MAYKEL MELO DA SILVA

DESENVOLVIMENTO DE UMA FERRAMENTA DE TESTES PARA UM SISTEMA INTEGRADO EM SOFTWARE E HARDWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS - BRASIL
2005

MAYKEL MELO DA SILVA

DESENVOLVIMENTO DE UMA FERRAMENTA DE TESTES PARA UM SISTEMA INTEGRADO EM SOFTWARE E HARDWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de concentração:
Engenharia de Software

Orientador:
Prof. Guilherme Bastos Alvarenga

LAVRAS
MINAS GERAIS - BRASIL
2005

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca Central da UFLA

Silva, Maykel Melo da

Desenvolvimento de uma Ferramenta de Testes para um Sistema Integrado em Software e Hardware

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Informática. 2. Engenharia de Software. 3.Testes de Sistemas. I. SILVA, M. M. II. Universidade Federal de Lavras. III. Título.

MAYKEL MELO DA SILVA

DESENVOLVIMENTO DE UMA FERRAMENTA DE TESTES PARA UM SISTEMA INTEGRADO EM SOFTWARE E HARDWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 27/04/2006

_____
Prof. Reginaldo Ferreira de Souza

_____
Prof. José Monserrat Neto

_____
Prof. Guilherme Bastos Alvarenga
(Orientador)

LAVRAS
MINAS GERAIS - BRASIL

Aos meus pais

Agradeço à minha família que sempre me apoiou e não mediu esforços para que eu cumprisse mais esta etapa de minha vida.

Agradeço à minha namorada, Nayara, pela paciência e companheirismo, principalmente nos momentos mais difíceis. Também por me mostrar que não podemos viver pensando somente em estudo e trabalho.

Agradeço ao professor Guilherme Bastos Alvarenga pela oportunidade e confiança em mim depositada.

Agradeço à equipe de desenvolvimento da Devex Tecnologia e Sistemas Ltda pela ajuda e participação.

Agradeço também aos colegas de turma pela companhia durante esses anos.

RESUMO

Este trabalho apresenta o desenvolvimento de um simulador de ambiente usado como ferramenta de teste para o SmartMine, uma solução integrada de hardware/sofware em tempo real para melhoramento da produtividade e qualidade em minas de céu aberto. A motivação para o desenvolvimento do mesmo é que o SmartMine mostrou-se um sistema difícil de ser testado em tempo de desenvolvimento visto que necessita de toda estrutura de uma mina em funcionamento para receber os dados que serão processados. Essa complexidade contribuía para que o sistema não fosse testado de forma efetiva e deixasse que defeitos de desenvolvimento chegassem ao cliente. O desenvolvimento foi bem sucedido e o software desenvolvido foi incorporado ao processo da organização.

Palavras-chave: simulador de ambiente, teste.

ABSTRACT

This work issues the construction of an enviroment simulator to be used as a SmartMine's testing tool. The SmartMine is an integrated real time hardware/software solution for productivity and quality improvement in open pit mines. The motivation to this work is that the SmartMine has showed itself as a difficult system to be tested in coding time, keeping in mind that the system needs all of the working mine structure to receive the data to be processed. This complexity contributes to the system do not be tested well enough and allows the defects appear for the users. At the end, it will be clear that the development has been successful and the software has been incorporated in the organization process.

Keywords: enviroment Simulator, testing.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

O presente trabalho foi desenvolvido durante um período de estágio na empresa Devex Tecnologia e Sistemas Ltda. Essa é uma empresa de desenvolvimento de sistemas com sede em Belo Horizonte - MG. A empresa mantém uma parceria com a UFLA oferecendo estágios aos graduandos em Ciência da Computação.

1.1 Conceitos Básicos Sobre Testes de Sistemas

A cada dia nos tornamos mais dependentes de sistemas de software. Muitos dos produtos que utilizamos incorporam, de alguma forma, computadores e softwares de controle. Por esse motivo, cada vez mais precisamos de mecanismos que auxiliem o desenvolvimento desses sistemas, para que uma série de garantias seja alcançada. Nesse sentido, os processos de software têm uma importante contribuição no mundo atual. Através da especificação de um conjunto de atividades e resultados associados, temos uma formalização das tarefas a serem executadas para se construir um produto (SOMMERVILLE, 2003)

A maioria dos processos de software incorpora, de alguma forma, atividades de testes em suas definições. De acordo com a definição da IEEE (1983) em sua Padronização para Documentação de Teste de Software, a atividade de teste é:

"O processo de análise de um item de software para detectar diferenças entre condições existentes e especificadas (isto é, bug) e validar as características dos itens de software."

Esse tipo de atividade deveria ser aplicado em todo o ciclo de vida do software, e não restrito à verificação (teste para determinar se uma fase do processo foi realizada corretamente) e à validação (teste para determinar se o produto completo segue as especificações) (SCHACH, 1996).

Essa definição da IEEE traz o item de software. De acordo com esse mesmo documento (IEEE, 1983), o item de software poderia ser um código fonte, um código de objeto, um código de controle de tarefa, um dado de controle ou uma coleção destes itens.

A atividade de teste é geralmente bastante onerosa no desenvolvimento de um software. Dependendo do tipo de sistema a ser desenvolvido, ela pode ser responsável por mais de 50% dos custos envolvidos (SOMMERVILLE, 2003). Isso ocorre porque uma parte significativa dessas atividades ainda é executada de forma manual, com a geração de

casos de teste baseada nos documentos de requisitos ou em outros documentos produzidos durante o processo de desenvolvimento. Nesse ponto percebe-se a importância das ferramentas de teste automatizadas no processo de desenvolvimento do software.

A presença de defeitos em softwares não causa apenas gastos excessivos no desenvolvimento. Dependendo do dispositivo executando o software, acidentes podem acontecer. Infelizmente, muitas dessas falhas já ocorreram, e muitas vidas foram perdidas. Como exemplo, podemos citar alguns casos em que a ocorrência de falhas no software ocasionou graves acidentes:

- Em 1999, um satélite enviado para aterrissar em marte foi destruído na tentativa de aterrissagem em virtude de problemas nos tipos de medidas utilizadas.
- Em 1994, um avião da China Airlines caiu nas proximidades do aeroporto de Nagoya, com 264 mortes. Houve um erro no software controlador de vôo.
- Em 1986, uma máquina de terapia radioativa controlada por computador (Therac 25) foi o responsável pela morte de duas pessoas, em virtude de uma dose excessiva de radiação aplicada no tratamento. O erro foi causado pela inabilidade da máquina em tratar condições de disputa.

Considerando a importância da atividade de teste no desenvolvimento de qualquer sistema, focaliza-se agora o SmartMine em particular.

1.2 Termos Importantes

A compreensão dos seguintes termos facilitará o entendimento do trabalho a partir deste ponto:

Equipamentos de mina: caminhões, escavadeiras, máquinas de carga, e equipamentos auxiliares responsáveis por extrair e transportar material na mina.

Computador de bordo: computador produzido pela Devex e instalado nos equipamentos de mina. É responsável pela comunicação com o GPS (sistema de localização terrestre baseado em comunicação com satélites) e envio de informações sobre a operação do equipamento para o módulo de Despacho.

Eventos: ações realizadas pelos equipamentos da mina que fornecem dados sobre sua produtividade para o módulo de Despacho. São enviados ao Despacho em pacotes de rede e é através destes que a produção da mineração pode ser obtida em tempo real.

Exemplos de eventos dos equipamentos de mina: carregamento, basculamento, pesagem, parada para manutenção, deslocamento vazio ou cheio, etc.

Access Point: Dispositivo de hardware responsável por captar os dados enviados pela rede sem fio, usada pelos computadores de bordo para enviar informações, e os encaminhar ao módulo de Despacho através de uma rede Ethernet pelo protocolo TPC/IP. A maneira como ocorre a comunicação entre os computadores de bordo e o módulo de Desapacho é descrita em detalhes por PASSOS (2002)

1.3 SmartMine

O SmartMine é uma solução integrada de hardware/sofware para melhoramento da produtividade e qualidade em minas de céu aberto. Este sistema armazena e trata cada viagem realizada pelos caminhões, a fim de controlar toda a movimentação de material, seja este minério ou estéril[1]. Ou seja, deseja–se saber entre outras questões, a origem e o destino do material, qual caminhão realizou a viagem, quanto tempo gastou, qual a massa produzida, etc. Esses dados são extremamente importantes para gerar resultados como a produtividade de cada equipamento, a qualidade de cada depósito, tempo ocioso de operação, entre outros.

O sistema foi projetado em cinco módulos (vistos com maiores detalhes em um capítulo posterior):

- Módulo de Planejamento;
- Módulo de Despacho;
- Módulo GPS;
- Módulo de Consulta;
- Módulo de Otimização.

1.4 Objetivo e Relevância do Trabalho

O objetivo do trabalho é desenvolver uma ferramenta de testes para o módulo de Despacho do sistema SmartMine.

Esse módulo possibilita que informações decisivas para o aumento de produtividade e qualidade do produto final cheguem a quem é necessário no tempo e locais corretos.

[1] Estéril: todo o material fora da especificação física ou química aceitável na usina de tratamento de minério, considerando o contexto atual e futuro.

Através deste módulo, é possível conhecer os locais exatos onde ocorrem os carregamentos, os pontos de basculamento, a produtividade instantânea nos diversos fluxos de produção, a qualidade do minério produzido e muito mais.

O módulo de Despacho é aquele para onde convergem todas as informações em tempo real do processo de produção da mina. A grande dificuldade em se testar este módulo consiste na integração deste com os computadores de bordo dos equipamentos da mina. Esses computadores (mais de trinta) enviam pacotes de rede a todo instante para o módulo de Despacho contendo informações sobre sua operação. É através destes pacotes que o Despacho obtém informações sobre produtividade e qualidade da mineração em tempo real.

Por ser um sistema de tempo real que necessita de toda uma infra-estrutura de mineração para receber dados de produção, o módulo de controle (ou módulo de Despacho) do SmartMine caracteriza-se como um sistema difícil de ser testado em tempo de desenvolvimento, como dito acima.

No momento da implementação do sistema, não há dados de produção de viagens, manutenções, atrasos, etc. para que seus desenvolvedores possam se certificar do correto funcionamento do sistema ao reagir aos pacotes recebidos dos equipamentos de mina.

Com essa dificuldade em se testar mais efetivamente o sistema, defeitos podem aparecer durante o uso do mesmo pelos clientes, o que gera uma situação de descontentamento em relação ao software por parte dos mesmos e pode trazer prejuízos às empresas, tanto cliente, que perderá dados de produção importantes, quanto para a Devex, que terá de deslocar uma equipe de suporte até a mina, alocar recursos (que poderiam estar trabalhando em novos projetos) para corrigir o problema e, principalmente, resultar na perda de credibilidade do SmartMine.

A função do simulador é gerar dados referentes à operação dos equipamentos da mina para serem enviados para o módulo de Despacho. Esses equipamentos podem ser tanto de transporte quanto de carga. Nos dados gerados estão contidas as informações sobre os eventos dos equipamentos da mina. Como dito anteriormente esses eventos são ações realizadas pelos caminhões, carregadeiras, escavadeiras, etc, que afetam a produtividade da mineração e precisam ser tratados pelo módulo de Despacho.

Entre os eventos gerados pelo simulador encontram-se: carregamento, basculamento, perda de comunicação com o GPS, restabelecimento da comunicação com o GPS, deslocamento e outros que serão vistos em detalhes posteriormente.

No mundo real, os dados fornecidos ao módulo de Despacho são gerados pelo computador de bordo de cada equipamento, conhecido como *tracker*. Os *trackers* enviam suas informações através de pacotes de rede e usam uma rede *wireless* para comunicação com o *Access Point*. O *Access Point* é um dispositivo de hardware fixo na mina que tem a função de captar os pacotes transmitidos na rede *wireless* e encaminhá-los ao módulo de Despacho através de uma rede Ethernet, pelo protocolo TCP/IP. Como o objetivo do simulador é gerar dados e enviá-los ao módulo de Despacho, é necessário que este possa fazê-lo através de uma rede Ethernet.

Este simulador, chamado *SMAccessPoint Simulator*, fará o papel do dispositivo *Access Point*, que enviará ao despacho todos os pacotes supostamente recebidos dos equipamentos através da rede sem fio. Mas ao invés de receber os pacotes dos equipamentos através da referida rede, o simulador estará gerando ele próprio os eventos que serão enviados ao módulo de controle. Portanto, o usuário do *SMAccessPoint Simulator* poderá gerar os eventos que forem necessários para testar o sistema quantas vezes precisar e na ordem em que desejar. O usuário terá controle total sobre todos os "equipamentos virtuais" da mineração.

Essa é uma idéia nova visto que os simuladores existentes atualmente não são capazes de simular os trackers dos equipamentos de mina para enviar dados ao Despacho. Como a Devex desenvolveu um protocolo próprio de comunicação entre o módulo de controle e os computadores de bordo, foi necessário desenvolver um sistema para testar a integração entre o software e o hardware de controle usando o protocolo de comunicação desenvolvido pela empresa.

O simulador de ambiente de mineração encontrado, mais próximo da necessidade da empresa, é um *template* desenvolvido para o programa ARENA.

O ARENA é um ambiente gráfico integrado de simulação, que contém todos os recursos para modelagem, animação, análise estatística e análise de resultados. Este usa a abordagem por processos para a execução da simulação. Essa técnica de simulação pode ser considerada como uma situação onde elementos estáticos, formando um ambiente bem definido com suas regras e propriedades, interagem com elementos dinâmicos, que fluem dentro desse ambiente. Em minas de céu aberto seria o caso de caminhões (elementos dinâmicos) interagindo com escavadeiras, britadeiras, pilhas de estéril e outros (elementos estáticos) que formam o ambiente de uma mina (NETO, 2004).

Com o uso deste simulador as dificuldades de se testar o sistema, devido à necessidade dos dados gerados pela operação da mina, será eliminada e o sistema será testado mais eficazmente.

1.5 Organização do Trabalho

Neste primeiro capítulo é apresentada uma introdução sobre a importância das atividades de teste no desenvolvimento de software e a motivação para desenvolver uma ferramenta de testes para o SmartMine. No segundo e terceiro capítulos é apresentado um levantamento teórico sobre sistemas de tempo real e sobre testes de software. O quarto capítulo descreve o sistema SmartMine, nos seus aspectos relevantes para este trabalho. No quinto capítulo, a metodologia empregada no trabalho é descrita. No sexto, sétimo e oitavo capítulos são apresentados os resultados, conclusões e trabalhos futuros.

2 SISTEMAS DE TEMPO REAL

2.1 Conceituação Básica e Caracterização de um Sistema de Tempo Real.

Excetuando sistemas computacionais voltados para o cálculo – normalmente identificados como *Sistemas Transformacionais* que calculam valores de saída a partir de valores de entrada e depois terminam seus processamentos, como ocorre, por exemplo, com compiladores, programas de engenharia econômica e programas de cálculo numérico, uma grande parte dos sistemas computacionais atuais interagem permanentemente com os seus ambientes. Entre esses últimos, distinguem-se os chamados *Sistemas Reativos* que reagem enviando continuamente respostas a estímulos de entrada vindos de seus ambientes. Sistemas de tempo real de uma forma geral se encaixam neste conceito de sistemas reativos (OLIVERIA, 2002):

Um Sistema de Tempo Real (STR, ou RTS "*Real Time System*") é um sistema computacional que deve reagir a estímulos oriundos do seu ambiente em prazos específicos.

O atendimento desses prazos resulta em requisitos de natureza temporal sobre o comportamento desses sistemas. Em conseqüência, em cada reação, o sistema de tempo real deve entregar um resultado correto dentro de um prazo específico, sob pena de ocorrer uma falha temporal. O comportamento correto de um sistema de tempo real, portanto, não depende só da integridade dos resultados obtidos (correção lógica ou "*correctness*") mas também dos valores de tempo em que são produzidos (correção temporal ou "*timeliness*"). Uma reação que ocorra além do prazo especificado pode ser sem utilidade ou até representar uma ameaça.

A maior parte das aplicações de tempo real se comporta então como sistemas reativos com restrições temporais. A reação dos sistemas de tempo real aos eventos vindos do ambiente externo ocorre em tempos compatíveis com as exigências do ambiente e mensuráveis na mesma escala de tempo.

A concepção do sistema de tempo real é diretamente relacionada com o ambiente no qual está inserido e com o comportamento temporal do mesmo.

Na classe de Sistema de Tempo Real na qual se encontram os sistemas embutidos ("*Embedded Systems*") e os sistemas de supervisão e controle, distinguem-se:

- Sistema a Controlar,
- Sistema Computacional de Controle
- e o Operador.

O Sistema a Controlar e o Operador são considerados como o Ambiente do Sistema Computacional. A interação entre os mesmos ocorre através de interfaces de instrumentação (compostas de sensores e atuadores) e da interface do operador. A figura representa esse tipo de Sistema de Tempo Real.

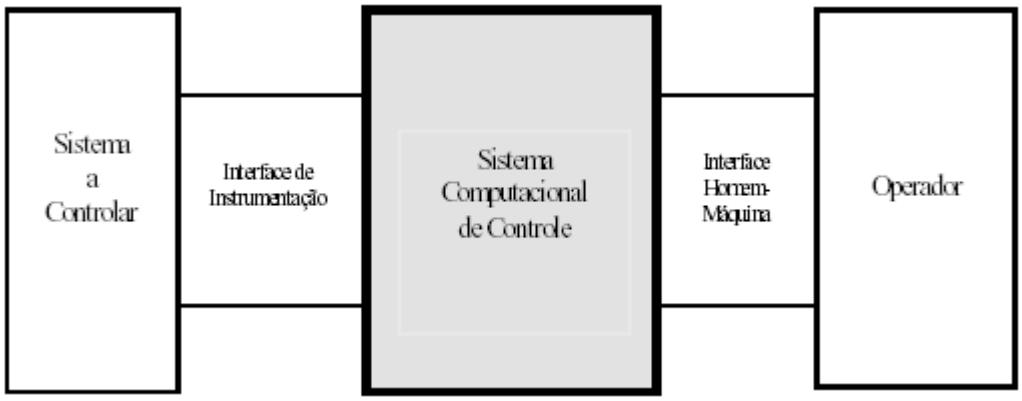

Figura 2.1 – Sistema de tempo real

Existem também situações nas quais as restrições temporais não são impostas pelo comportamento dinâmico de um eventual Sistema a Controlar, mas pelas exigências dos serviços a serem oferecidos a um usuário humano ou computacional (por exemplo, no caso do manuseio da apresentação de vídeos em sistemas multimídia). Nesses casos utiliza-se a noção de Serviço de Tempo Real como sendo um serviço que deve ser oferecido dentro de restrições de tempo impostas por exigências externas ao próprio Sistema Computacional. Então, numa generalização podemos assumir:

Um Sistema de Tempo Real deve ser então capaz de oferecer garantias de *correção temporal* em todos os seus serviços que apresentem *restrições temporais*.

O SmartMine é caracterizado como um sistema de tempo real porque precisa disponibilizar ao usuário toda a informação da operação da mina com um atraso máximo aceitável de três segundos. Essa informação deve ser recebida dos computadores de bordo dos equipamentos de mina, processada pelo sistema e disponibilizada ao usuário para que esse possa tomar decisões relativas à produção imediatamente. Esse atraso de três segundos foi estabelecido empiricamente com base em uma média do tempo gasto para os pacotes de rede gerados pelos *trackers* serem recebidos pelo módulo de Despacho.

2.2 Classificação dos Sistemas de Tempo Real

Os Sistemas de tempo real podem ser classificados a partir do ponto de vista da Segurança ("*Safety*") em:

Sistemas Não Críticos de Tempo Real ("*Soft Real Time Systems*"): também conhecidos como STR brandos, são sistemas cuja falha no cumprimento dos limites de tempo não acarreta em danos e/ou prejuízos significativos (por exemplo, sistema de comutação telefônico, sistema bancário, jogos de computador, etc).

Sistemas Críticos de Tempo Real ("*Hard Real Time Systems*"): também chamados de STR duros, são aqueles cujas conseqüências de uma falha no cumprimento dos limites de tempo podem ser catastróficas, ou melhor, o custo de tais falhas é de uma ordem de grandeza superior à da própria utilidade do sistema (por exemplo, sistema de controle de vôo em um avião, sistema de controle de sinalização em uma ferrovia, sistema de controle de uma planta nuclear, e outros sistemas dessa ordem de importância).

O grupo de Sistemas Críticos de Tempo Real ainda pode ser subdividido em:

Sistemas de Tempo Real Críticos Seguros em Caso de Falha ("*fail safe*"): onde um ou vários estados seguros podem ser atingidos em caso de falha (por exemplo, parada obrigatória de trens no caso de uma falha no sistema de sinalização ferroviária).

Sistemas de Tempo Real Críticos Operacionais em Caso de Falha *("fail operational*"): que na presença de falhas podem se degradar, mas ainda assim fornecer um serviço mínimo e continuar funcionando (por exemplo, um sistema de controle de vôo com funcionamento degradado, porém ainda operante).

O SmartMine pode ser caracterizado como um Sistema de Tempo Real Crítico Operacional em Caso de Falha, devido ao atraso de uma informação poder causar um grande prejuízo, mas não resultar necessariamente em acidentes, nem na parada do sistema. Como exemplo, pode-se supor que um pacote de dados indicando parada de um equipamento de carga não chegue ao módulo de Despacho e essa informação não seja mostrada ao usuário a tempo. O usuário, desconhecendo a parada do equipamento, pode enviar uma dúzia de caminhões para carregar na máquina de carga parada. Tal fato acarretaria grande prejuízo à empresa com perda de produtividade e gasto desnecessário de combustível, mas o sistema não precisou parar de funcionar devido ao atraso, nem ocasionou algum acidente.

3 TESTES DE SOFTWARE

Pode-se dizer que a atividade de teste é um elemento crítico da garantia de qualidade de software. Esta representa a última revisão de especificação, projeto e codificação, tal como descreve DEUTSCH (1979):

"O desenvolvimento de sistemas de software envolve uma série de atividades de produção em que as oportunidades de injeção de falhas humanas são enormes. Erros podem começar a acontecer logo no início do processo, onde os objetivos... podem estar erroneamente ou imperfeitamente especificados, além de erros que venham a ocorrer em fases de projeto e desenvolvimento posteriores... Por causa da incapacidade que os seres humanos têm de executar e comunicar com perfeição, o desenvolvimento de software é acompanhado por uma atividade de garantia de qualidade".

De acordo com PRESSMAN (1995), a atividade de teste também demonstra que as funções de software aparentemente estão trabalhando de acordo com as especificações, que os requisitos de desempenho aparentemente foram cumpridos. Além disso, os dados compilados quando a atividade de testes é levada a efeito proporcionam uma boa indicação da confiabilidade de software e alguma indicação da qualidade do software como um todo.

3.1 Erro, Falta e Falha

De acordo com PFLEGER (1996), há três tipos básicos de defeitos em Software:

Erro: todos os programadores cometem erros. Esses podem ser erros de documentação ou codificação. Na grande maioria das vezes são erros lógicos, de condições de funcionamento não previstas. Quase sempre esses erros são percebidos e corrigidos no mesmo instante, mas nem todos.

Falta: uma falta é causada por um erro humano deixado no produto de software. Essas faltas podem se situar em qualquer parte do produto e nunca serem executadas, mas, caso sejam, podem levar a uma falha do sistema.

Falha: uma falha é quando um usuário está usando o sistema e este não responde da maneira como deveria. Falhas são causadas quando uma falta é executada.

Figura 3.1 – Ilustração da terminologia de defeitos.

3.2 Métodos de Teste

Pode-se testar um software tanto durante a execução de seu código fonte, quanto sem a execução do mesmo.

Quando um software é testado sem sua execução, toda a documentação referente ao mesmo é analisada por uma equipe de especialistas em seu campo de atuação. ma equipe de especialistas no campo de atuade testes mpridos.□□Nesse modo de teste, busca-se por erros conceituais relativos ao funcionamento do programa. Possui grande valor tanto nos estágios iniciais do desenvolvimento, quanto no restante do processo de produção.

A grande vantagem de se testar um software sem execução é que o teste pode ser realizado antes mesmo de qualquer implementação, revisando os requisitos levantados e verificando se estes realmente atendem às necessidades do cliente.

O teste baseado na execução do software possui dois métodos principais: o teste de caixa branca e o teste de caixa preta.

3.2.1 Teste de Caixa Branca

Este método baseia-se num minucioso exame dos detalhes procedimentais. Os caminhos lógicos através do software são testados fornecendo-se casos de teste que põem à prova conjuntos específicos de condições e/ou laços. O status do programa pode ser examinado em vários pontos para determinar se o status esperado ou estabelecido corresponde ao real. Este é um método útil para encontrar erros no programa.

De acordo com PRESSMAN (1995), as seguintes técnicas principais podem ser aplicadas no teste de caixa branca:

- Teste de caminho básico
- Notação de grafo de fluxo
- Complexidade ciclomática

Ainda de acordo com PRESSMAN (1995), podemos realizar os seguintes testes no método de caixa branca:

- Teste de estrutura de controle
- Teste de condição
- Teste de fluxo de dados
- Teste de laços

3.2.2 Teste de Caixa Preta

Quando se considera o software de computador, a expressão teste de caixa preta refere-se aos testes que são realizados nas interfaces do software. Não obstante eles sejam projetados com o propósito de descobrir faltas e falhas, os testes de caixa preta são usados para demonstrar que as funções do software são operacionais; que a entrada é adequadamente aceita e a saída é corretamente produzida; que a integridade das informações externas (por exemplo, arquivos de dados) é mantida, etc. Um teste de caixa preta examina alguns aspectos de um sistema sem se preocupar muito com a estrutura lógica interna do software.

De acordo com PRESSMAN (1995) as seguintes técnicas principais podem ser aplicadas no teste de caixa preta:

- Particionamento de equivalência
- Análise de valor limite
- Técnicas de grafo de causa-efeito
- Testes de comparação.

3.3 Estratégias de Teste de Software

Considerando-se o processo de desenvolvimento de software de um ponto de vista procedimental, a atividade de teste dentro do contexto da engenharia de software é, de fato, uma série de quatro passos implementados seqüencialmente.

No início, os testes focalizam cada módulo individualmente, garantindo que ele funcione adequadamente como uma unidade. Por isso o nome **teste de unidade**. Esse faz muito uso das técnicas de teste de caixa branca, exercitando caminhos específicos da estrutura de controle de um módulo, a fim de garantir uma completa cobertura e máxima

detecção de erros. Os erros no software são, na maioria das vezes, encontrados durante o teste de unidade.

Em seguida os módulos devem ser montados ou integrados para formarem um pacote de software. O **teste de integração** cuida das questões associadas aos problemas da verificação e construção de programas. As técnicas de projeto de casos de teste de caixa preta são as mais encontradas durante a integração, não obstante uma quantidade limitada de teste de caixa branca possa ser usada para garantir a cobertura de caminhos importantes de controle. Durante o teste de integração é que as faltas e falhas no software começam a ser descobertas. Quando se integra dois ou mais módulos, os erros não capturados durante o teste de unidade (faltas) podem resultar em comportamento não especificado do programa (falha). Como os módulos são testados em número reduzido, e não na totalidade do software, as faltas são mais fáceis de serem identificadas.

Depois que o software foi integrado (construído), um conjunto de testes de alto nível é realizado. Os critérios de validação (estabelecidos durante a análise de requisitos) devem ser testados. O **teste de validação** oferece a garantia final de que o software atende a todas as exigências funcionais, comportamentais e de desempenho. As técnicas de caixa preta são usadas exclusivamente durante a validação. A partir desta etapa dos testes, como o software é tratado como uma caixa preta, apenas as falhas são evidentes à equipe de testes ou aos usuários.

A última etapa de testes de alto nível extrapola a fronteira da engenharia de software e situa-se no contexto maior da engenharia de sistemas computadorizados. O software, uma vez validado, deve ser combinado com outros elementos do sistema (por exemplo, hardware, pessoas, bancos de dados, etc). O **teste de sistema** verifica se todos os elementos combinam-se adequadamente e se a função/desempenho global do sistema é conseguida.

3.4 Teste de Sistema de Tempo Real

As características especiais de sistemas de tempo real apresentam grandes desafios quando a atividade de testes está sendo realizada. A natureza assíncrona (dependente do tempo) de muitas aplicações de tempo real acrescenta um novo e potencialmente difícil elemento à combinação de testes – o tempo. O projetista de casos de teste deve considerar

não somente os casos de teste de caixa preta e branca, mas também o *timing* dos dados e o paralelismo das tarefas (processos) que manipulam os mesmos.

Além disso, a relação íntima que existe entre o software de tempo real e seu ambiente de hardware também pode causar problemas de teste. Os testes de software devem levar em consideração o impacto das falhas de hardware sobre o processamento do software. Tais falhas podem ser extremamente difíceis de serem simuladas realisticamente.

Para o teste abrangente de sistemas de tempo real uma estratégia global de quatro passos pode ser proposta:

3.4.1 Teste de Tarefas

O primeiro passo da atividade de testes de software de tempo real consiste em testar cada tarefa independentemente (nesse contexto, vemos uma tarefa como um programa distinto que aceita entrada e produz saída). Ou seja, testes de caixa preta e de caixa branca são realizados para cada tarefa. Cada tarefa é executada independentemente durantes esses testes. O teste de tarefas revela erros de lógica e de função, mas não revelará erros comportamentais ou de tempo (*timing*).

3.4.2 Teste Comportamental

De acordo com a análise de documentos sobre o comportamento desejado do software, eventos externos (interrupções, sinais de controle, dados, etc) são projetados sental.
u de timing.s esses testes.dentemente (nesse contexto, vemos uma tarefa como um programa distinto que as de testes m os adolam os adosmas tam sistema sem se pro□□□para o teste. Cada um desses eventos é testado individualmente e o comportamento do modelo de sistema e do software executável podem ser comparados a fim de obter conformidade. Logo que cada classe de eventos tenha sido testada, eventos são apresentados ao sistema em ordem e freqüência aleatórias. O comportamento do software é examinado a fim de detectar erros de comportamento.

3.4.3 Teste Inter-Tarefas

Assim que os erros em tarefas individuais e no comportamento do sistema tiverem sido isolados, a atividade de teste desvia-se para os erros relacionados ao tempo. As tarefas

assíncronas, que sabidamente comunicam-se entre si, são testadas com diferentes taxas de dados e cargas de processamento para determinar se ocorrerão erros de sincronização inter-tarefas. Além disso, as tarefas que se comunicam por intermédio de uma fila de mensagens são testadas para descobrir erros na determinação do tamanho dessas áreas de armazenagem de dados.

3.4.4 Teste do Sistema

O software e o hardware são integrados e uma variedade completa de testes de sistema é levada a efeito, numa tentativa de descobrir erros na interface software/hardware.

3.5 Ferramentas de Teste Automatizadas

Uma vez que o teste de software freqüentemente é responsável por até 40% de todo o esforço despendido num projeto de desenvolvimento de software, as ferramentas que podem reduzir o tempo de teste (sem reduzir a eficácia) são muito valiosas. Reconhecendo os benefícios potenciais, pesquisadores e profissionais desenvolveram uma primeira geração de ferramentas de teste automatizadas (Pressman, 1995)

Citando MILLER (1979) e DUNN (1984) temos as seguintes classes principais de ferramentas automatizadas para teste de software:

Analisadores estáticos: Esses sistemas de análise de programa suportam a comprovação de afirmações estáticas, afirmações fracas sobre a estrutura e o formato de um programa.

Auditores de código: Esses filtros de propósito especial são usados para verificar a qualidade do software, a fim de garantir que ele atenda a padrões mínimos de codificação.

Processadores de asserção: Esses sistemas pré-processadores/pós-processadores são empregados para dizer se as afirmações fornecidas pelo ramentas de teste automatizaaaticaeduzir a eficftware, as ferramentas que odem erros relacionados ao temo\□□□□□□□□□□□□□□□□□□□□□programador, denominadas asserções, sobre o comportamento de um programa são de fato cumpridas durante as execuções reais do programa.

Geradores de arquivos de teste: Esses processadores geram, e preenchem com valores previamente determinados, arquivos de entrada típicos para programas que estão em teste.

Geradores de dados de teste: Esses sistemas de análise automatizados auxiliam o usuário a selecionar dados de teste que fazem um programa comportar-se de uma forma particular.

Verificadores de teste: Essas ferramentas medem a cobertura interna dos testes, freqüentemente expressa em termos que estão relacionados à estrutura de controle do objeto de teste, e relatam o valor da cobertura ao especialista em garantia da qualidade.

Comparadores de saída: Esta ferramenta torna possível a comparação de um conjunto de saídas de um programa com outro conjunto (previamente arquivado) para determinar a diferença entre eles.

Simuladores de ambiente: Essa ferramenta é um sistema computadorizado especializado que possibilita que o analista modele o ambiente externo do software de tempo real e depois simule dinamicamente as condições operacionais reais.

Analisadores de fluxo de dados: Essa ferramenta rastreia o fluxo de dados mediante um sistema (semelhante em muitos aspectos aos analisadores de caminhos) e tenta descobrir referências as dados, indexação incorreta e outros erros relacionados a dados.

4 O SMARTMINE

O SmartMine é uma solução integrada de hardware/sofware para melhoramento da produtividade e qualidade em minas de céu aberto. Este sistema armazena e trata cada viagem realizada pelos caminhões, a fim de controlar toda a movimentação de material, seja este minério ou estéril[2]. Ou seja, deseja–se saber entre outras questões, a origem e o destino do material, qual caminhão realizou a viagem, quanto tempo gastou, qual a massa produzida, etc. Esses dados são extremamente importantes para gerar resultados como a produtividade de cada equipamento, a qualidade de cada depósito, tempo ocioso de operação, entre outros. Assim, o SmartMine tem como finalidade, otimizar as operações das minas de céu aberto, através da análise de variáveis concorrentes como a qualidade do minério e a quantidade a ser produzida, evitando-se fila nos carregamentos e basculamentos, diminuindo o tempo ocioso dos equipamentos, entre outros.

O sistema foi projetado para receber informações a todo instante dos equipamentos da mina, através dos computadores de bordo, chamados de *Trackers*. Esses equipamentos podem ser caminhões tradicionais, caminhões fora-de-estrada, escavadeiras, carregadeiras e até mesmo equipamentos terceirizados. O SmartMine interage com esses equipamentos, através do *Tracker*. Através deste computador, o operador do equipamento recebe qual a sua próxima ação, ou seja, qual o novo ponto de basculamento ou carregamento, recebe mensagens de texto, alarmes quando está abaixo ou acima da velocidade permitida, alarmes quando está operando muito próximo a elementos de infra-estrutura da mina (como tubulações, cabos elétricos, etc) e outras informações relevantes ao bom desempenho de seu trabalho. É função deste computador de bordo também, informar a posição de cada equipamento, em um intervalo de tempo configurável, através de um módulo GPS (*Global Positioning System*). Adicionalmente neste computador de bordo, indica-se cada estado do equipamento, a fim de registrar o tempo de cada atividade e a produção de cada equipamento. Dessa maneira, o operador deve indicar qual a atividade está sendo executada no momento, ou caso esteja parado, indicar qual o motivo de parada. Alguns estados são obtidos automaticamente pelo computador de bordo como o estado basculando, obtido do sensor de báscula, ou o estado movimentando cheio, obtido pelo deslocamento posterior ao carregamento, entre outros. Os demais estados devem ser

[2] Estéril: todo o material fora da especificação física ou química aceitável na usina de tratamento de minério, considerando o contexto atual e futuro.

indicados pelo o operador como fila, atolamento, refeição, troca de turno, entre outros diversos estados possíveis. Com o objetivo de garantir o tratamento das informações obtidas, o SmartMine foi projetado em cinco módulos:

- Módulo de Despacho
- Módulo de Planejamento
- Módulo GPS
- Módulo de Consulta
- Módulo de Otimização

4.1 Módulo de Despacho

Como dito anteriormente, o módulo de Despacho (Figura 4.1) é aquele para onde convergem todas as informações em tempo real do processo de produção da mina. Esse módulo possibilita que informações decisivas para o aumento de produtividade e qualidade do produto final cheguem a quem é necessário no tempo e locais corretos.

Através deste módulo, é possível conhecer os locais exatos onde ocorrem os carregamentos, os pontos de basculamento, a produtividade instantânea nos diversos fluxos de produção, a qualidade do minério produzido e outras mais.

4.2 Módulo de Planejamento

O módulo de planejamento (Figura 4.2) é a interface entre os departamentos de planejamento e operação das minas. Neste módulo os usuários podem incluir caminhões, escavadeiras, carregadeiras, britadores, pilhas de estéril, estoques, importar áreas de escavação do plano de curto prazo da mina, definir o mapa da mina, etc. Todas as informações são armazenadas, atualmente, em um servidor de banco de dados *Oracle* ou *SQLServer*.

4.3 Módulo GPS

O módulo de GPS torna possível coletar informações relevantes em qualquer lugar, a qualquer momento. Este módulo é composto de diversos componentes em *hardware* e *software* embarcados, bem como uma rede sem fio, possibilitando que todos os equipamentos móveis, como escavadeiras, caminhões, perfuratrizes e demais estejam integrados ao sistema em tempo real.

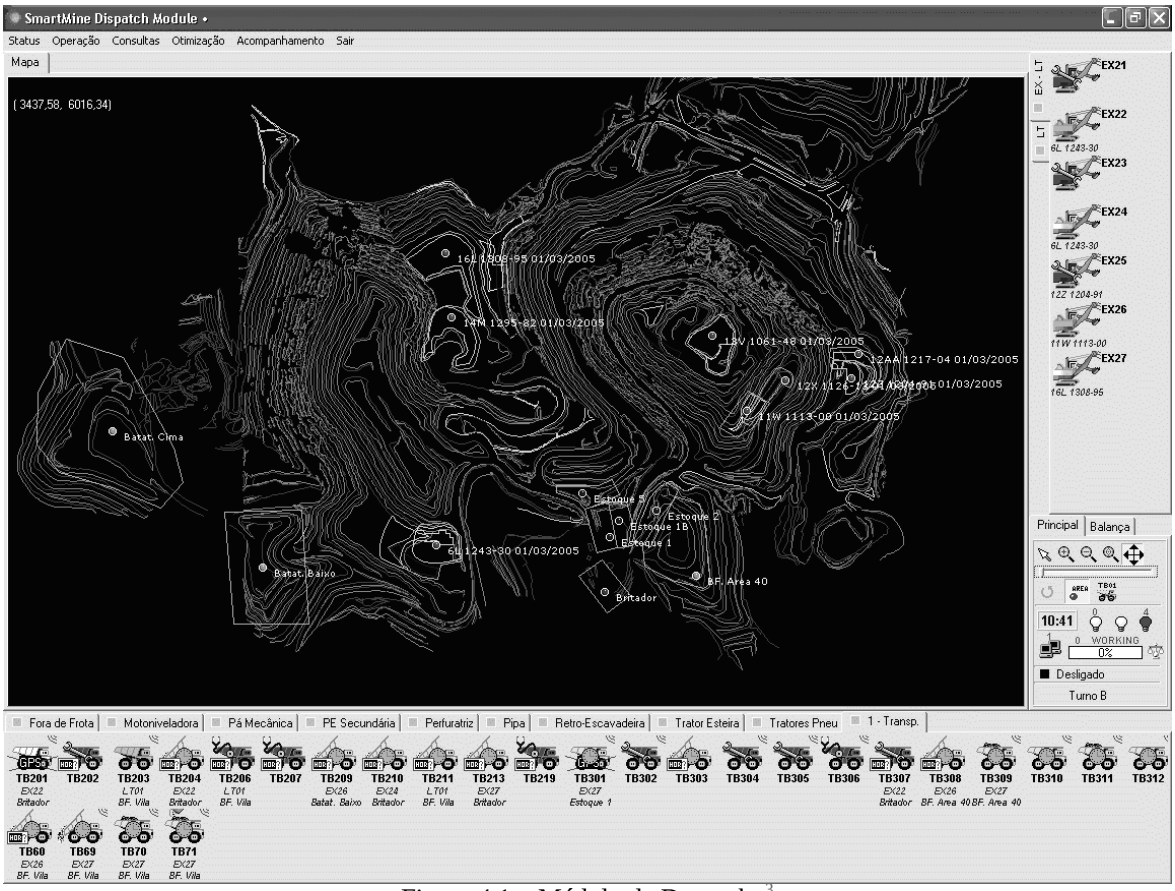

Figura 4.1 – Módulo de Despacho[3]

[3] As imagens utilizadas neste trabalho são de propriedade da Devex Tecnologia e Sistemas Ltda, e sua publicação neste trabalho foram autorizadas pela mesma. Não é permitido cópias ou reproduções das mesmas sem prévia autorização.

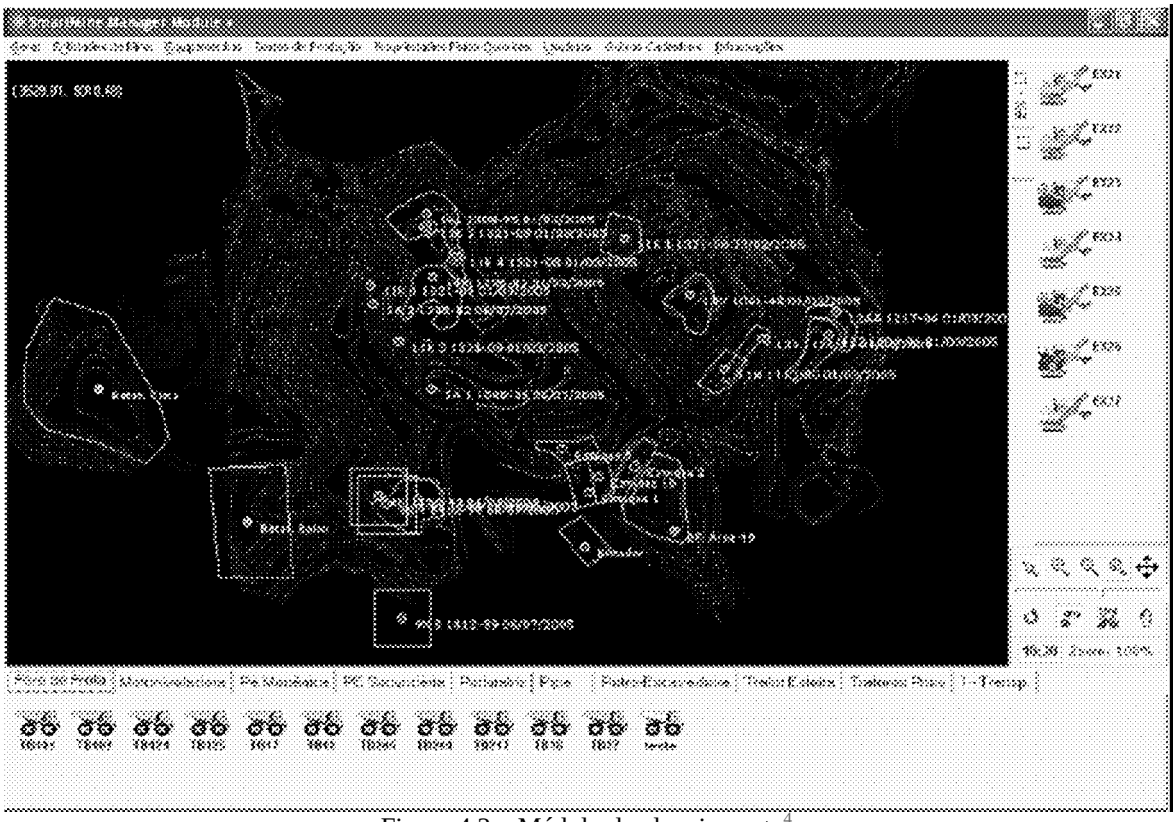

Figura 4.2 – Módulo de planejamento[4]

4.4 Módulo de consulta

O módulo de consulta é uma réplica da interface do módulo de controle. Assim o usuário pode visualizar tudo o que está ocorrendo na mina, mas não possui as funções de realizar comandos na operação. Este módulo pode ser acessado pelos usuários autorizados através da W*eb*, onde podem ser realizadas diversas consultas no sistema. A Figura 4.3 mostra a tela inicial do módulo de consulta.

[4] As imagens utilizadas neste trabalho são de propriedade da Devex Tecnologia e Sistemas Ltda, e sua publicação neste trabalho foram autorizadas pela mesma. Não é permitido cópias ou reproduções das mesmas sem prévia autorização.

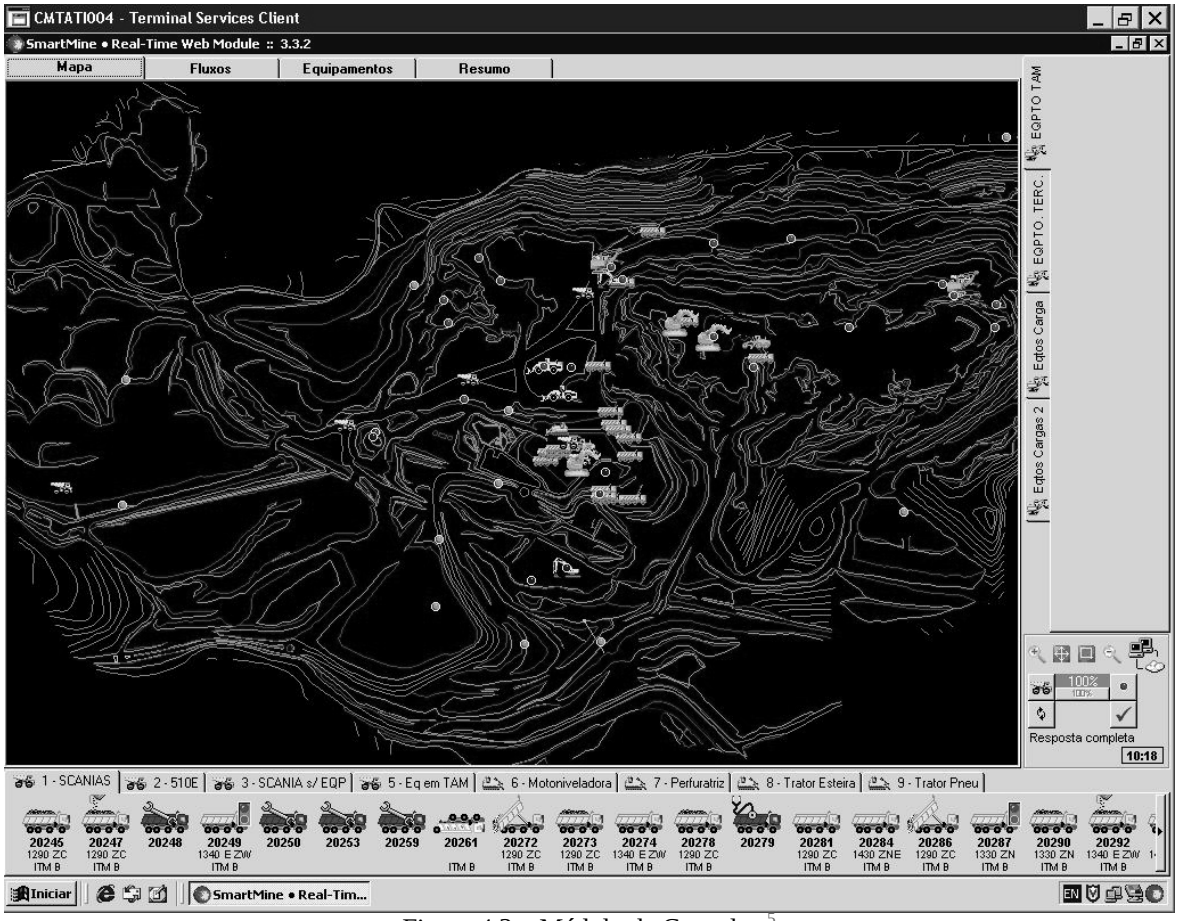

Figura 4.3 – Módulo de Consulta.[5]

4.5 Módulo de Otimização

O módulo de otimização recebe a todo instante os eventos da mina do módulo de despacho. Assim este módulo pode realizar a simulação de operação da mina nas próximas horas, resultando em uma sugestão da próxima ação do operador do módulo de despacho.

4.6 Como Eram Realizados os Testes Antes do Simulador

Antes do desenvolvimento do *SMAccessPoint Simulator* os testes do Despacho dependiam de equipamentos de hardware para serem realizados. Normalmente, na sede da empresa em Belo Horizonte, havia uma "bancada de testes", onde um *tracker* podia ser usado para enviar dados ao sistema.

[5] As imagens utilizadas neste trabalho são de propriedade da Devex Tecnologia e Sistemas Ltda, e sua publicação neste trabalho foram autorizadas pela mesma. Não é permitido cópias ou reproduções das mesmas sem prévia autorização.

A desvantagem dessa metodologia de testes é que nem sempre esta bancada de testes estava disponível, ou não havia a funcionalidade requerida implementada no tracker. Como apenas um *tracker* era usado, não era possível simular a interação entre diferentes equipamentos, o que limitava o escopo do teste.

Outra grande desvantagem é que não existia uma estrutura de testes como a citada acima na sede da empresa em Lavras. Então, quando uma funcionalidade nova era implementada, era necessário que os desenvolvedores de Belo Horizonte a testassem. Como não havia sempre disponibilidade de pessoal para realização dos testes com o hardware, algumas vezes, a integração com o computador de bordo não podia ser realizada no momento do desenvolvimento da funcionalidade.

5 METODOLOGIA

Este trabalho foi desenvolvido durante meu período de estágio na empresa Devex Tecnologia e Sistemas Ltda, no período de Agosto de 2005 a Dezembro de 2005.

5.1 Definição do Objetivo do Trabalho

O objetivo do trabalho foi definido após conversas e reuniões com os Gerentes de Módulos na Devex, sobre a necessidade de se testar mais eficazmente o sistema durante a fase de desenvolvimento.

O objetivo definido para o trabalho foi desenvolver uma ferramenta de teste para o módulo de despacho do SmartMine, sendo este o mais crítico, devido à dificuldade de se testar o sistema, como já dito anteriormente, durante a fase de desenvolvimento. Essa ferramenta deveria simular o ambiente externo ao sistema (a mina em funcionamento) e fornecê-lo dados necessários para sua execução. Em outras palavras, era necessário desenvolver um simulador de operação de mina para testar o sistema.

A partir daqui o Módulo de Despacho do SmartMine será referenciado apenas como Despacho.

Para exemplificar melhor o software a ser desenvolvido, têm-se as duas figuras abaixo. A Figura 5.1 representa o sistema funcionando na estrutura real de uma mina. A Figura 5.2 evidencia o uso do *SMAccessPoint Simulator* para substituir o ambiente da mineração no envio de dados para o Despacho.

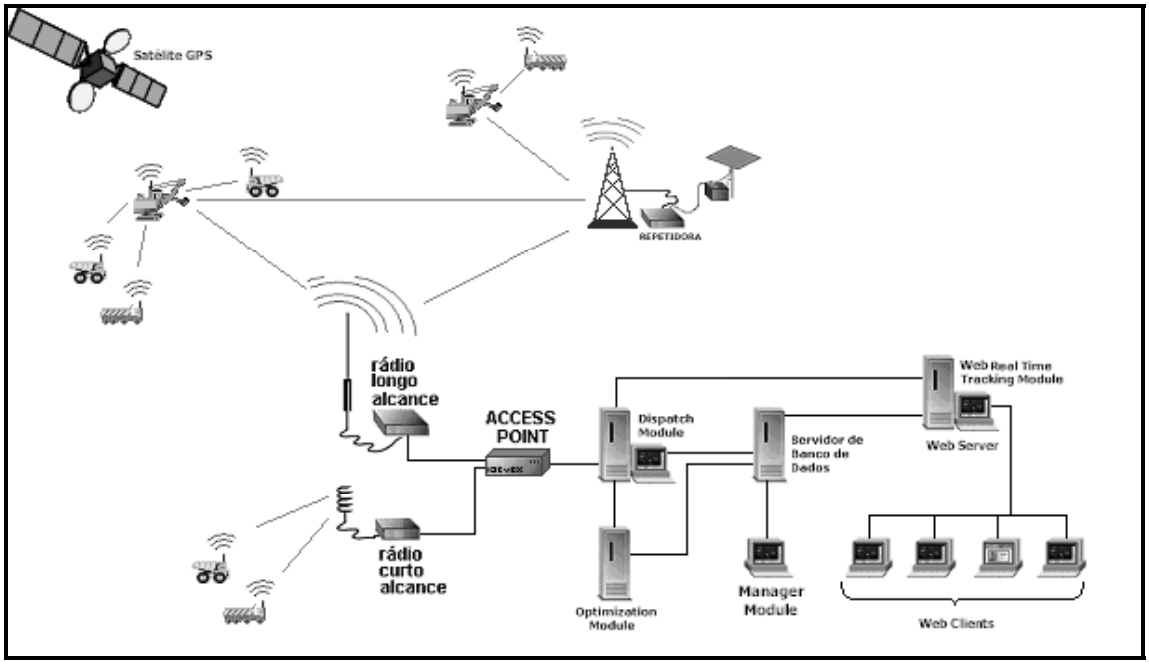

Figura 5.1 – Estrutura física real de uma mina.

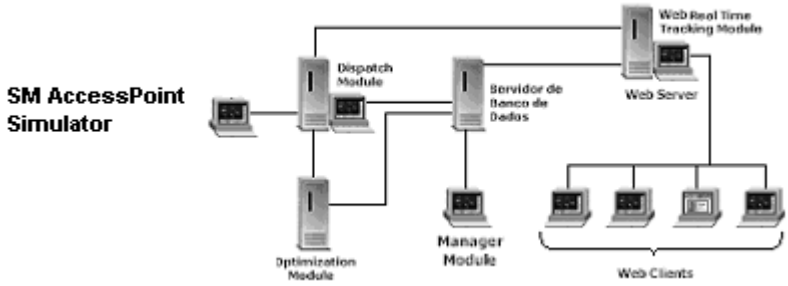

Figura 5.2 – SMAccessPoint Simulator usado para simular o funcionamento da mina

5.2 Concepção do Software

Logo após a definição do objetivo do trabalho, iniciou-se uma pesquisa sobre testes de software e sistemas de tempo real. Foram pesquisados livros e artigos, principalmente através da Internet. Como o ambiente de programação a ser usado seria o *Borland Delphi 5* (referenciado a partir deste ponto apenas como Delphi*)*, também foi necessário estudar e avaliar o suporte oferecido pela linguagem à aplicações que transmitem dados através de uma rede de computadores. Esse estudo foi realizado tanto procurando material na Internet quanto conversando com pessoas mais experientes na área, outros desenvolvedores da Devex, e assimilando seus conhecimentos.

Após o levantamento do referencial teórico foram realizadas algumas reuniões, via *MSN Messenger* (Software de comunicação pela Internet baseado em texto) *e Skype*

(Software de comunicação na Internet baseado em Voz Sobre IP ou VoIP – "*Voice Over IP*"), para realizar o levantamento dos requisitos iniciais do sistema. Para uma primeira versão do software, foram especificados os seguintes requisitos (que serão detalhados no decorrer do trabalho):

- O Sistema deveria ser capaz de se conectar a qualquer banco de dados de uma Mina, presente na Devex, para carregar os dados necessários para seu funcionamento. Dentre esses dados encontram-se equipamentos de transporte, equipamentos de carga, áreas de mineração (também chamadas entidades de mina, são os locais de onde se extrai minério), dados sobre a alocação dos equipamentos, etc.
- O Simulador deveria ser capaz de se conectar ao Despacho para enviar os dados gerados.
- Cada equipamento simulado no software deveria enviar, num intervalo de tempo estipulado, um pacote de status para o Despacho. Nesse pacote estão contidas algumas informações sobre o equipamento.
- Definir um destino de movimentação para um equipamento. O operador do *SMAccessPoint Simulator* deveria ser capaz de escolher um equipamento em particular e definir um destino de movimentação para o mesmo, indicando as coordenadas desse. O equipamento então se deslocaria até o destino especificado com uma velocidade parecida com a velocidade real dos equipamentos de mina.
- Definir posição do equipamento. O usuário poderia escolher um equipamento qualquer e definir uma posição no espaço para o mesmo, especificando as coordenadas desejadas. O equipamento deveria então ser transferido imediatamente para o local especificado pelo usuário.
- Deslocar equipamento dez metros. O usuário poderia simplesmente escolher um equipamento e deslocá-lo 10 metros em alguma das oito direções cardeais principais.
- Definir velocidade de um equipamento. O usuário poderia escolher um equipamento qualquer e definir a velocidade com a qual o mesmo deve se deslocar.
- Envio de evento de *login* do operador.
- Envio de evento de *logout* de operador.

- Envio de evento de troca de estados do equipamento.
- Envio de evento de carga efetuada.
- Invalidar comunicação do equipamento com o GPS.
- Validar comunicação do equipamento com o GPS.
- Desabilitar comunicação com o GPS.
- Habilitar comunicação com o GPS.
- Desabilitar comunicação do equipamento com o Access Point.
- Habilitar comunicação do equipamento com o Access Point.
- Sobrecarga de pacotes.
- Recebimento de comando do Despacho.
- Recebimento de mensagem de texto.
- Execução de arquivo de programação.

Após a análise dos requisitos, decidiu-se que no sistema deveriam ser modelados e implementados os seguintes elementos principais:

- ***Access Point*** – Dispositivo responsável por capturar os dados recebidos da rede sem fio e transmiti-los ao Despacho através da rede Ethernet.
- ***Tracker*** – Computador de bordo presente nos equipamentos da mina responsável pela interface entre o Despacho e o operador do equipamento.
- **Pacotes de rede** – existentes em vários formatos padronizados pela Devex.

Existem numerosas outras classes de objetos definidas e implementadas para que o sistema funcione corretamente, mas é a integração entre os três elementos citados acima que gera os dados necessários para a simulação da mina e o envio de dados ao Despacho.

O Sistema foi implementado usando o paradigma da orientação a objetos e estruturado em três camadas:

- Camada de interface
- Camada de processamento
- Camada de acesso a dados

As camadas foram divididas apenas logicamente no código fonte do software, sendo que todas fazem parte do mesmo executável.

5.3 Implementação dos Requisitos

O foco principal deste trabalho foi a camada de processamento. A preocupação era garantir que os pacotes de rede seriam enviados corretamente ao Despacho. A camada de acesso a dados é usada somente na inicialização do sistema, o que a torna bem simples e pontual. A camada de interface foi desenvolvida de modo que fornecesse apenas o mínimo necessário para entrada de dados a serem enviados, portanto conceitos de estética foram deixados de lado nesta versão do trabalho.

Para execução desta parte do trabalho foi necessário estudar o código fonte do Despacho que trata da comunicação com os dispositivos. Um processo de engenharia reversa foi usado para definir o formato dos pacotes a serem enviados do simulador para o Despacho.

5.3.1 Conexão com Banco de Dados

Para que o sistema seja capaz de conectar-se a qualquer banco de dados sem que seja necessário alterar o código fonte do mesmo, decidiu-se que a configuração da conexão seria definida em um arquivo de texto, com a extensão to tipo *".ini"*, chamado SMAccessPointSimulator.ini. Nesse arquivo estão presentes os seguintes dados:

- *Alias* do banco de dados.
- *Owner* das tabelas do banco de dados.
- Usuário para o banco de dados.
- Nome do *driver* usado para conexão com o banco de dados.

Uma classe de leitura do arquivo de configuração da conexão foi implementada e sempre que o sistema inicia, os dados da conexão são lidos do arquivo e a mesma é estabelecida.

Quando o software é iniciado e a conexão é estabelecida, os seguintes dados são consultados no banco de dados para criar os objetos representativos no sistema:

- Entidades de mina ativas.
- Dispositivos *AccessPoints* cadastrados.
- Equipamentos de carga ativos que possuam alguma alocação.
- Equipamentos de transporte ativos.
- Estados de operação possíveis para cada equipamento.

Na época em que este trabalho foi escrito, o sistema era capaz de conectar-se somente com o banco de dados *Oracle*.

5.3.2 Envio de Dados para o Despacho

Para implementação deste requisito foi necessário, inicialmente, estudar como a conexão entre Despacho e *Access Point* é realizada e qual o formato de mensagens no SmartMine.

Comunicação entre Despacho e *Access Point*

Resumidamente, a comunicação entre o Despacho e os *Access Points* é realizada através de uma rede Ethernet 10/100 Mbps baseada no protocolo TCP/IP. Tendo em vista que o Despacho deve estar preparado para se comunicar com quantos *Access Points* forem necessários. A conexão é baseada no paradigma cliente–servidor, para que o Despacho possa gerenciar todos os dispositivos conectados a ele. O Despacho assume a posição de servidor e pode ter quantos clientes, *Access Points*, forem necessários. A Figura 5.3 exemplifica esse processo.

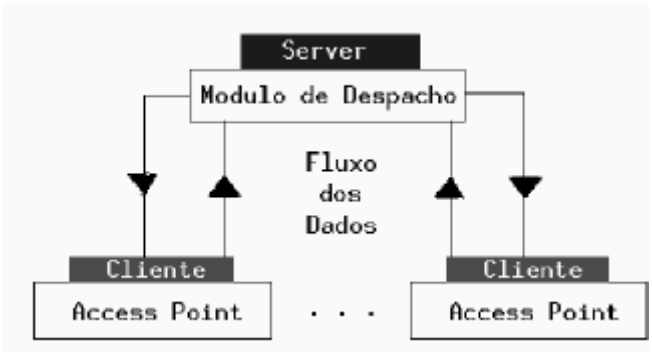

Figura 5.3 – Modelagem da conexão entre o Despacho e os dispositivos *AccessPoints*.

Assim como no modelo cliente-servidor, o Despacho não conhece o endereço (IP) de cada *Access Point* na rede Ethernet até que este solicite conexão. O que significa que o Despacho deve possuir um endereço IP fixo na rede ou que, pelo menos, a máquina em que este módulo é executado esteja registrada no DNS do servidor de rede.

Dessa maneira, parte do *Access Point* a iniciativa de solicitar uma conexão para transmitir seus dados, e cabe ao Despacho validar a conexão e gerenciá–la, tomando as devidas decisões na presença de erros. O *Access Point* foi modelado para tentar estabelecer uma conexão com o servidor em um intervalo de tempo configurável. Ao conseguir uma conexão, mesmo que não tenha dados a serem enviados, a conexão permanece até que o

servidor encerre a conexão, ou que o *Acess Point* "desconfie" que a conexão está com problemas e tente se reconectar. No melhor dos casos, cada *Access Point* se conecta apenas uma vez e mantém sua conexão.

A identificação de cada *Access Point* pelo Despacho é realizada através de um número inteiro, único por *dispositivo*, fornecido por este ao conectar–se ao módulo. Assim, cada número é associado a uma conexão. Cada conexão é um objeto do tipo "*socket*" implementado pelos sistemas operacionais, e que guarda informações como o endereço de rede de cada *Access Point*. O Despacho gerencia apenas estes objetos "*socket*", deixando o trabalho de transmissão e recebimento de dados dos *Access Points* certos, para o sistema operacional.

Tanto os clientes *Access Points* quanto o servidor Despacho, enviam uma mensagem *WatchDog* se algum deles perceber que nenhuma mensagem foi enviada em um intervalo de tempo *t*. Assim, se o cliente ou o servidor forem o receptor e perceber que nenhuma mensagem chegou pela conexão em um intervalo de tempo *t*, a conexão é encerrada pelo lado que percebeu o *TIMEOUT*. Cabe ao cliente solicitar nova conexão. Neste caso, sempre o *Access Point*.

Formato das mensagens

O formato das mensagens do protocolo de comunicação do SmartMine, pode ser visualizado na Figura 5.4.

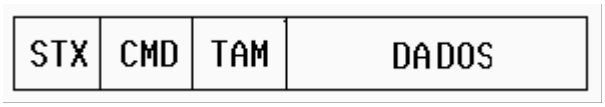

Figura 5.4 – Formato da mensagem

O campo *STX* (start of text) indica o início da mensagem e possui sempre o valor dois. Este valor foi escolhido aleatoriamente para significar início de mensagem. Portanto este campo possui apenas um byte.

O campo *CMD*, abreviação de *Comando*, indica qual o tipo da mensagem e possui também, apenas um byte. Este campo indica se a mensagem é um *WatchDog*, uma mensagem de dados, uma mensagem de controle, entre outras.

O campo *TAM*, abreviação de *Tamanho*, indica a quantidade de bytes a serem lidos do campo *Dados*. Também possui um byte. Assim, o cabeçalho de qualquer mensagem

trocada entre o *Access Point* e o Módulo de Despacho, devem conter estes três campos nesta ordem, o que significa que toda mensagem possui um *overhead* de três bytes.

Implementação da conexão

Como visto acima, o Despacho funciona como um servidor no modelo de comunicação cliente-servidor. Então para que o *SMAccessPoint Simulator* pudesse enviar dados ao módulo de controle, esse precisava fazer o papel de cliente no modelo citado.

Para que o simulador funcionasse como um cliente, foi necessário utilizar um componente, fornecido pelo Delphi, chamado *TClientSocket*. Ele fornece os métodos de conexão, envio e recebimento dos dados através de *sockets*.

A classe responsável por realizar a conexão com o Despacho representa o dispositivo *Access Point* do mundo real e é chamada TBaseAccessPoint.

O Dispositivo *Access Point* possui apenas a função de receber pacotes da rede sem fio, utilizada pelos computadores de bordo dos equipamentos de mina, e transmiti-los ao Despacho através da rede Ethernet. Nenhum processamento é realizado sobre os pacotes recebidos ou enviados, funcionando apenas como um *gateway* entre as duas redes.

A função da classe criada é, como no mundo real, apenas receber pacotes de rede de outros objetos e transmiti-los ao Despacho. A cada 5 segundos, se nenhum pacotes de rede foi enviado, esta envia um pacote *WatchDog* apenas para manter a conexão com o Despacho ativa.

O pacote WatchDog tem o seguinte formato:

- STX: Com o valor dois.
- CMD: Com o valor 128.
- TAM: Com o valor dois.
- AccessPointId: Dois bytes contendo o código numérico do Access Point.

5.3.3 Envio de Pacotes de Status para o Despacho

Pacotes de status são enviados ao Despacho para fornecer informações sobre os equipamentos da mina. Essas informações são armazenadas e enviadas pelo computador de bordo do equipamento. Para que o software simulasse os equipamentos da mina, foi implementada a classe TBaseIHM.Cada equipamento ativo no banco de dados representa uma instância dessa classe. Ela contém os atributos:

- **Smt**: Número do computador de bordo. Abreviação para *SmartMine Tracker*.
- **Nome do equipamento**: Nome do equipamento como cadastrado pelo módulo de planejamento da mina.
- **Número de matrícula**: Armazena o número de matrícula do operador que está autenticado no computador de bordo.
- **Posição leste base**: Coordenada leste em que o equipamento realmente se encontra.
- **Posição norte base**: Coordenada norte em que o equipamento realmente se encontra.
- **Posição leste a ser enviada**: Coordenada leste do equipamento a ser enviada para o Despacho considerando-se o erro de precisão do GPS.
- **Posição norte a ser enviada:** Coordenada norte do equipamento a ser enviada para o Despacho considerando-se o erro de precisão do GPS.
- **Posição leste de destino**: Coordenada leste do destino de movimentação do equipamento.
- **Posição norte de destino**: Coordenada norte do destino de movimentação do equipamento.
- **Altitude do equipamento**: Em metros, a partir do nível do mar.
- **Velocidade**: Em km/h.
- **Direção**: Indica qual a direção, em graus, do equipamento.
- **Último comando**: Identificador do último comando recebido
- **Estado atual**: Código do estado atual do equipamento.
- **Último estado**: Código do último estado do equipamento.
- **GPS válido**: Booleano indicando se o GPS está enviando as coordenadas corretamente.
- **GPS acessível**: Booleano indicando se o computador de bordo está comunicando com o hardware de GPS.
- **Access Point acessível**: Booleano indicando se o computador de bordo está comunicando com o *Access Point*.

O pacote de status tem a seguinte estrutura:

- **STX**: com o valor dois.

- **CMD**: com o valor zero.
- **TAM**: com o valor 29.
- **Código do computador de bordo**: dois bytes contendo o código. Dependente da instância do objeto. Cada computador de bordo possui um código cadastrado no banco de dados.
- **Rota**: dois bytes contendo o código da rota do equipamento. Atualmente o código da rota não é usado no SmartMine. Esse campo está neste pacote apenas para manter o formato do pacote. O valor é aleatório entre quinhentos e seiscentos.
- **Coordenada leste do equipamento**: quatro bytes indicando a coordenada do equipamento em coordenadas locais da mina, não em coordenadas geográficas. Essas coordenadas representam a distância, em centímetros, de um ponto qualquer ao ponto zero da mina. Esse ponto zero é definido pelo planejamento da mina. O erro de precisão do GPS já está embutido no valor dessa coordenada quando a mesma é enviada ao Despacho. O erro de precisão será discutido mais à frente.
- **Coordenada norte do equipamento**: exatamente como acima, mas representando a coordenada norte.
- **Altitude**: dois bytes representando a altitude, em metros, do equipamento em relação ao nível do mar.
- **Velocidade**: um byte representando a velocidade do equipamento em km/h. O cálculo da velocidade será discutido posteriormente.
- **Direção**: um byte representando a direção do equipamento. Será explicado mais a frente como é calculada a direção.
- **Estado atual**: quarto bytes com o código atual do estado do equipamento.
- **Tamanho do *buffer***: dois bytes indicando a quantidade de eventos armazenados na memória do computador de bordo. Na versão atual do *SMAccessPoint Simulator*, a memória do computador de bordo não está sendo utilizada, por isso sempre é enviado o valor zero neste campo.
- **Último comando recebido**: quatro bytes representando o código do último comando recebido do Despacho. Na versão atual o simulador não está

recebendo comandos do Despacho, por isso o valor zero sempre é enviado neste campo.

- **Potência do rádio**: um byte representando uma escala de potência do sinal de rádio da rede sem fio. Um valor aleatório entre zero e 255 é enviado.
- ***Flags***: um byte representando *flags* adicionais que podem ser enviadas. No momento o Despacho não utiliza essas *flags*, portanto o valor zero sempre é enviado.

Para enviar as coordenadas do equipamento, é verificado se o GPS está enviando as coordenadas corretamente ao computador de bordo (o próprio GPS indica quando seus dados são inválidos). Neste caso, de comunicação inválida, o valor 2147483647 (máximo valor suportado pelo tipo *longint* do Delphi, que é o tipo das variáveis que representam as coordenadas) é enviado tanto para coordenada leste quanto para a coordenada norte. Caso o GPS esteja enviando as coordenadas corretamente ainda é necessário simular o erro de precisão do GPS. Para o simulador foi estabelecido que o GPS teria um erro de precisão de até três metros. Portanto, quando uma coordenada é enviada, usa-se seu valor base mais um valor aleatório entre zero e trezentos (três metros em centímetros) para simular o referido erro.

A velocidade média de um equipamento, no simulador, é de 40 km/h. Quando a velocidade é enviada, usa-se esse valor mais um ajuste aleatório de até 10 km/h.

A direção do equipamento é calculada considerando a mina como um plano cartesiano e verificando o ângulo formado entre as coordenadas do equipamento e as coordenadas do destino de movimentação do mesmo.

A altitude do equipamento é um valor aleatório entre zero e cem metros. Esse valor foi convencionado durante a especificação do software.

Cada computador de bordo envia um pacote de status ao Despacho a cada cinco segundos.

5.3.4 Definir um Destino de Movimentação para o Equipamento

O usuário pode definir um destino de movimentação para um equipamento selecionando-o na interface gráfica e informando as coordenadas leste e norte do equipamento. As coordenadas precisam ser informadas em centímetros.

Os atributos que representam as coordenadas de destino do equipamento são atualizados e, a cada segundo subseqüente, este se deslocará em linha reta até seu objetivo. O equipamento se deslocará a uma velocidade de 40 km/h (11,11 m/s) mais uma variação de até 10 km/h (2,77 m/s) até que as suas coordenadas atuais e de destino sejam iguais.

A posição do equipamento é enviada para o Despacho no pacote de status. Portanto, sempre que o Despacho recebe um pacote de status de um equipamento, a posição do mesmo é atualizada e mostrada ao usuário.

5.3.5 Definir Posição do Equipamento

O usuário pode definir a posição do equipamento selecionando-o na interface gráfica e informando as coordenadas leste e norte para o mesmo. Estas precisam ser informadas em centímetros.

Os atributos que representam as coordenadas bases do equipamento são atualizadas com os valores informados, o que faz com que o equipamento seja automaticamente transferido para o local desejado. Da próxima vez que o equipamento enviar um pacote de status para o Despacho, este atualizará sua posição no mapa e mostrará ao usuário.

5.3.6 Deslocar Equipamento Dez Metros

O usuário pode definir a posição do equipamento selecionando-o na interface gráfica e escolhendo uma entre as oito direções possíveis: N, NE, E, SE, S, SW, S e NW. O equipamento então se deslocará 10 metros na direção escolhida, exatamente da mesma maneira como no requisito anterior.

Este é um modo de posicionar um equipamento sem ter de digitar o valor das coordenadas desejadas. Também neste requisito, a posição do equipamento será atualizada no Despacho quando este receber um pacote de status do *tracker* do equipamento.

5.3.7 Definir Velocidade de um Equipamento

O usuário pode selecionar um equipamento de transporte através da interface gráfica e digitar um novo valor de velocidade para o mesmo. A velocidade informada estará em km/h. O equipamento passará a movimentar-se com a nova velocidade a partir de então.

5.3.8 Envio de Evento de *Login* para o Despacho

Para que o usuário simule um evento de *login* no computador de bordo do equipamento é necessário que não haja nenhum usuário autenticado no mesmo. Para realizar o *login*, basta selecionar um equipamento na interface gráfica e informar um número de matrícula válido. O número de matrícula pode ser obtido consultando-se o banco de dados da mina que está sendo simulada.

Quando o *login* é realizado, o atributo referente ao número de matrícula no computador de bordo é atualizado e este é considerado como já possuindo um operador.

O pacote de *login* tem a seguinte estrutura:

- **STX**: com o valor dois.
- **CMD**: com o valor um.
- **TAM**: com o valor 15.
- **Código do evento**: com valor oito.
- **Código do computador de bordo**: dois bytes contendo o código. Dependente da instância do objeto. Cada computador de bordo possui um código cadastrado no banco de dados.
- **Identificador do evento**: quatro bytes representando o identificador do evento para futura confirmação de recebimento pelo Despacho.
- **Instante (UTC)**: quatro bytes representando o instante em que ocorreu o evento.
- **Número de matrícula do operador**: quatro bytes representando o número de matrícula do operador que está se autenticando no computador de bordo.

5.3.9 Envio de Evento de *Logout* para o Despacho

Para que o usuário simule um evento de *logout* no computador de bordo do equipamento é necessário que um usuário já esteja autenticado no mesmo. Para realizar o *logout*, basta selecionar um equipamento na interface gráfica e disparar o evento clicando em um botão criado para esse fim.

Quando o *logout* é realizado, o atributo referente ao número de matrícula no computador de bordo é atualizado e este é considerado como não possuindo nenhum operador.

O pacote de login tem a seguinte estrutura:

- **STX**: com o valor dois.
- **CMD**: com o valor um.
- **TAM**: com o valor 15.
- **Código do evento**: com valor nove.
- **Código do computador de bordo**: dois bytes contendo o código. Dependente da instância do objeto. Cada computador de bordo possui um código cadastrado no banco de dados.
- **Identificador do evento**: quatro bytes representando o identificador do evento para futura confirmação de recebimento pelo Despacho.
- **Instante (UTC)**: quatro bytes representando o instante em que ocorreu o evento.
- **Número de matrícula do operador**: quatro bytes representando o número de matrícula do operador que está autenticado no computador de bordo.

5.3.10 Envio de Evento de Troca de Estados do Equipamento

Estado de um equipamento significa estado de operação do mesmo. Por exemplo, um caminhão pode estar movimentando vazio, movimentando cheio, carregando, atolado, em fila para basculamento, etc. Todos os citados anteriormente são estados do equipamento. Estes estados são cadastrados pelo planejamento de mina no módulo de Planejamento.

Para alterar o estado de um equipamento é necessário selecioná-lo na interface gráfica. Quando um equipamento é selecionado, a lista de estados possíveis é atualizada para o equipamento em questão. Portanto, para trocar de estado, basta selecionar um equipamento e escolher qual o novo estado do mesmo.

O pacote de troca de estados tem a seguinte estrutura:

- **STX**: com o valor dois.
- **CMD**: com o valor um.
- **TAM**: com o valor 29.
- **Código do computador de bordo**: dois bytes contendo o código. Dependente da instância do objeto. Cada computador de bordo possui um código cadastrado no banco de dados.

- **Identificador do evento**: quatro bytes representando o identificador do evento para futura confirmação de recebimento pelo Despacho.
- **Código do evento**: um byte representando o código do evento. Com o valor um.
- **Instante (UTC)**: quatro bytes representando o instante em que ocorreu o evento.
- **Coordenada leste do equipamento**: quatro bytes indicando a coordenada do equipamento em coordenadas locais da mina, não em coordenadas geográficas.
- **Coordenada norte do equipamento**: exatamente como acima, mas representando a coordenada norte.
- **Altitude**: dois bytes representando a altitude, em metros, do equipamento em relação ao nível do mar.
- **Estado anterior**: quatro bytes representando o código do estado anterior do equipamento.
- **Estado atual**: quatro bytes representando o código do novo estado do equipamento.

5.3.11 Envio de Evento de Carga Efetuada para o Despacho

Este pacote indica que determinado equipamento de transporte foi carregado. Na interface gráfica é possível escolher o equipamento de transporte a ser carregado, o equipamento de carga que realizou o carregamento e o tipo de material carregado (minério ou estéril).

O pacote de carga efetuada tem a estrutura:

- **STX**: com o valor dois.
- **CMD**: com o valor um.
- **TAM**: com o valor 24.
- **Código do computador de bordo**: dois bytes contendo o código. Dependente da instância do objeto. Cada computador de bordo possui um código cadastrado no banco de dados.

- **Identificador do evento**: quatro bytes representando o identificador do evento para futura confirmação de recebimento pelo Despacho.
- **Código do evento**: um byte com o valor sete.
- **Instante (UTC):** quatro bytes representando o instante em que ocorreu o evento.
- **Coordenada leste do equipamento**: quatro bytes indicando a coordenada do equipamento em coordenadas locais da mina, não em coordenadas geográficas.
- **Coordenada norte do equipamento**: exatamente como acima, mas representando a coordenada norte.
- **Altitude**: dois bytes representando a altitude, em metros, do equipamento em relação ao nível do mar.
- **Código do equipamento de carga**: dois bytes representando o código do equipamento de carga que realizou o carregamento.
- **Tipo de material**: um byte indicando se o carregamento foi de minério ou estéril.

Estéril é o material fora da especificação física ou química aceitável na usina de tratamento de minério.

5.3.12 Invalidar Comunicação do Equipamento com o GPS.

No mundo real ocorre quando o equipamento está comunicando com o GPS, mas não há condições para o cálculo de uma localização precisa. Por exemplo:

O equipamento não consegue comunicar com um número suficiente de satélites para uma localização exata.

O equipamento está em uma área onde a comunicação com os satélites é interrompida. Como exemplo, garagens cobertas, áreas com interferências climáticas, etc.

Para invalidar a comunicação de um equipamento com o GPS, basta selecioná-lo na interface gráfica e clicar em um botão criado para tal.

O pacote de comunicação com o GPS inválida tem a seguinte estrutura:

- **STX**: com o valor dois.
- **CMD**: com o valor um.

- **TAM**: com o valor 28.
- **Código do computador de bordo**: dois bytes contendo o código. Dependente da instância do objeto. Cada computador de bordo possui um código cadastrado no banco de dados.
- **Identificador do evento**: quatro bytes representando o identificador do evento para futura confirmação de recebimento pelo Despacho.
- **Código do evento**: um byte com o valor 134.
- **Instante (UTC)**: quatro bytes representando o instante em que ocorreu o evento.
- **Coordenada leste do equipamento**: quatro bytes indicando a coordenada do equipamento em coordenadas locais da mina, não em coordenadas geográficas.
- **Coordenada norte do equipamento**: exatamente como acima, mas representando a coordenada norte.
- **Altitude**: dois bytes representando a altitude, em metros, do equipamento em relação ao nível do mar.
- **Número de matrícula do operador**: quatro bytes representando o número de matrícula do operador que está autenticado no computador de bordo.
- **Tipo do evento**: um byte com o valor cinco.
- **Argumentos**: quatro bytes não usados atualmente. O valor zero é enviado. Este campo está no pacote para manter a compatibilidade com o SmartMine.

Quando um computador de bordo tem a comunicação com o GPS invalidada, seu atributo que representa a validade de sua comunicação com o GPS assume o valor falso. A partir de então, as coordenadas leste e norte do equipamento, nos pacotes enviados ao Despacho, assumem o valor 2147483647, como explicado anteriormente. Essas coordenadas serão enviadas até que a comunicação com o GPS volte a se estabilizar.

5.3.13 Validar Comunicação do Equipamento com o GPS.

Ocorre quando o equipamento estava com comunicação com o GPS inválida e voltou a comunicar-se corretamente.

Para validar a comunicação de um equipamento com o GPS, basta selecionar um que esteja com a comunicação inválida e clicar no botão criado para tal.

O pacote de validação da comunicação com o GPS tem a estrutura:

- **STX**: com o valor dois.
- **CMD**: com o valor um.
- **TAM**: com o valor 28.
- **Código do computador de bordo**: dois bytes contendo o código. Dependente da instância do objeto. Cada computador de bordo possui um código cadastrado no banco de dados.
- **Identificador do evento**: quatro bytes representando o identificador do evento para futura confirmação de recebimento pelo Despacho.
- **Código do evento**: um byte com o valor 134.
- **Instante (UTC)**: quatro bytes representando o instante em que ocorreu o evento.
- **Coordenada leste do equipamento**: quatro bytes indicando a coordenada do equipamento em coordenadas locais da mina, não em coordenadas geográficas.
- **Coordenada norte do equipamento**: exatamente como acima, mas representando a coordenada norte.
- **Altitude**: dois bytes representando a altitude, em metros, do equipamento em relação ao nível do mar.
- **Número de matrícula do operador**: quatro bytes representando o número de matrícula do operador que está autenticado no computador de bordo.
- **Tipo do evento**: um byte com o valor três.
- **Argumentos**: quatro bytes não usados atualmente. O valor zero é enviado. Este campo está no pacote para manter a compatibilidade com o SmartMine.

Quando a comunicação do computador de bordo do equipamento com o GPS estabiliza-se novamente, seu atributo que representa a validade da comunicação com o GPS assume o valor verdadeiro. Nesse caso as coordenadas leste e norte serão enviadas com seu valor real nos pacotes para o Despacho.

5.3.14 Desabilitar Comunicação com o GPS.

No mundo real ocorre quando não é possível estabelecer uma conexão entre o computador de bordo do equipamento e o hardware de GPS instalado no mesmo. Por exemplo, quando há um rompimento no cabo que liga os dois aparelhos.

Quando a comunicação com o GPS é desabilitada, o atributo do computador de bordo que representa a comunicação com o GPS assume o valor falso. Neste caso, as coordenadas leste e norte são tratadas da mesma maneira como quando ocorre a invalidação da comunicação com o GPS.

Para desabilitar a comunicação de um equipamento com o GPS, basta selecioná-lo na interface gráfica e clicar no botão criado para esse fim.

O pacote para este evento tem a estrutura:

- **STX**: com o valor dois.
- **CMD**: com o valor um.
- **TAM**: com o valor 28.
- **Código do computador de bordo**: dois bytes contendo o código. Dependente da instância do objeto. Cada computador de bordo possui um código cadastrado no banco de dados.
- **Identificador do evento**: quatro bytes representando o identificador do evento para futura confirmação de recebimento pelo Despacho.
- **Código do evento**: um byte com o valor 134.
- **Instante (UTC)**: quatro bytes representando o instante em que ocorreu o evento.
- **Coordenada leste do equipamento**: quatro bytes indicando a coordenada do equipamento em coordenadas locais da mina, não em coordenadas geográficas.
- **Coordenada norte do equipamento**: exatamente como acima, mas representando a coordenada norte.
- **Altitude**: dois bytes representando a altitude, em metros, do equipamento em relação ao nível do mar.
- **Número de matrícula do operador**: quatro bytes representando o número de matrícula do operador que está autenticado no computador de bordo.
- **Tipo do evento**: um byte com o valor dois.

- **Argumentos**: quatro bytes não usados atualmente. O valor zero é enviado. Este campo está no pacote para manter a compatibilidade com o SmartMine.

5.3.15 Habilitar comunicação com o GPS

Ocorre quando a comunicação entre o computador de bordo do equipamento e o hardware de GPS do mesmo é restabelecida após um período sem comunicação.

Quando a comunicação com o GPS é restabelecida, o atributo do computador de bordo que representa a comunicação com o GPS assume o valor verdadeiro. Neste caso, as coordenadas leste e norte são tratadas da mesma maneira como quando ocorre a validação da comunicação com o GPS.

Para restabelecer a comunicação de um equipamento com o GPS, basta selecioná-lo na interface gráfica e clicar no botão criado para esse fim.

O pacote de rede para o evento tem a estrutura:

- **STX**: com o valor dois.
- **CMD**: com o valor um.
- **TAM**: com o valor 28.
- **Código do computador de bordo**: dois bytes contendo o código. Dependente da instância do objeto. Cada computador de bordo possui um código cadastrado no banco de dados.
- **Identificador do evento**: quatro bytes representando o identificador do evento para futura confirmação de recebimento pelo Despacho.
- **Código do evento**: um byte com o valor 134.
- **Instante (UTC)**: quatro bytes representando o instante em que ocorreu o evento.
- **Coordenada leste do equipamento**: quatro bytes indicando a coordenada do equipamento em coordenadas locais da mina, não em coordenadas geográficas.
- **Coordenada norte do equipamento**: exatamente como acima, mas representando a coordenada norte.
- **Altitude**: dois bytes representando a altitude, em metros, do equipamento em relação ao nível do mar.

- **Número de matrícula do operador**: quatro bytes representando o número de matrícula do operador que está autenticado no computador de bordo.
- **Tipo do evento**: um byte com o valor três.
- **Argumentos**: quatro bytes não usados atualmente. O valor zero é enviado. Este campo está no pacote para manter a compatibilidade com o SmartMine.

5.3.16 Desabilitar Comunicação do Equipamento com o *Access Point*

Ocorre quando o computador de bordo do equipamento não consegue estabelecer comunicação com o *Access Point*. Por exemplo, quando o equipamento está fora da área de cobertura da rede *wireless*.

Quando um computador de bordo tem a comunicação com o Access Point desabilitada, seu atributo que representa a comunicação com o Access Point assume o valor falso. A partir desse momento o computador de bordo não enviará mais pacotes de rede ao Despacho, até que sua comunicação seja restabelecida.

Para desabilitar essa comunicação, basta selecionar um equipamento na interface gráfica e clicar no botão criado para tal.

5.3.17 Habilitar Comunicação do Equipamento com o *Access Point*.

Ocorre quando o computador de bordo do equipamento restabelece sua comunicação com o *Access Point*.

Quando um computador de bordo tem a comunicação com o Access Point restabelecida, seu atributo que representa a comunicação com o Access Point assume o valor verdadeiro. A partir desse momento o computador de bordo volta a enviar pacotes normalmente.

Para desabilitar essa comunicação, basta selecionar um equipamento na interface gráfica e clicar no botão criado para tal.

5.3.18 Sobrecarga de Pacotes

Envia uma grande quantidade de pacotes de eventos de troca de estado, visto na subseção 5.3.10 para o Despacho num intervalo de tempo muito curto. É necessário para

avaliar como o Despacho se comporta quando tem que processar uma grande quantidade de dados enfileirados em sua memória.

Cem pacotes de troca de estado do primeiro equipamento de transporte são enviados para o Despacho para testar como ele se comporta com uma alta carga de dados. Os pacotes são enviados consecutivamente em um *loop* e a aplicação, geralmente, envia todos em um intervalo entre um e dois segundos.

5.4 Requisitos Não Implementados

Os requisitos não implementados até a época da escrita deste trabalho são:

- Recebimento de comando do Despacho.
- Recebimento de mensagem de texto.
- Execução de arquivo de programação.

5.4.1 Recebimento de comando do Despacho.

Este requisito diz respeito ao Despacho enviar um comando a um computador de bordo e o simulador tratar o comando como se fosse o próprio computador de bordo. Há um número muito grande comandos possíveis de serem enviados pelo Despacho, por isso ainda é necessário delimitar o escopo dos comandos a serem tratados pelo *SMAccessPoint Simulator*.

5.4.2 Recebimento de mensagem de texto

O Despacho pode enviar tanto mensagens de informação quanto mensagens de interrogação aos operadores dos equipamentos, que são recebidas através do computador de bordo.

No caso de mensagens de informação o operador do equipamento precisa apenas informar que está ciente da mensagem, confirmando-a no computador de bordo.

As mensagens de interrogação sempre devem ser formuladas para terem “sim” ou “não” como resposta. Nesse caso a mensagem é exibida para o operador do equipamento e este a aceita ou rejeita.

O simulador deve receber tais mensagens, mostrá-las ao usuário e permitir que este a confirme, no caso de informação, ou que este a rejeite ou aceite, no caso de interrogação.

5.4.3 Execução do arquivo de programação

O sistema poderá ser programado para funcionar de acordo com um arquivo de programação definido pelo usuário. Nesse arquivo deverão estar listados todos os eventos a serem enviados pelo simulador para cada equipamento desejado.

A linguagem a ser usada para gerar o arquivo de programação ainda terá de ser definida.

A execução dessa programação pode ser única ou cíclica.

Execução única indica que uma vez executados todos os comandos do arquivo, o simulador irá parar de enviar eventos.

Execução cíclica indica que uma vez executados todos os comandos do arquivo, estes serão reenviados continuamente, na ordem definida pelo usuário, até que o mesmo decida parar a programação.

5.5 Dificuldades Encontradas no Desenvolvimento

As principais dificuldades encontradas durante o desenvolvimento do software foram:

- Parte do código legado não orientado a objetos e de difícil entendimento. O código do SmartMine responsável pela comunicação com os dispositivos de hardware não foi todo desenvolvido usando-se a orientação a objetos e tinha uma estrutura, muitas vezes, difícil de ser compreendida. Esse fato contribuiu para que a atividade de engenharia reversa demorasse muito mais do que o desejado.
- Falta de documentação do software existente. Além do código não estar orientado corretamente, como dito acima, não existia nenhuma documentação sobre o mesmo. Portanto, não havia alternativa ao fato de ter de gastar mais tempo analisando o código legado.
- Disponibilidade de comunicação. Muitas vezes surgiam dúvidas de como o hardware, tanto o computador de bordo quanto o *Access Point*, deviam funcionar. Para retirar tais dúvidas era necessário consultar um desenvolvedor do módulo de hardware do SmartMine. Mas como todos estavam dedicados a outros projetos correntes da Devex, era muito difícil

obter tais informações de forma rápida e eficiente. Esse fato contribuiu para que alguns requisitos demorassem mais tempo do que seria necessário se uma pessoa do módulo de hardware estivesse sempre disponível para esclarecimentos.

5.6 Interface do Simulador

Como explicado anteriormente, a interface gráfica do simulador não recebeu nenhuma atenção especial, limitando-se ao mínimo necessário para que o usuário possa fornecer dados ao sistema.

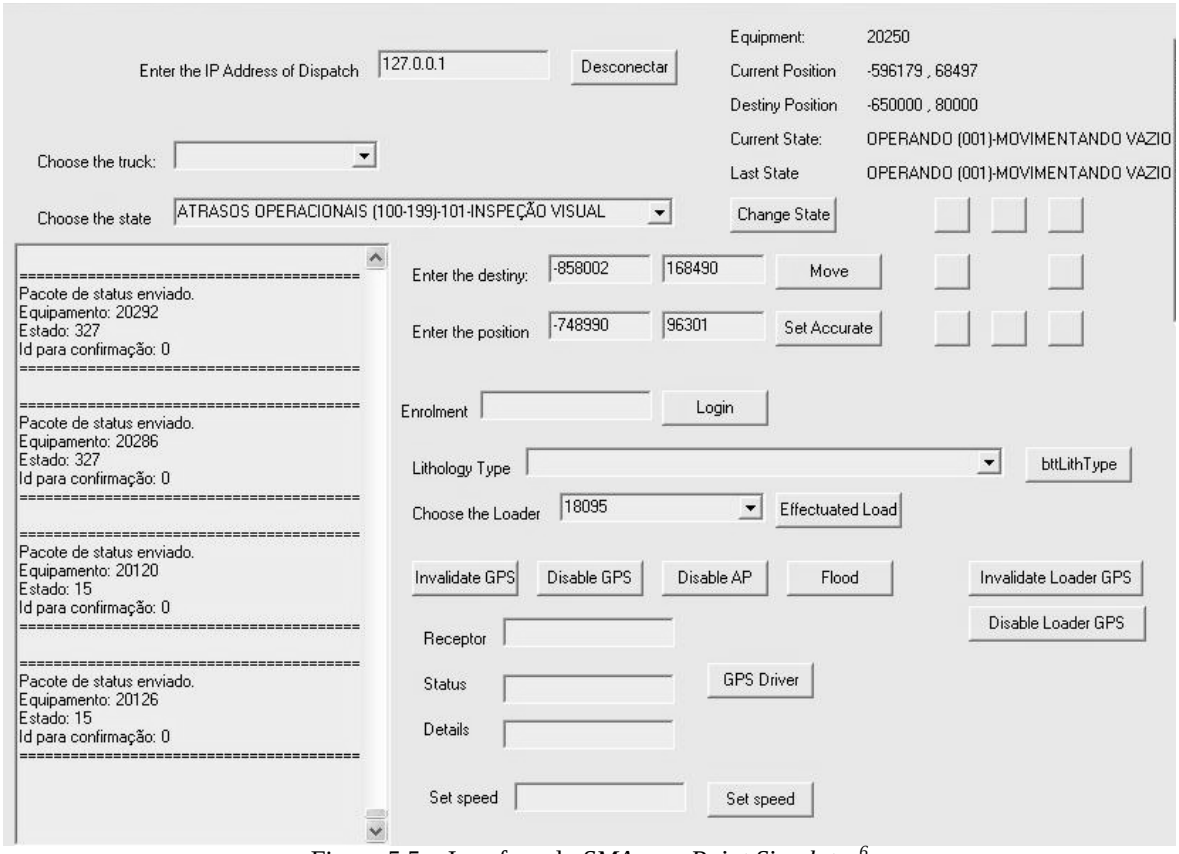

Figura 5.5 – Interface do *SMAccessPoint Simulator*[6]

5.7 Despacho com o Simulador

Na figura 5.6 pode-se notar equipamentos de mina sendo exibidos no módulo de Despacho. Estes equipamentos estão sendo exibidos porque o simulador está gerando

[6] As imagens utilizadas neste trabalho são de propriedade da Devex Tecnologia e Sistemas Ltda, e sua publicação neste trabalho foram autorizadas pela mesma. Não é permitido cópias ou reproduções das mesmas sem prévia autorização.

dados relativos à posição e deslocamento dos mesmos. Esses dados são enviados ao Despacho através de pacotes de rede e este consegue exibi-los para o usuário.

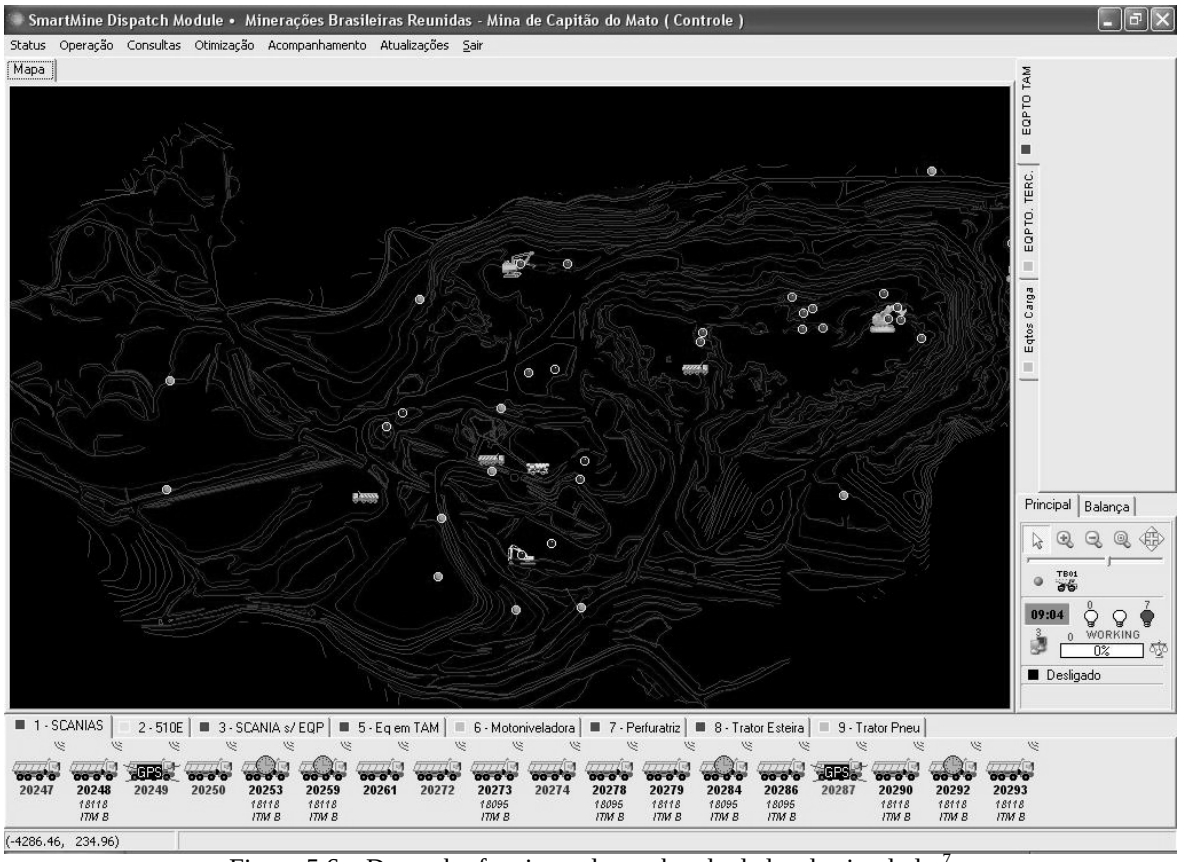

Figura 5.6 – Despacho funcionando recebendo dados do simulador[7]

[7] As imagens utilizadas neste trabalho são de propriedade da Devex Tecnologia e Sistemas Ltda, e sua publicação neste trabalho foram autorizadas pela mesma. Não é permitido cópias ou reproduções das mesmas sem prévia autorização.

6 RESULTADOS

Após a implementação dos requisitos listados anteriormente o sistema pôde ser usado satisfatoriamente para simular, inicialmente, a movimentação dos equipamentos na mina. O usuário pode escolher qualquer equipamento individualmente e definir um destino de movimentação para o mesmo. O software simula o deslocamento de cada equipamento e, a cada cinco segundos, envia um pacote de status por equipamento para o Despacho. A partir deste pacote de status, a imagem do equipamento é atualizada no mapa da mina.

A simulação de envio de eventos ao Despacho também foi utilizada com sucesso. O Despacho recebe os pacotes de rede enviados pelo simulador e os trata da mesma maneira como se tivessem sido enviados por um computador de bordo de um equipamento real.

A possibilidade de alterar a velocidade e posição de qualquer equipamento simulado foi de vital importância para testar algoritmos de controle do Despacho, como tratamento de caminhões e escavadeiras que estão em áreas de escavação, controle de velocidade dos equipamentos, entre outros.

O sistema também foi utilizado com sucesso para testar a integração dos computadores de bordo com o GPS de alta precisão, utilizados em uma mina de diamantes na África. Para o teste dessa integração foi necessário implementar o envio de um pacote de rede que não estava inicialmente especificado. Mas como o sistema estava dividido em camadas, bem modularizado e orientado a objetos, a implementação foi fácil e rápida.

Como dito anteriormente, a construção do software usando camadas bem definidas e orientado a objetos permite que funcionalidades extras e novos pacotes de rede sejam adicionados com grande facilidade ao sistema.

Com apenas os requisitos implementados, o sistema pôde ser usado para simular uma mina inteira em funcionamento enviando dados ao Despacho. Esses dados foram capturados e usados em uma campanha de marketing da empresa.

O sistema foi utilizado com sucesso também para simular o funcionamento de uma mina e mostrar o comportamento do Despacho em uma feira internacional sobre mineração ocorrida em Belo Horizonte em 2005, a Exposibram.

O *SMAccessPoint Simulator* pode ser usado para testar uma funcionalidade do sistema a qualquer momento do desenvolvimento, não necessitando nem mesmo de uma rede de computadores montada. O usuário pode rodar o software de Despacho e o simulador na mesma máquina, especificando o endereço IP 127.0.0.1. Com este endereço,

ou qualquer outro que aponte para a própria máquina, os pacotes de rede são montados e enviados corretamente ao Despacho usando-se apenas um computador para os dois softwares.

O simulador também pode ser usado em rede, enviando dados para o software de Despacho que esteja ativo em outro computador. Para isto basta especificar qual o endereço IP da máquina que esteja rodando o Despacho.

7 CONCLUSÕES

A maioria dos processos de software incorpora, de alguma forma, atividades de teste em suas definições para determinar se uma tarefa foi realizada corretamente. Atualmente a atividade de teste pode ser bastante onerosa no desenvolvimento de software porque grande parte desta ainda é executada de forma manual.

A presença de defeitos em programas pode causar não somente gastos excessivos no desenvolvimento, mas também acidentes, envolvendo até mesmo vidas humanas, dependendo do dispositivo que o use. Portanto o objetivo principal da atividade de teste é encontrar defeitos em programas.

Um defeito em um programa pode ser caracterizado como erro, falta ou falha.

Um software pode ser testado com ou sem sua execução. Teste sem execução envolve revisão de conceitos sobre o mesmo. Testes com execução visam analisar o comportamento do programa quando em funcionamento. Um software pode ser testado em nível de unidade, de integração e de validação. Para se testar um software podem ser usados métodos de caixa branca (testa-se o funcionamento interno. Usados em nível de unidade e integração) ou caixa preta (testa-se a interface do sistema. Usados em nível de integração e validação).

Como a atividade de teste pode representar uma atividade onerosa para a empresa, cada vez mais se busca usar ferramentas de teste automatizadas para reduzir este custo.

Apesar de já existirem simuladores de mineração no mercado, nenhum deles realizava o que era necessário para testar o módulo de Despacho: simular um tracker e enviar dados usando o protocolo de comunicação desenvolvido pela empresa. Essa foi a novidade inserida por esse trabalho, desenvolver um simulador de ambiente capaz de comunicar com um produto usando-se um protocolo já existente.

O *SMAccessPoint Simulator* mostrou-se como uma grande ferramenta de testes, pois agilizou o processo e garantiu maior autonomia para a equipe de desenvolvimento que não possuía plataforma de testes com hardware. O processo foi agilizado porque não é mais necessário, sempre que se precisa testar uma nova funcionalidade, configurar um computador de bordo e fazê-lo enviar dados ao Despacho. Agora é necessário apenas ligar o simulador e vários “equipamento virtuais” estarão à completa disposição para enviar quais e quantos dados forem necessários. A equipe de desenvolvimento conseguiu maior autonomia porque não depende mais da estrutura de teste com hardware ou disponibilidade

de pessoal para realizar os testes. O próprio desenvolvedor pode usar o simulador para enviar os dados de que precisa para seus testes.

Um Sistema de Tempo Real (STR, ou RTS "*Real Time System*") é um sistema computacional que deve reagir a estímulos oriundos do seu ambiente em prazos específicos. Portanto, o comportamento correto de um sistema de tempo real não depende somente da integridade dos resultados obtidos, mas também dos valores de tempo em que são produzidos. Um resultado que não seja produzido no prazo limite especificado pode até representar uma ameaça ao sistema.

Os sistemas de tempo real podem ser classificados, do ponto de vista da segurança, em sistemas não críticos de tempo real (uma falha não representa uma ameaça real) ou sistemas críticos de tempo real (uma falha pode até causar uma catástrofe). Os sistemas críticos de tempo real ainda podem ser subdivididos em sistemas seguros em caso de falha e sistemas ainda operacionais em caso de falha.

O SmartMine foi desenvolvido para funcionar 24 horas por dia durante sete dias por semana processando uma enorme quantidade de dados recebidos de computadores de bordo dos equipamentos de mina. A necessidade de processamento dos dados recebidos e envio de respostas em um tempo muito curto aos equipamentos evidencia uma das características de tempo real do SmartMine. Foi mostrado que o SmartMine se encontra na categoria de Sistemas Críticos de Tempo Real Operacionais em Caso de Falha, e por isso é importante um sistema de teste que simule o ambiente com o qual o SmartMine interage, para simular o envio de dados e verificar como o sistema se comporta.

O sistema comunica-se com os equipamentos de hardware da mina através de rede *wireless* e rede Ehternet. Para que os dados possam ser transportados de uma rede à outra foi necessário o desenvolvimento de um dispositivo chamado *Access Point*, que funciona como um *gateway* entre as redes, apenas transportando pacotes de uma para a outra, sem nenhum processamento sobre os mesmos.

Dadas as características do sistema SmartMine, pode-se notar que esse é um software bastante complexo de ser testando durante o desenvolvimento, pois nesta fase os dados provenientes dos equipamentos da mina não estão disponíveis para servirem de entrada para o sistema. Para possibilitar que o sistema fosse testado mais efetivamente foi necessário desenvolver um simulador de ambiente da mina para que este pudesse enviar os dados ao SmartMine. Com esse simulador, o sistema pode ser testado como se estivesse recebendo dados dos computadores de bordo dos equipamentos de mina do mundo real.

Para o desenvolvimento do simulador, após o levantamento dos requisitos e concepção do software, foi necessário estudar o código fonte do SmartMine referente à comunicação do módulo de controle com os dispositivos de hardware. Uma atividade de engenharia reversa foi realizada para definir o formato dos pacotes de rede correspondentes a cada evento especificado nos requisitos. O código legado do SmartMine não orientado a objetos e com uma estrutura difícil de ser entendida dificultou o processo, consumindo mais tempo do que o desejado. O código legado apresenta essa característica porque, muitas vezes, o software foi codificado apressadamente, devido à pressão exercida pelo cliente para a entrega do mesmo. Desenvolver o código desta maneira possibilitou a entrega do sistema no prazo, mas dificultou as futuras atividades de manutenção. Portanto antes de codificar o programa, deve-se gastar um tempo planejando como fazê-lo de modo a facilitar sua manutenção.

O desenvolvimento de software orientado a objetos e estruturado em camadas distintas de interface, processamento e acesso a dados possibilitará uma maior facilidade de manutenção do mesmo. Como o simulador foi desenvolvido dessa maneira, é bastante fácil adicionar funcionalidades e corrigir erros.

O simulador gera dados relativos à movimentação dos equipamentos, envio de eventos ao Despacho e integração dos computadores de bordo tanto com os *Access Point* quanto com o sistema de GPS.

O simulador desenvolvido atingiu, com elevado grau de satisfação, o objetivo proposto, funcionando para simular o ambiente de mineração e enviar dados ao Despacho. Os dados recebidos eram tratados da mesma maneira como seriam os dados recebidos dos computadores de bordo dos equipamentos, não havendo distinção entre os dados simulados e os dados reais.

8 TRABALHOS FUTUROS

Alguns dos trabalhos futuros que podem ser desenvolvidos a partir deste são:

- Conectar ao banco de dados SQLServer.
- Tratar as variações de velocidade do equipamento, altitude do equipamento e potência do rádio de maneira mais contínua. Atualmente a variação está muito aleatória.
- Implementar os requisitos restantes.
- Melhorar a interface gráfica.
- Desenvolver uma funcionalidade no simulador para realizar o *playback* da mina. Essa funcionalidade seria usada para repetir um período de operação da mineração. Como todos os eventos são armazenados no banco de dados, o simulador poderia consultar todos esses dados e enviá-los novamente ao Despacho. E esse envio de dados poderia ser realizado na velocidade que o usuário desejar. Por exemplo, dados de produção armazenados em dois dias de trabalhos poderiam ser enviados ao Despacho em um intervalo de 15 minutos.
- Implementar outros eventos dos equipamentos não especificados nesta primeira versão.
- Adicionar a simulação de dispositivos de hardware usados na mina, como balanças, painéis, pontos de basculamento e outros. Todos esses dispositivos também podem interagir com o Despacho da mesma maneira como os *trackers* dos caminhões o fazem.

9 **REFERENCIAL BIBLIOGRÁFICO**

CHINA AIRLINES. AIRCRAFT ACCIDENT INVESTGATION REPORT 96-5. Aircraft Accident Investigation Commission. Ministry of Transport. Japan. 1999.

DEUTSCH, M., “Verification and Validation”, in Software Engineering, (R. Jensen e C. Tonies, eds.) Prentice-Hall, 1979, pp. 329-408.

DUNN, R., Software Defect Removal, McGraw-Hill, 1984.

FARINES, J., FRAGA J. S., OLIVEIRA, R. S. Sistemas de tempo real, 2002.

FENTON, N. E.; PFLEEGER, S. L., Software Metrics, 1996 Cambridge United Press, United Kingdom

IEEE Standard for Software Test Documentation, 1983 Std 829-1983

MILLER, E., Automated Tools for Software Engineering, IEEE Computer Society Press, 1979, p. 169.

NASA. Mars Climate Orbiter Official Website.

PASSOS, R. M., Modelagem e implementação de protocolos para comunicação com e sem fio de um sistema integrado Hardware/Software em tempo real, UFLA, 2002

PRESSMAN, R. S., Engenharia de Software, Makron Books 1995.

SCHACH, S. R. Testing: Principles and Practice. ACM Computer Surveys, Vol 28, No 1, March 1996.

SOMMERVILLE, I., Engenharia de Software. Addison Wesley. 6ª edição. 2003

NETO, A. N. R., Template do programa Arena para simulação das operações de carregamento e transporte em minas a céu aberto. 2004.