ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de Sistemas e Computação

PROJETO BASEADO NO NAGIOS PARA MONITORAÇÃO DE AUTORIZADORES DE TEF

Trabalho de Conclusão de Curso

Engenharia da Computação

Rogério Pereira Pontual

Orientador: Prof. Sérgio Murilo Maciel Fernandes

UNIVERSIDADE
DE PERNAMBUCO

Rogério Pereira Pontual

PROJETO BASEADO NO NAGIOS PARA MONITORAÇÃO DE AUTORIZADORES DE TEF

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, junho de 2009.

Agradecimentos

Agradeço primeiramente a meus pais, Fausto Falcão Pontual e Marcina Pereira Pontual, por proporcinarem uma boa educação que me permitiu chegar a este ponto satisfeito com seu resultado.

Agradeço a todos os professores de Computação da Universidade de Pernambuco que contribuiram com a minha formação acadêmica.

Ao professor Sérgio Murilo Maciel Fernandes, pelo estímulo ao raciocínio crítico no desenvolvimento deste projeto e por sua orientação cujo resultado foi a produção de um texto mais científico.

Por fim, aos meus amigos da POLI e de outras épocas pelas conversas que me apoiaram reafirmando ou com críticas construtivas a respeito de decisões acadêmicas ou profissionais.

Resumo

Os sistemas de Transferência Eletrônica de Fundos (TEF) pertencem à categoria de sistemas de missão crítica uma vez que suas falhas têm como potencial um grande prejuízo financeiro ou de reputação. São os sistemas responsáveis pela autorização de pagamentos, principalmente com cartão magnético ou com chip, que processam desde a captura dos dados da transação até ao envio dos comprovantes.

Dentre as ferramentas de suporte, de modo geral, vale destacar as de monitoração que, quando construídas e utilizadas com eficácia, notificam os administradores de sistema sobre alguma suspeita de mau funcionamento, ou ainda, fornecem informações detalhadas aos engenheiros de planejamento de capacidade.

Motivado por uma necessidade real de uma empresa local fornecedora de softwares de TEF, pelo alto valor comercial e nível de especialização dessas ferramentas, neste projeto é desenvolvida uma ferramenta potencialmente de baixo custo, o Vigilante TEF (VTEF), para monitoração para Autorizadores de TEF. Este sistema de TEF é que decide em última instância a aprovação das transações.

O VTEF tem o propósito de identificar eventos que virtualmente afetam a alta disponibilidade do Autorizador de TEF. Isto será feito pela detecção de erros e cálculo de métricas de estabilidade e desempenho.

Para observação e tratamento destes eventos é utilizado o servidor de monitoração Nagios, por conta de sua disponibilidade bibliográfica e por ser um software *open source*, gratuito e reconhecido pela imprensa especializada. Alinhado com os objetivos do projeto, apresenta as seguintes características:

- Permite identificação de ocorrências através de interface gráfica;
- Notifica alertas através de email ou celular;
- Apresenta uma plataforma estável, com dez anos em desenvolvimento;

Abstract

The Electronic Funds Transfer (EFT) systems belong to the category of mission critical systems since its failures have the potencial of a big financial loss or reputation damage. These systems are responsible for payment authorization made mainly with magnetic or smart cards. Their processing goes from the transaction data capturing to the formatting and transmission of the sale/payment receipt.

It's worthy to highlight the monitoring tools, among the support tools used with this systems, because they notify systerm administrators about suspected bad system behaviour or because they give detailed information to capacity planning engineers. These benefits are achieved only when they are built and used effectively.

This project is motivated by a actual need of a local company that sells EFT software, as well as by the high commercial value and high especialization of these tools. This project develops a low cost tool, the Vigilante TEF (VTEF), proper to EFT Authorizers monitoring. This EFT system is the last one in the chain and is responsible for the transactions approval.

The VTEF has the purpose of identifying events that can affect the EFT Authorizer high availability. This will be done by detecting erros and calculating performance and stability metrics.

The VTEF will use the Nagios monitoring software in order to observe and handle these events. Some of the reasons for this are the spread out bibliography related to Nagios and the fact that it's a free open source software recognized by the especialized press. Besides this, aligned with immediate and future project targets, it presents the following features:

- *It allows the identification of incidents through a web graphical user interface;*
- *It notifies alerts through email or cellphone messages;*
- *It's a mature plataform with ten years of development;*

Sumário

Índice de Figuras

Tabela de Símbolos e Siglas

TEF – Transferência Eletrônica de Fundos

NSCA - *Nagios Service Check Acceptor* (Aceitador de Verificações de Serviços do Nagios)

CPU – *Central Processing Unit* (Unidade Central de Processamento)

TCP – *Transmission Control Protocol* (Protocolo de Controle de Transmissão)

IP – *Internet Protocol* (Protocolo de Internet)

PoS – *Point of Sale* (Ponto de Venda – PDV)

SQL – *Structured Query Language* (Linguagem Estruturada de Consulta)

API – *Application Program Interface* (Interface de Programa de Aplicação)

SGBD – Sistemas de Gerenciamento de Banco de Dados

TMR – Tempo Médio de Resposta

TPPS – Transações Processadas por Segundo

TPS – Transações por Segundo

Capítulo 1
Introdução

Um dos desafios da Engenharia é atingir o equílibrio ideal não apenas entre custo e qualidade, ou agilidade de produção e novas características, mas também entre otimização e segurança. Porém, existe uma categoria de software na qual o desempenho pode ser tão importante quanto a confiabilidade. Tratam-se dos sistemas de missão crítica, caracterizados principalmente pelo potencial de impacto causado por uma falha. São vidas perdidas, no caso de uma pane em um sistema de controle aéreo, prejuízos financeiros ou de reputação, quando de um sistema de comércio eletrônico ou de outro que viabiliza o pagamento de uma compra.

Os sistemas de TEF (Transferência Eletrônica de Fundos) são aqueles que efetuam a autorização de pagamentos, principalmente com cartão magnético ou com chip, e são responsáveis desde a captura dos dados da transação até ao envio dos comprovantes. Devido à necessidade de alta disponibilidade e rapidez nas operações, a arquitetura desses sistemas evoluiu a ponto de envolver, fim a fim, vários componentes de hardware e software.

Uma pesquisa da Dataquest [1] aponta um alto crescimento na indisponibilidade, não planejada, de serviços de missão crítica, mesmo com a melhoria dos hardwares, dos processos de desenvolvimento e das ferramentas de suporte. Dentre essas ferramentas, vale destacar as de monitoração que, quando construídas e utilizadas com eficácia, notificam os administradores de sistema sobre alguma suspeita de mau funcionamento, ou ainda, fornecem informações detalhadas aos engenheiros de planejamento de capacidade.

Alguns dos fatores que podem influenciar negativamente o grau de adesão dessas ferramentas são o alto custo do software e o nível de especialização [2] destas. Em ambientes complexos de infra-estrutura, as falhas podem advir de várias áreas e de diversos modos em cada uma delas: em hardware, em configuração de Rede, em sistemas de Banco de Dados, por esgotamento instantâneo ou persistente de recursos, por erros de aplicação/programação, dentre outros [3].

Há uma frase clássica na Administração, um velho adágio, que resume esse contexto dizendo que “você não consegue gerenciar aquilo que você não consegue medir”. Logo, como forma de contribuir para solução deste problema, neste projeto de monografia será desenvolvida uma ferramenta, potencialmente de baixo custo, para monitoração de Autorizadores de TEF. O Autorizador é o software que decide, em última instância, sobre a aprovação da transação enviada pelo Servidor de TEF,

contendo, esta, os dados do meio de pagamento do cliente (ex: cartão de crédito), capturados no caixa de uma loja.

Será adotado o Nagios como servidor de monitoração, com o qual a ferramenta deverá se integrar, por conta de sua disponibilidade bibliográfica e por ser um software *open source*, gratuito e reconhecido pela imprensa especializada [4]. Além disso, alinhado com os objetivos do projeto, imediatos ou futuros, apresenta as seguintes características:

- Permite identificação de ocorrências através de interface gráfica;
- Notifica alertas através de email ou celular;
- Apresenta uma plataforma estável, com dez anos em desenvolvimento;
- Suporta plugins para extensão do alcance da monitoração.

1.1 Objetivo

Serviços de missão crítica precisam ser monitorados preventivamente de modo que sejam emitidos alertas em situações de risco. Eles requerem, em geral, uma resposta rápida e segura quanto à análise de casos e correção de *bugs*.

A proposta é que a ferramenta do projeto, o Vigilante TEF, permita a notificação quase instantânea de informações pertinentes a eventos no domínio de negócio do TEF, ou relacionados a erros de infra-estrutura, que possam sugerir ou apontar o impacto negativo no funcionamento do Autorizador de TEF.

Para enviar os dados desses eventos, em virtude da facilidade de integração com o Nagios, será utilizado um de seus programas complementares, o NSCA, de modo a viabilizar os seguintes requisitos funcionais da ferramenta:

- Notificar erros, ocorridos no Autorizador TEF, no processamento de regras de negócio relacionados a módulos como por exemplo de acatamento de cheque, cartões pré-pagos, pagamento de títulos, etc.
- Notificar erros, ocorridos no Autorizador TEF, classificados por tipo de API e que identifique a origem levando em consideração uma arquitetura em *cluster*;
- Calcular periodicamente indicadores de estabilidade do ambiente operacional, com base na frequência e resultado das mensagens do protocolo de aplicação específicas do TEF.

É natural que, como todo software do tipo servidor, sem operador humano, o AutorizadorTEF utilize arquivos de log para registrar as inconsistências. Todavia, se

esse arquivo contém pouca informação, isso dificulta o diagnóstico do erro e, por outro lado, havendo excesso, aumenta o custo de entrada e saída da aplicação e eleva a importância dos cuidados com gerenciamento dos logs, principalmente com relação à limpeza e *backups*. Dada essa condição, as notificações de monitoração propostas pelo projeto visam diminuir a dependência com relação aos arquivos de log na detecção de erros do aplicativo e de infra-estrutura.

1.2 Estrutura do trabalho

Este trabalho está organizado da seguinte maneira: os três primeiros capítulos abrangem os temas relacionados ao contexto do desenvolvimento da ferramenta; o quarto apresenta a metodologia de desenvolvimento adotada; no quinto são exibidos e avaliados os resultados. Por último, na conclusão, aborda-se de maneira geral se os objetivos foram atingidos e são sugeridos trabalhos futuros. O propósito de cada capítulo é descrito a seguir:

- Capítulo 2: Apresenta o Nagios, software servidor utilizado para tratar, acompanhar e notificar os eventos da monitoração.
- Capítulo 3: Expõe uma introdução ao TEF: sua arquitetura, protocolo de comunicação e fluxos que definem as transações eletrônicas.
- Capítulo 4: Define o planejamento para desenvolvimento do projeto e apresenta os artefatos de sua execução.
- Capítulo 5: Avalia o resultado dos testes planejados e analisa os impactos nas fábricas de software e de serviços.
- Capítulo 6: Recapitula a proposta e os resultados do projeto e fala dos possíveis trabalhos futuros.

Outro aspecto para melhorar a legibilidade do texto foi a adoção de um tipo diferente de letra para facilitar a percepção dos termos contidos em arquivo de configuração ou em código fonte. A mesma abordagem foi aplicada ao nome dos arquivos ou diretórios, por exemplo, `/usr/local/nagios` é o diretório padrão de instalação do Nagios.

Capítulo 2
NAGIOS

Nagios(R) é um software livre, gratuito, com maturidade na plataforma Linux/Unix, para monitoração de redes, aplicações e recursos computacionais tais como *CPU*, memória e espaço em disco. Em termos gerais sua função principal é realizar verificações configuráveis de *hosts* e serviços, locais ou remotos, e quando houver algo de errado nestes, permitir a notificação aos administradores de sistema e gerentes através de aviso sonoro, *e-mail*, pager ou mensagens de celular.

Sua principal vantagem, explorada em detalhes a seguir, é que ele não foi construído como uma peça monolítica de software, pretendendo dispor de todas características (*features*) de monitoração necessárias a uma ou mais categorias de aplicações ou equipamentos. Em vez disso, além de sua licença, *GNU General Public License Version 2*, dar condições legais para modificação de seu código fonte, sua estrutura modular permite a interoperabilidade com outras aplicações através de integrações simples, e consequentemente, a extensão de sua funcionalidade para atender a demandas específicas. Não obstante, não são poucas as opções embutidas (*built-in*) no Nagios. Sua instalação contém em torno de dez arquivos de configuração somando ao todo mais de cem parâmetros ou diretivas abrangendo várias etapas da monitoração, verificação, deteção de falha, ação corretiva, até a notificação. Algumas de suas características prontas são [5]:

a. Monitoração de serviços de rede (SMTP, POP3, HTTP, NNTP, PING, etc.);
b. Monitoração de recursos de *hosts* (carga do processador, espaço livre em disco, etc.);
c. Checagem paralela de serviços;
d. Habilidade de definir hierarquia de *hosts* e serviços permitindo a distinção entre aqueles que estão fora / indisponíveis e os não alcançáveis;
e. Notificação de grupo de contatos quando um problema ocorrer ou for resolvido;
f. Habilidade de definir manipuladores de eventos para disparar resoluções pró-ativas;
g. Rodízio automático de arquivo de log;
h. Interface web opcional para visualização de estados e administração.

Essa seria sua visão pragmática ou utilitária porém para tirar proveito de seu potencial e saber como melhor aplicá-lo à monitoração de serviços de TEF, faz-se

necessário abordar outros aspectos, cada um numa visão própria, que respondem as questões abaixo:

- Visão de Implantação: Quais as interfaces de integração com os demais componentes da solução?
- Visão Operacional: Quais os procedimentos ou conhecimentos necessários para instalar, configurar, executar e começar a utilizá-lo?

Entretanto, antes de responder a essas questões, algumas observações devem ser feitas. Apesar do Nagios possuir meios de monitorar máquinas Linux/Unix, servidores Netware, impressoras de rede e roteadores/switches, este estudo está delimitado à integração com máquinas Windows, onde roda atualmente o serviço Autorizador de TEF, utilizado na validação da solução final proposta. Sabe-se também que a disponibilidade de aplicações distribuídas depende do bom estado geral dos equipamentos e fluxos de rede. Contudo, estes não serão monitorados diretamente, mas através do reflexo de seu mau funcionamento neste serviço.

2.1 Visão de Implantação

Existem dois modos de monitorar um serviço ou *host* com o Nagios: ativamente ou passivamente. As checagens ativas são iniciadas por seu *daemon* e executadas em base regular, enquanto que as passivas são iniciadas por processos externos, locais ou remotos, cujos resultados são submetidos de forma arbritrária para o processamento do Nagios. Logo, a checagem passiva deve ser utilizada em pelo menos duas ocasiões:

- Para eventos de natureza assíncrona e que não podem, portanto, ser monitorados eficientemente realizando *polling* do seu estado, a exemplo de timeout de conexão TCP/IP ou outros erros de programas.
- Quando um serviço ou *host* está localizado atrás de um firewall cuja regra proíbe o recebimento de conexões para a finalidade em questão.

Para realizar os dois tipos de checagens remotamente, o Nagios dispõe de dois *addons*, que são softwares para estender sua capacidade de monitoração e integrá-lo a outras aplicações: são eles o NRPE (*Nagios Remote Plugin Executor*) e o NSCA (*Nagios Service Check Acceptor*).

Segue abaixo um detalhamento do NSCA já que neste projeto, apesar de serem executadas verificações síncronas, este *addon* foi suficiente como meio de reportar os eventos de monitoração ao Nagios.

NSCA

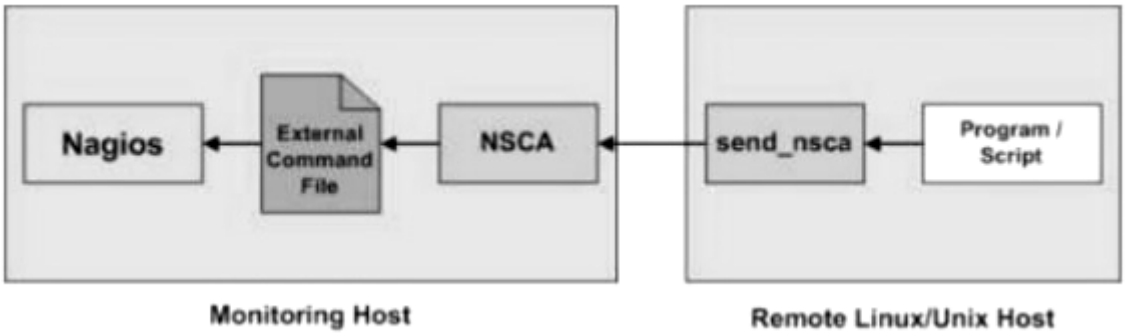


Figura 1. Checagem Passiva Remota [5]

Toda checagem passiva se inicia em um processo (programa ou *script*), geralmente remoto, que executa um programa chamado `send_nsca` passando como parâmetro os dados do item monitorado *(*`send_nsca_win32.exe` quando distribuído no projeto NSCA Win32Client do Nagios EXCHANGE [6]). Este então envia via TCP/IP os dados para o NSCA que formata uma mensagem do tipo `PROCESS_SERVICE_CHECK_RESULT` e a escreve no Arquivo de Comando Externo (External Command File) cuja implementação é de um *pipe* nomeado, na prática funcionando como uma fila, e que é criado pelo Nagios na sua inicialização e removido na finalização.

2.2 Visão Operacional

2.2.1 Instalação Básica

O procedimento de instalação e configuração básica dos *releases* Nagios 3.x é bem definido no Guia de Instalação Rápida da documentação oficial [5] para as distribuições Linux *Fedora Core 6*, *openSUSE 10.2* e *Ubuntu 6.10* (no experimento foi utilizado o *Ubuntu 8.04* para o qual o guia também foi válido). Ao final das instruções o resultado será um ambiente com:

- Nagios e os plugins oficiais instalados em `/usr/local/nagios`;
- Nagios configurado como monitor de alguns recursos do *localhost* (carga de CPU, espaço livre em disco, etc.);
- Interface web acessível por `http://localhost/nagios`.

Depois disso é necessário instalar o NSCA e seu respectivo módulo cliente:

- O NSCA conforme manual NSCA_Setup.pdf [7];
- O `send_nsca_win32` baixando o binário send_nsca_win32_bin.zip [8];

2.2.2 Configuração Básica

Intuitivamente a configuração de sistemas de monitoração pode ser vista como um meio de associar a *hosts*, ou serviços, uma ação inicial, que é um comando de verificação executado em certos momentos, e uma ação final, que são medidas de alerta ou corretivas, baseadas no estado resultante desta verificação.

Com o Nagios não é diferente mas vai além possibilitando a definição de vários tipos de particularidades: na verificação, a exemplo de um critério de detecção de falha para evitar alarmes falsos, na notificação dos contatos sobre quais eventos estes precisam ser informados, dentre várias outras.

Quanto à forma, a configuração é realizada através de arquivos de texto puro, situados por padrão em `/usr/local/nagios/etc/`, utilizando uma linguagem muito simples composta por três tipos de declarações:

TIPO DE DECLARAÇÃO	SINTAXE
Variáveis	`<nome>=<valor>`
Macros	`$<nome_macro>$=<valor>`

Definição de Objeto	`define <nome_objeto> {`
	`  <nome_diretiva1>  <valor_diretiva>`
	`  <nome_diretiva2>  <valor_diretiva >`
	`  ...`
	`  <nome_diretivaN>  <valor_diretiva >`
	`}`

Variáveis são opções de configurações pré-definidas que afetam o modo de operação do processo *deamon* ou dos CGIs da interface web e que são situadas respectivamente no arquivo de configuração principal (`nagios.cfg`) ou no de CGI (`cgi.cfg`).

As macros são parâmetros que podem ser definidos pelo usuário e são declaradas nos arquivos de recursos, definidos pela variável `resource_file`, que é por padrão apenas o `resources.cfg`.

Por último, objetos são todos elementos envolvidos na lógica de monitoração ou notificação [5]. Aqueles que são obrigatórios e necessários são declarados como exemplos pela instalação rápida. No arquivo `templates.cgi` encontra-se demonstrações da aplicação do conceito de herança, pela diretiva `use`, na qual um *host* ou serviço adquire as caracterísiticas de um outro objeto abstrato.

Na documentação encontra-se a definição das dezenas de variáveis que regulam vários aspectos comportamentais do Nagios. As macros são poucas e podem

ser analisadas, utilizadas ou declaradas sob demanda. Já os objetos, compõem a idéia central da monitoração. Segue uma descrição para cada tipo de objeto essencial à monitoração, e as possíveis associações entre eles representadas pelo diagrama UML da figura 2.

TIPO DE OBJETO ESSENCIAL	**DESCRIÇÃO**
`hostgroup`	Agrupamento de hosts para facilitar a visão pela interface web do estado de seus integrantes
`host`	Dispositivo físico na rede (servidor, estação de trabalho, roteador, impressora, etc.) a monitorar
`servicegroup`	Agrupamento de serviços com finalidade análoga ao `hostgroup`
`service`	Um elemento operacional de um `host` (um processo, funcionalidade ou recurso) e junto a este compõe os possíveis alvos de monitoração.
`contact`	Identifica uma pessoa que precisa ser notificada quando ocorre algum problema
`contactgroup`	Agrupamento de responsáveis por um ou mais *hosts* ou serviços
`command`	Define como executar um programa ou script que pode ser usado para checagem, notificação e tratamento de eventos (medida reativa)
`timeperiod`	Controla quando hosts e serviços devem ser monitorados e o período em que os contatos podem ser notificados

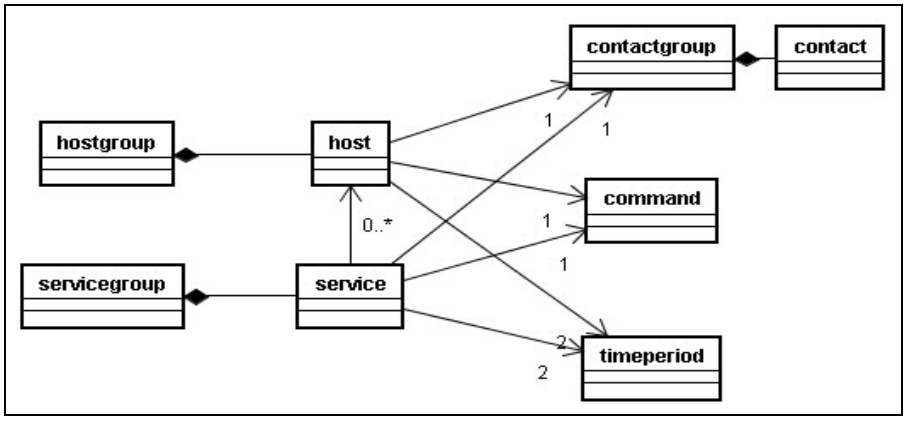


Figura 2. Cardinalidade/multiplicidade das declarações dos objetos básicos

Nota-se na figura 2 que um *host* ou serviço é verificado por um comando e possui um grupo de contatos responsáveis que recebem notificações na ocorrência de eventos escolhidos em suas diretivas. Os dois *timeperiods* são períodos de ativação, um para verificação e outro para notificação. Por fim, é demonstrado que a cardinalidade de um serviço para *host* é de 0 para N, ou seja, que é permitido declarar a verificação apenas de *host*, onde não interessaria nenhum de seus serviços, ou a verificação de um serviço para ser executada em um ou mais *hosts*.

2.2.3 Visualização Básica

A interface gráfica da web disponibiliza fundamentalmente páginas, cujas imagens encontram-se no Anexo A, que exibem o estado corrente dos *hosts* e serviços em monitoração mas também provê várias outras funcionalidades tais como:

- Ferramentas de tendências e dados históricos;
- Agendamento de *downtime*;
- Ativação ou desativação de checagens de serviços ou notificações;
- Meios para examinar as configurações atuais;
- Ferramentas de mapeamento gráfico do ambiente;
- Ferramentas para obter o *status* do *daemon* do Nagios.

O menu principal, os links e conteúdos das páginas são organizados de forma intuitiva tal que durante a navegação já se percebe o propósito de cada opção. Nesta seção são destacados os dados das páginas que usam uma terminologia definida pelo Nagios, ou por uma de suas regras, e que estão relacionados ao momento em que os administradores precisam visualizar a situação presente dos *hosts* e serviços (item **A**). Em seguida, é analisada criticamente a implementação da interface do ponto de vista do desempenho (item **B**).

A) Estado na terminologia Nagios

As verificações do Nagios podem ser realizadas, em cima de hosts e/ou serviços, por intermédio de plugins ou passivamente pela integração do NSCA.

Os *hosts* checados podem ficar em um dos três estados: *UP* (Ativo), *DOWN* (Inativo) ou *UNREACHABLE** (indeterminado ou inalcançável, quando o *host* no qual dependa, declarado com a diretiva `parent`, está inativo). As verificações dos plugins, por sua vez, retornam os resultados OK, *WARNING* (alerta), *UNKNOWN* (desconhecido) ou *CRITICAL* (crítico).

Como o Nagios traduz esses resultados nos estados, depende de duas etapas. Na primeira etapa o resultado é mapeado a um estado preliminar, que pode ser alterado pelo processamento da segunda etapa, de acordo com a tabela abaixo:

RESULTADO DO PLUGIN	**ESTADO PRELIMINAR**
OK	UP
WARNING	UP ou DOWN
UNKNOWN	DOWN
CRITICAL	DOWN

Na segunda, se o estado preliminar for DOWN, o Nagios tentará determinar se o estado do host é de fato DOWN ou UNREACHABLE. A distinção entre esses dois estados é importante porque permite ao administrador determinar a causa de um problema de rede mais rapidamente. A tabela abaixo apresenta a regra, baseada no estado do host pai, para definir o estado final do host após a verificação.

ESTADO PRELIMINAR	**ESTADO DO HOST PAI**	**ESTADO FINAL DO HOST**
DOWN	Pelo menos o de um pai é UP	DOWN
DOWN	Todos pais são DOWN ou UNREACHABLE	UNREACHABLE

Já a definição do estado de um serviço é determinada unicamente pelo resultado do plugin. No entanto, é preciso saber que tanto para este como para um *host* esses estados não são tomados como verdadeiros de imediato pois existe o conceito de tipo de estado que pode ser *SOFT* ou *HARD*.

O estado assume o tipo *SOFT* nas seguintes situações:

- Quando uma verificação resulta em estado não OK e esta não foi realizada o número de vezes especificado na diretiva `max_check_attempts`. No caso dos hosts, o não OK é o que não é *UP*.
- Quando um serviço ou host se recupera de um erro *SOFT*. Por exemplo, após a primeira verificação o serviço está em *UNKOWN* e na segunda está OK. Este OK fica marcado ainda como *SOFT*.

O estado assume o tipo *HARD* nas demais situações que são:

- Quando uma verificação resulta em estado de erro e já foi executada tantas vezes quanto definido em `max_check_attempts`.
- Quando a transição de estado ocorre de um erro para outro (ex: *WARNING* para *CRITICAL*).
- Quando uma verificação de serviço resulta em estado de erro e seu *host* correspondente está *DOWN* ou *UNREACHABLE*.
- Quando um host ou serviço se recupera de um estado *HARD*.
- Quando o resultado de uma verificação passiva é recebido ao menos que a variável `passive_host_checks_are_soft` esteja habilitada.

B) Implementação

A renderização de seu conteúdo dinâmico é baseada em CGI, que é uma tecnologia de programas escritos em C, cuja limitação de desempenho aplica-se apenas a um cenário de várias requisições simultâneas, sendo improvável ao contexto do projeto porque as empresas dificilmente possuem vários administradores de sistema ou de redes, para a mesma infra-estrutura. Ainda que este fosse o caso, seria necessário haver uma sincronização de *refresh* na tela destes.

Logo, as requisições dos links são tratadas por esses CGIs, que por padrão no Nagios requerem que o usuário esteja autenticado ao servidor web hospedeiro. A tabela abaixo contempla aqueles fundamentais à percepção do estado corrente.

NOME DO CGI	FUNÇÃO
`status.cgi`	O mais importante porque oferece uma visão geral e detalhada do estado de todos objetos monitorados
`tac.cgi`	Visão aérea de toda atividade de monitoração destacando os problemas tratados e os que precisam de atenção
`config.cgi`	Exibe detalhes de configuração dos objetos
`cmd.cgi`	Envia comandos ao processo do Nagios

2.2.4 Execução

Nagios é instalado como serviço e as explicações de como executá-lo são detalhadas no tópico *Starting and Stoping Nagios* [5]. Além disso é importante ler também o tópico *Verifying Your Configuration* já que o Nagios finaliza em caso de erro de configuração. É necessário executar manualmente os *deamons* de suporte às verificações ativa e passiva remotas, o NRPE e o NSCA, cujos procedimentos são encontrados nas respectivas documentações [9,7].

Capítulo 3
Transferência Eletrônica de Fundos

3.1 Definição

Os sistemas de engenharia podem ser definidos através de sua estrutura física ou lógica. Porém, outras vezes, prefere-se abordá-los de forma mais pragmática, por sua finalidade e funcionalidades. As diferentes definições de Transferência Eletrônica de Fundos, ou simplesmente TEF, encontradas na internet [10][11][12][13], geralmente se enquadram na segunda alternativa e se referem ao ato de efetuar qualquer tipo de transação financeira com o propósito de autorizar uma instituição a realizar um crédito ou débito em uma conta bancária.

O TEF é tipificado legalmente nos Estados Unidos através do Ato Regulatório E (Regulation E), a fim de estabelecer os direitos e deveres dos consumidores. Seu conceito destaca também os meios tecnológicos, no qual, TEF significa qualquer transferência de fundos iniciada através de um terminal eletrônico, um telefone ou um computador, para solicitar ou autorizar uma instituição financeira a debitar ou creditar uma conta de um consumidor [14]. Neste mesmo Ato, o conceito inclui os seguintes tipos de transferências:

i. Realizada por PDV (Ponto de Venda), que é o conjunto hardware e software de Automação Comercial responsável pelo registro dos itens do pagamento e pela integração com o Emissor de Cupom Fiscal (ECF);
ii. Transferências de caixas de atendimento automático;
iii. Depósitos ou saques diretos de fundos;
iv. Transferências iniciadas por telefone;
v. Transferências resultantes de transações com cartão de débito.

O projeto se concentrará nos itens (i) e (v), os quais ocorrem com frequência no cotidiano do consumidor brasileiro, principalmente no caso dos clientes de grandes varejistas, como supermercados. Esta frequência, devido à popularização dos meios de pagamento eletrônicos, exige do sistema características de alta disponibilidade e desempenho, uma vez que proporcionalmente à intensidade do fluxo de compras, alguns segundos de atraso no processamento do pagamento de cada cliente podem representar minutos na medida em que a fila dos caixas for aumentando. Supondo, por exemplo, uma fila de 22 pessoas para dois caixas, se cada um destes atrasar 30 segundos, e o tempo de atendimento dos dois caixas for

o mesmo, as 21ª e 22ª pessoas da fila terão esperado 300 segundos (6 minutos) a mais para serem atendidos. Esta situação se agrava ainda mais quando o sistema fica fora do ar, ou quando o tempo de totalização do cupom da compra, mais o tempo de pagamento, excede o tempo de chegada de um novo cliente à fila.

3.2 Arquitetura e Transações

De modo a minimizar a probabilidade dessas incoveniências, que diminuem o índice de satisfação do consumidor, a arquitetura dos sistemas de TEF evoluiu de modo que todos recursos tecnológicos para concretização da venda sejam pré-alocados (dedicados) [15], eliminando o tempo de preparação para processar as transações. Essa arquitetura é representada pela figura abaixo:

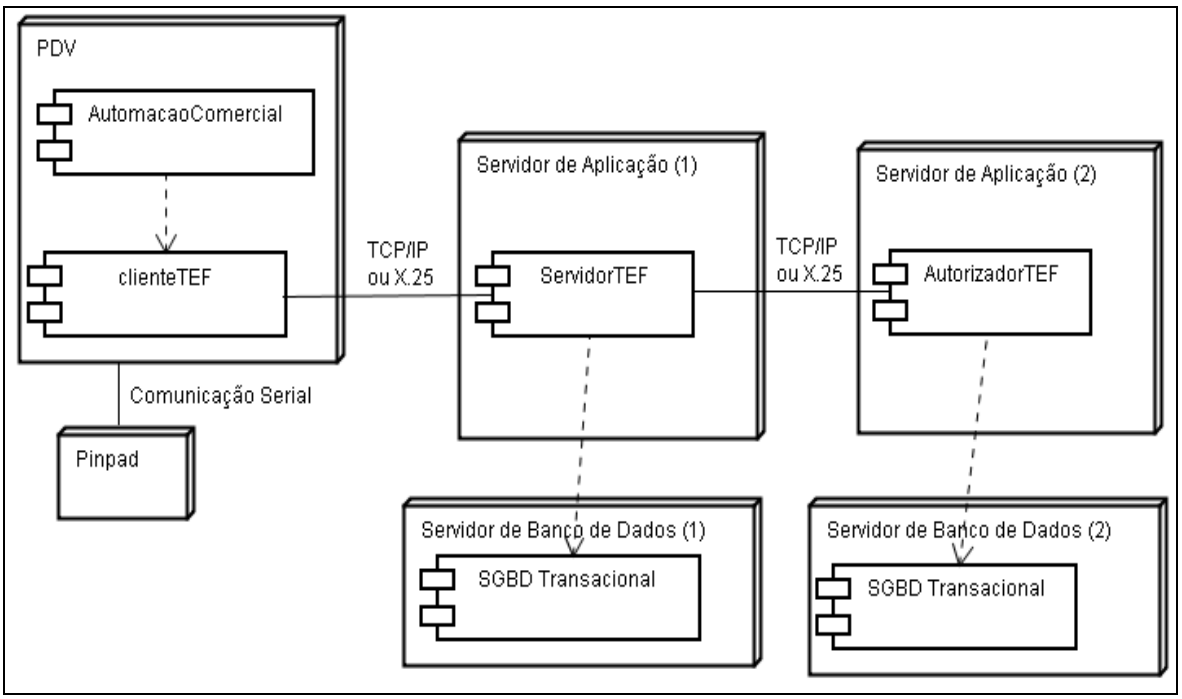


Figura 3. Diagrama UML de Implantação da arquitetura de TEF

Observa-se que o PDV, auxiliado pelo programa clienteTEF, realiza a captura dos dados do cartão do consumidor através do *Pinpad*, e envia a transação contendo estes dados mais os dados do pagamento. O servidor de TEF é uma aplicação que funciona como um *gateway* para que as transações cheguem ao Autorizador de TEF. Sua função é validar as mensagens recebidas de sua aplicação cliente, formatada em protocolo privado definido pela empresa fornecedora, e convertê-la para uma transação ISO 8583, que é um protocolo padrão internacional utilizado por redes administradoras de cartões, nacionais e internacionais. Esta mensagem ISO 8583 é enviada ao Autorizador de TEF, que a processa e deve respondê-la de imediato no mesmo protocolo e canal de comunicação se a transação foi aprovada com sucesso. No caso

em que ela não é aprovada, o AutorizadorTEF responde no campo "código de resposta", do protocolo, qual foi o motivo da negação da transação [16].

De maneira a viabilizar toda sorte de serviço financeiro, ou operação eltrônica financeira, realizada no caixa, os tipos de transações enviadas para o Servidor TEF, e para o AutorizadorTEF, são:

- Pagamento: pagamento de fatura, resultando na transferência financeira da conta do varejista para a conta do cedente do boleto ou da consessionária de luz, telefone, etc.;
- Compra/Venda: o portador do cartão paga por bens ou serviço prestado, afetando o saldo de sua conta bancária ou de crédito;
- Estorno: o varejista cancela a transação realizada anteriormente pelo portador do cartão, para que a administradora de cartão não faça a cobrança na fatura;
- Consulta: transação sem custo ao consumidor que retorna informações associadas a uma compra ou pagamento passado ou futuro (ex: limite de crédito ou juros de um pagamento);

Além destes, existem outros tipos como as transações que adicionam créditos a uma conta de celular pré-pago (*E-Top-Up*) ou as de *Cashback*, na qual o portador realiza um saque em sua própria conta através do caixa da loja. Essas e as demais devem ser classificadas em um dos tipos acima para que a ferramenta do projeto as identifique na monitoração. No caso dessas duas, por exemplo, a *E-Top-Up* pode ser considerada como um pagamento e a *Cashback* como uma compra, para efeito de monitoração.

Outro aspecto do TEF relevante ao entendimento dos resultados do Vigilante TEF é o do gerenciamento do *status* de cada transação, quanto ao seu estágio de processamento, realizado pelo Servidor e Autorizador de TEF. Para compreendê-lo é necessário conhecer quais mensagens ISO 8583 enviadas para cada tipo de transação, nos respectivos cenários em que são requisitadas. Nesse sentido, é valido esclarecer as seguintes questões, com relação a cenários de exceção, que justificam o envio de algumas mensagens do padrão ISO:

- Devido à possibilidade de falhas físicas ou lógicas na transmissão das mensagens, qual deve ser a ação do ServidorTEF caso a resposta do Autorizador não seja recebida ou recebida com atraso (*timeout*)?
- Caso a resposta seja uma aprovação, o saldo do consumidor já está comprometido mesmo quando o PDV não recebe a resposta, que contém os comprovantes para impressão?

Com o objetivo de tratar essas questões, de acordo com o padrão ISO 8583:87, toda transação de venda, pagamento ou estorno, que afetam o saldo do consumidor, apenas é concluída (com sucesso) se o ServidorTEF, ao receber uma resposta de aprovação da transação em questão, enviar com sucesso uma mensagem de confirmação (automatica) para o AutorizadorTEF. No entanto, se o ServidorTEF enviá-la e esta confirmação não for recebida ou processada pelo AutorizadorTEF, a transação ficará com *status* confirmada/concluída no ServidorTEF porém com *status* pendente de confirmação no AutorizadorTEF.

Essa situação requer a utilização de outro tipo de mensagem definida pelo protocolo ISO, com propósito de contigenciamento, que é a sonda. Esta mensagem é uma consulta periódica que o AutorizadorTEF faz ao ServidorTEF sobre o *status* de uma determinada transação que ficou pendente. De outro modo, caso o ServidorTEF não tenha enviado a confirmação, por alguma falha de processamento, a transação ficará pendente no próprio e exigirá um procedimento manual da equipe de suporte, contratada pelo varejista, para resolução da pendência utilizando uma ferramenta de retaguarda (*backoffice*). Nesta ferramenta, o usuário tem a opção de confirmar a venda (ou pagamento ou estorno), finalizando-a com sucesso caso seja recebida e processada pelo AutorizadorTEF, ou revertê-la, fazendo com que o ServidorTEF envie um outro, e último tipo, de mensagem de exceção que é o desfazimento (*reversal*).

As transações ISO8583 que demandam a confirmação para conclusão, porque afetam o saldo do consumidor, são chamadas de transações de 3 pernas (neste tipo são incluídas as de compra, pagamento e estorno). Nas seções a seguir, são apresentados diagramas que representam a troca de mensagens nos cenários discutidos acima e se aplicam a todas transações de 3 pernas. Ao lado de cada mensagem encontra-se o *status* da transação (A: Em Andamento, P: Pendente, C: Concluída, D: Desfeita), no respectivo sistema de TEF, ao receber ou enviar a mensagem.

Fluxo de Compra com sucesso

A figura 4 representa as mensagens ISO8583 transmitidas entre um ServidorTEF e um AutorizadorTEF para conclusão de uma transação de compra com sucesso.

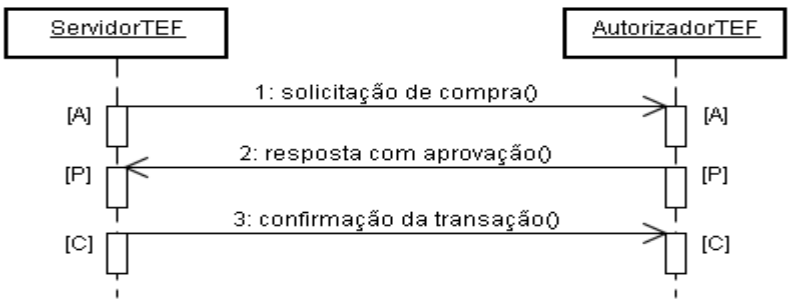


Figura 4. Sequência de mensagens ISO8583 para conclusão com sucesso

Fluxo de Confirmação através de Sonda

A figura 5 representa o cenário em que a mensagem de sonda ISO8583 confirma uma compra quando a mensagem de confirmação não foi recebida ou processada no AutorizadorTEF.

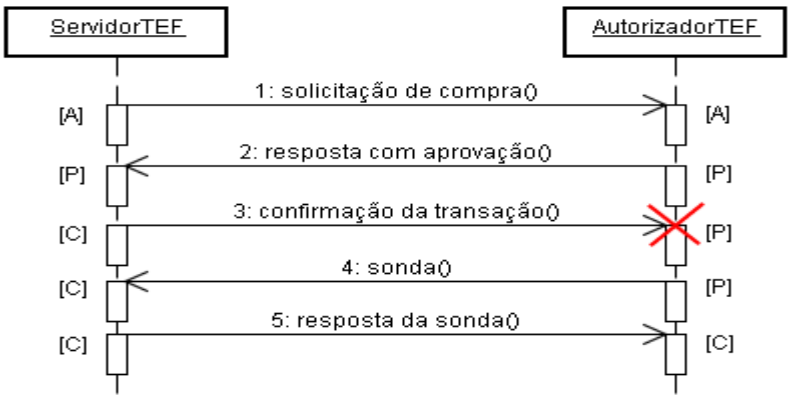


Figura 5. Sequência de mensagens ISO8583 para conclusão pela Sonda

Fluxo de Desfazimento de Transação

A figura abaixo representa o cenário em que a mensagem de confirmação não foi enviada e o estado da transação continuou pendente no ServidorTEF. A sonda não resolve neste caso porque ela não está com *status* confirmada no ServidorTEF então se o consumidor não levou a mercadoria ou fez outro pagamento, a loja pode desfazer a transação e consequentemente não será cobrada do consumidor.

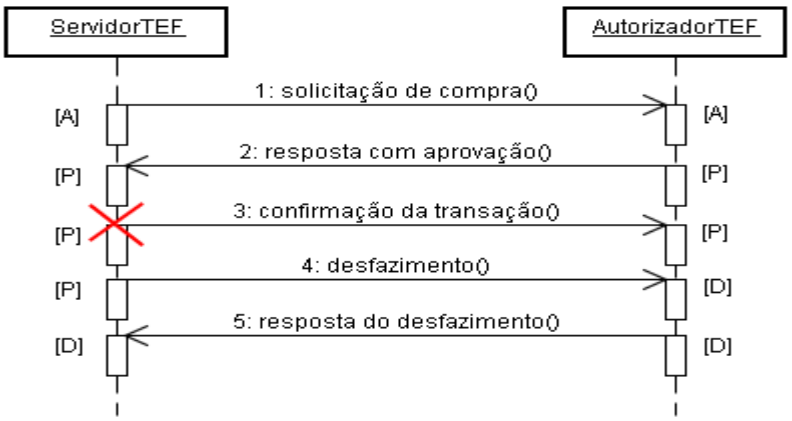


Figura 6. Sequência de mensagens ISO8583 para conclusão com Desfazimento

Capítulo 4
Desenvolvimento do Projeto

Este capítulo apresenta a metodologia e o desenvolvimento do projeto, abrangendo desde o planejamento, o qual foi definido como um guia para execução do projeto, até o relato do progresso, expondo o que foi produzido neste desenvolvimento.

4.1 Planejamento

O planejamento realizado consiste na adoção de uma metodologia baseada na abordagem iterativa incremental do RUP [17], um modelo de processo de desenvolvimento. Desse modelo foram selecionadas apenas as atividades de engenharia de software que, intuitivamente e baseado em experiência de projetos anteriores do autor, são necessárias à execução deste por apenas uma pessoa, a qual assume diversos papéis, tais como analista, desenvolvedor e testador.

O modelo *Waterfall* [18] (em cascata), utilizado como contra-exemplo, é um processo de desenvolvimento composto pelas fases de Requisitos, *Design*, Codificação e Testes, no qual cada uma dessas fases é executada sequencialmente, sempre nessa ordem e após a conclusão da anterior, considerando todo escopo do projeto.

Sabe-se que a abordagem desse modelo possui riscos inerentes e isso é evidenciado pelo artigo de David Parnas [19], cuja tradução de parte do texto é a seguinte:

“Muitos dos detalhes [dos sistemas] somente se tornam conhecidos por nós à medida que progredimos na sua implementação [do sistema]. Algumas coisas que aprendemos invalidam nosso *design* e nós precisamos retroceder.”

Sendo assim, a idéia de definir um processo para o desenvolvimento deste projeto vem no sentido de mitigar o risco desse tipo de retrocesso que gera mudanças e retrabalhos desnecessários. Desse modo, decidiu-se pela adoção da estratégia iterativa incremental do RUP porque, através dela, o projeto é dividido em subprojetos, que são executados, em iterações, um após o outro. Logo, em cada iteração são executadas todas as atividades de engenharia de software demandadas por um projeto as quais, no RUP, são organizadas em *workflows*.

Neste projeto foram selecionados os *workflows* básicos de Requisitos, Análise e *Design*, Implementação e Testes para compor cada iteração de modo que a construção de um subconjunto do sistema possa se dar com base nas lições aprendidas no desenvolvimento de um subconjunto anterior. Em cada *workflow* foi gerado um artefato, que pode ser um documento, um modelo, arquivos de código fonte (ou de configuração) ou um relatório.

A esses artefatos foram atribuídas finalidades para avançar em questões específicas do projeto. Essas finalidades são descritas abaixo e distribuídas em duas iterações.

A. Documento de Especificação de Requisitos

1ª Iteração

- Definir o raio ou alcance da monitoração quanto aos aspectos da aplicação que precisam ser monitorados;
- Definir o formato da mensagem dos eventos de monitoração em conformidade com a interface do Nagios e que contenha também a identificação da *thread* e do *host* para o caso de uma arquitetura em cluster ou algum outro tipo de redundância;
- Classificar os erros que serão monitorados por tipo de API ou recurso utilizado pelo Autorizador TEF (ex: sistema de arquivos local ou distribuído, banco de dados de conexão ou transacional, redes X.25 ou TCP/IP, etc.);
- Definir as informações específicas de cada evento de API como parâmetros e contexto de execução;
- Definir as métricas mais relevantes ao negócio do TEF no que concerne à monitoração do Autorizador;
- Definir o componente que será responsável pela geração dessas métricas;

2ª Iteração

- Especificar novas métricas para monitoração;
- Refinar as informações específicas dos eventos de monitoração que se deseja obter/notificar;

B. Modelo Estrutural e de Dados

1ª Iteração

- Projetar o componente de integração entre Autorizador TEF e o Nagios;

2ª iteração

- Projetar o mecanismo de detecção de mudança na configuração do Autorizador TEF;

C. Código Fonte

1ª Iteração

- Implementar o gerador de métricas selecionando uma de cada tipo, aquelas cujos cálculos possuem maior diferença entre si;

2ª Iteração

- Concluir implementação do gerador de métricas;
- Converter o componente de verificação passiva (`send_nsca.exe`) para DLL Windows de forma que possa ser integrado ao Autorizador por linkedição.

D. Arquivos de Configuração da Monitoração

1ª Iteração

- Configurar o Nagios para realizar as verificações ativas e tratar o resultado das passivas;
- Configurar a integração dos programas participantes do processo de monitoração;

2ª Iteração

- Organizar os resultados das verificações;
- Configurar o Nagios para realizar as notificações por email.

E. Plano de Testes

1ª Iteração

- Criar scripts para simulação de eventos de monitoração;
- Elaborar um roteiro de testes para validação do processo de geração, envio e processamento de eventos de monitoração (erros de API e métricas de TEF), abragendo as implementações e a configuração do Nagios;
- Executar a sequência de testes projetada e fazer as devidas correções;

2ª Iteração

- Complementar o plano com testes de novas métricas;
- Reexecutar os testes após refinamento dos requisitos e implementação;
- Reportar o resultado dos testes através de traces, gráficos, tabelas ou imagens capturadas da interface *web*.

Cada um destes artefatos foi produzido na execução do planejamento do projeto e são expostos a seguir com exceção do resultado dos testes que é avaliado no próximo capítulo. Os código fonte e dos arquivos de configuração são apresentados nos termos de sua estrutura e aspectos mais artificiosos.

4.2 Especificação dos Requisitos

Histórico de Revisão

Versão	Período	Descrição
1.0	1a Iteração	*Draft* com todas seções incluindo os requisitos funcionais: - Captura de Erros - Identificação de Erros de API de Infra-Estrutura - Identificação de recurso redundante - Cálculo de Métricas de Estabilidade - Visualização dos Eventos
1.1	2a Iteração	Refinamento dos requisitos: - Cálculo de Métricas de Estabilidade e de Desempenho - Visualização dos Eventos

4.2.2 Introdução

Propósito

A Análise de Requisitos divide-se na identificação de problemas que se deseja solucionar, na avaliação desses problemas e na sintetização dos que se deseja resolver através do sistema. O resultado desse processo é o que está especificado através dos requisitos funcionais, que definem o que o software faz para atender às necessidades do usuário, e os não funcionais, também chamados de requisitos de qualidade, que versam sobre aspectos como segurança, confiabilidade, usabilidade, etc. Nesta especificação os funcionais são classificados de acordo com a prioridade de implementação (alta, média ou baixa).

Além do detalhamento dos requisitos, neste artefato são apresentadas informações que contornam a idéia geral do Vigilante TEF (VTEF), o software a ser desenvolvido neste projeto. Dentre outras, inclui-se o escopo do projeto, as interface de interação do VTEF, o diagrama de implantação para contextualização do software no ambiente operacional.

Escopo do Projeto

O VTEF é um software para monitoração, de Autorizadores TEF. Suas funcionalidades incluem a monitoração indireta de *switches*, roteadores, *array* de discos e demais equipamentos utilizados pelo Autorizador TEF. A detecção de toda falha será realizada através do impacto desta infra-estrutura no Autorizador TEF, ou seja, através dos erros recebidos por este. Consequentemente são considerados como alvos de monitoração o Autorizador TEF e sua máquina hospedeira.

4.2.3 Descrição Geral

Características do Produto

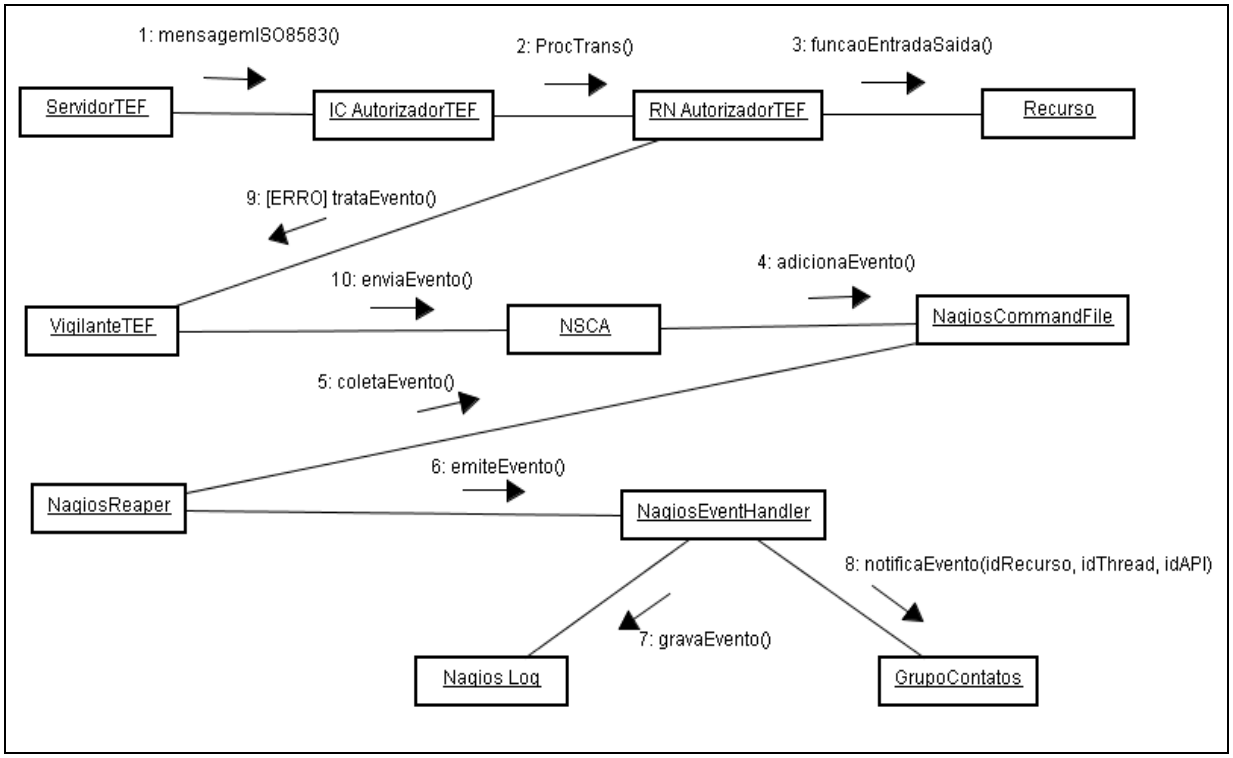


Figura 7. Diagrama UML de Colaboração da Verificação Passiva

A figura 7 representa um fluxo operacional do VigilanteTEF cujo objetivo, além de computar métricas com base no histórico de transações financeiras, é de intermediar o tratamento dos eventos enviados pelo Autorizador para o NSCA. Eventos estes que alimentam a base de dados do Nagios para que os administradores de infra-estrutra (redes, banco de dados, sistema operacional, etc.) sejam notificados de anormalidades impactantes no Autorizador.

O fluxo começa com uma mensagem ISO8583 enviada pelo ServidorTEF, em consequência de uma transação iniciada em um PDV. O Autorizador, ao recebê-la, cria

uma *thread* para seu processamento que, pelo porte e propósito da aplicação, acessa informações persistentes, local ou remotamente, executando uma função de Entrada ou Saída (E/S). Esta função pode retornar um erro que deve ser capturado pelo Autorizador e enviado para o servidor de monitoração através do VTEF e do NSCA.

O fluxo finaliza com a execução dos módulos do Nagios, responsáveis pela coleta do evento até sua notificação e gravação para verificação quase imediata ou posterior pela interface web.

Classificação dos Usuários

O projeto tem, como potenciais beneficiados, os usuários seguintes:

- Administradores de Redes
- Administradores de Bancos de Dados
- Administradores de Sistema
- *Staff* de Suporte do Servidor ou Autorizador TEF
- Gerentes de Infra-Estrutura

Ambiente Operacional

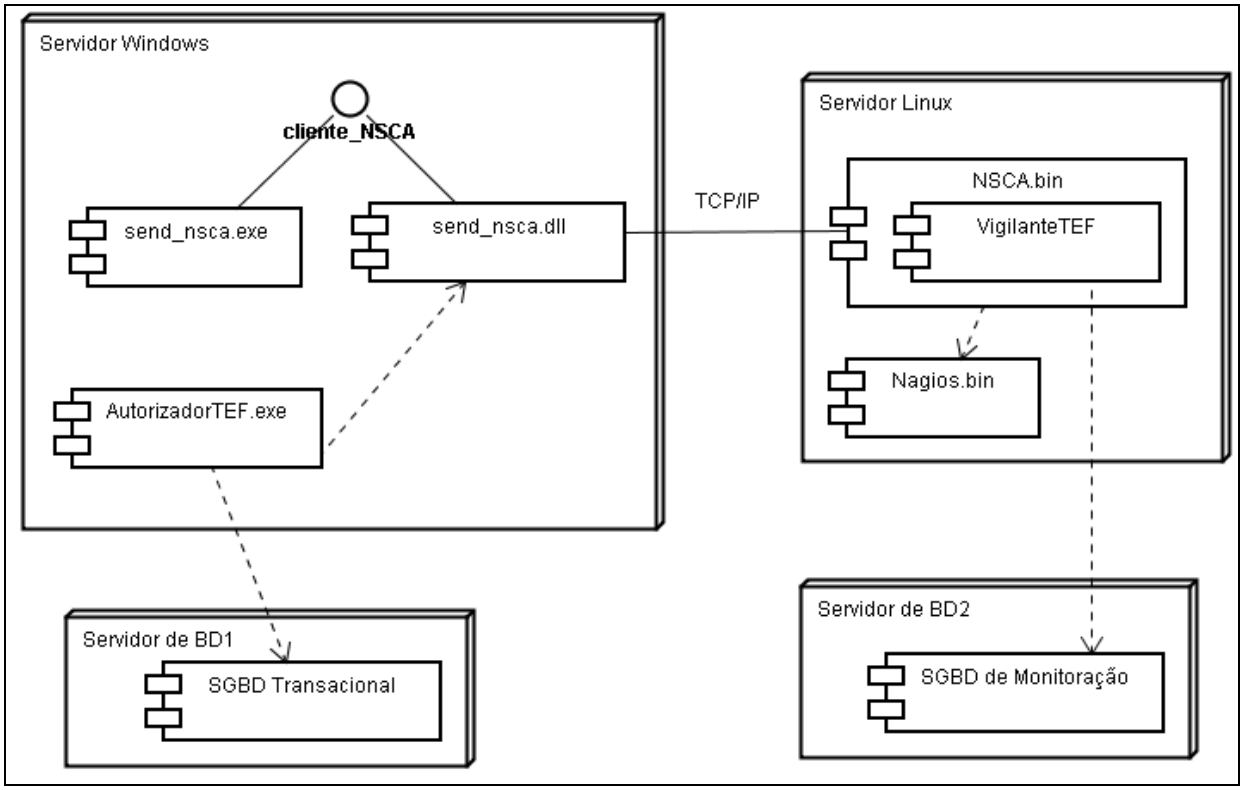


Figura 8. Diagrama UML de Implantação do VTEF

O VTEF é um módulo implementado no NSCA utilizado para filtrar e processar resultados de verificações passivas enviados ao *daemon* do NSCA, relativos ao domínio de negócio do TEF. O NSCA, e consequentemente o VTEF, executam na mesma máquina Linux do servidor Nagios. Uma biblioteca que implementa o mesmo protocolo do `send_nsca.exe` é carregada pelo AutorizadorTEF. O executável não é utilizado para se comunicar com o NSCA dado que a criação de um processo, a cada envio de evento, possui um custo de processamento mais alto do que simplesmente esperar pela chamada de envio.

Premissas e Pré-Requisitos

As premissas para viabilização operacional do VTEF, que não dependem do projeto, são:

a) Atendimento aos requisitos de sistemas recomendados para o ambiente do Nagios;

b) Funcionamento correto das *features* (características) declaradas pelo Nagios;

c) O fornecedor do Autorizador TEF é responsável pela replicação dos dados do SGBD Transacional para o SGBD de Monitoração realizando os procedimentos cabíveis de equivalência de modelos e transformação de dados;

4.2.4 Requisitos Não Funcionais de Interface

Interfaces com Usuário

O VTEF não possui interface com usuário, gráfica ou de linha de comando, cuja finalidade seja um meio de executar um caso de uso. O carregamento de seu módulo e sua execução são de total responsabilidade do NSCA.

Há, no entanto, um meio de parametrização de seu comportamento através da tabela de configuração detalhada no artefato de Modelo de Dados. O log de erros escrito nas saídas de console `stdout` e `stderr` é uma outra interface que pode ser utilizada pelo administrador de sistemas.

Interfaces de Comunicação

A comunicação do Autorizador TEF com o VTEF se dá através de conexão TCP/IP com o NSCA e por meio de mensagens claras ou criptografadas, dependendo do parâmetro `encryption_method` escolhido nos arquivos `nsca.cfg` e `send_nsca.cfg`.

O protocolo de aplicação utilizado é o mesmo do NSCA, composto simplesmente por apenas dois tipos de mensagens e com os propósitos específicos de:

a) Verificação de Serviço

```
<nome_host>\t<nome_servico>\t<codigo_resultado>\t<dados_evento>\n
```

b) Verificação de Host

```
<nome_host>\t<codigo_resultado>\t<dados_evento>\n
```

Onde os caracteres precedidos de '\' são de escape e os tokens entre '<' e '>' possuem a seguinte definição:

<nome_host>: Nome do host na configuração do Nagios

<nome_servico>: Nome do serviço na configuração do Nagios

<codigo_resultado>: caractere '0' (para OK), '1' (*WARNING*) ou '2' (*CRITICAL*).

<dados_evento>: Dados específicos de cada evento

Interfaces de Software

A interface do VTEF é utilizada no NSCA, alterando o código fonte original distribuído pelo Nagios. Outra interface é disponibilizada para o Autorizador no cliente do NSCA, o `send_nsca`, composta por funções exportadas na DLL contendo controle de exclusão mútua, com as quais é possível enviar o resultado de uma verificação.

Interfaces de Hardware

Não se aplica pois o acesso ao hardware ocorre apenas indiretamente pelo uso de API do sistema operacional para sistemas de arquivos, threads, mutex, redes e banco de dados.

4.2.5 Requisitos Funcionais

RF01 - Captura de Erros

Descrição

Primeiramente faz-se necessário explicar em qual sentido é utilizada a palavra erro. Para efeito prático, o conceito de erro adotado nesta especificação, é de seu código que representa uma falha no sistema, falha esta que pode ser de ordem de *design*, física ou de interação[20].

As falhas de *design* não dependem nem de uma falha de hardware, nem e codificação. Analisemos o exemplo de *timeout* de conexão com banco de dados, comum a quase toda lista de fluxos de exceção de aplicativos com memória persistente. A implementação de uma aplicação desse tipo e do SGBD podem estar corretas, mas mesmo assim, não havendo pelo menos uma conexão livre para utilização, a aplicação

receberá um erro indicando que a conexão não foi bem sucedida. Outro exemplo de erro por *timeout* seria quando um comando SQL é enviado mas a capacidade de processamento do SGBD está comprometida com outras requisições. Ou ainda, para o caso de qualquer aplicação conectada por TCP, enviando uma mensagem no momento em que a rede está bastante congestionada, a depender da quantidade de bytes escrita na pilha (camada) do protocolo, também irá ocasionar um erro de *timeout*. Nota-se como esse tipo de erro está mais relacionado a uma imprecisão de projeto do que propriamente de sua implementação, ou seja, mais ao subdimensionamento de recursos do que a uma estrutura física ou lógica.

Há também as falhas físicas decorrentes de um erro na construção ou produção do hardware utilizado, ou até mesmo devido a sua deterioração ou interferência devido a algum campo magnético.

Por fim há as falhas derivadas de interação maliciosa ou por engano de entrada. Instalação ou configuração de ambiente inadequados pertencem a essa classe de falhas. Exemplos desta podem ser um roteador X25 configurado para utilizar um *user data* diferente, que é um identificador da chamada X25, ou a não criação no banco de dados de um objeto, como uma tabela, necessário ao modelo de dados.

Definido o conceito de erro, é necessário por fim observar que nem todo erro é tratável pois existem aqueles que provocam uma finalização anormal (*abend*) de aplicações escritas em linguagem C.

Como primeiro passo antes do tratamento ou remoção de uma falha, que compreendem seu diagnóstico e adoção de medidas corretivas, o Autorizador TEF deve realizar a detecção de erros capturando todo erro tratável e enviando um evento correspondente para o servidor de monitoração através do VTEF.

Prioridade

Requisito Essencial – Prioridade Alta.

RF02 - Identificação de Erros de API de Infra-Estrutura

Descrição

A identificação inequívoca de uma situação de erro possui 3 pré-requisitos:

i) Identificação da biblioteca/API a que está associado o código que representa o erro;

ii) A precisão e clareza da descrição deste erro;

iii) A associação unívoca deste código a uma descrição, ou seja, a localização exata da descrição correspondente ao código.

Em algumas linguagens de programação, a exemplo do Java, a maioria dos comportamentos inexperados são chamados de exceções e identificados por seu nome. Por outro lado, o processo de identificação de erro do projeto atenta apenas para a maneira de representação de erros por código, no formato numérico, destinado principalmente à programação com linguagem C. Além disso quanto ao item ii), não foram reescritas ou traduzidas as descrições, para melhorar o significado.

A lista de códigos de erro de uma API de infra-estrutura (redes, banco de dados, sistema operacional, etc.) é geralmente definida por seu fabricante/autor, e diferentes autores podem utilizar faixas de código com valores que se sobrepõem, impossibilitando a descoberta do erro apenas através de seu código. Caso a API seja definida por um consórcio ou organização que estabeleça um padrão *de facto,* as chances disso ocorrer são bem menores mas infelizmente não é a realidade para tecnologias mais antigas e que não possuem foco em portabilidade, como a linguagem C.

Além disso, há ainda dois outros atributos de API que podem levar à divulgação de uma lista com códigos diferentes para a mesma situação de erro: o número e sistema operacional da versão. Isso porque por menor que seja a diferença de uma versão para outra, ela pode passar a retornar outros códigos para tratamento de erros novos ou legados mesmo devido a uma evolução mínima na implementação ou porque tenha sido portada para outra plataforma.

Dadas as condições acima, em consequência, para se obter a descrição de um erro é necessário conhecer o fabricante, a versão e o sistema operacional da API que gerou o código de erro. Mas não pára por aí, para concluir a análise, na prática, existem pelo menos duas maneiras de organizar uma lista de códigos. A Microsoft adota uma lista para seu conjunto de APIs, chamada de *System Error Codes*, que é dividida em faixas numéricas objetivando evitar que o mesmo código seja utilizado por APIs com finalidades distintas. Já no Linux um código pode ser utilizado por diversas APIs e o esclarecimento de seu significado é dado pelo contexto e pela documentação, e por conta disto, faz-se necessário notificar também o nome da função da API que retornou o código.

Em vista disso, o evento de erro enviado ao VTEF contém um mnemônico para informar o tipo de API, o código original do erro e o nome da função geradora (este por conta da organização dos códigos no Linux), enquanto que, a versão propriamente dita da API incluindo seu nome, fabricante, numeração e sistema operacional devem ser identificados por outro meio, dissociado do VTEF, através de uma ferramenta de Gerência de Correções/Mudanças como o Mantis, por exemplo. Esta decisão foi tomada por dois motivos:

i) Incluir estas informações no evento tornaria a notificação burocrática obrigando o programador a conhecer detalhes da implementação e da biblioteca para codificação da chamada, infringindo o princípio de encapsulamento.

ii) Idealmente, o VTEF, ao receceber os dados de um evento de erro, deveria ser capaz de identificar univocamente o erro, conhecer seu significado (descrição) e sugerir ações para sua correção. Contudo, tanto o mapeamento do significado quanto as sugestões serão consideradas como trabalhos futuros. Por hora, atribui-se à empresa que implementou o Autorizador TEF a responsabilidade sobre o conhecimento da versão utilizada para determinada API no sentido de viabilizar a identificação do problema através da documentação do autor da API, uma vez que são conhecidos o tipo da API, onde ocorreu o erro (função geradora) e seu código original.

Em resumo, os dados dos eventos de erro, que deve ser enviado pelo Autorizador TEF, assumem o seguinte formato (em ASCII):

```
ERR:XXX,<nome_funcao>,NNNNNN,0xHHHHH\<br\>
COD.FONTE:<nome_arquivo>,<num_linha>,<str_parametros>,<info_contexto>
```

Onde XXX são 3 caracteres correspondentes ao mnemônico do tipo de API, <nome_funcao> é o nome da função da API que gerou o erro, NNNNNN são 6 caracteres numéricos do código, HHHHH são 5 caracteres do código convertido em hexadecimal. O mnemônico pode assumir os seguintes valores:

ARQ: Sistema de Arquivo Padrão

THR: Thread do Sistema Operacional

SCO: Sincronização de Concorrência

TCP: Redes TCP

X25: Redes X.25

ABD : Acesso a Banco de Dados

Além disso, nota-se que o evento traz também informações para localização do erro no código fonte do Autorizador onde <nome_arquivo> é o arquivo que contém a chamada à função, <num_linha> é o número da linha da chamada neste arquivo, <str_parametros> é uma string informando quais parâmetros de entrada foram passados para a função e <info_contexto> deve apontar qual fluxo da aplicação foi executado já que a mesma função pode ser chamada em contextos distintos.

Prioridade

Requisito Essencial – Prioridade Alta

RF03 - Identificação de recurso redundante

Descrição

Devido à criticidade da finalidade do Autorizador TEF, a arquitetura desse sistema requer a adoção de estratégias de alta disponibilidade cuja principal prática é a utilização de recursos redundantes.

Se o recurso puder ser identificado na descrição do erro ou pelo conhecimento do Autorizador TEF, quando o próprio é responsável pelo mecanismo de contingência no qual um dos recursos redundantes é acionado, então essa informação deve ser adicionada no campo <info_contexto> do requisito acima.

Um Autorizador TEF coorporativo, que atende apenas ao movimento das lojas de uma empresa, possivelmente não se utiliza de um mecanismo de balanceamento ou de contingência para alta disponibilidade. Logo, em casos deste tipo, o identificador do recurso redundante nunca estará presente nos dados do evento.

Prioridade

Requisito Essencial – Prioridade Baixa

RF04 - Cálculo de Métricas de Estabilidade e de Desempenho

Descrição

Para evitar eficientemente as falhas em um sistema não basta saber dos erros que lhe afetam explicitamente. Complementarmente, é importante conduzir uma avaliação contínua do comportamento do sistema com respeito à combinação de eventos que possa levar a um estado errôneo ou à diminuição da qualidade de serviço. Isto é, atuar de forma preditiva como meio de consolidar a estabilidade, desempenho e dependabilidade do sistema [20].

Visando corroborar com esta linha de atuação, métricas no domínio de negócio de TEF e a utilidade destas na identificação de possíveis correlações são descritas abaixo (quando a transação de venda é citada, leia-se venda ou pagamento):

i. Índice diário elevado de cancelamento (estorno) por loja: Uma vez que essa transação ocorre com baixa frequência, uma loja que se destaque neste aspecto pode demonstrar um indício de fraude. Por parte dessas tentativas pode surgir um certo volume de transações negadas. Logo, esta métrica não deve incluir apenas os cancelamentos efetivos;

ii. Índice diário elevado de cancelamento (efetivo ou não): Apresenta a soma dos percentuais do índice de cancelamento por loja;

iii. Índice diário elevado de sondas: a sonda é uma transação de contingência quando uma transação de três pernas não é confirmada ou desfeita, ou seja, permanece com status pendente. Isto pode ser causado pelo mau funcionamento da infra-estrutura do Servidor TEF ou do Autorizador TEF, ou até mesmo dessas aplicações, que levam a uma perda de uma das pernas da transação financeira. Como a sonda é enviada para um endereço que não tem a obrigatoriedade de ser o mesmo da origem da transação pendente, esta métrica não é agrupada por endereço de destino da sonda.

iv. Índice diário elevado de sondas sem resposta, por endereço de destino: Por questões contratuais, o varejista ou a empresa do Servidor TEF tem o dever de resolver as pendências (confirmar ou desfazer as transações pendentes). A sonda é a responsável por consultar o Servidor TEF sobre o *status* da transação que continua pendente no Autorizador TEF. Se não há resposta, é necessário notificar os administradores da rede local para verificar se os pacotes estão saindo da rede. Caso positivo, deve-se abrir um chamado tanto para o provedor de rede quanto para o responsável pelo gerenciamento do Servidor TEF;

v. Índice diário elevado de desfazimento por loja: Desfazimento é uma transação referente a uma compra (ou pagamento ou cancelamento) para informar que esta compra não será confirmada, ou seja, que houve desistência da efetivação da operação. Além da desistência do cliente no PDV, pode ocorrer quando a 1ª ou 2ª perna não foi processada com sucesso no Servidor TEF. É importante também que esta métrica inclua todas tentativas de desfazimentos, não apenas os efetivos, uma vez que esse tipo de transação pode ser gerada automaticamente pelo Servidor TEF, havendo a possibilidade de falha na implementação deste mecanismo;

vi. Índice diário elevado de desfazimento (efetivo ou não): Apresenta a soma dos percentuais do índice de desfazimento por loja;

vii. Índice diário de timeout de envio por cada tipo de transação, exceto a sonda (venda, consulta, cancelamento e desfazimento): A ausência do envio da 2ª perna é um problema mais relacionado ao Autorizador TEF do que quanto à infra-estrutura do ambiente do Servidor de TEF ou da comunicação com este. Pode representar uma falha no tratamento da mensagem de entrada (1ª perna), no código fonte da montagem da 2ª perna ou na infra-estrutura do Autorizador TEF (limite de threads excedido, sistema de arquivo sem espaço, problema no acesso ao banco de dados, etc.);

viii. Índice diário elevado de negação por cada tipo de transação (exceto a sonda), por loja: Um alto percentual de negações, para um total de transações relevante, pode indicar uma não conformidade à especificação na

implementação do Servidor TEF, ou do Autorizador TEF, implicando a continuidade da ocorrência das negações caso nenhuma medida seja tomada. A negação causada por má configuração de um dos sistemas também é uma possibilidade. Um outro motivo que pode causar a negação ocorre quando o conteúdo de um campo do protocolo não respeita uma regra de negócio. Por exemplo, o número de um boleto de pagamento deveria vir formatado com a linha digitável enquanto que em vez disso é enviado o código de barras. Este e outros erros podem ocorrer por alguma falha de hardware ou de software no PDV ou seus periféricos, daí a importância de calcular esta métrica por cada loja;

ix. Índice diário elevado de negação por cada tipo de transação: Apresenta a soma dos percentuais do índice de negação por loja;

x. Tempo médio de resposta (TMR) elevado da transação de venda, à cada minuto, ou número máximo de transações: A venda é a transação mais importante do TEF e precisa ser rápida o suficiente para não incomodar os clientes no caixa. Um valor alto deste tempo indica uma possível demanda por redimensionamento ou otimização do sistema através de ajustes de desempenho (tunning) do SGBD, melhoria da implementação do framework do Autorizador TEF (pool de threads por exemplo), ou dependendo da disponibilidade financeira e tecnológica, pela melhoria dos hardwares do servidor de aplicação, de banco de dados ou dos equipamentos de rede. Outro aspecto desta métrica é sua periodicidade. Ela é calculada à cada minuto, porém se ocorrerem mais transações do que um número máximo estipulado, 600 transações por exemplo, neste minuto, a média é calculada em cima destas 600 transações. Isto é feito para evitar o desvio na métrica pela quantidade excessiva de transações já que isto provoca um aumento no TMR;

xi. Tempo médio de resposta elevado incluindo todo tipo de transação exceto a sonda, à cada minuto (TMR-Geral): Apesar da maior relevância da transação de venda, é necessário monitrar o desempenho das demais transações, apesar de menos frequentes (questione-se pela sua própria experiência quantos estorno você realizou com seu cartão). Além disso, essa métrica traz o total de transações que ocorreram no período permitindo comparar com a quantidade de vendas informada no índice anterior (TMR-VendaPgto);

xii. Quantidade elevada de transações de venda por segundo (TPPS-VendaPgto): O pico de atividade representa uma das virtuais causas do baixo desempenho no processamento de transações. O administrador tem a opção de zerar o TPPS notificado através de um comando de shell script;

xiii. Quantidade elevada de transações por segundo (TPPS-Geral): Tem próposito semelhante da métrica anterior porém capta o excesso de transações indepedentemente de seu tipo (as sondas não entram na soma).

As transações utilizadas nessas métricas devem estar salvas em um SGBD de Monitoração, contendo a replicação das transações salvas pelo Autorizador TEF no SGBD Operacional, evitando a concorrência entre as *threads* do VTEF com as decorrentes do movimento de loja. O VTEF, à cada intervalo de poucos minutos (configurável), acessa o banco de dados de Monitoração e calcula essas métricas (índices, totalizadores e médias). Apenas devido ao curto espaço de tempo da métrica TPPS, o intervalo de sua verificação é configurado em um parâmetro próprio.

Além destas métricas seria possível calcular outras, tais como o índice diário elevado de cancelamento efetivo, porém a ocorrência desta transação por acidente ou erro caracteriza um defeito grotesco já que para efetivar um cancelamento são necessárias 6 mensagens (3 pernas da venda mais 3 pernas do cancelamento). Essa métrica poderia sugerir algum comportamento do consumidor (frequente devolução de produtos) ou do operador do caixa (erro de digitação do valor da transação, por exemplo). As medidas com este propósito estão fora do escopo do trabalho.

Definidas as métricas, são especificados com detalhe os cálculos de cada uma:

i. Índice diário elevado de cancelamento (estorno) por loja = contagem das transações de estorno da loja no dia / contagem das transações de venda da loja no dia;

ii. Índice diário elevado de cancelamento = contagem das transações de estorno da empresa no dia / contagem das transações de venda da empresa no dia;

iii. Índice diário elevado de sondas por endereço de destino = contagem das sondas no dia para o destino / contagem das transações realizadas pelo destino da sonda, no dia, que podem ficar pendentes (venda ou cancelamento);

iv. Índice diário elevado de sondas sem resposta = contagem das sondas com status “em andamento” no dia para o destino / contagem das sondas no dia para o destino;

v. Índice diário elevado de desfazimento por loja = contagem das transações de desfazimento da loja no dia / contagem das transações de 3 pernas (venda e cancelamento) da loja, no dia, sem incluir as que ainda estão em andamento;

vi. Índice diário elevado de desfazimento (efetivo ou não) = contagem das transações de desfazimento da empresa no dia / contagem das transações de 3 pernas da empresa, no dia, sem incluir as que ainda estão em andamento;

vii. Índice diário de *timeout* de envio por cada tipo de transação, exceto a sonda = contagem das transações, do tipo em questão, com status “em andamento” ocorridas no dia / contagem do mesmo tipo de transação com status diferente de “em andamento”, no dia;

viii. Índice diário elevado de negação por cada tipo de transação, exceto a sonda, por código de resposta = contagem agrupada por código de resposta, das transações negadas (código de resposta diferente de “00”), do tipo em questão, com status “concluída” ocorridas no dia/ contagem destas transações com status diferente de “em andamento”, no dia;

ix. Índice diário elevado de negação por cada tipo de transação = contagem das transações negadas da empresa, ocorridas no dia, do tipo em questão e com status “concluída”, / contagem destas transações com status diferente de “em andamento”, no dia;

x. TMR elevado de venda, à cada minuto ou número máximo de transações = soma do tempo de resposta das transações de venda, que ocorreram no período ou das primeiras até o número máximo (parâmetro), e que não estão com status “em andamento” / quantidade destas transações;

xi. TMR elevado das transações (exceto a sonda), à cada minuto ou número máximo de transações = soma do tempo de resposta das transações, que não seja uma sonda, que ocorreram no período, ou das primeiras até o número máximo, e que não estão com status “em andamento” / quantidade destas transações;

xii. TPPS de venda = contagem das transações de venda que ocorreram no minuto, agrupadas por segundo;

xiii. TPPS geral = contagem das transações que ocorreram no minuto, agrupadas por segundo;

Após o cálculo dessas métricas o VTEF opta pelo envio de eventos associados a um resultado que pode ser OK, de alerta (*WARNING*) ou crítico (*CRITICAL*), de acordo com os *thresholds* definidos no modelo de dados. Por exemplo: se o índice de desfazimento for no momento até 15%, é enviada uma mensagem com resultado OK, entre 15 e 30% resulta em alerta e acima de 30% é uma situação crítica.

Prioridade

Requisito Essencial – Prioridade Média

RF05 - Visualização das Métricas

Descrição

As métricas de cancelamento e desfazimento efetivos diários de cada loja devem ser exibidas em seus próprios objetos de serviço no Nagios, para que estejam sempre visíveis os eventos ocorridos em cada loja. O mesmo se aplica as métricas de negação de transações específicas. Assim o formato da diretiva `service_description`, que identifica um objeto de serviço, deve seguir o padrão

TEF.<ESTAB> Canc.Loja NNNN/dia onde <ESTAB> é a sigla da empresa e NNNN é o código de uma de suas lojas. Por exemplo, TEF.LBSA Canc.Loja0120/dia | 2% (4/200) exibe um índice de 2% das transações (4 de um total de 200) sendo canceladas na loja 0120 da LBSA, Lojas Brasileiras SA.

As demais estatísticas podem ser exibidas agregando-as por empresa e declarando apenas um objeto de serviço para cada uma delas. Por exemplo: TEF.LBSA Perc.Sonda/dia | 4% (6/150) exibe o percentual de sondas da LBSA.

Além disso, para melhorar a visualização dos eventos, o Nagios permite o agrupamento de serviços, separando-os em links e telas distintas. Logo, as métricas são agrupadas da seguinte maneira (onde <TipoTrans> aplica-se aos tipos de transação de Consulta, Cancel., Desfaz. e Venda).

Grupo <ESTAB>-MetricasPorLoja

Este grupo agregada as métricas cujos resultados são exibidos separadamente para cada loja. São incluídas também as métricas que totalizam os resultados das lojas.

- TEF.<ESTAB> Perc.Cancel.Lj-NNNN/dia
- TEF.<ESTAB> Perc.Desfaz.Lj-NNNN/dia
- TEF.<ESTAB> Perc.Cancel/dia
- TEF.<ESTAB> Perc.Desfaz /dia
- TEF.<ESTAB> Perc.Neg.<TipoTrans>.Lj-NNNN/dia
- TEF.<ESTAB> Perc.Neg.<TipoTrans>/dia
- TEF.<ESTAB> Perc.Neg.<TipoTrans>.Lj-NNNN/dia
- TEF.<ESTAB> Perc.Neg.<TipoTrans>/dia

Grupo <ESTAB>-MetricasPorEndereco

- TEF.<ESTAB> Perc.SondaSemResposta/dia
- TEF.<ESTAB> Perc.Sonda/dia *(porque agrega informação à anterior)*
- TEF.<ESTAB> Perc.TO.<TipoTrans>.Lj-NNNN/dia
- TEF.<ESTAB> Perc.TO.<TipoTrans>/dia

Grupo <ESTAB>-MetricasDesempenho

- TEF.<ESTAB> TMR-Venda/min/maxTrs
- TEF.<ESTAB> TMR-Geral/min/maxTrs
- TEF.<ESTAB> TPPS-Venda/min
- TEF.<ESTAB> TPPS-Geral/min

Prioridade

Requisito Essencial – Prioridade Média

4.2.6 Requisitos Não Funcionais Adicionais

Desempenho

O desempenho do VTEF não é tão crítico, tendo em vista que não executa na máquina do Autorizador TEF, mas deve ser rápido o suficiente para processar as métricas em sua devida periodicidade considerando um fluxo máximo de 100 TPPS durante 1 minuto (6000 transações) e uma quantidade máxima de 1 milhão de transações na base de dados, já incluindo as 6000. Essa meta é estabelecida para o requisito de sistema mínimo de uma máquina com processador de 2.0Ghz e 1GB de memória RAM, ou seja, com a mesma configuração da destinada aos testes da versão do VTEF resultante deste projeto.

Confiabilidade

O Nagios permite a implantação de uma arquitetura em cluster para dar maior disponibilidade ao serviço de monitoração. A princípio esta arquitetura não será utilizada, para efeitos práticos de testabilidade, porém, elencada com trabalho futuro. Esta arquitetura torna-se relevante, principalmente, a partir da percepção do valor que os sistemas de monitoração têm na continuidade do operacional do Varejo.

Segurança

Em sua primeira versão não será utilizada criptografia já que não serão trafegados dados sensíveis dos clientes nas mensagens de evento.

Interoperabilidade

A clareza e simplicidade da interface da DLL send_nsca proporcionam virtualmente a integração com qualquer Autorizador TEF da plataforma Windows. Para atender a sistemas de outras plataformas, é necessário converter o programa send_nsca para seu respectivo formato de biblioteca dinâmica (`.so` no Linux). Já a integração das bases de dados deve ser realizada através da replicação e eventual transformação dos dados do SGBD Transacional do Autorizador TEF para o SGBD de Monitoração definido pelo VTEF, mais especificamente no layout da tabela log_tef, definida na seção 4.3.2 - Modelo de Dados.

4.3 Modelagem Estrutural e de Dados

O propósito deste artefato é ajudar na descrição e no entendimento do sistema identificando suas ações e dados que são manipulados.

4.3.1 Modelo Estrutural

Foi elaborado um modelo orientado a objetos com UML, em virtude de uma maior familiaridade com este paradigma na fase de modelagem. Na fase de implementação, cada classe foi mapeada, no modelo estrutural do programa procedural, para um módulo, no sentido de um conjunto de funções correlatas, mais especificamente relacionadas com algum tipo de dado ou processamento em comum.

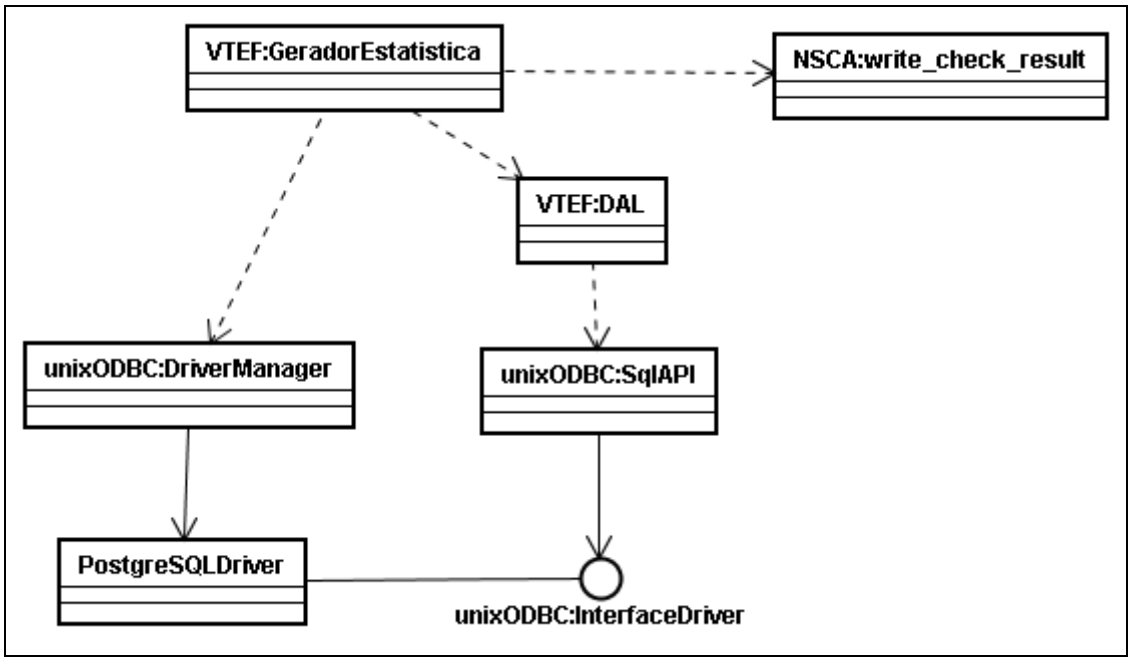


Figura 9. Diagrama UML de Classes (Módulos) do VTEF

Além da classe, que corresponde a um módulo neste modelo, foram utilizados três outros elementos do UML– dependência, associação, interface e realização – aos quais foram atribuídos os seguintes significados:

Dependência

A seta tracejada entre classes indica que um módulo chama uma ou mais funções de outro módulo. Neste diagrama, o Gerador de Estatísticas do VTEF realiza chamadas à camada de acesso a dados (DAL, *Data Access Layer*), que por sua vez

utiliza a API de SQL implementada na biblioteca dinâmica unixODBC. Os outros dois casos são a utilização de seu gerenciador de drivers para instânciar o driver do PostgreSQL e a chamada ao módulo do NSCA para o envio do resultado da verificação.

Associação

No sentido em que um módulo é responsável pelo instanciamento, disponibilização, ou acesso a outro, que é o caso do `DriverManager` que carrega o driver específico do PostgreSQL. Um outro sentido para associação é quando um módulo abstrai a chamada a outro através de uma interface padronizada, que é um dos principais objetivos do ODBC em qualquer aplicação, para que não sejam utilizadas funções específicas das implementações dos fabricantes que podem variar de um SGBD para outro. No caso deste modelo, tanto o controle da conexão como a execução dos comandos são iniciados ou disparados pela `SqlAPI` da biblioteca unixODBC acessando uma interface genérica padronizada do ODBC.

Interface e Realização

Simbolizados por um círculo e uma reta, representam a necessidade de implementação, ou de reuso de componentes, que abstraia os detalhes da variação de implementação, ou seja, que um módulo possa utilizar uniformemente um outro de modo independente da plataforma ou sistema. O *driver* PostgreSQL atua na realização da interface de *Driver* ODBC que padroniza as assinaturas (protótipos) das funções dos drivers específicos para um SGBD.

4.3.2 Modelo de Dados

Ao fim da 2ª iteração, o modelo é composto por apenas duas tabelas:

i. **log_tef**: contém os registros das transações ISO8583.

ii. **config_vtef**: contém um registro das configurações para cada estabelecimento.

Considerando que as duas tabelas não possuem relacionamento não foi necessário utilizar o Modelo de Entidade e Relacionamentos (MER) para descrever os dados do sistema.

A estrutura de cada tabela foi definida usando a modelagem física, aplicando os termos do SGBD adotado para o desenvolvimento, o PostgreSQL 8.3. Todas as colunas são de preenchimento obrigatório (*NOT NULL*) e as chaves primárias são indicadas com a sigla PK.

i. log_tef

Nome da Coluna	*Tipo de Dado*	*Comentário*
nsu_aut (PK)	numeric(9,0)	NSU da transação gerado pelo Autorizador.
nu_iso_trans	numeric(4,0)	Identificador do Tipo de Mensagem (MTI) da transação ISO8583 (ex: 0200,0400,0420,etc.).
nu_pcode	numeric (6,0)	Código de Processamento definido para o tipo da serviço (ex: 330657 para recarga de celular).
cd_status_trans	char (1)	Código de status de progresso da transação (A: andamento, P: pendente, C: concluída).
cd_tipo_trans	char (1)	Tipo da transação mapeado pelo autorizador pela combinação do nu_iso_trans com o nu_pcode (V: Venda/Pagamento, C: Consulta, D: Desfazimento, E: Estorno, S:Sonda).
dh_trans_aut	timestamp	Data hora da transação no Autorizador.
nu_tempo_resposta	numeric(6, 3)	Segundos e milésimos de segundo do tempo de resposta da transação, ou seja, a diferença da hora entre o envio da 2ª e o recebimento da 1ª perna.
cd_resposta	char (2)	Código de resposta da transação (00 indica sucesso, os demais são códigos de erro).
cd_empresa	char (5)	Código que identifica, no cadastro do Autorizador, o estabelecimento (empresa) de onde partiu a transação.
cd_loja	char (4)	Código que identifica, no cadastro do Autorizador, a loja de onde partiu a transação.
nu_endereco_ip	char(15)	Endereço IP do Servidor TEF (destino da sonda).

ii. config_vtef

Nome da Coluna	*Tipo de Dado*	*Comentário*
cd_empresa (PK)	char(5)	Código que identifica o estabelecimento no cadastro do Autorizador.
id_empresa	char (10)	Identificador da empresa. Pode ser uma sigla, nome ou código.
limite_warn_<metrica>	numeric(6,3)	Valor/*Threshold* da métrica que dispara evento de verificação passiva com resultado de alerta (WARNING).
limite_crit_<metrica>	numeric(6,3)	Valor/*Threshold* da métrica que dispara evento de verificação passiva com resultado crítico (CRITICAL).
nu_max_trs_tmr	integer	Número máximo de transações para evitar calcular o TMR com distorção.

Esta tabela possui ao todo 38 colunas. Destas, 36 são referentes ao threshold, de alerta ou criticidade, dos eventos enviados para cada tipo de métrica computada pelo VTEF (36/2 métricas no total).

4.4 Código Fonte

Objetivando evitar o *overhead* da criação de processo o executável send_nsca.exe foi convertido para DLL Windows. Com isso, foi substituída a função main pela `DllMain`, chamada apenas no carregamento e liberação da DLL, e exportada uma função que inicia o processamento do envio de eventos ao NSCA. Neste processo, o parâmetro `ul_reason_for_call`, da `DllMain`, que pode assumir os valores `DLL_PROCESS_ATTACH`, `DLL_PROCESS_DETACH`, `DLL_THREAD_ATTACH`, `DLL_THREAD_DETACH`, não foi tratado uma vez que esta DLL não demanda nenhum pré-processamento no carregamento da DLL em memória ou na sua liberação, sua ação é concentrada inteiramente na função exportada. Na assinatura desta função, cada *flag*/argumento de execução do `send_nsca.exe` foi mapeado como um parâmetro, assumindo a seguinte forma:

```
/* hostaddr: Endereço IP do NSCA
 * port: Porta TCP do NSCA
 * to_sec: Segundos antes de ocorrer timeout
 * config_file: Caminho do arquivo de configuração
 */
__declspec(dllexport)
void envia_nsca(char *hostaddr, int port,
                int to_sec, char *config_file);
```

O VTEF foi implementado como um módulo do NSCA (servidor) e dividido em dois submódulos: um para funções relacionadas ao domínio de negócio do TEF, o cálculo das métricas, e outro unicamente para ler as transações do banco de dados. Semelhante a uma arquitetura de três camadas com a diferença de que a camada de apresentação pertence a outro software, ao Nagios que exibe os eventos na sua interface *web*.

O módulo de geração das métricas foi estruturado de maneira que as funções de cálculo sejam executadas respeitando três aspectos de requisitos e abordando outros dois de programação. Cada um destes aspectos justifica um ou mais argumentos de suas assinaturas, conforme comentários a seguir:

i. Executada para um determinado estabelecimento (empresa) em consonância com o modo de agrupamento (aspecto de Requisito, argumento: `pc_id_estab`);

ii. O valor da métrica é considerado crítico ou de alerta a depender dos thresholds (Requisito, `limite_crit` e `limite_warn`);

iii. Cada métrica é exibida individualmente na interface web do Nagios, cada uma no seu próprio objeto de serviço (Requisito, `pc_str_servico`);

iv. As funções foram reusadas para o cálculo de métricas semelhantes precisando apenas informar como selecionar as transações (Implementação, `pc_strsql`);

v. As funções utilizam uma mesma conexão pré-estabelecida com o banco de dados (Implementação, `hDbc`);

Assinaturas das funções de cálculo:

```
//Calcula o TPPS de venda ou geral
void VTEF_calc_TPPS(char *pc_strsql,
     double limite_crit, double limite_warn,
     char * pc_id_estab, char * pc_str_servico, SQLHDBC hDbc);

// Calcula o TMR de venda ou geral selecionando no máximo um certo
// num. de transações que tenham ocorrido até um determinado momento
// (fim do intervalo de 1 minuto ou menos)
void VTEF_calc_TMR(double limite_crit, double limite_warn,
     char * pc_id_estab, char * pc_str_servico, SQLHDBC hDbc,
     time_t * pi_fim_intervalo, int max_trs);

// Executa o cálculo cada TPPS e TMR selecionando as transações que
// ocorreram a partir de um certo instante
void VTEF_calc_desempenho(time_t * pi_seg_calc,
     config_vtef_t * ps_config);

// Calcula as métricas de sonda, que são agregadas por destino (IP)
void VTEF_calc_metrica_ip(double limite_crit, double limite_warn,
     char * pc_id_estab, char * pc_str_servico, SQLHDBC hDbc);

// Calcula as métricas agregadas por loja
void VTEF_calc_metrica_lj(double limite_crit, double limite_warn,
     char * pc_id_estab, char * pc_str_servico, SQLHDBC hDbc);
```

Observa-se que as funções de cálculo não retornam nenhum dado do evento de monitoração, nem mesmo o valor da métrica. Isto porque elas próprias enviam esse valor no evento cujo procedimento é encapsulado em outra função do próprio NSCA.

Essas funções, e as demais do VTEF, são executadas por apenas uma *thread*. Ela é criada na inicialização no NSCA para execução do VTEF e é responsável por carregar inicialmente as configurações de *threshold* e depois gerar periodicamente todas métricas. Foi decidido o uso de apenas uma *thread* já que as métricas de TMR e TPPS possuem um requisito temporal mais exigente, no máximo é computada de minuto a minuto, enquanto que para as métricas diárias o intervalo mínimo é de 5 minutos. Logo, caso futuramente se deseje diminuir a periodicidade destas métricas diárias, ou caso o processamento demore mais por sobrecarga significativa na máquina do SGBD, o VTEF não dependerá do algoritmo de escalonamento do sistema operacional para garantir a regularidade no cumprimento

do período máximo dessas métricas. Isso porque ele verifica se deve computar novamente a TMR após o cálculo de cada métrica diária.

Já as funções de acesso aos dados do banco apenas recebem um comando (*statement*) de consulta SQL, o executa e lê os dados dos registros retornados colocando-os em uma estrutura de um dos tipos de dados definido no VTEF, exibidos abaixo. Apenas no caso da TMR ou TPPS, são atribuídos mesmo a variáveis de tipos primitivos.

```
typedef struct {
     char cd_loja[TAM_CD_LOJA+1];
     int  qtd_trs_espec;
   int  qtd_trs_total;
} trs_loja_t;

typedef struct {
     char nu_endereco_ip[MAX_TAM_IP+1];
     int  qtd_trs_espec;
     int  qtd_trs_total;
} trs_ip_t;
```

Quanto ao reuso dessas funções de acesso aos dados, vale observar que para o cálculo das métricas de estabelecimento (ex: TEF.<ESTAB> Perc.Cancel/dia), que traz o somatório dos resultados das lojas, também foi utilizada a estrutura `trs_loja_t` com a única diferença que no comando SQL o resultado foi identificado como pertencente à loja “0000”.

4.5 Arquivos de Configuração da Monitoração

A configuração do Nagios para o tratamento dos resultados das verificações passivas, no projeto VTEF, demanda a declaração de um objeto *host* para representar o Autorizador TEF e, de acordo com o requisito de visualização, a declaração de um objeto *service* para cada tipo de evento (de erro de API, de erro de regra de negócio ou de métrica de estabilidade).

Para evitar a redundância de atributos na declaração de objetos *service* semelhantes foi definido um *template* chamado de `remote-service-nsca` cujos atributos são herdados e descritos a seguir:

- `flap_detection_enabled`: Valor 0 para desabilitar detecção de flapping. O *flapping* acontece quando há alternância de eventos OK, de alerta ou críticos. Isso pode ocorrer quando há instabilidade em um software ou ambiente do TEF. Como na detecção de flapping, o evento não é notificado aos contatos, o atributo foi desabilitado.

- `active_checks_enabled`: Valor 0 para indicar que os eventos deste objeto *service* são apenas de origem passiva.
- `passive_checks_enabled`: Valor 1 para habilitar o tratamento dos eventos passivos.
- `check_freshness`: Valor 1 para verificação de reportagem ociosa (*stale*), ou seja, para sinalizar caso o VTEF não esteja enviando evento algum durante intervalo máximo definido para a métrica (à cada dia ou minuto).

Além disso foi necessário utilizar uma característica específica do Nagios para que a notificação aos contatos ocorra sempre que um evento crítico ou de alerta seja recebido. Isso é feito atribuindo valor zero ao parâmetro `passive_host_checks_are_soft` definindo portanto todos eventos passivos como do tipo *HARD.* Por padrão, todos contatos definidos com o template `generic-contact` do Nagios já recebem notificações por email. Para funcionar é preciso configurar uma conta para o utilitário de SMTP `mail` ou `mailx` do Linux.

Por fim quanto à configuração do Nagios, foi parametrizado um intervalo de 50 segundos para atualização automática da tela da interface web visto que a menor periodicidade dos eventos é de 1 minuto (do TMR).

Quanto à configuração dos demais programas participantes do processo de monitoração, a única mudança foi no parâmetro `encryption_method`. Foi atribuído valor 0 para desabilitar a criptografia no tráfego entre o send_nsca e o NSCA, conforme ressaltado pelo requisito de segurança.

4.6 Plano de Testes

Este plano é organizado em casos de teste que abrangem principalmente a validação funcional das métricas e do desempenho das TMR e TPPS. Para isso foi implementado um programa (`cargavtef`) para inserir toda variedade de transações no SGBD de monitoração. O roteiro dos casos de teste é apresentado após a examinação a seguir de alguns aspectos relevantes ao método de validação.

Quanto à validação da aplicação dos *thresholds* (limites) corretos no envio do resultado das métricas, o código que seleciona o limite crítico ou de alerta, ou nenhum dos dois, é bastante simples e pode ser validado através de revisão de código. Quanto à leitura da tabela config_vtef, poderia ser gerada uma *string* pelo VTEF formada pela concatenação dos valores dos limites e a mesma *string* através de uma *query* (consulta) SQL adaptada e executada externamente. Como isso não garante nem a ordem correta

de cada limite na string da aplicação, nem na consulta SQL externa, também foi realizada apenas a revisão do código fonte do VTEF que faz a leitura destes limites.

A validação dos resultados das métricas, exibidos no Nagios, demanda a validação de dois tipos de operações:

- Das *queries* que precisam recuperar do banco apenas as devidas transações de cada métrica;
- Do cálculo da métrica utilizando o resultado das *queries*.

A estratégia adotada para essa validação é composta pelos passos:

i. Implementar, no `cargavtef`, o cálculo das métricas, baseado exclusivamente nas fórmulas do requisito funcional.
ii. Imprimir, em arquivo texto, o resultado de cada métrica computada pelo `cargavtef`, no formato em que é exibido pelo Nagios, contendo o percentual e os valores do numerador e denominador.
iii. Modificar também o VTEF para escrever no seu arquivo de logs o resultado dos cálculos enviado para o Nagios, e no seu caso, o numerador e o denominador representam o total de transações selecionadas pelas *queries*.

Como os valores no `cargavtef` são obtidos diretamente por sua interface de entrada, sem precisar consultar as transações, logo, a comparação da saída dos dois arquivos possibilita a validação as duas operações citadas acima. Consequentemente, a validação das métricas pode ser executada de forma completamente automática por um *shell script*, utilizando utilitários do Linux, filtrando o resultado das métricas nos dois arquivos (utilitário `grep`), ordenando o resultado de cada um (`sort`) e comparando, com o `diff`, o resultado das métricas ordenadas computadas pelo VTEF e o `cargavtef`.

É necessário discutir também a questão da amostragem de transações. Essa validação não deve ser realizada individualmente para cada métrica, havendo no banco de dados apenas as transações que supostamente e unicamente são selecionadas pelas queries da respectiva métrica. Em vez disso, é importante que se faça uma carga completa com toda variedade possível de transação para verificar se não há inclusão de indevida de transações na seleção dos registros pelas *queries*.

Some-se a isso, o fato de que as variáveis de um experimento com uma métrica possuem, em alguns casos, dependência com as variáveis de outra métrica. Por exemplo: a métrica de percentual de desfazimento calcula a frequência deste tipo de transação com relação as de venda e estorno, ao passo que o cálculo das métricas percentuais de cancelamento e de negação de vendas também precisa que estejam no

banco estes tipos de transações. Em outras palavras, não é viável realizar a validação através de um programa interativo que iterativamente solicita os dados de entrada para calcular uma determinada métrica e apresenta seu resultado por que a quantidade final de transações de um certo tipo será diferente daquele digitado para uma certa métrica, fazendo com que o resultado não seja o mesmo do exibido no Nagios.

CT01 –Métricas de Estabilidade

Descrição

Valida a computação e envio dos resultados de todas as métricas exceto as de TMR e TPPS.

Pre-Condição

- Estar logado na interface web do Nagios;
- Haver na tabela `log_tef` apenas transações de dias anteriores (inseridas com `carga_vtef` passando como parâmetro `-o` para utilizar a data de ontem);

Execução

Passos	**Resultados Esperados**
1) Executar o `nsca` do VTEF passando seus parâmetros e esperar 1 min.	O Nagios exibe todas as métricas zeradas;
2) Executar o `carga_vtef` com toda variedade possível de transações (estados diferentes) e esperar 1 min.	O Nagios exibe os resultados e o VTEF os escreve no arquivo `vtef.log`;
3) Executar o *script* `diff_res.sh` (apêndice A);	Arquivo `diff.out` está vazio (nenhuma diferença entre `vtef.grep` e `carga.grep`);

CT02 –Métricas TPPS

Descrição

Valida a computação e envio dos resultados das métricas TPPS-Geral e TPPS-Venda até o fluxo máximo de transações definido no requisito de desempenho.

Pre-Condição

- Estar logado na interface web do Nagios;

- Haver entre 990 mil e 1,01 milhão de transações de dias anteriores na base de dados de monitoração;

Execução

Passos	Resultados Esperados
1) Executar o `nsca` do VTEF passando seus parâmetros;	O Nagios exibe o resultado da TMR sem alterações (igual ao que estava antes de executar o nsca/VTEF);
2) Em apenas uma excecução do `cargavtef`, inserir 50 transações de venda, 20 de consulta, 10 de desfazimento, 20 de cancelamento e 10 de sonda. Essas transações devem possuir *status* e códigos de respostas diversos, mas diferente de em andamento. Além dessas inserir 10 transações de cada tipo em andamento.	O Nagios exibe um TPPS-Venda de 40 transações e o TPPS-Geral de 90 transações (não são consideradas nem a sonda nem as em andamento);

CT03 – Desempenho da TMR

Descrição

Testa a capacidade do VTEF calcular a TMR em tempo hábil, menos de 1 minuto, seguidas vezes, considerando as metas do requisito não funcional de desempenho.

Pré-Condição

- Estar logado na interface web do Nagios;
- Haver entre 990 mil e 1,01 milhão de transações, de dias anteriores, na base de dados de monitoração;

Execução

Passos	Resultados Esperados
1) Executar o `nsca` do VTEF passando seus parâmetros;	O Nagios exibe o resultado das TMRs sem alterações (igual ao que estava antes de executar o nsca/VTEF);
2) Inserir 3000 transações de venda com tempo de resposta (TR) = 1s e 3000 com TR = 2s; Aguardar 1 min. pelo	O Nagios exibe o TMR-Venda igual a 1.5s para um total de 6000 transações;

resultado.	
3) Inserir 1000 transações de venda com TR = 6s e 4000 com TR = 1s; Aguardar mais ou menos 1 min. pelo resultado.	O Nagios exibe o TMR-Venda igual a 2s para um total de 5000 transações;
4) Inserir 500 transações em andamento, 100 sondas e 2000 transações diversas diferentes dessas primeiras, das quais 1000 com TR=2s e 1000 com TR=4s.	O Nagios exibe o TMR-Geral igual a 3s para um total de 2000 transações (não considera as sondas e as em andamento);

Capítulo 5
Resultados e Impactos

A validação das funcionalidades implementadas do VTEF foi realizada pela execução do Plano de Testes do projeto. Este capítulo apresenta os resultados desta validação e, como consequência destes, as correções e melhorias realizadas nos artefatos de requisitos, código fonte e no próprio roteiro de testes.

Em seguida são analisados resultados técnicos do ponto de vista da usabilidade da ferramenta e da utilidade e representatividade das informações com relação ao fluxo de transações.

Por fim são observados quais impactos desta ferramenta nas fábricas de software ou de serviços (área de suporte de Tecnologia da Informação) bem como os benefícios para o contratante do serviço de TEF (por exemplo, um varejista).

5.1 Execução dos Casos de Teste

Na primeira rodada de testes de sistema, após os testes unitários realizados durante a implementação, ao executar o caso de teste 01 (CT01), que se refere à corretude das métricas de estabilidade, foram apontadas as divergências abaixo, entre a computação do `cargavtef` e do próprio VTEF, utilizando o *script* `diff_res.sh` (apêndice A):

```
< TEF.LBSA Perc.Desfaz/dia: 4.464% (50 / 1120)
---D
> TEF.LBSA Perc.Desfaz/dia: 5.495% (50 / 910)
19,20c19,20
< TEF.LBSA Perc.Sonda/dia: 0.0.0.0: 16.071% (180 / 1120)
---
> TEF.LBSA Perc.Sonda/dia: 0.0.0.0: 19.780% (180 / 910)
```

Através desse resultado identificou-se que as métricas acima, quando calculadas pelo VTEF, estavam incluindo erroneamente no denominador da frequência relativa as transações de venda ou estorno com *status* em andamento. Com o intuito de assegurar a obtenção de indicadores mais significativos para o negócio de TEF, selecionando o grupo de transações mais adequado à cada métrica de estabilidade, foram revisadas todas as fórmulas especificadas e implementadas no programa de validação `cargavtef`.

Em consequência da revisão, foram incluídos os desfazimentos não efetivos na contagem da métrica de desfazimentos por loja, a exemplo dos desfazimentos por estabelecimento (`TEF.LBSA Perc.Desfaz/dia`). Isso porque o fato de uma loja enviar muitos desfazimentos, quer eles tenham sido aprovados ou não, já não é um bom indício. Além disso, existe a métrica de negação para indicar quanto dessas tentativas de desfazimento foram efetivadas. O outro exemplo de métrica revisada é o da sonda. Como a sonda não é obrigatoriamente enviada para a instância de Servidor de TEF que enviou a transação pendente, a métrica deve agregar o resultado por estabelecimento (endereço `0.0.0.0`) em vez de por endereço de destino (ServidorTEF). A agregação por endereço foi mantida apenas na métrica de sondas sem resposta. A figura abaixo mostra o resultado com sucesso do cálculo e da exibição das métricas quando as divergências e melhorias acima foram concluídas.

Service	Status	Last Check	Duration	Attempt	Status Information
TEF.LBSA_Perc.Cancel.Lj-0001/dia	WARNING	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	6.098% (25 / 410)
TEF.LBSA_Perc.Cancel.Lj-0002/dia	WARNING	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	5.385% (35 / 650)
TEF.LBSA_Perc.Cancel/dia	WARNING	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	5.660% (60 / 1060)
TEF.LBSA_Perc.Desfaz.Lj-0001/dia	OK	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	6.061% (20 / 330)
TEF.LBSA_Perc.Desfaz.Lj-0002/dia	OK	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	5.172% (30 / 580)
TEF.LBSA_Perc.Desfaz/dia	OK	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	5.495% (50 / 910)
TEF.LBSA_Perc.Neg.Cancel.Lj-0001/dia	CRITICAL	05-26-2009 20:35:39	9d 0h 7m 38s	1/1	25.000% (5 / 20)
TEF.LBSA_Perc.Neg.Cancel.Lj-0002/dia	WARNING	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	16.667% (5 / 30)
TEF.LBSA_Perc.Neg.Cancel/dia	CRITICAL	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	20.000% (10 / 50)
TEF.LBSA_Perc.Neg.Consulta.Lj-0001/dia	CRITICAL	05-26-2009 20:35:39	9d 0h 7m 38s	1/1	16.667% (10 / 60)
TEF.LBSA_Perc.Neg.Consulta.Lj-0002/dia	CRITICAL	05-26-2009 20:35:39	9d 0h 7m 38s	1/1	20.000% (25 / 125)
TEF.LBSA_Perc.Neg.Consulta/dia	CRITICAL	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	18.919% (35 / 185)
TEF.LBSA_Perc.Neg.Desfaz.Lj-0001/dia	CRITICAL	05-26-2009 20:35:39	9d 0h 7m 38s	1/1	33.333% (5 / 15)
TEF.LBSA_Perc.Neg.Desfaz.Lj-0002/dia	CRITICAL	05-26-2009 20:35:39	9d 0h 7m 38s	1/1	20.000% (5 / 25)
TEF.LBSA_Perc.Neg.Desfaz/dia	CRITICAL	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	25.000% (10 / 40)
TEF.LBSA_Perc.Neg.Venda.Lj-0001/dia	OK	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	3.226% (10 / 310)
TEF.LBSA_Perc.Neg.Venda.Lj-0002/dia	CRITICAL	05-26-2009 20:35:39	9d 0h 7m 38s	1/1	9.091% (50 / 550)
TEF.LBSA_Perc.Neg.Venda/dia	WARNING	05-26-2009 20:35:39	8d 21h 29m 29s	1/1	6.977% (60 / 860)
TEF.LBSA_Perc.Sonda/dia	WARNING	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	0.0.0.0: 19.780% (180 / 910)
TEF.LBSA_Perc.SondaSemResposta/dia	WARNING	05-26-2009 20:35:39	0d 0h 23m 8s	1/1	10.0.0.1: 8.333% (5 / 60) -- 10.0.0.2: 12.500% (15 / 120)

Figura 10. Cálculo e exibição com sucesso das métricas de estabilidade

Após a execução do CT01, foram implementadas as métricas TPS e TMR. Ao executar o CT02, foi observado que mesmo inserindo as transações no mesmo segundo (mesmo instante de tempo), as 40 transações de venda não apareciam na métrica TPS-Venda (apenas 35), nem as 90 de todo tipo na TPS-Geral (apenas 56) conforme Figura 11. Após executar diretamente no banco de dados a mesma *query* utilizada pelo VTEF e obter o mesmo resultado, de 35 e 56 transações, foi percebido que a *query* não incluia na contagem as transações em andamento e as de sonda. Apesar da diferença, este é um comportamento esperado da *query*, porém como não foi descrito nas fórmulas do requisito RF04, levou a um erro de definição do seu

caso de teste CT02. Conceitualmente a TMR não leva em conta as sondas, porque são originadas no AutorizadorTEF, e as transações ainda sem resposta (em andamento). Sabendo o autor, que um alto TPS é uma das principais causas da elevação no tempo de resposta, a TPS foi implementada considerando as mesmas transações do TMR, mesmo sem estar especificado. Este é o motivo pelo qual o VTEF calculou corretamente as 56 e 35 transações.

TEF.LBSA_TPS.Geral/min	OK	05-26-2009 22:28:14	0d 0h 25m 34s	1/1	qtd_trs=56
TEF.LBSA_TPS.Venda/min	OK	05-26-2009 22:28:15	0d 0h 25m 34s	1/1	qtd_trs=35

Figura 11. Resultado falho por erro de procedimento do CT02

Após ajustar o arquivo de entrada do `cargavtef` (`carga.dat`), e o procedimento do CT02, para que fossem incluidas as 40 vendas pendentes ou concluídas e as 90 transações gerais, sem contar as sondas e as em andamento, o teste foi aprovado como mostra a figura abaixo.

TEF.LBSA_TPS.Geral/min	OK	05-27-2009 00:19:51	0d 2h 16m 57s	1/1	qtd_trs=90
TEF.LBSA_TPS.Venda/min	OK	05-27-2009 00:19:51	0d 2h 16m 57s	1/1	qtd_trs=40

Figura 12. Resultado positivo do CT02

Ainda sobre o resultado do CT02, em virtude do propósito de não incluir as transações em andamento, a métrica foi renomeada para uma sigla mais significativa, TPPS (Transações Processadas Por Segundo), enquanto a TPS indica a medida da capacidade de recebimento de um sistema.

Para concluir a execução do Plano de Testes, depois da implementação do TPPS, foram implementadas as métricas TMR, cuja especificação foi ratificada pelas correções nas iterações anteriores das outras métricas, e executado o CT03, obtendo um resultado conforme esperado e exibido abaixo:

TEF.LBSA_TMR.Geral/min/maxTrs	OK	05-27-2009 21:00:19	0d 0h 0m 32s	1/1	1.500 (qtd_trs=6000)
TEF.LBSA_TMR.Venda/min/maxTrs	OK	05-27-2009 21:00:19	0d 0h 0m 32s	1/1	1.500 (qtd_trs=6000)

Figura 13. Resultado do passo 2 do CT03

TEF.LBSA_TMR.Geral/min/maxTrs	OK	05-27-2009 21:01:19	0d 0h 1m 23s	1/1	2.000 (qtd_trs=5000)
TEF.LBSA_TMR.Venda/min/maxTrs	OK	05-27-2009 21:01:19	0d 0h 1m 23s	1/1	2.000 (qtd_trs=5000)

Figura 14. Resultado do passo 3 do CT03

TEF.LBSA_TMR.Geral/min/maxTrs	OK	05-27-2009 21:02:19	0d 0h 2m 13s	1/1	3.000 (qtd_trs=2000)
TEF.LBSA_TMR.Venda/min/maxTrs	OK	05-27-2009 21:02:19	0d 0h 2m 13s	1/1	2.000 (qtd_trs=500)

Figura 15. Resultado do passo 4 do CT03

5.2 Análise dos Resultados do Projeto

Embora não se tenha dado tanta ênfase à abordagem iterativa-incremental, ao explicitar apenas 3 iterações (uma para cada caso de teste e implementação de seu respectivo tipo de métrica), ela foi fundamental importância para conclusão do projeto porque a implementação não foi postergada em detrimento do refinamento dos requisitos.

Em termos práticos, algumas métricas somente foram especificadas, e sua importância elucidadas, após questionamentos que se referiam a outras já implementadas. Por exemplo, após a implementação do desfazimento, analisando criticamente suas causas, se chegou à metrica do TPS elevado. Isso porque quando sobrecarregado, o AutorizadorTEF pode não responder, provocando *timeout* e deixando a transação em andamento, o que exige o envio do desfazimento pelo ServidorTEF (ele não sabe se a transação ficou pendente no Autorizador). Já a TMR foi idealizada logo em seguida para medir o impacto do TPS no sistema. Ainda sobre o desfazimento, durante a implementação da contagem das transações, percebeu-se que a métrica tinha maior importância se levasse em conta não apenas dos efetivos mas de todas tentativas de desfazimento. Para não perder a informação de quantos foram efetivados, replicou-se a métrica de negação para este tipo de transação quando pensava-se em aplicá-la apenas à consulta e à venda ou estorno.

Por outro lado, apesar dos resultados perceptíveis de planejamento e execução do projeto, faz-se necessário apontar possíveis melhorias. Algumas métricas podem ser complementadas seja através da mudança no cálculo, ou na agregação de seu resultado, ou mesmo através da criação de uma nova métrica. Foram identificados 3 casos desses:

i. A TPPS não capta um pico de transações quando estas se dividem em dois segundos contíguos porém ocorrido no mesmo intervalo de um segundo.

ii. O resultado das métricas de negação deveria ser agregado por código de resposta para saber se a causa é devido a erro de entrada (validação) ou problema operacional.

iii. A TMR é uma média, e para maior confiabilidade, poderia ser acompanhada de uma métrica de desvio padrão, ou outra medida de dispersão.

Outros aspectos que poderiam sofrer melhorias são relacionados a periodicidade das métricas de estabilidade. O resultado poderia se referir a eventos não apenas ocorridos no dia mas em uma semana, quinzena ou até em um mês (a depender da capacidade de desempenho). Por fim, o intervalo de ação do VTEF no cálculo dessas métricas, ou seja, a taxa de verificação para enviar os resultados poderia ser parametrizada.

5.3 Impactos da ferramenta

Os principais impactos advindos da utilização do VTEF como ferramenta de monitoração de Autorizadores de TEF, em ambiente de homologação ou produção, se desdobram a partir da independência dos traces de arquivos de log para identificar anormalidades neste sistema ou em sua infra-estrutura. Neste projeto essa independência se dá através da notificação imediata de todos envolvidos e interessados nos problemas que afetam direta ou indiretamente o Autorizador TEF proporcionando maior rapidez no conhecimento do problema para atuar na solução.

Para a fábrica de SW fornecedora do AutorizadorTEF, essas notificações interessam nos casos de erros causados por falha do sistema, enquanto que para a fábrica de Serviços interessam quando são abertos chamados de erros que afetam de imediato a disponibilidade do serviço de TEF. Todavia, nem sempre essas fábricas dispõem de contingente humano para atender a uma demanda de imediato, aparentemente ofuscando o benefício do parágrafo anterior, porém, quando há um colaborador disponível, a empresa se beneficia com conhecimento prévio dos erros. Além disso, em termos de identificação preventiva de instabilidade, caso o sistema não seja monitorado com este intuito, considerando que todos casos são urgentes, em virtude da natureza do sistema, poderá haver problema de alocação de profissional para realizar o atendimento, principalmente em ocorrências simultâneas enquanto que monitoradas preventivamente poderiam ser avaliadas em sequência.

Como consequência, sendo as notificações suficientemente esclarecedoras e funcionais, a probabilidade de um problema ser resolvido mais rapidamente é maior, aumentando a disponibilidade do sistema, e portanto, gerando economia para empresa que utiliza o Autorizador TEF na aprovação de seus pagamentos devido à diminuição nas perdas de venda por causa de indisponibilidade não planejada.

Outra consequência positiva da monitoração constante e preventiva é a identificação de problemas que anteriomente ocorriam em uma frequência baixa o suficiente para passarem desapercebidos. Isso pode vir a ocorrer por exemplo quando a rede de comunicação tem uma certa taxa de perda de pacotes que provoca *timeouts* de tempos em tempos, levando a alguns clientes esperarem mais que o necessário para pagarem suas compras ou correspondencias, gerando insatisfação.

Capítulo 6
Conclusão e Trabalhos Futuros

Nesse trabalho foi proposto o desenvolvimento do VTEF, uma ferramenta de monitoração de Autorizadores de TEF integrada ao Nagios, o qual foi realizado com sucesso, sendo capaz de identificar, mais rapidamente, problemas que afetam a disponibilidade do sistema, e consequentemente, a organização do trabalho nas fábricas de software e de serviços.

De acordo com os resultados observados na execução do Plano de Teste, e na análise dos resultados, o VTEF apresentou funcionalidades e qualidades em conformidade com o especificado. Verificou-se que um fator cooperante para atingir esse objetivo foi a adoção da estratégia iterativa-incremental que proporcionou a idealização de certas métricas justificadas por outras já haviam sido validadas.

Foram identificadas também necessidades de melhorias, porém, no geral, as métricas definidas no projeto representaram bem o fluxo das transações codificando-o em informações significativas ao negócio de TEF. Tanto individualmente como no caso da métrica de tentativas de desfazimento de uma loja, quanto em combinação com a métrica de negação de desfazimento, que mede a quantidade das tentativas contadas pela primeira que não foram efetivadas.

Vale salientar, dentre os principais impactos técnico-econômicos, o ganho de produtividade na análise dos problemas que atingem o AutorizadorTEF e a possibilidade de planejar melhor, com antecedência, a alocação dos profissionais das fábricas a partir de uma visão preventiva de erros no sistema ou no ambiente.

Concluindo, foram identificadas as seguintes oportunidades de trabalhos futuros que podem derivar deste:

- A aplicação no VTEF de itens que ficaram no escopo negativo como segurança na monitoração, monitoração de segurança, gerenciamento de redes, administração da utilização de recursos dos *hosts* (disco, CPU e memória) e métricas de eficiência de vendas.
- A utilização no VTEF de métodos, como correlação linear, para verificação, no histórico das transações, da relação entre um determinado nível elevado de TPS com problemas de conexão com o banco de dados, ou entre TPPS e TMR.

Bibliografia

[1] Dataquest Insight: Unplanned Downtime Rising for Mission-Critical Applications. http://www.gartner.com/DisplayDocument?ref=g_search&id=770813. Visitado em 8 de fevereiro de 2009.

[2] JOSEPHSEN, David. Building a Monitoring Infrastructure with Nagios. 1ª edição. Editora Prentice Hall, p. xix e xx, 2007.

[3] MARCUS Evan, STERN Hal. Blueprints for High Availability. 2ª edição. Editora Wiley Publishing Inc, p.20, 2003.

[4] Nagios Propaganda - Media Coverage, Awards and Recognition, Books. http://www.nagios.org/propaganda/. Visitado em 8 de fevereiro de 2009.

[5] Nagios Version 3.x Documentation. Disponível em: http://nagios.sourceforge.net/docs/nagios-3.pdf. Visitado em 9 de fevereiro de 2009.

[6] Nagios Exchange. http://www.nagiosexchange.org. Visitado em 14 de fevereiro de 2009.

[7] NSCA (with Nagios). Disponível em: http://nagios.sourceforge.net/download/contrib/documentation/misc/NSCA_Setup.pdf. Visitado em 14 de fevereiro de 2009.

[8] NSCA_Win32Client. Disponível em: http://nagiosexchange.altinity.org/nagiosexchange/NSCA_Win32Client/. Visitado em 14 de fevereiro de 2009.

[9] NRPE Documentation. Disponível em: http://nagios.sourceforge.net/docs/nrpe/NRPE.pdf. Visitado em 14 de fevereiro de 2009.

[10] 2007 EFT-8 Electronic Funds Transfer Guide. Disponível em: http://www.revenue.state.il.us/publications/ElectronicServices/eft8.pdf. Visitado em 20 de fevereiro de 2009.

[11] Electronic Funds Transfer EFT. Disponível em: http://www.in.gov/dor/files/eft-100.pdf. Visitado em 20 de fevereiro de 2009.

[12] Electronic Funds Transfer Information Guide, Publication 80. Disponível em: http://www.boe.ca.gov/pdf/pub80.pdf. Visitado em 20 de fevereiro de 2009.

[13] Regulation 1707 - Electronic Funds Transfer. Disponível em: http://www.boe.ca.gov/pdf/reg1707.pdf. Visitado em 20 de fevereiro de 2009.

[14] FDIC: FDIC Law, Regulations, Related Acts - Consumer Protection. http://www.fdic.gov/regulations/laws/rules/6500-3100.html#6500205.3. Visitado em 20 de fevereiro de 2009.

[15] READ, R. J. EFTPOS: electronic funds transfer at point of sale. Electronic & Communication Engineering Journal. P. 263, 1989.

[16] ISO8583 International Standard, Version 1993. p. 47,77.

[17] The Spirt of the RUP. Disponível em: http://download.boulder.ibm.com/ibmdl/pub/software/dw/rationaledge/dec01/TheSpiritoftheRUPDec01.pdf. Visitado em 7 de março de 2009.

[18] ROYCE, W. W. Managing the Development of Large Software Systems. Disponível em: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf. Visitado em 7 de março de 2009.

[19] PARNAS, D. A Rational Design Process: How and Why to Fake It. IEEE Transactions on Software Engineering, Vol. 12, Issue 2, p. 251 – 257, 1986.

[20] AVIZIENIS, A. et al. Fundamental Concepts of Computer System Dependability. IARP/IEEE-RAS Workshop on Robot Dependability: Technological Challenge of Dependable. Robots in Human Environments – Seoul, Korea, May 21-22, 2001.

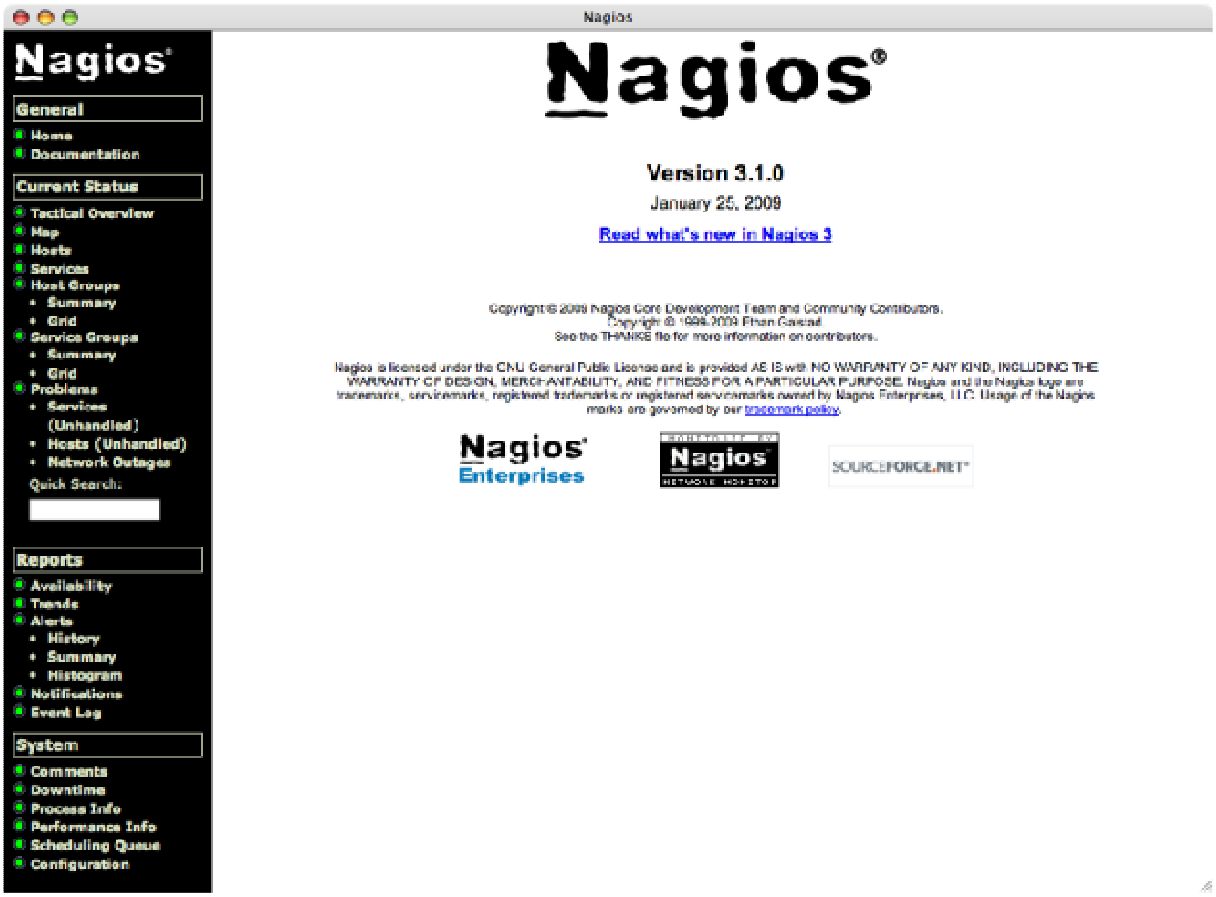
Apêndice A
Scripts de Teste

1) Nome: `diff_res.sh`
   Propósito: Compara os resultados das métricas escritos na saída padrão pelo VTEF e pelo programa de teste `carga_vtef`.
   Conteúdo do *script*:

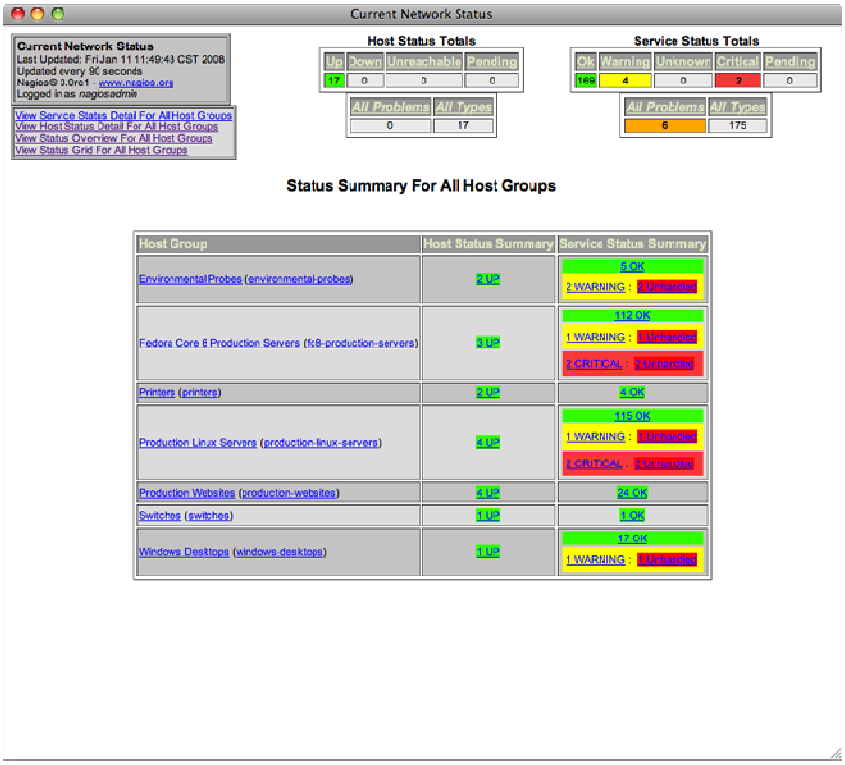
```
cp /usr/local/nagios/bin/vtef.log .
grep TEF.LBSA vtef.log | sort > vtef.grep
grep TEF.LBSA carga.log | sort > carga.grep
diff vtef.grep carga.grep > diff.out
```

Telas do Nagios

1) Homepage



2) Sumário de *Hostgroup*



3) Detalhes de Serviços