Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Curso de Ciência da Computação
Disciplina de Estágio Integrado

Implementação de uma biblioteca científica para o software SmartPumping

Relatório de Estágio Integrado

Laboratório de Sistemas Distribuídos

Francisco Vilar Brasileiro
Orientador Acadêmico

Bárbara Lopes Voorsluys
Orientadora Técnica

Cícero Alan Leite Cruz
Estagiário

Campina Grande, Setembro de 2007.

Biblioteca Setorial do CDSA. Maio de 2021.

Sumé - PB

Trabalho desenvolvido como relatório de estágio integrado orientado pelo professor Dr. Francisco Vilar Brasileiro durante o período letivo de 2007.1.

Aluno

Nome: Cícero Alan Leite Cruz
Curso: Ciência da Computação
Matrícula: 20411002
Endereço Residencial: Rua João Julião Martins, 642, Apt. 303 – Bodocongó – CEP 58109-090, Campina Grande – PB – Brasil.
Telefone: (83) 8861 7247
E-mail: alan@lsd.ufcg.edu.br

Orientador

Nome: Francisco Vilar Brasileiro
Endereço Profissional: Universidade Federal de Campina Grande, Departamento de Sistemas e Computação, Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, 882, Bloco CO. CEP 58109-970, Bodocongó, Campina Grande, PB, Brasil.
Telefone: +55 (83) 3310 1365 (ramal: 30)
E-mail: fubica@dsc.ufcg.edu.br

Supervisora Técnica

Nome: Bárbara Lopes Voorsluys
Endereço Profissional: Universidade Federal de Campina Grande, Departamento de Sistemas e Computação, Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, 882, Bloco CO. CEP 58109-970, Bodocongó, Campina Grande, PB, Brasil.
Telefone: +55 (83) 3310 1365 (ramal: 23)
E-mail: barbara@lsd.ufcg.edu.br

Sumário

1. Introdução

O *software SmartPumping* foi concebido com o objetivo de simular e otimizar a operação de malhas de escoamento de líquidos produzidos na prospecção terrestre de petróleo, de forma a garantir a máxima eficiência de transporte com o menor custo de energia, redução de pressão nos dutos, dos riscos de falhas operacionais e da poluição ambiental e da perda de produção. É uma ferramenta computacional que visa dar suporte aos operadores das malhas de escoamento e contribuir para a automação do seu controle [2].

Ao longo da duração da simulação, o *SmartPumping* calcula os valores de vazão e velocidade em cada duto, da pressão em cada nó da malha e dos níveis dos tanques, das propriedades do fluido em todos os pontos da malha, derivados da mistura dos diferentes fluidos extraídos dos poços de petróleo, do custo da operação simulada, resultante do custo relativo ao consumo e à demanda de energia. É possível visualizar os gráficos de todas essas variáveis e observar como o sistema comporta-se durante o horizonte de operação simulado.

A ferramenta de otimização fornece um escalonamento das bombas que otimiza o escoamento da produção na malha de oleodutos durante um certo período no futuro (horizonte de operação), atendendo a restrições de vazões e pressões nos dutos (mínimas e máximas), da capacidade de armazenamento dos tanques, de operação de bombas e gerenciais ou administrativas, obtendo-se a máxima produção com o menor custo.

Para tratar de tantas variáveis envolvidas nos cálculos realizados por todas as ferramentas do *software*, optou-se pela utilização de uma biblioteca *Java* responsável por realizar conversões de unidades, operações matemáticas entre valores de diferentes unidades e operações de álgebra linear. Porém, a biblioteca escolhida, denominada *JScience*, acabou por trazer algumas incertezas nos resultados obtidos pelo *SmartPumping*.

Buscando solucionar esses e outros problemas relacionados com os cálculos realizados pelo *software*, foi desenvolvida uma nova biblioteca, a *SP-Science*.

Este relatório é o resultado do trabalho de estágio na implementação da biblioteca *SP-Science*. Este trabalho está dividido da seguinte forma. Na seção 2 é descrito o ambiente de estágio. Na seção 3 é descrita a metodologia utilizada para o desenvolvimento do estágio. Na seção 4 descreve-se alguns problemas encontrados pelos desenvolvedores e pelos usuários do *SmartPumping* quanto às implicações do uso da biblioteca *JScience*. Na seção 5 é mostrada uma solução para o problema descrito na seção 4. Na seção 6 são detalhadas as atividades desenvolvidas e finalmente na seção 7 são tecidas algumas considerações finais.

2. Ambiente do Estágio

2.1. Descrição do Ambiente de Estágio

A execução do estágio foi realizada no Laboratório de Sistemas Distribuídos, na Universidade Federal de Campina Grande, situada à Av. Aprígio Veloso, 882, Bodocongó.

O LSD conta com a colaboração de dezenas de alunos desenvolvendo trabalhos de doutorado, mestrado e de iniciação científica, além de vários pesquisadores. Os diferentes projetos em execução estão distribuídos em oito salas, cada uma contando com uma estante com livros de temas diversos relacionados à área de computação e engenharia.

O projeto *SmartPumping* dispõe dos equipamentos descritos na Tabela 1.

Quantidade	Tipo de equipamento	Função
3	Computador Pentium 4 – 3GHz – 1GB de RAM	Atividades de desenvolvimento
1	Computador Pentium 4 – 3GHz – 1GB de RAM	Atividades de gerência e servidor de banco de dados
1	Computador Pentium 4 – 2.8GHz – 512 MB de RAM	Ativiades de desenvolvimento
1	Computador Athlon XP – 1.54GHz – 1.21GB de RAM	Atividades de Engenharia
1	Notebook Pentium M – 1500MHz – 1GB de RAM	

Tabela 1 – Equipamentos do SmartPumping

2.2. Equipe

O projeto *SmartPumping* é uma parceria entre a Petrobras e a UFCG e conta com uma equipe de desenvolvimento formada por quatro alunos de graduação do curso de Ciência da Computação desta universidade e com uma equipe de engenharia formada por dois alunos de pós-graduação em Recursos Hídricos e um aluno de graduação em Engenharia Civil. A

equipe é gerenciada por uma mestra em Ciência da Computação, que cuida do planejamento, da execução e da evolução do projeto, motivando a equipe e acompanhando seu desempenho.

São coordenadores do projeto os professores Carlos de Oliveira Galvão, do Departamento de Engenharia Civil, e Francisco Vilar Brasileiro, do Departamento de Sistemas e Computação, ambos desta universidade.

2.3. Aspectos Positivos

Parte do conhecimento adquirido durante o curso de Bacharelado em Ciência da Computação pôde ser posto em prática durante o desenvolvimento das atividades no laboratório. A integração com alunos do curso de Engenharia Civil e com professores com um vasto conhecimento em suas áreas propiciou um aprendizado extracurricular importante para minha vida acadêmica e profissional.

2.4. Aspectos Negativos

A impossibilidade de realizar exclusivamente este trabalho durante todo o tempo de estágio foi um ponto negativo. Em alguns momentos foi necessário fazer um intervalo nas atividades de implementação/integração da biblioteca para atender a demandas urgentes que surgiam no projeto.

3. Metodologia

Há vários processos de desenvolvimento de software que podem ser aplicados. Entre estes não há o melhor processo, mas o que melhor se adequa ao projeto, considerando equipe e o problema.

No projeto *SmartPumping*, foi utilizado o processo de desenvolvimento do LSD, o qual congrega algumas das práticas de *eXtreme Programming* [3] e que se mostrou adequado às necessidades do projeto.

Nas próximas subseções será melhor detalhado o processo de desenvolvimento do LSD [4] e como este foi utilizado durante a realização deste estágio.

3.1. Princípios do processo

Projeto simples: Incentiva práticas que reduzam a complexidade do sistema e prioriza arquitetura simples que suporte a funcionalidade especificada pelo cliente. O desenvolvedor não deve inserir funcionalidades que acha que serão necessárias no futuro. Este princípio foi aplicado a este trabalho, da forma que se priorizou a implementação de funções mais críticas para o software e que pudessem ser testadas e agregadas continuamente.

Testes freqüentes: Os testes devem ser elaborados antes do código. Tornam o software capaz de aceitar mudanças, pois qualquer modificação realizada deverá passar por toda a bateria de testes já criados. Toda funcionalidade deve ter um teste automático.

A codificação da biblioteca foi inteiramente guiada por testes de unidade, utilizando para isso o *framework JUnit* [5]. A cada módulo implementado, eram realizados testes de desempenho, nos quais o tempo de realização de determinadas operações era medido e confrontado com as operações correspondentes na biblioteca *JScience*, acompanhados de perfilamentos do código, realizados com a ferramenta *JProfiler* [6], na finalidade de realizar melhorias sempre que possível.

Os vários testes de aceitação do *SmartPumping* foram executados continuamente durante a fase de integração da biblioteca ao *software*, para assim garantir a corretude do sistema. Além destes, testes de uso foram realizados a cada nova versão lançada, com a finalidade de assegurar a satisfação do usuário frente às mudanças pelas quais todo o código foi submetido.

Refatoramento: Deve ocorrer quando o programador percebe que o código pode ficar mais simples ou quando o sistema dá sinais de que um refatoramento é necessário. Deve-se garantir que todos os testes continuarão passando.

Mesmo com todos os esforços presentes neste estágio para que refatoramentos não fossem necessários, em alguns momentos, principalmente após observar pontos críticos no código através de perfilamentos, eram realizados refatoramentos visando, neste caso, melhorias no desempenho.

Programação em par (*pair programming*) e revisão de código: Enquanto um desenvolvedor implementa parte do código, o outro estará pensando na solução como um todo. Todo o código deve ser escrito por duas pessoas trabalhando na mesma máquina.

No projeto, a programação em par, em alguns momentos, não pôde ser seguida, pois a demanda de atividades do projeto necessitava que o par dividisse as atividades com a finalidade de concluir a tarefa em menos tempo. Porém, nessas ocasiões, os dois

programadores mantinham uma comunicação constante, além da integração do código, fazendo com que o trabalho fosse do inteiro conhecimento de ambos.

Posse coletiva do código: Sempre que alguém perceber que algo pode ser melhorado deve manifestar-se. Todo o time é responsável por todo o código do sistema.

Integrações contínuas: Todo o código deve estar em um repositório comum. O código é integrado com o repositório e testado em curtos espaços de tempos. Todos os testes devem ser executados diariamente ou sempre depois de uma modificação.

Durante este estágio, apenas o repositório local era constantemente atualizado, pois o objetivo era liberar e integrar ao produto apenas no momento em que a biblioteca estivesse concluída e devidamente testada. Com a biblioteca implementada iniciou-se a fase de sua integração ao software. Todos os testes de unidade e de aceitação existentes no *SmartPumping* eram executados continuamente, à cada modificação, para assim garantir a corretude e integridade do sistema.

Participação ativa do cliente: O cliente deve continuamente se reunir com time para responder perguntas e estabelecer prioridades. É fundamental que este cliente seja alguém que irá realmente usar o sistema.

O projeto, porém, não pôde dispor do cliente como membro efetivo da equipe. A forma encontrada para superar esta dificuldade foi a diminuição do intervalo entre as liberações, e a cada entrega fazer uma reunião com todos os clientes e representantes dos futuros usuários do *software*.

Padrões de codificação: Deve haver um padrão para evitar que cada desenvolvedor escreva da sua maneira, e posteriormente ninguém entenda o código. Para o desenvolvimento dos trabalhos deste estágio, foi seguido o padrão pré-estabelecido para o projeto, baseado nos padrões fornecidos pela *Sun*.

Pequenas liberações: Liberar várias versões em curtos espaços de tempo. Liberações com novas funcionalidades ou melhorias em funcionalidades já existentes. O objetivo é colocar rapidamente novas versões do sistema em produção.

Esboço de projeto: Um novo módulo ou funcionalidade do sistema deve ter sua especificação escrita em US. Se esta US requer a implementação de novos módulos, deve ser escrito um esboço de projeto para ser discutido anteriormente com a equipe. Esta prática ajuda a aparar as arestas e facilitar o conhecimento coletivo do sistema.

Utilização de uma ferramenta para comunicação interna: Uso do *wiki*, interno, para divulgar as informações das linhas. Tudo relativo à gerência de configuração do projeto deve estar documentada, pública e atualizada, com domínio coletivo. O uso de uma

ferramenta de comunicação proporciona, de forma simples, uma iteração dinâmica entre as equipes. Como todas as pessoas têm uma visão integral do projeto, em muitos casos elas podem assumir diferentes papeis.

No projeto *SmartPumping*, todos os princípios descritos no processo foram seguidos o máximo possível. A presença de quatro integrantes na equipe de desenvolvimento fez com que itens como posse coletiva de código, integrações contínuas e padrões de codificação fossem uma necessidade.

3.2. Papéis do processo

Existem nove papéis desempenhados por pessoas no processo do LSD: Cliente, Gerente de Projeto, Pesquisador, Consultor, *Coach*, Líder de Linha, Desenvolvedor, Testadores e *Tracker*. Nesta seção são detalhadas algumas das atividades realizadas pelas pessoas que desempenham esses papéis.

O **Cliente** é o representante da empresa financiadora dos projetos do LSD. Suas atividades principais são: idealizar o produto através da descrição de suas necessidades, priorizar o que deve ser feito, descrever e executar os testes de aceitação. O papel do cliente em alguns projetos é difícil de definir. Por isso, algumas vezes um professor da UFCG, um pesquisador, coordenador ou outra pessoa pode assumi-lo.

O **Gerente de Projeto** é uma pessoa interna ao LSD responsável pela gerência de RH, de custos, de aquisições, riscos, garantia de comunicação com o cliente. Pode assumir ainda algumas atividades que poderiam ser de responsabilidade de um gerente financeiro.

O **Pesquisador** é um aluno ou profissional responsável pelo estudo e pesquisa em áreas afins conforme os interesses do LSD. Suas Atividades são: fundamentar teoricamente as suas áreas de pesquisa e propor novas soluções, ferramentas, métodos, modelos que possam aprimorar o desenvolvimento e concepção dos projetos. Normalmente professores da UFCG atuam como pesquisadores.

O **Consultor** é um especialista de determinada área e pode ajudar as equipes de várias formas. Suas atividades são: debater junto com líderes das linhas sobre questões que exijam uma fundamentação teórica nas suas áreas de pesquisa, auxiliar na definição de Visões do Cliente e Tarefas nas linhas de pesquisa e, quando requisitado, produzir ou revisar relatórios para a linha. Em muitos casos os Pesquisadores podem assumir o papel de Consultores.

O *Coach* é uma Pessoa interna ao LSD responsável por coordenar, avaliar, acompanhar o plano de trabalho das equipes, bem como promover melhorias no processo de desenvolvimento.

O **Líder de Linha** é uma pessoa interna ao LSD responsável por coordenar, avaliar, acompanhar o trabalho da sua equipe.

Os **Desenvolvedores** são pessoas internas ao LSD responsáveis pela implementação do produto, bem como testes e interfaces. Dentre suas principais atividades têm-se: estimar o tempo de desenvolvimento das atividades, atualizar o status de suas atividades diariamente, implementar testes de unidade, de aceitação e integração, refatorar constantemente para manter o código "limpo", integrar diariamente com o repositório, seguindo os cuidados mínimos exigidos, revisar o código e, quando requisitado, produzir ou revisar relatórios para a linha.

Os **Testadores** são pessoas (internas ou externas ao LSD) responsáveis por validação e verificação dos produtos de software em desenvolvimento e consequentemente garantir a qualidade do software produzido. Existem muitos níveis de testadores, a contratação de pessoas internas para esse papel normalmente têm o perfil de desenvolvedores na condição de testadores.

O *Tracker* é uma pessoa responsável por acompanhar o andamento das atividades da equipe LSD. Sua atividade é a de acompanhar o andamento das atividades garantindo o cumprimento do plano de liberação. Normalmente o *coach* ou o líder de linha assumem este papel.

Quanto aos papéis desempenhados no projeto, Bárbara assumiu as atividades de gerente de projeto, *coach*, *tracker* e líder de linha. As atividades de desenvolvimento foram destinadas a quatro estudantes do curso de Ciência da Computação, que ao mesmo tempo eram testadores do *software*. Já as atividades de pesquisa foram assumidas por dois engenheiros, estudantes da pós-graduação em Engenharia Civil, que também assumiram papéis de testadores.

Especificamente para a realização deste estágio, Bárbara, supervisora técnica deste trabalho, além dos seus demais papéis assumidos no projeto, desempenhou a função de cliente. O desenvolvimento ficou a cargo exclusivamente do estagiário, tendo, porém, a participação dos demais membros da equipe de desenvolvimento como testadores da biblioteca.

4. Descrição do Problema

Desenvolvido com a linguagem de programação *Java*, o *SmartPumping* fazia uso em larga escala de uma biblioteca científica denominada *JScience*. Esta biblioteca provê, entre outras funcionalidades, conversões de unidades, operações entre valores de diferentes unidades, operações entre matrizes e em álgebra linear, que são necessárias na execução de várias funções em todos os módulos do *SmartPumping*, inclusive na interface com o usuário.

Para um sistema de apoio à decisão da grandeza do *SmartPumping*, qualquer imprecisão nos cálculos de uma simulação ou otimização, podem converter-se em resultados errôneos e influenciar a decisão final a não optar pelo que seria a melhor escolha. Para garantir essa qualidade, a equipe de desenvolvimento preocupa-se constantemente em implementar e executar testes de unidade, bem como executar testes de aceitação definidos pelos usuários.

A biblioteca científica *JScience*, no entanto, trouxe sérios problemas aos desenvolvedores do *software* e certa insatisfação por parte dos usuários. Muito freqüentemente, operações matemáticas realizadas por classes do *JScience* resultam no valor real da operação acrescido de um pequeno valor (maior que zero e menor que um), funções que definem se dois valores são próximos acabam retornando verdadeiro para valores não tão próximos assim, entre outros problemas.

Imprecisões como essas acabavam deixando o usuário confuso quando mostradas na interface gráfica, além de provocar resultados errados. Tal situação fazia com que os programadores procurassem soluções alternativas, tornando o código mais vulnerável a erros.

As operações realizadas pelo *JScience* mostraram-se ainda muito ineficientes no que diz respeito ao desempenho computacional. Perfilamentos do código mostram que boa parte do tempo de execução de uma simulação era devido a operações desse tipo.

O *SmartPumping*, portanto, necessitava de uma implementação que substituísse a referida biblioteca, de modo a garantir a precisão das operações e que reduzisse ao máximo possível o tempo de processamento destas operações, proporcionando assim a confiabilidade que um *software* deste porte necessita.

5. Proposta de Solução

A biblioteca *SP-Science*, implementada na linguagem *Java*, foi idealizada para substituir a biblioteca *JScience*, com a finalidade de sanar todos os problemas por ela trazidos e atender às exigências do cliente.

Esta implementação foi sucedida por um refatoramento de todo o código do *SmartPumping*, motivado pelo uso da nova biblioteca em substituição ao *JScience* e por remover todas as soluções alternativas codificadas para contornar os erros descritos na seção 4, deixando o código mais limpo e claro.

Nas próximas subseções, será descrita a forma como a nova biblioteca foi implementada e como foi realizado o refatoramento.

5.1. Implementação da biblioteca

A nova biblioteca foi desenvolvida com base na mesma estrutura da anterior, porém sempre buscando a simplicidade de código na finalidade de tornar as operações mais diretas e precisas.

De início, realizou-se um perfilamento de algumas operações realizadas pelo *JScience*, para assim identificar os principais pontos passíveis de melhorias quanto ao tempo de processamento. Alguns pacotes e suas respectivas classes destacaram-se neste contexto, o que pode ser visto no **Anexo B**. Através de um estudo do seu código fonte e de codificação de alguns testes de unidade, procurou-se identificar ainda as implementações de métodos que retornavam resultados incertos.

O *JScience* disponibiliza em sua página *web* o seu código fonte e declara em seu termo de uso que todos os seus módulos são livres, que a permissão para usar, copiar, modificar e distribuir estes módulos está concedida livremente, contanto que as observações de *copyright* sejam preservadas [1].

Portanto, buscou-se reaproveitar os pacotes desta biblioteca que não representavam um impacto tão expressivo no seu desempenho e que não forneciam erros aos resultados das operações. Com isso, realizou-se um planejamento do que seria implementado e do que seria reutilizado, com base nas necessidades mais urgentes do *software*.

A codificação do *SP-Science* é composta dos seguintes pacotes:

- **org.smartpumping.science.unidades:** Neste pacote encontram-se as classes responsáveis por representar e construir unidades e realizar operações entre elas.

As unidades foram classificadas em duas categorias: as unidades base e as unidades compostas, as quais podem ser construídas a parir de operações com unidades base, ou com outras unidades compostas. Faz parte da implementação base da nova biblioteca.

- **org.smartpumping.science.quantity:** As classes deste pacote são responsáveis por realizar a associação de tipos de quantidades (comprimento, massa, tempo, pressão, vazão, etc.) a suas respectivas unidades *default*. Faz parte da implementação base da nova biblioteca.
- **org.smartpumping.science.conversores:** Este foi o único pacote reutilizado da biblioteca *JScience*. Suas seis classes têm a função de construir conversores dos valores de uma unidade para outra. Estas classes preservam sua documentação original, na qual está descrito o termo de uso do código do *JScience*, citado anteriormente.
- **org.smartpumping.science.measure:** O correspondente a este pacote na biblioteca *JScience* foi o responsável pela maioria das suas imperfeições e, por este motivo, destinou-se uma maior atenção à sua nova implementação. As classes deste pacote são responsáveis pela representação e construção de medidas, bem como pelas operações realizadas entre elas. Estas medidas consistem na associação entre um valor e uma unidade. Optou-se por uma implementação simples e direta, desconsiderando alguns atributos e tratamentos, como uma faixa de erro atribuída a cada medida, realizados pelo *JScience* e descartáveis pelo uso no *SmartPumping*, proporcionando um ganho no desempenho das operações e um maior nível de confiança nos seus resultados. Faz parte da implementação intermediária da nova biblioteca.
- **org.smartpumping.science.math.funcao:** Este pacote reúne as classes responsáveis por representar funções polinomiais e suas operações. Construir e manipular funções com *JScience* não são tarefas tão triviais, por isso, buscou-se uma interface que proporcionasse ao programador uma fácil utilização das classes deste pacote.

Os perfilamentos realizados no código recém implementado serviram para identificar pontos críticos, propiciando o aperfeiçoamento desta codificação através de pequenos refatoramentos. O uso de estruturas de dados, por exemplo, tais como listas e mapas, foi revisto cuidadosamente.

Como resultado, observou-se uma diminuição no tempo de processamento das operações implementadas, quando comparadas a operações correspondentes na biblioteca *JScience*. Para se chegar a tal conclusão, realizaram-se várias execuções destas operações, gerando-se um comparativo que pode ser visto no **Anexo C**.

A corretude dos resultados pôde ser assegurada durante toda a fase de implementação por meio da execução dos testes de unidade. As simplificações realizadas na codificação, principalmente por tratar medidas de quantidades como uma simples combinação de um tipo primitivo de Java, representando o valor, e de uma unidade, foram cruciais para a erradicação de casas decimais indesejáveis e de outras imperfeições. Tudo isto contribuiu para que os resultados produzidos pela *SP-Science* fossem mais confiáveis.

5.2. *Integração ao SmartPumping e refatoramento*

Com a nova biblioteca devidamente implementada e testada, iniciou-se a fase de integração ao código do *SmartPumping*. Como a biblioteca *JScience* estava envolvida em todos os cálculos realizados por todos os módulos do software, inclusive o seu núcleo, foi necessária a preparação de um maior ambiente de testes, desta vez englobando todos os testes de unidade e de aceitação existentes no *SmartPumping*.

Para isto, iniciou-se um trabalho de conversão de arquivos, denominados arquivos de previsão da produção, de cada simulação a ser executada pelos testes de aceitação. Estas conversões eram necessárias devido à mudança de biblioteca. Com todos estes arquivos atualizados e todos os testes de aceitação reunidos e passando, pôde-se dar início à integração.

Quase todo o código foi modificado para se adequar ao *SP-Science*, o que representou uma operação muito arriscada. Cada classe, porém, foi modificada juntamente com a sua classe de teste correspondente, de modo que a tarefa de integração para esta classe era apenas concluída quando seu teste de unidade estivesse passando. Ao final da integração de cada módulo, todos os testes de aceitação eram executados. Qualquer erro detectado por eles era imediatamente investigado.

À medida que era feita a integração, realizou-se também um refatoramento para remover do código todas as soluções alternativas implementadas para contornar os erros apresentados na seção 4. Modificações para contribuir na melhoria do desempenho também foram realizadas sempre que possíveis. Como exemplo disto, pode-se citar a substituição de todas as multiplicações e divisões de um objeto *Measure* com outro adimensional pelas

mesmas operações sendo, porém, realizadas com valores representados por tipos primitivos de *Java*, no lugar de objetos *Measure* adimensionais.

5.3. Resultados

Ao final da integração e refatoramento, observou-se uma diminuição significativa no tempo de execução das funcionalidades do *SmartPumping*. Para comprovar este fato, foram realizadas várias execuções utilizando o código de duas versões distintas do *software*, uma com a biblioteca *JScience*, e outra com a *SP-Science*. Houve, em média, uma melhoria de 58% na simulação de uma malha pequena e cerca de 25% para uma malha de grande porte. Estes comparativos podem ser vistos no **Anexo D.**

Quanto à corretude dos resultados, observou-se que, com a simples substituição da biblioteca, alguns *bugs* do *software* que estavam cadastrados foram resolvidos, as casas decimais indesejáveis não mais ocorriam e os resultados dos cálculos igualavam-se até varias casas decimais com aqueles obtidos pela equipe de engenharia.

Estes resultados, portanto, se mostraram bastante satisfatórios e mostram que os objetivos esperados no Plano de Estágio (ver Anexo E) puderam ser alcançados.

6. Detalhamento das Atividades Desenvolvidas

As atividades do estágio foram distribuídas ao logo de sua duração como mostra o **Gráfico 1**, onde o valor em porcentagem indica a proporção do tempo gasto com a atividade em relação ao tempo total.

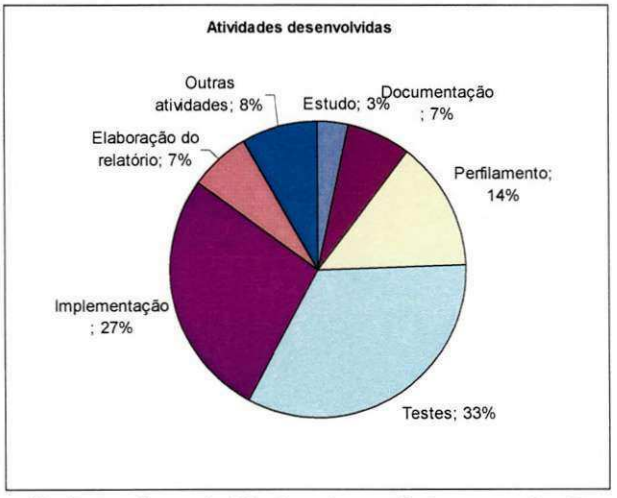


Gráfico 1: Distribuição das atividades do estágio em relação ao tempo total.

Observa-se que as atividades de testes e perfilamentos ocuparam quase 50% do tempo total do estágio, o que mostra a preocupação constante em garantir a corretude e a melhoria de desempenho do sistema.

7. Considerações Finais

Como pôde ser observado, a biblioteca *SP-Science* é uma solução para os problemas encontrados no *SmartPumping* com o uso da biblioteca *JScience*. Além disso, a melhoria no desempenho proporcionada ao software após a sua integração, foi um resultado que agradou aos usuários, fazendo assim com que o estagiário alcançasse os objetivos apresentados no Plano de Estágio (ver Anexo E).

O estágio foi de grande importância para o aprendizado do estagiário e sua experiência do que diz respeito ao desenvolvimento de *software*, pois se puderam colocar em prática várias teorias que foram contempladas em disciplinas da graduação, além da realização intensa de testes e perfilamentos para obter a validação necessária do produto final. Portanto, houve um aprimoramento na habilidade do estagiário em realizar estas tarefas que são de suma importância à sua formação acadêmica e ingresso no mercado de trabalho.

Dois itens são de grande importância em um trabalho de estágio: uso de uma metodologia e orientação de um profissional experiente. O fato de não se ter utilizado o processo de desenvolvimento do LSD com todas as suas atividades não invalidou a metodologia, mas privou o trabalho de alguns de seus benefícios, como a prática de *pair programming*. Contudo, considera-se que este trabalho tenha sido de um grande aprendizado tanto em nível de pesquisa como em nível profissional.

Agradecimentos

Gostaria de agradecer especialmente a Bárbara por ter dado toda atenção à execução deste trabalho, cuidando para que as interrupções provocadas por outras demandas do projeto não prejudicassem o andamento das atividades do estágio; Ao professor Fubica pela orientação e pela oportunidade de executar este trabalho; Finalmente agradeço aos demais integrantes da equipe de desenvolvimento do *SmartPumping*, que em muito me ajudaram com conversas muito produtivas e esclarecimentos de dúvidas.

Referências

[1] JScience. Disponível em: http://www.jscience.org/. Acessado em: 05/06/2007.

[2] SmartPumping. *Manual Técnico, 2007*. Campina Grande, 2007.

[3] Extreme Programming: A gentle introduction. Disponível em: http://www.extremeprogramming.org/. Acessado em: 01/10/2007.

[4] Laboratório de Sistemas Distribuídos. *Processo de desenvolvimento do LSD*. Disponível em: http://www.lsd.ufcg.edu.br/. Acessado em: 01/10/2007.

[5] JUnit.org Resources for Test Driven Development. Disponível em: http://www.junit.org/. Acessado em: 03/10/2007.

[6] Java Profiler – JProfiler. Disponível em: http://www.ej-technologies.com/products/jprofiler/overview.html. Acessado em: 03/10/2007.

ANEXOS

ANEXO A

Declaração de Aprovação do Estágio

Declaro, para os devidos fins, que **Cícero Alan Leite Cruz**, estudante do curso de Ciência da Computação da Universidade Federal de Campina Grande, matrícula 20411002, cumpriu corretamente suas atividades de desenvolvimento da biblioteca SP-Science, que foi o principal alvo de seu estágio no projeto *SmartPumping*, realizado no período de 04/06/2007 a 28/09/2007.

_____

Bárbara Lopes Voorsluys
Supervisora Técnica

ANEXO B

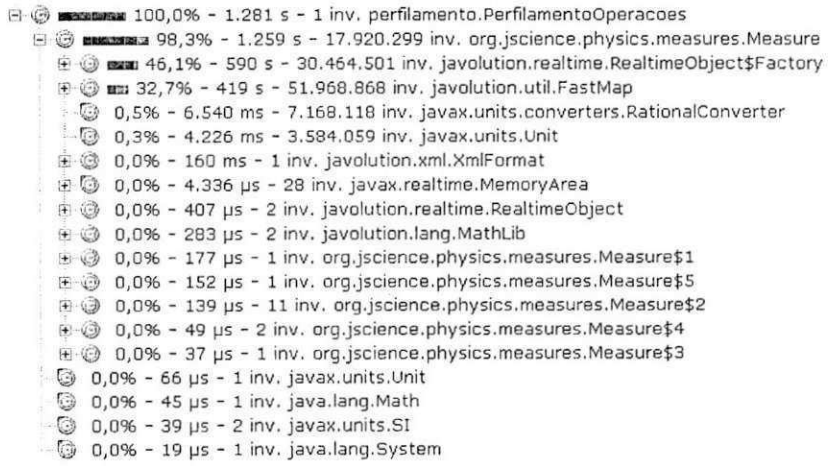
Resultados de perfilamentos de algumas operações realizadas pelo *JScience*, com o objetivo de identificar pontos críticos no que diz respeito ao tempo de processamento.

Os perfilamentos listados abaixo correspondem à execução da operação:
P = (C/D^5) * F * Q^2 * M * q * (coef), que representa o cálculo da perda de pressão em um duto, realizada no simulador do *SmartPumping*, onde: P = perda de pressão (Pascal), C = comprimento do duto (metro), D = diametro do duto (metro), F = fator de atrito (adimensional), Q = vazão no duto (m^3/s), M = massa específica do fluido (kg/m^3), q = quota de volume do fluido em relação ao fluido total contido no duto (adimensional), coef = (8.0 /PI^2).

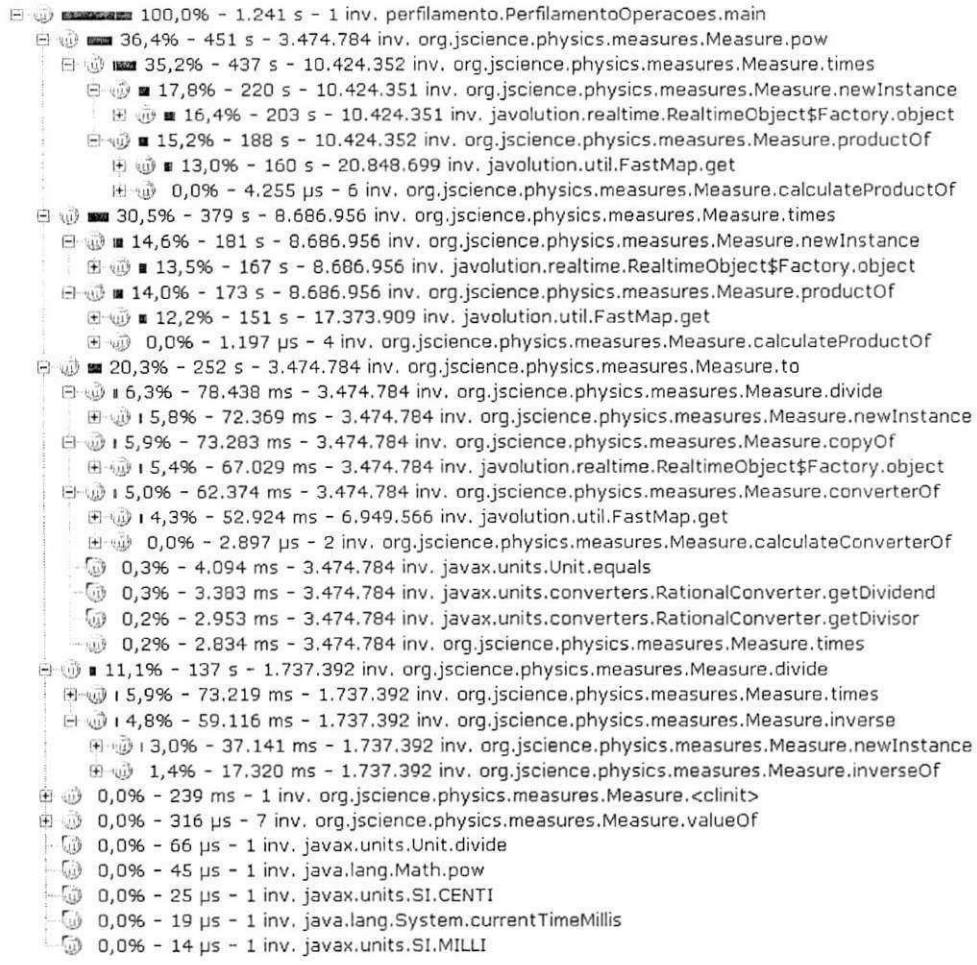
A distribuição do tempo de processamento gasto na operação pode ser observado com relação aos pacotes, classes e métodos envolvidos.

- Pacotes:

- Classes:



- Métodos



Como pode ser observado, as classes pertencentes ao pacote *org.jscience.physics.measures*, principalmente a classe *Measure*, em combinação com classes de outra biblioteca usada pelo *JScience*, a *Javolution*, consomem a maior parte do tempo de processamento da operação.

ANEXO C

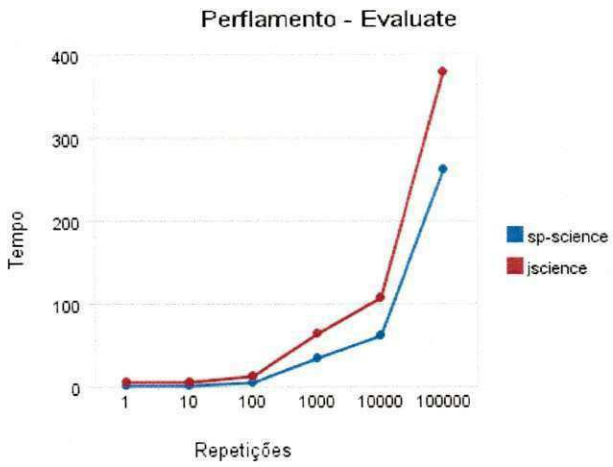
Comparativos do tempo de execução do uso das bibliotecas *JScience* e *SP-Science*.

- Execuções simultâneas de operações em funções polinomiais implementadas pelo *SP-Science* e pelo *JScience*.

  - **Operação: *Evaluate***

    Avaliação de um polinômio de segundo grau da forma $Ax^2 + Bx + C$, onde A, B e C são constantes de unidades $s^2/m^{\wedge}5$, s/m^2 e m, respectivamente, e sendo a variável x atribuída com o valor de 2 m^3/s.

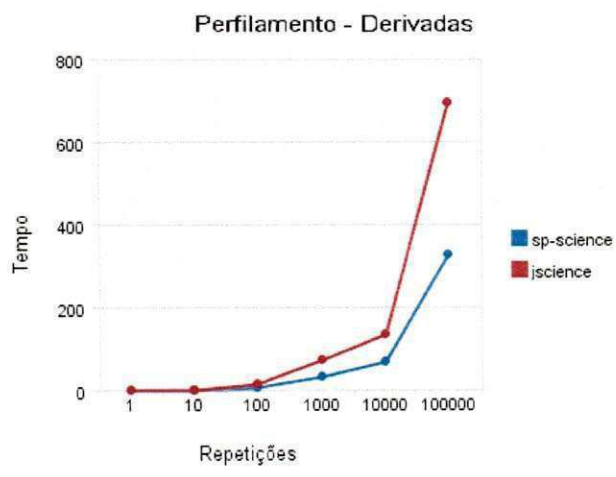
Repetições	Tempo (ms)	
	sp-science	jscience
1	1	5
10	2	6
100	6	13
1000	38	64
10000	68	107
100000	262	379
1000000	2120	2831
10000000	20274	28206



  - **Operação: *Differenciate***

    Cálculo da derivada de um polinômio de segundo grau da forma $Ax^2 + Bx + C$, com relação à variável x, onde A, B e C são constantes de unidades $s^2/m^{\wedge}5$, s/m^2 e m, respectivamente.

Repetições	Tempo (ms)	
	sp-science	jscience
1	0	0
10	1	1
100	6	14
1000	33	74
10000	70	133
100000	328	694
1000000	2854	6330
10000000	28420	64046

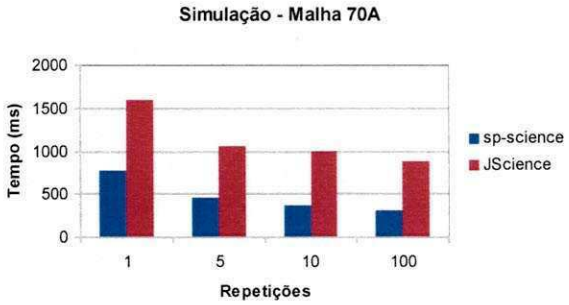


ANEXO D

- Tempo de execução de simulações:

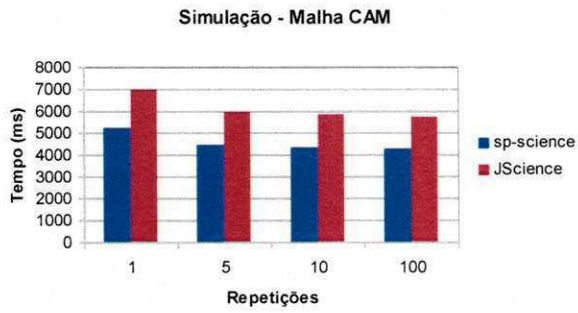
  - **Malha 70A:** Simulação de uma malha com 4 estações, 4 tanques, 6 bombas, 5 dutos e 5 nós.

REPETIÇÕES	TEMPO (ms)		MELHORIA
	sp-science	JScience	
1	772	1587	51,35%
5	466	1053	55,75%
10	366	992	63,10%
100	306	868	64,75%



  - **Malha CAM:** Simulação de uma malha com 20 estações, 20 tanques, 47 bombas, 78 dutos e 62 nós.

REPETIÇÕES	TEMPO (ms)		MELHORIA
	sp-science	JScience	
1	5222	6933	24,68%
5	4457	5932	24,87%
10	4362	5823	25,09%
100	4283	5692	24,75%



ANEXO E

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO

Plano de Estágio Integrado

Projeto SmartPumping
Laboratório de Sistemas Distribuídos

Cícero Alan Leite Cruz

Mat.: 20411002

Orientador: Prof. Francisco Vilar Brasileiro

Maio de 2007

1. Ambiente do Estágio

A execução do estágio será realizada no Laboratório de Sistemas Distribuídos, na Universidade Federal de Campina Grande, situada à Av. Aprígio Veloso, 882, Bodocongó.

O laboratório está instalado em um prédio com 550m^2 de área e conta com a colaboração de dezenas de alunos desenvolvendo trabalhos de doutorado, mestrado e de iniciação científica, além de vários pesquisadores.

O LSD utiliza-se da organização dos desenvolvedores e pesquisadores em espaços denominados *ilhas* que são divididas em 4 (quatro) espaços ocupados pelos mesmos, ou em mesas de trabalho. Os diferentes projetos em execução estão distribuídos em 8 (oito) salas, cada uma contando com quadro branco, pincéis, ar-condicionado e estante com livros de temas diversos relacionados à área de computação e engenharia.

O projeto *SmartPumping* é uma parceria entre a Petrobras e a UFCG e conta com uma equipe de desenvolvimento formada por quatro alunos de graduação do curso de Ciência da Computação desta universidade e com uma equipe de engenharia formada por dois alunos de pós-graduação em Recursos Hídricos e um aluno de graduação em Engenharia Civil. A equipe é gerenciada por uma mestra em Ciência da Computação, que cuida do planejamento, da execução e da evolução do projeto, motivando a equipe e acompanhando seu desempenho.

2. Supervisão

2.1. *Supervisor Acadêmico*

Nome: Francisco Vilar Brasileiro

Endereço: Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, 882, Bloco CO.
58109-970, Bodocongó,
Campina Grande, PB, Brasil.
Telefone: +55 (83) 3310 1365, ramal 30
Fax: +55 (83) 3310 1498

Email: fubica@dsc.ufcg.edu.br

2.2. *Supervisora Técnica*

Nome: Bárbara Lopes Voorsluys

Endereço: Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, 882, Bloco CO.
58109-970, Bodocongó,
Campina Grande, PB, Brasil.
Telefone: +55 (83) 3310 1365, ramal 28
Fax: +55 (83) 3310 1498

Email: barbara@lsd.ufcg.edu.br

3. Resumo do Problema

O *software SmartPumping* foi concebido com o objetivo de simular e otimizar a operação de malhas de escoamento de líquidos produzidos na prospecção terrestre de petróleo, de forma a garantir a máxima eficiência de transporte com o menor custo de energia. É uma ferramenta computacional que visa dar suporte aos operadores das malhas de escoamento e contribuir para a automação do seu controle.

Desenvolvido com a linguagem de programação *Java*, o *SmartPumping* faz uso em larga escala de uma biblioteca científica denominada *JScience*. Esta biblioteca provê, entre outras coisas, conversões de unidades, operações entre valores de diferentes unidades, operações entre matrizes e operações em álgebra linear, que são necessárias na execução de várias funções em todos os módulos do *SmartPumping*, inclusive na interface com o usuário.

Para um sistema de apoio à decisão da grandeza do *SmartPumping*, qualquer imprecisão nos cálculos de uma simulação ou otimização, podem converter-se em resultados errôneos e influenciar a decisão final a não optar pelo que seria a melhor escolha. Para garantir essa qualidade, a equipe de desenvolvimento preocupa-se constantemente em implementar e executar testes de unidade, bem como executar testes de aceitação definidos pelo usuário.

A biblioteca científica *JScience*, no entanto, tem trazido sérios problemas aos desenvolvedores do *software* e certa insatisfação por parte dos usuários. Muito freqüentemente, operações matemáticas realizadas por classes do *JScience* resultam no valor real da operação acrescido de um pequeno valor (maior que zero e menor que um) e funções que definem se dois valores são próximos acabam retornando verdadeiro para valores não tão próximos assim.

Imprecisões como essas acabam deixando o usuário confuso quando mostradas na interface gráfica, além de provocar resultados errados. Tal situação faz com que os programadores procurem soluções alternativas, tornando o código mais vulnerável a erros.

As operações realizadas pelo *JScience* mostraram-se ainda muito ineficientes no que diz respeito ao desempenho computacional. Perfilamentos do código mostram que boa parte do tempo de execução de uma simulação é devido a operações desse tipo.

O *SmartPumping*, portanto, necessita de uma implementação que substitua a referida biblioteca, de modo a garantir a precisão das operações e que busque reduzir ao máximo o tempo de processamento destas operações, proporcionando assim a confiabilidade que um *software* deste porte necessita.

4. Proposta de Solução

A proposta deste estágio está baseada na implementação de uma biblioteca científica, na linguagem *Java*, para substituir a biblioteca atualmente usada, com a finalidade de sanar todos os problemas por ela trazidos e atender às exigências do cliente. O desenvolvimento será baseado na metodologia XP (*Extreme Programming*). Este trabalho será uma grande oportunidade de praticar todas as técnicas e conhecimentos obtidos ao longo do curso.

A metodologia deste trabalho estará dividida em partes. Inicialmente será realizado um estudo sobre este tipo de biblioteca e dos requisitos necessários, bem como um embasamento teórico sobre os diferentes tipos de operações a serem realizadas por ela. Em seguida será iniciada a codificação, utilizando técnicas de desenvolvimento orientada a testes (*Test Driven Development*) por meio de ferramentas, como *JUnit* e *EasyAccept*.

Com a biblioteca implementada será iniciada a fase de sua integração ao software que será feita de forma pontual. Todos os testes de unidade e de aceitação existentes no *SmartPumping* serão executados continuamente, à cada modificação, para assim garantir a corretude e integridade do sistema.

Considerando essas técnicas de desenvolvimento como sendo de extrema importância para a formação profissional e pessoal, o estágio integrado deverá permitir que as práticas do processo de desenvolvimento de software possam ser executadas de forma eficiente, para que assim, possamos obter os benefícios desejados na execução destas atividades.

5. Atividades a serem Desenvolvidas

As atividades planejadas, com suas respectivas estimativas de tempo, são as seguintes:

Atividades	Horas estimadas
Estudo Bibliográfico e definição de requisitos	20
Implementação da biblioteca	80
Testes	80
Integração da biblioteca ao *SmartPumping*	100
Testes de desempenho do sistema	20
Total	**300**

O desenvolvimento das atividades durante o estágio ocorrerá de acordo com o cronograma abaixo:

	Jun/07				Jul/07				Ago/07				Set/07			
Planejamento do Estágio																
Estudo Bibliográfico sobre sistemas de unidades e bibliotecas matemáticas.																
Implementação da biblioteca																
Integração da biblioteca ao *SmartPumping*																
Testes																
Elaboração do Relatório																
Defesa do Estágio																

Obs.:

- Cada casela corresponde a uma semana. O tempo total de execução do estágio será de 15 semanas, com dedicação de 20 horas por semana.
- A elaboração do relatório final do estágio será feita continuamente, a fim de concentrar as principais observações e experiências adquiridas ao longo do período.