UNIVERSIDADE FEDERAL DO AMAZONAS

FACULDADE DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA

DE PRODUÇÃO

AVALIAÇÃO DOS TEMPOS DE TESTE DE PRODUTOS ELETRÔNICOS

LEANDRO ALVES FERRAZ

MANAUS

2012

UNIVERSIDADE FEDERAL DO AMAZONAS

FACULDADE DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

LEANDRO ALVES FERRAZ

AVALIAÇÃO DOS TEMPOS DE TESTE DE PRODUTOS ELETRÔNICOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal do Amazonas, como requisito parcial para a obtenção do título de Mestre em Engenharia de Produção, área de concentração Gerência da Produção.

Orientadora: Profª Drª Silvana Dacol *(in memorian)*

MANAUS

2012

LEANDRO ALVES FERRAZ

AVALIAÇÃO DOS TEMPOS DE TESTE DE PRODUTOS ELETRÔNICOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal do Amazonas, como requisito parcial para a obtenção do título de Mestre em Engenharia de Produção, área de concentração Gerência da Produção.

Aprovado em 08 de Janeiro de 2010.

BANCA EXAMINADORA

Prof[a] Dr[a] Silvana Dacol, Presidente
Universidade Federal do Amazonas

Prof Dr Idélcio Cardoso, Membro
Instituto Nokia de Tecnologia

Prof[a] Dr[a] Fabiana Lucena, Membro
Nokia do Brasil

À Margaux, por estar comigo em todos os momentos dessa etapa da minha vida.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus por me dar suporte para tudo o que enfrentei;

Aos meus pais que mesmo não estando fisicamente próximos, sempre estiveram presentes em minha vida;

À minha orientadora Silvana pela orientação e cobrança na medida certa;

Ao professor Antônio Marcos, pela orientação e suporte no início do trabalho;

À Ocileide Silva por acreditar que seria possível;

Ao INdT por me dar tempo e recursos para que eu pudesse realizar o curso;

À Nokia do Brasil, pela utilização de suas instalações.

A resposta certa, não importa nada: o essencial é que as perguntas estejam certas.

Mário Quintana

.

Resumo

A introdução do conceito de linha de montagem por Henry Ford, trouxe um grande avanço à eficiência do uso dos recursos e transformações durante o processo fabril. Mais tarde com Ohno (1988), mostrou-se que durante o processo fabril, todo o desperdício deve ser removido do processo. Ohno ainda explica que desperdício é toda atividade que não agrega valor ao produto final. Como o teste de produção não agrega valor ao produto final, apenas garante a confiabilidade do mesmo, o presente trabalho tem como objetivo desenvolver uma metodologia para identificar e reduzir os tempos de testes mais demorados de uma sequência de testes de produtos eletrônicos sem que seja diminuída a qualidade dos produtos. Baseado na análise da utilização dos instrumentos dos testes é possível a identificação dos pontos candidatos a serem executados concorrentemente (paralelismo). A execução de dois ou mais testes em paralelo nas sequências torna possível a redução do tempo total de testes do produto. Um histórico dos tempos e resultados foi catalogado para através da utilização de índices estatísticos como a média e o desvio padrão dos resultados, juntamente com o índice Cpk dos limites dos testes, poder comprovar não só a eficiência, mas também a eficácia da metodologia. Além da metodologia, o trabalho também cita as ferramentas de análise e automação de redução de tempos de testes que foram desenvolvidas ao longo do projeto. A metodologia se mostrou eficaz e não apenas os tempos de testes foram reduzidos, mas, também, os resultados se mostraram igualmente confiáveis.

Palavras-chave

Redução de Tempo de Testes, Paralelismo, Utilização de Instrumentos

Abstract

The production line concept introduction by Henry Ford, brought a great enhancement in the resource usage efficiency and transformation during the industrial process. Afterwards Ohno (1988) presented that during the industrial process all the waste should be removed from the process. Ohno also explains that waste is any activity that do not aggregate value in the final product. As a production test do not aggregate value, it simply ensure the product reliability, this paper has the purpose to develop a methodology to evaluate, identify and reduce the longest tests' time of an electronic test sequence without decrease the product quality. Based on the instrument driver's utilization is possible to identify opportunities points to be executed concurrently (parallelism). The execution of two or more tests in parallel make possible the time reduction of the whole product test. One baseline of times and results were created to act as a benchmark and based on statistical process control indexes such as result mean and standard deviation and also the test acceptance limits' cpk index, it will be possible to confirm the methodology efficacy and efficiency. Besides the methodology, the paper also refers the analysis and automation tools that were created during the time optimization project.

Key-words

Test Time Reduction, Parallelism, Instrument drivers utilization

Sumário

Lista de figuras

Lista de Quadros

Lista de Tabelas

ABREVIATURAS

Sigla	Significado
ATS	Sistema Automático de Teste
CEP	Controle Estatístico de Processos
CPU	Unidade Central de Processamento
DLL	Biblioteca de Ligação Dinâmica
DUT/UUT	Dispositivo/Unidade em Teste
GSM	Sistema Global de Comunicação Móvel
JIT	*Just in Time*
LES	Limite de Especificação Superior
LEI	Limite de Especificação Inferior
PM	*Process Model*
TP	Plano de Teste
TPO	*Test Plan Optmizer*
TPS[1]	Sistema de Produção da Toyota
TPS[2]	Plano de Teste do Produto
XML	Linguagem Estendida de Marcação

1 INTRODUÇÃO

Em 1913, Henry Ford introduziu uma das maiores inovações da era da máquina, a linha de montagem em movimento para a fabricação de automóveis Ford (DAVIS et al 1999).

Uma linha de produção é um processo de manufatura no qual partes são adicionadas em um produto de uma forma sequencial usando um planejamento logístico para gerar um produto final de forma muito mais eficiente que métodos de manufatura tradicionais.

Segundo Shingo(1996), um processo pode ser entendido como um fluxo de materiais no tempo e no espaço, ou seja, é a transformação da matéria-prima em um produto finalizado. Por sua vez, as operações podem ser vistas como o trabalho realizado para efetivar essa transformação.

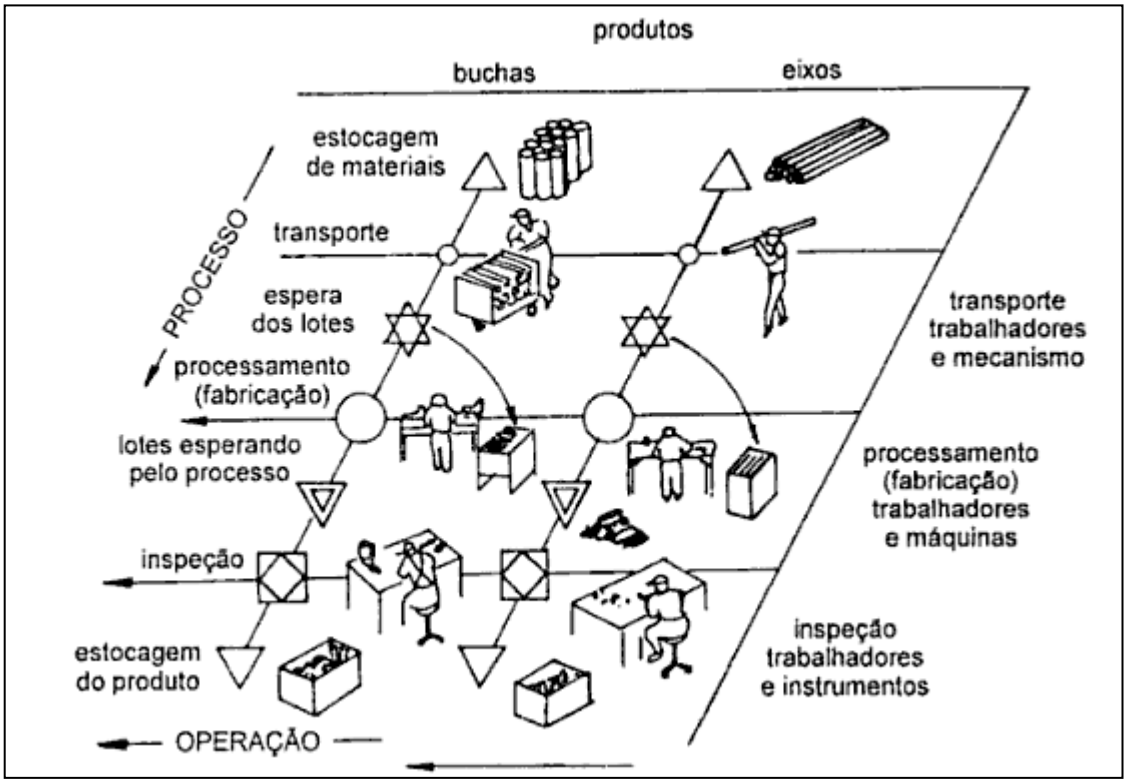


Figura 1: A estrutura de uma linha de produção.
Fonte: Shingeo(1996)

Cinco elementos distintos podem ser identificados no fluxo de produção da Figura 1. São eles: processamento, inspeção, transporte, espera do processo e espera do Lote.

Santos e Urbina (2002) mostram que os efeitos ocasionados pela inovação tecnológica são observados na otimização dos processos de produção, que modificam a organização do trabalho, e reduzem os custos de produção, buscando maior qualidade e velocidade de entrega do produto.

De modo a garantir “defeito zero”, as empresas tem adotado um sistema de inspeção chamado de inspeção sucessiva, no qual o produto é inspecionado após cada operação realizada (SHINGO, 1996).

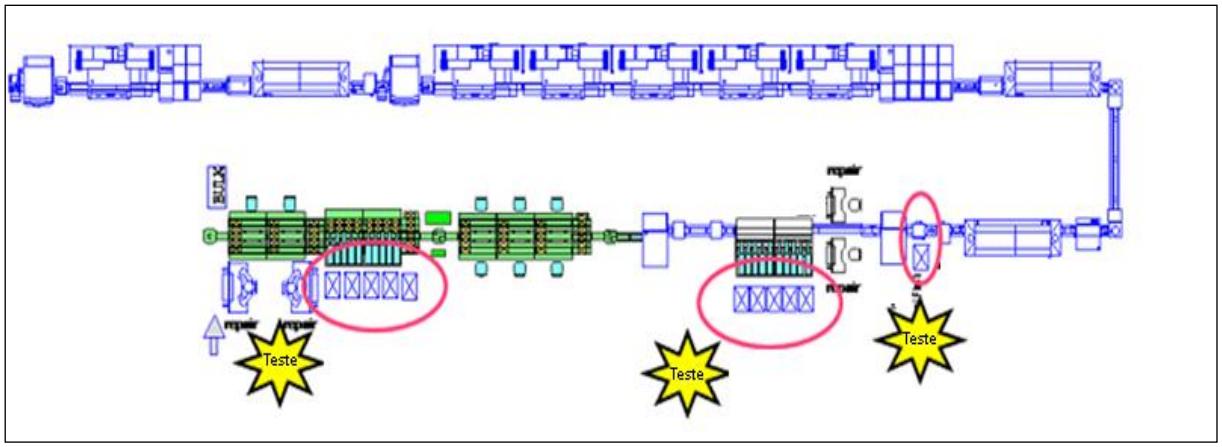


Figura 2: Inspeção sucessiva
Fonte: Autor

Ramalho e Medeiros (2003) afirmam que os sistemas de teste de visão de máquina são auto calibráveis e imunes a aspectos intrínsecos à natureza humana (estresse, emoções, fadiga e julgamentos subjetivos) e que tais sistemas contribuem para a melhoria contínua da produção e o aumento da produtividade dada a sua adaptabilidade e capacidade de redução de erros decorrentes da monotonia e da subjetividade em tarefas executadas por humanos.

A empresa estudada possui fases de testes que são realizados sucessivamente às operações realizadas. Os equipamentos de teste da empresa estudados, não compartilham equipamentos entre si, ou seja, cada testador tem todos os seus equipamentos de teste à disposição por todo o tempo.

O presente trabalho tem como propósito avaliar o sistema de teste automático de uma linha de produção de produtos eletrônicos, avaliando os testes e gerando uma redução de tempo.

1.1 JUSTIFICATIVA

A redução de tempo de testes pode gerar redução da quantidade de equipamentos de que não agregam valor de uma linha de produção, reduzindo o custo do produto final e possibilitando adicionar mais linhas de produção, proporcionando um aumento da produção, no espaço que antes era dedicado aos testadores.

Ferraz e Dacol (2009) mostraram que se a fase de teste for o gargalo, o benefício pode ser um aumento de produção devido a redução dos tempos de teste. Em seu estudo eles mostraram também que mesmo se a fase de teste não for gargalo, outros benefícios podem ser obtidos, tais como redução de testadores, operadores, manutenção e espaço físico ocupado na linha de produção

Um teste de um produto deve ser eficaz para não permitir que um produto defeituoso seja lançado no mercado, mas também deve ser desenvolvido rapidamente para que a sua criação não atrase o lançamento do produto. Por isso, os desenvolvedores de teste nem sempre gastam esforços para otimizar os testes que projetam. Manter uma equipe para otimização dos testes lançados é uma

solução que certos fabricantes têm usado, entretanto essa solução representa um custo adicional ao projeto.

No caso da empresa estudada, os testes eram desenvolvidos para serem eficazes, porém pela urgência de se lançar um produto no mercado e falta de desenvolvedores de testes para analisar a eficiência dos testes, o tempo dos testes não era um fator fundamental para impedir ou não o seu uso.

No processo produtivo foi identificado um grande dispêndio de tempo, custos e espaço devido à fase de testes e segundo Robles (2003) as empresas, desde a década de 90, já entraram na era da qualidade e o desperdício deve ser evitado ao máximo.

Como a etapa de testes não agrega valor ao produto final, o teste é uma das fases de produção que mais se deseja reduzir. Por outro lado, a qualidade dos produtos é um importante fator na escolha da marca do produto pelo cliente. Sendo assim, o teste tem que garantir uma alta confiabilidade dos produtos da empresa, além de não poder onerar muito o produto, pois de acordo com Marinho *et al* (2003) de nada adiantaria fabricar um produto de ótima qualidade se não houvesse compradores dispostos a pagar um alto preço por ele.

1.2 PROBLEMA DE PESQUISA

A empresa estudada já faz uso de um sistema automático de testes para testar seus produtos, entretanto seus testes estavam tomando muito tempo do processo produtivo.

Tendo em vista a necessidade da empresa, o problema de pesquisa que orienta esse trabalho então é: É possível desenvolver uma metodologia que consiga

reduzir os tempos de testes de produtos eletrônicos (aumentar sua eficiência), porém sem reduzir a eficácia dos mesmos?.

1.3 OBJETIVO

O desafio proposto foi que fosse demonstrado um método que consiga reduzir os tempos de um produto com alto volume de produção em pelo menos 15% nas fases de calibração e testes funcionais.

1.3.1 Objetivo Geral

O objetivo do trabalho é avaliar o tempo dos testes dos produtos eletrônicos de uma linha de produção e permitir a redução dos mesmos sem que a qualidade dos mesmos seja comprometida.

1.3.2 Objetivos Específicos

Os objetivos específicos do presente trabalho são listados a seguir na ordem em que foram obtidos:

1. Demonstrar os tempos e resultados dos testes do produto selecionado para efeito de histórico;
2. Identificar os possíveis pontos de redução de tempo nas sequências de teste;
3. Propor uma metodologia para reduzir os tempos de teste baseado no uso de paralelismo e testes concorrentes;
4. Aplicar a metodologia proposta em duas fases de teste (calibração e funcionais) do produto selecionado;
5. Comparar os tempos dos testes originais com os testes modificados para calcular o ganho;

6. Validar a redução de tempo com base nos resultados dos testes originais e modificados.

1.4 ESTRUTURA DA DISSERTAÇÃO

Nesse tópico é apresentada a estrutura deste trabalho, descrevendo sucintamente o conteúdo de cada capítulo.

No primeiro capítulo é apresentada uma introdução, explicando a justificativa e a motivação que originou o trabalho. Logo em seguida é apresentado o objetivo geral do trabalho e os objetivos específicos, realizados para que fosse possível alcançar o objetivo geral.

O segundo capítulo apresenta ao leitor conceitos fundamentais para que o conteúdo do trabalho possa ser entendido. A fundamentação teórica é composta por tópicos de gestão da produção, paralelismo computacional, sistemas automáticos de testes e controle estatístico de processos.

No capítulo de gestão de produção são discutidos conceitos como estratégias de manufatura e *lead time* e esclarecidos como os termos eficiência e eficácia foram empregados na dissertação. Em paralelismo computacional, o objetivo é apresentar uma introdução aos processos computacionais, sua estrutura e como realizar paralelismo entre eles.

Para explicar o que é um sistema automático de teste, mote do trabalho, foi criado um subcapítulo, contextualizando o leitor e explicando os principais problemas enfrentados.

No tópico de controle estatístico de processos, o leitor aprenderá sobre os conceitos de média, capacidade de um processo e o princípio de Pareto, fundamentais para a análise e validação do trabalho.

O terceiro capítulo contém a contribuição do autor para a comunidade científica. Toda a metodologia está contida nele.

O capítulo começa descrevendo a metodologia proposta para a redução de tempos de testes, explicando o procedimento que foi feito para investigação dos pontos de otimização, técnicas de paralelismo e como seria medido os ganhos do trabalho.

Os experimentos realizados também são descritos detalhadamente neste capítulo, indicando onde foram criados os testes concorrentes juntamente com uma breve análise dos mesmos.

As análises de tempo e resultados dos testes são realizadas após os experimentos, para explicar que o experimento foi realizado com êxito.

Ao final do terceiro capítulo são apresentadas duas ferramentas desenvolvidas pelo autor ao longo do trabalho. Essas ferramentas foram de grande ajuda para análise e automação da redução dos tempos de teste.

Finalmente, no último capítulo, o autor expõe suas conclusões, justificando as contribuições do trabalho e propondo futuros trabalhos para estudos.

1.5 LIMITAÇÕES DO TRABALHO

A metodologia descrita se aplica nos casos em que cada Sistema Automático de Teste (ATS) possui exclusivamente os próprios instrumentos e equipamentos de teste, isto é, não há o compartilhamento de equipamentos de testes entre os diferentes ATS.

Um dos resultados da avaliação dos tempos de testes é um plano de testes em que as atividades são executadas concorrentemente, sendo esse plano de teste estático, ou seja, todas as atividades tem uma ordem predefinida e o plano é imutável.

Com o compartilhamento de recursos entre os testadores, essa ordem das atividades deixaria de ser estática, pois a lista de atividades não seria um plano de testes, mas um conjunto de planos de testes.

Para que a metodologia funcionasse com ATS compartilhando recursos entre si, a introdução dos dispositivos nos ATS precisaria ser síncrona, desse modo o conjunto dos planos de testes poderia ser considerado como uma sequência estática de atividades. No caso da empresa estudada, a alimentação dos ATS é feita por operadores humanos e pode ser considerada que é feita de forma aleatória, o que inviabiliza a possibilidade uma introdução síncrona dos dispositivos.

Na seção de trabalhos futuros, existe uma proposta de um escalonador de atividades de teste. Como o escalonador é executado em tempo real, a limitação de compartilhamento de recursos não se aplicaria a ele, porque a cada execução o escalonador poderia produzir um plano de teste diferente.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 GESTÃO DA PRODUÇÃO

A gestão de operações trata de atividades de gerenciamentos estratégicos de recursos escassos, de sua interação e dos processos que produzem e entregam bens ou serviços visando atender as necessidades de desejo e qualidade, tempo e custo de seus clientes. Além disso, deve correlacionar este objetivo com a eficiência do uso dos recursos.

Como nem todos os critérios de desempenho são igualmente valorizados por todos os clientes, é importante que se faça uma análise e priorização desses critérios para tomadas de decisão.

Corrêa e Corrêa (2004) destacam os seguintes critérios de desempenho, para uma empresa de manufatura de produtos eletrônicos e algumas métricas, conforme apresentado no Quadro 1.

Critérios	Descrição	Exemplo de métricas
Preço/Custo	Custo de se produzir o produto	Custo de manufatura (operação)
Velocidade	Tempo para entregar o produto	Lead times internos
Confiabilidade	Cumprimento das promessas feitas	Percentual de entregas no prazo
Qualidade	Se o produto está de acordo com as especificações e sem falhas	Número de falhas no campo
Flexibilidade	Habilidade de modificar o *mix* de produtos de forma econômica	Quanto o desempenho de entregas não é afetado pelo mix de produtos

Quadro 1: Critérios de Desempenho
Fonte: Adaptado de Corrêa e Corrêa (2004)

É importante perceber que métricas que foram utilizadas com sucesso em um contexto podem não ser adequadas em outro. Portanto, os indicadores devem estar alinhados com a intenção estratégica da organização.

O'Mara *et al*(1998), diz que um sistema de medição de desempenho não fornece apenas meios e dados necessários para a gerencia e controlar as várias atividades da empresa, mas também influenciam as decisões e o comportamento organizacional.

Já Neely *et al* (1995), complementa dizendo que uma medição de desempenho pode ser compreendida como a técnica usada para quantificar a eficiência e a eficácia das atividades de negócio. Os termos eficiência e eficácia devem, então, ser usados com precisão nesse contexto.

De acordo com Atkinson (1998), eficácia é a característica que se refere à habilidade de um processo em alcançar seus objetivos enquanto que eficiência é um processo característico que se refere à habilidade de usar um mínimo de recursos para fazer alguma coisa.

O mesmo Atkinson (1998) considera a mensuração de desempenho é, talvez a mais importante, a mais incompreendida e a tarefa mais difícil da atividade gerencial e que uma medição de desempenho deva fornecer feedback para ajudar os membros da empresa a identificarem problemas e oportunidades para melhorias.

De acordo com Corrêa e Corrêa (2004), a única solução para uma empresa sobreviver as mudanças contínuas e os concorrentes cada vez mais competitivos é a melhoria. As melhorias podem ser contínuas (usando uma abordagem evolutiva incremental ou radicais (quando envolve uma mudança de paradigma ou alguma evolução disruptiva.

Uma técnica bastante utilizada para identificar pontos de melhoria é o *Benchmarking* e segundo Bogan e English (1997), é um método sistemático de procurar os melhores processos, as ideias inovadoras e os procedimentos de operação mais eficazes que conduzam a um desempenho superior.

A instituição de programas de melhoramento sem uma orientação estratégica, entretanto, não é uma boa prática gestional, pois gerir é administrar recursos escassos e, portanto os esforços devem ser direcionados a privilegiar os objetivos priorizados pela estratégia da operação.

Ohno (1988) ,criador do Sistema de Produção Toyota (do inglês TPS) explica em seu livro que tudo que deve ser feito é analisar a linha de tempo do momento em que o cliente cria uma ordem de produção até o momento que é coletado o dinheiro e que a redução desse tempo é obtida com a redução dos desperdícios que não agregam valor.

Um dos pilares do TPS é o JIT (do inglês *Just in Time*) que possui como objetivos operacionais fundamentais a qualidade e a flexibilidade. Isso é possível colocando duas metas de gestão acima de qualquer outra: melhoria contínua e eliminação de desperdícios.

Para o JIT a redução dos tempos envolvidos no processo de produção têm dois efeitos fundamentais: melhoria da flexibilidade e confiabilidade e ambas caminham de mãos dadas (WILSON, 2009). Essas melhorias são conseguidas pelo fato da produção não estar mais comprometida a um programa de produção por um prazo longo, adaptando-se mais facilmente às variações de curto prazo da demanda.

Reduzir o *lead time* também permite que se faça redução de inventário no processo. Isso reduz a exposição a fatores ambientais e possíveis danos ao produto, aumentando assim a qualidade do produto final.

De acordo com Wilson (2009), uma companhia com *lead times* curtos quase sem exceção também terá outros benefícios como:

a. Boa gerência de inventário;
b. Alta qualidade;

c. Boa performance de entrega de produtos;
d. Boa disponibilidade de máquina;
e. Boa soluções de problemas
f. Baixos níveis de variação;
g. Um processo estável.

2.2 PARALELISMO COMPUTACIONAL

Computadores modernos consistem hoje em um ou mais processadores, memórias, discos, interface de rede e uma outra gama de periféricos. Brinch (2001) explica que o sistema operacional é o responsável por prover uma ordeira e controlada alocação de processadores, memórias e dispositivos de entrada e saída entre os vários programas que estão competindo por eles. Em suma, a tarefa do sistema operacional é monitorar quem está usando cada recurso, gerenciar os pedidos dos recursos e mediar conflitos de pedidos entre diferentes programas e usuários.

De acordo com Tanenbaum (2006), para realizar um gerenciamento de recursos é necessário que haja multiplexação. Quando um recurso é multiplexado no tempo, diferentes programas ou usuários podem usá-los em turnos diferentes (*time sharing*). Por exemplo, em um computador com apenas um processador, quando diferentes programas precisam ser executados, o sistema operacional aloca o processador primeiro para um programa e então após um certo período de tempo (fatia de tempo ou *time slice*) o sistema operacional aloca o CPU para o outro programa e assim por diante.

Todo *software* que é executado no computador é organizado como um processo sequencial. Um processo é um programa que está sendo executado,

juntamente com os seus estados, registradores e varáveis. Conceitualmente, cada processo tem seu próprio processador virtual, mas de fato o que ocorre é que o processador fica trocando de uma forma muito rápida os processos que são executados.

Quando o processador troca de um processo para o outro, o sistema operacional deverá salvar o estado do processo atual (nos registradores e variáveis do mesmo processo), antes de carregar o processo novo. Essa troca de processos que o processador realiza é chamada de troca de contexto (SILBERSCHATZ *et al*, 1999)

A capacidade do sistema operacional executar múltiplos programas “ao mesmo tempo” é chamada de múlti programação. A **Error! Reference source not found.**(a) mostra um computador fazendo múlti programação com 4 programas na memória. Na Figura 1(b), é mostrado os 4 processos, cada qual com seu próprio fluxo de execução executando de forma independente. Já a Figura 1(c) mostra o que foi citado anteriormente ao longo de um intervalo de tempo. Todos os processos fizeram progressos, porém apenas 1 processo foi executado em um dado instante de tempo.

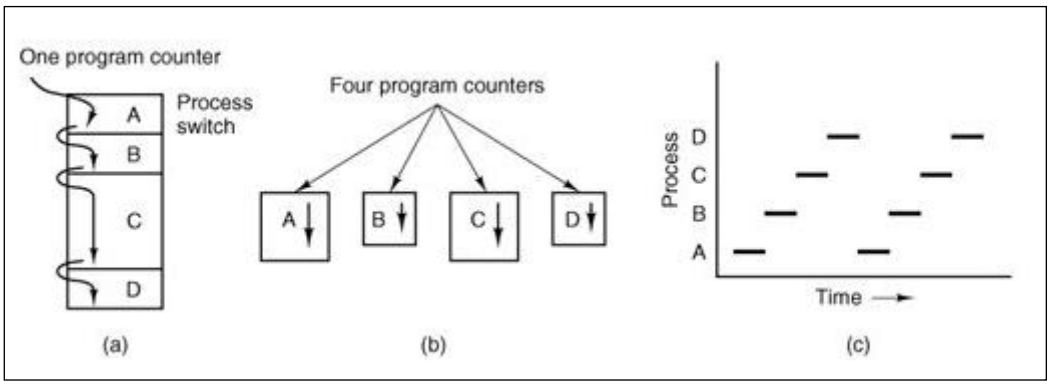


Figura 3:Multi-programação
Fonte: Silberschatz(1999)

Embora cada processo seja uma entidade independente, eles necessitam de se interagir e sincronizar com outros processos. Um processo pode produzir alguma saída que um outro processo requeira como entrada. Nesse caso, dependendo da velocidade relativa entre os processos um processo deverá ser bloqueado até a finalização do primeiro.

Stallings (2004) afirma que os estados dos processos podem ser classificados em três grandes grupos:

a. Prontos (*Ready*): O processo está esperando para ser alocado a um processador;
b. Bloqueado (*Bloqued*): Impossibilitado de ser executado até que algum evento externo aconteça;
c. Executando (*Running*): O processador está executando o processo no momento.

A **Error! Reference source not found.** mostra a transição dos processos pelos diferentes estados. A transição 1 acontece quando o processo que está sendo executado não pode prosseguir. Um exemplo dessa transição ocorre quando um processo faz uma chamada a uma impressora.

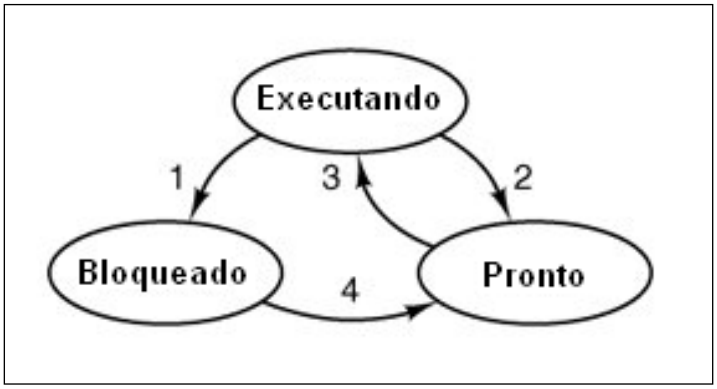


Figura 4: Transição dos processos
Fonte: Tanenbaum (2006)

As transições 2 e 3 são causadas pelo escalonador de processos, que parte do sistema operacional. A transição 2 ocorre quando o escalonador decide que o processo que está sendo executado já utilizou o processador por tempo suficiente. A transição 3 acontece quando o escalonador decide que já é hora do processo que estava aguardando a CPU, poder utilizá-la.

A transição 4 acontece quando o evento externo que o processo estava esperando (por exemplo: a resposta da impressora) acontece. Se nenhum outro processo estiver executando, então a transição 3 será disparada automaticamente e o processo irá iniciar sua execução.

Tanenbaum (2006) lembra que nos sistemas operacionais antigos, cada processo possuía seu próprio espaço de memória e um único fluxo ou linha de execução. Entretanto existem situações em que é desejado que o processo tenha mais de um fluxo de execução, podendo executar diferentes tarefas em paralelo como se fossem processos independentes (excetuando o fato desses diferentes fluxos compartilharem o mesmo espaço de memória). Esses fluxos de execução chamam-se *threads.*

Silberschatz *et al* (1999) afirmam que as *threads* permitem que múltiplas execuções sejam realizadas em um mesmo processo. A figura 3(a) mostra 3 processos tradicionais. Cada processo tem o seu próprio endereço na memória e apenas 1 thread. Diferentemente a figura 3(b) apresenta um processo único com 3 *threads*.

Múltiplas *Threads* permitem que processos sequenciais possam realizar chamadas a sistemas bloqueantes, ao mesmo tempo que explora o paralelismo. Em um processo com múltiplas *threads*, enquanto uma *thread* está esperando, uma outra pode executar uma tarefa diferente.

Processos precisam trocar informações com outros processos. Um dos motivos é a sincronização das atividades realizadas por eles. Se o processo B necessita de informação produzida processo A, o processo B terá que esperar até que A termine de ser executado. Essa necessidade de troca de informação também vale para threads.

Uma das formas de se gerenciar exclusões mútuas é com o uso de *mutex*. Um *mutex* é uma variável que possui 2 estados: travado e destravado. Quando um processo ou *thread* precisa acessar uma região crítica da memória ele "trava" a região impedindo que outros processos ou *threads* acessem a mesma área simultaneamente. Quando o processo ou *thread* termina o uso ele libera a região da memória destravando-a.

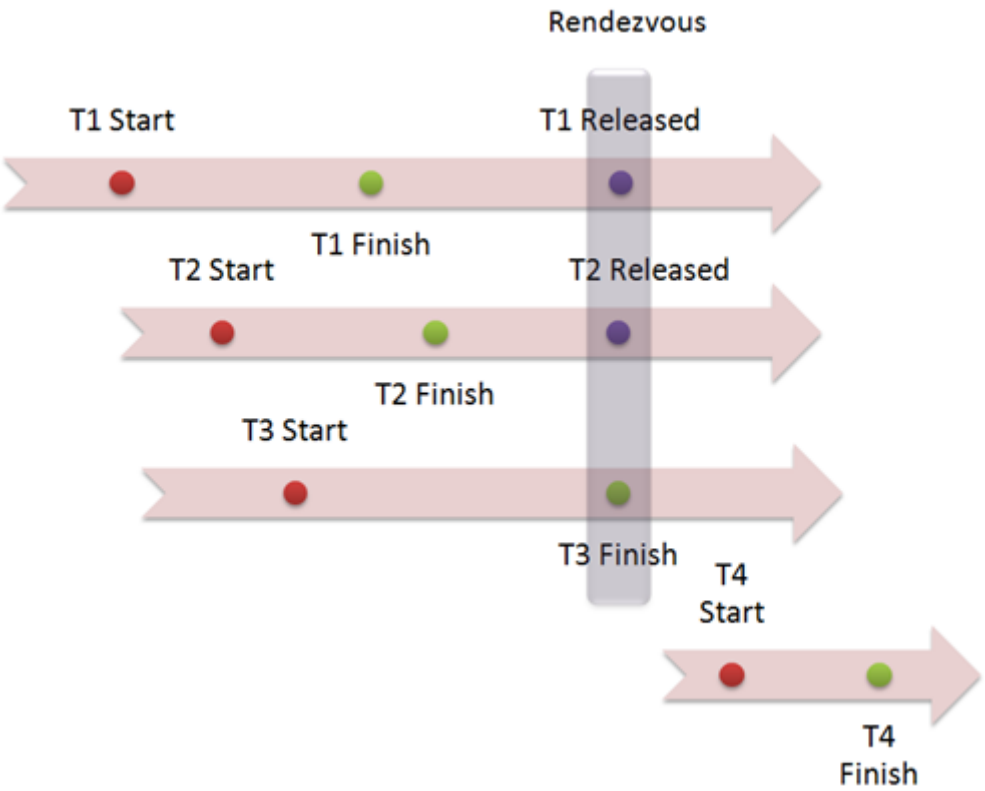


Figura 5:Barreira de capacidade 3
Fonte: Retirado de <http://www.rgoarchitects.com/nblog/content/binary/WindowsLiveWriter/RendezvousThreadsynchronization_284/image_4.png> Acesso em: 27/10/2009

Uma forma mais elaborada de realizar sincronização de atividades é com o uso de barreiras (ou rendezvous). De acordo com a **Error! Reference source not found.** quando uma *thread* ou processo encontra uma barreira, ela ficará em espera até que um número específico de *threads* encontre a mesma barreira, para só então seja possível que todas as threads possam continuar suas execuções.

2.3 SISTEMAS AUTOMÁTICOS DE TESTES

Sistemas Automáticos de testes (ATS) são derivados dos equipamentos manuais de testes. Weiss (1994) explica que a necessidade de um ATS se originou do aumento da complexidade do dispositivo de teste (DUT ou UUT). Pelo fato de ser muito custoso manter uma réplica de todo o ambiente que o dispositivo será inserido, pelo fato dos resultados do ATS ter uma grande confiança e porque o ATS pode gerar sinais mais complexos que um simulador, o ATS foi introduzido e se tornou parte integrante de uma linha de produção.

Quando o ATS foi introduzido, as linhas de produção eram grandes em detrimento da baixa confiabilidade de seus produtos. Essa situação, juntamente com a baixa diferenciação dos produtos favoreceu ATS dedicados para um determinado modelo de dispositivo.

Com o avanço da tecnologia, os dispositivos tiveram um aumento contínuo na sua complexidade e o uso dos ATS fez com que a confiabilidade dos testes fosse aumentada junto com a complexidade dos produtos. Isso gerou uma grande dependência dos produtos em relação a um sistema de teste automático.

Normalmente um ATS tem um ciclo de vida mais longo que o dispositivo que a UUT que ele suporta. Conforme novos dispositivos eram criados, eles geralmente passaram a ser testados nos ATS de produtos anteriores. Essa capacidade de

suportar diferentes produtos veio do desenvolvimento do *software* de teste, conhecido como plano de teste do produto(TPS).

A customização dos programas de TPS, entretanto, não é tão trivial. Cada UUT tem um *hardware* específico e o plano de teste está fortemente amarrado com o *hardware* que a ATS possui. Uma característica nova de um dispositivo pode requerer a aquisição de um novo ATS ou ao menos uma atualização de seus equipamentos.

O modelo tradicional de desenvolvimento de TPS desencoraja o reuso de *software*. As técnicas atuais de programação de TPS, normalmente amarram uma função de *software* com a próxima. Essas funções são geralmente específicas a um produto ou família de *software* e por isso não podem ser usadas em outros tipos de dispositivos.

Essa amarração de funções faz com que os testes sejam executados sequencialmente. Esses testes são aplicados em um sub componente do dispositivo por vez, gerando testes custosos e que deixam o ATS ocioso mais de 50% do total do plano de testes (XIA *et al,* 2007).

2.3.1 Otimização De Testes

Atualmente os grandes fabricantes de equipamentos estão sobre uma tremenda pressão para reduzir seus custos e consequentemente para reduzir os custos e tempos de teste (WEISS, 1994). Avanços na linha de produção permitiram que hoje se pudesse produzir mais rápido e mais barato. Entretanto os testes não evoluíram na mesma escala da produção, pois, ainda hoje, os testes são realizados na maior parte das vezes sequencialmente (RIVOIR, 2004)

A aquisição de novos testadores é a solução mais óbvia para reduzir o tempo de teste de produção. Aumentando o número de testadores aumenta-se o número de unidades em teste e assim a capacidade de lidar com uma maior demanda. Entretanto Boctor (2005) afirma que o aumento de testadores requer mais espaço no chão de fábrica, um aumento de investimento, mais operadores e mais tempo de manutenção, o que vai totalmente de encontro à necessidade de reduzir custos.

Soluções recentes com testes concorrentes estão conseguindo reduzir o tempo dos testes de produção sem gerar um aumento de custos e espaço, pois usam os mesmos equipamentos de teste, porém reduzindo o tempo significantemente (MOLAVI e MCPHEETERS, 2007).

Realizar testes concorrentes significa testar múltiplos sub componentes de uma UUT ao mesmo tempo, o que vai ao encontro da estratégia de aumentar a eficiência da utilização dos instrumentos. Isso faz com que aumente a vazão do sistema de teste e consequentemente caia os custos de testes significantemente.

Xia *et al* (2007) afirma que com os ATS de hoje concentrados em aumentar a eficiência dos testes, é crítico a necessidade de maximizar a vazão do sistema e fazer melhor uso dos, cada vez mais caros, equipamentos de testes.

Segundo dados da National Instruments (2006), a maioria dos sistemas de testes não concorrentes testa apenas uma funcionalidade por vez, deixando ociosos por bastante tempo os seus equipamentos. Em um teste de um produto o tempo total pode ser calculado como a soma do tempo efetivo de teste mais o tempo que o operador (humano ou robô) leva para trocar a UUT do testador.

Pode-se considerar, então, o tempo efetivo de teste sendo proporcional ao tempo total do teste. Segundo Rivoir (2004), esse fator de proporcionalidade pode

ser decomposto em vários fatores de correção. Para um testador por unidade de teste esse fator é decomposto segundo a Equação 1.

$$k = \frac{\hat{t}_{Test}}{t_{Test}} = k_{Conc} \cdot k_{Seq} \cdot k_{Fail} \cdot k_{Retest}$$

Equação 1: Tempo dos testes
Fonte: Rivoir (2004)

K_{Conc} representa a redução quando múltiplas funcionalidades de um dispositivo são testadas concorrentemente. Kretest é o número de vezes que a UUT precisa ser retestada. K_{Seq} e K_{fail} são usados quando o teste é feito com múltiplos testadores. Para um testador único, eles podem ser igualados a 1.

Supondo que ao transformar o teste original em teste concorrente, ele não modifique a quantidade de vezes que a UUT precise ser testada (K_{Retest} constante), o resultado do tempo de teste é reduzido a uma fração do tempo original, onde Pconc é a porcentagem do tempo de teste que pode ser "ocultada" em outro teste, ou seja, ser colocada em paralela a outro teste, conforme pode ser observado na **Error! Reference source not found.**

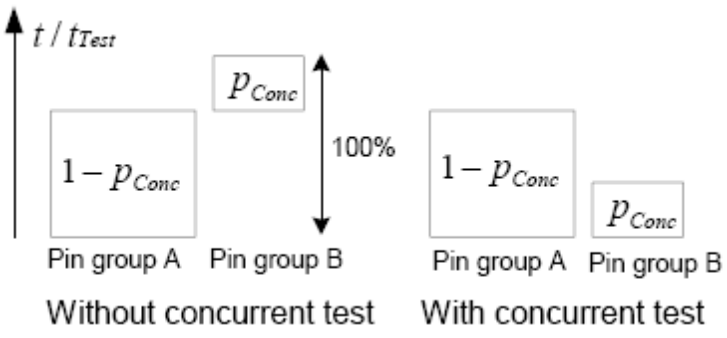


Figura 6: Gráfico de tempo de testes.
Fonte: Rivoir (2000)

Entretanto, os testes concorrentes aumentam e muito a complexidade do TPS. Testes concorrentes geram conflitos entre si devido a limitações de recurso e restrições e dependências geradas pelos requisitos de testes do dispositivo.

Testes concorrentes se não forem cuidadosamente escalonados podem causar falhas de difícil correção e detecção. Por isso, a pesquisa em métodos e modelos de TPS paralelos será fundamental nos próximos anos.

2.4 CONTROLE ESTATÍSTICO DE PROCESSO

O controle estatístico de processo (CEP) foi introduzido no início dos anos 30 por Walter Shewhart da *Bell Telephone and Telegraph*. O objetivo original de Shewhart era prover uma maneira simples e intuitiva de sumarizar a história de um processo (PUC-RS, 2009).

O CEP é um método de análise e melhorias de processos. Os processos são analisados através de amostragem e gráficos de controle (WALKER, 1998). Um dos pilares dos estudos em estatística é a amostragem. Populações, em geral, são grandes demais para serem analisadas em grandes detalhes item por item, já a seleção de amostras de tamanho muito menor que a população reduz os custos e paradoxalmente acaba representando melhor as características da população.

Segundo Carvalho e Paladini (2005), a ideia principal do CEP é que processos de produção com menos variabilidade propiciem níveis melhores de qualidade nos resultados da produção. Uma segunda razão para a aplicação de CEP é que o número e percentagem de peças defeituosas produzidas na fábrica sejam diminuídos com as melhorias na linha de produção.

Qian-Li Zhang (2004) afirma que o sucesso de uma organização está em ser capaz de realizar predições e em se comprometer a produzir o que foi acordado. A

medição possibilita as pessoas detectar tendências e antecipar problemas, propiciando assim um melhor controle de um processo e garantindo que os objetivos sejam alcançados.

De acordo com Carvalho e Paladini (2005), quando o gerente de produção mede uma característica de uma linha de produção, por exemplo, ele tem em mente a melhoria de um processo. O estatístico, por outro lado, por sua vez, vai ver se os números gerados são centrados e simétricos ao redor de uma tendência central, se existe ou não alguns dados muito discrepantes dos outros, se existiu não relações entre variáveis. Portanto, ambos têm muito para ganhar trabalhando juntos.

A seguir são listados os principais indicadores do processo que devem ser medidos e analisados:

a. Tendência central: Uma indicação da localização ou da centralidade da informação. As medidas mais comuns de tendência central são a média e a mediana.
b. Variação: Representam como os dados se espalham ao redor da média. É a flutuação dos resultados do processo. A variância e o desvio padrão são as medidas mais utilizadas.
c. Estabilidade: Representa a variação ao longo do tempo. Estabilidade é o que permite dizer se os resultados são repetíveis.
d. Capacidade do processo: Um processo capaz é um processo estável cuja performance satisfaz os requisitos. A capacidade do processo será discutida em detalhes no tópico seguinte.

2.4.1 Capacidade Do Processo

De acordo com Nist e Sematech (2006), a capacidade do processo é uma propriedade mensurável de um processo atingir a especificação. Ele compara a saída de um processo controlado com os limites de sua especificação, usando índices de capacidade

A capacidade de um processo pode ser avaliada através do índice Cpk. O Cpk é um índice que mede quão consistente um processo está atingindo sua média, ou seja, quão perto está em relação a seus limites (COSTAS, 1992). Quanto maior o índice, menor a probabilidade que o processo saia dos limites especificados.

O índice de capacidade Cpk é definido pela fórmula (ROTH, 2005):

$$Cpk = \text{mín}\left(\frac{\text{LES} - \text{média}}{3\sigma}, \frac{\text{média} - \text{LEI}}{3\sigma}\right),$$

onde LES é o limite de especificação superior, LEI é o limite de especificação inferior e σ é o desvio padrão da amostra, conforme ilustrado na figura 1.

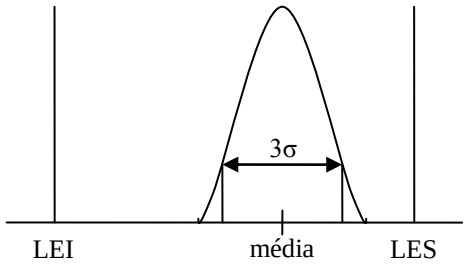


Figura 7: Ilustração do Cpk
Fonte: Roth(2005)

:

Walker (1998) afirma que um índice Cpk igual a 1 significa que o processo produz 2700 defeitos a cada 1 milhão de produtos produzidos. Cpk igual a 1.33 é igual a 63 defeitos por milhão e um Cpk igual a 2 equivale a 3.4 defeitos a cada milhão de produtos. De acordo com Spínola (2007), se o índice Cpk for menor que 1,

o processo é incapaz. Se 1 ≤ Cpk ≤ 1,33, o processo tem capacidade média. Se o Cpk > 1,33, o processo tem alta capacidade.

2.4.2 Diagrama De Pareto

Kielty e Delahunty (1990), explicam que Pareto foi um acadêmico que viveu durante o século 19 e que graficamente demonstrou a distribuição desproporcional entre a riqueza das classes sociais. Ele descobriu que 20% das pessoas detinham 80% da riqueza. Essa relação ficou conhecida como “Princípio de Pareto” ou “Princípio 80/20” e se mostrou verdadeira para várias situações (YU-CHI, 1994).

Segundo Kielty e Delahunty (1990), o diagrama de Pareto também é conhecido como os poucos vitais e os muitos triviais e prioriza as áreas relevantes. Algumas vezes um problema está tão internamente escondido que se torna difícil até saber como começar a solucioná-lo.

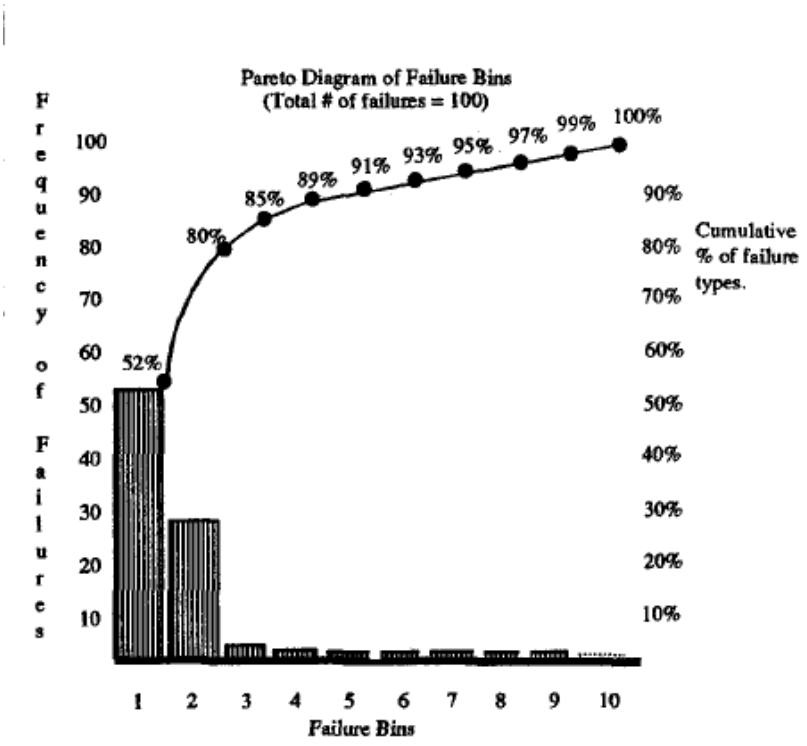


Figura 8: Diagrama de Pareto
Fonte: Kielty (1990)

A figura 2 mostra um diagrama de Pareto. O eixo da esquerda representa a quantidade de produtos rejeitados por falha. O eixo da direita representa o percentual cumulativo de produtos rejeitados por falha. Desse modo, o diagrama de Pareto mostra que os dois primeiros tipos de falha representam 80% de todos os produtos rejeitados. Assim, para melhor utilizar um número limitado de técnicos disponíveis, a empresa deve se concentrar em identificar e remover as causas dos primeiros dois tipos de falhar

3 AVALIAÇÃO DOS TEMPOS DE TESTE DE PRODUTOS ELETRÔNICOS

O projeto de redução de tempo das sequencias de testes de manufatura surgiu da necessidade de tornar mais rápida e econômica a etapa de testes de produtos eletrônicos.

Neste projeto, foi usado como material de suporte técnicas de processamento em paralelo, testes concorrentes e também alguns conceitos de controle estatísticos de processo para validação das novas sequencias otimizadas.

O conceito de otimização de tempos, no presente trabalho, será utilizado como redução de tempos, pois será feita de maneira empírica, baseada na observação da utilização dos instrumentos de teste.

O projeto demonstrará um método simples e eficaz de análise dos passos de teste, para escolher poucos passos relevantes (que contribuem com mais tempo com o tempo total da sequência de testes) dentre um universo de centenas de testes.

Após a análise será mostrada uma técnica para redução de tempo, empregando paralelismo dentro e entre os passos de testes da sequência.

O projeto mostrará um caso real de redução de tempo de teste de um produto eletrônico em duas fases diferentes: Fase de calibração e fase de testes funcionais. Será abordada as ferramentas e técnicas usadas para a criação da metodologia, assim como os critérios de validação da mesma.

O presente trabalho também mostrará as ferramentas de análise e automação da otimização que foram desenvolvidas ao longo de todo projeto. A primeira permite analisar os possíveis testes candidatos à otimização, identificar os pontos em que possam ser empregados testes concorrentes (paralelos) e a verificar a eficiência e

eficácia dos resultados baseado nas comparações dos tempos dos testes e dos resultados em relação aos limites.

A segunda ferramenta servirá para que o desenvolvedor de sequência de testes possa, baseado em um conjunto de regras, criar ou otimizar um plano de testes ótimo.

3.1 METODOLOGIA PARA A REDUÇÃO DOS TEMPOS DE TESTE

A metodologia proposta se divide em duas partes. A primeira parte explica como, dentro de um universo de centenas de passos de testes, identificar uma dezena de passos que são realmente relevantes e que mais contribuam com o tempo total da sequência de teste.

A segunda parte da metodologia trata da redução dos tempos identificados anteriormente. Nessa parte explica-se o porquê dos passos de testes demorarem e como reduzir o tempo de cada teste.

3.1.1 Criação de uma *Baseline*

As seguintes versões foram registradas para que ao final do trabalho os resultados e tempos dos testes originais pudessem ser comparados com os testes após a otimização exatamente nas mesmas condições.

a. Versão do TestStand
b. Versão do programa executor de testes
c. Versão da linguagem que compilou o código-fonte dos testes
d. *API* dos dispositivos
e. Código-fonte dos drivers dos instrumentos
f. *Firmware* dos equipamentos de teste

Foram utilizadas seis amostras do mesmo modelo do dispositivo em teste para a criação da *baseline*. Cada amostra foi testada dez vezes em sequência em modo manual. Isto é, o dispositivo foi inserido no testador, testado e ao final do teste o dispositivo foi retirado e recolocado no testador. Informações obtidas através dos engenheiros da linha de produção do telefone alertaram que em modo automático o dispositivo que está sendo testado pode aquecer durante execuções consecutivas e apresentar falhas ou diferenças de tempo de teste.

A sequência de teste utilizada foi a mesma utilizada na produção do dispositivo pela fábrica. O computador que realizou o teste executou apenas os aplicativos necessários para que os testes fossem realizados. Não havendo, assim, processos extras que pudessem interferir na execução do teste.

No caso da falha de uma amostra, seus resultados não foram considerados, entretanto foi reportado que a amostra falhou, e também em que passo de teste e a respectiva razão da falha. O teste então foi repetido. Essa decisão foi tomada para que uma falha não produzisse um desvio no resultado do teste, ou seja, somente foi levado em consideração o desvio dos testes com sucesso. O número de falhas foi avaliado em separado.

Cada grupo de teste, realizado com a mesma amostra foi salvo em um relatório independente. Depois que os testes foram realizados, os testes que falharam, juntamente com o primeiro teste foram eliminados do relatório. O primeiro teste teve de ser eliminado do relatório devido a sequência de inicialização que o testador executa para carregar os drivers. Essa inicialização ocorre apenas na primeira execução do programa de teste.

Informações oriundas da fábrica relataram um pequeno problema de vazamento de memória gerada por essa sequência de teste. Para mitigar esse problema, para cada grupo de testes o programa executor de teste foi reinicializado.

Os testes tiveram de ser executados com a mesma configuração utilizada na fábrica. Essa configuração não permite depurar os testes, entretanto os tornam mais rápidos e diminuem a possibilidade desses pontos de depuração influírem no resultado da *baseline*.

Os resultados da *baseline* foram analisados individualmente ou agrupados, conforme mostra a Tabela 1. Com base nesses valores, foi possível construir um histograma que agrupou os passos de testes pelo tempo consumido e que mostrou o quão representativo os grupos são em relação ao tempo de teste total.

Descrição	Agrupamento
Média de tempo e dos valores retornados de cada passo	Por amostra
Média do tempo de todo o teste	Por amostra
Cálculo dos valores médios e do desvio padrão dos tempos e dos valores	Por amostra
Cálculo dos valores médios e do desvio padrão dos tempos e dos valores	Total
Cálculo do índice CPK dos resultados de cada passo de teste	Total

Tabela 1: Resultados da baseline

3.1.2 Seleção Dos Testes Para Serem Otimizados

Uma sequência de teste de um produto eletrônico da empresa estudada pode ter centenas de passos de testes, tornando inviável a tentativa de otimização de todos os passos de testes.

De acordo com o Princípio de Pareto (Princípio 80/20) existe um grande desequilíbrio entre causas e efeito. O Princípio afirma que 80% dos resultados alcançados são consequências de 20% dos esforços gastos.

Com base no Princípio de Pareto, foi realizada uma investigação para descobrir os testes mais relevantes, ou seja, os que contribuem com a maior parte do tempo dos testes, para dessa forma ser possível otimizar apenas os testes mais demorados.

Uma vez que o programa executor de testes não fornece os tempos gastos por cada equipamento, para a aplicação do método proposto, foi necessário implementar funções adicionais no programa de testes original para a observação destes tempos. Essas funções adicionais afetaram o desempenho do sistema, aumentando o tempo total de teste aproximadamente em 1%.

Como o plano de teste se divide na sequência de teste (TestPlan) e nas funções que cada passo de teste chama (funções do CVI) e tanto o CVI quanto o testplan permitem executar passos/funções em paralelo, a metodologia contemplará os dois ambientes.

A solução mais óbvia para reduzir os tempos de teste é a de se eliminar todos os tempos de espera do teste. Esses tempos de espera são geralmente criados pelo desenvolvedor de teste para fazer com que um determinado teste aguarde o equipamento terminar de ser configurado ou terminar sua execução. Como alguns instrumentos não enviam mensagem de término de operação/configuração, o desenvolvedor de teste adiciona um valor de espera baseado no tempo médio para cada operação. Durante o tempo de espera de um equipamento, um outro equipamento pode ser configurado ou um outro teste pode ser realizado.

Outra solução para reduzir o tempo de testes é executar dois testes independentes ao mesmo tempo. É preciso tomar cuidado para que esses dois testes não influenciem o resultado do outro. Algumas vezes o ruído, vibração ou o

campo eletromagnético produzido por um teste afeta diretamente outros executados em paralelo.

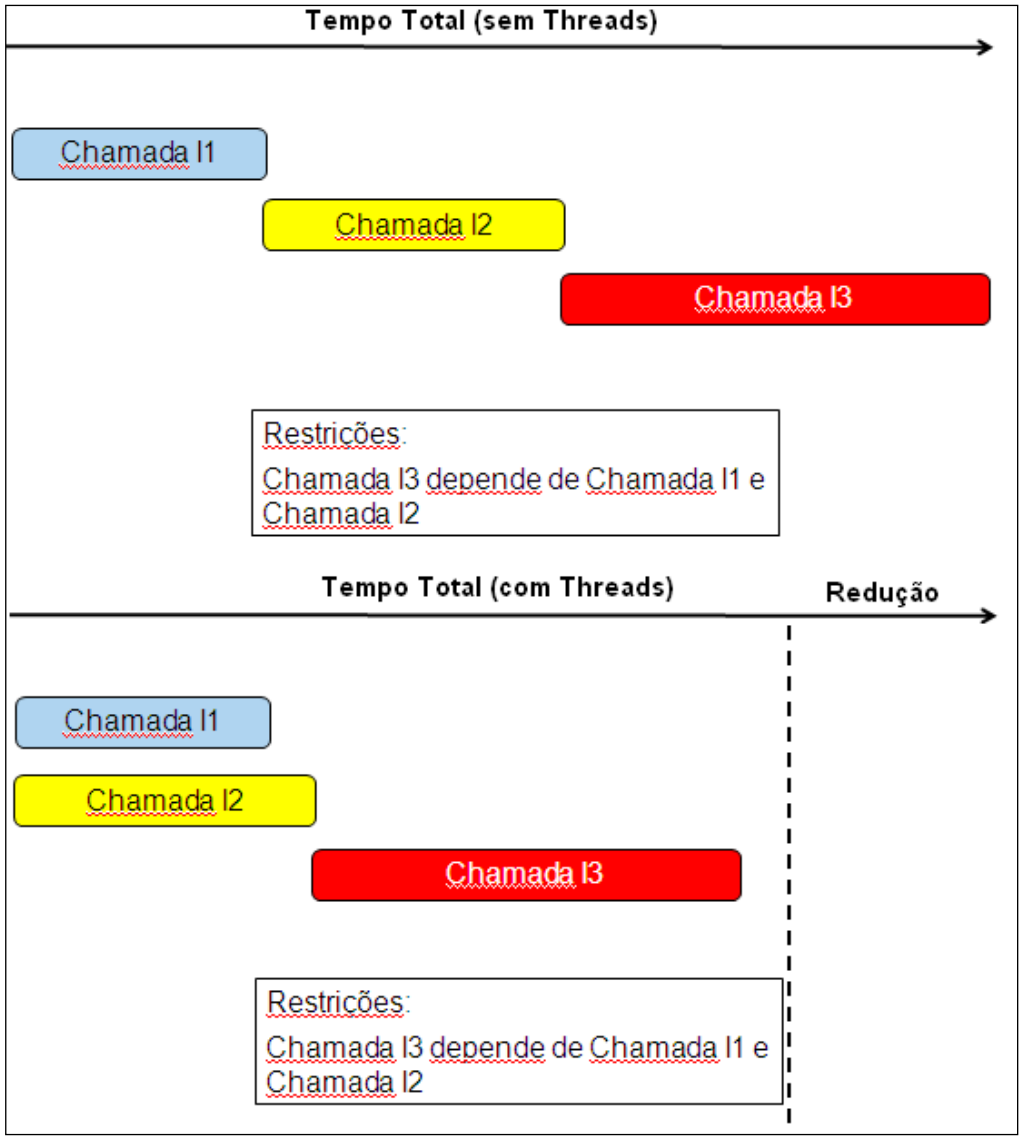


Figura 9: Testes sem e com threads
Fonte: Autor

Pela dependência que existe entre os testes, a existência de recursos limitados e a influência que um teste exerce sobre o outro, não é possível executar todos os testes em paralelo. Por isso, foi proposta a divisão dos testes em configuração, execução e medição dos resultados. Assim, enquanto se configura um ou mais instrumentos é possível realizar outros testes em paralelo.

3.1.3 Paralelismo No CVI

Para possibilitar a execução de testes em paralelo dentro dos testes, ou seja no CVI, foi necessário construir um ambiente e as estruturas de controle para o programa teste executar *multithreads*.

O primeiro passo para implementar múltiplas *threads* foi a construção de um arquivo de cabeçalho com todas as informações pertinentes a thread e suas estruturas. Esse arquivo de cabeçalho definiu:

a. Número máximo de threads ao mesmo tempo

b. Número máximo de parâmetros de cada thread

c. Uma estrutura de tipos contendo todos os valores de parâmetros possíveis

d. A estrutura da thread, que será composta de um campo para registrar as mensagens de erro e outro campo com os parâmetros (item c)

e. Um identificador do *pool* de threads

Foi criado um identificador para cada instrumento de teste diferente, para que fosse possível utilizar qualquer combinação de instrumentos em paralelo. Assim todos os testes poderiam ser habilitados a executar em paralelo. Isso não significa dizer que todos os testes foram executados em paralelo, pois a execução de um teste pode influenciar o resultado de outro, como já foi citado anteriormente.

Foi necessário ainda criar uma função de sincronização, isto é, uma função que espera uma ou mais *threads* terminarem para que se possa continuar o fluxo normal de execução do programa de testes.

Ao término da execução da *thread* se viu necessário a recuperação dos resultados e, no caso de ocorrido algum erro, as mensagens de erro. Como já previsto no cabeçalho (item d), os parâmetros e as mensagens de erro foram recuperadas através de seus respectivos campos.

Quanto aos tipos dos erros registrados, existem 2 possibilidades. O primeiro é quando um determinado teste que executava concorrentemente falha. O segundo acontece se existir algum problema nas sincronizações das diferentes *threads*.

3.1.4 Paralelismo No TestStand

O TestStand, que é a ferramenta utilizada pela empresa para criar e executar a sequência de testes também permite que sejam colocadas instruções em paralelo. Nesse caso, ao invés de se colocar funções em *threads* diferentes, as próprias sequencias foram executadas em *threads* diferentes.

No fluxo normal de execução, cada sequencia ou passo é executado em uma ordem única, um após outro. Quando é determinado que uma sequência deva ser executada em outra *thread*, o fluxo aumenta em um e agora existem dois caminhos de execução.

Também no TestStand é necessário passos de sincronização, tanto para se obter os valores da sequência que foi executada em paralelo, quanto para esperar que uma thread acabe e o fluxo do programa possa continuar.

A F mostra uma sequência que executará o passo *Action1*, e depois executará os passos da sequência *SequenceCall* em paralelo aos passos *Action2* e *Action3*. O passo *Wait_Th* é um passo de sincronização e nele o programa irá esperar que a sequência *SequenceCall* termine sua execução e irá também obter o resultado dela. Como o passo de sincronização está abaixo do passo *Action3*, para o passo *Action4* ser executado, obrigatoriamente os passos *Action3* e a sequência *SequenceCall* devem ter sido executados.

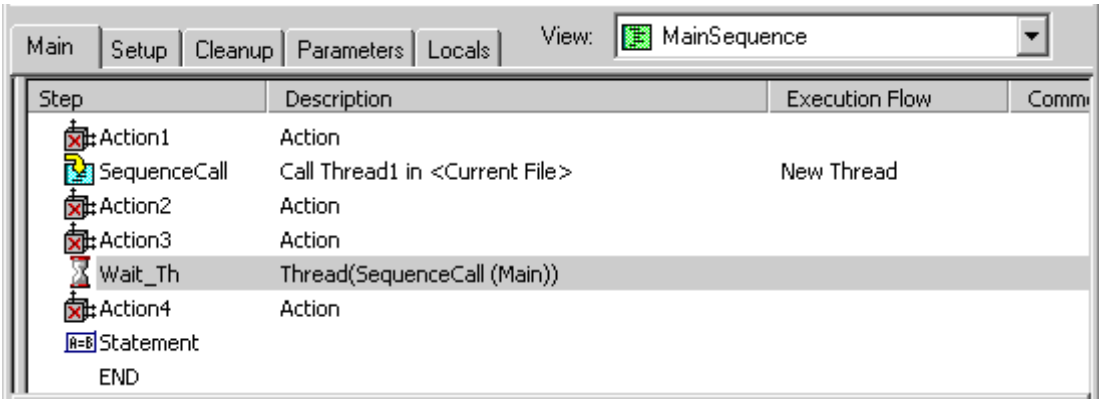


Figura 10: Exemplo de Paralelismo no TestStand
Fonte: Autor

No caso do *TestStand*, o passo de sincronização *Wait* tem visibilidade limitada a sequência que ele está inserido. Para duas sequencias distintas poderem ser sincronizadas é necessário o uso do passo *Rendezvous.*

O *Rendezvous* funciona como uma barreira e precisa que determinado número de mensagens sejam enviadas para ele para que os fluxos de execução possam continuar. Uma *thread* que use um *Rendezvous* ficará bloqueada no passo de sincronização, até que todas as outras threads que usem o mesmo *rendezvous* atinjam o passo de sincronização, conforme explicado na fundamentação teórica.

A F mostra um exemplo de duas sequencias que usam o mesmo *rendezvous* e estão bloqueadas. Nesse exemplo, as sequencias só continuarão a sua execução, quando uma terceira e última thread que usa o mesmo rendezvous atingi-lo. Nesse momento as três sequencias irão seguir a sua execução.

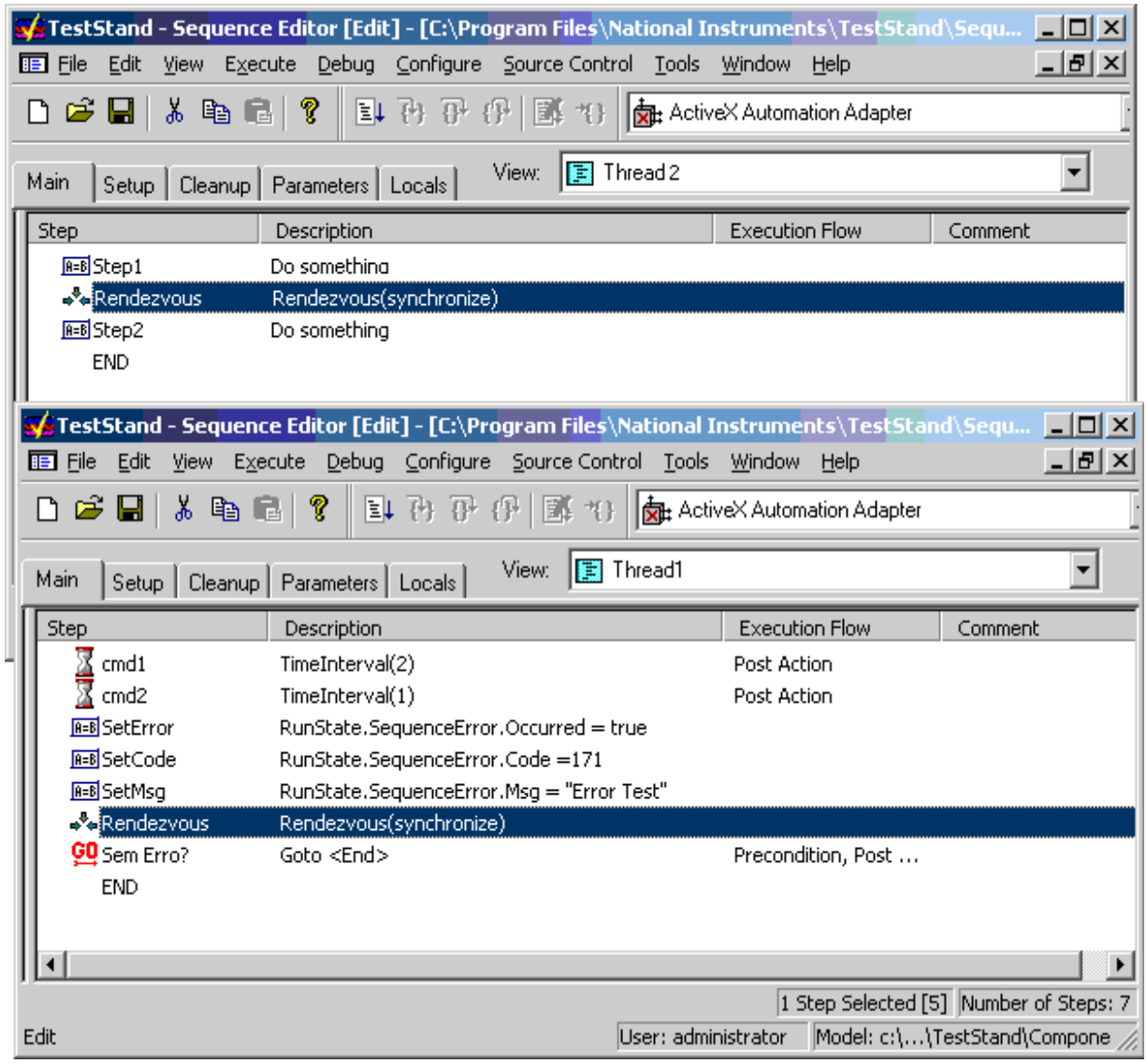


Figura 11: Exemplo do uso de Rendezvous
Fonte: Autor

3.2 EXPERIMENTOS

Foi selecionado um produto de grande volume de produção da fábrica na época dos testes e a metodologia foi aplicada em duas fases distintas: Calibração e Teste Funcional do mesmo produto. Seleção dos candidatos a teste

A sequência de testes do produto escolhido na fase de calibração possuía 197 testes enquanto a sequência de teste da fase de testes funcionais possuía 105 testes. A otimização de 302 testes (197 da primeira sequência e 105 da segunda) é

inviável devido ao tempo que precisaria ser gasto para analisar todos os possíveis pontos de otimização em cada teste.

Para poder ser escolhidos apenas os testes mais relevantes, os que poderiam ter os tempos mais reduzidos, foi gerado um diagrama de Pareto, onde os testes seriam classificados de acordo com o tempo de teste (**Error! Reference source not found.** e Fonte).

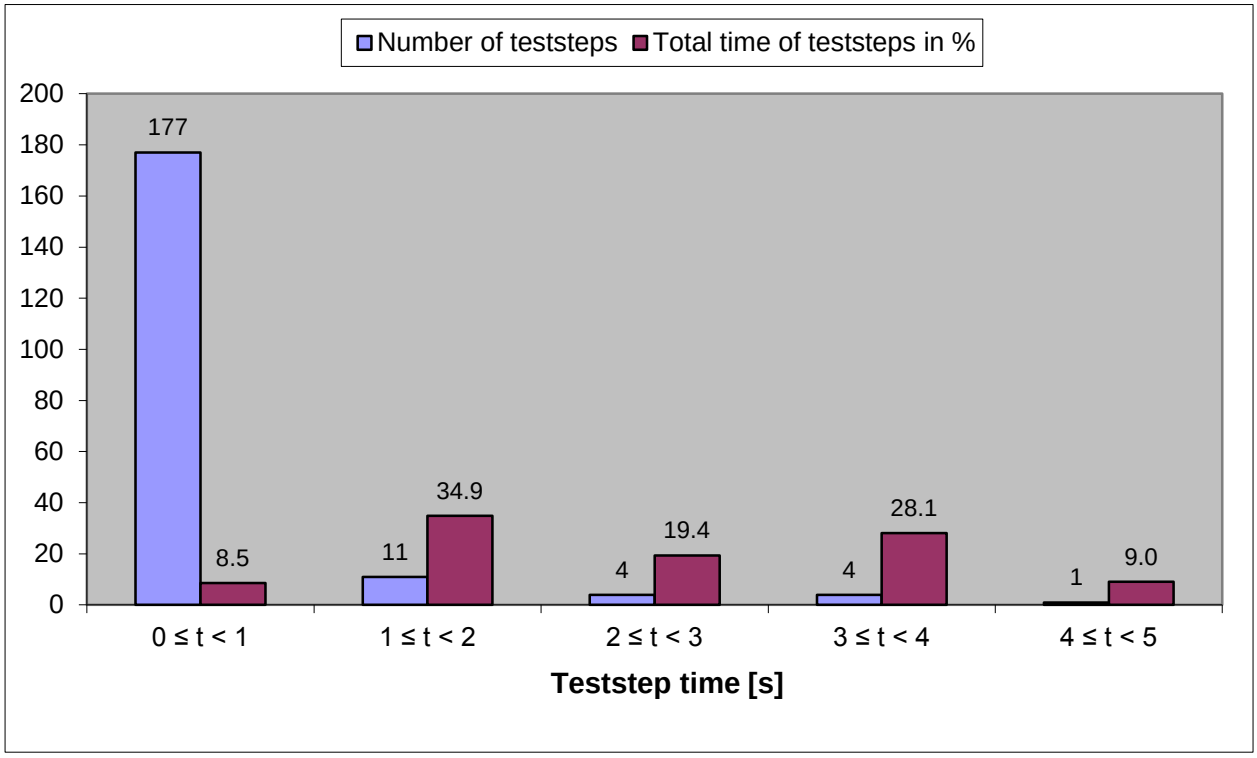


Figura 12: Diagrama de Pareto (Fase de Calibração)
Fonte: Autor

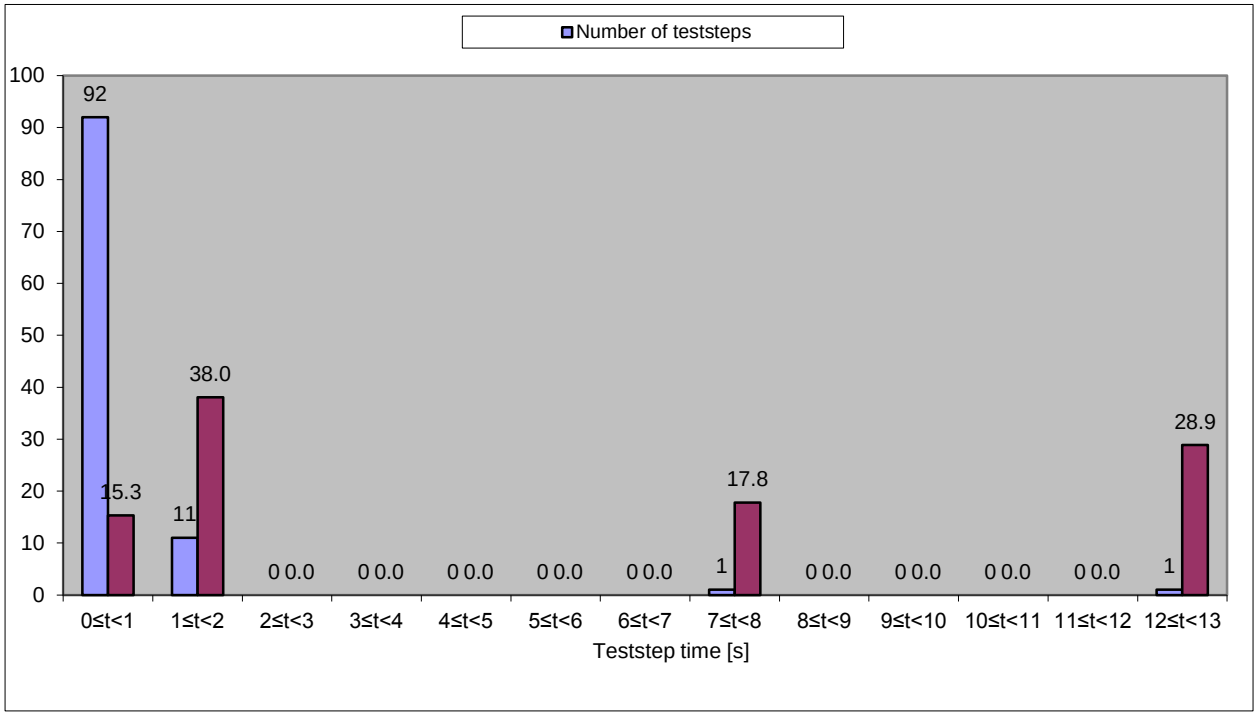


Figura 13: Diagrama de Pareto (Fase de testes funcionais)
Fonte: Autor

Com base no diagrama de Pareto das duas fases foram selecionados 32 testes (19 para a fase de calibração e 13 para a fase de testes funcionais) de um total de 302 testes.

Com o número de testes reduzidos a aproximadamente 10% do total de testes foi possível analisar cada teste individualmente e trabalhar em duas frentes opções de paralelismos.

3.2.1 Paralelismos Dentro Dos Passos De Teste

Um passo de teste é uma chamada de uma função que irá realizar alguma operação (configuração ou medição) em algum instrumento, no computador, ou no produto a ser testado.

Uma função típica pode ser dividida em três partes: a) Configuração dos instrumentos, b) Execução do teste, c) Registro dos resultados.

O registro dos resultados normalmente é bem rápido, pois envolve apenas operações computacionais e não possui relevância para a otimização.

A fase de execução dos testes pode ser demorada e precisa que todos os instrumentos envolvidos no teste estejam previamente configurados e transmitindo ou recebendo alguma informação. Como geralmente um passo de teste executa uma medição (um teste), não é possível aplicar o paralelismo dentro do passo de teste na etapa de medição.

A fase de configuração dos instrumentos de teste também é uma etapa demorada, onde todos os instrumentos necessários para a realização de um teste precisam ser configurados. Nos testes selecionados (testes com alto tempo de execução) normalmente são configurados diversos tipos de instrumentos.

A F mostra um exemplo de otimização obtido na fase de teste calibração. O passo de teste configurava três equipamentos diferentes (caixa de testes, dispositivo e analisador de sinais).

Após configurar o tipo de medida (*GSM*) no analisador de sinais, o analisador necessitava de aproximadamente 300ms antes de receber as informações sobre os parâmetros dos tipos de medida, sendo assim nenhum comando era enviado para nenhum dispositivo e o teste ficava ocioso.

A segunda parte de configurações era no dispositivo, onde demorava cerca de 400 ms para configurar os canais e mais 200ms para começar a transmitir os dados.

Sem Otimização

Ação	Driver	Início (ms)	Fim (ms)	Duração
Configurar rota entre analisador de sinais e dispositivo	Caixa de teste	0	0.015	0.015
Configurar tipo de medida GSM	Analisador de Sinais	0.032	0.328	0.296
Tempo Ocioso (Sleep)	N/A	N/A	N/A	0.33
Configurar parâmetros do sinal GSM	Analisador de Sinais	0.626	0.672	0.046
Configurar freqüência	Dispositivo	0.672	0.907	0.235
Definir canais de transmissão e recepção	Dispositivo	0.907	1.11	0.203
Configurar canal de transmissão para transmitir sinal pulsado	Dispositivo	1.11	1.297	0.187
Configurar freqüência central	Analisador de Sinais	1.297	1.297	0
Definir atenuação	Analisador de Sinais	1.297	1.438	0.141
Configurar o tipo de medição	Analisador de Sinais	1.438	2.376	0.938
Sincronizar fase e freqüência	Analisador de Sinais	2.376	2.406	0.03
Esperar o trigger	Analisador de Sinais	2.406	2.438	0.032
Recuperar o resultado medido	Analisador de Sinais	2.954	2.985	0.031
				2.484

Com Otimização

Ação	Driver	Início (ms)	Fim (ms)	Duração
Configurar rota entre analisador de sinais e dispositivo	Caixa de teste	0	0.015	0.015
Configurar tipo de medida GSM	Analisador de Sinais	0.031	0.328	0.297
Tempo Ocioso (Sleep)	N/A	N/A	N/A	0.33
Configurar parâmetros do sinal GSM	Analisador de Sinais	0.627	0.672	0.045
Configurar freqüência	Dispositivo	0	0.237	0.237
Definir canais de transmissão e recepção	Dispositivo	0.237	0.439	0.202
Configurar canal de transmissão para transmitir sinal pulsado	Dispositivo	0.44	0.627	0.187
Configurar freqüência central	Analisador de Sinais	0.673	0.673	0
Definir atenuação	Analisador de Sinais	0.673	0.814	0.141
Configurar o tipo de medição	Analisador de Sinais	0.814	1.752	0.938
Sincronizar fase e freqüência	Analisador de Sinais	1.752	1.784	0.032
Esperar o trigger	Analisador de Sinais	1.784	1.817	0.033
Recuperar o resultado medido	Analisador de Sinais	2.33	2.361	0.031
				2.488

Figura 14: Otimização de configuração de instrumentos
Fonte: Autor

A estratégia que foi tomada, então, foi a de colocar em *threads* diferentes a configuração da caixa de teste e analisador de sinais e a configuração do dispositivo. Após as duas configurações, foi inserido um sincronizador para que o passo *Configurar frequência central* só fosse iniciado após as duas configurações.

Ao somar o tempo de todas as chamadas aos instrumentos, mais o tempo que o teste ficou ocioso (*Tempo Ocioso*) os resultados são praticamente quando o teste foi executado com e sem otimização. Entretanto, ao comparar o tempo total do teste (*Fim - Início*) é possível verificar que o tempo total do teste foi reduzido em mais de 600 ms (2.985 contra 2.361).

Ao analisar a utilização dos instrumentos dentro de um passo de teste, é possível detectar quando o passo de teste faz chamadas a instrumentos em paralelo ou não. A F mostra uma série de chamadas a instrumentos diferentes dentro de um passo de teste da fase de testes funcionais utilizando paralelismo.

Os instrumentos *MM*, *TB*, *PS* e *PCI* são chamados ao simultaneamente ao longo do mesmo passo de teste. Consequentemente o passo de teste em questão tem a sua duração reduzida.

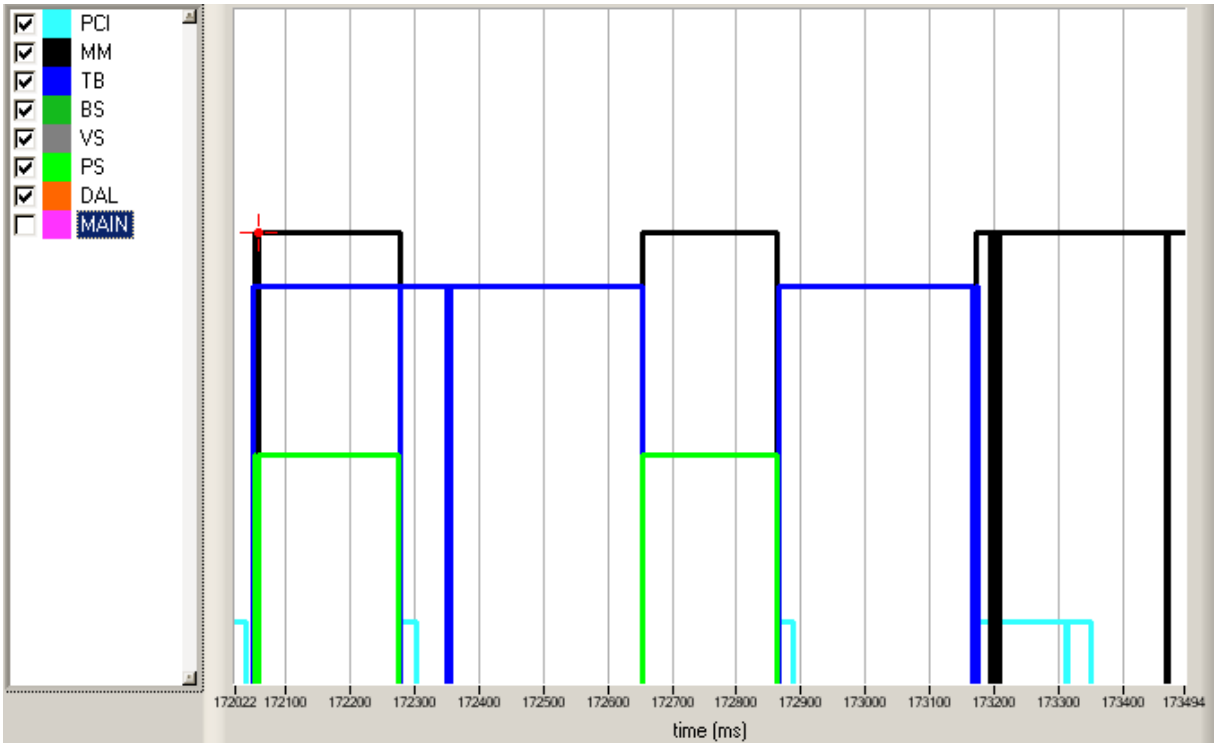


Figura 15: Chamadas de instrumentos em paralelo dentro de um passo de teste
Fonte: Autor

O ganho dos tempos de testes com otimização dentro do passo de testes apenas foi medido em ambas as fases de teste do produto (Calibração e Testes Funcionais) e o ganho em cada fase está demonstrado na figura 15.

A F mostra dois quadros. A coluna 1 indica a amostra que foi feita os testes. A coluna 2 mostra o tempo médio dos 60 testes aplicados em cada amostra.

Calibração

Amostra	Original	PM	PM+DLL
1	54.48	55.57	51.28
2	54.79	54.91	49.49
3	54.19	56.67	48.48
4	54.32	54.67	48.39
5	55.70	56.28	50.38
6	54.34	54.42	48.42
7	54.11	54.34	48.81
8	54.50	54.30	48.45
9	54.56	54.21	48.85
10	54.31	54.34	48.69
Média	54.53	54.97	49.12
Stdev	0.45	0.89	0.98
Redução		-0.81%	9.91%

Teste Funcional

Amostra	Original	PM	PM+DLL
1	50.35	50.63	48.07
2	50.74	49.83	48.07
3	49.3	50.61	47.96
4	49.9	49.96	48.42
5	49.75	49.14	48.55
6	49.52	50.36	47.98
7	50.05	50.07	48.51
8	50.34	50.04	48.13
9	49.38	49.96	47.51
10	49.33	49.95	48.95
Média	49.87	50.06	48.22
Stdev	0.50	0.43	0.40
Redução		-0.38%	3.31%

Figura 16: Ganho com paralelismo dentro dos passos de teste
Fonte: Autor

A terceira coluna mostra o tempo dos testes com a sequência preparada para executar os paralelismos. É possível perceber que em ambos os casos é gerado um overhead na sequência para habilitar o registro dos resultados e a execução de múltiplas threads.

Na última coluna estão os tempos de testes com o ganho do paralelismo dentro do passo deste, ou seja, dentro da biblioteca (*dll*) que se encontra as funções de teste. No caso de se contar apenas o ganho do paralelismo (sem o overhead), a redução nos valores seriam de 10.72% e 3.49% para a fase de calibração e testes funcionais respectivamente .

3.2.2 Paralelismo Dentro Da Sequência De Testes

Uma sequência de testes é um conjunto de passos de teste que executados em uma ordem específica realizará testes em um dispositivo. Uma sequência de testes pode possuir inúmeros passos de teste ou subsequências dentro dela.

As funcionalidades similares a serem testadas de um dispositivo são agrupadas em subsequências, conforme visto na figura abaixo. Essas funcionalidades similares representam um grupo de teste que pode ser aplicado. No caso da sequência de testes funcionais, as subsequências são responsáveis pelos testes de display, câmera, áudio, teclado, entre outros.

Para otimizar as sequencias de teste, a estratégia que foi traçada foi a de executar testes mutualmente independentes em paralelo.

A utilização dos instrumentos é um outro fator relevante também na escolha dos testes ou sequencias a serem executados em paralelo. Testes que usam um mesmo recurso crítico não podem ser executados em paralelo.

Um recurso crítico pode ser considerado como um instrumento ou parte de um instrumento real. Por exemplo, uma fonte de alimentação pode ser considerada como um recurso crítico, porém se a fonte hipotética possuir mais de um canal, cada canal será um recurso crítico independente. No caso das sequencias de teste de calibração e testes funcionais, cada instrumento é um recurso crítico.

Com base na análise de utilização dos instrumentos de teste, é possível identificar testes candidatos a serem otimizados. Na F, foi identificado uma instrução de movimentar o robô da caixa de teste para perto do microfone e após a movimentação um comando longo era enviado ao dispositivo de testes.

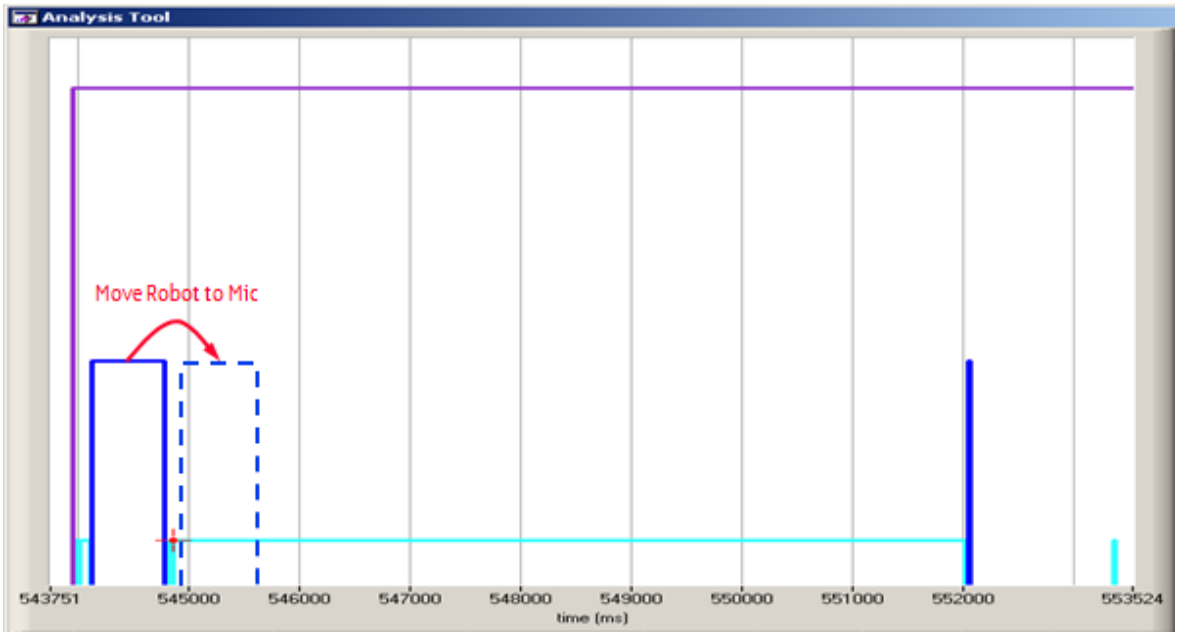


Figura 17: Paralelismo nas sequencias de teste
Fonte: Autor

Nesse caso em específico, o passo de teste responsável pelo movimento do robô foi invocado em paralelo ao comando de configuração do dispositivo, reduzindo o tempo da sequência em 655 milissegundos.

Outra oportunidade de ganho com a otimização nas sequencias é o carregamento de bibliotecas de instrumento de medição em paralelo com algum outro teste.

No caso da sequência de testes funcionais, as bibliotecas dinâmicas de um equipamento de teste eram carregadas no início da sequência de teste e após a otimização foram carregadas em paralelo com as configurações de outro instrumento de teste (**Error! Reference source not found.**), reduzindo em quase 2 segundos o tempo total da sequência.

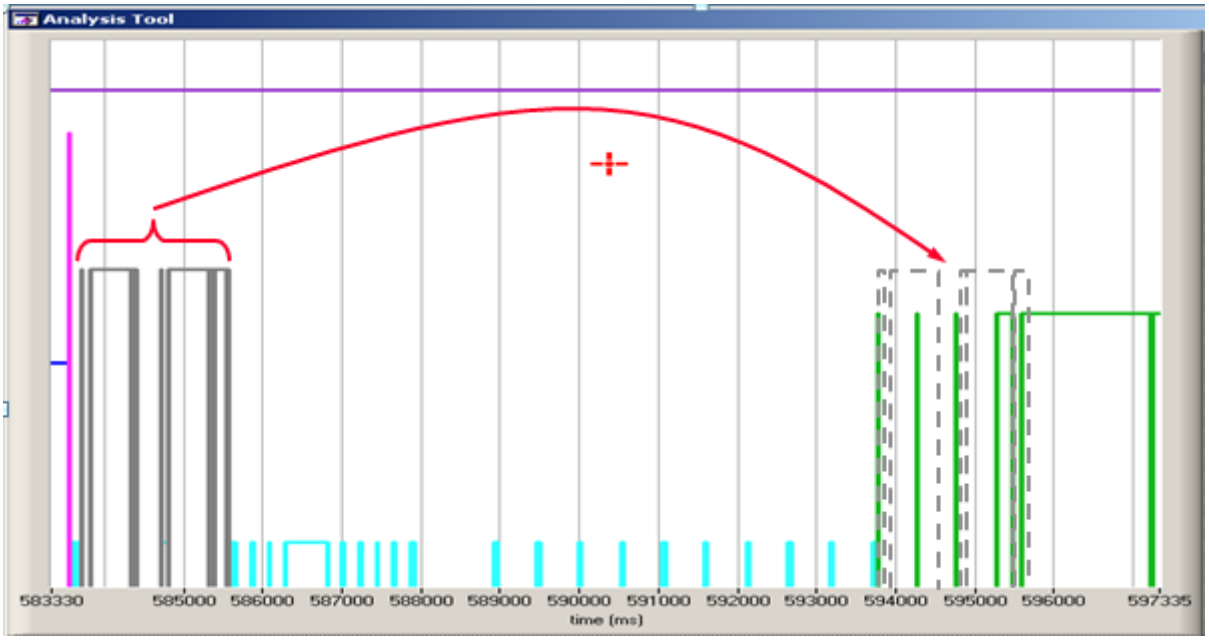


Figura 18: Carregamento de bibliotecas dinâmicas em paralelo
Fonte: Autor

O programa de criação e edição de sequencias de teste da empresa escolhida é a ferramenta Test Stand da National Instruments. Essa ferramenta não permite que um passo seja executado em paralelo, mas permite que subsequências sejam.

Um exemplo de execução de paralelismo em sequencias é apresentado na F. Na sequência de calibração, um passo de configuração do analisador de sinais foi inserido em uma nova subsequência e essa subsequência foi configurada para ser executada em uma thread diferente da *thread* principal de execução.

Um passo de sincronização foi inserido na sequência de teste para impedir que a thread principal prosseguisse sem que a subsequência *Thread Set SA GSM900* tivesse terminado.

Essa sincronização foi fundamental para garantir que a subsequência de configuração do analisador de sinais, que foi executada em paralelo, terminasse

antes de começar a subsequência que iria executar testes com o analisador de sinais já previamente configurados.

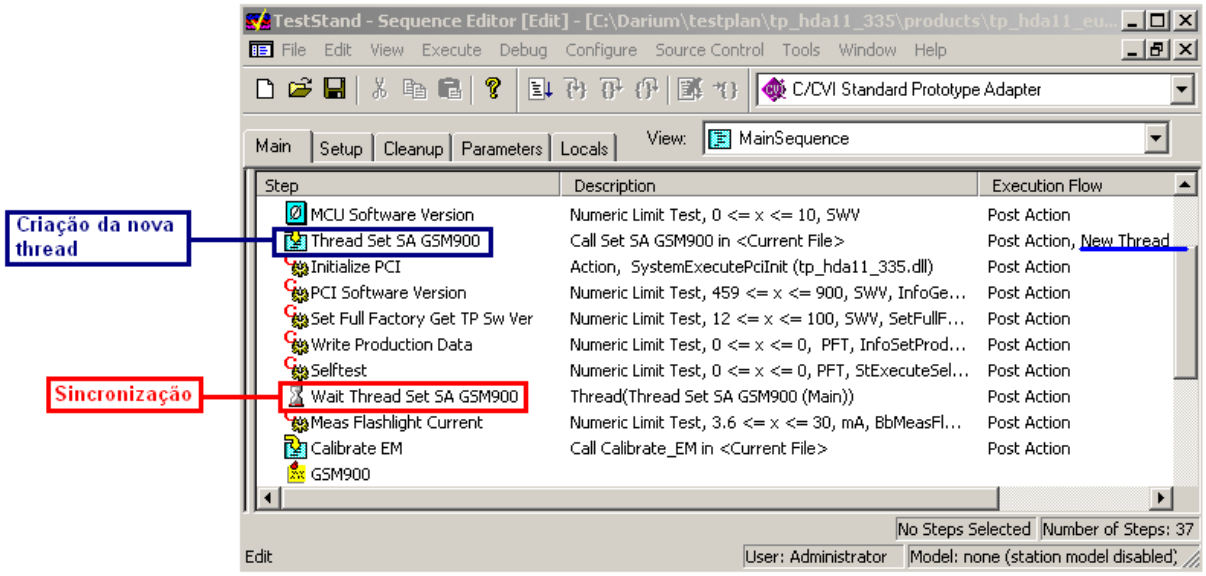


Figura 19: Implementação de paralelismo na sequência de teste
Fonte: Autor

Contudo, não é possível executar testes em paralelo apenas observando a utilização dos instrumentos e as dependências dos testes.

O teste do vibrador do dispositivo juntamente com o teste de teclado. O teste foi invalidado, pois a vibração gerada pelo robô ao pressionar as teclas do dispositivo gerou um ruído de aproximadamente 3 volts no teste do vibrador que tem como limites inferior e superior 0.9V e 10V respectivamente.

Novamente foi tentado executar o teste do vibrador em paralelo com o teste de áudio, porém o ruído (som) gerado pelo vibrador influenciou o teste de áudio o tornando inválido.

Em relação aos ganhos obtidos com o paralelismo na sequencias de testes, a sequência de testes funcionais apresentou uma redução significativamente maior se comparada a sequência de calibração (F).

Calibração

Amostra	Original	PM	PM+DLL	PM+TP+DLL
1	54.48	55.57	51.28	45.53
2	54.79	54.91	49.49	46.23
3	54.19	56.67	48.48	45.25
4	54.32	54.67	48.39	45.46
5	55.70	56.28	50.38	46.79
6	54.34	54.42	48.42	45.27
7	54.11	54.34	48.81	45.24
8	54.50	54.30	48.45	45.41
9	54.56	54.21	48.85	45.53
10	54.31	54.34	48.69	45.23
Média	54.53	54.97	49.12	45.59
Stdev	0.45	0.89	0.98	0.51
Redução		-0.81%	9.91%	16.39%

Testes Funcionais

Amostra	Original	PM	PM+DLL	PM+TP+DLL
1	50.35	50.63	48.07	40.24
2	50.74	49.83	48.07	39.85
3	49.3	50.61	47.96	40.01
4	49.9	49.96	48.42	39.73
5	49.75	49.14	48.55	39.68
6	49.52	50.36	47.98	38.84
7	50.05	50.07	48.51	39.68
8	50.34	50.04	48.13	40.88
9	49.38	49.96	47.51	39.43
10	49.33	49.95	48.95	39.22
Média	49.87	50.06	48.22	39.76
Stdev	0.50	0.43	0.40	0.56
Redução		-0.38%	3.31%	20.27%

Figura 20: Comparativo de ganhos com as otimizações
Fonte: Autor

.

Ao comparar o ganho com otimização apenas dentro das sequencias de teste (e sem contar com o *overhead*), a fase de testes de calibração apresentou um ganho de médio de 7.3% enquanto a fase de testes funcionais apresentou um ganho médio de 17.3%.

Esse comportamento é justificável pela natureza das fases de testes. Enquanto na fase de calibração os testes utilizam basicamente analisadores e geradores de sinais e multímetro em quase toda a sequência, na fase de testes funcionais, os testes funcionais (áudio, display, teclado, etc.) usam equipamentos diferentes para serem realizados e portanto podem ser facilmente executados em paralelo.

3.3 ANÁLISE DOS RESULTADOS

De modo a verificar a eficácia e a eficiência do método foram feitas duas análises distintas dos resultados.

A primeira foi realizada em cima dos resultados dos testes e seus limites de aprovação, para garantir que os testes após a otimização continuem confiáveis.

A segunda análise foi feita em relação ao tempo dos testes e tem como função verificar se a redução dos tempos dos testes foi significante em relação ao tempo das sequencias originais.

3.3.1 Análise Do Cpk Dos Resultados Dos Testes

De acordo com os critérios de aceitação definidos anteriormente, os índices Cpk dos resultados dos testes deveriam ficar acima de 1.33 para garantir que os testes continuariam confiáveis mesmo após as otimizações.

A sequência de calibração dos instrumentos é a que possui os testes mais rigorosos e sensíveis a variações. Os resultados dos testes da sequência original apresentaram índice cpk inferiores a 1.33 em 6 dos 197 testes, sendo que 2 deles tiveram cpk inferior a 1. Já a sequência otimizada apresentou em 7 dos seus 200 testes, resultados com cpk inferiores a 1.33, contudo não apresentou nenhum teste com índice menor que 1.

	Passo de Teste	Limite Inferior	Limite Superior	Média	Desvio Padrão	Cpk
Seqüência Original	GSM1800 Sample Power @ COEF 14	-25.00	37.00	-3.95	5.56	1.26
	GSM1800 Sample Power @ COEF 15	-30.00	37.00	-7.98	6.89	1.07
	Meas Flashlight Current UEMC4.4	70.00	20.00	34.30	1.98	2.41
	Get GSM900 I-DC Offset	-3.50	3.50	-0.69	0.42	2.22
	Get GSM900 Q-DC Offset	-3.50	3.50	0.97	0.70	1.21
	Get GSM900 Aligned Relative f0 power	-200.00	-30.00	-51.35	8.06	0.88
	Get GSM900 Aligned Relative f0+67k power	-200.00	-35.00	-57.47	8.26	0.91
	Get GSM1800 I-DC Offset	-3.50	3.50	-0.69	0.42	2.22
	GetGSM1800 Q-DC Offset	-3.50	3.50	0.97	0.70	1.21
Seqüência	GSM1800 Sample	-25.00	37.00	-5.99	2.62	2.42

Otimizada	Power @ COEF 14					
	GSM1800 Sample Power @ COEF 15	-30.00	37.00	-10.65	3.62	1.78
	Meas Flashlight Current UEMC4.4	20.00	70.00	28.88	2.93	1.01
	Get GSM900 I-DC Offset	-3.50	3.50	-1.26	0.61	1.23
	Get GSM900 Q-DC Offset	-3.50	3.50	1.43	0.62	1.11
	Get GSM900 Aligned Relative f0 power	-200.00	-30.00	-51.25	4.90	1.45
	Get GSM900 Aligned Relative f0+67k power	-200.00	-35.00	-58.22	5.39	1.44
	Get GSM1800 I-DC Offset	-3.50	3.50	-1.26	0.61	1.23
	GetGSM1800 Q-DC Offset	-3.50	3.50	1.43	0.62	1.11

Quadro 2: Cpk inferiores a 1.33 na sequência de calibração
Fonte: Autor

A explicação de esses testes estarem abaixo do limite de 1.33 é que o resultado dos testes variam de acordo com a utilização das agulhas de teste, temperatura do dispositivo e do próprio valor de atenuação da tabela de calibração. Engenheiros e pesquisadores especialistas de diferentes centros de pesquisa da mesma empresa avaliaram os testes e garantiram que os mesmos estavam confiáveis.

Os dados da baseline indicam que na fase de testes funcionais, 2 dos 105 testes já possuíam um Cpk inferior a 1.33. Esse resultado foi melhorado após a otimização, como mostra o quadro 1. Os dois primeiros testes do quadro na sequência original que apresentavam índices Cpk abaixo de 1.33 passaram a ter índices acima de 2.

Essa melhora na confiabilidade dos testes se deve a alteração das posições dos testes durante a implementação de paralelismo na sequência otimizada.

Já a sequência otimizada apresentou um teste com cpk abaixo de 1.33, caso que não ocorria com a sequência otimizada. Pelo fato da média do teste e do desvio padrão serem relativamente pequenos em relação aos limites de aceitação do teste, nenhuma medida foi tomada.

Sequência	Passo de Teste	Limite Inferior	Limite Superior	Média	Desvio Padrão	Cpk
Sequência Original	GSM1800 Phase Error RMS MID 0	-5.00	5.00	3.24	0.75	0.78
	Check HOOK wo MicBias2	2400.00	3700.00	2737.52	3.80	29.61
Seqüência Otimizada	GSM1800 Phase Error RMS MID 0	-5.00	5.00	2.58	0.28	2.85
	Check HOOK wo MicBias2	2400.00	3700.00	2490.72	23.91	1.26

Quadro 3:Cpk inferiores a 1.33 na sequência de Testes Funcionais
Fonte: Autor

3.3.2 Análise Dos Resultados Dos Tempos De Teste

Ao analisar a comparação entre os tempos de testes individuais da baseline e da sequência após a otimização da etapa de testes de calibração, nota-se que os tempos individuais dos testes otimizados foram reduzidos de forma expressiva, como mostra a **Error! Reference source not found.**.

O somatório dos testes da tabela sem otimização apresenta uma diferença de aproximadamente 6 segundos e representa os ganhos obtidos com os paralelismos dentro dos passos de teste.

Sem Otimização		
Teste	Tempo(ms)	Desvio padrão (ms)
BbTuneEm	4144	13
GSM900 RfTuneTxPower	1892	47
GSM1800 RfTuneTxPower	2178	356
Meas Tx GSM900 Ramping Spectrum_HIGH	1885	15
Meas Tx GSM900 Ramping Spectrum_LOW	3015	8
Meas Tx GSM1800 Ramping Spectrum_HIGH	3016	8
Meas Tx GSM1800 Ramping Spectrum_LOW	3062	90
Com Otimização		
Teste	Tempo(ms)	Desvio padrão(ms)
BbTuneEm	2997	10
GSM900 RfTuneTxPower	1494	4
GSM1800 RfTuneTxPower	1495	4
Meas Tx GSM900 Ramping Spectrum_HIGH	1095	10
Meas Tx GSM900 Ramping Spectrum_LOW	1998	38
Meas Tx GSM1800 Ramping Spectrum_HIGH	2002	33
Meas Tx GSM1800 Ramping Spectrum_LOW	1998	26

Figura 21: Tempo dos testes individuais
Fonte: Autor

Na fase de testes funcionais, entretanto, a diferença entre o somatório de todos os tempos de testes individuais é pequena, se compararmos à diferença entre os tempos das sequencias de testes original e baseline, como mostra o Quadro 4

Resultados	sequência Original	sequência Otimizada
∑ Testes (ms)	44726	42734
Desvio padrão (ms)	893	939
Tempo Total (ms)	49866	39756
Desvio padrão (ms)	498	559

Quadro 4: Comparação dos tempos na sequência de Testes Funcionais
Fonte: Autor

O somatório dos tempos dos testes ficou 2.3% menor na sequência otimizada, porém ao contar o tempo total gasto das sequencias, foi obtido uma redução de 20.3% da sequência otimizada em relação à original.

Resultados	Seqüência de Calibração	Seqüência de Testes Funcionais
Tempo Total Original (ms)	49866	54530
Desvio padrão Original (ms)	498	453
Tempo Total Otimizado (ms)	39756	44050
Desvio padrão Otimizado (ms)	559	441
Ganho com otimização (ms)	10110	10480
Ganho com otimização (%)	20.27%	19.22%

Quadro 5: Sumário dos resultados
Fonte: Autor

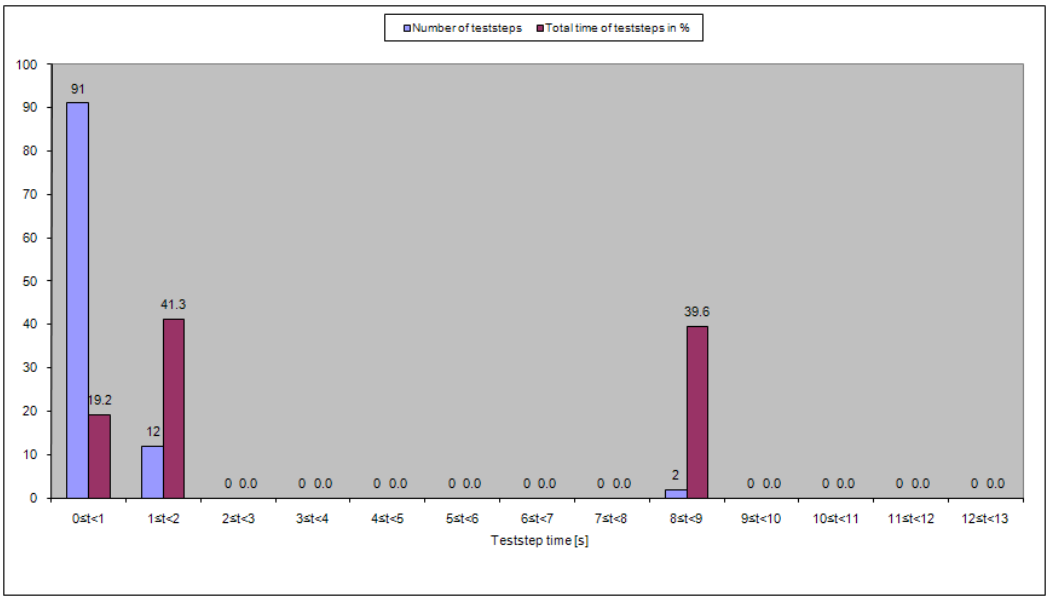
3.4 FERRAMENTAS DESENVOLVIDAS

Para ajudar no trabalho de análise dos dados e posteriormente na automação do processo de redução dos tempos de teste, foram desenvolvidas duas ferramentas . A primeira delas chama-se *Time Stamp Tool* e tem como objetivo permitir a visualização da utilização dos instrumentos, bem como a permitir a análise da sequência de testes. A segunda ferramenta, *Test Plan Optimizer*, consegue através de uma série de regras estipuladas pelo desenvolvedor de testes, gerar uma sequência otimizada baseada na simulação de milhares de sequencias possíveis.

3.4.1 Time Stamp Tool

De modo a analisar os dados da *baseline* tornou-se necessário a leitura dos tempos de testes para a categorização dos mesmos de acordo com os seus tempos. A primeira solução foi exportar cada passo de teste, juntamente com os seus tempos para uma planilha Excel.

Com base na planilha, foi possível criar um gráfico de colunas (histograma) e criar diversas séries, filtrando pelo tempo conforme mostrado na F.



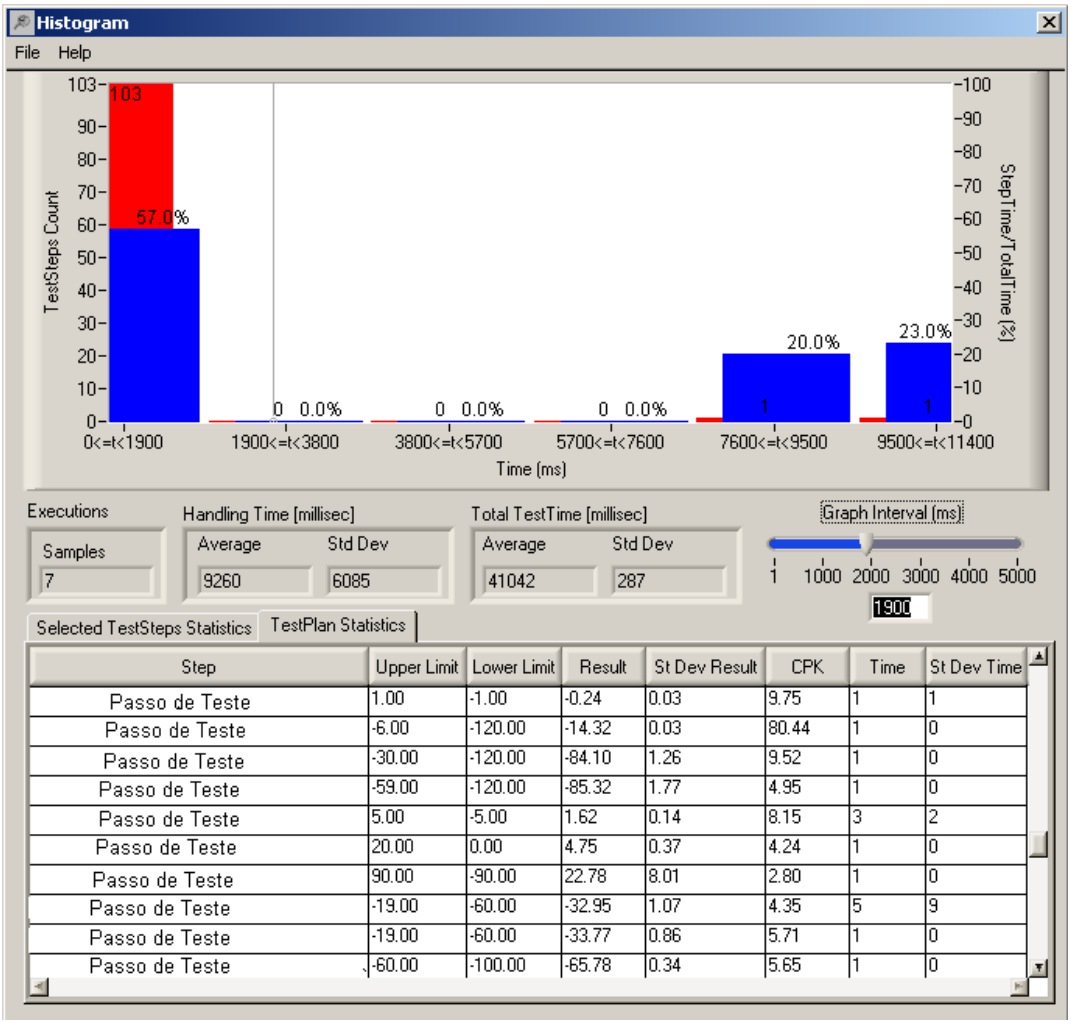
Condition	Total count	Count	Total time [s]	Time [%]	Category (x) axis label
< 1000	91	91	8.195	19.2	0≤t<1
< 2000	103	12	25.836	41.3	1≤t<2
< 3000	103	0	25.836	0.0	2≤t<3
< 4000	103	0	25.836	0.0	3≤t<4
< 5000	103	0	25.836	0.0	4≤t<5
< 6000	103	0	25.836	0.0	5≤t<6
< 7000	103	0	25.836	0.0	6≤t<7
< 8000	103	0	25.836	0.0	7≤t<8
< 9000	105	2	42.762	39.6	8≤t<9
< 10000	105	0	42.762	0.0	9≤t<10
< 11000	105	0	42.762	0.0	10≤t<11
< 12000	105	0	42.762	0.0	11≤t<12
< 13000	105	0	42.762	0.0	12≤t<13
				100.0	

Figura 22: Histograma
Fonte: Autor

.

Com base no diagrama de Pareto foi possível identificar quais testes possuíam uma maior relevância em relação ao tempo total da sequência .

Posteriormente, no decorrer do trabalho de otimização, foi necessária uma análise mais profunda, onde além de separar os testes por tempo, ainda fosse possível analisar cada detalhe do teste que compunha o histograma. A **Error! Reference source not found.** mostra a tela de histogramas da ferramenta *Time Stamp Tool*.



Step	Upper Limit	Lower Limit	Result	St Dev Result	CPK	Time	St Dev Time
Passo de Teste	1.00	-1.00	-0.24	0.03	9.75	1	1
Passo de Teste	-6.00	-120.00	-14.32	0.03	80.44	1	0
Passo de Teste	-30.00	-120.00	-84.10	1.26	9.52	1	0
Passo de Teste	-59.00	-120.00	-85.32	1.77	4.95	1	0
Passo de Teste	5.00	-5.00	1.62	0.14	8.15	3	2
Passo de Teste	20.00	0.00	4.75	0.37	4.24	1	0
Passo de Teste	90.00	-90.00	22.78	8.01	2.80	1	0
Passo de Teste	-19.00	-60.00	-32.95	1.07	4.35	5	9
Passo de Teste	-19.00	-60.00	-33.77	0.86	5.71	1	0
Passo de Teste	-60.00	-100.00	-65.78	0.34	5.65	1	0

Figura 23: ***Time Stamp Tool*** **(Histograma)**
Fonte: Autor

Nessa tela era possível analisar não apenas os dados da sequência, assim como o número de amostras, a média do tempo de manipulação do operador, a média do tempo de teste e o desvio padrão, mas também os dados de cada teste individual como o seu resultado, os limites (inferiores e superiores), o desvio padrão, o índice Cpk e o tempo e seu desvio padrão.

Com essa ferramenta ainda era possível combinar diversos arquivos de logs para aumentar o número de amostras e garantir uma maior confiabilidade nas análises.

Para identificar onde seria possível aplicar o paralelismo, tanto dentro dos passos de testes como dentro da sequência de testes, foi gerado um gráfico de utilização de instrumentos numa planilha Excel, conforme mostra a F.

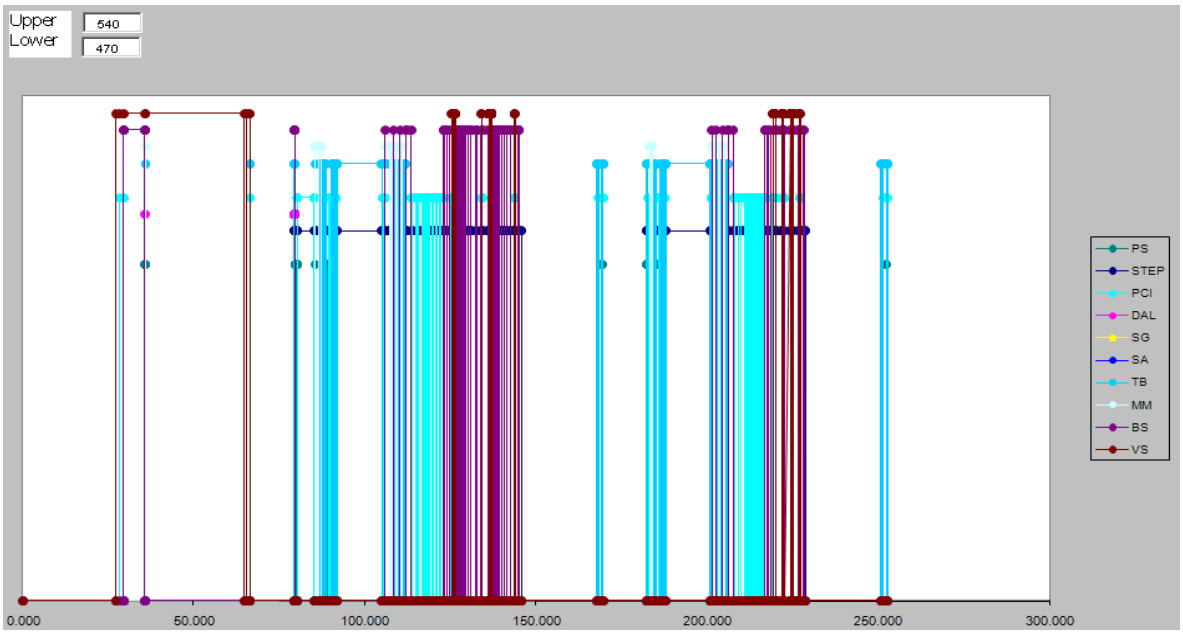


Figura 24: Utilização dos instrumentos (*Excel*)
Fonte: Autor

Para ajudar com as análises de identificar potenciais oportunidades de redução de tempos, foi incorporada essa função à ferramenta (F).

Nela foi desenvolvida diversas funcionalidades de gráfico (*zoom*, arrastar, etc.), marcar pontos no gráfico e exportar o gráfico como figura. Mais tarde ainda foi desenvolvida uma opção de comparação entre dois gráficos e a de se o usuário tivesse o código a ser otimizado e a linguagem de programação instaladas em seu computador, o Time Stamp Tool abriria o código com o arquivo desejado aberto e na linha correspondente à chamada de instrumentos (F).

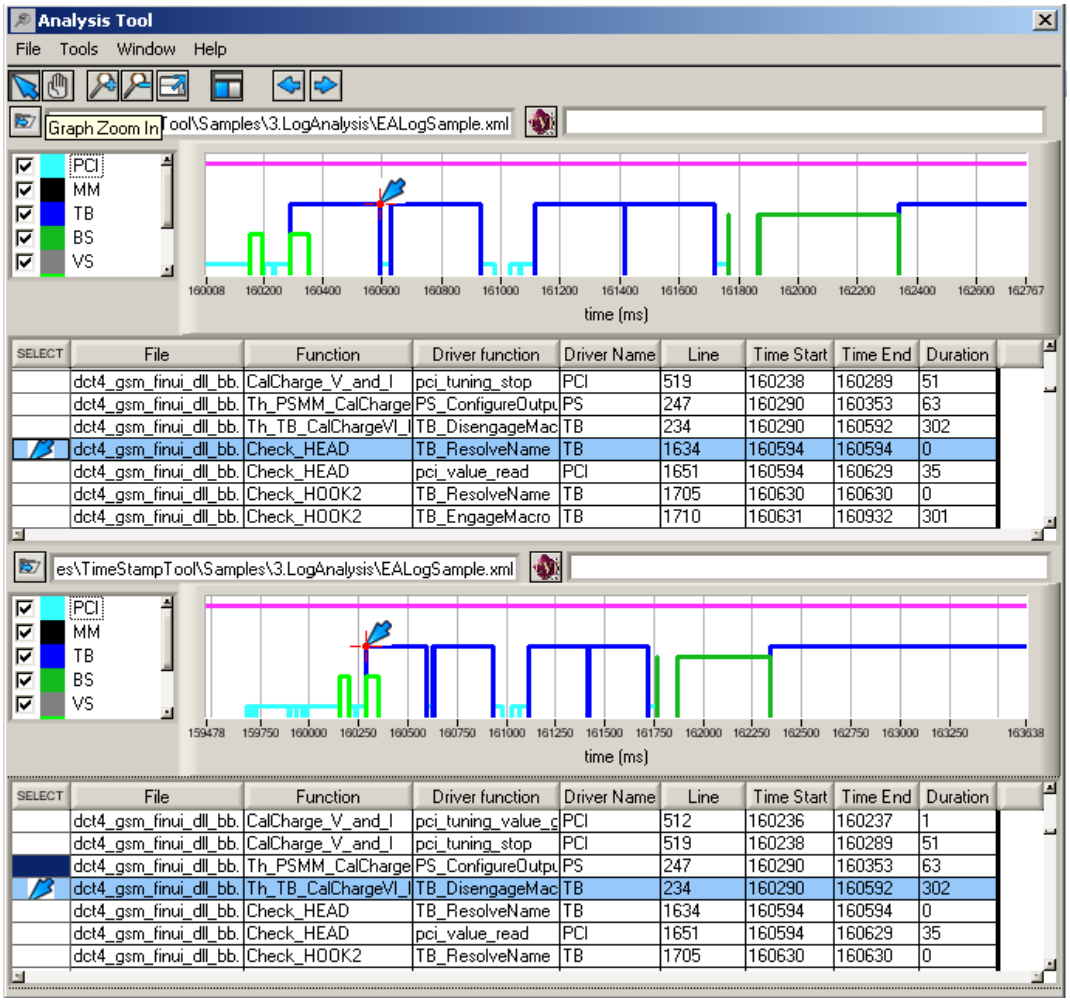


Figura 25: *Time Stamp Tool* (Utilização dos Instrumentos)
Fonte: Autor

3.4.2 Test Plan Optimizer

O processo de otimização de tempo de testes se mostrou eficaz, no entanto para poder implantar o paralelismo nos testes indicados é necessário um gasto de tempo para descobrir os testes mais relevantes e após essa análise, investigar os possíveis pontos de otimização dentro e entre os passos de teste.

De maneira a tornar mais eficiente esse processo, foi desenvolvida uma ferramenta, o *Test Plan Optimizer* (TPO) que tem por objetivo implementar o

paralelismo entre os passos de testes e gerar uma sequência otimizada baseada em um conjunto de regras.

A ideia do TPO é mudar o conceito de sequência de testes. Uma sequência por definição é um conjunto de elementos em uma ordem específica. A ferramenta (*Test Stand*) que a empresa estudada usa é uma ferramenta que trabalha com o conceito de sequência de testes, porém permite que testes (subsequências) sejam executadas em paralelo.

Com a ajuda do TPO é possível transformar uma sequência de testes em um plano de testes, ou seja, tirar a obrigatoriedade de uma ordem específica. Um plano de testes pode ser entendido como um conjunto de testes que deve ser executado obedecendo a certas condições, porém não necessariamente uma ordem fixa. O TPO atualmente está em processo de submissão para o grupo de patentes da empresa (*Patent Board*).

De modo a permitir a otimização, é necessária uma mudança na maneira de se desenvolver as sequencias de testes dos dispositivos. Ao invés de definir as ordens de todos os passos de testes (sequência de testes) é necessário criar regras que especifiquem quais testes não podem ser executados em paralelo e quais testes dependem de outros.

Baseada nessas duas condições (testes mutualmente exclusivos e testes que tenham outros testes como pré-requisitos) e mais a disponibilidade dos instrumentos é possível ser gerada uma sequência otimizada.

O TPO permite que o desenvolvedor de testes cadastre os instrumentos disponíveis no equipamento de testes (recursos críticos) e dentre desses, os instrumentos que serão usados no plano de testes específico, conforme apresentado na Fonte.

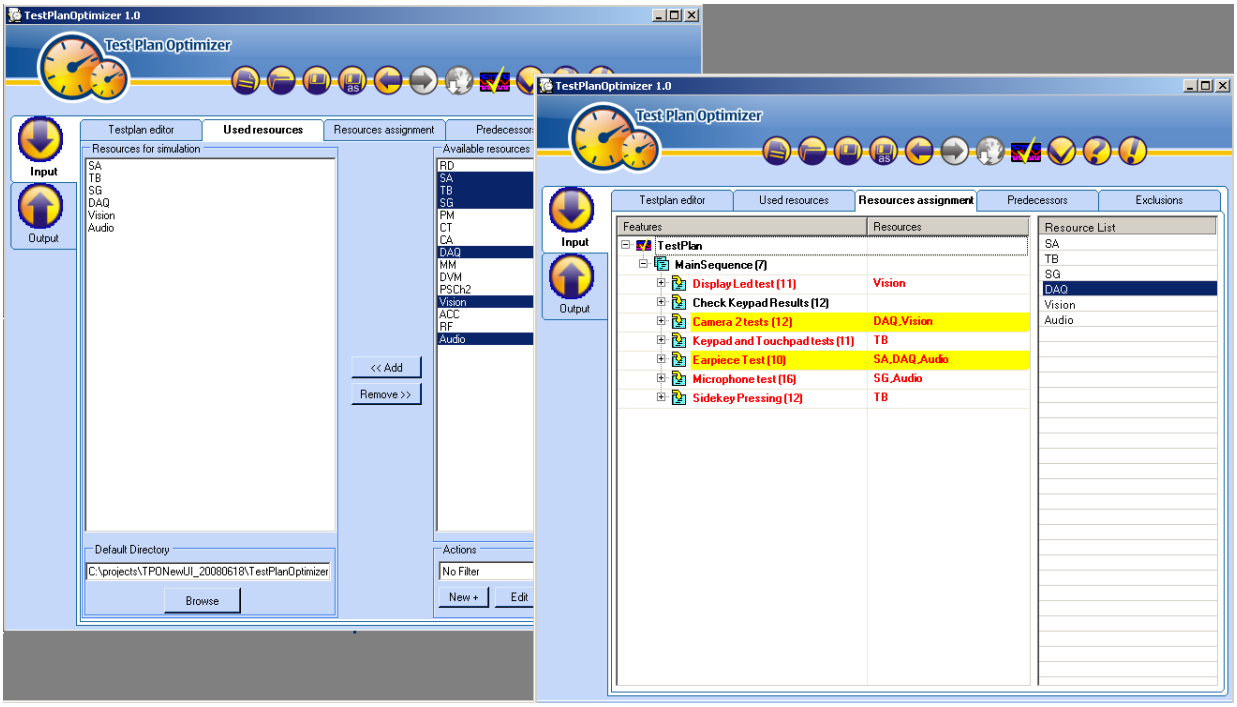


Figura 26: Tela de cadastro de equipamentos
Fonte: Autor

O próximo passo é o cadastro de cada subsequência de testes. Essas subsequências que serão executadas em paralelo. É necessário que o usuário cadastre as subsequências de testes juntamente com o tempo que cada subsequência demore a executar. O TPO comunica com um sistema de dados da empresa, permitindo assim importar a média dos tempos que cada subsequência leva para ser executada. A **Error! Reference source not found.** mostra um exemplo de sequência de testes com os tempos de cada subsequência já cadastrados.

Figura 27: Tela de edição do plano de testes
Fonte: Autor

Ainda nas subsequências, o usuário deve informar quais são os recursos críticos necessários para executar a subsequência.

Após o cadastro de tempo e utilização dos instrumentos o desenvolvedor de teste terá que indicar quais testes que tem outros testes como predecessores (dependência) e quais testes não podem ser executados em paralelo (mutualmente exclusivos).

Com base nessas informações o *Test Plan Optimizer* simulará todas as combinações possíveis e irá gerar o plano de testes (em formato XML) que teoricamente consumirá menos tempo para ser executada. No exemplo, a figura 23

mostra um tempo total de 12753 milissegundos, uma redução de 44% em relação aos quase 23 segundos da Fonte.

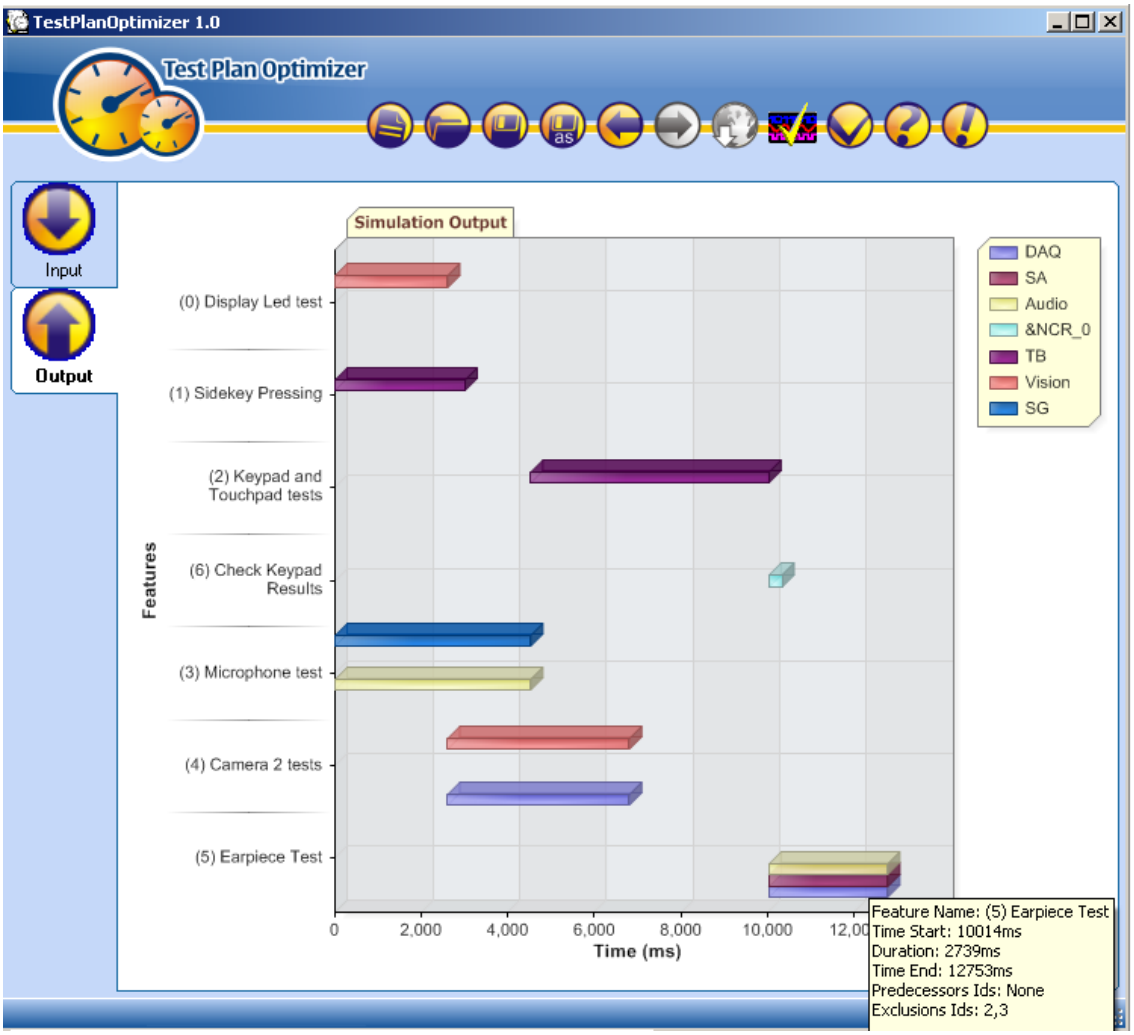


Figura 28: Tela de visualização gráfica da sequência otimizada
Fonte: Autor

O TPO ainda pode exportar seu plano de testes para o *Test Stand* invocando todas as subsequências em paralelo e usando os mecanismos de sincronização do próprio Test Stand para que as restrições indicadas pelo desenvolvedor de teste sejam respeitadas.

Além disso, o TPO ainda gera na sequência exportada passos que fazem com que os testes possam ser reportados na ordem correta, ou seja, na ordem

esperada pelo banco de dados de testes e não na ordem (não fixa) que os passos são executados.

Testes preliminares foram feitos com o TPO e seus ganhos foram bem expressivos. Em algumas simulações alguns testes tiveram seu tempo reduzido em cerca de 50%. Entretanto não há comparações reais entre sequencias originais e otimizadas com o TPO.

4 CONCLUSÕES E RECOMENDAÇÕES

A definição e criação da *baseline* foi de fundamental importância no trabalho pois através dela foi possível a identificação de pontos chaves de melhorias nos testes originais, além de servir de *benchmark* dos testes.

A *baseline* deve conter todas as informações sobre as condições em que foram feitos os testes, para que nenhuma variável externa possa aparecer durante a comparação dos testes originais e modificados.

A utilização do princípio de Pareto (80/20) para identificar os pontos candidatos à redução de tempo de testes foi o que permitiu que o trabalho fosse finalizado, porque de outra forma o esforço para se trabalhar em toda a sequência de teste poderia inviabilizar o trabalho, além de não apresentar ganhos mais significativos.

A metodologia que foi proposta provou ser bastante eficaz para identificar tanto os pontos de aplicação de chamadas paralelas aos instrumentos de teste quanto para os testes que puderam ser executados concorrentemente com resultados de mais de 20% de redução nos seus respectivos tempos.

A aplicação da técnica de programação com múltiplas *threads* nas fases de teste de calibração e funcionais, permitiu que fosse implementado paralelismo dentro dos testes como entre os testes. O uso dos sincronizadores (como por exemplo, o *mutex*) também funcionou para os dois casos e permitiu que as dependências entre os testes não fossem quebradas, e que os recursos críticos (instrumentos de teste) fossem utilizados corretamente.

Comparando os tempos dos testes foi possível medir que em média 20% do tempo dos testes foi salvo para as duas fases de teste. Foi ainda possível perceber

que no caso dos testes de calibração o ganho com o paralelismo entre e dentro dos testes foi semelhante, enquanto na fase de testes funcionais quase a totalidade do ganho foi obtido executando passos de teste concorrentemente. Isso se deve principalmente ao modo em que os instrumentos de testes são utilizados em cada fase e ainda às dependências dos testes.

Uma análise crítica dos resultados dos testes foi feita, levando em consideração a média, desvio padrão, limites de controle e o cálculo do índice Cpk. Essa análise permitiu mostrar que os testes em ambas as fases ficaram mais capazes e, com isso, menos suscetíveis a variações do processo.

A redução de 20% nos tempos dos testes permitiu que testadores caros fossem retirados da linha de produção da empresa estudada e remanejados para serem usados em outros produtos.

Os testes modificados foram usados nas filiais da empresa estudada que fabricavam o mesmo produto, assim os ganhos com redução de equipamentos, mão de obra, calibração e espaço físico não ficaram restritos apenas a filial estudada.

A metodologia de redução de teve tanta importância que vários produtos fabricados pela empresa em diferentes filiais do mundo passaram a aplicá-la posteriormente.

As ferramentas produzidas ao longo do projeto (*Time Stamp Tool* e *Test Plan Optmizer*) foram também utilizadas por centros de pesquisa da empresa estudada.

Como próximos trabalhos, é possível que seja pesquisado e/ou desenvolvido um escalonador de testes em tempo real que possa executar os teste não de acordo com uma sequência de testes, mas baseado na utilização do instrumento e nas dependências entre os testes.

Evidente que para a criação ou uma melhor utilização desse escalonamento em tempo real, modificações maiores na arquitetura de testes, bem como uma melhor utilização dos instrumentos de testes precisa também ser realizadas.

Os sistemas de testes automáticos são há alguns anos partes integrantes do processo de produção de produtos eletrônicos e como todo processo ele deve ser constantemente analisado e melhorado.

REFERÊNCIAS

Davis, M., Chase, R.; Aquilano, N. (1999). FUNDAMENTOS DA ADMINISTRAÇAO DA PRODUÇAO. Nova York: McGraw-Hill.

Ohno, T. (1988). TOYOTA PRODUCTION SYSTEM, BEYOND LARGE-SCALE PRODUCTION, New York: Productivity.

Shingo, S. (1996). SISTEMA TOYOTA DE PRODUÇAO, O: DO PONTO DE VISTA DA ENGENHARIA DE PRODUÇAO. Porto Alegre: Artmed.

Santos, A.; Urbina, L. (2002). INOVAÇÃO NA LINHA DE PRODUÇÃO DA EMBRAER: A MONTAGEM EM DOCA. VIII ENCITA - Encontro de Iniciação Científica e de Pós-Graduação do ITA. São José dos Campos: Anais da VIII ENCITA.

Ramalho, G.; Medeiros, F. (2003). UM SISTEMA DE INSPEÇÃO VISUAL AUTOMÁTICA APLICADO À CLASSIFICAÇÃO E SELEÇÃO DE LARANJAS. IV Congresso Brasileiro Da Sociedade Brasileira De Informática Aplicada À Agropecuária E Agroindústria (Sbi-Agro) (pp. 197 - 200). Porto Seguro: Sbi-Agro.
ATKINSON, A. (1998). Contabilidade Gerencia. São Paulo: Atlas.

BOCTOR, G. (2005). A nonmultiplexed tester with integrated functional-test resources unblocks end-of-line production bottlenecks. Digitaltest - Test & Measurement World .

BOGAN, C. E.; ENGLISH, M. J. (1997). Benchmarking, aplicações práticas e melhoria contínua. São Paulo: Makron Books.

Brinch, H. (2001). Classic Operating System: from batch processing to distributed system. Nova York: Springer-Verlag.

Carvalho, M. M.; Paladini, E. P. (2005). Gestão da Qualidade: Teoria e Caos. São Paulo: Ed. Campus.

Corrêa, H. L.; Corrêa, C. A. (2004). Administração de Produção e Operações. São Paulo: Atlas.
Costas, J. S. (1992). Statistical process control in semiconductor manufacturing. Proceedings of IEEE (pp. 819-830). IEEE CNF.

Ferraz, L.; Dacol, S. (03 a 05 de Novembro de 2009). Redução do custo dos testes de manufatura utilizando testes. XVI Congresso Brasileiro de Custos , p. 87.

Kielty, T.; Delahunty, J. (1990). Automated Pareto Analysis For Continuously Improving a VLSI Fabrication Area's Process Stability. IEEVSEMI Advanced Semiconductor Manufacturing Conference (pp. 113-116). IEEE CNF.

MARINHO, S., BRAZILEIRO, R.; BRUNI, A. L. (2003). Custos de Falhas Externas: Um Estudo de Caso de uma Empresa Brasileira. X Congresso Brasileiro de Custos .

MOLAVI, S.; MCPHEETERS, T. (08-11 de Outubro de 2007). Concurrent Test Implementations. Asian Test Symposium , p. 214.

National Instruments. (2006, Setembro 06). NI Developer Zone. Retrieved Novembro 11, 2009, from Nationa Instrument: http://zone.ni.com/devzone/cda/tut/p/id/3757

NEELY, A., Gregory, M.; Platts, K. (1995). Performance measurement system design: A literature review and research agenda. International Journal of Production Economics , 80-116.

NIST/SEMATECH. (18 de 07 de 2006). e-Handbook of Statistical Methods. Estados Unidos da América.

O'Mara, C. E., Hyland, P. W.; Chapman, R. L. (1998). Performance measurement and strategic change. Managing Service Quality , 178-182.
Ohno, T. (1988). Toyota Production System, Beyond Large-Scale Production. Nova York: Productivity Press.

Qian-Li Zhang, J. G. (2004). APPLYING SPC TO AUTONOMIC COMPUTING. Proceedings of the Third International Conference on Machine Learning and Cybernetics (pp. 744-749). Xangai: IEEE CNF.

Rivoir, J. (14-16 de Julho de 2004). Parallel Test Reduces Cost of Test More Effectively Than Just a Cheap Tester. Electronics Manufacturing Technology Symposium , pp. 263-272.

ROBLES JR, A. (2003). Custos da Qualidade: aspectos econômicos da gestão da qualidade e da gestão ambiental. São Paulo: Atlas.

Silberschatz, A., Galvin, P.; Gagne, G. (1999). Operating System Concepts. Nova York: John Wiley & Sons.

Stallings, W. (2004). Operating Systems: Internals and Design Principles. Nova York: Prentice Hall.

Tanenbaum, A. S. (2006). Operating Systems Design and Implementation. Massachusetts: Prentice Hall.

Walker, W. (31 a 07 de Outubro e Novembro de 1998). Why a Quality System and Spc? Digital Avionics Systems Conference , pp. 1-6.

Weiss, D. H. (20-22 de Setembro de 1994). Achieving Tester Independence. AUTOTESTCON '94. IEEE Systems Readiness Technology Conference , pp. 321 - 327.

Puc-RS. (18 de Dezembro de 2009). História da Estatística. Acesso em 18 de Dezembro de 2009, disponível em http://www.pucrs.br/famat/statweb/historia/ daestatistica/biografias/Shewhart.htm

Wilson, L. (2009). How to Implement Lean Manufacturing. Nova York: McGraw-Hill.

Xia, R., Xiao, M.-Q.; Cheng, J.-J. (17-20 de Setembro de 2007). PARALLEL TPS DESIGN AND APPLICATION BASED ON SOFTWARE ARCHITECTURE, COMPONENTS AND PATTERNS. Autotestcon , pp. 234 - 240.

Yu-Chi, H. (1994). Heuristics, Rules of Thumb, and the 80/20 Proposition. IEEE JNL , 1025-1027.

Spínola, M.M.; Pessoa, M.S.P.; Tonini, A.C. *The Cp and Cpk Indexes in Software Development Resource Relocation*, 2007 PICMET Proceeding