TestView: análise multidisciplinar sobre um projeto de automação industrial

Rodolfo Boffo de Souza
Orientador: Prof. Dr. Marco Dimas Gubitoso

29 de novembro de 2012

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao caro professor orientador Marco Dimas Gubitoso por todo o apoio dedicado à esta monografia. Em seguida agradeço a todos os meus familiares e amigos por me apoiarem em todas as decisões tomadas, e por último, mas não menos importante, agradeço à todas as pessoas que fizeram parte da minha vida profissional durante o meu estágio supervisionado na *Dinateste*, em especial ao meu amigo Ricardo Zanetti.

Sumário

1 Introdução

Durante a graduação, com frequência desenvolvemos pequenos projetos para aprimorar nosso raciocínio lógico, além do objetivo de conhecer novas tecnologias e aprender novos conceitos e diversas estruturas de dados. Em contrapartida, a maioria destes projetos possui o domínio de aplicação bastante reduzido, já que o mesmo deve abranger somente um semestre do ano letivo e também dividir tempo com projetos de outras disciplinas.

Por esses motivos temos pouco contato com clientes reais e poucas possibilidades de se desenvolver uma aplicação (comercial ou não) com abrangência multidisciplinar, ou seja, um projeto onde possamos encontrar um ambiente mais próximo possível do mercado de trabalho, e que possamos exercitar simultâneamente o conhecimento adquirido em várias disciplinas de nosso contato no curso de computação.

Neste trabalho vamos abordar o desenvolvimento de um sistema de automação insdustrial, projeto que se originou em um estágio supervisionado durante o período da graduação, e que foi escolhido por abranger diversos tópicos vistos durante o curso, como programação orientada a objetos, banco de dados, métodos númericos, entre outros.

1.1 Motivação

Este projeto teve início com o desafio de se criar e comercializar um *software* capaz de obter dados de uma máquina universal de ensaios, analisá-los e gerar relatórios para ensaios de tração em corpos de prova de materiais metálicos.

O desenvolvimento deste primeiro produto se deu em meados de 2007, e levou cerca de 6 meses para obtermos uma versão estável que pudesse ser comercializada. Seu escopo inicial abrangia apenas a coleta de dados dos conversores analógico-digitais ligados à máquina universal de ensaios, a exibição de um gráfico da tensão exercida sobre o corpo de prova em função de sua deformação, seguida de uma análise para obtenção de informações sobre as características dos materiais metálicos, e por fim a possibilidade de exportar tais dados para uma planilha ou então a preparação de um relatório de ensaio contendo as informações mais importantes do ensaio.

Com o sucesso do desenvolvimento e comercialização deste produto, houve a posterior demanda de desenvolvimento de um novo *software*, em sua maior parte semelhante ao primeiro, porém com o objetivo de coletar dados de uma máquina para ensaios de compresssão, analisá-los e gerar relatórios para ensaios de compressão em corpos de prova de concreto.

Podemos dizer que os dois *softwares* eram idênticos quanto à coleta de dados, pois os mesmos conversores analógico-digitais que eram usados na

máquina universal de ensaios seriam também usados na coleta de dados do equipamento de ensaios de compressão e também por seus extensômetros. Em contrapartida, os ensaios eram bastante diferentes em diversos pontos: enquanto o ensaio de tração utilizava um único extensômetro, o ensaio de compressão utilizaria três; enquanto a coleta de dados do ensaio de tração era temporizada e rápida, a do ensaio de compressão era lenta e deveria coletar um único ponto a cada comando do operador; e por fim os gráficos, as análises pós-ensaio e os relatórios conseqüentemente seriam também diferentes.

Decidimos então reaproveitar o código responsável pela comunicação serial e coleta de dados, porém devido a escassez de tempo disponível para a entrega do novo produto, optou-se então pela duplicação do código do primeiro produto, seguida de adaptações para este novo tipo de ensaio.

Em um primeiro momento, isso nos trouxe grandes vantagens, pois o novo produto pode ser desenvolvido em menos tempo quando comparado com o tempo de desenvolvimento da aplicação de ensaios de tração. Porém, no decorrer de alguns meses, através das demandas de clientes por melhorias e adaptações em ambos os produtos, notamos o crescimento do tempo necessário e da complexidade para se realizar a manutenção do código, já que cada alteração na parte comum deveria ser feita no código-fonte das duas aplicações.

Isso nos trouxe a necessidade de refatorar o código dos dois produtos, lançando mão de melhores práticas de desenvolvimento, reutilização de código e padrões de projeto.

1.2 Objetivos

Tem-se como objetivo prático deste trabalho a refatoração do código de dois produtos distintos, visando o desenvolvimento de um arcabouço de ensaios capaz de centralizar e disponibilizar funções comuns como gerenciamento de equipamentos, calibração dos instrumentos de medição, funções de acesso ao banco de dados, criação e impressão de relatórios e funções de armazenamento e recuperação de ensaios.

Uma característica alvo dessa refatoração é a criação do arcabouço utilizando uma arquitetura de *plugins*, ou seja, cada módulo de ensaio deveria ser compilado em uma única biblioteca capaz de ser detectada pelo arcabouço, tornando assim disponíveis os recursos necessários às operações relacionadas a realização e análise do ensaio específico.

O objetivo teórico deste trabalho é a discussão dos conceitos e tecnologias empregadas no desenvolvimento da solução acima descrita. Estes conceitos serão tratados ao longo de toda a monografia.

2 Conceitos teóricos

Nesta seção serão brevemente introduzidos deversos tópicos teóricos estudados durante o desenvolvimento do projeto.

2.1 Padrões de projeto

Em [1], podemos entender o conceito de padrão de projeto simplesmente como a descrição de uma solução para um problema que ocorre repetidas vezes em um ambiente.

Um padrão de projeto é definido descrevendo-se um problema e uma solução para tal problema. O problema especifica o contexto onde o padrão de projeto pode ser utilizado, e sua solução descreve de maneira abstrata o uso de um modelo que pode ser aplicado para sanar o problema.

2.1.1 Arcabouço

Segundo [1], podemos definir um arcabouço, também conhecido como *framework*, como um conjunto de classes que cooperam entre si com a finalidade de formar um design reutilizável para um escopo específico de *software*.

Arcabouços são usados com o objetivo de facilitar o desenvolvimento de aplicações, pois seu design predefine a estrutura global da aplicação, bem como as responsabilidades das classes, como as classes e objetos interagem e o fluxo de controle da aplicação, deixando para o desenvolvedor apenas o trabalho relacionado às epsecificidades de sua aplicação. Dessa forma, pode-se dizer que arcabouços enfatizam a reutilização de design em comparação com a reutilização de código.

Ainda segundo [1], esse tipo de reutilização leva a uma inversão de controle entre sua aplicação e o arcabouço. Em geral quando usamos uma biblioteca convencional de subrotinas, escrevemos o corpo principal da aplicação e chamamos a rotina que desejamos reutilizar. Por sua vez, quando utilizamos um arcabouço, reutilizamos seu corpo principal e implementamos apenas o código que é chamado por ele. Para tanto devemos escrever o código seguindo certas convenções de nomenclatura sobre as rotinas. Isso torna o desenvolvimento das aplicações mais simples e rápido, e as aplicações resultantes tendem a ter estruturas similares, resultando em um código de fácil manutenção.

O desenvolvimento de arcabouços é relativamente difícil pois temos que assumir que todas as aplicações dentro de um escopo definido serão atendidas pela arquitetura planejada, e qualquer alteração em sua interface resultará

em alterações no código das aplicações específicas. Portanto é de extrema importância que o arcabouço seja tão flexível e extensível quanto possível.

Um arcabouço pode conter outros diversos exemplos de padrões de projeto dentro de sua implementação. Podemos citar por exemplo a necessidade do uso do padrão de projeto Fábrica por arcabouços que manipulam diversos tipos de arquivo, assim como o Testview.

O uso de Fábricas, ou *factory methods* se faz necessário quando a aplicação desconhece em tempo de compilação qual classe deve instanciar. Para isso, um método fica responsável por decidir de qual classe deverá ser criada uma instância. Podemos exemplificar melhor esse cenário, utilizando o seguinte diagrama de sequência 2.1.1

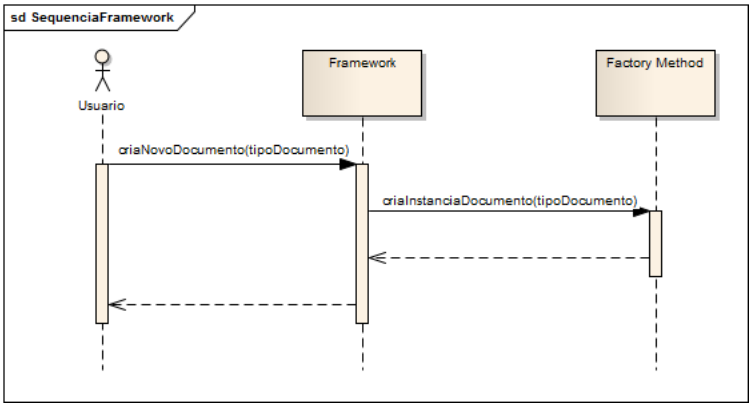

Figura 1: Diagrama de sequência esquemático de um *factory method*.

O usuário solicita ao arcabouço a criação de um novo arquivo; o arcabouço solicita ao método fábrica a criação de um novo arquivo, passando como argumento o tipo de arquivo solicitado; o método fábrica, por sua vez, cria uma nova instância da classe correspondente ao argumento recebido.

De acordo com [1], podemos dizer que Fábricas encapsulam o conhecimento para instanciar classes.

3 Atividades realizadas

Nesta seção serão descritas detalhes da implementação do arcabouço, organização de classes, modelo de dados, entre outras decisões importantes que foram tomadas ao longo do projeto.

3.1 O Arcabouço: TestView

3.1.1 Ambiente de desenvolvimento

Segundo [2], ainda no ano de 2012, mais de 80% dos computadores conectados à rede mundial possuíam alguma versão do Microsoft Windows como sistema operacional. Devido a esse fato, desde o princípio tínhamos o objetivo de desenvolver nossa solução com foco neste sistema operacional.

A interface gráfica do arcabouço deveria ser simples e de fácil compreensão, pois seria usada, em geral, dentro de indústrias. Levando em consideração que a elaboração da interface gráfica tomaria boa parcela de tempo do trabalho como um todo, deveríamos escolher um ambiente de desenvolvimento que facilitasse esta etapa.

De acordo com [5], atualmente C# é uma das linguagens de programação mais utilizadas e sua sintaxe é semelhante à linguagens como Java, C ou C++.

O conjunto desses fatores nos levou a optar pela linguagem de programação C#, utilizando o Microsoft Visual Studio 2010 como ambiente de desenvolvimento. Entre outras características, o Microsoft Visual Studio possui um editor de interface gráfica aponte-e-clique, característica que facilita o desenvolvimento de interfaces gráficas.

3.1.2 Organização de classes

Nessa seção serão brevemente descritas as responsabilidades das principais classes que compõem a aplicação principal.

Condicionador de sinal A classe CondicionadorDeSinal representa um componente eletrônico, responsável por transformar sinais analógicos em digitais. Este conversor analógico-digital necessita ser calibrado juntamente com o instrumento de aferição que será usado em conjunto durante a realização do ensaio, portanto um objeto dessa classe é composto, entre outros objetos, por um objeto do tipo Calibração.

Outro objeto que compõe a classe CondicionadorDeSinal é do tipo Configuração, este responsável por armazenar algumas propriedades do conversor analógico-digital não relacionados à sua calibração.

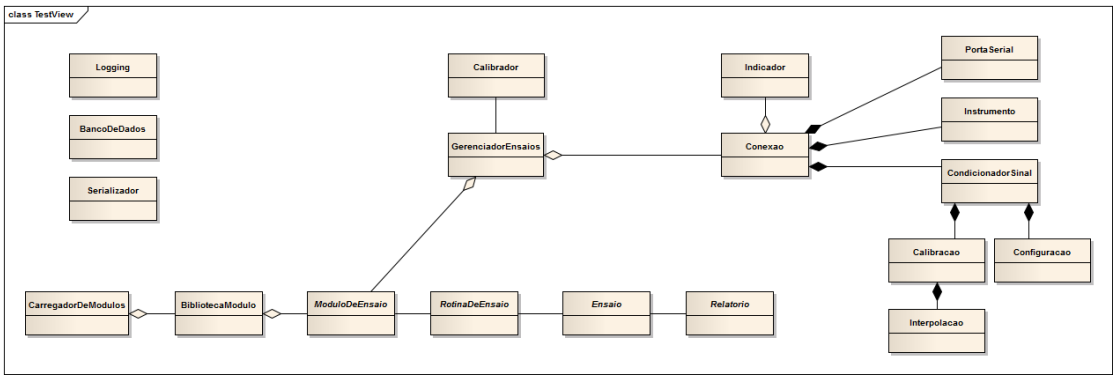

Figura 2: Principais classes do arcabouço.

Calibracao Essa classe contém detalhes da calibração de um condicionador de sinal, como por exemplo as informações de unidade de calibração, capacidade nominal, bem como sua curva de calibração, ou seja, seu conjunto de pares ordenados que relacionam a saída do conversor digital à grandeza aferida em determinado momento da calibração.

Conexao A classe Conexao concentra o conhecimento necessário para se obter dados dos conversores analógico-digitais conectados ao computador. Objetos dessa classe são formados por objetos das classes Instrumento, PortaSerial e CondicionadorDeSinal.

Informações sobre o instrumento utilizado no momento do ensaio podem ser obtidos consultando seu objeto do tipo Instrumento.

Instrumento A classe instrumento apenas mantém informações detalhadas sobre o tipo de instrumento utilizado no momento do ensaio, não sendo necessariamente igual ao instrumento utilizado no momento da calibração do condicionador de sinal.

Indicador A classe Indicador implementa um indicador digital temporizado. Quando associado a uma Conexão e iniciado, a cada intervalo de tempo ele realiza uma leitura do conversor analógico-digital através da interface disponibilizada pela conexão, transformando a saída coletada de acordo com a calibração do condicionador de sinal.

GerenciadorDeEnsaios O GerenciadorDeEnsaios é a classe central do arcabouço. Entre outras funções, de acordo com seu nome, ela é responsável pelo gerenciamento de ensaios. Isto abrange a criação de novos ensaios, responsável pelo controle de início e término de ensaios

e também pelo carregamento e armazenamento de ensaios. Além disso, através do GerenciadorDeEnsaios é possível adicionar e gerenciar as conexões com os condicionadores de sinal existentes e também configurar a conexão com o banco de dados.

CarregadorDeModulos Esta classe tem o papel de verificar a existência de bibliotecas de módulos de ensaio instalados e tornar as bibliotecas disponíveis para o arcabouço em tempo de execução.

BibliotecaModulo Esta classe contém informacões sobre as bibliotecas de ensaios que foram carregadas pelo CarregadorDeModulos. Contém os nomes das classes concretas que estendem as classes abstratas necessárias para a realização do ensaio específico. Ela é também responsável por criar as instâncias dos ensaios específicos através de suas Fábricas.

ModuloDeEnsaio O ModuloDeEnsaio é uma classe abstrata que define a interface necessária para que o arcabouço possa obter os elementos específicos do ensaio, como por exemplo sua interface gráfica e seus métodos de controle de ensaio.

RotinaDeEnsaio A RotinDeEnsaio é a classe abstrata que define a interface para o controle do ensaio. Implementações concretas dessa classe abstrata devem conter toda a lógica de ensaio, sendo assim responsáveis por coletar os dados das conexões do arcabouço e por quaisquer tarefas de análise, cálculos e também por armazenar todos esses dados no seu respectivo objeto de Ensaio.

Ensaio A classe abstrata Ensaio define as interfaces necessárias para serialização e deserialização do arquivo de ensaio. As suas implementações concretas têm o objetivo de armazenar todos os dados coletados e calculados no momento do ensaio para que possam ser armazenados e recuperados posteriormente.

Relatorio Classe abstrata que define métodos para a geração e impressão dos relatórios de ensaio.

Serializador Classe utilitária usada para serializar e deserializar objetos.

3.1.3 Extensões

Também conhecidas como *plug-in* ou *add-in*, as extensões são componentes de *software* que adicionam habilidades específicas à uma aplicação maior. Esses componentes possuem uma interface comum que possibilita à aplicação

detectar e registrar a extensão em questão. Por sua vez, extensões usualmente não podem funcionar por si mesmas e portanto necessitam de uma interface de serviços disponibilizada pela aplicação.

Para que a instalação de cada um dos módulos de ensaio fosse simples e rápida, tomou-se como objetivo a elaboração de uma arquitetura de extensões. Cada um dos módulos de ensaio desenvolvidos para o arcabouço *TestView* devem ser compilados em uma única biblioteca e essa biblioteca deve estar localizada no mesmo diretório onde a aplicação principal reside.

Quando iniciada a aplicação, a classe *CarregadorDeModulos* varre o diretório em busca de bibliotecas que contenham ao menos duas classes que implementem as classes abstratas *ModuloDeEnsaio* e *Ensaio*, respectivamente. Essa tarefa é realizada por meio de reflexões.

Denomina-se *Reflexão* a habilidade de uma aplicação de examinar e modificar a estrutura e o comportamento de um objeto em tempo de execução. Neste caso, utilizamos reflexões para listar as classes contidas nas bibliotecas encontradas e verificar quais delas implementam as referidas classes abstratas.

3.1.4 Coleta de dados

O arcabouço foi construído com o propósito inicial de trabalhar com a coleta de dados a partir de um conversor analógico-digital específico, fabricado pela empresa alemã *HBM - Hottinger Baldwin Messtechnik*.

O conversor disponibiliza uma interface serial RS-232 e define um conjunto de comandos que possibilita a comunicação entre ele e o computador.

Por sua vez, o arcabouço disponibiliza uma interface de gerenciamento de conexões, onde o usuário pode definir parametros para efetuar a conexão com um ou mais conversores digitais. É necessário que o arcabouço conheça a velocidade de conexão e o nome da porta com a qual deve tentar efetuar uma conexão.

Esse tipo específico de conversor digital possui muitas características interessantes que podem ser alteradas através de comandos enviados pelo computador, como por exemplo a frequência de seu filtro digital, porém vamos nos concentrar em apenas descrever detalhes da maneira de como é realizada a coleta de dados.

Quando uma coleta de dados é necessária, o método *getIndicacao* da classe *Conexao* é chamado. Este método é responsável por enviar o comando *"MSV?1;"* para a porta serial e, logo em seguida, aguardar o recebimento da resposta.

O comando *"MSV"* pode ser interpretado como um mnemônico para *measured value*, e *"?1"* significa a solicitação de apenas um valor obtido

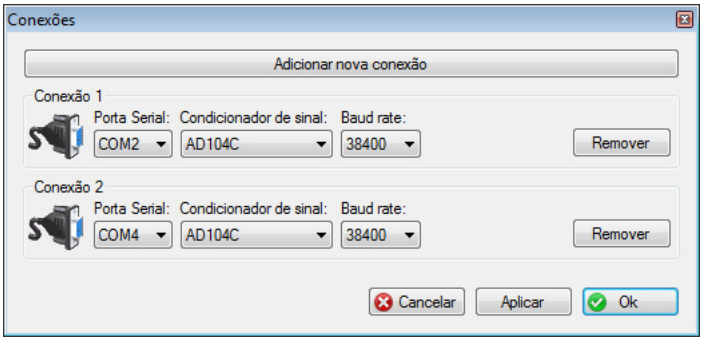

Figura 3: Janela de gerenciamento de conexões do TestView

e, por fim, o caractere *";"* (ponto e vírgula) indica o final do comando ao conversor.

Este comando faz com que o conversor analógico-digital envie como resposta uma *string* de dez caracteres numéricos correspondentes ao valor mensurado no momento da solicitação.

3.1.5 Armazenamento e recuperação de ensaios

O arcabouço está pronto para armazenar e recuperar ensaios, fazendo uso da classe utilitária *Serializador*. Esta classe é responsável por serializar e deserializar objetos marcadas com o atributo *DataContract*. O *Data Contract Serializer* é um motor de serialização disponível no *.NET Framework*, capaz de transformar dados em XML e vice-versa.

Quando solicitado a exportar um ensaio realizado por um determinado módulo de ensaios, o arcabouço chama o método abstrato *serializaEnsaio* da classe *Ensaio*, o qual deve ser sobreescrito pela implementação específica do módulo de ensaio. O método *serializaEnsaio* por sua vez, faz uso do método *serializaObjeto* da classe *Serializador*, passando como um dos argumentos a instância específica do ensaio a ser salvo em disco.

A recuperação de um arquivo XML é realizada de forma análoga: quando solicitado a recuperar um ensaio, o arcabouço percorre uma estrutura contendo todos os módulos de ensaio carregados, chamando para cada um deles o método *deserializaEnsaio*. Por simplicidade, quando o método falha na deserialização sabemos que o arquivo não corresponde a esse módulo de ensaios e passamos ao próximo módulo, até que um deles consiga deserializar o arquivo de ensaio ou até que todos os módulos sejam consultados.

Quando carregado com sucesso, o arcabouço adiciona o ensaio ao conjunto de ensaios abertos contido na classe *GerenciadorDeEnsaios*, disponibilizando

assim as funções de visualização do ensaio e impressão de relatórios.

É necessário que o arcabouço delegue o trabalho de serialização e deserialização para os módulos de ensaio, já que ele desconhece como serializar ou deserializar um ensaio específico.

3.1.6 Modelo de dados

O modelo de dados implantado pelo arcabouço é um modelo simples e minimalista, suficiente para armazenar os ensaios realizados pelo operador.

Por exigência e necessidade dos clientes, foi escolhido um gerenciador de banco de dados relacional amplamente conhecido na plataforma *Microsoft Windows*, o *Microsoft SQL Server*.

O modelo consiste em apenas duas entidades, sendo elas a entidade *Ensaio* e *ModuloDeEnsaio*, que por sua vez representam respectivamente as abstrações de ensaio e as extensões do arcabouço denominadas módulos de ensaio.

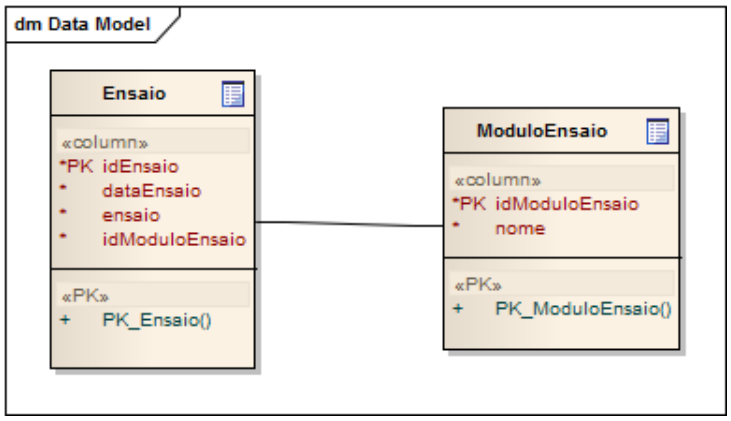

Figura 4: Modelo entidade-relacionamento representando a implementação do modelo de dados do TestView

Como um módulo de ensaio pode ser relacionado a muitos ensaios, existe um relacionamento entre as duas entidades implementado como chave estrangeira em *Ensaio*, ou seja, uma tupla na tabela *Ensaio* possui um valor que contém a chave da tupla *ModuloDeEnsaio* com a qual mantém relacionamento.

Esse modelo permite a busca de ensaios por módulo e/ou por data de realização.

3.2 Módulo de ensaio de tração

Nesta seção apresentaremos informações sobre a rotina de um ensaio de tração, bem como detalhes relevantes sobre a implementação da extensão do arcabouço responsável pela realização dos ensaios de tração em materiais metálicos.

3.2.1 Ensaio de tração

A aplicação da força em um corpo sólido promove uma deformação do material na direção do esforço e o ensaio de tração consiste em submeter um material a um esforço que tende a esticá-lo ou alongá-lo. Geralmente, o ensaio é realizado num corpo de provas de formas e dimensões padronizadas, para que os resultados obtidos possam ser comparados ou, se necessário, reproduzidos. Este corpo de prova é fixado numa máquina de ensaio que aplica esforços crescentes na sua direção axial, sendo medidas as deformações correspondentes por intermédio de um aparelho especial (o mais comum é o extensômetro). Os esforços ou cargas são medidos na própria máquina de ensaio e o corpo de prova é levado até sua ruptura.

Com esse tipo de ensaio, de acordo com [3], pode-se afirmar que praticamente as deformações promovidas no material são uniformemente distribuídas em todo o seu corpo, pelo menos até ser atingida uma carga máxima próxima do final do ensaio e, por consequência, o ensaio de tração permite medir satisfatoriamente a resistência do material. A uniformidade da deformação permite ainda obter medições precisas dessa deformação em função da tensão aplicada. Essa variação é determinada pelo traçado da curva tensão-deformação.

A uniformidade da deformação termina no momento em que é atingida a carga máxima suportada pelo material, quando começa a aparecer o fenômeno da estricção ou diminuição da secção do corpo de prova. A ruptura sempre se dá na região estrita do material. A ductilidade do material, ou seja, o grau de deformação que o material suporta até o momento de sua fratura, pode ser medido através do percentual de estricção (variação da área da secção transversal do corpo de prova).

A velocidade de ensaio geralmente é dada pelos métodos de ensaios estabelecidas pelas diferentes associações de normas técnicas, porém quando se realiza um ensaio para fins de estudo, essa velocidade pode ser alterada conforme o caso.

Quando um corpo de prova é submetido a um ensaio de tração, pode-se construir um gráfico tensão-deformação através das medidas diretas de carga (ou tensão) e de deformação.

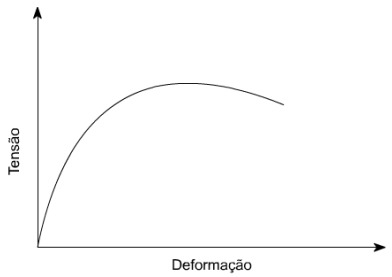

Figura 5: Gráfico tensão-deformação de um metal ou liga metálica

Verifica-se inicialmente que o gráfico é linear e representado pela equação 1 ou 2 (onde σ é a tensão média de tração produzida no corpo de prova e ϵ é a sua deformação linear média) que corresponde a lei de Hooke. A constante E é conhecida como módulo de elasticidade ou módulo de Young.

$$\sigma = E \cdot \epsilon \tag{1}$$

$$E = \frac{\sigma}{\epsilon} \tag{2}$$

Ao ser atingida uma tensão em que o material já não mais obedece à lei de Hooke, chega-se ao ponto determinado limite de proporcionalidade. Terminada a zona elástica atinge-se a zona plástica, onde a tensão e a deformação já não são mais relacionadas por uma simples constante.

O início da plasticidade é verificado em vários materiais pelo fenômeno do escoamento. O escoamento é um tipo de transição caracterizada por um aumento relativamente grande da deformação com variação pequena da tensão durante a sua maior parte. Denomina-se limite de escoamento à tensão atingida durante o escoamento e é dado pela expressão 3, onde Q_e é a carga de escoamento e S_0 é a área da secção transversal do corpo de prova.

$$\sigma_e = \frac{Q_e}{S_0} \tag{3}$$

Terminado o escoamento, o material entra na fase plástica e o ensaio segue até ser atingida uma tensão máxima suportada pelo material, que caracteriza o final da zona plástica.

Após ser atingida a carga máxima, entra-se na fase de ruptura do material, caracterizada pelo fenômeno de estricção, que é a diminuição da secção transversal do corpo de prova numa certa região ao longo do mesmo. É nessa região que acontece a ruptura do corpo de prova, encerrando o ensaio.

3.2.2 Rotinas de ensaio

Antes de iniciarem as rotinas do ensaio de tração, o arcabouço verifica algumas condições obrigatórias para que o ensaio possa ser realizado.

Primeiramente verificamos se existem ao menos duas conexões realizadas pelo arcabouço, sendo uma delas uma conexão a um conversor analógico-digital relacionado a grandeza de força, e uma conexão a outro conversor relacionado a grandeza deformação. Com essas duas conexões ativas, verifica-se também a existência de um extensômetro associado à conexão de deformação.

Após todas as condições descritas serem satisfeitas, o ensaio de tração é iniciado e a classe responsável pela rotina de ensaio fica preparada para receber os eventos relacionados à coleta de novos dados pelo arcabouço.

A cada momento que uma nova coleta de dados é realizada, a rotina de ensaio analisa os pontos coletados para determinar certas características do ensaio, como força máxima, taxa de variação da força aplicada, tava de variação da deformação do corpo de prova, determinação do módulo de elasticidade, tensão de escoamento e, por fim, quando o rompimento do corpo de prova é detectado, o ensaio é finalizado.

Além das características acima descritas, é necessário que a rotina de ensaio alerte o operador no momento em que se torna necessária a remoção do extensômetro envolvido no ensaio. Caso o extensômetro não seja removido em tempo adequado, é possível que ele sofra danos causados pelo rompimento do corpo de prova.

A seguir serão descritas algumas propriedades do ensaio e como são obtidas durante a sua rotina de ensaio.

Força máxima Sendo inicializado com valor zero, a propriedade *ForcaMaxima* da classe *EnsaioTracao* recebe o máximo entre a força máxima atual e o último valor de força coletado pelo arcabouço.

Taxa de variação de força e deformação Para estimar a taxa de variação instantânea da força, o arcabouço agrupa os pontos de força coletados durante os últimos 1,5 segundos, e sobre esse conjunto realiza um ajuste de curva usando um polinômio de grau um, assumindo então a inclinação da reta obtida como taxa de variação instantânea da força aplicada sobre o material. A taxa de variação instantânea da deformação foi estimada de maneira idêntica.

O ajuste polinomial foi implementado utilizando-se o método de mínimos quadrados por fatoração QR (usando triangularização de Householder). Detalhes sobre a teoria envolvida no método e sobre a implementação podem ser encontrados em [4].

Módulo de elasticidade O módulo de elasticidade é determinado também através da realização de um ajuste de curva usando um polinômio de primeiro grau, porém sobre um subconjunto de pontos do gráfico tensão-deformação. Este subconjunto é geralmente compreendido pelos pontos que possuem tensão entre 20% e 40% em relação à tensão máxima do fundo de escala do gráfico tensão-deformação, porém essa faixa pode ser alterada pelo operador do ensaio. À partir da reta obtida pelo ajuste, tomamos a sua inclinação como módulo de elasticidade característico do material.

Detecção de rompimento do corpo de prova A rotina de ensaio de tração assume que o corpo de prova se rompeu quando o valor de força coletado é menor que 75% em relação à força máxima aplicada sobre o material durante o ensaio. Este limiar também pode ser alterado previamente pelo operador, quando necessário.

4 Resultados

Obtivemos como resultado deste projeto uma aplicação principal, um arcabouço de ensaios que concentra e disponibiliza funcionalidades necessárias a realização de diversos ensaios de materiais, como coleta de dados de equipamentos de ensaio através de conversores analógico-digitais, bem como funções relacionadas ao armazenamento e recuperação de ensaios realizados e geração e impressão de relatórios de ensaio.

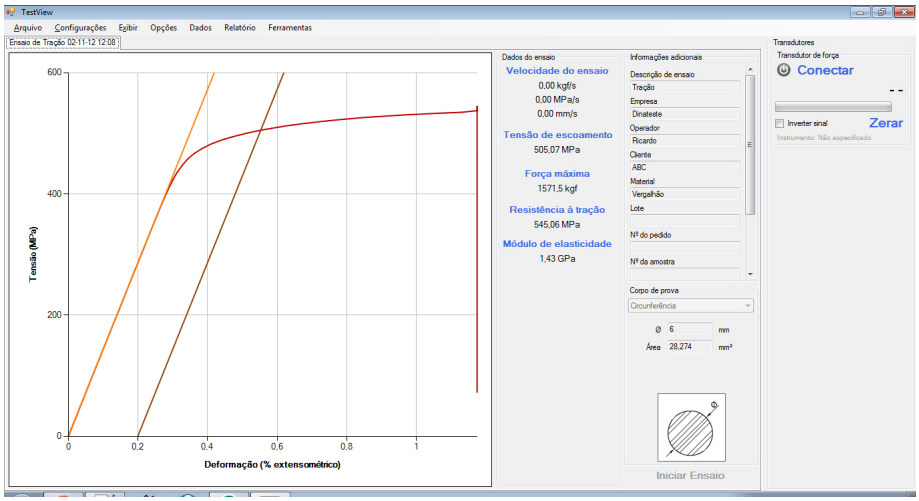

Figura 6: Captura de tela da aplicação durante a exibição de um ensaio de tração previamente realizado.

Também foi desenvolvido o módulo de ensaio de tração em materiais metálicos, capaz de analisar os dados coletados e calcular diversas caractrísticas inerentes ao material ensaiado, como por exemplo o módulo de elasticidade ou módulo de Young, sua tensão de escoamento, bem como características do ensaio, como por exemplo a força máxima aplicada ao material, entre outras.

5 Conclusões

Com o desenvolvimento deste projeto, podemos perceber que padrões de projeto facilitam o desenvolvimento e posterior manutenção de aplicações pois, seu uso agrega soluções largamente conhecidas ao código fonte, aumentando a sua coesão por trazer responsabilidades bem definidas a diversas classes, possibilitando assim uma melhor organização e compreensão do código e suas funcionalidades como um todo.

Em particular, o uso do padrão de projeto Arcabouço fez com que funcionalidades comuns a todos os módulos de ensaio ficassem disponíveis em nossa aplicação principal, tornando possível a diminuição do tempo de manutenção das aplicações através da eliminação de código replicado nas aplicações anteriores ao desenvolvimento deste projeto.

A implementação de uma arquitetura de extensões padronizou a instalação de nossas aplicações em todos os clientes, já que a instalação da aplicação principal é única, e a instalação de todos os módulos de ensaio pode ser realizada apenas copiando a sua respectiva biblioteca para dentro do diretório da aplicação principal.

Por fim, é importante citar a importância da refatoração de código. A refatoração das soluções anteriores visando a criação deste projeto trouxe muitos benefícios com relação a clareza e organização da nova arquitetura, pois a evolução de código fonte sem uma refatoração envolve, muitas vezes, a inserção de lógicas complexas e até mesmo *code hacking* ou manobras para que a nova funcionalidade proposta seja implementada.

6 Trabalhos futuros

Este projeto deixa como trabalho futuro o desenvolvimento de novos módulos de ensaio, como o módulo de ensaio de compressão em concreto, ou até mesmo ensaios não mecânicos, utilizando outros tipos de instrumentos como termômetros, entre outros.

Uma limitação da atual arquitetura do arcabouço está no suporte a apenas um modelo de conversor analógico-digital. Se o uso de outro modelo de conversor for comercialmente interessante, será necessária a evolução da arquitetura do arcabouço, flexibilizando a escolha do conversor a ser utilizado no ensaio.

Uma solicitação não funcional dos clientes sugere uma mudança na interface gráfica, com a inserção de botões com efeitos de transição, personalização das cores da aplicação, entre outras. Com isso, no futuro, temos a intenção de criar um projeto de interface gráfica usando a tecnologia *Windows Presentation Foundation*.

Podemos listar também a necessidade de exportar os relatórios de ensaio em documentos de texto, apontando como um trabalho futuro a implementação do suporte a criação de documentos no formato *Open Document*.

7 Análise subjetiva

Esta seção tem como objetivo relacionar o trabalho realizado com algumas disciplinas mais relevantes sobre o tema escolhido e com o curso de Ciência da Computação de modo geral.

7.1 Desafios encontrados

O desafio mais relevante durante a execução deste projeto, desde sua concepção até o fim desta monografia, foi a tarefa de se conciliar o tempo necessário aos estudos e o tempo necessário à vida profissional. A vida profissional em paralelo com a graduação trazem muitos benefícios como, por exemplo, a experiência no mercado de trabalho e a independência financeira. Porém não existe almoço grátis, e muito provavelmente essa decisão acarretará o alongamento do tempo necessário para se completar o curso.

De modo geral, o planejamento e desenvolvimento deste arcabouço ocorreu sem maiores frustrações pois eu já havia desenvolvido a primeira versão dos *softwares* de ensaio de tração e compressão e, por consequência, já conhecia bem o ponto de partida e o objetivo a ser alcançado.

Talvez a frustração mais relevante encontrada foi a impossibilidade de ter desenvolvido a interface gráfica do novo projeto usando a tecnologia *Windows Presentation Foundation*, pois não haveria tempo suficiente para que eu aprendesse de maneira razoável a linguagem *WPF*. Então tomei a decisão de continuar usando *Windows Forms* pela minha familiaridade com esta tecnologia.

7.2 Disciplinas mais relevantes

Programação Orientada a Objetos O estudo da programação orientada a objetos traz uma melhor idéia de como se deve organizar um sistema em classes e como elas devem se relacionar, facilitando a concepção de qualquer projeto de *software*.

Métodos Numéricos da Álgebra Linear Disciplina responsável por introduzir o método de mínimos quadrados, que foi implementado e utilizado no módulo de ensaio de tração.

Laboratório de Programação Extrema Além de exercitar técnicas de programação extrema, essa disciplina insere o aluno em um ambiente de trabalho onde se deve cooperar no desenvolvimento de projetos reais.

Referências

[1] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Objected-Oriented Software*. Addison-Weskey Professional; 1 edition, November 10th, 1994

[2] W3 Schools *OS Platform Statistics*. 2012 http://www.w3schools.com/browsers/browsers_os.asp

[3] Souza, Sérgio Augusto de. *Ensaios mecânicos de materiais metálicos. Fundamentos teóricos e práticos*. Edgard Blücher; 5ª edição, 1982

[4] L. N. Trefethen, D. Bau III. *Numerical Linear Algebra*.

[5] *TIOBE Programming Community Index for November 2012*. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html