Glauber Moreira Prates

Treemaps Aplicados ao Gerenciamento de Ativos em Rede

Belo Horizonte

2008

Glauber Moreira Prates

Treemaps Aplicados ao Gerenciamento de Ativos em Rede

Monografia apresentada para a conclusão do curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Minas Gerais

Orientador:
Constantino Seixas Filho

Supervisor:
Denilson Marcelino Costa

Universidade Federal de Minas Gerais

Belo Horizonte

2008

Resumo

No domínio de gestão de processos e plantas industriais, a tomada de uma decisão tem que ser feita, geralmente, em um intervalo curto de tempo e baseada em uma grande quantidade de informações. A maneira como as informações de interesse são apresentadas e as técnicas de visualização podem facilitar muito essa tarefa. Dessa maneira, a forma de apresentar tais informações tornou-se fator muito importante e tem sido alvo de muitas pesquisas na área da computação gráfica.

Este trabalho apresenta uma das técnicas mais modernas de representar informação quantitativa de estruturas hierárquicas, explorando conceitos de ergonomia. Todos os algoritmos expostos foram estudados e comparados, visando escolher aquele que fosse mais adequado para representar indicadores de desempenho de equipamentos de redes de comunicação de dados.

Definido o algoritmo mais apropriado, a ferramenta foi desenvolvida e incorporada ao software XRatel fornecendo para os seus usuários uma ferramenta mais eficaz no auxílio à tomada de decisão.

Abstract

When it comes to processes management and industrial plants a decision has to be made, normally, in a short time, based in a great amount of information. The way the important information is presented and the visualization techniques can make this job much easier. Therefore, the way information is presented is very important and is the object of many researches in the area of graphic computer science.

This work presents one of the most modern techniques to show quantitative information of hierarchical structures, exploring the concept of ergonomics. All the algorithms showed were studied and compared, in order to choose the more adequate to represent indicators of performance of equipments of data communication nets.

Once the most appropriate algorithm was chosen, the tool was developed and incorporated to the XRatel software, giving the users a more effective tool to help the decision making.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

TI Tecnologia da Informação

RPM Gerência de Performance em Tempo Real

ISO International Organization for Standardization

TA Tecnologia da Automação

OPC OLE for Process Control / interOperability, Productivity & Collaboration

PIMS Process Information Management System

MIB Management Information Base

SNMP Simple Network Management Protocol

WMI Windows Management Instrumentation

UDP User Datagram Protocol

SMI Estrutura de Informações de Gerenciamento

OID Object ID

CLP Controlador Lógico Programável

CIM Common Interface Model

MOF Managed Object Format

SCADA Supervisory Control and Data Acquisition

DDE Dynamic Data Exchange

KPI Key Performance Indicator

1 Introdução

A indústria química e petroquímica, ou plantas industriais com grande número de dispositivos em rede, são exemplos práticos de aplicações que precisam estar constantemente sendo verificadas e analisadas para diagnosticar desvios de funcionamento que podem prejudicar o desempenho econômico da planta.

Nesse contexto, procurar qual malha de controle é a responsável pelo baixo desempenho de um setor na indústria, dentro de uma planta industrial composta por centenas ou milhares de malhas, ou descobrir qual Controlador Lógico Programável (CLP), Roteador ou Servidor de Aplicação está sendo responsável pelo baixo desempenho em uma rede ethernet industrial, pode ser uma tarefa tão difícil quanto "procurar uma agulha no palheiro" (MITCHELL; SHOOK, 2005).

1.1 Motivação

Para evitar prejuízos relacionados a problemas como os apresentados, alguns softwares têm sido desenvolvidos com o objetivo de verificar e avaliar constantemente tais malhas ou dispositivos com o intuito de perceber eventuais desvios de funcionamento e permitir uma atuação preventiva por parte do operador. No entanto, a eficiência de tais softwares passou a estar relacionada com fatores humanos no que diz respeito à capacidade de análise, pelo operador, da quantidade de informação disponibilizada pelos programas.

O pouco tempo que um operador tem para tomar uma decisão com base nas informações apresentadas e o impacto direto que essa decisão terá no desempenho econômico

dessa planta constituem uma restrição cada vez mais relevante e contribuem diretamente para o sucesso, ou não, de estratégias de gerenciamento de performance em tempo real. Além dessa, a restrição de espaço, na hora de exibir graficamente estruturas tão grandes, é um problema também relevante (SHNEIDERMAN, 1992) e tem impacto direto na facilidade de análise dos dados por parte do usuário.

Por isso, métodos de visualização de hierarquias de informações, que integrem de forma mais suave a tarefa que se deseja desenvolver, com o grau de conhecimento do usuário que busca as informações, são de extrema importância e têm sido objetos de estudo de muitos pesquisadores atualmente.

Para permitir uma melhor integração entre usuários e os métodos de visualização dessas estruturas, pesquisadores afirmam que alguns cuidados devem ser tomados e algumas restrições obedecidas para desenvolver métodos de visualização de informações cada vez mais voltados à tomada de decisão.

1.2 A empresa: XRatel Software

Esse projeto foi desenvolvido na XRatel Software, subsidiária da Atan Sistemas, que desenvolve softwares totalmente baseados nos conceitos de Gerência de Performance em Tempo Real (RPM), cujas funcionalidades foram projetadas para atender às necessidades específicas dos gerentes de infra-estrutura de Tecnologia da Informação (TI) e de Automação.

As ferramentas da XRatel Software permitem que os usuários obtenham, de forma rápida, informações detalhadas de cada sistema monitorado (dispositivos conectados em rede), ajudando-os a manter esses sistemas operando corretamente.

Devido à grande quantidade de informação que o software permite disponibilizar, o grande problema passa a ser a determinação, por parte do usuário, de quais dados são relevantes e como trabalhar com esses dados para retirar as informações necessárias à

manutenção de um bom desempenho da rede.

1.3 Treemaps

Treemaps são mapas bidimensionais que facilitam a visualização de estruturas hierárquicas para propiciar uma rápida tomada de decisão. Tais mapas exploram conceitos básicos de ergonomia que estabelecem que o ser humano foque inicialmente seu olhar em figuras grandes para só depois atentar para figuras menores. As dimensões (área da figura) e cor são utilizadas para codificar os atributos (importância econômica, desempenho, entre outros) dos nodos folhas. Com essas informações, essa ferramenta gráfica permite uma rápida análise de nodos e subárvores facilitando comparações e permitindo a detecção de padrões e exceções.

Na área de automação, treemaps têm sido usados para exibir a condição operacional de malhas de controle das diversas unidades que compõem as áreas de processo de uma ou mais plantas industriais. No entanto, novas aplicações para as mesmas têm sido sugeridas devido à facilidade que tal método de visualização proporciona aos usuários no que diz respeito à análise comparativa de uma grande quantidade de informação (SHAH; MITCHEL; SHOOK, 2006) (MATRIKON, 2005) (HOLDEN, 2007).

1.4 Objetivos do Projeto

É de interesse da empresa parceira disponibilizar um componente que implemente um treemap que deverá ser desenvolvido em Delphi e incorporado às aplicações da XRatel. Com essa nova ferramenta de visualização, pretende-se melhorar a interface do programa com o usuário permitindo uma integração mais suave da ferramenta com os propósitos a que ela se destina. Portanto, faz parte dos objetivos deste trabalho:

1. Levantar uma forma tradicional de visualização de estruturas hierárquicas, apontando os seus problemas e fazendo um paralelo com as vantagens da utilização de

Treemaps.

2. Apresentar os diversos algoritmos existentes para a construção de treemaps definindo o algoritmo mais adequado com base nas características da aplicação.

3. Definir os Key Performance Indicator (KPI) que serão utilizados para construção dos treemaps.

4. Implementar, em Delphi, a classe treemap a ser utilizada pelo XRatel.

1.5 Organização do Trabalho

No capítulo 2 são apresentadas as características dos softwares do XRatel e um detalhamento das tecnologias e padrões utilizados por ele. Nesse capítulo é feito, também, um levantamento das características e restrições impostas a métodos de visualização de estruturas hierárquicas fazendo uma comparação entre o método atualmente utilizado pelo XRatel e o novo método implementado, procurando ressaltar os ganhos que se pretende obter com a utilização do novo método.

O capítulo 3 apresenta os principais algoritmos disponíveis para a criação dos treemaps e define quais foram os critérios utilizados e as características relevantes para a escolha do algoritmo utilizado.

O capítulo 4 é destinado à apresentação das decisões de implementação adotadas e as justificativas para a escolha feitas no que diz respeito aos aspectos mais importantes da ferramenta.

O capítulo 5 apresenta os resultados obtidos e as métricas utilizadas para validação da ferramenta proposta.

Por fim, o capítulo 6 apresenta os passos percorridos para atingir os objetivos propostos com um breve comentário a respeito do resultado obtido.

2 Fundamentos Teóricos

Em grandes redes, muitas aplicações, processos e dispositivos estão conectados e dependem, todo o tempo, uns dos outros. Nessas condições, é essencial garantir a eficiência e a confiabilidade desses componentes de forma a não comprometer o ambiente de negócios.

Contudo, é muito importante, para se evitar perdas, que seja possível identificar o problema e tomar as providências necessárias para o restabelecimento normal da operação no menor período de tempo possível. Para tanto, é preciso monitorar continuamente o comportamento de todo o sistema utilizando o conceito de gerenciamento de performance em tempo real (BRANT et al., 2006).

No entanto, visualizar e analisar estruturas com a complexidade desse tipo de rede tem se tornado uma tarefa cada vez mais difícil devido à grande quantidade de dados que se consegue reunir com o auxílio de softwares baseados no conceito de gerenciamento de performance em tempo real. Apresentar tanta informação em uma região pequena como a tela de um computador, de forma clara e de fácil compreensão por parte do usuário final, não é uma tarefa trivial e tem sido assunto de pesquisa de diversos estudiosos (FEW, 2006). Dessas pesquisas, diversos métodos de visualização de estruturas hierárquicas têm sido desenvolvidos.

2.1 O Gerenciamento de Redes

2.1.1 Considerações Gerais

De acordo com a tabela 1, apresentada por Wilson (2003), que mostra um estudo de caso dos prejuízos advindos da interrupção de funcionamento das redes de TI de seis empresas, fica notável a importância de se ter uma infra-estrutura de TI confiável e que opere regularmente.

Case Study	**Annual Revenue**	**Downtime Cost**	**Cost/Hour**
Energy	$6,75 billion	$4,3 million	$1624
High tech	$1,3 billion	$10,2 million	$4167
Health care	$44 billion	$74,6 million	$96632
Travel	$850 million	$2,4 million	$38710
Finance (U.S.)	$4,0 billion	$10,6 million	$28342
Finance (Europe)	$1,2 billion	$379000	$1573

Tabela 1: Custo anual de falhas em redes: Produtividade vs. Produção

Notada a importância econômica desta área, os padrões de gerenciamento de rede começaram a amadurecer no final da década de 80 e diversos estudos foram desenvolvidos. No intuito de fornecer pilares para a discussão do assunto a International Organization for Standardization (ISO) criou um modelo de gerenciamento de rede que apresenta de forma mais estruturada as possíveis áreas de aplicação:

- **Gerenciamento de Desempenho:** busca quantificar, medir, analisar e controlar o desempenho dos diferentes ativos da rede como enlaces, roteadores, e computadores.
- **Gerenciamento de Falhas:** objetiva registrar, detectar e reagir às condições de falha da rede como interrupção de serviço em enlaces ou falha de software ou hardware de servidores.
- **Gerenciamento de Configuração:** permite que o administrador da rede saiba

quais dispositivos fazem parte da rede administrada, e quais são suas configurações de hardware e software.

- **Gerenciamento de Contabilização:** permite a especificação, registro e controle dos recursos da rede disponível a cada usuário ou dispositivo.

- **Gerenciamento de Segurança:** controla o acesso aos recursos da rede de acordo com a política de segurança do administrador da rede.

O modelo apresentado é bastante difundido entre os profissionais de TI, e diversas ferramentas foram propostas permitindo o gerenciamento a nível corporativo de todas as áreas definidas pela ISO. No entanto, na área de Tecnologia da Automação (TA), a realidade não é bem essa. As ferramentas propostas para a área de TI não podem ser prontamente utilizadas nessa área, por não atenderem requisitos de desempenho e aplicabilidade, além de não permitirem uma integração com sistemas de automação (via OPC, por exemplo) para monitoração em tempo real das redes de automação (SEIXAS FILHO et al., 2005).

A necessidade de ferramentas que permitam o monitoramento de redes de TA como é feito com as redes de TI torna-se ainda mais eminente, tendo em vista o crescimento acelerado dessas redes e das novas funcionalidades que são, continuamente, adicionadas às configurações existentes. Nesse contexto de crescimento acelerado, manter um gráfico de tendência do desempenho dessas redes à medida que vão sendo adicionadas novas funcionalidades é essencial para evitar que problemas venham a ocorrer. Por isso, a possibilidade de integrar o sistema de monitoração com sistemas Process Information Management System (PIMS) torna-se bastante relevante.

Outra preocupação que tem levado as empresas a buscarem sistemas com essa finalidade é o fato das redes de TA estarem, também, vulneráveis a invasões indesejadas.

Os principais componentes de uma arquitetura de gerenciamento de rede são, segundo Kurose e Ross (2004), formados basicamente por três componentes: a Entidade Gerenci-

adora, o Dispositivo Gerenciado e o Protocolo de Gerenciamento utilizado.

A Entidade Gerenciadora é a aplicação que fornece ao responsável pela rede as informações desejadas, permitindo a análise e identificação de desvios de comportamento que podem ser prejudiciais ao desempenho do sistema.

O Dispositivo Gerenciado é um ativo de rede que faz parte da rede gerenciada. Nesse dispositivo, é executado um processo denominado agente de gerenciamento de rede, que é responsável por se comunicar com a entidade gerenciadora e realizar ações locais sob o comando e o controle da mesma. Toda informação disponibilizada pelo dispositivo gerenciado é organizada em uma estrutura de dados que pode ser acessada pela entidade gerenciadora.

O último componente é o Protocolo de Gerenciamento de Rede que é executado entre a entidade gerenciadora e o agente de gerenciamento que permite que a primeira investigue o estado dos dispositivos gerenciados.

Os protocolos utilizados pelo XRatel são o Simple Network Management Protocol (SNMP) projetado para ser independente de produtos de fabricantes específicos e o protocolo de gerenciamento específico da Microsoft, o Windows Management Instrumentation (WMI). Uma breve abordagem destes dois protocolos será apresentada nas seções seguintes.

2.1.2 O Protocolo SNMP

O SNMP foi projetado, implementado e disponibilizado por um grupo de pesquisadores, usuários e administradores de rede em poucos meses. Desde então, evoluiu até a versão mais recente, lançada em abril de 1999, denominada SNMPv3.

Como o protocolo foi desenvolvido rapidamente e disponibilizado em uma época em que a necessidade de gerenciamento de rede começava a se difundir, encontrou uma ampla aceitação e se tornou o modelo mais usado.

Esse protocolo não é orientado a conexões, pertence à suíte IEEE TCP/IP 802 e define detalhes triviais, como a estrutura que será utilizada para apresentar e organizar os dados, os tipos de dados que serão trocados, o protocolo para transportar as informações e comandos e questões relativas à segurança (principal aprimoramento do SNMPv3 em relação ao SNMPv2) que precisam ser formalizados para permitir a troca de informações entre a entidade gerenciadora e os dispositivos gerenciados.

Outra vantagem do protocolo é que ele trabalha sobre a camada de transporte User Datagram Protocol (UDP), sendo uma implementação leve.

A Estrutura de Informações de Gerenciamento (SMI) é um conceito bastante importante no protocolo, e apesar de possuir um nome pouco representativo, trata-se da linguagem na qual as informações de gerenciamento estão especificadas. Essa linguagem é necessária para assegurar que a sintaxe e a semântica dos dados de gerenciamento de rede sejam bem definidas e não apresentem ambigüidade.

Outro componente importante do protocolo é a estrutura de dados que guarda as informações referentes a um dispositivo da rede, denominada Management Information Base (MIB), cujos valores representam, em conjunto, o estado atual do dispositivo.

A MIB consiste em um conjunto de objetos de dados, os quais são chamados de Object ID (OID). Os OIDs são definidos em grupos, de acordo com suas características de aplicação, tipos de dados e permissões de leitura e escrita. Os grupos padrões são divididos em dados de sistema, interfaces de rede e interfaces dos protocolos principais da suíte TCP/IP. Basicamente, os OIDs definidos como padrões, que estão em todos os dispositivos que trabalham com o SNMP, são destinados ao gerenciamento de redes, fornecendo informações como:

- Taxa de utilização dos links/portas.
- Contadores de erros, descartes e colisões.
- Contadores de pacotes, broadcast, unicast, etc.

- Endereçamento e estado das portas (link ativo/inativo).
- Utilização de recursos (processamento, memória, etc.).

OIDs complementares podem ser definidos pelos fabricantes de equipamentos para ampliar as MIBs padrões em função das necessidades da aplicação. Dessa forma, os fabricantes fornecem os arquivos MIB específicos para os seus produtos, os quais são interpretados pelo SNMP Manager possibilitando a comunicação.

Com o crescimento do uso da rede Ethernet nos sistemas de automação, os fabricantes de equipamentos para uso industrial (principalmente CLP), começaram a disponibilizar módulos de comunicação com SNMP e definir MIBs específicas para os seus produtos, os quais acrescentam OIDs para monitoração dos dispositivos de automação. Esta é uma tendência real para a monitoração de redes de automação, em todos os níveis. Alguns exemplos de OIDs adicionais encontrados nas MIBs de equipamentos de automação são:

- Dados e estados da aplicação (nome, parada, funcionando, etc.).
- Dados dos protocolos de controle usados (Ethernet/IP, ModbusTCP, etc.).
- Nome do dispositivo, número de série, nome do fornecedor, revisão do hardware, versão do firmware, etc.

Todas as informações definidas nas MIBs podem ser solicitadas pela entidade gerenciadora, ou enviadas pelo agente presente nos dispositivos gerenciados. Dessa forma, é possível fazer um monitoramento eficiente da rede como um todo.

2.1.3 A Tecnologia WMI

O WMI é uma infra-estrutura para gerenciamento de dados e operações para sistemas baseados na plataforma Windows.

Essa tecnologia usa um padrão público conhecido como Common Interface Model (CIM), que é orientado a objetos, para armazenar os dados.

Os objetos básicos do padrão são as classes que representam objetos reais (aplicativos, processadores, memórias e outros) que possuem propriedades a serem monitoradas.

Como o protocolo SNMP apresentado anteriormente, essa tecnologia define estruturas que formalizam a troca de informações entre um dispositivo gerenciado e a entidade gerenciadora (MSDN, 2007).

Da mesma forma que o SNMP define uma linguagem na qual as informações de gerenciamento estão especificadas, o WMI define sua própria linguagem cuja sintaxe representa a forma textual em que são definidos as informações referentes aos objetos gerenciados.

As informações disponibilizadas via esse protocolo também estão organizadas em estruturas que facilitam a identificação da informação que se deseja obter através de uma requisição.

2.2 O XRatel como Entidade Gerenciadora

O XRatel foi concebido a partir de necessidades identificadas em vários sistemas industriais e ambientes de missão crítica, visando atender a premissas como confiabilidade, alto desempenho, facilidade de configuração e integração com outros sistemas e a capacidade de prover informações sumarizadas para facilitar o diagnóstico.

As informações fornecidas pela ferramenta de gerenciamento devem ser confiáveis e retratar a real condição do sistema, na freqüência exigida pelo administrador da rede. Neste contexto, problemas como, por exemplo, a amostragem simultânea de parâmetros que são utilizados no cálculo dos índices, torna-se uma tarefa crítica. No XRatel, esta confiabilidade é conseguida devido à existência de um núcleo *multi-thread*, que permite a requisição de indicadores simultaneamente em mais de um dispositivo gerenciado.

Um sistema de monitoramento, como o XRatel, tem que ser realmente eficiente em situações críticas de operação. Por exemplo, quando existe um problema em um link de comunicação e este fica congestionado devido a algum inconveniente, a ferramenta

de gerenciamento tem que acrescentar tráfego a este link para realizar diagnósticos e obter respostas em tempos prédeterminados, sem causar mais congestionamento no link problemático. Para que estas premissas de alto desempenho sejam obedecidas, o XRatel trabalha sobre a camada de transporte UDP obtendo uma implementação bastante leve.

Como apresentado anteriormente, é indispensável que sistemas como este possuam uma integração com sistemas PIMS e Supervisory Control and Data Acquisition (SCADA). No XRatel, isto pode ser facilmente conseguido pelo fato do mesmo disponibilizar uma interface de comunicação nos padrões OPC-DA e Dynamic Data Exchange (DDE).

A ferramenta de visualização, como foi discutido, deve ser voltada à tomada de decisão, e apresentar a informação na quantidade necessária de acordo com o grau de instrução do operador e a tarefa que o mesmo pretende realizar com base nestas informações.

Além da forma como a informação é disponibilizada, o acompanhamento do desempenho de redes de TA e TI deve ser feito através de sistemas que gerem índices consolidados a partir de uma informação bruta coletada dos equipamentos, tornando possível uma análise rica, mesmo em ambientes que não disponham de especialistas em redes de comunicação e sistemas.

Nestes casos, quando utilizamos ferramentas convencionais para diagnosticar uma rede de automação, a análise somente pelos indicadores convencionais pode levar a conclusões precipitadas, por isso, combinar alguns indicadores pode possibilitar um diagnóstico rápido e eficiente com alto desempenho para aplicações de tempo real.

Foi nestes termos que surgiu no XRatel o conceito de índice saúde do equipamento que é calculado a partir da composição ponderada de alguns parâmetros, tais como taxa de utilização, efetividade de transmissão e taxa de *broadcast*. Além deste índice predefinido em cima de estudos, é permitido ao administrador da rede criar seus próprios indicadores.

Uma peculiaridade bastante interessante do XRatel é que ele não necessita de um agente específico, instalado no dispositivo gerenciado, para a coleta de dados. Isso evita ter que instalar softwares extras nos ativos gerenciados. Desta forma, a única instalação

que precisa ser feita é a da máquina que foi destinada para ser o servidor desta aplicação.

Além de permitir a utilização de banco de dados externo através da comunicação nos padrões OLE for Process Control / interOperability, Productivity & Collaboration (OPC) e DDE, o XRatel permite a gravação de dados históricos somente com as ferramentas próprias do software possibilitando a geração posterior de gráficos de tendência entre outras funcionalidades.

O XRatel disponibiliza ferramentas para o administrador de uma rede atuar apenas nas áreas de gerência de desempenho e de falhas. Sendo assim, qualquer dispositivo que disponibiliza informações utilizando o SNMP ou que tenha o Windows instalado, disponibilizando dados via WMI podem ser gerenciados por este servidor, e as informações de interesse disponibilizadas para um sistema PIMS, base de dados históricas ou qualquer aplicação compatível com o OPC-DA.

Com a utilização efetiva desta ferramenta monitorando redes de automação, o benefício mais notável seria o gerenciamento em tempo real da rede permitindo ao administrador da mesma agir de forma pró-ativa, evitando problemas, ou reduzindo o tempo de retomada da operação normal, em casos de:

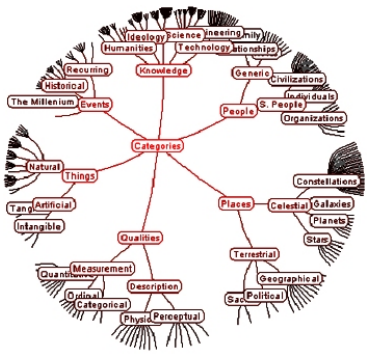
- Defeitos na infra-estrutura física da rede, como quebra de cabos ou problemas de conexão devido a falha de equipamentos.
- Comprometimento da banda de comunicação devido a excesso de tráfego oriundo de, por exemplo, instrumentação, dispositivos inteligentes ou de defeitos em equipamentos de campo.
- Degradação de desempenho devido a gargalos de comunicação de dispositivos como CLPs ou servidores de sistemas SCADA.

2.3 Métodos de Visualização de Estruturas Hierárquicas de Informação

2.3.1 Classificações

Para facilitar a comparação entre métodos de visualização diferentes, os mesmos, como apresentado por Schulz e Schumann (2006), são divididos, geralmente, em dois: os que apresentam estruturas hierárquicas e os destinados a apresentar classes de grafos mais gerais. Os métodos utilizados para apresentar estruturas hierárquicas possuem uma segunda subdivisão que os separam em métodos explícitos e implícitos.

Schulz e Schumann (2006) apresentam outras subdivisões nos métodos de visualização de estruturas hierárquicas, no entanto, é suficiente para os objetivos deste trabalho conhecer apenas as classificações já mencionadas, uma vez que o método atualmente utilizado pelo XRatel, como pode ser observado na figura 1 letra (b), é um método de visualização de estruturas hierárquicas classificadas como explícitas, e o método que será desenvolvido neste trabalho, como pode ser observado na figura 2 letra (a), é um método de visualização de estruturas hierárquicas classificado como implícito.

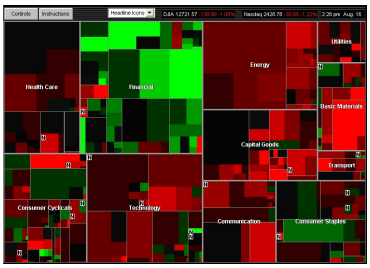


(a) Uso de Hiperbolic Tree para visualização de uma estrutura hierárquica. (HEER, 2007)

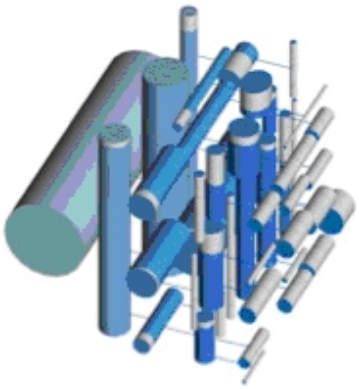
(b) Uso de árvore para visualização de ativos em rede. (XRATEL SOFTWARE, 2007)

Figura 1: Métodos de visualização explícitos

Os métodos explícitos, apresentados na figura (1), são classificados dessa forma por mostrarem as ligações entre os elementos da hierarquia na forma tradicional, em que os



(a) Uso de treemaps para visualização do desempenho de ações no mercado financeiro mundial. (SMARTMONEY.COM, 2007)



(b) Uso de Beamtrees para visualização de uma estrutura hierárquica. (SHAPLEY, 2007)

Figura 2: Métodos de visualização implícitos

nós são ligados por linhas aos seus pais e filhos. Já o método implícito, mostrado na figura (2), utiliza uma grande variedade de elementos como caixas, linhas ou círculos e as relações hierárquicas entre eles são representadas por meio das relações espaciais existentes (interseção, inclusão, adjacência,etc.).

Todas essas classificações apresentam vantagens e desvantagens e a escolha de qual método deverá ser utilizado depende dos objetivos de cada usuário, além de outras restrições que serão apresentadas a seguir.

2.3.2 Restrições na Escolha da Ferramenta de Visualização de Dados

Tarefas como: reunir, procurar, visualizar, filtrar e armazenar, quando relacionadas à recuperação de informação, assumem hoje relevância muito maior, uma vez que explorar coleção de informações torna-se uma tarefa cada vez mais difícil à medida que o volume de dados aumenta.

Essa importância levou pesquisadores da linguagem visual e designers gráficos a tentarem formalizar como se dá a pesquisa por informações. Com base nesses estudos, Shneiderman (1996) propôs o que ele chamou de "*Visual Information Seeking Mantra*".

"Overview first, zoom and filter, then details-on-demand"

Segundo o autor, uma pessoa interessada em analisar um conjunto de informações ou em buscar uma informação específica em meio a uma grande quantidade de dados, passa geralmente por três etapas básicas: primeiro, o leitor ou pesquisador dá uma olhada mais geral no conjunto de informações como um todo, visando delimitar os limites de sua busca, estabelecer relações e fazer comparações entre os dados. A segunda etapa consiste em focar suas atenções em um conjunto mais restrito de dados obtidos através da eliminação de dados desnecessários. A terceira etapa é a fase em que a pessoa já encontrou alguma coisa que satisfaz seu critério de busca, e a partir daí, somente quando achar conveniente, busca mais detalhes daquela informação previamente selecionada.

Shneiderman (1996), com o objetivo de desenvolver ferramentas de visualização e busca de informações que obtivessem sucesso comercial e fossem realmente orientadas à tomada de decisão, propôs ainda que tais ferramentas deveriam ser desenvolvidas de acordo com o tipo de visualização dos dados (uni-, bi-, tri-dimensional, temporal e hierárquicos, entre outros) e com a tarefa a qual um usuário da ferramenta estaria interessado em realizar como: obter uma visão geral, selecionar dados, obter detalhes de uma informação, ver relações entre dados, entre outras tarefas.

Seguindo as constatações feitas por Shneiderman (1996), Schulz e Schumann (2006), anos mais tarde, também apresentaram critérios importantes a serem observados na hora do desenvolvimento ou da escolha da ferramenta a ser utilizada e, com base em seus estudos, será exposto restrições de usuário e de tipo de dados que devem ser consideradas para obter uma ferramenta realmente útil.

As restrições de usuário buscam nortear as decisões de implementação ou de escolha da ferramenta que maximizariam a utilidade da mesma, tendo em vista as tarefas desempenhadas pelo usuário, a forma como a informação é apresentada e a quantidade de informação disponibilizada.

Nesse contexto, a bagagem teórica de um usuário e o tipo de informação que esta bagagem permite ao usuário extrair, por meio da integração com o método, deve ser

considerada na hora da escolha desse, uma vez que pode facilitar ou dificultar a extração destas informações.

Quanto à compatibilidade entre a interação técnica do usuário com o método de visualização, é importante expor que a forma de visualização tem que ser adequada à ação que será tomada com base nas informações extraídas, assegurando um alto grau de usabilidade para a ferramenta. A adequação do método, por meio de simplificações no *layout* do gráfico, às restrições de tempo que o usuário possui, é de extrema importância. O usuário tem que ser capaz de obter uma visão global da massa de dados, encontrar a informação que precisa e tomar a decisão correta em tempo hábil.

A compatibilidade entre diferentes técnicas de visualização pode ser obtida, fazendo uso de uma estrutura ou forma de visualização associada a outros métodos. No caso dos treemaps (um tipo implícito de visualização bidimensional de estruturas) a quantidade de informação textual é bastante limitada, no entanto, é comum utilizar listas, consideradas um tipo de visualização unidimensional (SHNEIDERMAN, 1996), para apresentar tais informações sob demanda.

Enquanto as restrições de usuário retiram da gama de possibilidades métodos que não são apropriados, devido a características do usuário, eles não levam em consideração a natureza dos dados a serem visualizados. Sendo assim, existem mais algumas restrições que devem ser consideradas.

No que diz respeito ao tipo de dados e a frequência de atualização dos mesmos, pode -se afirmar que existem algumas informações que podem ser completamente reunidas e armazenadas antes do início da análise, já em outros casos, a informação muda constantemente e tais mudanças devem ser passadas para o usuário o mais rápido possível. Tipos diferentes de informação exigem características diferentes do método de visualização. Informações voláteis exigem métodos estáveis que, ao reorganizar os dados, permitam que os mesmos não sejam expostos de uma forma muito diferente da forma exposta anteriormente (ENGDAHL, 2005), e que possam ser redesenhados rapidamente. Já informações que

não mudam de forma considerável, podem ser representadas por métodos pouco estáveis de visualização que enfatizem uma visão mais geral da estrutura, permitindo ao usuário formar uma imagem mental mais consolidada, facilitando o processo de análise.

2.3.3 O Uso de Árvores para a Visualização de Estruturas Hierárquicas

A visualização de estruturas hierárquicas em árvores é um método bastante comum de visualização de informações, e talvez por esse motivo, tenha sido escolhido como o método atualmente utilizado pelo XRatel. Tais estruturas são coleções de itens em que a raiz é colocada no topo da área destinada à visualização, e os nós filhos, aqueles situados hierarquicamente abaixo dos pais, são ligados a estes por meio de linhas.

O fato de as árvores serem métodos explícitos de visualização faz com que as mesmas sejam bastante eficientes na apresentação da estrutura hierárquica propriamente dita, dando poucas informações referentes ao conteúdo das folhas. Esta característica é um dos pontos fortes deste método e justifica, na maioria das vezes, a utilização do mesmo.

No entanto, dispor informações em árvores se torna praticamente insustentável quando a quantidade de dados é muito grande. Alguns problemas decorrentes deste fato, considerando a visualização da estrutura em uma tela de computador, podem ser observados em diversas situações práticas e representa um entrave à utilização do método em algumas aplicações.

Quando as estruturas são muito complexas, com vários níveis hierárquicos, e se deseja expandir todos os nós da hierarquia, a árvore pode se tornar muito grande, não cabendo na tela destinada à sua visualização, sendo assim, torna-se necessário a utilização de barras de rolagem para a visualização da estrutura como um todo. No entanto, tal artifício faz com que o usuário perca constantemente a orientação da hierarquia, que é, como citado anteriormente, uma das principais características que justificam a utilização deste método de visualização.

Outro problema bastante comum dos métodos explícitos, como as árvores, é o ocultamento de informações estruturais, quando se tem uma quantidade muito grande de informações. Este problema é conhecido como "*screen bottleneck*", e ocorre quando a quantidade de nós para se mostrar é tão grande que excede o número de pixels disponíveis na tela.

Como citado anteriormente, apenas a relação hierárquica é mostrada. Informações relacionadas ao conteúdo de cada nó, que permitam uma análise mais proveitosa dos dados, não fica disponível de forma imediata, sendo que, para isso, o usuário precisa, por exemplo, movimentar a barra de rolagem ou abrir e fechar nós constantemente para visualização dos filhos. Esse excesso de procedimentos para visualização da informação torna bastante difícil, para o usuário, formar uma imagem mental da totalidade da estrutura.

Nas árvores, menos de 50% do espaço é utilizado para apresentar a estrutura, sendo que a maior parte é ocupada pelo fundo de tela sem nenhuma informação.

No XRatel, a utilização de uma estrutura hierárquica foi proposta com o simples intuito de permitir uma melhor organização das máquinas dentro do software de gerenciamento. Um bom exemplo de como a estrutura hierárquica poderia ser montada dentro do XRatel seria o apresentado na figura (3) em que esta organização nada mais é do que a representação da hierarquia departamental da empresa que possui a rede.

Esta empresa fictícia apresentada na figura (3) é composta por diversos setores: o setor administrativo, o de finanças, o de recursos humanos (RH) e o setor de produção. Além desses setores, o administrador da rede julgou conveniente agrupar os servidores da empresa e as impressoras em grupos próprios. O fato de o administrador da rede ter como agrupar da forma que achar conveniente as máquinas que serão gerenciadas, evidenciam a importância secundária da estrutura da árvore e demonstram que a rede poderia ser organizada de diversas outras formas sem prejuízo para o objetivo final de gerenciamento da rede.

Como se pode observar, a estrutura hierárquica pouco ou nada importa para o objetivo

final de gerenciamento das máquinas pertencentes à rede desta empresa. Com exceção do fator organização, a estrutura hierárquica não tem grande importância para o usuário. São as folhas da árvore (máquinas) que contêm as informações como saúde do equipamento e relevância econômica, que são realmente importantes.

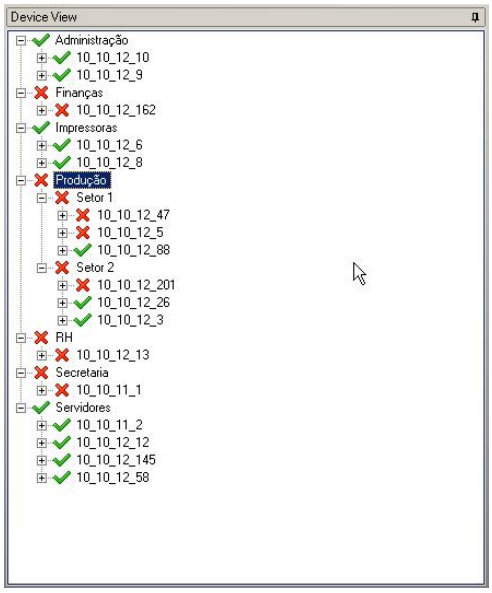


Figura 3: Estrutura hierárquica utilizada para representar os ativos de uma empresa fictícia

2.3.4 O Uso de Treemaps para a Visualização de Estruturas Hierárquicas

Com base nos problemas expostos, Johnson e Shneiderman (1991) e Shneiderman (1992) propuseram uma nova forma de visualização de estruturas que fornecesse uma melhor representação dos dados hierárquicos.

Os treemaps foi o método de visualização desenvolvido por eles. As soluções propostas visavam, basicamente, apresentar mais informações de forma explícita e otimizar a utilização do espaço destinado à apresentação das informações, uma vez que com o aumento da quantidade de dados a serem visualizados, a área utilizada para apresentar tais informações passa a ser um recurso escasso.

O método proposto obtém um melhor aproveitamento do espaço disponível, utilizando

100% da área para mapear, completamente, a estrutura dentro de uma região retangular. Dessa forma, os treemaps permitem apresentar estruturas maiores em um espaço menor sem perder qualidade estética, além de disponibilizar uma visão geral de toda a hierarquia não incorrendo em trabalhos extras com navegação.

Como resposta ao problema da pequena quantidade de informações apresentadas de forma imediata pelas árvores, os treemaps fornecem, além da estrutura hierárquica, a apresentação de forma imediata de dados relacionados ao conteúdo de cada nó, permitindo ao usuário estabelecer comparações entre os dados, fornecendo, desta forma, uma ferramenta orientada à tomada de decisões.

Segundo os idealizadores dos treemaps, a representação espacial, utilizando imagens, em vez de utilizar texto para informar atributos, é outra qualidade relevante, uma vez que pessoas têm mais facilidade para relacionar e extrair informações de figuras do que de textos.

As soluções de implementação encontradas pelos idealizadores dos treemaps são, ao mesmo tempo, simples e eficientes.

Nos Treemaps, um peso (tamanho de um arquivo, importância econômica de uma máquina, etc.) determinará o tamanho de cada nó na estrutura, e deverá ser atribuído a cada um em um estágio anterior ao da construção do treemap. Muito provavelmente este atributo não mudará após a definição inicial.

Outra solução proposta foi que a cor de preenchimento de cada nó deve ser utilizada para representar uma importante característica como desempenho de uma malha de controle, nível de utilização de uma máquina, etc. Apresentando estas características desta forma, os treemaps permitirão tomadas de decisões rápidas e precisas. Quanto à taxa de atualização deste atributo, diferentemente do tamanho de cada nó, a cor deve acompanhar em tempo real a característica que representa.

Uma das soluções encontradas para evitar o excesso de informação apresentada de forma desnecessária foi fazer o uso de *pop-ups* quando o mouse é passado sobre a região

representativa do nó de interesse para apresentar as informações detalhadas de cada nó. Isso evita a poluição visual da ferramenta.

As informações estruturais são apresentadas de forma implícita, colocando os filhos dentro das regiões dos pais, permitindo um melhor aproveitamento da área.

Essa forma implícita de representar a estrutura hierárquica deu origem às seguintes propriedades observadas por Johnson e Shneiderman (1991):

1. Se o nó-1 for um ancestral de um nó-2, então o contorno do retângulo que representa o nó-1 engloba completamente ou é igual ao contorno do nó-2.
2. Os contornos de dois nós se interceptam se um nó é ancestral do outro.
3. O peso de um nó é maior ou igual à soma dos pesos de seus filhos.

Pelo apresentado nesta seção, a utilização de treemaps é bastante apropriada para representar as máquinas em rede, tendo em vista os problemas expostos ao final da seção 2.3.3. Uma representação em forma de treemaps da estrutura hierárquica apresentada na figura (3) pode ser observada na figura (4).

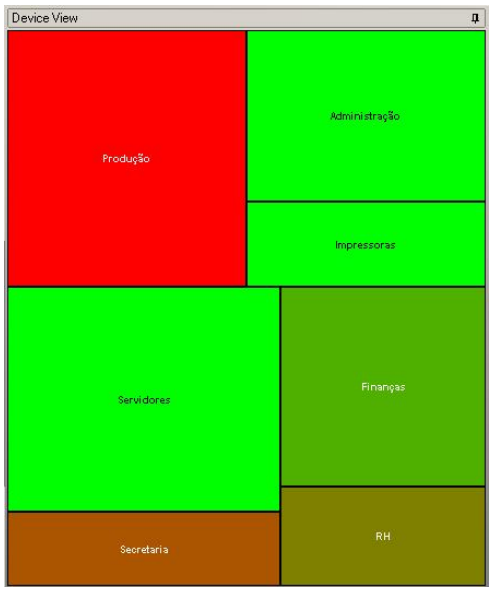


Figura 4: Representação em treemap dos ativos da empresa fictícia

Fazendo uma comparação entre as duas representações, podemos evidenciar diversas vantagens obtidas com a utilização de treemaps para representar a rede gerenciada.

Primeiramente, pelo fato de o treemap ser um método de visualização implícito, ele apresenta de forma indireta a estrutura hierárquica, procurando dar mais evidência e importância para as informações referentes a cada nó da árvore como importância econômica e saúde do equipamento.

Outra grande vantagem de se utilizar os treemaps é que no caso de a saúde do equipamento ser o atributo representado pela cor de preenchimento de cada nó, o usuário do XRatel teria uma informação muito mais rica sobre esse atributo. Na árvore exposta na figura (3), a saúde do equipamento é representada pela cor do nó na árvore. Um nó considerado em boas condições de saúde apresenta a cor verde; já um nó em más condições de saúde apresenta a cor vermelha.

A utilização desta representação com apenas dois níveis de cor impossibilita o usuário de perceber problemas em suas fases iniciais, ou seja, somente quando o nó passar da cor verde para a cor vermelha é que a pessoa responsável pela rede perceberia a existência de algum problema. Outra limitação desta forma de representação é o fato de este método não permitir uma comparação imediata entre os nós. Essa limitação fica evidente se tentarmos, por exemplo, comparar na árvore apresentada os nós referentes aos setores de Produção e Secretaria. Segundo a forma de representação utilizada na árvore da figura (3), o setor de produção está tão ruim quanto o setor da secretaria, no entanto, o setor de produção apresenta muito mais máquinas com problemas do que a secretaria.

Considerando que a cor verde claro representa o melhor estado de saúde, a cor vermelho claro o pior estado, e a cor marrom um estado intermediário e, considerando que a medida que um equipamento passa de um ótimo estado de saúde para um péssimo estado, sua cor varia do verde claro para o vermelho claro passando pelo marrom. Podemos identificar na figura (4) que o setor de finanças tem problemas apesar de a cor predominante de preenchimento ser o verde e que é evidente pela tonalidade de cor da secretaria que

este setor encontra-se em uma condição de saúde melhor que o setor de produção.

Outra vantagem da utilização de treemaps é que a pessoa responsável pela rede tem, de imediato, a informação da importância econômica da cada um dos nós. Esta informação é importante porque, caso existam dois nós com problemas e os recursos para solucioná-los forem limitados, o usuário do XRatel poderia decidir por solucionar o problema que lhe traria maior prejuízo em detrimento de um que lhe traria um prejuízo menor. Se, por exemplo, o setor de produção estivesse com problemas e a secretaria também, a comparação da área de cada um dos dois departamentos permitiria ao gerente da rede decidir para qual dos dois ele deslocaria sua equipe de manutenção.

3 Algoritmos

A parte principal do algoritmo de construção do treemap é aquela que calcula a posição e a forma dos retângulos. Como citado anteriormente, existem diversos algoritmos que apresentam soluções diferentes, sendo que uma solução apresenta vantagens e desvantagens em relação às outras. São estas variações e as vantagens e desvantagens de cada uma que serão apresentadas neste capítulo.

Apesar da parte variável, os algoritmos de construção de treemaps seguem basicamente a estrutura apresentada no código 1:

```
1  desenhaNos(no[] Nos, regiaoTotal)
2     {
3          Para cada no em Nos faça
4                  regiaoNo = calculaRegiao(Parametros);
5                  regiaoTotal = regiaoTotal - regiaoNo;
6                  Se (no tem filho)
7                          desenhaNos(noFilho[], regiaoNo);
8                  Fim Se;
9          Fim Loop;
10    }
```

Código 1: Pseudo-código de desenho de Treemaps

Este algorítmo recebe como parâmetro um conjunto de nós em um vetor e uma variável que fornece os limites destinados à construção do treemap. Os nós são processados individualmente a cada loop do algoritmo. Se um nó específico possuir nós filhos, o algoritmo é chamado recursivamente para os filhos de cada nó.

A quarta linha é a parte variável do algoritmo. Nessa linha, é feita a chamada da função que irá calcular os contornos do nó do loop corrente com base nos parâmetros

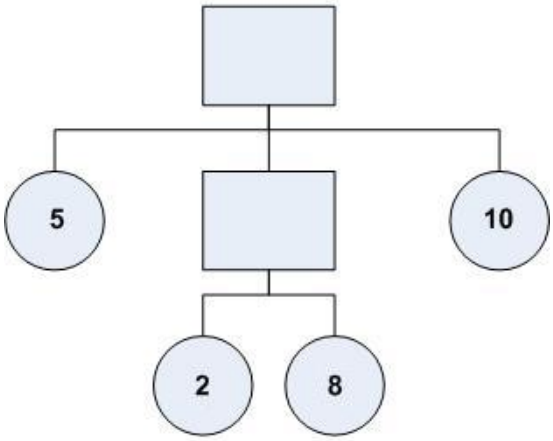


Figura 5: Exemplo de árvore a ser transformada em treemap

passados. Após o cálculo, esses contornos são atribuídos ao nó em questão e a área total destinada ao treemap é recalculada.

A árvore apresentada na figura 5 servirá como exemplo de árvore a ser transformada em treemap pelos algoritmos apresentados neste capítulo.

3.1 *Slice and Dice*

Esse foi o primeiro algoritmo de cálculo da área e da posição dos nós na construção de treemaps proposto por Johnson e Shneiderman (1991). Partindo do algoritmo apresentado por eles, é apresentado a seguir um pseudo-código.

```
1 SliceAndDice(raiz, filho[], lowerLeft[], upperRight[], eixo)
2         largura = upperRight[eixo]-lowerLeft[eixo];
3         Para i = 1 até tamanho(filhos) faça
4                 upperRight[eixo] = lowerLeft[eixo] +
5                 (tamanho(filho[i])/tamanho(raiz))*largura;
6                 SliceAndDice(i, filho[i], lowerLeft, upperRight, 1-eixo);
7                 lowerLeft[eixo] = upperRight[eixo];
8         Fim Para;
9 Fim SliceAndDice;
```

Código 2: Algoritmo *Slice and Dice*

As variáveis utilizadas possuem os seguintes significados:

Raiz: é um apontador para a raiz da árvore ou subárvore.

Filho: é um vetor que contém apontadores para os filhos do nó a ser desenhado.

LowerLeft: é um vetor de duas posições contendo as coordenadas X e Y, nesta ordem, do canto inferior esquerdo do nó a ser desenhado.

UpperRight: é um vetor de duas posições contendo as coordenadas X e Y, nesta ordem, do canto superior direito do nó a ser desenhado.

Eixo: pode assumir o valor zero ou um para indicar se os cortes serão feitos na vertical ou na horizontal, respectivamente.

Como pode ser observado no pseudo-código 2, assumindo que a variável "eixo" tenha sido inicializada com o valor zero, inicialmente o algoritmo obtém a dimensão referente ao eixo X da área destinada à criação do treemap, fazendo a subtração da coordenada X do canto superior direito ("upperRight") menos a coordenada X do canto inferior esquerdo ("lowerLeft"), e divide esta dimensão entre os filhos do nó raiz passado como parâmetro, respeitando a proporção de tamanho entre eles. Sendo assim, considerando a largura do retângulo passado como parâmetro, como sendo um valor de 100 unidades, o algoritmo coloca o primeiro nó ocupando toda a dimensão do eixo Y do retângulo e define a largura do mesmo, atribuindo um valor de coordenada X do canto superior direito como sendo 20 unidades maior que a coordenada X do canto inferior esquerdo. Como este nó não possui filhos, o algoritmo segue para o processamento do próximo nó da estrutura. A coordenada X do canto inferior esquerdo é atualizado para o valor da coordenada X do canto superior direito e desta forma, o próximo nó será acrescentado imediatamente após o primeiro. Como o peso desse segundo nó é 10, o dobro do peso do nó anterior (5), é de se esperar que a dimensão X deste seja o dobro da dimensão X do anterior, ou seja, 40.

Como esse segundo nó apresenta filhos, o algoritmo é chamado recursivamente, no entanto, o retângulo passado como parâmetro para a construção do treemap é o retângulo que acabou de ser gerado para representar esse segundo nó. Outra alteração é o valor da variável "eixo" que passa a ter o valor de *1-eixo*. Dessa forma, a direção dos cortes desses filhos serão feitos em uma direção diferente dos cortes que eram feitos para os pais.

Sendo assim, considerando como 100 unidades a dimensão Y desse segundo nó, temos que o primeiro filho teria uma dimensão Y de 20 unidades e o segundo uma dimensão Y de 80, ambos ocupando toda a coordenada X do nó pai. Caso esses filhos tivessem filhos, o algoritmo seria chamado recursivamente passando como retângulo, o retângulo que possui esses filhos e realizando cortes na direção utilizada inicialmente, ou seja, a direção dos nós avós destes que estão sendo criados. O algoritmo continua até que todos os filhos do nó raiz da árvore tenham sido representados.

As dimensões dos retângulos que vão sendo criados no passo a passo de execução desse algoritmo, quando é passado para o mesmo, o nó raiz da árvore apresentada na figura 5, podem ser visualizadas na figura 6.

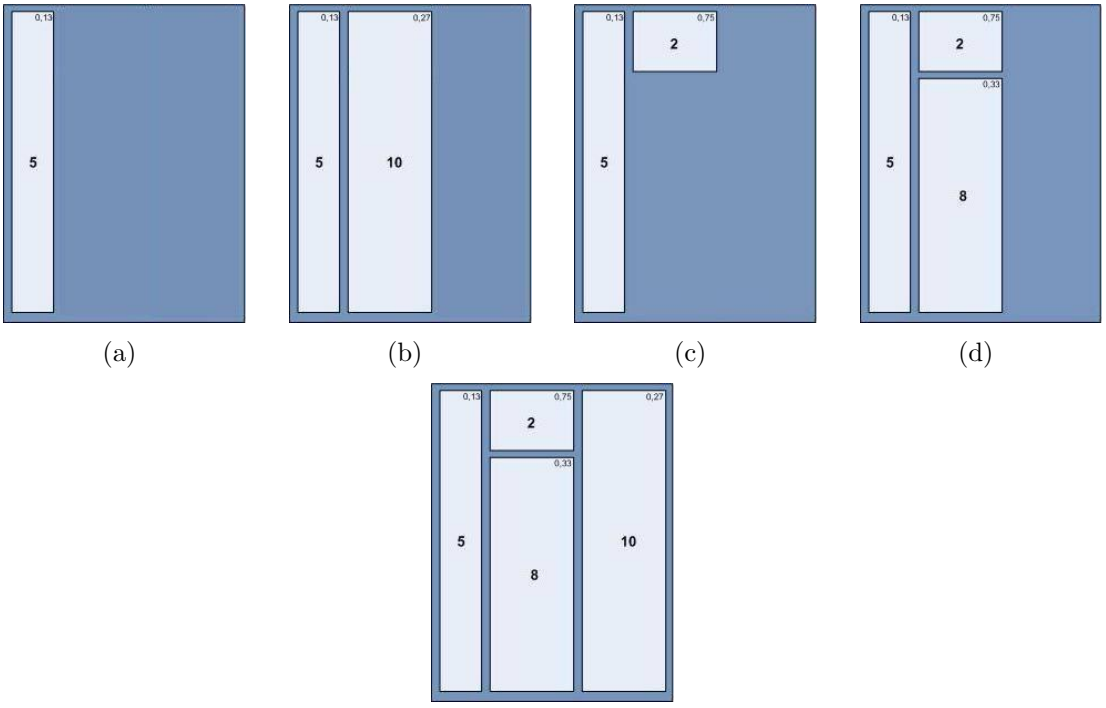


Figura 6: Passo a passo de execução do algoritmo *Slice and Dice*

Uma análise do resultado apresentado por esse algoritmo permite-nos classificá-lo como ordenado (ENGDAHL, 2005) por preservar a ordem natural do conjunto de dados. Essa característica é importante para um conjunto de dados que muda constantemente, uma vez que quando um nó é acrescentado ou retirado, a disposição geral dos nós remanescentes não é alterada de forma dramática, propiciando ao usuário maior facilidade na

hora de procurar por nós cuja localização ele já conhecia previamente. A característica negativa desse algoritmo é que ele apresenta como solução um treemap que possui retângulos cuja razão entre a largura e o comprimento é muito grande. Esta característica é considerada ruim pelo fato de que retângulos muito finos e compridos são de difícil seleção por parte do usuário, tornam a comparação entre nós mais difícil, além de tornar o algoritmo menos útil quando utilizado para uma massa de dados muito grande.

3.2 *Squarified*

Baseado no código publicado por Engdahl (2005), a solução apresentada no pseudocódigo 3 foi proposta com o intuito de resolver o problema do algoritmo *Slice And Dice* apresentado na seção 3.1 de produzir retângulos com a largura e o comprimento muito desproporcionais, causando os problemas já mencionados. O algoritmo apresentado nessa seção busca atender da melhor forma possível o critério:

$$\forall r \in R, \frac{max(r.largura,\ r.altura)}{min(r.largura,\ r.\ \ altura)} \approx 1$$

que sugere que o valor máximo entre a largura e a altura dividido pelo valor mínimo entre a largura e a altura de um conjunto de retângulos que compõem o treemap seja o mais próximo da unidade quanto possível.

A aproximação proposta deve ser bastante flexível para não tornar o esforço computacional muito elevado.

No algoritmo 3 as variáveis e funções utilizadas possuem os seguintes significados:

Ordena(): esta função retorna o vetor de nós, passado como parâmetro, ordenado em ordem decrescente de tamanho.

Filho(): retorna um vetor contendo os atributos referentes à área de todos os filhos do nó passado como parâmetro.

Linha[]: é o vetor que armazena os nós que serão desenhados na próxima linha do

treemap.

Worst(): é a função que retorna o maior valor da razão entre a largura e a altura de uma lista de retângulos.

Nodes[]: é uma pilha contendo os filhos da raiz de uma dada árvore.

Desempilha(): função que retorna o primeiro elemento da pilha de nós.

DesenhaLinha(): função que acrescenta uma nova linha de nós à área destinada à confecção do treemap.

```
1  Squarify(nodes[])
2  {
3          Ordena(nodes[])
4          Enquanto Tamanho(nodes[]) > 0 faça
5                  no = Desempilha(nodes[])
6                  Se worst(Linha[] + no) < worst(Linha[]) então
7                          Linha[] = Linha[] + no
8
9                  Senão
10                         {
11                         desenhaLinha(Linha[])
12                         Limpa(Linha[])
13                         Linha[] = Linha[] + no
14                         }
15                 fim Se
16         fim Enquanto
17         Para cada no em nodes[] faça
18                 Squarify(Filhos(no))
19 }
```

Código 3: Algoritmo *Squarified*

A ordem em que os nós são adicionados é bastante importante, por isso eles são ordenados do maior para o menor, fazendo com que os nós maiores sejam adicionados primeiramente.

Com a pilha de nós ordenada, o algoritmo vai desempilhando um nó de cada vez e acrescentando-os a um vetor que armazena as dimensões de cada nó, formando as linhas que serão acrescsentados ao treemap. Em uma parte não transcrita do pseudo-código, o algoritmo define se as linhas serão verticais ou horizontais, de acordo com o tamanho da altura ou da largura da área destinada à disposição dos mesmos. Se a largura da área remanescente é maior do que a altura, a linha é adicionada verticalmente, caso contrário, a linha é adicionada horizontalmente.

Para o exemplo apresentado na figura 7, considerando que a coordenada X (largura) tem uma dimensão de 100 unidades e a coordenada Y (altura), uma dimensão de 150 unidades, o algorítmo decidiria por acrescentar as linhas horizontalmente.

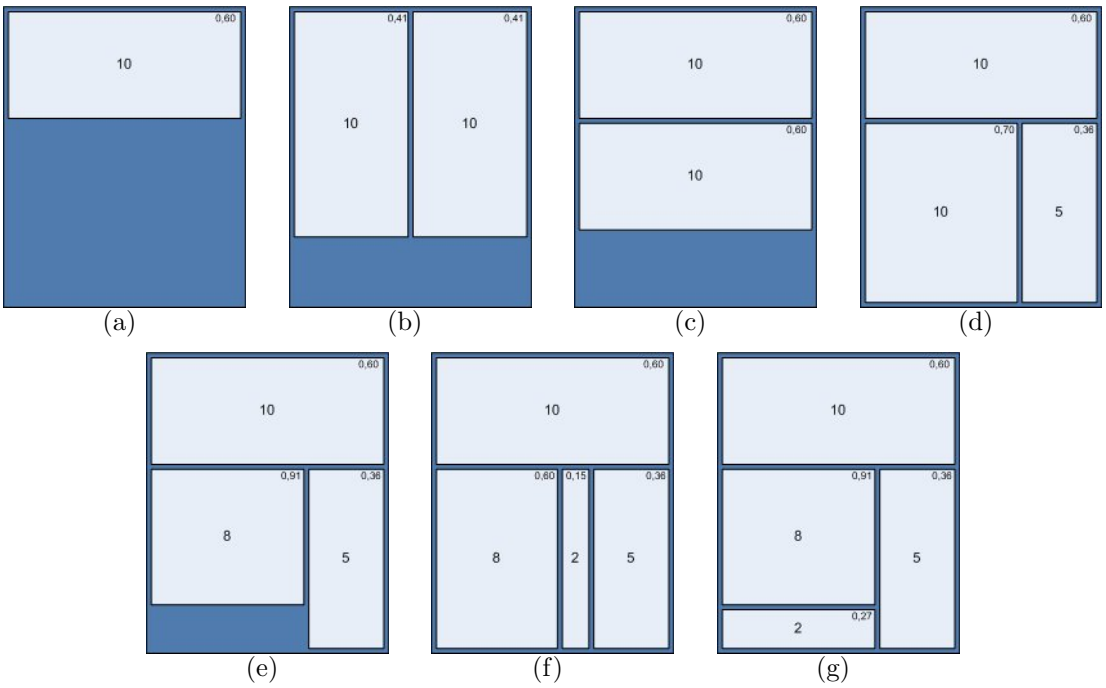


Figura 7: Passo a passo de execução do algoritmo *Squarify*

Quando cada nó é desempilhado, outra escolha tem que ser feita: ou o nó é adicionado à linha corrente ou a linha corrente é considerada completa e uma nova linha é criada com esse novo retângulo. A decisão depende apenas de saber se ao adicionar o nó em processamento, se ele vai melhorar ou não a forma dos outros retângulos pertecentes a linha corrente.

No caso do primeiro nó de uma linha (linha definida neste caso para ser horizontal), o mesmo é acrescentado a princípio ocupando toda a dimensão X da área destinada, a dimensão Y é obtida com base na dimensão da área destinada à confecção do treemap, e na área do nó em relação aos demais nós da estrutura. Para o primeiro nó, a conta seria:

Área destinada ao treemap (a):

$$a = 150 \times 100 = 15000$$

Área destinada ao primeiro nó (*a1*):

$$\frac{15000}{10+10+5} = 600$$

$$a1 = 600 \times 10 = 6000$$

Coordenada Y (altura) do nó(*Y1*):

$$Y1 = \frac{6000}{100} = 60$$

Para acrescentar o novo nó, é preciso redistribuir a coordenada X destinada ao treemap entre os nós (neste momento dois) que comporão a linha. Essa distribuição tem que ser proporcional às áreas dos nós que comporão a linha. Como o segundo nó tem área igual ao primeiro, a coordenada X é dividida igualmente para os dois nós, ou seja, 50 unidades para cada nó. Feita esta divisão, é necessário recalcular a altura dos nós da linha.

Definidas as dimensões do nó acrescentado, é feita uma avaliação para saber se ao acrescentar o último nó na linha houve uma piora na aparência da mesma gerando retângulos com uma razão entre largura e altura pior do que se tinha anteriormente. Na figura 7 esta razão é apresentada dentro de cada nó no canto superior direito, e comparando os valores desta razão do primeiro nó nas situações apresentadas pelas figuras 7 letras (a) e (b), ficou evidente que acrescentar o último nó a essa linha piorou a razão entre largura e altura, o que significa que o nó deverá ser acrescentado a uma nova linha como é apresentado na figura 7 letra (c).

Este procedimento é feito para todos os nós da árvore e, em seguida, o algoritmo é chamado recursivamente para os filhos dos nós, sempre realizando uma ordenação prévia, e tendo as mesmas preocupações mencionadas na representação dos pais.

Pelo fato deste algoritmo necessitar de uma ordenação prévia dos nós, ele tem um custo computacional mais elevado que o algoritmo apresentado na seção 3.1.

Além do mais, a ordem natural do conjunto de dados não é preservada o que é pouco

aconselhável, como dito anteriormente, para um conjunto de dados muito dinâmico.

No entanto, este algoritmo procura obter um conjunto de retângulos com a razão entre a largura e o comprimento dos mesmos o mais próximo possível de 1 (um).

3.3 *Split*

O algoritmo apresentado na seção 3.1, apesar de manter a ordem original dos nós e possuir uma boa estabilidade, como já foi dito, possui o grande inconveniente de se tornar pouco eficiente por produzir retângulos com a largura e a altura muito desproporcionais quando o número de nós aumenta muito. Já o algoritmo apresentado na seção 3.2, apesar de apresentar uma razão entre a largura e a altura bem próxima da unidade, realiza uma ordenação prévia dos nós gerando treemaps cuja disposição dos retângulos varia muito de um treemap para outro à medida que novos nós vão sendo acrescentados.

Engdahl (2005), tentando uma solução que atendesse da melhor forma possível os dois critérios, propôs o algoritmo *Split* que mantém a ordem original dos nós e produz, na maioria das vezes, treemaps cujos retângulos apresentam uma razão entre largura e altura melhor que a razão geralmente obtida pelo algoritmo *Slice and Dice*.

O algoritmo recebe como entrada uma lista ordenada (L) de nós e um retângulo (R) onde os nós serão distribuídos. Além disso, define-se como peso da lista *w(L)* o valor da soma dos tamanhos de todos os seus elementos.

O algoritmo segue um processo recursivo em que L é dividido em duas metades, *L1* e *L2*, fazendo com que *w(L1)* seja o mais próximo possível de *w(L2)*. Nessa divisão, pode ser que fiquem mais nós em uma metade que em outra, no entanto, a ordem dos mesmos é preservada. Nessa divisão teremos, então, duas listas ordenadas, sendo que os índices dos nós de *L1* são menores que os índices dos nós de *L2*.

```
1  Split(List itens, Retangulo r)
2          {
3          If (itens.length == 0)
4          return;
5          If (itens.length ==1)
6                  {
7                  itens.bounds =r;
8                  Split(itens.children, r);
9                  }
10         List l1, l2;
11         Retangulo r1, r2;
12         Double halfSize = w(itens)/2;
13         double w1 = 0, tmp = 0;
14
15         For All itens
16                 {
17                 Item front = itens[front];
18                 tmp = w1 + front.size;
19                 If (abs(halfSize-tmp) > abs(halfSize-w1))
20                         break;
21                 l1.enqueue(itens.dequeue);
22                 w1 = tmp;
23                 }
24         End For
25
26         l2 = itens;
27         r1 = new Retangulo(r.x, r.y, r.width*w(l1)/w(l1+l2), r.height);
28         If (r1.height) > (r1.width)
29                 r1.width = r1.height
30                 r2 = new Retangulo(r.x, r.y + r1.width, r.heigth, r.width -
31                 r1.width);
32         Else
33                 r2 = new Retangulo(r.x + r1.width, r.y, r.width - r1.width,
34                 r.height);
35         Split(l1,r1);
36         Split(l2,r2);
37         }
```

Código 4: Algoritmo *Split*

Define-se *a(R)* como sendo a área do retângulo R. O retângulo R é dividido horizontalmente ou verticalmente, dependendo do fato de a altura ser maior que a largura ou vice-versa, obtendo assim, dois retângulos, *R1* e *R2*, cujas áreas representam o tamanho dos elementos de *L1* e *L2*. Com essa divisão, obtemos as igualdades apresentadas nas equações seguintes.

$$\frac{a(R1)}{a(R)} = \frac{w(L1)}{w(L)}, \frac{a(R2)}{a(R)} = \frac{w(L2)}{w(L)}$$

Recursivamente, como mostra o algoritmo 4, desenhamos os retângulos contidos em L1 e L2 dentro dos retângulos R1 e R2.

O algoritmo, primeiramente, separa os nós da lista passada como parâmetro *itens* em

duas listas. Essa separação é feita no primeiro *For* do algoritmo em que a cada *loop* um nó é desempilhado da lista *itens* e empilhada na lista *L1* até que o tamanho da lista *L1* seja maior que o tamanho da metade da lista *itens*. Quando essa condição é satisfeita, o que sobrou na lista passada como parâmetro é atribuído à lista *L2*.

Em seguida, o algoritmo define se o retângulo passado como parâmetro será dividido entre as listas *L1* e *L2* verticalmente ou horizontalmente, de acordo com o tamanho da altura ou da largura da área destinada ao treemap. Se a largura da área desse retângulo é maior do que a altura, a divisão será horizontal, caso contrário, a divisão será vertical.

No caso apresentado, a divisão será vertical, logo a coordenada Y do retângulo passado como parâmetro, contendo as dimensões que o treemap irá ocupar, é dividida entre as duas listas, *L1* e *L2*, mantendo a proporção entre o tamanho das duas. É de se esperar que essa divisão seja o mais igualitária possível, uma vez que as listas tendem a ter o mesmo tamanho.

Para a árvore apresentada na figura 5, a lista *L1* possuiria dois nós, totalizando um tamanho de 15 unidades, e a lista *L2* ficaria com apenas um nó, totalizando um tamanho de 10 unidades.

Considerando a área destinada ao desenho do treemap como sendo um retângulo de largura (coordenada X) de 100 unidades, e altura (coordenada Y) de 150 unidades, e considerando que a lista *L1* tem um tamanho de 15 unidades, o retângulo *R1*, que é proporcional ao tamanho da lista *L1*, ficaria com a dimensão X igual a 100 unidades e a dimensão Y igual a 90 unidades. Da mesma forma, o retângulo R2 teria a dimensão X igual a 100 unidades e a dimensão Y igual a 60 unidades.

Antes de desenhar esses retângulos, o algoritmo é chamado recursivamente, passando, dessa vez, como parâmetros, a lista *L1* e o retângulo *R1*. Sendo assim, o algoritmo vai sendo chamado recursivamente até que a lista *L1* contenha apenas um nó com suas respectivas coordenadas definidas no retângulo *R1*. Este retângulo é, então, desenhado como pode ser visto na figura 8 letra (a).

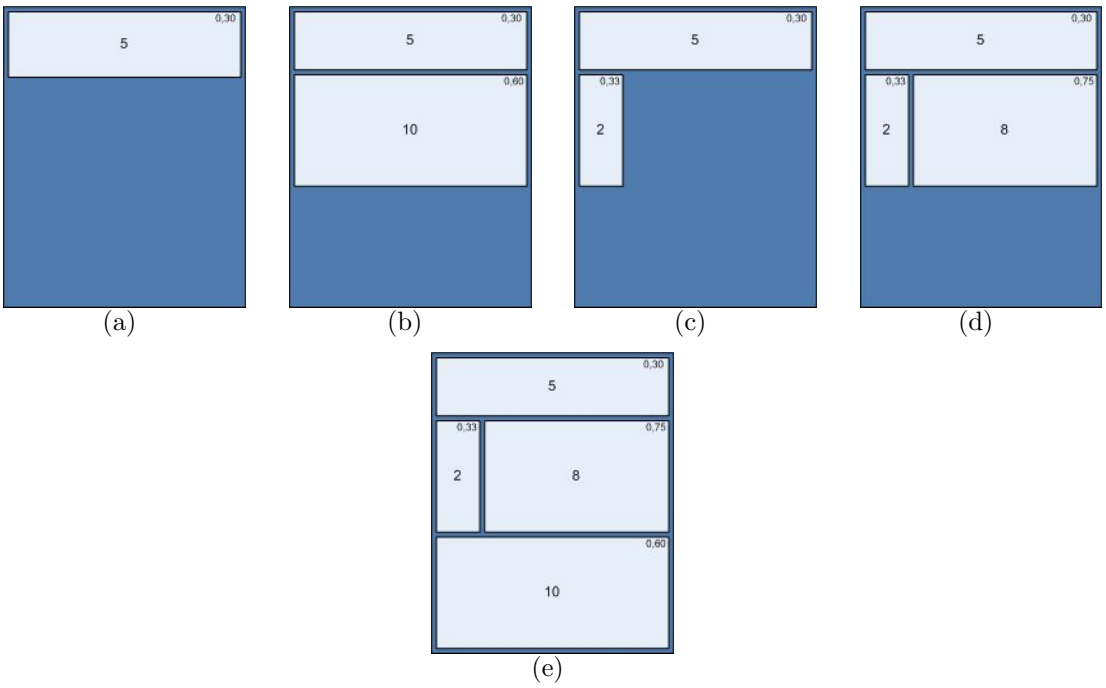


Figura 8: Passo a passo de execução do algoritmo *Split*

Caso o retângulo *R1* tenha filhos, o algoritmo é chamado recursivamente passando como parâmetros o retângulo *R1* e a lista de filhos do nó em questão.

Só após processar toda a lista *L1* da forma como foi exposto é que a lista *L2* passa a ser processada.

O passo a passo de execução do algoritmo para gerar um treemap com base na árvore apresentada na figura 5 pode ser acompanhado na figura 8. Como pode ser observado, apenas um nó é desenhado por vez.

3.4 Escolha do Algoritmo

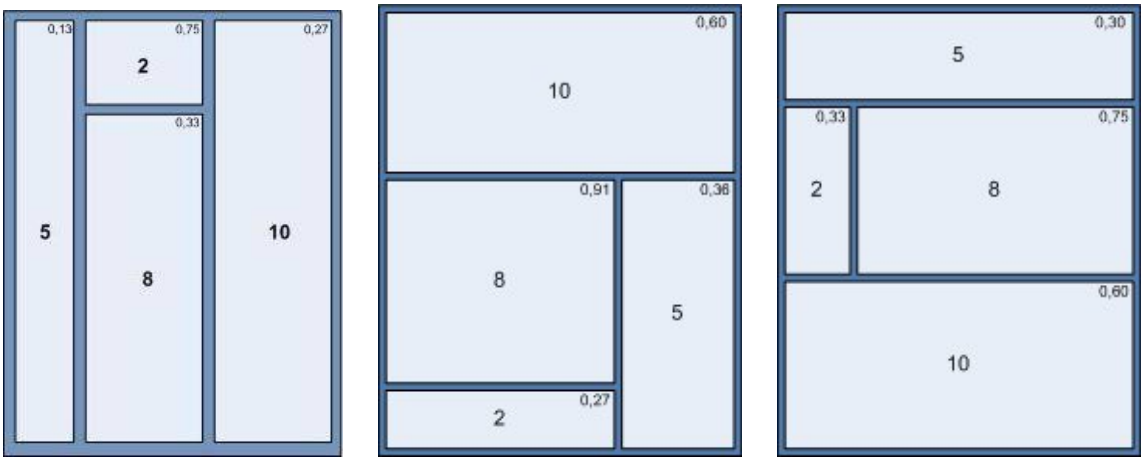
A escolha de qual algoritmo usar para a geração dos treemaps tem que ser feita com base nas características da ferramenta que pretende apresentar a estrutura hierárquica. Fazendo essa análise, é possível decidir qual algoritmo utilizar, sem a necessidade de codificar os três algoritmos.

Nesse caso, a ferramenta pretende apresentar para o usuário, a estrutura e as má-

quinas de uma rede de dispositivos a ser gerenciada. Sendo assim, as características que influenciam a escolha do algoritmo são a estabilidade e a capacidade do mesmo em representar grande quantidade de informações sem gerar um treemap que seja desconfortável para sua interação com o usuário.

A estabilidade, apesar de não ser crítica para a aplicação em questão, é uma característica desejável para o treemap que se deseja obter. Essa característica não é crítica, porque a estrutura da rede gerenciada é geralmente bastante estável e, raramente, é necessário acrescentar ou retirar máquinas dessa rede. Sendo assim, o problema da estabilidade afetaria pouco essa aplicação.

O desconforto de interação do usuário com o treemap, no caso de o mesmo estar representando uma grande quantidade de informação, seria o fato de os nós dentro do treemap ter uma largura e altura muito desproporcional, como acontece com o algoritmo *Slice and Dice*. Essa característica não é de forma alguma desejável, uma vez que é bem provável que a rede a ser gerenciada será suficientemente grande para causar esse problema.



(a) (b) (c)

Figura 9: Comparação entre os resultados obtidos com os três algoritmos para a árvore da figura 5

Sendo assim, excluindo da gama de opções o algoritmo *Slice and Dice*, a escolha fica entre o *Squarified* e o *Split*. Como o algoritmo *Split* apresenta uma razão entre largura e altura tão boa quanto o algoritmo *Squarified*, como pode ser observado na figura 9, e

o mesmo ainda apresenta uma estabilidade maior, foi decidido por utilizar esse algoritmo na geração do treemap do XRatel.

4 Implementação

4.1 Requisitos do XRatel

Uma exigência do XRatel foi que o treemap desenvolvido não substituísse a forma tradicional de visualização. Sendo assim, era preciso definir onde seria acrescentado o treemap, e como o usuário faria para acessá-lo.

A decisão foi a de utilizar a mesma área destinada à visualização existente para apresentar o treemap, e caberia ao usuário escolher qual forma de visualização ele acharia mais conveniente.

Foi decidido, juntamente com a equipe do XRatel, que as informações que seriam apresentadas de forma imediata ao responsável pelo gerenciamento da rede seriam a importância econômica, representada pela área que uma determinada máquina ocuparia no treemap, e o *Health* da máquina, representado pela cor de preenchimento do retângulo que a representa. A decisão pela primeira grandeza foi baseada nas aplicações que já utilizam treemaps como forma de visualizar estruturas hierárquicas e na experiência dos envolvidos na concepção do XRatel. Já a segunda grandeza foi uma escolha óbvia, uma vez que a mesma é a informação mais elaborada fornecida pela ferramenta de gerenciamento de ativos em rede.

O treemap deveria apresentar a estrutura da árvore exatamente como ela é, ou seja, nenhum nó da árvore deveria ser omitido e nenhum nó deveria ser acrescentado, com a finalidade de viabilizar a construção do treemap. Como a árvore existente no *Health Analyst* possui nós que representam grupos de máquinas, máquinas, e *tags* (representando

as grandezas medidas pelo *Health Analyst*), o treemap deveria permitir ao usuário atribuir valores de importância econômica para todos esses tipos de nós.

Como o XRatel foi desenvolvido em Delphi, seria necessário que o treemap fosse desenvolvido na mesma linguagem de programação. Na época do desenvolvimento do treemap, a versão do Delphi que estava sendo utilizada pela equipe do XRatel era a 7.0 e, portanto todo esse trabalho foi desenvolvido, utilizando essa mesma versão rodando sobre plataforma Windows XP em Inglês com Service Pack 2.

Era interessante também que todas as bibliotecas utilizadas na codificação do treemap fossem bibliotecas desenvolvidas pelo XRatel ou fizessem parte de bibliotecas padrões do Delphi. Sendo assim, o desenvolvimento dessa ferramenta não deveria gerar nenhum custo extra para o XRatel, no que diz respeito à compra de novas bibliotecas.

4.2 Decisões de Implementação

Definidas as grandezas a serem representadas pelo treemap, e pelo fato de a grandeza de importância econômica não estar disponível a priori no software, fez-se necessário definir o momento e a forma como essa informação seria inserida pelo usuário.

Como a maioria dos softwares de grande porte, o XRatel é uma ferramenta composta por diversos módulos que são responsáveis por fornecer todas as funcionalidades necessárias à tarefa de gerenciamento de redes. Com o desenvolvimento do treemap, apenas alguns módulos são afetados, como: os módulos de configuração da rede a ser gerenciada, denominados pela equipe do XRatel como *SNMP Config* e *PerfMon Config* e o módulo de visualização e análise da mesma denominado *Health Analyst*.

Nos módulos *SNMP Config* e *PerfMon Config* o usuário do XRatel escolhe as máquinas que serão gerenciadas e as organiza em uma árvore, da forma que julgar conveniente. Como sugerido anteriormente, essa estrutura é, na maioria das vezes, uma representação da organização setorial de uma empresa, ou a distribuição das máquinas de acordo com

a funcionalidade das mesmas. Pelo fato de a árvore ser formada com tal objetivo, é de se esperar que a estrutura da mesma não varie com muita facilidade, uma vez que a organização departamental de uma empresa ou a funcionalidade de uma máquina em uma rede não é uma coisa que varie constantemente.

Como a importância econômica de uma máquina também não é uma grandeza constantemente alterada, e por ela estar associada a uma máquina individualmente, ficou definido que essa grandeza seria informada no momento em que a máquina é escolhida para ser gerenciada.

Outra decisão, de grande relevância, referente a essa grandeza seria decidir como o usuário entraria com essa medida de importância econômica. Pelo fato de a importância econômica de uma máquina em uma determinada rede ser um valor altamente subjetivo, foi necessário deixar para o usuário a decisão de quão grande é esse valor. No entanto, permitir que esse valor fosse obtido sem qualquer parametrização poderia trair o objetivo principal do treemap que é otimizar a tomada de decisão por parte do usuário. Sendo assim, a forma como essa informação é inserida deve minimizar as possibilidades de o usuário atribuir informações inconsistentes com a realidade da rede.

A primeira forma de minimizar os inconvenientes da subjetividade do valor da importância econômica de uma máquina seria definir limites para essa grandeza. Definir como limite mínimo o valor 1 (um) e o limite máximo o valor 10 (dez) poderia trazer ao usuário da ferramenta um certo desconforto na hora de escolher se uma máquina tem importância 7 (sete) ou 8 (oito). Além do mais, na hora de comparar as duas máquinas em um treemap, poderia não ficar muito evidente qual máquina é a mais importante.

Pelo exposto, foi definido que ao usuário caberia, apenas, escolher se uma máquina tem importância econômica Alta, Normal ou Baixa. Dessa forma, os problemas da subjetividade seriam minimizados, uma vez que a pessoa responsável pelo gerenciamento da rede caso conhecesse bem as máquinas que a compõe, saberia dizer com facilidade se uma máquina tem uma importância Alta, Normal ou Baixa para uma dada rede.

Como sugerido por Shneiderman (1992), qualquer informação adicional a respeito de um nó que fosse necessária ser apresentada ao usuário deveria ser feita por meio de *pop-up* no momento em que o usuário desejasse e passasse o mouse sobre a área que representa aquele nó. A utilização de *pop-up*, portanto, além de estar totalmente de acordo com a teoria dos treemaps vinha ainda atender a uma necessidade do XRatel: por algum motivo o XRatel não disponibilizava de forma direta, no *Health Analyst*, o valor atual do *Health* das máquinas. Esse valor só estava disponível em gráficos históricos de tal forma que não permitia ao usuário saber o valor numérico exato do *Health* no instante atual.

Por esse motivo, foi decidido apresentar ao usuário na forma de *pop-up* o valor numérico exato do *Health* no momento em que o usuário passar o mouse sobre a máquina de interesse, ou no caso desse valor estar indisponível por algum problema (como falha de comunicação), apresentar o valor "??".

Outra decisão de implementação que foi necessário tomar foi quanto à profundidade da árvore que seria mostrada de imediato para o usuário, ou seja, quando o usuário olhasse para o treemap seria interessante que ele fosse capaz de ver quantos níveis da árvore? A resposta a essa pergunta foi obtida através da consulta aos membros da equipe do XRatel e através da análise da teoria que existe por trás dos Treemaps. Como apresentado anteriormente, faz parte da filosofia dos treemaps mostrar detalhes apenas sob a demanda do usuário, sendo assim, a teoria corroborou com a opinião da equipe do XRatel que preferiu apresentar de imediato apenas o primeiro nível da árvore, cabendo ao usuário a decisão de aprofundar ou não nos outros níveis.

Como mapear o valor do *Health* em uma cor foi uma escolha baseada nas próprias características do XRatel. Para as máquinas cujo o *Health* estava abaixo de um certo valor, o *Health Analyst* atribuía a cor vermelha indicando que a máquina estava precisando de uma intervenção; para as máquinas cujo *Health* estava acima de um dado valor, era atribuída a cor verde, indicando que a máquina estava em boas condições e não precisava de intervenção.

Com base nessas características, foi definido que a cor verde pura (RGB = 0,255,0) representaria o valor de *Health* igual a 100 e a cor vermelha pura (RGB = 255,0,0) representaria o valor de *Health* igual a 0. A função que definiria valores intermediários seria linear, como é apresentado na figura 10.

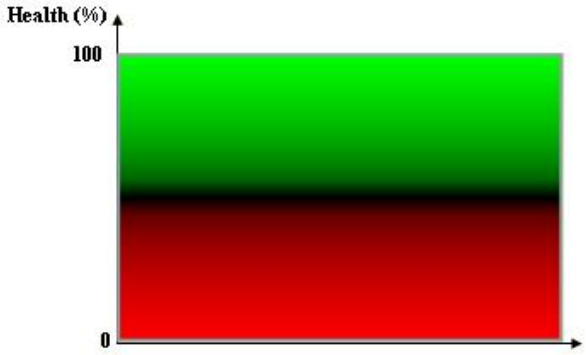


Figura 10: Função linear de mapeamento da cor do equipamento em função da saúde do mesmo

Para os casos em que o valor de *Health* estivesse indefinido por problemas de comunicação ou qualquer outro problema, a cor do nó no treemap ficaria preta (RGB = 0,0,0), não permitindo, dessa forma, que o usuário obtivesse uma informação incorreta.

A navegação do usuário na estrutura do treemap foi implementada com base nas ações do usuário com o *mouse*. Caso o usuário queira entrar em um dado nó para acessar seus filhos, é necessário que o mesmo dê um duplo clique, com o botão esquerdo do *mouse*, sobre o retângulo que representa o nó. Feito isso, um novo treemap é gerado, representando a estrutura que tem como nó raiz o nó clicado. Esse novo treemap ocupa toda a região disponível para a geração do treemap para, desta forma, apresentar ao usuário apenas a informação que tem interesse para ele neste momento.

Para voltar ao nível imediatamente acima do nó que o usuário está visualizando em um dado momento, é necessário que o mesmo dê um clique com o botão direito do mouse sobre este nó. Feito isso, um novo treemap é gerado, representando toda a estrutura que tem como raiz o avô do nó clicado.

Um clique simples com o botão esquerdo do mouse apenas seleciona o nó de interesse, sem no entanto, "andar" na estrutura da árvore. Esse comando é necessário para que o

usuário do *Health Analyst* possa escolher qual nó ele quer visualizar no gráfico histórico.

Se o interesse do usuário é, apenas, saber qual o valor atual do *Health* de um dado equipamento, é necessário somente passar o mouse sobre o nó de interesse que a borda deste retângulo muda de cor e um *pop-up* apresenta o valor do *Health*.

5 *Resultados*

5.1 Mudanças Visíveis para o Usuário do XRatel

Para a geração do treemap, a informação de "Importância Econômica" que anteriormente não era fornecida pelo usuário passou a ser necessária e, como exposto na Seção 4.2, foi definido que essa informação deveria ser adicionada no momento que uma dada máquina é escolhida para ser gerenciada. Sendo assim, a figura 11 mostra o local e a maneira como o usuário deverá entrar com essa informação.

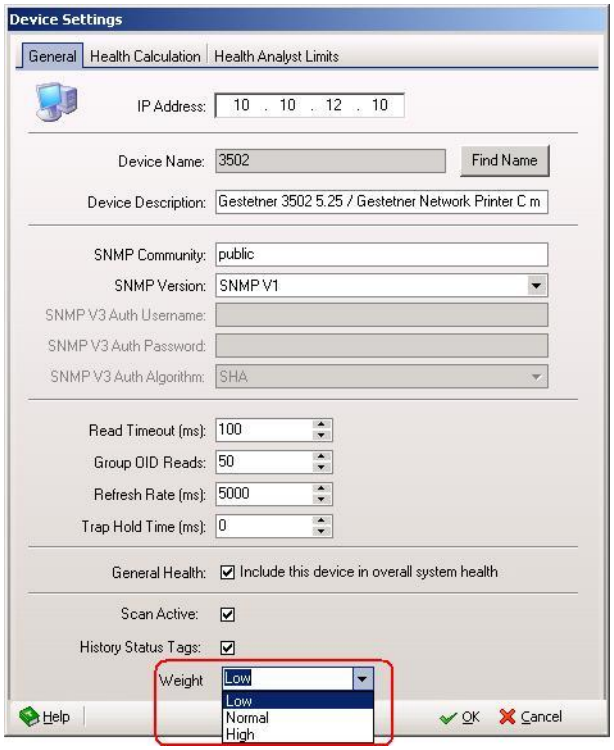


Figura 11: Janela de configuração da máquina escolhida pelo usuário para ser gerenciada

No detalhe, são observadas as opções de "Peso" (Importância Econômica) que o usuário tem para caracterizar a máquina.

As interfaces para acrescentar a importância econômica dos grupos de máquinas e dos *tags* são apresentadas nas figuras 12 e 13, respectivamente.

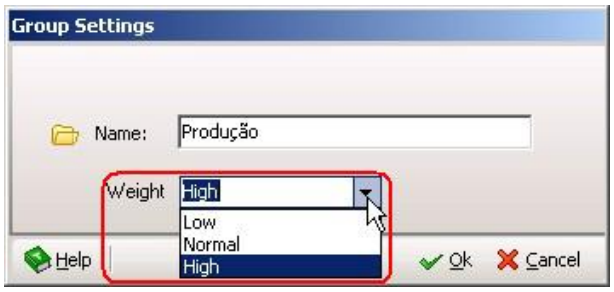


Figura 12: Janela de configuração do grupo de máquinas criado pelo usuário

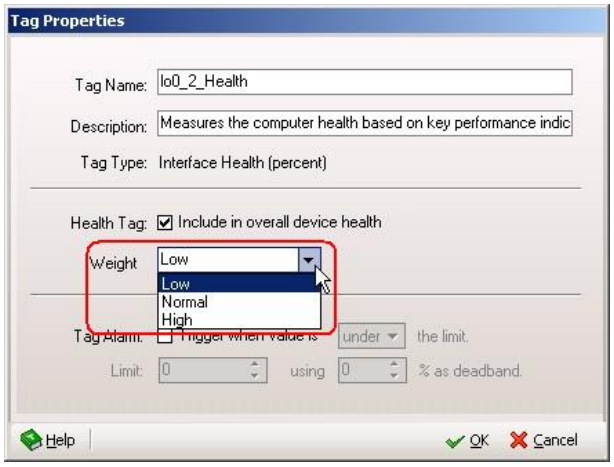


Figura 13: Janela de configuração das *tags* da máquina escolhida pelo usuário para ser gerenciada

Essas foram as mudanças visuais necessárias nos módulos *SNMP Config* e *PerfMon Config*.

As mudanças que foram feitas no módulo do *Health Analyst* foram aquelas que possibilitam ao usuário escolher o tipo de visualização da rede gerenciada e a área destinada à geração do treemap. A figura 14 mostra no detalhe o menu pelo qual o usuário pode fazer a seleção do modo de visualização da estrutura. Feita a seleção, caso o usuário opte por visualizar a rede na forma de treemap, a área que nessa figura está apresentando uma árvore passa a apresentar um treemap.

As demais informações do menu e do restante da interface apresentada na figura 14 são próprias do XRatel e não foram desenvolvidas neste trabalho.

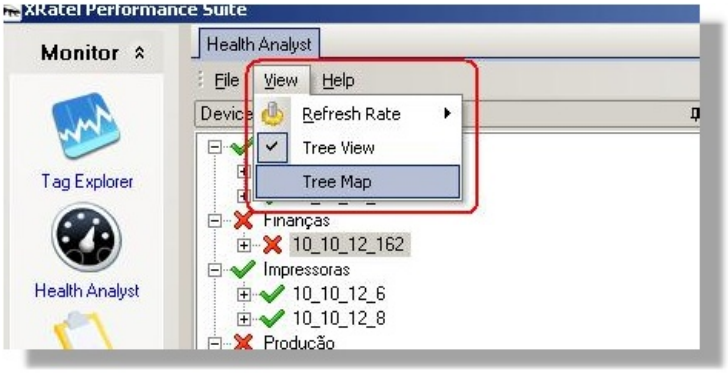
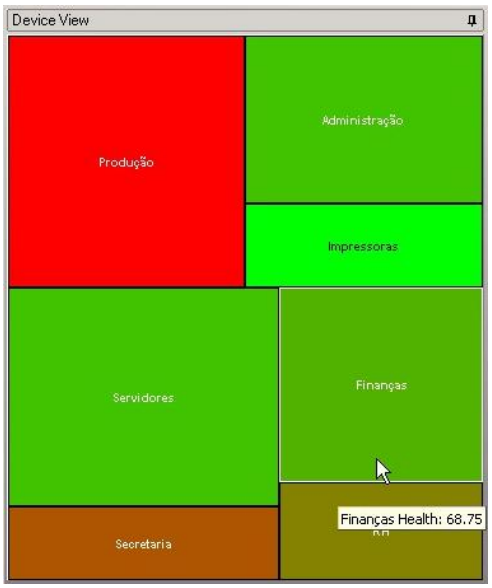
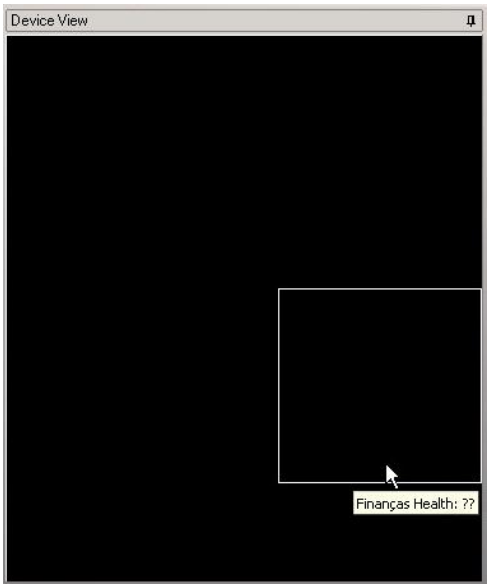


Figura 14: Detalhe do menu de seleção do modo de visualização da rede no módulo *Health Analyst* do XRatel

Um exemplo da apresentação de informação sob demanda do usuário na forma de *pop-up* e a cor de preenchimento do nó em um caso de funcionamento normal do XRatel e em um eventual caso de perda de comunicação com o servidor pode ser observado na figura 15, letras (a) e (b) respectivamente.



(a) *Pop-Up* apresentando o *Health* do equipamento na hora em que o mouse é passado sobre a máquina desejada e a cor que representa este valor de *Health*.



(b) Momento da perda de comunicação com o servidor do XRatel, e o comportamento do treemap nesta situação.

Figura 15: Exemplo da utilização de *pop-up* para a apresentação de informações sob demanda

5.2 Métricas de Validação da Ferramenta

O resultado final deste trabalho é uma ferramenta de visualização de estruturas hierárquicas baseadas nos conceitos dos treemaps. Essa ferramenta tem como objetivo auxiliar o usuário do XRatel na tomada de decisão, fornecendo-lhe mais informações que sejam realmente úteis para a função que desempenham.

Apesar de a ferramenta estar em uma versão comercial do XRatel, a pesquisa com o publico a respeito da eficiência da ferramenta no que diz respeito ao auxílio na tomada de decisão não pode, ainda, ser traduzida em números. No entanto, uma comparação entre as duas formas de visualização podem nos dar uma idéia de como o treemap desenvolvido pode auxiliar numa tomada de decisão.

Comparando as figuras 16 e 17, podemos notar algumas melhoras no que diz respeito à quantidade de informação útil para uma tomada de decisão apresentada pelo treemap e que não era apresentada na forma de visualização tradicional.

Quando o usuário olha para a árvore apresentada na figura 16, a informação que ele consegue obter de forma imediata é que existem quatro grupos de máquinas com problemas na rede que ele está gerenciando. Apenas essa informação não permite que o administrador da rede decida de forma segura a qual grupo de máquinas ele deve voltar sua atenção. Nesse momento, perguntas como: “Quais máquinas são mais importantes para mim?”, ou “Das máquinas com problemas qual está em um estado mais crítico?” são de extrema importância para uma tomada de decisão e não possuem respostas imediatas nessa forma de apresentação da rede.

No entanto, na visualização da rede em forma de treemap, como apresentado pela figura 17, no exato momento em que olhamos para a representação da rede, já voltamos a nossa atenção (como apresentado por Shneiderman (1996)) para o grupo que ocupa a maior área na representação da rede gerenciada, que é, por sua vez, o grupo mais importante. Além disso, nessa primeira visão da estrutura, podemos concluir que existem

quatro máquinas que não estão em perfeitas condições de funcionamento e, dessas máquinas, é possível avaliar comparativamente quais se encontram em uma situação mais crítica.

É importante notar que, por uma forma de visualização, o usuário do XRatel conclui que existem quatro máquinas com problemas, mas não sabe a qual voltar sua atenção por não saber qual grupo é o mais importante e nem qual grupo está em pior situação. Já por meio da outra forma de visualização, o usuário do XRatel conclui que existem quatro máquinas que não estão em perfeitas condições de funcionamento, mas já sabe a qual grupo de máquinas ele deve direcionar seus esforços.

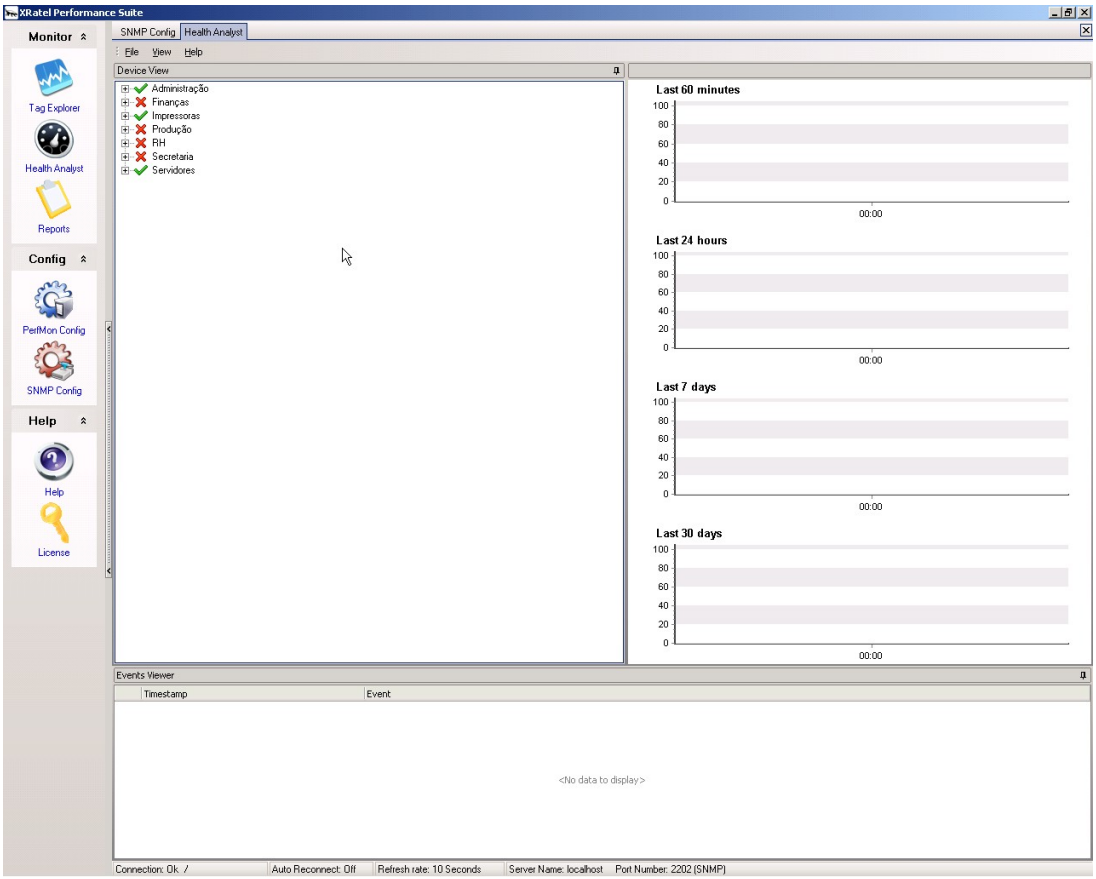


Figura 16: Visualização tradicional de um conjunto de máquinas gerenciadas pelo XRatel, organizadas em uma árvore que representa uma empresa fictícia

Outra informação que não é apresentada de forma imediata para o usuário do XRatel no *Health Analyst* quando a forma de visualização da rede não é o treemap é o valor numérico do *Health* das máquinas. Esse valor só pode ser inferido olhando o último ponto

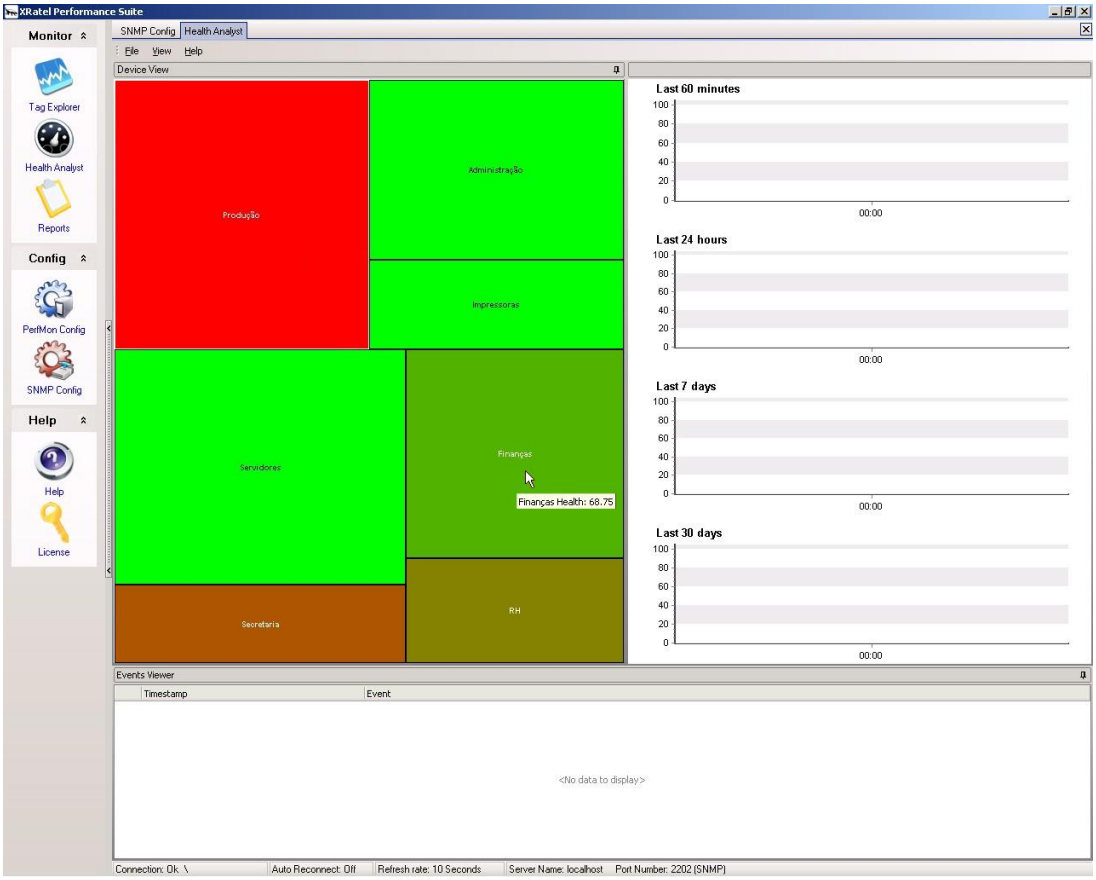


Figura 17: Visualização em treemap de um conjunto de máquinas gerenciadas pelo XRatel, organizadas em uma árvore que representa uma empresa fictícia

do gráfico histórico, dando um valor de *Health* com uma precisão muito baixa, uma vez que o eixo do gráfico que representa o valor do *Health*, apresenta apenas seis divisões, com intervalos de 20 unidades, como pode ser observado na figura 18.

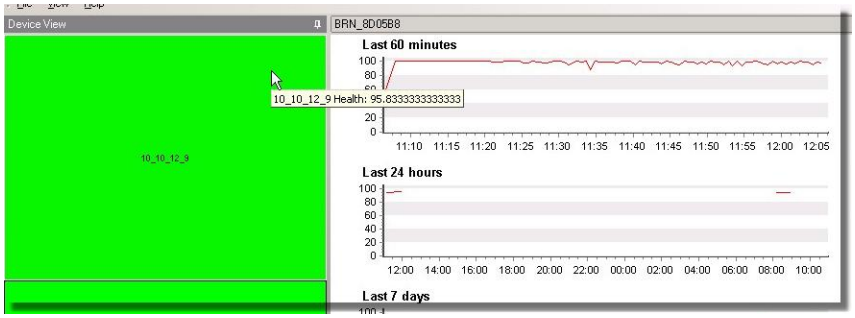
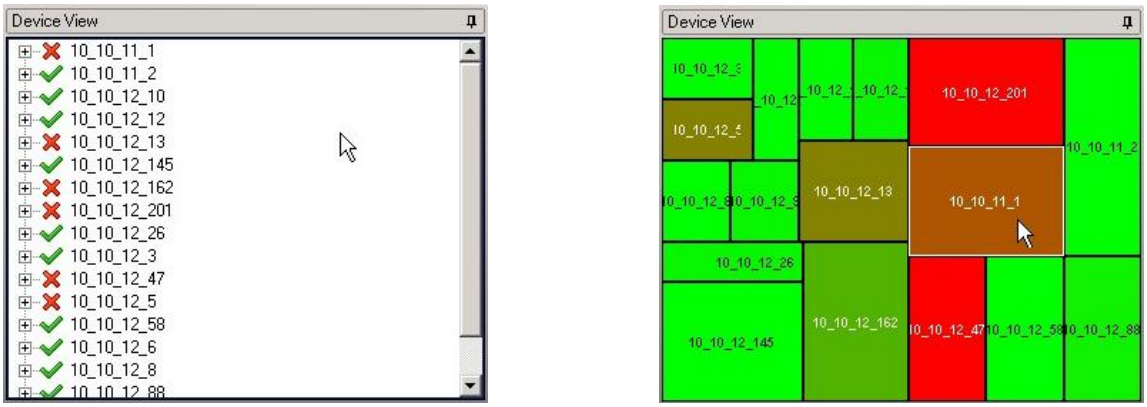


Figura 18: Comparação entre o valor de *Health* fornecido pelo *pop-up* e a forma como esse valor poderia ser obtido com o gráfico histórico

Outra melhoria obtida com a visualização na forma de treemaps que fica evidente na figura 19 é a melhor utilização da área disponibilizada para visualização da árvore. Na

forma de visualização tradicional, o usuário do XRatel não consegue olhar para a rede como um todo sem precisar ter o trabalho de recorrer à barra de rolagem. Isso impede que o usuário faça comparações entre todas as máquinas de forma direta. Já a visualização por meio de treemaps sempre permitirá ao usuário analisar a rede como um todo independente da quantidade de máquinas gerenciadas.



(a) Utilização da área destinada à apresentação da árvore

(b) Utilização da área destinada à apresentação do TreeMap

Figura 19: Comparação entre a utilização da área destinada à apresentação da árvore e do treemap

É interessante observar que apesar de o usuário necessitar fazer uso da barra de rolagem para visualizar a estrutura inteira, muito pouco da área destinada à visualização da árvore é utilizada para apresentar informações úteis.

Para avaliar o impacto do processamento do treemap no tempo de processamento do *Health Analyst*, foi feita uma medição do tempo que era gasto para que o *Health Analyst* atualizasse a informação da estrutura de visualização original como um todo. Em seguida, foi feita uma medição que avaliasse o novo tempo que o *Health Analyst* estava gastando para atualizar as duas formas de visualização: a original e o treemap.

Para a confecção do gráfico, foram feitas medidas do tempo de processamento gasto na atualização das estruturas à medida que o número de nós representados nas árvores foram variadas de 5 até 75. Como os tempos medidos eram muito pequenos, variando de dezenas a centenas de milisegundos, foi decidido, para trazer mais confiabilidade às medidas, tomar cinco medições de cada teste e, em seguida, considerar o valor como sendo

uma média desses valores. O gráfico apresentado na figura 20 mostra o resultado dessas medições.

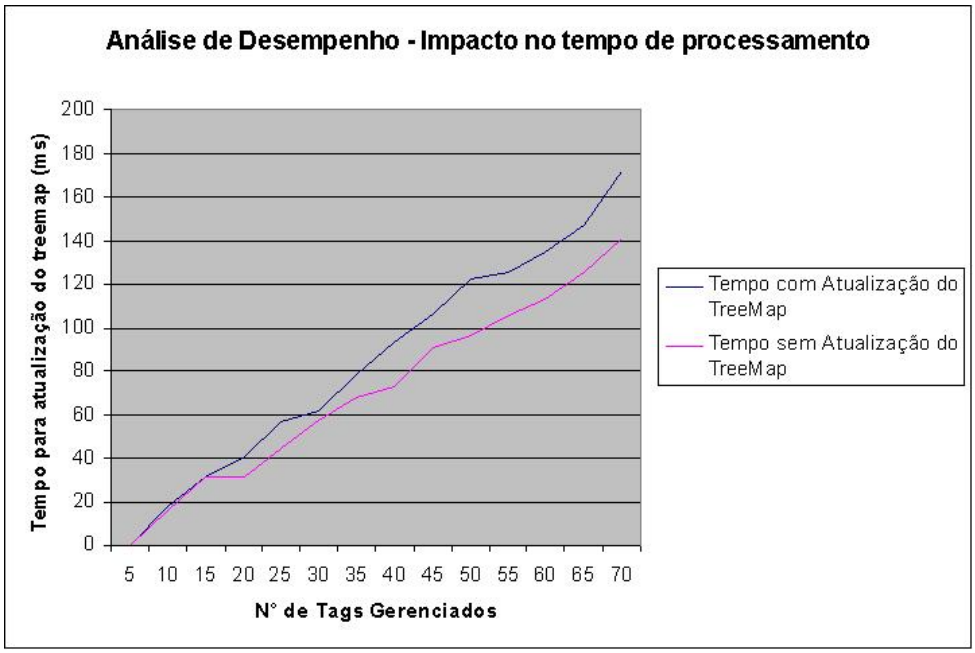


Figura 20: Comparação do tempo de processamento gasto para atualização do *Health Analyst* com e sem treemap

Uma análise rápida do gráfico permite concluir que o aumento do tempo de processamento foi pouco significativo em relação ao tempo de processamento antes da incorporação do treemap. Para uma estrura contendo 55 nós eram gastos 105ms para atualizar a estrutura como um todo; com o treemap, o tempo de atualização das duas estruturas (a antiga e a criada) passou para 125ms, ou seja, para atualizar o treemap, que é uma estrutura de mesmo tamanho da original, foram gastos apenas 20ms a mais.

Medições nos outros procedimentos desenvolvidos não apresentaram valores maiores que 1ms de tempo de execução e, por este motivo, foram considerados insignificantes para causar perda de performance no *Health Analyst*.

6 Conclusão

Considerando a importância da qualidade de uma ferramenta gerencial no que diz respeito ao auxílio na tomada de decisão e verificando a forte tendência de tais ferramentas na utilização de treemaps como uma alternativa moderna e eficiente para auxiliar a tomada de decisão quando a mesma depende da análise de estruturas hierárquicas, ficou evidente a aplicabilidade dessa ferramenta no software de gerenciamento de ativos em rede desenvolvido pela empresa XRatel Software.

Ficou determinado, por tanto, como objetivo principal deste trabalho, o desenvolvimento de um treemap para apresentar a rede gerenciada pelo XRatel, que já era apresentada na forma de uma árvore.

Para tanto, foi feita uma discussão das tecnologias e conceitos empregados em uma ferramenta de gerenciamento de ativos em rede, apresentando alguns detalhes do software XRatel, além de apresentar um breve estudo das formas de visualização de estruturas hierárquicas existentes, dando maior ênfase às características da maneira utilizada atualmente pela empresa parceira. Nesse estudo, foi apresentado os pontos fortes e fracos do modelo tradicional de visualização de estruturas hierárquicas adotado e os possíveis ganhos que se obteria com a inserção do treemap como forma de visualização da estrutura da rede.

Em um segundo momento, foi exposto um estudo dos algoritmos existentes para geração de treemaps em que se procurou apresentar uma evolução dos algoritmos desenvolvidos desde a criação dos mesmos apresentando as características importantes de cada algoritmo e a relação dessas características com a função da ferramenta que deseja utilizá-los para

a geração do treemap. Com base nessas características, foi escolhido o algoritmo denominado *Split* desenvolvido por Engdahl (2005) que até a conclusão deste trabalho não se tinha notícia da utilização do mesmo em nenhum software comercial.

Concluído o desenvolvimento, foi feita uma apresentação das melhorias obtidas com a nova forma de visualização, podendo perceber claramente a melhoria que tal ferramenta trouxe ao usuário do software no que diz respeito ao auxílio da tomada de decisão por parte deles.

A ferramenta se mostrou eficiente e robusta, sem trazer diminuição de performance para o software, e acrescentando uma funcionalidade que traz muito mais facilidade de assimilação das informações apresentadas por parte do usuário.

Referências

BRANT, A. C. et al. Sistema PIMS na gestão de ativos de TI para automação. In: ASSOCIAÇÃO BRASILEIRA DE METALURGIA E MATERIAIS. *X Seminário de Automação de Processos*. Belo Horizonte, Brasil, 2006. p. 11.

ENGDAHL, B. *Ordered and unordered treemap algorithms and their applications on handheld devices*. 46 p. Dissertação (Master's degree project) — Royal Institute of Technology, Stockholm, 2005.

FEW, S. *Information Dashboard Design*. California: O'Reilly, 2006.

HEER, J. *Position Paper*. 2007. Acessado em 30 de Agosto de 2007. Disponível em: <http://vw.indiana.edu/ivsi2004/jherr/index.html>.

HOLDEN, G. *Implementing Decision-support Portals based on Data Visualization Best Practices*. [S.l.], Jun. 2007.

JOHNSON, B.; SHNEIDERMAN, B. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. *International Ieee Visualization Conference*, v. 1, p. 284–291, 1991.

KUROSE, J. F.; ROSS, K. W. *Rede de Computadores e a Internet*: Uma nova abordagem. São Paulo: Pearson Education, 2004.

MATRIKON. *Production Optimization through Advanced Condition Monitoring*. [S.l.], 2005. Disponível em: <www.matrikon.com>.

MITCHELL, W.; SHOOK, D. *Finding the Needle in the Haystack*: An innovative means of visualizing control performance problems. Canada, 2005.

MSDN. *Windows Management Instrumentation*. [S.l.], 2007. Acessado em 13 de Setembro de 2007. Disponível em: <http://msdn2.microsoft.com/en-us/library/aa394582.aspx>.

SCHULZ, H.-J.; SCHUMANN, H. Visualizing graphs: A generalized view. *Proceedings Of The Information Visualization*, Rostock, v. 6, p. 8, 2006.

SEIXAS FILHO, C. et al. Monitoração de desempenho de redes de automação usando snmp. In: ASSOCIAÇÃO BRASILEIRA DE METALURGIA E MATERIAIS. *IX Seminário de Automação de Processos*. Curitiba, Paraná, Brasil, 2005. p. 11.

SHAH, S. L.; MITCHEL, W.; SHOOK, D. Integration of controller performance metrics in the asset management workflowprocess. In: *Workshop on Solving Industrial Control and Optimization Problems*. Gramado, Brasil: [s.n.], 2006. p. 14.

SHAPLEY, R. *Visualizing the Tree of Life*. 2007. Acessado em 30 de Agosto de 2007. Disponível em: <http://www.rebeccashapley.com/cipres/bibliography.htm>.

SHNEIDERMAN, B. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, Maryland, v. 11, p. 92–99, jan. 1992.

SHNEIDERMAN, B. The eyes have it: A task by data type taxonomy for information visualizations. *IEEE*, IEEE, v. 6, p. 8, 1996.

SMARTMONEY.COM. *Uso de treemaps para visualização do desempenho de ações no mercado financeiro mundial*. 2007. Acessado em 16 de Agosto de 2007. Disponível em: <http://www.smartmoney.com/marketmap/>.

WILSON, J. *The cost of network downtime*. 25 Apr. 2003. Acessado em 28 de Setembro de 2007. Disponível em: <http://telephonyonline.com/analysts/infonetics/telecom>.

XRATEL SOFTWARE. *XRatel Performance Suite Help*. Brasil, 2007.