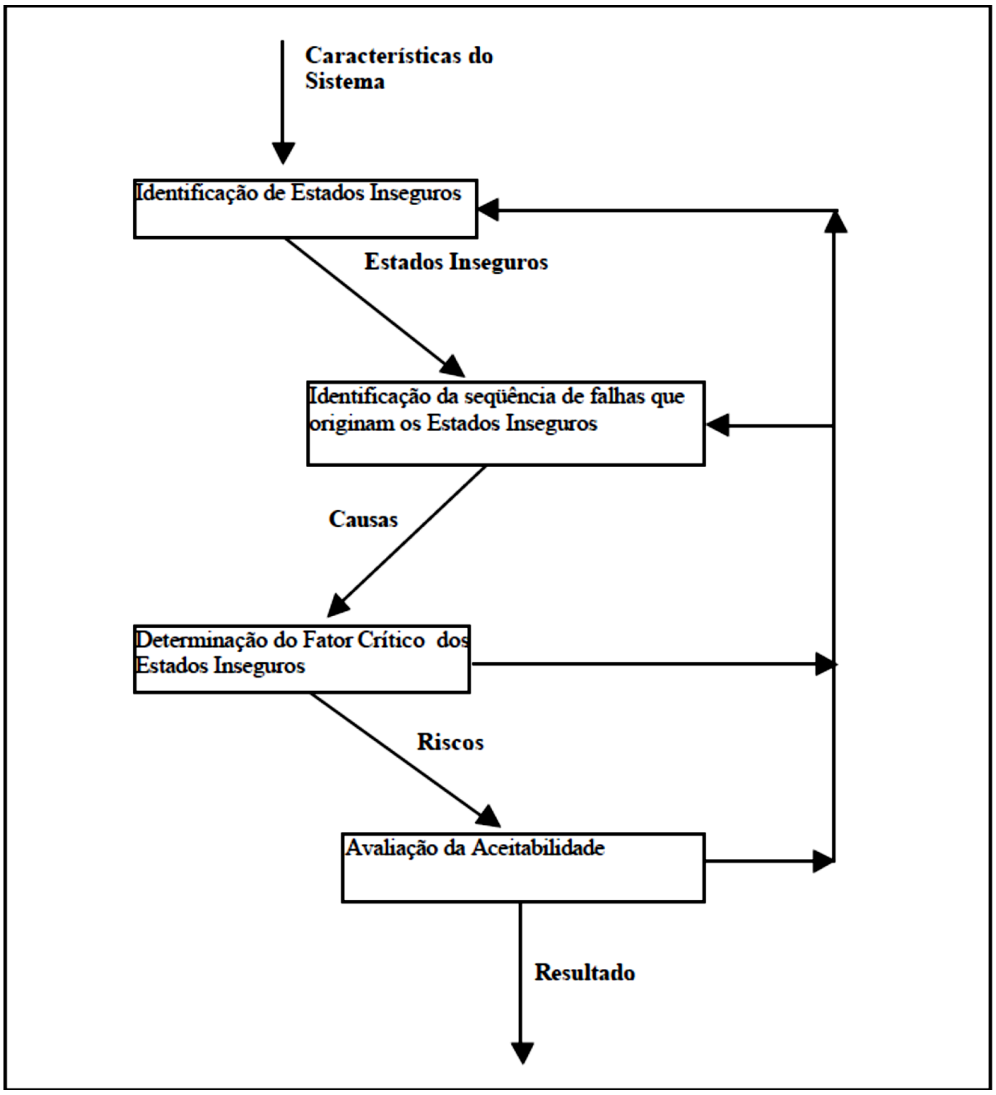
No tópico subjacente, será abordado a Análise de Segurança de Software, onde por meio de uma sistemática irá auxiliam na identificação de Estados Inseguros em sistema crítico.

1.3.1. Análise de segurança de software

De acordo com Lovisi Filho e Cunha (2001), a Análise de Segurança de Software (*Software Safety Analysis*) baseia-se em determinar situações que possam levar um sistema a apresentar Estados Inseguros em seu fluxo de operações. A análise possuem diversas atividades que visam: Identificar os Estados Inseguros e falhas que os originaram; Determinar o Fator Crítico (*Criticality*), dos mesmos; E a aceitação dos níveis de Segurança do sistema.

A Figura 2 ilustra os comportamentos das atividades referente a Análise de Segurança de Software apresentando suas entradas e saídas correspondentes.

Figura 2 - Procedimento para Análise de Segurança de Software.



Fonte: MOURA, 1996. p 41.

> As atividades da Análise de Segurança de Software devem se repetir à medida que a quantidade de informações aumenta; logo, não se pode esperar que as mesmas, obedeçam a uma rígida ordem cronológica, devendo-se revisar estas análises a cada modificação no projeto do sistema (LOVISI FILHO e CUNHA, 2001).

Além de utilizar a Análise de Segurança de Software, existem outras análises que podem ser aplicadas para auxiliarem na descobertas de Estados Inseguros em uma aplicação. Essas análises usufruem de técnicas que possibilitam a identificação de falhas provenientes de Estados Inseguros de uma aplicação.

Lovisi Filho e Cunha (2001) descrevem algumas destas análises, apresentando suas característica e propósitos:

A Análise Preliminar de Insegurança (*Preliminary Hazards Analisysisis*), que visa

determinar os subsistemas críticos e se possível propor alternativas de controle, Baseia-se nos dados existentes e na experiência dos analistas para desenvolver uma análise em alto nível das principais funções e interfaces;

A Análise de Insegurança de Subsistemas (*Subsystem Hazard Analisysis*) objetiva a identificação dos Estados Inseguros e suas falhas no projeto de cada subsistema e de sua interface. Concentra-se basicamente em aspectos tais como: desempenho, degradação no funcionamento e falhas funcionais, procurando determinar os modos de falhas e seus efeitos;

A Análise de Insegurança de Sistema (*System Hazards Analysis*) identifica os Estados Inseguros e suas falhas, associados às interfaces entre os subsistemas, incluindo erros potenciais humanos;

A Análise de Insegurança na Operação e Suporte (*Operation and Support Hazard Analysis*) avalia os Estados Inseguros ocorridos durante as etapas de uso e manutenção do sistema, especialmente àqueles provenientes da interação homem e sistema.

Para utilização de tais análises é essencial que se obtenha um prévio conhecimento sobre a arquitetura do sistema crítico a ser trabalhado, pois cada análise pode apresentar um melhor desempenho dependendo do cenário em que for empregada em uma aplicação crítica.

Moura (1992) descreve de maneira sucinta porém direta atividades que resumem a aplicação da Análise de Segurança de Software:

- Garantir que a segurança seja projetada juntamente com o produto, em um grau consistente com os requisitos de operação:
    - *Foco Principal: Prevenção de Defeitos*
- Identificar, eliminar ou controlar em níveis aceitáveis as vulnerabilidade/riscos associados ao produto e a seus componentes e unidades:
    - *Foco Principal: Remoção de Erros*
- Controlar as vulnerabilidade/riscos que não puderem ser eliminados, de modo a proteger vidas humanos, patrimônio e meio ambiente:
    - *Foco Principal: Convivência com Falhas*

O processo de Análise de Segurança de Software auxilia na prevenção de risco em uma aplicação, identificando supostos agravantes que possibilitem o mal funcionamento de um software crítico. A utilização de seus procedimentos podem auxiliar a estabilizar a segurança do software, onde evitará o surgimento de falhas que possibilitem a surgimento de cenários catastróficos.

1.4. Considerações finais

Este trabalho tem como foco sistema críticos voltados a área de negócios (definido na seção 1.1), na qual será utilizada a dimensão de segurança referente à propriedade de confiança (Seção 1.2, ilustrado na Figura 1). Busca-se com este trabalho o encontro de possíveis fatores que ofereçam riscos e que possam prejudicar a desenvoltura do negócio para o qual sistema foi projetado. Assim sendo, a avaliação do sistema caberá a análise por meio do módulo de software, onde serão analisadas suas especificação e estrutura em busca de falhas que proporcione o surgimento de estados inseguros que possibilitem situações incoerentes para aplicação.

2. CAPÍTULO 2 - WRAPPER E APLICAÇÃO DE WRAPPER EM SISTEMAS CRÍTICOS

Neste capítulo, é apresentada uma revisão bibliográfica referente ao *Wrapper*, em que é descrito seus conceitos básicos e sua aplicação. Também é abordada a interação entre *Wrapper* e sistemas críticos.

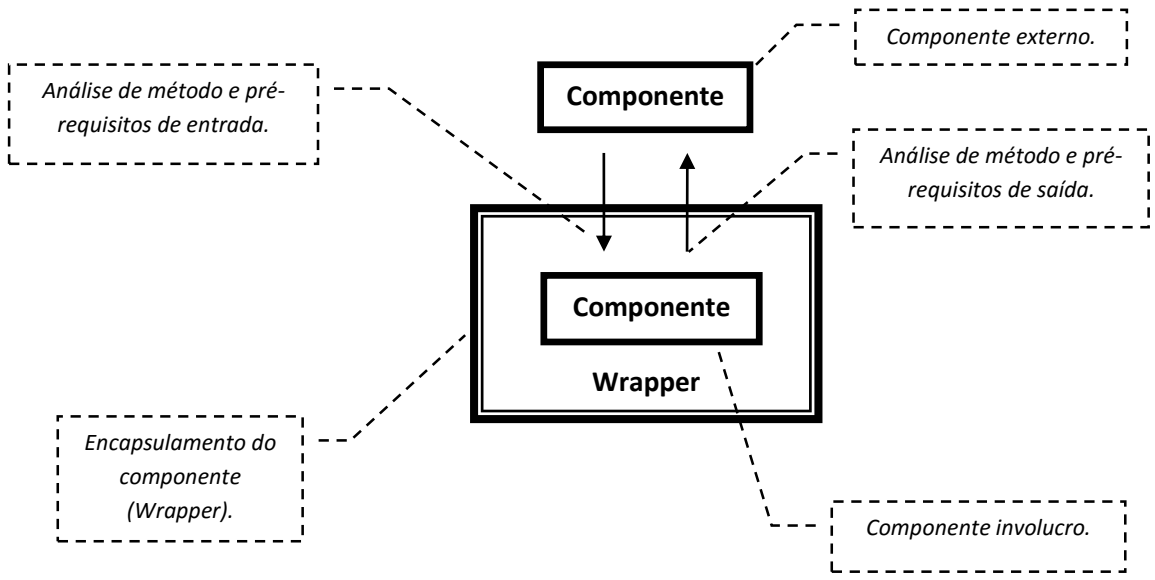
2.1. Wrapper

Em uma definição em contexto não específico, *Wrapper* em termos computacionais pode ser entendido como uma forma de envolver componentes de uma determinada arquitetura com o propósito de restringir o acesso externo a suas especificações.

Segundo Moraes (2003), "Wrappers são componentes especiais que são inseridos entre um componente e seu ambiente para lidar com fluxos de controle e dados que entram e/ou saem do componente protegido".

A Figura 3 ilustra a aplicação de um *Wrapper* em um determinado componente provido por um ambiente:

Figura 3 - Representação de Wrapper.



Fonte: Autoria Própria (ilustração baseada no exemplo de Edwards (2001)).

Na Figura 3, observa-se um componente envolto por uma estrutura *Wrapper*, onde por meio do processo de encapsulamento passa a compartilhar suas respectivas propriedades com a estrutura do *Wrapper*. Isso possibilita que o *Wrapper* mantenha a mesmas funcionalidades

do componente invólucro e realize possíveis modificações sem que afete a estrutura original do componente. Além disso, o *Wrapper* pode estabelecer uma intervenção do fluxo de dados entre o componente envolto e agentes externos de sua estrutura, tornando possível o controle de determinadas entradas e saídas processadas pelo componente do qual é baseado. Para os componentes externos o *Wrapper* passa ser um representação aperfeiçoada do componente involucro, na qual oferece funcionalidades aprimoradas do próprio para um objetivo ao qual foi designado.

O *Wrapper* na computação é utilizado em variados contexto, onde busca aplicar sua funcionalidade de forma adaptada ao ambiente em que esteja empregado.

Em *Design Patterns*, *Wrapper* é definido como um padrão estrutural, onde sua funcionalidade é definir uma entidade invólucro que encapsula e oculta complexidades subjacentes de outra entidade. Também é associado aos padrões de projeto *Adapter* e *Decoretor* que possuem semelhança em seus aspectos comparados aos definido para o *Wrapper*.

Como definido por Gamma et al. (2000), “Os padrões estruturais preocupam com a forma como classes e objetos são compostos para formar estruturas maiores. Os padrões estruturais de classe utilizam a herança para compor interfaces ou implementações”.

Em Orientação a Objeto, *Wrapper* pode ser correspondido por *Wrapper Class* ou Classe Empacotadora e seu objetivo é encapsular uma determinada entidade gerando uma nova classe que herda as informações subjacentes proporcionadas pela classe involucra. Assim, possibilita ao desenvolvedor realizar alterações em métodos pré-estabelecidos na classe pai sem alterar o modelo original.

Em Teste de Software, o *Wrapper* pode ser utilizado para verificação de rotinas pré-estabelecidas em um sistema, na qual possibilita a criação de um modelo do componente a ser testado. Assim, chamadas ao componente invólucro passam a ser correspondidas pelo *Wrapper*, proporcionando a aplicabilidade do teste sem que haja interferência no fluxo da atual da aplicação.

De acordo com Edwards (2001), um *Wrapper* é um verificador que rodeia um componente em teste sem que haja modificação em sua estrutura de código.

Segundo a explicação feita por Edwards et al (2004), um contrato *Wrapper* oferece a mesma interface da entidade contratante assumindo o papel de invólucro e disponibilizando seus componentes subjacentes. Desta forma, tais componentes quando invocados passam a ser interpretados pela classe empacotadora.

A autora Moraes (2003) justifica a aplicação de *Wrapper* da seguinte forma, “Os

wrappers podem ser utilizados quando é impossível ou muito caro mudar os componentes que estão sendo reutilizados como parte de um sistema, ou quando é mais fácil se adicionar novas características incorporando-as aos wrappers".

2.2. Wrapper aplicado em sistemas críticos

Em sistemas críticos a aplicação de *Wrapper* é utilizada de acordo com a necessidade que o sistema possui e seu modelo arquitetural. É possível deparar-se com aplicação de *Wrapper* em sistemas críticos que possuam deficiências arquiteturais na integração entre componentes da aplicação, desta forma o *Wrapper* entra para projetar interfaces que possibilitem que tal necessidade seja suprida pelo sistema.

Por se tratar de sistemas críticos é essencial ter cuidado ao manuseá-lo, pois dependendo do modelo de sistema crítico alguns componentes que o engloba podem ser escassos, de alto custo ou levar perigo aos envolvidos em seu manuseio. Neste caso o *Wrapper* entra para gerar um molde proporcional aos componentes do sistema, na qual possibilita que tais possam ser preservados ao serem submetidos a testes sem que cause nenhum tipo de problema para o sistema e aos seus usuários.

Um teste frequentemente aplicado em sistemas críticos são os de injeção de falhas, cujo o objetivo, como o próprio diz, é introduzir falhas em busca de brecha no sistema que possam levar a resultados inesperados comparado a que está definido em sua especificação.

"A injeção de falhas pode ser uma abordagem interessante tanto para validar componentes isolados como para validar a integração desses componentes em um sistema" (Moraes, 2003) .

Portanto, torna-se necessária a utilização de moldes providos por *Wrapper*, por não afetarem os reais componentes do sistema e apresentaram resultados semelhantes aos que poderiam ocorrer caso fossem testados no componentes originais da aplicação. Assim, proporcionar a confiança e segurança na execução de processos providos por tais componentes.

"Numa arquitetura onde os wrappers são utilizados, esses wrappers também são componentes importantes para serem utilizados como pontos de controle e observação durante os testes por injeção de falhas" (Moraes, 2003) .

Desta forma, pode-se concluir que utilização de *Wrappers* em sistemas críticos torna-se um fator importante principalmente quando existe a necessidade de submeter o sistema a teste, na qual deve-se preservar seus componentes e a segurança das pessoas envolvidas com a

aplicação.

2.3. Exemplo da aplicação de Wrapper

Na linguagem Java, *Wrappers* são utilizados para o encapsulamento de Tipos Primitivos que compõem a linguagem. Os Tipos Primitivos ou Tipos Básicos são modelos de dados específico que uma variável pode assumir caso sua declaração seja referenciada a valores básicos como *inteiros*, *reais*, *textos* ou *lógicos*.

Como a linguagem Java tem como base o paradigma de Orientação a Objetos os Tipos Primitivos acabam sendo impedidos de realizarem ações que só são possíveis com a utilização de objetos. Isso proporcionou a criação de Wrappers para solucionar tal dependência gerada pela linguagem, onde os Tipos Primitivos Java também passaram a ser representados como classes.

Na Tabela 1 são apresentados os Tipos Primitivos em Java e seus respectivos Wrappers.

Tabela 1 - Classes Wrappers em Java.

Tipos Primitivos	**Wrappers**
boolean	Boolean
byte	Byte
char	Character
int	Integer
float	Float
double	Double
long	Long
short	Short

Fonte: Autoria Própria

Além de possibilitar a interação dos componentes primitivos da linguagem ao paradigma de Orientação a Objeto. A aplicação do Wrapper proporcionou um vasto conjunto de funcionalidades para a melhor manipulação dos respectivos valores primitivos diante do programa.

No Código Fonte 1 é feita uma demonstração entre a utilização de uma Classe

Wrapper (*Integer*) e seu respectivo Tipo Primitivo (*int*) em Java, onde é atribuída a ambos a função de converterem um determinado valor em binário.

Código Fonte 1 - Classe ExempleWrapper.

```java
public class ExempleWrapper
{
  /**
  * Converção para um valor binário
  * utilizando a classe Wrapper Integer.
  */
  private void binaryWithWrapper()
  {
     System.out.println(Integer.toBinaryString(10));
  }

  /**
  * Converção para um valor binário
  * utilizando a primitiva int.
  */
  private void binaryWithPrimitive()
  {
    int numero     = 10;
    String binario = "";

    while (numero>0) {
      binario = ((numero % 2 == 0) ? "0": "1" ) + binario;
      numero /= 2;
    }

    System.out.println(binario);
  }
}
```

Fonte: Autoria Própria.

Com o Wrapper o processo é realizado por meio de uma funcionalidade já pré-definida em sua estrutura, enquanto com a primitiva é necessário descrever toda a funcionalidade de conversão para se obter o respectivo resultado esperado.

O exemplo do Código Fonte 1 é uma das muitas situações que a implantação do Wrapper proporcionou aos desenvolvedores Java. Outros processos comuns ao lidar com tipos primitivos também foram implementados em seus respectivos Wrappers (comparações, conversões e etc).

Pode-se concluir que a aplicação do componente Wrapper(seção 2.1) na linguagem Java possibilitou maior praticidade ao lidar com valores primitivos em seu ambiente adaptando sua estrutura a necessidades apresentada pela ambiente no qual era impedido de interagir.

2.4. Considerações finais

O *Wrapper* é um componente essencial para a implementação do mecanismo de controle de falhas. Utilizando de sua conceptualização, a ferramenta possibilitará que componentes falhos sejam envolvidos por uma estrutura *Wrapper*, na qual passará a ter controle sobre suas propriedades e o fluxo de dados proporcionado pelo componente involucro. Isso possibilitará que modificações corretivas sejam aplicadas em sua estrutura permitindo que o sistema normalize a execução de suas operações novamente.

3. CAPÍTULO 3 - PROBLEMÁTICA

Neste capítulo, é realizada uma abordagem referente ao problema que este trabalho propõe a resolver, onde é apresentado o ambiente e as possíveis consequências atreladas a ele caso não seja solucionado.

3.1. Serviços de intermediação de pagamento

Atualmente, é notória a presença de serviços de intermediação de pagamentos no comércio eletrônico. Seja em e-commerce ou *e-bussines*, sempre nos deparamos com opções de serviço especializado em gerenciar intermediações de compras entre lojista e consumidores na web. Segundo o Informativo da Secretária de Comércio e Serviços (SCS) (2015). No Brasil, o comércio eletrônico é composto por 450 mil *e-commerces* em atividades até o momento(informação obtida até 1º de Março de 2015). Desses empreendimentos, 40% utilizam algum tipo de intermediador para realização de pagamentos em sua plataforma comercial. Tal informação é proveniente da pesquisa realizada pela *Big Data Corporation* ao *PayPal* (atualmente uma das maiores plataformas de intermediação de pagamentos no mundo) em busca de desvendar qual o perfil dos *e-commerces* brasileiros.

A principal característica de um serviço de intermediação é a variedade de opções financeiras oferecidas aos seus usuários, na qual possibilita a realização de transações pela *internet* com segurança e confiança garantindo o resguardo e confidencialidade das informações de seus usuários.

De acordo com estudos realizados por Kim et al (2009), usuários são extremamente sensíveis quando riscos envolvem a segurança, privacidade e integridade de suas informações pessoais em sistemas de pagamentos. Sendo assim, tais serviços passaram a dar prioridade em pesquisas referente a técnicas que possibilitassem maior proteção, privacidade e integridade as informações de seus usuários.

Kim et al (2009) também define três áreas que influenciam para percepção de segurança e confiança dos usuários sendo elas: *condições de segurança; procedimentos transacionais; e as técnicas de proteção*. As condições de segurança referem-se à informação prestada ao consumidor e associadas as operações do serviço e suas soluções de segurança. Os procedimentos transacionais referem-se às etapas projetadas para facilitar as ações dos consumidores e eliminar os seus receios referente a segurança da aplicação. Já as técnicas de proteção estão relacionadas aos mecanismos e procedimentos de segurança realizados para

prevenir e proteger as transações dos seus consumidores.

Segundo Junxuan (2010), há uma série de benefícios ao utilizar-se serviços de intermediação pagamento. Primeiramente, esse modelo de serviço consegue atingir comerciantes em proporções globais, na qual os serviços tradicionais não possuem tal capacidade restringindo-se ao comercio local. Também é possível realizar o processamento de transações de seus usuários em escala global, assim possibilitando que os serviços de intermediação se destaque perante ao seus concorrentes. Outra vantagem ao utilizar-se a intermediação de pagamento é que muitas vezes as taxas são menores do que os serviços tradicionais cobram. Isto deve-se ao fato que tal serviço possui uma relação com variados bancos, assim conseguindo praticar valores mais acessíveis aos seus consumidores.

Outras vantagens também podem ser encontradas na especificações de plataformas como Paypal, PagSeguro, MercadoPago, BCash, Moip entre outras que estão em atividade atualmente no mercado. A seguir são listadas algumas vantagens apresentadas por tais plataformas em seus sites:

- A facilidade em sua implementação em ambientes distintos;
- Remoção de responsabilidades burocráticas de seus usuários (exemplo: lojista não necessita gerar contratos com bandeiras, banco e etc.);
- Taxas acessíveis para lojistas na realização de transações com seus clientes utilizando a plataforma;
- Análise e amparo em situações que apresentem características fraudáveis ao consumidor;
- Comprovação de qualidade por meio de certificados e selos fornecidos por autoridades certificadoras reconhecidas por órgãos públicos e privados.

A integração com serviços externos, o gerenciamentos de processos internos e a divulgação de informação precisas aos seus usuários são fatores essenciais para que esse modelo de negócio possa estar em operação. De acordo com Jing (2009), sistemas transacionais são altamente integrados. Além do fator de segurança a gestão de seus processos é importante para aplicação, pois é nela que estão estabelecidas as propriedades decisivas para o funcionamento do negócio. Desta maneira, qualquer inconformidade apresentada na execução de seus processos pode levar ao surgimento de problemas na aplicação, onde dependendo da situação pode agrava-se prejudicando a operação do sistema e levar a estágios irreversíveis como a perca de informação, fraudes e etc.

Segundo Jing (2009), uma vez que haja erros que comprometam a integridade da

informação em um ambiente de pagamentos online a possibilidade de transmissão de dados incoerentes aumentam e consequentemente acabam afetando o funcionamento do sistema e as informações contidas nele.

Assim, a execução deste serviço pode ser considerada como complexa e de alta periculosidade tanto para os envolvidos quanto aos negócios da empresa. Portanto, considera-se esse modelo de serviço como sendo uma aplicação crítica, pois seus fatores envolvem ações que possibilitam o surgimento de riscos aos seus usuários.

Por se tratar de uma aplicação financeira, os serviços de intermediação de pagamentos detêm informações confidenciais de usuários sobre dados e movimentações financeiras em sua posse. Sendo assim, é indispensável que tal serviço demonstre excelência na execução de seus processos possibilitando um ambiente seguro e confiável aos seus consumidores na utilização dos serviços ofertados.

Realizar uma transação em um serviço de intermediação de pagamentos requer a operação conjunta de processos inter-relacionados que ao serem processados possibilitam que tal ação seja efetivada em um sistema. Portanto, ao considerar que nenhum sistema esteja livre de falhas, como enfatizados nos primeiros capítulos deste trabalho, tais processos também estão sujeitos a falharem em suas execuções podendo proporcionar resultados divergentes ao esperado pelo processo transacional e comprometendo a realização de ações subsequentes na aplicação.

Esse trabalho busca intervir em situações de ausência de controle transacional, na qual possibilite que dados transacionais sejam processados de forma incoerente pelo sistema proporcionando resultados indesejados e que possam levar a aplicação ao colapso.

3.2. Processo transacional

Nesta seção, é especificado o Processo Transacional, onde são apresentados sua definição, as propriedades que a compõem e a importância de utiliza-lo em processos.

De acordo com Garcia-Molina et al (2002), uma transação é um processo pelo qual uma aplicação empacota uma sequência de operações que passam a ser interpretadas pelo sistema como uma ação única. Assim, pode-se garantir a execução sistemática de todos os processos de acordo com regra de negócio definida e a consistência de suas informações, e caso algum processo venha a apresentar inconsistências em sua operação a ação não se torna efetiva.

Um processo transacional deve conter as seguintes propriedades: *atomicidade*;

consistência; *isolamento*; e *durabilidade* (*ACID*):

- *atomicidade:* as operações associadas a transação devem ser totalmente executadas, caso contrário nada deve ser persistido.
- *consistência:* permite que uma transação seja executada desde o início a fim sem que haja interrupções de outras transações, ou seja a execução de uma transação é isolada tornando-a consistente para o banco de dados.
- *isolamento:* garante que a transação possa parecer isolada ao sistema, mesmo possuindo outras transação simultâneas em concorrência. O sistema deverá escalonar esses processos, onde uma transação só poderá ser executada caso outra tenha seu processo encerrado.
- *durabilidade:* garantir que as mudanças sejam realizadas no banco de dados após o encerramento do processamento da transação.

Segundo Deng et al. (2003), o processamento transacional é de grande relevância para a sociedade atual que vivencia a era da informação, pois a precisão de dados tornou-se um fator essencial para que um sistema possa proporcionar informações coerentes de acordo com o esperado pelo seus usuários. Caso esse processamento não seja implementado, a possibilidade de informações inconsistentes surgirem torna-se grande podendo levar a aplicação a apresentar futuros problemas em sua execução.

Normalmente em sistema de gerenciamento de base de dados (também conhecido por SGBD), são oferecidos automatização para o processo transacional conhecido como Controle de Concorrência. Esse controle busca atender todas as propriedades necessárias para que o processo transacional seja efetivado em uma base de dados possibilitando a persistência de dados com maior probabilidade de consistência e sem irregularidades proporcionadas pelos processos da aplicação.

3.3. Consequência

A falta do controle transacional em operações de precisão pode proporcionar sérias consequências afetando usuários e sistema, e assim prejudicando o desenvolvimento do negócio perante ao mercado. Os problemas podem tanto oferecer informações divergentes ao esperado pelo sistema quanto brechas no fluxo transacional da aplicação, possibilitando que possíveis fraudes sejam realizadas por usuários mal intencionados por meio da plataforma e colocando em risco outros usuários e ao próprio negócio.

Em algumas situações, os esquemas de fraudes só são descobertos após uma auditoria

na aplicação ou um robusto monitoramento dos processos do sistema. Assim, os invasores exploram da ausência de tais fatores para realização de ataques em busca de se beneficiarem ou prejudicar as atividades do sistema e seus usuários.

Geralmente ataques fraudulentos estão ligados a fatores relacionados ao benefício próprio do invasor (na seção 3.4 serão apresentados alguns relatos), em que não envolve uma grande quantidade de vítimas (usuários da aplicação) em seus atos usufruindo de tais brechas para a sua prática criminosa. Além disso, o usuário mal intencionado busca demonstrar comportamentos que para o sistema não caracterize como uma situação de fraude, na qual consegue estabelecer maior prazo até que seja descoberto pelo sistema.

Já os problemas que afetam uma quantidade significativa de usuários ou aos serviços da aplicação possuem maiores chances de serem descobertos, pois possivelmente estão relacionadas a falhas que surgiram no decorrer da execução do sistema. Em muitos casos a divulgação da falha acaba levando os próprios usuários da plataforma a cobrarem providências por parte da empresa para que o problema seja solucionado o mais breve possível.

Serviços como os de intermediação de pagamentos devem disponibilizar de mecanismos que estabeleçam a normalidade de suas atividades em um curto prazo de tempo caso problemas surjam. Por se tratar de uma aplicação crítica as consequências geradas podem tornarem-se drásticas ao sistema, portanto evitar que situações como descritas anteriormente se propaguem pela aplicação é essencial para prevenção de riscos ao negócio e aos seus consumidores.

3.4. Relatos de situações fraudulentas

Nesta seção, são abordados alguns casos que proporcionaram fraudes em serviços de intermediação de pagamentos por falta de gestão transacional, onde possibilitou a abertura e exploração de brechas para ataques de usuários mal intencionados.

PagSeguro e o Golpe do Vendedor.

Segundo relatos de vítimas, por volta do ano de 2011 foi descoberta uma brecha no serviço de pagamentos oferecidos pela empresa PagSeguro (atualmente pertencente ao grupo Universo Online (UOL)), na qual beneficiava vendedores mal intencionados.

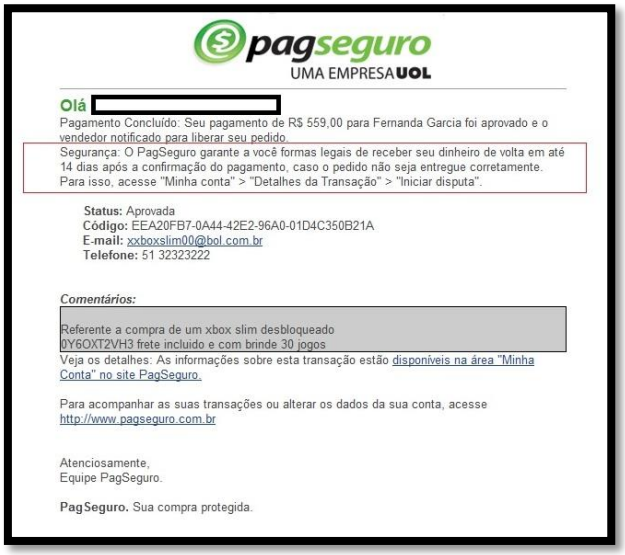
A brecha possibilitava que os lojistas (vendedores) realizassem vendas se passando por Pessoas Físicas, assim conseguiam driblar as políticas da plataforma específicas para

Pessoas Jurídicas e concretizavam a ação como C2C (*Consumer to Consumer*). De acordo com Junior (2012), a expressão C2C no mercado digital é considerada como um ato de venda de produto ou prestação de serviços entre Pessoas Físicas. Desta forma, a transação realizada era efetivada como "não comercial" possibilitando que os procedimentos de compra exigidos para transações entre lojista e consumidores B2C (*Bussines to Consumer*) fossem ignorados pela aplicação.

A fraude era realizada no momento do preparo da forma de pagamento pelo lojista dentro da plataforma do PagSeguro, onde ao selecionar o tipo de pagamento o lojista especificava que a transação não era comercial e enviava a cobrança ao comprador. Assim o comprador efetuava o pagamento e em nenhum momento era alertado sobre o tipo de transação escolhida pelo lojista.

A Figura 4, ilustra uma notificação referente a confirmação do pagamento realizado pelo comprador a um lojista mal intencionado.

Figura 4 - E-mail de Confirmação de Pagamento ao Comprador (PagSeguro).


pagseguro
UMA EMPRESA UOL

Olá [redacted]

Pagamento Concluído: Seu pagamento de R$ 559,00 para Fernanda Garcia foi aprovado e o vendedor notificado para liberar seu pedido.

Segurança: O PagSeguro garante a você formas legais de receber seu dinheiro de volta em até 14 dias após a confirmação do pagamento, caso o pedido não seja entregue corretamente. Para isso, acesse "Minha conta" > "Detalhes da Transação" > "Iniciar disputa".

Status: Aprovada
Código: EEA20FB7-0A44-42E2-96A0-01D4C350B21A
E-mail: xxboxslim00@bol.com.br
Telefone: 51 32323222

Comentários:

Referente a compra de um xbox slim desbloqueado
0Y6OXT2VH3 frete incluido e com brinde 30 jogos

Veja os detalhes: As informações sobre esta transação estão disponíveis na área "Minha Conta" no site PagSeguro.

Para acompanhar as suas transações ou alterar os dados da sua conta, acesse http://www.pagseguro.com.br

Atenciosamente,
Equipe PagSeguro.

PagSeguro. Sua compra protegida.

Fonte: Disponível em:

[http://golpepagseguro.blogspot.com.br/2011/12/entenda-o-golpe-do-pagseguro-o.html]

Como o lojista não enviava a mercadoria, o comprador entrava em disputa para conseguir o reembolso de seu dinheiro. Após o período estipulado pelo serviço para análise da disputa, o comprador era informado que o lojista não possuía crédito em sua conta para que o reembolso pudesse ser realizado deixando os compradores apavorados com a situação.

A seguir é demonstrada de uma forma sistemática referente a aplicação da fraude

realizada pelo lojista aos seus compradores descrita por uma vítima do golpe:

1. O vendedor passa o link do PagSeguro para o comprador.
2. O comprador abre o link e recebe o boleto do PagSeguro.
3. O comprador paga o boleto do PagSeguro.
4. Após compensar o boleto o recurso é passado da conta do comprador para o vendedor.
5. O vendedor transfere o recurso para outra conta do PagSeguro.
6. O comprador não recebe o produto.
7. O comprador abre uma disputa menos de 14 dias.
8. O vendedor não responde a disputa, nem dá qualquer satisfação.
9. O PagSeguro dá a disputa como ganha para o comprador, mas não pode fazer nada porque a conta do vendedor não tem dinheiro.

Na maioria dos relatos encontrados o desfecho da situação não era informado pelas vítimas, porém a grande maioria alegavam que iriam recorrer ao Procon (Programa de Proteção e Defesa do Consumidor) para o resguardo de seus direitos como consumidores.

PayPal Permite Duplicar Valores Em Contas de Seus Usuários.

Em 2010, o serviço de intermediação de pagamentos online Paypal foi alertado sobre uma brecha em seu sistema identificada por um hacker, na qual seu serviços possibilitava que usuários duplicassem os valores monetários contidos em suas contas aplicando um simples processo.

O hacker é Răzvan Manole Cernăianu apelidado de TinKode e bem conhecido por seu feitos como invasões a NASA, ESA e ao The Royal Navy (Marinha Britânica). Segundo Tinkode, o processo de fraude era simples e qualquer usuários comum poderia aplicar ou já estava usufruindo de tal brecha.

De acordo com o relato feito por TinKode, a fraude era realizada utilizando um processo comum em operadoras de cartões de créditos chamado de *chargeback*. A *chargeback* é um procedimento realizado a pedido do comprador quando ocorre uma transação suspeita utilizando dados referente ao seu cartão de crédito, assim pede que a operadora realize o cancelamento da transação.

O processo era realizado utilizando três conta criadas dentro da plataforma, na qual uma era autêntica e as outras duas serviam como contas fantasmas. A fraude era aplicada na transferência de dinheiro entre as contas. A seguir é descrito um exemplo demonstrando como

a fraude era aplica:

Suponha que o usuário xpto possuísse R$ 500,00 em sua conta principal. Com um pretexto para compra de um determinado produto, ele realizava uma transação com uma segunda conta (conta fantasma) cujo o valor era equivalente ao caixa da sua conta principal. Assim, ao receber o dinheiro do produto a conta temporária presenteava com tal quantia uma terceira conta (conta fantasma).

Após um determinado período da realização da compra, o usuário xpto entrava com um pedido de chargeback referente a transação enviando as informações necessária para que seu reembolso fosse realizado.

A plataforma então iniciava um processo em que aguardava provas sobre a transação de ambas as partes envolvidas, porém como o usuário era o único a enviar as informações o serviço dava como causa ganha ao consumidor. Como não era realizada uma análise referente ao caixa da conta fantasma, a plataforma então estornava o valor referente a transação ao usuário e negativava a conta fantasma com o valor ressarcido ao cliente. Sendo assim, o usuário duplicava seu dinheiro, pois o valor retornava a sua conta principal e também estava contido na terceira conta para qual foi enviando o valor da transação.

Tinkode alertou o PayPal sobre a falha que admitiu o problema como sendo uma falha em seus Termos de Serviço, mas não como um vulnerabilidade na aplicação. Porém se analisarmos o caso com mais cautela podemos verificar que a falta de um gerenciamento transacional impulsionou a realização da fraude. A falta de avaliação de pagamento, monitoração de transações e análise de riscos proporcionaram que a fraude fosse realizada pelo o invasor e proporcionando tal consequência ao negócio.

3.5. Considerações finais

Em aplicações críticas como serviços de pagamentos, o surgimento de eventos inesperados podem encadear uma série de problemas que não são nenhum pouco satisfatórias a aplicação e seus envolvidos. Caso seja descoberta divergências que proporcione inconformidades ao sistema, a equipe técnica deve agir de prontidão para que o problema seja solucionado e não se dissemine pela aplicação comprometendo outros usuários e possibilitando maiores prejuízos ao negócio.

Ao analisarmos os relatos da seção anterior, podemos constatar que os problemas estão diretamente ligados a falhas interna da aplicação, onde o sistema proporcionava inconsistências em sua implementação arquitetural favorecendo que supostas fraudes fossem

elaboradas ao utilizar o serviço fornecido pela plataforma.

Este trabalho viabiliza a intervenção corretiva à falhas arquiteturais que venham a surgir em processos críticos relacionados aos serviços de intermediação de pagamentos, no qual busca impedir a proliferação da falha pelo sistema em curto prazo de tempo após a identificação do problema pela equipe de desenvolvimento.

4. CAPÍTULO 4 - SOLUÇÃO

Neste capítulo, são descritos os procedimentos tomados para solucionar a problemática abordada por esse trabalho. Nos tópicos subjacentes são realizadas explicações sobre o mecanismo desenvolvido, as ferramentas utilizadas em sua implementação e a sua aplicação em um ambiente simulado semelhante a um serviço de intermediação de pagamentos.

4.1. Ferramentas de desenvolvimento

Nesta seção são abordadas as ferramentas utilizadas para o desenvolvimento tanto do Mecanismo *Wrapper* quanto protótipo referente a um processo de uma aplicação crítica, na qual o mecanismo é implantado para solucionar o erro.

4.1.1. Linguagem PHP

A sigla PHP é um acrônimo de "PHP: *Hypertext Preprocessor*" (Pré-processador de Hipertexto) que originalmente era chamada de *Personal Home Page*. É uma linguagem de script de código aberto com suporte a sintaxe de Orientação a Objetos e usada geralmente em aplicações *server-side* com objetivo de gerar conteúdo dinâmico para a *Web*.

A linguagem PHP é um software livre de código aberto, ou seja qualquer pessoa pode utiliza-la e modifica-la sem que tenha custo com pagamento de licenças de uso ou royalties. Seu modelo de licenciamento é o GPL (Licença Pública Geral), muito utilizada por softwares livres.

A escolha da linguagem PHP para o desenvolvimento do mecanismo deu-se pela necessidade de simular uma aplicação com fatores críticos presente em um ambiente web, sendo assim, a linguagem foi escolhida por ser bem presente neste tipo de cenário.

4.1.2. Framework CodeIgniter

O CodeIgniter é uma plataforma de desenvolvimento de aplicações em PHP. Seu objetivo é proporcionar o desenvolvimento de projetos muito mais rápido, por meio de um abrangente conjunto de bibliotecas para tarefas comuns em qualquer projeto como desenvolvimento de simples interfaces e estruturação lógica para tais bibliotecas. O

framework permite maior foco no desenvolvimento de seu projeto minimizando a quantidade de código necessário para realização de uma determinada tarefa.

O framework é baseado no paradigma de Orientação a Objeto utilizando o padrão de arquitetura MVC (*Model*, *View* e *Controller*).

Segundo Gamma et al (2000), o padrão MVC é composto por três tipos de objetos. O Modelo (*Model*) é o objeto de aplicação, a Visão (*View*) é a apresentação de tela e o Controlador (*Controller*) é o que define a maneira como a interface do usuários reage às possíveis entradas do mesmo.

A seguir são definidas a utilização das camadas pelo framework de acordo com a sua documentação:

- O Modelo representa as suas estruturas de dados. Tipicamente suas classes de modelo que contém funções que o ajudam a recuperar, inserir e atualizar informações em sua base de dados.
- A Visão é a informação que está sendo apresentada ao usuário. Ela normalmente é uma página web, mas no CodeIgniter, uma visão também pode ser um fragmento de página como um cabeçalho ou rodapé. Ele também pode ser uma página RSS, ou qualquer outro tipo de "página".
- O Controlador serve como um intermediário entre o Modelo, a Visão, e quaisquer outros recursos necessários para processar a solicitação HTTP e gerar uma página web.

O CodeIgniter também utiliza um vasto número de padrões de projetos em sua arquitetura, assim possibilitando que o framework ofereça formas mais adequadas para criação, estruturação e de comportamento aos seus componentes.

A escolha do framework CodeIgniter deu-se pela necessidade de criar um aplicação em curto prazo de tempo e com qualidade. Além disso, o framework possibilita que seus módulos presentes em seu *core* sejam sobrescritos, na qual possibilitou a aplicação do mecanismo com mais praticidade nesta arquitetura.

4.1.3. MySQL

O MySQL é um sistema de gestão de base de dados (SGBD) que utiliza a linguagem SQL (Linguagem de Consulta Estruturada) como base em seus processos. Atualmente, o MySQL é mantido pela Oracle Corporation. É um software livre com base na licença GPL, porém softwares que não possuam licença GPL devem adquirir uma licença do sistema para

utilização.

4.1.4. JSON

O *JavaScript Object Notation* ou abreviadamente JSON, é um formato para armazenamento e transmissão de dados em formato de texto. Mesmo sendo um notação de objeto da linguagem JavaScript, o seu uso é bem difundindo entre inúmeras linguagens pela facilidade em sua manipulação e o baixo custo de processamento de sua estrutura comparada a outros meios de transmissão de dados.

4.2. Mecanismo Wrapper

Nesta seção é contextualizado o Mecanismo *Wrapper* proposto por este trabalho, onde são descritas as motivações para sua criação, a interação com a ferramenta e seu desenvolvimento.

4.2.1. Contextualização

A proposta de criação do mecanismo baseou-se nos estudos realizados referentes aos sistemas críticos e nos possíveis riscos que o surgimento de falhas em seus processos podem causar ao ambiente e aos demais envolvidos com o sistema. Dentre os três tipos de sistemas críticos abordados nesta pesquisa (Capítulo 1), a escolha se deu pelas aplicações voltadas aos negócios, na qual seu fator crítico envolve falhas que possam proporcionar perdas financeiras significativas podendo levar o empreendimento ao insucesso.

A atuação do mecanismo está voltada ao fator de software em uma aplicação crítica, no qual se enquadra como uns dos três fatores falíveis em um sistema crítico como descrito no Capítulo 1 deste trabalho. Assim, o mecanismo busca solucionar supostos problemas cometidos por enganos no desenvolvimento do software que proporcionaram a introdução de processos incorretos na estrutura da aplicação possibilitando o surgimentos de falhas no sistema.

O mecanismo tem como objetivo a exploração do aspecto de segurança em sistemas crítico, em que busca assegurar a execução do sistema sem que apresente riscos as entidades que possuam envolvimento com tal aplicação. O aspecto de segurança é uma das dimensões que complementam a propriedade de confiança (Capítulo 1) em aplicações críticas, na qual

sua função é analisar e verificar supostas complicações no sistema que possibilitem a má operabilidade de seus processos que possam proporcionar riscos ao ambiente e aos seus usuários.

A atividade de *Análise de Insegurança de Sistema* descrita por Lovisi Filho e Cunha (2001) abordada na seção 1.3.1 deste trabalho, é uma das propriedades atribuídas ao mecanismo. Tal análise se encarrega de sondar falhas associadas ao sistema que possivelmente possa ter erro humano envolvido em sua origem.

Em busca de corrigir tais vulnerabilidades no sistema, a ferramenta dispõem de processos que possibilitem a eliminação ou o controle do risco evitando que a falha se propague pelo sistema causando maiores transtornos ao negócio.

Dentro desta contextualização, analisou-se um cenário em que uma aplicação crítica venha a apresentar problemas em sua execução, no qual a equipe de desenvolvimento necessite conter o problema em curto prazo sem que as atividades do sistema sejam interrompidas como em um processo de implantação.

Um processo de implantação também conhecido como *deploy* consiste em um conjunto de atividades cujo objetivo é a realização de instalações e manutenção de uma determinada aplicação. Essas atividades envolvem a interação de práticas e procedimentos para que o processo de implantação possa ser concretizada com êxito.

Segundo Coupaye e Estublier (2000), a implantação de software é um processo complexo que abrange todas atividades a serem realizadas após o final do desenvolvimento para instalação ou manutenção de uma aplicação.

Carzaniga et al (1998), define algumas atividades geralmente executadas em um processo de implantação de um sistema que são definidos a seguir:

A liberação (*Release*) é a atividade que segue o fim do processo de desenvolvimento incluindo operações para o preparo de um versão executável do sistema e informações para execução das atividades posteriores.

A instalação (*Install*) é a atividade de implantação inicial do software em ambiente de produção. Essa atividade engloba ferramentas para configuração e transferência da aplicação do ambiente de desenvolvimento ao de produção.

A ativação (*Activate*) é a atividade de inicialização do programa em seu ambiente de operação. Em sistemas simples normalmente utiliza-se de comandos para sua execução. Já em sistemas mais complexos essa atividade pode depender da adequação de fatores atreladas a aplicação para seu funcionamento.

Em sistemas de grande porte, normalmente essa atividade é estabelecida tanto no

ambiente de produção quanto em um ambiente de teste denominado como homologação.

A desativação (*Deactivate*), como o próprio nome diz, é o processo inverso da ativação, na qual busca interromper qualquer componente que proporcione a execução do software. Essa atividade é geralmente necessária em processos como o de atualização do sistema.

A atualização (*Update*) é um processo semelhante a de instalação, porém consta com a realização de melhorias ou modificações na aplicação. No entanto, é menos complexo, pois em alguns casos os recursos necessários para a aplicação da atividade já foram estabelecidos no processo de instalação.

A adaptação (*Adapt*) como o processo de atualização, envolve procedimentos para modificações do sistema previamente instalado. O que difere a adaptação da atualização é que na atividade de atualização inicia-se por eventos remotos como uma nova versão da aplicação ou correções de bugs. Já a adaptação está relacionada com medidas corretivas que mantenham a regularidade das operações do sistema no ambiente em que esteja aplicado.

A desinstalação (*Deisntall*) é atividade destinada a remoção de um sistema desnecessário a um determinado ambiente. Também envolve alguns processos de reconfiguração do sistema para retirada de dependências ou arquivos sem utilização pela aplicação.

A remoção (*Retire*) é a atividade de retirada de um sistema já obsoleto que esteja em produção. Como a desinstalação, cuidados devem ser tomados para garantir que a retirada não cause problemas ao sistema.

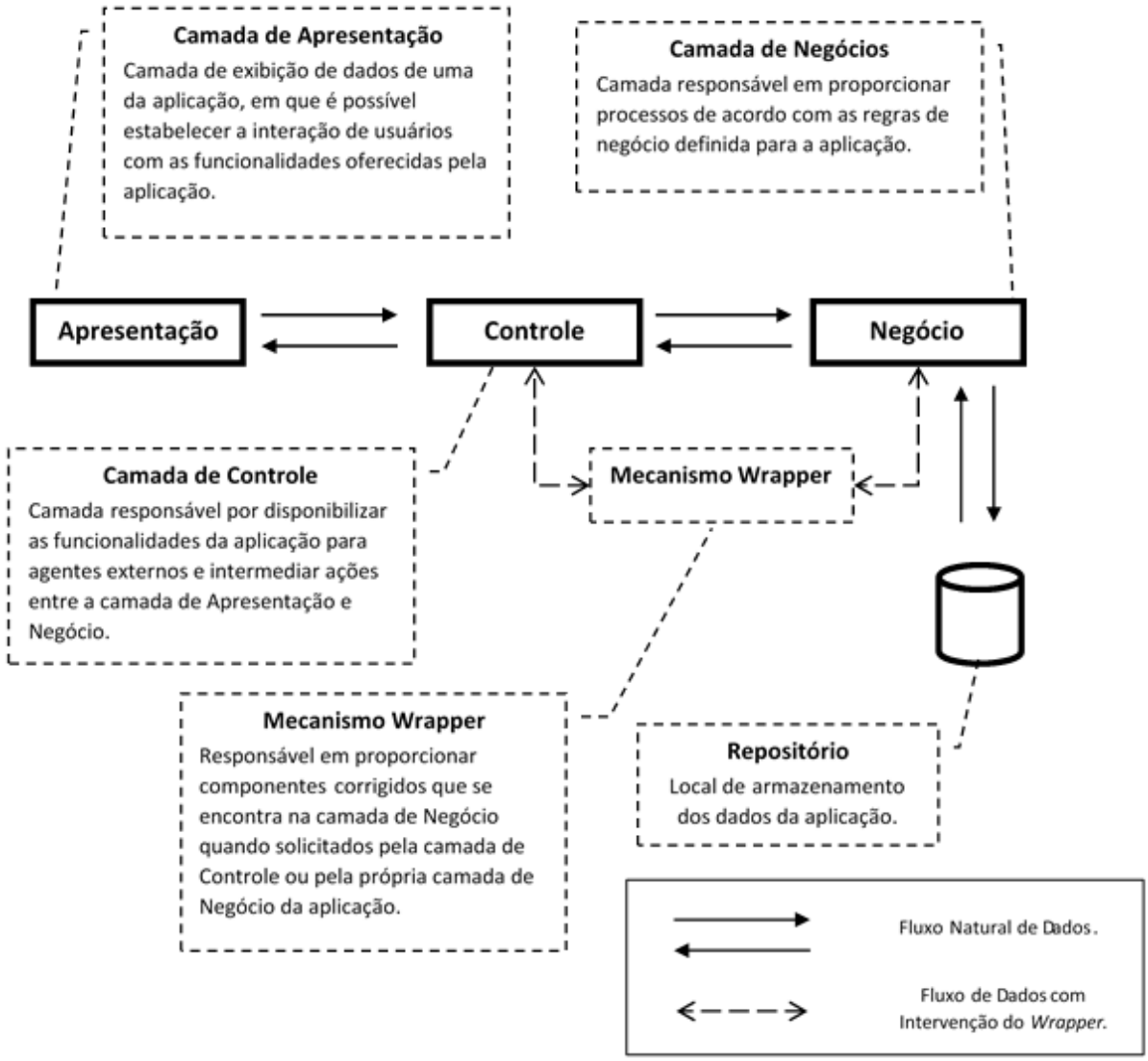
Carzaniga et al (1998), realça que embora podemos generalizar as atividades a serem executadas em um processo de implantação como mostrado anteriormente, não podemos definir com precisão quais práticas e procedimentos serão realizados em cada atividade definida. Isso depende do tipo de sistema, seus fatores e necessidades impostas pela ambiente de aplicação, ou seja pode ser uma atividade de rápida execução ou levar até meses para ser concretizada.

O mecanismo tende a solucionar os problemas encontrados em tempo de execução em uma aplicação de maneira que possa intervir em sua propagação pelo sistema. Sendo assim, o mecanismo pode proporcionar uma correção definitiva ou soluções paliativas até que uma possível correção seja realizada em um procedimento formal como o de implantação (*deploy*) seguindo todos os passos definidos em seu planejamento.

A Figura 5 ilustra a implantação do mecanismo *wrapper* em uma arquitetura MVC, na qual podemos encontrar as camadas de apresentação, controle, negócio e armazenamento de

dados.

Figura 5 - Aplicação do Mecanismo Wrapper em uma Arquitetura MVC.



Fonte: Autoria Própria.

Na Figura 5, pode-se observar a ação do mecanismo aplicado em uma arquitetura MVC, na qual interage com a camada de Controle e Negócio. Funcionalmente, o mecanismo mapeia a camada de Negócio analisando as estruturas ali definidas, assim possibilitando que os componentes pertencente a tal estrutura possuam um *Wrapper*. Desta forma, o mecanismo monitora as requisições feitas a essa camada (podendo partir tanto do Controlador como da própria camada) e caso o componente solicitado possua um *Wrapper* o mecanismo redefine o fluxo natural da aplicação e passa a processar o componente gerado pelo mecanismo.

4.2.2. Estrutura Wrapper

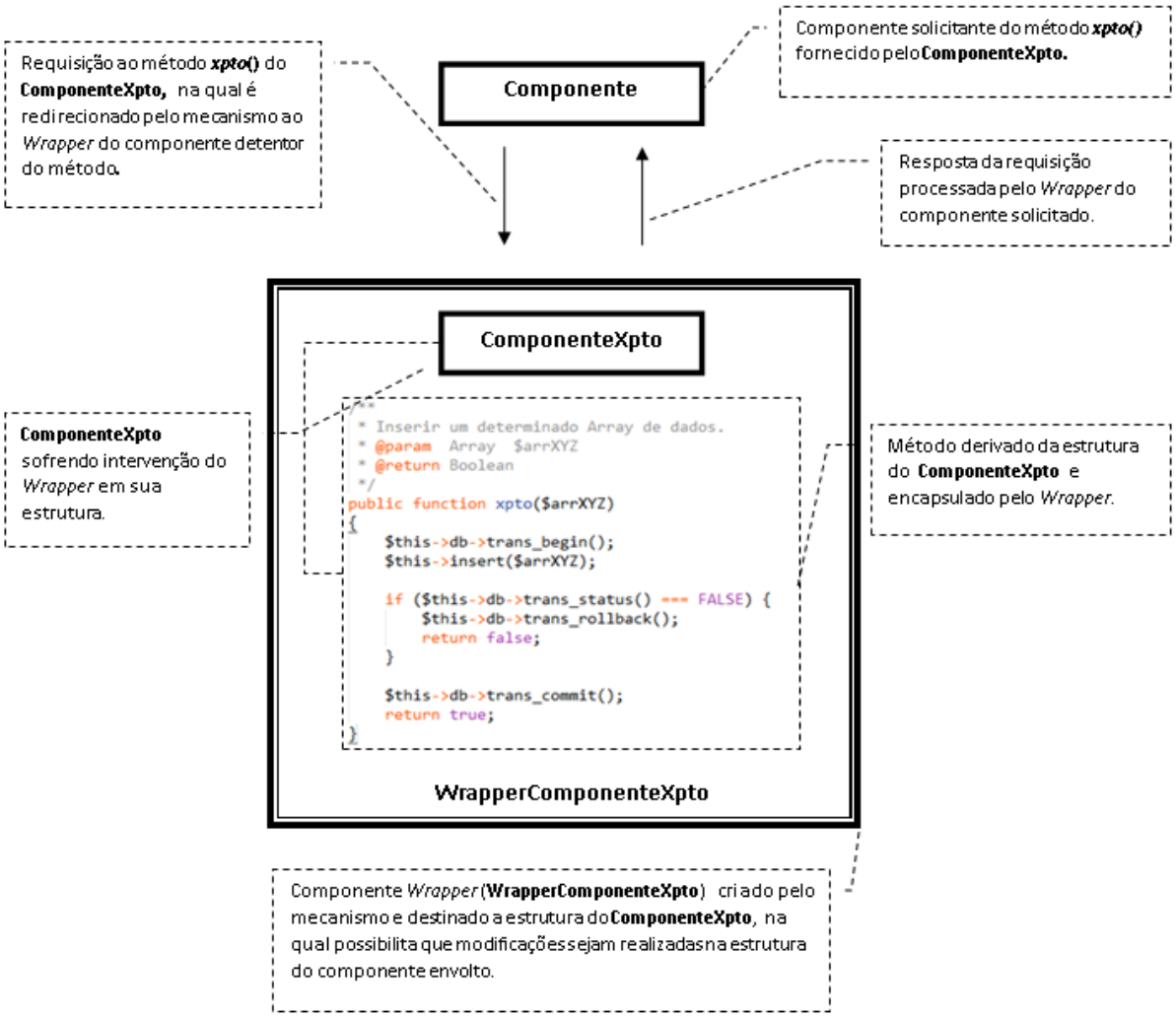
A funcionalidade de um *Wrapper* como descrita e ilustrada no Capítulo 2, tem como objetivo privar o acesso direto a um determinado componente por meio da ação de encapsulamento de sua estrutura. Wrappers possibilitam o controle dos dados processados pelo componente envolto por meio da intervenção realizada em sua estrutura utilizando da sobrescrita de suas funcionalidades.

O mecanismo baseia-se na mesma estrutura descrita anteriormente com foco na contextualização de *Wrapper* quando utilizado em Teste de Software(Capítulo 2), porém ao invés de ser aplicado para atividades de validação, verificação e testes das funcionalidades de um determinado componente ele proporciona uma solução ao sistema contra as divergências providas em seus processos.

A aplicação do mecanismo possibilita que o desenvolvedor encapsule as funcionalidades do elemento falho em outro componente (*Wrapper*), assim tornando possível realizar modificações necessárias para corrigi-lo. Um fator importante é a utilização da própria estrutura do componente defeituoso para o reparo da falha, assim possibilitando que o *Wrapper* gerado usufrua da estrutura original sem alterá-la diretamente evitando que outros problemas venha a ocorrer na aplicação.

A Figura 6 ilustra a atuação do *Wrapper* gerado pelo mecanismo a um determinado componente, no qual restringe o acesso externo a ele tornando-se um intermediador no fluxo de dados entre o componente solicitante e o componente protegido.

Figura 6 - Estrutura do Wrapper Gerado pelo Mecanismo.



Fonte: Autoria Própria.

Na Figura 6, é possível ter noção da utilização do *Wrapper* em uma aplicação cujo o mecanismo esteja implantado, na qual possibilita a manipulação do componente envolto em sua estrutura. Com tal funcionalidade o *Wrapper* consegue controlar o fluxo de dados entre o componente encapsulado e o restante da aplicação sem que interfira em sua estrutura original.

O processo de encapsulamento realizado pelo *Wrapper* deve priorizar a intervenção em processos internos do componente a qual sua estrutura envolve. Interfaces de comunicação entre o componente involucro e agentes externos deve ser mantidas ou possuírem modificações mínimas, algo que não afete na interação com tais componentes possibilitando maiores transtornos para a aplicação. A ideia da aplicação do *Wrapper* é que a comunicação seja transparente aos componentes externos, ou seja não necessariamente eles devem ter conhecimento com quem estão interagindo, mas sim se o resultado está sendo entregue como

o esperado.

Como ilustrado na Figura 6, o método denominado *xpto()* é uma funcionalidade mantida pela estrutura do componente envolto pelo *Wrapper*. Sendo assim, o processo de encapsulamento deve apenas interferir no contexto apresentado por tal método. Caso seja necessária a modificação da declaração da função, tal tarefa deve ser executada de modo a não afetar a integração entre os agentes externos conectados ao componente envolto pelo *Wrapper*.

Para que a interoperabilidade entre os componentes do sistema seja mantida, é realizado um desvio para que agentes externos possam estabelecer conexões com as possíveis funcionalidades de componentes envoltos por estruturas *Wrappers*. Isso deve-se a uma função implantada pelo Mecanismo Wrapper (definida na seção 4.4.5), que faz o redirecionamento de requisições direcionadas aos possíveis componentes falhos para seus respectivos *Wrappers* gerados. Assim, torna-se possível que os *Wrappers* sejam implantados na aplicação sem que haja a reestruturação do sistema na qual a problema esteja estabelecido.

Nos tópicos subjacentes é definido de maneira mais prática a utilização do mecanismo desenvolvido, onde na Seção 4.4 é realizada uma descrição de sua estrutura e funcionalidades; Seção 4.5 como foi realizado o desenvolvimento do mecanismo; e Seção 4.6 o mecanismo sendo aplicado em um ambiente em que se estabelece um sistema crítico simulado.

4.3. Trabalhos Relacionados

Nesta seção é descrito um trabalho cujo o contexto apresentado pelo mesmo se identifica com o descrito por nesta monografia, onde são abordados conceitos que envolvam o controle da falha em tempo de execução e a recuperação do sistema inibindo maiores transtornos ao fluxo de dados de um aplicação crítica.

4.3.1. Intelligente Health Monitoring of Critical Infrastructure System

Polycarpou et al (2010) descreve em seu trabalho o desenvolvimento de agentes cooperativos inteligentes que possam diagnosticar possíveis falhas em aplicações de grau crítico e assim estabelecer um controle tolerante a falhas nestes tipos de sistemas.

O trabalho faz uma abordagem conceptiva relacionando a importância de sistemas críticos atrelado ao cotidiano social atual, onde aponta os possíveis cenários de sua atuação e os retornos que possibilitam à sociedade por meio da prestação de seus serviços. Assim, caso

irregularidades venham a surgir na aplicação procedimento para controle, isolamento e diagnóstico do problema devem ser tomados de maneira ágil e eficaz evitando possíveis contratempos futuros.

A proposta do trabalho é o desenvolvimento de um algoritmo que interaja entre os componentes da aplicação crítica, na qual possa identificar precocemente e isolar irregularidades que possam levar o sistema a uma catástrofe. Para que isso seja possível é proposta a utilização da técnica de redundância, na qual possibilita a existência de extensões de componentes nativos da aplicação.

Isso possibilita que componente existentes caso venham apresentar falhas sejam trocados por extensões próprias, onde pode ser uma extensão estática (sem modificação na estrutura original) ou dinâmica (com alterações em sua estrutura) a nível de hardware ou software. Assim, o sistema pode voltar a operar sem complicações em seu processo tendo isolada a falha de outros componentes da aplicação mantendo o fluxo de dados sob controle.

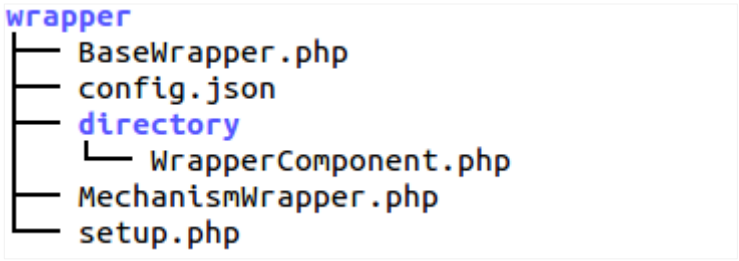
4.4. Interação com o Mecanismo

Nesta seção é feita uma abordagem referente a utilização do Mecanismo *Wrapper*, onde é realizada a explicação de sua estrutura e a especificação das funcionalidades disponibilizadas pela ferramenta.

A ferramenta desenvolvida é uma forma de possibilitar a automatização do processo de encapsulamento realizado por *Wrapper* à componentes falhos encontrados na aplicação (como definido na seção 4.2.2). Assim, a ferramenta dispõem de funcionalidades que agilizam o processo de geração de *Wrappers* e controle do mesmo diante da aplicação, onde pode ser realizadas a ações de criação, listagem e remoção de componentes *Wrappers*. Também é possível indicar o repositório no qual o mecanismo realizará a monitoração de possíveis componentes(Figura 5) por meio da funcionalidade de configuração fornecida pela ferramenta.

A seguir é mostrada a estrutura de arquivos que o mecanismo proporciona quando aplicado em um sistema.

Figura 7 - Estrutura do Mecanismo Wrapper.



Fonte: Autoria Própria.

A Figura 7 é compotas por arquivos que proporcionam o funcionamento da ferramenta quando utilizada em uma determinada aplicação. A seguir é feita uma breve descrição da função de cada componente encontrada dentro desta estrutura.

O *BaseWrapper.php* é o arquivo em que a arquitetura base dos componentes que podem sofrer a intervenção da ferramenta deve ser especificada.

O *config.json* é o arquivo em que são salvas as configurações de mapeamento para que o mecanismo possa explorar os componentes da aplicação em uma determinado local dentro do sistema.

O *diretory* é o repositório na qual o mecanismo deve armazenar os componentes wrappers gerados. O nome do repositório refere-se ao nome do diretório mapeado no momento da configuração da ferramenta. Por exemplo: Caso seja informado o diretório denominado "*libraries*" o nome do *diretory* (uma forma representativa para esta explicação) passa a ser "*libraries"*.

O *WrapperComponent.php* refere-se ao componente wrapper gerado pelo mecanismo, na qual sua estrutura é preparada para realização do encapsulamento de um determinado componente da aplicação.

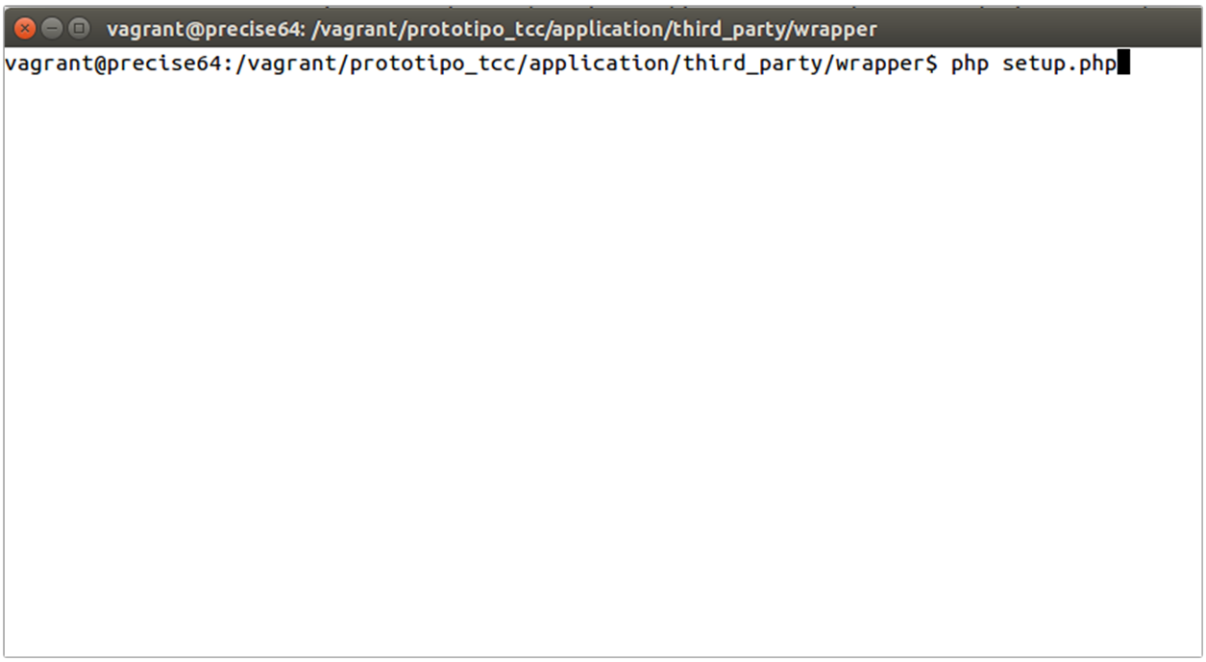
O *MechanismWrapper.php* refere-se a classe na qual disponibiliza as principais funcionalidades para o gerenciamento de wrappers da ferramenta.

O *setup.php* é o arquivo responsável por executar o Mecanismo Wrapper.

O mecanismo disponibiliza de um interface simples acessada via terminal (CLI), onde são apresentadas as funcionalidades para criação, listagem, remoção de wrappers e a configuração da ferramenta. A execução do mecanismo é feita por meio do script *setup.php* localizado no diretório raiz da ferramenta.

Na Figura 8, é mostrado o procedimento realizado através do terminal de comando para execução do mecanismo.

Figura 8 - Executando o Mecanismo Wrapper.



Fonte: Autoria Própria.

Ao realizar o primeiro acesso ao mecanismo, é solicitado para o usuário que especifique o diretório e o seu caminho de origem possibilitando que a ferramenta faça um mapeamento de sua estrutura. Desta forma, o mecanismo obtém o conhecimento sobre quais componentes podem sofrer a intervenção dos respectivos wrappers gerados pela ferramenta.

A Figura 9, ilustra o processo de configuração do mecanismo ao realizar a primeira interação do usuário com a ferramenta.

Figura 9 - Configuração Inicial (Mecanismo Wrapper).

```
================================================================================
                PROJETO DE TCC - CONFIGURAÇÃO
================================================================================

Informe o diretório a ser mapeado: libraries

Informe o caminho do diretório: ../../libraries/
```

Fonte: Autoria Própria.

Tais informações são armazenadas em uma estrutura JSON e salva em um arquivo denominado *config.json* no diretório em que está localizado o executável do mecanismo.

A estrutura JSON é utilizada para o armazenamento das informações requeridas no momento da configuração da ferramenta, assim possibilitando um baixo nível de processamento pelo programa ao salvar, modificar ou recuperar tais informações.

O Código Fonte 2, apresenta a estrutura JSON em que são salvos os dados de

configuração.

Código Fonte 2 - Estrutura JSON para Configuração do Mecanismo.

```
{
    "dir": "libraries",
    "path_dir": "..\/..\/libraries\/"
}
```

A estrutura consta com dois índices *"dir"* e *"path_dir"*. O índice *"dir"* referência o diretório em que estão localizados os possíveis componentes que possibilitam a intervenção do mecanismo em seus processos. Já o índice *"path_dir"* é utilizado para que o mecanismo consiga realizar com maior precisão o mapeamento dos componentes em processos atribuídos as suas funcionalidades (exemplo: validação da existência do componente para geração do componente *wrapper*).

Após a realização da configuração do mecanismo é possível obter acesso ao menu principal, onde são mostradas as seguintes funcionalidades: Gerar *Wrapper*; Listar *Wrapper*; Remover *Wrapper*; Configurações e Sair.

A Figura 10 ilustra o Menu Principal da ferramenta e suas funcionalidades. Para acessar tais funcionalidades o usuário deve informar o respectivo número atribuído a ela no Menu Principal.

Figura 10 - Menu Principal (Mecanismo Wrapper).

```
================================================================================
                PROJETO DE TCC - MECANISMO WRAPPER
================================================================================

[1]     Gerar Wrapper
[2]     Listar Wrapper
[3]     Remover Wrapper
[4]     Configurações
[0]     Sair

Informe a opção:
```

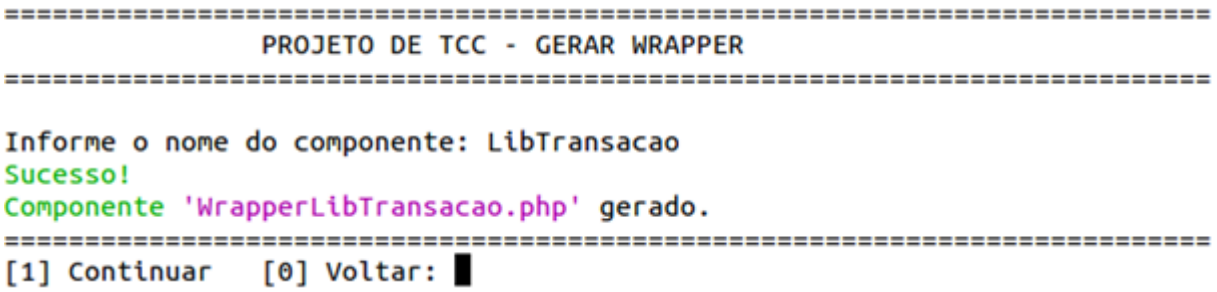
Fonte: Autoria Própria.

Nas subseção subjacentes são descritas tais funcionalidades disponibilizadas pelo mecanismo.

4.4.1. Gerar Wrapper

A Figura 11, ilustra o procedimento realizado para a geração de wrapper para um componente específico da aplicação.

Figura 11 - Gerando um Wrapper (Mecanismo Wrapper).



Fonte: Autoria Própria.

A funcionalidade Gerar Wrapper é responsável por proporcionar a criação dos componentes wrappers do mecanismo. Tal função necessita que o usuário informe o nome de um determinado componente, na qual possa ser encontrado de acordo com as informações fornecida no momento da configuração da ferramenta.

Ao ter acesso ao nome do componente, a funcionalidade realiza uma busca no repositório configurado para validar a existência do mesmo. Caso o dado seja válido, o mecanismo realiza o preparo da estrutura base que deve ser pré-definida pelo usuário no arquivo BaseWrapper.php.

O arquivo BaseWrapper.php é uma forma do mecanismo saber qual a estrutura base dos componentes que se encontram no repositório para que se possa gerar os seus respectivos wrappers.

O Código Fonte 3 mostra a estrutura definida no arquivo BaseWrapper.php para realização do encapsulamento do componente informado à funcionalidade.

Código Fonte 3 - Arquivo BaseWrapper.php.

```
1  <?php
2  require_once APPPATH.<path_require>;
3
4  class Wrapper<class> extends <class>
5  {
6      public function __construct()
7      {
8          parent::__construct();
9      }
10 }
```

A estrutura do arquivo BaseWrapper.php estabelece tags que servem como identificadores do componente que se encontram em processo encapsulamento pelo mecanismo. A tag <*path_require*> indica qual a localização em que o componente se encontra dentro do repositório pré-configurado para que possa ser inserido no script quando executado. A tag <*class*> define o nome do componente na qual o wrapper está sendo gerado, assim possibilitando a denominação do componente wrapper e também no processo de encapsulamento realizado pela ação *extends* (herança em PHP).

Todo componente wrapper gerado deve conter a nomenclatura *"Wrapper"* seguida pelo nome do componente envolvido. Por Exemplo: *"WrapperLibTransacao"*.

O Código Fonte 4, mostra o resultado após o termino da funcionalidade abordada nesta seção.

Código Fonte 4 - Componente Wrapper.

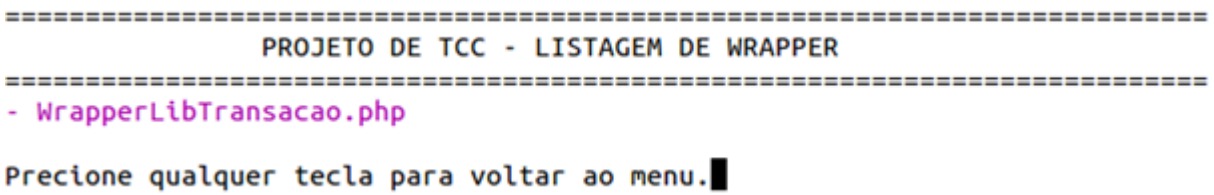
```php
<?php

require_once APPPATH.'libraries/LibTransacao.php';

class WrapperLibTransacao extends LibTransacao
{
    public function __construct()
    {
        parent::__construct();
    }
}
```

Após o componente wrapper ser gerado é possível que o desenvolvedor trabalhe em sua estrutura modificando ou acrescentando novas funcionalidades ao componente envolto, na qual possibilite a resolução do processo falho do componente encapsulado.

4.4.2. Listar Wrapper

A Figura 12, ilustra a tela na qual é possível realizar a listagem dos wrappers gerados pelo mecanismo.

Figura 12 - Listagem de Wrappers (Mecanismo Wrapper).



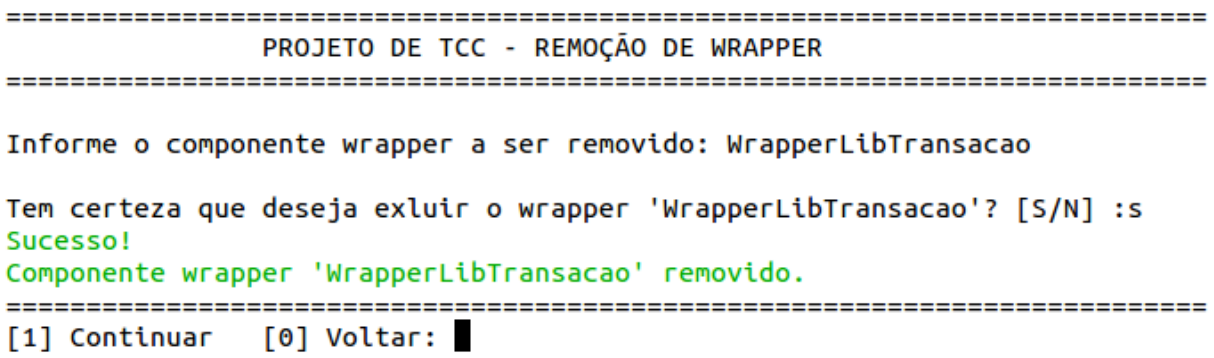
Fonte: Autoria Própria.

A funcionalidade de Lista Wrapper tem acesso ao repositório em que estão localizados todos os componentes wrappers gerados pela ferramenta. Assim, quando solicitada ela realizada uma varredura no repositório retornando todos os componentes wrappers que estão em operação na aplicação.

4.4.3. Remover Wrapper

A Figura 13, ilustra a tela na qual é possível aplicar a funcionalidade de remoção dos componentes wrappers gerados pelo mecanismo.

Figura 13 - Remoção de Componente Wrapper (Mecanismo Wrapper).



Fonte: Autoria Própria.

A funcionalidade Remover Wrapper, como o próprio nome condiz é o processo utilizado pelo mecanismo para deletar componentes wrappers sem mais utilidade para a aplicação. A funcionalidade requer que o usuário informe o nome do componente wrapper gerado e caso localizado o solicitante tem a opção de remover ou não o componente. Caso confirme a operação, o wrapper será removido do repositório liberando o componente até então envolto por suas funcionalidades

4.4.4. Configuração

A Figura 14, ilustra a o processo realizado para a reconfiguração do mecanismo caso o usuário necessite modificar as determinadas informações referente ao repositório mapeado pela ferramenta.

Figura 14 - Configuração (Mecanismo Wrapper).

```
================================================================================
                PROJETO DE TCC - CONFIGURAÇÃO
================================================================================
Diretório:      'libraries'
Caminho:        '../../libraries/'
Deseja reconfigurar o mecanismo? [S/N] :s

Informe o diretório a ser mapeado: libraries

Informe o caminho do diretório: ../../libraries/
Sucesso!

Configuração realizada.
================================================================================
```

Fonte: Autoria Própria.

A funcionalidade de configuração possibilita que o usuário refaça os mesmos procedimentos realizados no primeiro acesso ao mecanismo, no qual irá informar o diretório e o seu caminho para que a aplicação possa realizar o mapeamento de seus componentes e possibilitar a execução das tarefas descritas anteriormente.

4.5. Desenvolvimento do Mecanismo

Para o desenvolvimento do mecanismo foram utilizadas a linguagem de programação PHP versão 5.4.45 junto ao framework CodeIgniter versão 3.0.2.

Atualmente o mecanismo atende especificamente a linguagem PHP junto a estrutura proporcionada pelo framework CodeIgniter, porém nada impede que ele seja desenvolvido para outras linguagens e frameworks encontrados no mercado atualmente.

4.5.1. Classe MechanismWrapper

A classe *MechanismWrapper* disponibiliza dos principais métodos para que o mecanismo faça a inserção de Wrappers na estrutura da aplicação na qual se encontra implantado.

O Código Fonte 5, apresenta uma breve estrutura da classe *MechanismWrapper* utilizada para a automatização das funcionalidades do mecanismo na aplicação.

Código Fonte 5 - Classe *MechanismWrapper*

```
1   <?php
2   class MechanismWrapper
3   {
4       private $arrConfig    = array();
5       private $argExtension = '.php';
6
7       public function __construct($argExtension = '')
8       {...
14      }
15
16      /**...
```

Nas seções subsequente são apresentados os códigos fontes implementados das funcionalidades desenvolvidas para o gerenciamento dos componentes Wrappers criados a partir do mecanismo. Também é feita uma menção em relação aos código implementados no framework CodeIginiter para que o mecanismo possa operar em sua arquitetura.

4.5.2. Função de Criação do Wrapper

O Código Fonte 5, apresenta o método de criação de componentes Wrappers disponibilizado pela classe *MechanismWrapper*, na qual recebe por parâmetro o nome do componente que pretende ser envolto pela estrutura Wrapper.

Código Fonte 6 - Método de Criação de Componente Wrapper.

```php
    public function createWrapper($argComponent)
    {
        $argSubDir = '';
        //Verifica se o componente existe.
        $argSubDir = $this->searchComponent($argComponent);
        if ($argSubDir === false) {...
        }
        //Criar o diretório caso não exista.
        if (!file_exists($this->arrConfig['dir']))
            mkdir($this->arrConfig['dir'], 0775);

        $componentWrapper = "Wrapper{$argComponent}{$this->argExtension}";
        //Verifica de o wrapper já existe para o componente informado.
        if (is_file($this->arrConfig['dir'].'/'.$componentWrapper)) {...
        }

        //Monta a estrutura base do wrapper para o componente informado.
        $fileWrapperBase = file_get_contents('BaseWrapper.php');
        $requrireDir     = "'/".$this->arrConfig['dir']."/{$argSubDir}{$argComponent}'";
        $arrSearch       = array('<class>', '<path_require>');
        $arrReplace      = array($argComponent, $requrireDir);
        $fileWrapperBase = str_replace($arrSearch, $arrReplace, $fileWrapperBase);

        //Salva o componente wrapper.
        if (!file_put_contents($this->arrConfig['dir'].'/'.$componentWrapper, $fileWrapperBase)) {
            $message = sprintf("Falha ao gerar o componente '%s'.", $componentWrapper);
            return array('success' => false, 'message' => $message);
        }

        $message = sprintf("Componente \033[0;35m'%s'\033[0m gerado.", $componentWrapper);
        return array('success' => true, 'message' => $message);
    }
```

A ação do método estabelece validações de acordo com as configurações pré-estabelecidas ao mecanismo, na qual busca validar a existência do componente no diretório mapeado pelo mecanismo e a já existência de um possível Wrapper para o próprio.

4.5.3. Função de Busca de Wrapper

Código Fonte 7 - Método de Recuperação de Componentes Wrappers Gerados.

```php
    public function listWrapper()
    {
        $arrWrapper = array_diff(scandir($this->arrConfig['dir']), array('.', '..'));

        if (!$arrWrapper)
            return array('success' => false, 'message' => 'Nenhum componente wrapper encontrado.');

        return array('success' => true, 'data' => $arrWrapper);
    }
```

O Código Fonte 7, faz um mapeamento do diretório em que se encontra os componentes Wrappers gerados para aplicação, e caso sejam encontrado é retornado ao solicitante um *array* contendo os nomes dos componentes Wrappers.

4.5.4. Função de Remoção de Wrapper

Código Fonte 8 - Método de Remoção de Wrapper.

```php
public function removeWrapper($argComponent)
{
    if (!unlink("{$this->arrConfig['dir']}/{$argComponent}.php")) {
        return array(
            'success' => false,
            'message' => sprintf("Falha ao excluir o componente '%s'.", $argComponent));
    }

    return array(
        'success' => true,
        'message' => sprintf("Componente wrapper '%s' removido.", $argComponent));
}
```

O Código Fonte 8, apresenta o método de remoção de possíveis componentes Wrappers sem utilidade para aplicação, no qual é informado o nome do Wrapper que deve ser removido pelo mecanismo.

4.5.5. Preparo do Framework

Nesta subseção são descritos os procedimentos tomados para a adequação do framework ao mecanismo e os trechos de códigos implementados para proporcionar maior automatização aos processos.

Código Fonte 9 - Condição de Componente Wrapper.

```php
$filepath = $path.'libraries/'.$subdir.$class.'.php';

//Verificaçao da existência do componente wrapper
//para o componente que esteja sendo carregado.
$pathWrapper = $this->loadWrapper($class, 'libraries');
if ($pathWrapper) {
    $prefix   = 'Wrapper';
    $filepath = $pathWrapper;
}
```

O Código Fonte 9 acima é implementado dentro da estrutura do método *"_ci_load_library"* proporcionado pelo framework. Esse trecho de código tem a função de monitorar todos os componentes que estão sendo carregados pela aplicação em tempo real e verificar se eles possuem um componente Wrapper implementado no mecanismo. Caso a condição seja verdadeira, esse código modifica o caminho do componente original e passa informar o caminho do Wrapper gerado, assim a aplicação passa a processar o Wrapper e não

mais o componente original.

Código Fonte 10 - Método de Carregamento do Wrapper.

```
111     * verifica se a classe possui algum Wrapper.
112     * @param  String $argClass
113     * @return String|Boolean
114     */
115    private function loadWrapper($argClass, $argDirectory)
116    {
117        $arrFilePath = array_diff(scandir(PATH_WRAPPER.$argDirectory), array('.', '..'));
118
119        if ($key = array_search("Wrapper{$argClass}.php", $arrFilePath)) {
120            return PATH_WRAPPER."{$argDirectory}/{$arrFilePath[$key]}";
121        }
122
123        return false;
124    }
125 }
```

No Código Fonte 10 é realizada a procura do Wrapper referente ao componente em fase de carregamento pela aplicação (item monitorado pelo Código Fonte 9). Caso seja constatada a existência do Wrapper para o componente monitorado a funcionalidade deve retornar o caminho referente a tal, na qual possibilita que o Código Fonte 9 realize a modificação do trajeto referente ao componente em fase de carregamento pelo framework.

4.6. Aplicando o Mecanismo Wrapper

Para demonstrar a aplicação do mecanismo foi desenvolvido um ambiente em que pudesse ser simulado um sistema crítico voltado ao negócio. Assim, utilizou-se como base para a criação do protótipo os serviços de Intermediação de Pagamentos (Capítulo 3), cujo o foco foi implementar um módulo transacional que apresenta-se características em comuns aos de tais sistema.

O desenvolvimento do protótipo foi realizado utilizando a linguagem PHP junto ao framework CodeIgniter e para o armazenamento de dados utilizou-se o gerenciador de banco de dados MySQL.

4.6.1. O Protótipo

A aplicação simula um módulo transacional que realiza o processamento financeiro de pedidos realizados em comércios eletrônico que utilizam a plataforma para o gerenciamento de suas operações financeiras (terceirização).

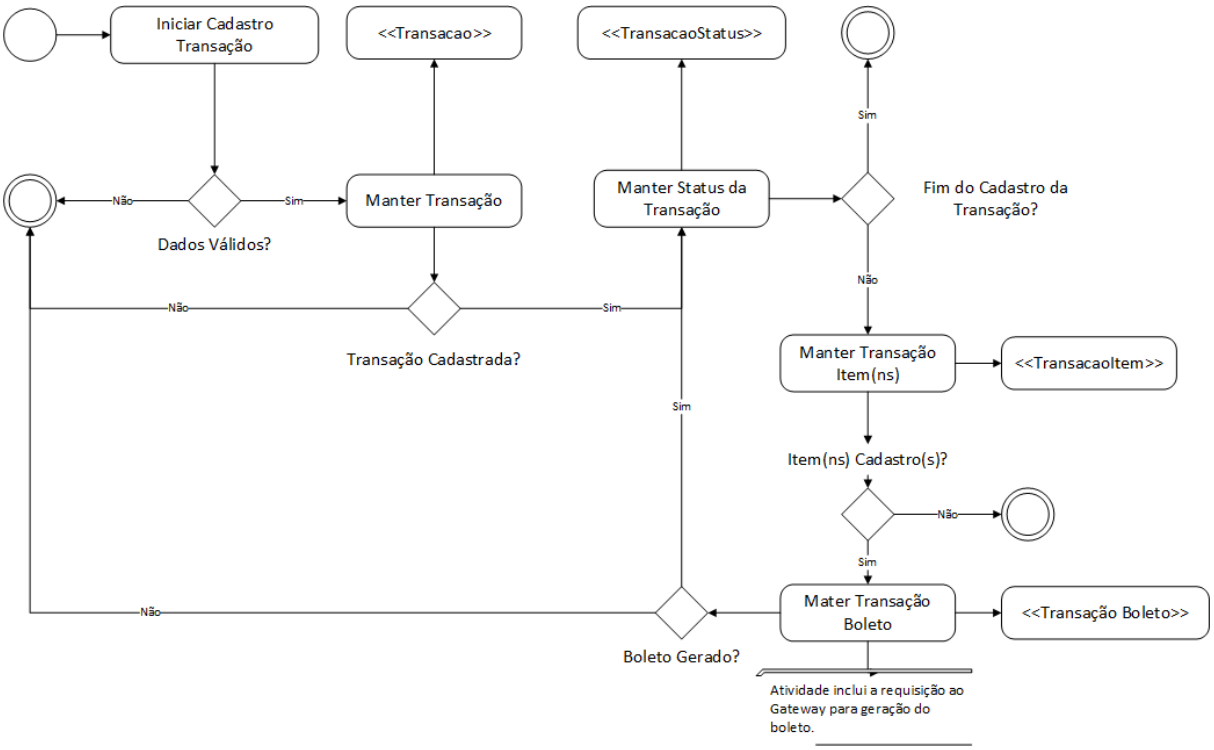
A plataforma utiliza o boleto bancário como forma de pagamento entre lojistas e

compradores cadastrados em sua base de dados. Para geração dos boletos são realizadas requisições a um serviço externo (*Gatewray de Pagamento*), em que são executados todos os procedimentos necessários para a emissão do documento.

Os *Gatewray de Pagamento* são serviços disponibilizados em servidores remotos por operadores financeiros que autorizam os pagamentos de transações realizadas por meio do mercado eletrônico. Tais serviços são base para a existência das aplicações referente a intermediação de pagamentos, pois é com a utilização de tais serviços que se torna possível exercer a estratégia de negócio por trás dessas aplicações.

A seguir é ilustrado em um diagrama de atividade todos os procedimentos realizados pela plataforma para a geração de uma transação de um determinado pedido.

Figura 15 - Diagrama de Atividade Processo de Criação de Transação.



Fonte: Autoria Própria.

Para a geração de uma transação é necessário que se informe alguns parâmetros que possibilite a sua criação por parte da plataforma. Dentre os dados informados estão contidos informações referente ao lojista, comprador, produtos e valores associados ao frete e desconto.

A Tabela 2 apresenta a estrutura de requisitos esperados pela plataforma para que se possa realizar a geração de um transação.

Tabela 2 - Requisitos para Gera uma Transação.

Parâmetro	Tipo	Descrição
lojista	*Array*	*Informações do lojista.*
lojista.cnpj	*String*	*CNPJ do lojista contratante.*
lojista.token	*String*	*Token de autenticação do lojista com a plataforma.*
comprador	*Array*	*Informações do comprador.*
comprador.cpf	*String*	*CPF do comprador.*
comprador.nome	*String*	*Nome do comprador.*
comprador.email	*String*	*E-mail do comprador.*
produto	*Array*	*Informações de produto(s) da compra.*
produto.id	*Int*	*Identificador do produto pelo lojista.*
produto.descricao	*String*	*Descrição do produto.*
produto.quantidade	*Int*	*Quantidade adquirida referente ao produto informado.*
produto.preco	*Float*	*Valor unitário do produto.*
valor_frete	*Float*	*Valor referente ao frete de envio do pedido.*
valor_desconto	*Float*	*Valor referente ao desconto do pedido.*

Fonte: Autoria Própria.

As transações geradas a partir da plataforma são atribuídas a status que determinam qual a sua situação atual perante ao sistema. Na Tabela 3 são descritos os tipos de status e suas definição.

Tabela 3 - Tabela de Tipos de Status (Protótipo).

Tipo de Status	Descrição
INICIADA	Transação gerada, porém sem boleto atribuído a ela.
AGUARDANDO PAGAMENTO	Transação liberada para a ser paga pelo consumidor.
APROVADA	Transação paga.
CANCELADA	Transação não paga.
BLOQUEADA	Transação em disputa entre lojista e comprador.

Fonte: Autoria Própria.

A aplicação depende da interação com serviços externos para a execução de sua principal forma de pagamento que são os boleto bancários. Os pagamentos dos pedidos realizados por meio da plataforma só são liberados aos compradores caso seja constatada a emissão do boleto referente a sua respectiva transação. Neste caso, a estabilidade entre os dois

sistema torna-se um fator importante para o funcionamento do plataforma.

No Anexo 1 é ilustrado o Diagrama de Entidade-Relacionamento gerado para o protótipo, no qual é possível ter noção da estrutura de dados do ambiente desenvolvido.

4.6.2. O Problema Crítico

Com o decorrer da execução do sistema, observou-se que em alguns períodos o serviço de geração de boleto passou a apresentar instabilidade em seu funcionamento, no qual proporcionava resultados incoerentes ao aguardado pela aplicação (protótipo). Mesmo possuindo tratamentos prévios para os possíveis retornos do *gateway*, por algum descuido da equipe de desenvolvimento no momento da implementação do processo a aplicação passou a persistir dados de transações de forma incoerentes ao esperado, assim disponibilizando informações divergentes aos usuários do sistema.

As transações afetadas por tal falha mantiveram o ciclo normal da aplicação saindo do status *INICIADA* para *AGUARDANDO PAGAMENTO* agravando ainda mais o problema. Como descrito na Tabela 3, o status *AGUARDANDO PAGAMENTO* define que uma transação já esteja disponível para o pagamento, assim possibilitando que os usuários possuam acesso aos seus boletos.

A Figura 16 ilustra um conjunto de transações afetadas pela falha, na qual permitiu a persistência de registros de boletos (coluna *transacao_boleto_id*) com inconsistência de dados para determinadas transações.

Figura 16 - Registros de Transações com Falha.

transacao_id	STATUS	transacao_boleto_id	codigo_barra	valor_documento	boleto	data_processamento
1375	Aguardando Pagamento	991		0.00	OB	(NULL)
1374	Aguardando Pagamento	990		0.00	OB	(NULL)
1373	Aguardando Pagamento	989		0.00	OB	(NULL)
1370	Aguardando Pagamento	986		0.00	OB	(NULL)
1366	Aguardando Pagamento	982		0.00	OB	(NULL)
1365	Aguardando Pagamento	981		0.00	OB	(NULL)
1364	Aguardando Pagamento	980		0.00	OB	(NULL)
1360	Aguardando Pagamento	976		0.00	OB	(NULL)
1359	Aguardando Pagamento	975		0.00	OB	(NULL)
1355	Aguardando Pagamento	971		0.00	OB	(NULL)

Fonte: Autoria Própria.

Em uma situação em que o lojista venham a depender de determinadas informações para que possa gerir seus processos de vendas em seu comércio, tal falha pode acabar dificultando a realização de seus negócios proporcionando transtornos junto aos seus clientes. A plataforma também passa a proporcionar desconfiança entre seus usuários não conseguindo assegurar que sua aplicação corresponda de forma eficiente as dimensões que constituem a propriedade de confiança em sistema. Desta forma, torna-se essencial que tal aplicação

disponibilize de ferramentas que possam restituir suas operações em curto prazo impossibilitando o agravamento de tal situação.

4.6.3. Intervenção do Mecanismo

Para a identificação da falha foram realizados suítes de teste baseados nos requisitos necessários para geração de transações por meio da plataforma(Tabela 4), no qual foi possível identificar a origem do problema e sua localização na estrutura do sistema.

O problema estava atrelado a um retorno não esperado pela plataforma, assim possibilitando que tais informações fossem persistidas no sistema sem uma validação prévia. A falha localizava-se no componente responsável pela geração do boleto para a transação, no qual possibilitava que informações com conteúdo em “branco” fossem inseridas na base de dados (Figura 16).

A seguir é ilustrado a estrutura do método responsável pelo surgimento da falha.

Código Fonte 11 - Método de Origem da Falha.

```
37      /**
38       * Gera pagamento da transação.
39       * @param  Array $arrTransacao
40       * @return Boolean
41       */
42      protected function pagamentoBoleto($arrTransacao)
43      {
44          $arrBoletoInfo = $this->prepararBoleto($arrTransacao);
45
46          $this->CI->load->library('LibGateWay');
47          $arrRequest =  array('boleto' => json_encode($arrBoletoInfo));
48          $arrResponse = $this->CI->libgateway->gateway(TIPO_PAGAMENTO_BOLETO, $arrRequest);
49
50          $arrResponse = json_decode($arrResponse, true);
51
52          if (!$arrResponse)
53              return false;
54
55          $arrBoleto = array(
56              'transacao_id'       => $arrTransacao['id'],
57              'codigo_barra'       => str_replace(array(' ', '.'), '', $arrResponse['codigo_barra']),
58              'valor_documento'    => isset($arrResponse['valor_documento']) ? $arrResponse['valor_documento'] : 0.00,
59              'cedente'            => CEDENTE,
60              'agencia'            => CB_AGENCIA,
61              'conta'              => CB_CONTA,
62              'data_vencimento'    => $arrBoletoInfo['data_vencimento'],
63              'data_documento'     => $arrResponse['data_processamento'],
64              'data_processamento' => $arrResponse['data_processamento'],
65              'boleto'             => htmlspecialchars_decode($arrResponse['boleto_layout']));
66
67          $this->CI->load->library('LibTransacaoBoleto');
68          $this->CI->libtransacaoboleto->setModel('Modeltransacaoboleto');
69
70          if (!$this->CI->libtransacaoboleto->inserir($arrBoleto))
71              return false;
72          return true;
73      }
```

Fonte: Autoria Própria.

O método *“pagamentoBoleto”* apresentado acima é uma das funcionalidades que

complementam a atividade “Manter Transação Boleto” (Figura 15). Tal método é responsável em pegar o retorno do *gateway* e criar a estrutura de persistência para as informações retornadas pelo serviço gerando assim um registro de boleto para transação. Como a condição referente a linha 52 era invalidada o processo prosseguia o seu fluxo criando a estrutura (*Array $arrBoleto*) e persistindo na base de dados (Linha 78) dando o processo como válido ao sistema.

Para que a falha pudesse ser contida em curto prazo de tempo foi realizado a aplicação do mecanismo ao componente *”LibPagamento”* detentor do método falho, assim possibilitando a realização da interrupção imediata da falha crítica.

A Figura 17 apresentada o componente Wrapper gerado a partir do mecanismo para que a falha pudesse ser contida.

Figura 17 - Encapsulando o Método Falho.

```
================================================================================
                PROJETO DE TCC - MECANISMO WRAPPER
================================================================================

[1]     Gerar Wrapper
[2]     Listar Wrapper
[3]     Remover Wrapper
[4]     Configurações
[0]     Sair

Informe a opção:2

================================================================================
                PROJETO DE TCC - LISTAGEM DE WRAPPER
================================================================================
- WrapperLibPagamento.php

Precione qualquer tecla para voltar ao menu.
```

Fonte: Autoria Própria.

Com o componente *WrapperLibPagamento*(Anexo 2) gerado foi possível realizar ajustes ao método *“pagamentoBoleto”* sem que a estrutura do componente original (*LibPagamento*) sofresse modificações diretamente, assim diminuindo a possibilidade do surgimento de outros possíveis problemas ao sistema.

Foi realizada uma solução paliativa à funcionalidade falha do componente, no qual impedisse que informações inconsistentes fossem persistidas no sistema evitando a aprovação de transação para o pagamento sem a emissão de seu respectivo boleto.

Código Fonte 12 - Solução ao Componente Falho.

```
41          /*
42          Solução paliativa aplicada pelo Mecanismo Wrapper para correção da inserção
43          de informações inconsistentes na base de dados
44          */
45          $this->CI->db->trans_begin();
46          if (!$this->CI->libtransacaoboleto->inserir($arrBoleto) || empty($arrBoleto['codigo_barra'])) {
47              $this->CI->db->trans_rollback();
48              return false;
49          }
50          $this->CI->db->trans_commit();
51          return true;
```

Fonte: Autoria Própria.

No Código Fonte 12 é mostrado a intervenção realizada por meio do componente Wrapper gerado pelo mecanismo, na qual utiliza-se de métodos referente ao Processo Transacional (seção 3.2) disponibilizado pelo framework para o controle das inserções falhas. Na linha 45 do código fonte é realizado a abertura do controle transacional (*trans_begin()*) da base de dados, na qual irá gerenciar todas as modificações realizadas no banco de dados a partir da sua declaração. Para validação dos dados retornados é colocada uma segunda cláusula na condição *"if"* verificando se a chave referente ao código de barras do boleto não encontra-se vazia. Caso a condição seja falsa, será realizado um *rollback (trans_rollback())* inibindo que a inserção de registros inconsistentes sejam persistidas na base de dados. Caso contrário, é realizada a inserção do registro referente boleto referente a transação *(trans_commit()).*

As transações que não obtiveram a emissão de seus boletos permaneceram com status *INICIADA*, até que o serviço externo se estabilize e possa responder de acordo com os dados esperado pela aplicação. Isso possibilita a resolução do problema em curto prazo até que medidas cabíveis para soluções preventivas pela aplicação sejam desenvolvidas e implantadas.

A Figura 18 ilustra o resultados obtidos por meio da aplicação do mecanismo junto a sua solução paliativa, na qual transações realizadas no momento da instabilidade e que não obtiveram sucesso na emissão de seus respectivos boletos mantiveram o status de *INICIADA* sem que fossem gerados registros de boletos (*transacao_boleto_id*) para tais.

Figura 18 - Registros de Transações sem Falhas

transacao_id	status	transacao_boleto_id	codigo_barra	valor_documento	boleto	data_processamento
1460	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)
1459	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)
1452	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)
1451	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)
1450	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)
1447	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)
1446	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)
1443	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)
1441	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)
1437	Iniciada	(NULL)	(NULL)	(NULL)	(NULL) OK	(NULL)

Fonte: Autoria Própria.

A Figura 19 apresenta registros que obtiveram sucesso após a aplicação da solução proporcionada pelo mecanismo.

Figura 19 - Registro de Transações Consistentes

transacao_id	STATUS	transacao_boleto_id	codigo_barra	valor_documento	boleto	data_processamento
1462	Aguardando Pagamento	1078	00191234132345671724010403O(	7685.40	<!DOCTYPE h... 41K	2015-10-31 12:52:12
1461	Aguardando Pagamento	1077	00191234132345671724010403O(	3907.60	<!DOCTYPE h... 41K	2015-10-31 12:52:11
1458	Aguardando Pagamento	1074	00191234132345671724010403O(	643.10	<!DOCTYPE h... 41K	2015-10-31 12:52:03
1457	Aguardando Pagamento	1073	00191234132345671724010403O(	5426.60	<!DOCTYPE h... 41K	2015-10-31 12:52:03
1456	Aguardando Pagamento	1072	00191234132345671724010403O(	4087.80	<!DOCTYPE h... 41K	2015-10-31 12:52:01
1455	Aguardando Pagamento	1071	00191234132345671724010403O(	8741.30	<!DOCTYPE h... 41K	2015-10-31 12:51:59
1454	Aguardando Pagamento	1070	00191234132345671724010403O(	6582.90	<!DOCTYPE h... 41K	2015-10-31 12:51:59
1453	Aguardando Pagamento	1069	00191234132345671724010403O(	4202.40	<!DOCTYPE h... 41K	2015-10-31 12:51:57
1449	Aguardando Pagamento	1065	00191234132345671724010403O(	6251.60	<!DOCTYPE h... 41K	2015-10-31 12:51:46
1448	Aguardando Pagamento	1064	00191234132345671724010403O(	1697.70	<!DOCTYPE h... 41K	2015-10-31 12:51:45

Fonte: Autoria Própria.

4.7. Considerações finais

A solução proposta por este trabalho, possibilita que ações de controle de falhas e riscos sejam realizadas de forma eficaz diante da problemática apresentado pelo mesmo. De acordo com os resultado apresentados pela Seção 4.6.3, o mecanismo desenvolvido (definido na seção 4.2) mostrou-se capaz de agir em uma situação crítica (apresentada na seção 4.6.3) e reestabelecer a operabilidade de processos falhos de forma ágil e prática.

Mesmo que a proposta do mecanismo não seja proporcionar soluções definitivas ao sistema defeituoso. Em momentos de falhas como os de aplicações críticas, impedir que tais irregularidades se propaguem pela aplicação é um ponto essencial para evitar maiores transtornos ao sistema e seus envolvidos.

Destaca-se neste trabalho a utilização do componente Wrapper (definido no capítulo 2) como base para o desenvolvimento do mecanismo (definido na seção 4.2.2), possibilitando que o objetivo deste trabalho pudesse ser alcançado.

A abordagem desta solução baseia-se no modelo de solução referente a interrupção imediata da falha crítica, onde além de possibilitar a realização da correção também conseguiu manter a operação do sistema sem que sua paralização fosse necessária.

CONCLUSÃO

Considerando o nível de periculosidade que um sistema crítico possa vir apresentar caso problemas surjam em seus processos em momento de execução, assim colocando em riscos ao seu ambiente, seus usuários e ao próprio negócio como contextualizado nesta pesquisa. Esse trabalho teve como objetivo apresentar uma proposta de ferramenta que possibilite que situações de ricos sejam controladas de maneira ágil em determinadas aplicações evitando o seu agravamento perante aos fatores (diretos ou indiretos) relacionados à tais sistemas.

A contribuição deste trabalho está atrelada a segurança de software, na qual enfatiza o resguardo do sistema de possíveis problemas que possam vir a comprometer sua e execução e assim prevenindo-o de possíveis agravamentos que possam ocorrer em seus processos.

Atualmente o Mecanismo Wrapper está disponível apenas para aplicações desenvolvidas por meio do framework CodeIgniter, assim impossibilitando que sistemas fora dos padrões desta arquitetura implante tal ferramenta como apresentado neste trabalho.

Mesmo sendo aplicado em uma situação específica como abordado neste trabalho, o projeto do mecanismo pode ser levado a outros cenários, onde deverão adequar-se para que possam realizar a implantação da ferramenta em sua estrutura.

Como melhorias futuras para a ferramenta, pretende-se deixar o processo de encapsulamento de forma totalmente automatizada, onde além de estabelecer a estrutura Wrapper do componente involucro a ferramenta também possibilitará que a funcionalidade na qual a falha esteja definida no componente Wrapper gerado sem a necessidade da prática manual como ocorre atualmente.

Os resultados obtidos por esse trabalho possibilitou que falhas fossem contidas por meio da implantação do Wrapper. A abordagem feita pelo exemplo do capítulo 4 (seção 4.2.2) demonstra a aplicação do Wrapper em um cenário, no qual irregularidades poderiam causar possíveis danos ao sistemas e seus usuários. Assim, com a intervenção ágil do mecanismo foi possível impedir a disseminação da falha pelo ambiente com a inserção de uma solução paliativa por meio do Wrapper assegurando a execução do sistema até a implantação de uma correção definitiva ao sistema.

Esse trabalho impôs pequenas dificuldades em sua elaboração, sendo uma delas a busca por casos reais referente a sistemas críticos de negócios voltados a web. Por se tratar de informações confidencias, em muitos casos, as organizações tendem a manter tais situações em sigilo por questões de segurança, assim dificultando o encontro de variados casos para

complementar a problemática abordada por esta pesquisa.

O desenvolvimento deste trabalho possibilitou a formação de uma melhor compreensão a respeito da importância de se manter aplicações críticas seguras e livre de problemas e quais as consequências que podem vir a proporcionar caso venham a apresentar problemas na execução de seus processos. Além disso, a elaboração desta pesquisa possibilitou a desenvolver cientifico proporcionando um bagagem de conhecimento para futuras pesquisas que possam vir a serem realizadas.

REFERÊNCIAS

BUJA, G. et al. **Dependability of Safety-Critical Systems**. [S.l.]: IEEE International Conference on Industrial Technology (ICIT). 2004. p. 1561-1566.

CARZANIGA, A. et al. **A Characterization Framework for Software Deployment Technologies**. Department of Computer Science. [S.l.]. 1998.

COUPAYE, T.; ESTUBLIER, J. **Foundations of Enterprise Software Deployment**. Software Maintenance and Reengineering. Gieres: [s.n.]. 2000.

DENG, Y.; FRANKL , P.; CHEN, Z. **Testing Database Transaction Concurrency**. Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on. [S.l.]: [s.n.]. 2003. p. 184 - 193.

EDWARDS, S. **A Framework for Practical, Automated Black-Box Testing of. Component-Based Software.**, Virgina, 2001. Disponivel em: <http://people.cs.vt.edu/~edwards/downloads/edwards-wapatv-00.pdf>. Acesso em: 27 Outubro 2015.

EDWARDS, S. et al. **Contract-Checking Wrappers for C++ Classes**. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. [S.l.]: [s.n.]. Novembro 2004.

ENGINEERS, THE INSTITUTE OF ELECTRICAL AND ELECTRONICS. Systems and software engineering - Vocabulary. **International Standart ISO/IEC/IEEE/24765**, 15 dez. 2010.

GAMMA, E. et al. **Padrões de Projeto:** soluções reutilizáveis de software orientado a objetos. Tradução de Luiz A. Meirelles Salgado. Porto Alegre: Bookman, 2000.

GARCIA-MOLINA, H.; ULLMAN, J. D.; WIDOM, J. **Database Systems:** The Complete Book. 2ª. ed. New Jersey: Pearson Education Inc, 2009.

HECHT, H. **Dependability Improvement for Critical Digital Systems**. 17th IEEE Pacific Rim International Symposium on Dependable Computing. Culver City: [s.n.]. 2011.

JING, Y. **On-line Payment and Security of E-commerce**. International Symposium on Web Information Systems and Applications. Handan: [s.n.]. 2009.

JUNXUAN, Z. **Research On E-Payment Model**. International Conference on E-Business and E-Government. Shanghai: [s.n.]. 2010. p. 339 - 341.

KANDASAMY, N.; HAYES, J. P.; MURRAY, B. T. **Tolerating Transient Faults in Statically Scheduled Safety-Critical**. Reliable Distributed Systems, 1999. Proceedings of the 18th IEEE Symposium on. Lausanne: [s.n.]. 19 Outubro 1999. p. 212 - 221.

KIM, C. et al. **An empirical study of customers perceptions of security and trust**. Electronic Commerce Research and Applications 9. [S.l.]: Elsevier B.V. 2009. p. 84–95.

LAWRENCE, J. D. **Software Safety Hazard Analysis**. Livermore: Lawrence Livermore National Laboratory, v. 2, 1995.

LOVISI FILHO, E.; CUNHA, M. D. **Uma Proposta de Sistemática para a Análise de Segurança de Software Crítico Embarcado Aeroespacial**. QUATIC 2001. Lisboa: [s.n.]. 2001. p. 147-153.

MELO, A. D.; NASCIMENTO, M. G. F. **PHP Profissional:** aprenda a desenvolver sistemas profissionais orientados a objetos com padrões de projeto. São Paulo: Novatec Editora Ltda., 2007.

MORAES, R. L. D. O. **Importância da Arquitetura de Sistemas Baseados em Componentes para**. Campinas: Centro Superior de Educação Tecnológica - CESET. 2003.

MOURA, C. A. T. D. **Uma Estratégia de Análise de Segurança de Software para Aplicações Críticas**. São José dos Campos: Divisão de Pós-Graduação do Instituto. Tecnológico de Aeronáutica, 1996.

NETO, J. R.; DOS SANTOS, M. C. N. **Teste de Software – Uma Introdução e Exemplos**. [S.l.]: [s.n.]. Outubro 2001.

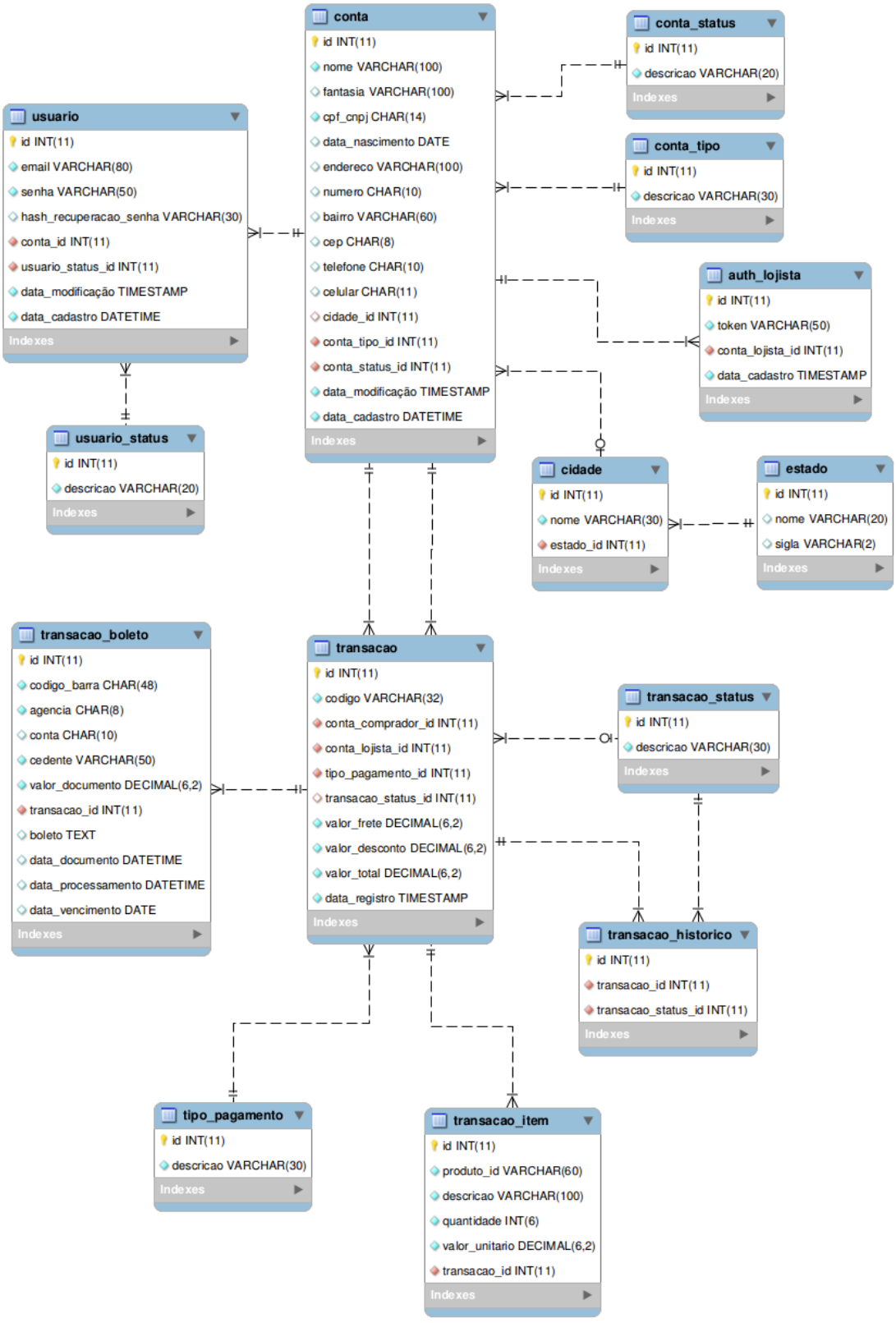
OLIVEIRA, A. D. **Análise inteligente de falhas para apoiar decissões estratégicas em projetos de sistemas críticos**. São Paulo: Escola Politénica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais, 2009. Disponivel em: <http://www.teses.usp.br/teses/disponiveis/3/3141/tde-21122009-141400/pt-br.php>. Acesso em: 27 out. 2015.

SECRETÁRIA DE COMÉRCIO E SERVIÇOS. **Informativo da Secretaria de Comércio e Serviços**. [S.l.]. 2015.

SOMMERVILLE, I. **Engenharia de Software**. 6ª. ed. [S.l.]: Addison Wesley Longman Limited, 2003.

TU, D. et al. **A Method of Log File Analysis for Test Oracle**. Scalable Computing and Communications; Eighth International Conference on Embedded Computing, 2009. SCALCOM-EMBEDDEDCOM'09. International Conference on. Dalian: IEEE. Setembro 2009. p. 351 - 354.

ANEXO 1 – DIAGRAMA ENTIDADE-RELACIONAMENTO (PROTÓTIPO)



ANEXO 2 – COMPENETE WRAPPERLIBPAGAMENTO

```php
<?php
require_once APPPATH.'/libraries/LibPagamento.php';

class WrapperLibPagamento extends LibPagamento
{
    public function __construct()
    {
            parent::__construct();
    }

    /**
     * Gera pagamento pelo boleto.
     * @param  Array $arrTransacao
     * @return Boolean
     */
    protected function pagamentoBoleto($arrTransacao)
    {
            $arrBoletoInfo = $this->prepararBoleto($arrTransacao);

            $this->CI->load->library('LibGateWay');
            $arrRequest =  array('boleto' => json_encode($arrBoletoInfo));
            $arrResponse = $this->CI->libgateway->gateway(TIPO_PAGAMENTO_BOLETO, $arrRequest);

            $arrResponse = json_decode($arrResponse, true);

            if (!$arrResponse)
                    return false;

            $arrBoleto = array(
                    'transacao_id'       => $arrTransacao['id'],
                    'codigo_barra'       => str_replace(array(' ', '.'), '', $arrResponse['codigo_barra']),
                    'valor_documento'    => isset($arrResponse['valor_documento']) ? $arrResponse['valor_documento'] : 0.00,
                    'cedente'            => CEDENTE,
                    'agencia'            => CB_AGENCIA,
                    'conta'              => CB_CONTA,
                    'data_vencimento'    => $arrBoletoInfo['data_vencimento'],
                    'data_documento'     => $arrResponse['data_processamento'],
                    'data_processamento' => $arrResponse['data_processamento'],
                    'boleto'             => htmlspecialchars_decode($arrResponse['boleto_layout']));

            $this->CI->load->library('LibTransacaoBoleto');
            $this->CI->libtransacaoboleto->setModel('Modeltransacaoboleto');

            /*
            SoluÃ§Ã£o paliativa aplicada pelo Mecanismo Wrapper para correÃ§Ã£o da inserÃ§Ã£o
            de informaÃ§Ãµes inconsistentes na base de dados
            */
            $this->CI->db->trans_begin();
            if (!$this->CI->libtransacaoboleto->inserir($arrBoleto) || empty($arrBoleto['codigo_barra'])) {
                    $this->CI->db->trans_rollback();
                    return false;
            }
            $this->CI->db->trans_commit();
            return true;
    }

}
```