Centro de Ciência e Tecnologia

Departamento de Sistemas e Computação

Ciência da Computação

Relatório técnico de estágio Integrado: CELB/SAELPA/UFCG

Implementação de Funcionalidades na Estação de Gerência de um Sistema de Medição Remota de Energia Elétrica

Estagiário: Gustavo Wagner Diniz Mendes

Professor Orientador: Jacques Philippe Sauvé

Professor Colaborador: Pedro Sergio Nicolletti

Campina Grande, Março de 2004

Trabalho desenvolvido como relatório de estágio integrado orientado pelo professor Dr. Jacques Philippe Sauvé durante o período letivo de 2002.2.

Dados do aluno

Nome: Gustavo Wagner Diniz Mendes
Curso: Ciência da Computação
Matrícula: 20011032
Endereço Residencial: Rua João Julião Martins, 623 – Bodocongó – CEP 58.109-090, Campina Grande – PB – Brasil
Fone: (83) 88063013
e-mail: gustavo@dsc.ufcg.edu.br

Dados do Orientador

Nome: Jacques Philippe Sauvé
Endereço Profissional: Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Departamento de Sistemas e Computação.
Av. Aprígio Veloso, s/n
Bodocongó, Campina Grande – PB – Brasil
e-mail: jacques@dsc.ufcg.edu.br
Home page: http://jacques.dsc.ufcg.edu.br

Sumário

Sumário 3
1. Introdução 5
2. Ambiente de Estágio 6
2.1 Ambiente de estágio 6
2.1.1 Descrição do Ambiente de Estágio 6
2.1.2 Equipe 6
2.1.3 Aspectos Positivos 7
2.1.4 Aspectos Negativos 7
3. Metodologia 7
3.1 Papéis do Processo 7
3.2 Atividades do Processo 9
3.2.1 Atividades de Planejamento 9
3.2.2 Atividades de Projeto 10
3.2.3 Atividades de Testes 11
3.2.4 Atividades de Integração 11
3.2.5 Atividades de Gerência 12
3.3 Uso do processo 13
4. Descrição do problema 14
5. Proposta de Solução 14
5.1 Arquitetura do sistema SALAME 15
5.1.1 Estação de Gerência 16
5.1.2 Concentrador Regional 16
5.1.3 Concentrador de Transformador 16
5.1.4 Medidores 16
5.2 Funções da Estação de Gerência 16
5.2.1 Controle de acesso ao sistema 17
5.2.2 Gerência de erros de comunicação 17
5.2.3 Controle remoto dos dispositivos 18
5.2.4 Polling dos dispositivos 19
5.2.5 Gerência de *logs* de execução 19
5.2.6 Controle dos dispositivos na rede (inserção, alteração e remoção de medidores e concentradores) 20
5.2.7 Relatórios aos clientes 20
5.3 User Stories 21
6. Detalhamento das Atividades Desenvolvidas 23
6.1 Atividades 23
6.1.1 Tratamento de exceções 23
6.1.2 Erro de Comunicação com Concentrador Regional 25
6.1.3 Exibição das Últimas Leituras de Consumo 26
6.2 Planos de Iteração 27
Plano de iteração: 19/02/2003-12/03/2003 27
Plano de iteração: 05/02/2003-19/02/2003 28
Plano de iteração: 22/01/2003-05/02/2003 28
Plano de iteração: 06/01/2003-22/01/2003 28
7. Considerações Finais 29
8. Agradecimentos 30
Referências 31
Anexos 32

Anexo A : Artigo - Uso de SNMP para a Gerência de Sistemas de Medição de Energia Elétrica..... 32
Anexo B : Plano de Estágio..... 32
Anexo C : Manual de Instalação da Estação de Gerência..... 32

1. Introdução

A gerência de redes de energia elétrica por parte das distribuidoras depara-se com o problema de dependência de informação humana, que pode ser imprecisa e colhida de forma ineficiente. Imprecisa pois medições feitas por um humano podem ser falhas, e ineficiente pois controlar toda uma rede complexa como a de distribuição de energia elétrica pode se tornar inviável apenas com uma grande quantidade de pessoal. É necessário que haja uma automação nesse processo com o intuito de reduzir gastos e aumentar a precisão dos dados colhidos. Além disso, consumidores estão desprovidos de informações mais precisas sobre sua utilização de energia elétrica, tanto quantitativamente (consumo) quanto qualitativamente (utilização de energia elétrica no tempo).

Um sistema de medição é um conjunto de componentes e a forma de interação entre estes para prover os dados. O sistema de medição mais utilizado atualmente é o de medição presencial, formado por medidores e leituristas, onde estes têm o papel de realizar a coleta da leitura de consumo nas residências ou em estabelecimentos industriais e comerciais. Como já explanado, esse tipo de leitura traz imprecisão e ineficiência ao processo. Sistemas de medição remota trazem uma maior agilidade e possibilita um maior controle sobre seus componentes. É possível a esses sistemas ler a qualquer momento os consumos dos medidores, além da gerência (monitoração e controle) sobre seus dispositivos.

Buscando solucionar esses e outros problemas relacionados com sistemas de medição, o sistema SALAME (Sistema Remoto de Medição de Energia Elétrica) foi desenvolvido pela Universidade Federal de Campina Grande em parceria com a CELB/SAELPA.

Este relatório é o resultado do trabalho de estágio no sistema SALAME. Este trabalho está dividido da seguinte forma. Na seção 2 descrevemos o ambiente de estágio. Na seção 3 descrevemos a metodologia utilizada para o desenvolvimento do estágio. Na seção 4 descrevemos alguns problemas encontrados pelas concessionárias CELB/SAELPA quanto à leitura de energia elétrica. Na seção 5 mostramos uma solução para o problema descrito na seção 4. Na seção 6 detalhamos as atividades desenvolvidas e finalmente na seção 7 tecemos algumas considerações finais.

2. Ambiente de Estágio

2.1 Ambiente de estágio

2.1.1 Descrição do Ambiente de Estágio

Durante o estágio utilizamos as instalações do Laboratório de Automação de Sistemas Elétricos (LASE). O LASE está localizado no bloco CJ do Departamento de Engenharia Elétrica - DEE sob direção administrativa do professor Dr. Luis Reyes Rosales.

O LASE é utilizado como um ambiente de pesquisa e desenvolvimento de soluções em sistemas elétricos. Atualmente as pesquisas que estão sendo desenvolvidas são um esforço para a automação de medição elétrica para a distribuidora de energia elétrica CELB/SAELPA.

Os equipamentos que o LASE dispõe estão descritos na Tabela 1.

Quantidade	Tipo de Equipamento	Função
1	Computador Pentium III – 800MHz – 64MB	Servidor de aplicação
1	Computador Pentium I – 400MHz – 32MB	Manter o Concentrador Regional
1	Computador Pentium III – 1.2GHz – 64MB	Desenvolvimento de aplicação
1	Computador Pentium III – 1.2GHz – 64MB	Desenvolvimento de aplicação
9	Medidor Monofásico	Testes das aplicações

Tabela 1 - Equipamentos do LASE

Os trabalhos realizados pelos estagiários no LASE foram de pesquisa e desenvolvimento de soluções para o projeto SALAME. O laboratório foi utilizado pelos alunos do DEE e do Departamento de Sistemas e Computação - DSC.

2.1.2 Equipe

O LASE comportava professores e alunos dos cursos de Ciência da Computação e Engenharia Elétrica. O projeto era gerenciado pelo professor do Departamento de Engenharia Elétrica Dr. Luis Reyes Rosales. O professor Dr. Jacques Philippe Sauvé e o

professor Pedro Sérgio Nicolletti (Peter) eram os consultores de informática. Eu e Hugo Ranieri Di Assunção Brasilino fomos os alunos de Computação responsáveis por acrescentar funcionalidades à estação de gerência. O professor Jacques nos orientava e gerenciava nas tarefas a serem desenvolvidas.

2.1.3 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 Elétrica e com professores com um vasto conhecimento em suas áreas nos propiciou um aprendizado extracurricular importante para a nossa vida acadêmica e profissional.

2.1.4 Aspectos Negativos

Não havia uma infra-estrutura adequada para a realização das atividades. Havia poucos computadores para a quantidade de pessoas que utilizavam o laboratório, além da sala ser de pequeno porte. Era necessário trabalhar à noite para que não houvesse choque de horário no uso dos computadores.

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 a um determinado problema.

Buscamos um processo de desenvolvimento que fosse adequado para uma equipe de pequeno porte e que não fosse denso quanto à produção de artefatos. Desta forma resolvemos utilizar o XP1 [XP1], um processo de desenvolvimento de software que congrega algumas das práticas de eXtreme Programming [XP] e que se mostra ideal para projetos que são desenvolvidos no meio acadêmico.

Nas próximas subseções detalhamos melhor o processo XP1.

3.1 Papéis do Processo

Existem três papéis desempenhados por pessoas no processo XP1: i) Cliente, ii) Desenvolvedor e iii) Gerente de projeto. Nesta seção detalhamos algumas das atividades realizadas pelas pessoas que desempenham esses papéis.

O Cliente é responsável por:

- Descrever as funcionalidades desejadas para o sistema, também chamadas de *User Stories*;
- Descrever teste(s) de aceitação para cada *User Story*. Esta é uma ferramenta importante para o cliente, pois com testes de aceitação pode-se validar o que ele requisitou;
- Descrever os requisitos não funcionais do sistema. Requisitos não funcionais são restrições impostas ao sistema, tais como: tempo de resposta de uma consulta a um banco de dados, tipo de interface, entre outras;
- Definir o plano de *release*, no qual são definidas as datas de liberação de uma unidade executável do sistema. O plano de *release* é o projeto de desenvolvimento em alto nível;
- Escolher quais *User Stories* serão desenvolvidas em cada iteração. Iteração é uma unidade de planejamento detalhado. Em XP1, uma iteração dura duas semanas.

O Desenvolvedor é responsável por:

- Ajudar no levantamento de *User Stories* e requisitos não funcionais junto ao cliente;
- Estimar o tempo para desenvolver cada *User Story*. O cliente é quem define as *User Stories*, mas o desenvolvedor é quem sabe quanto tempo se levará para realizá-las;
- Dividir as *User Stories* em tarefas e estimar o tempo de desenvolvimento de cada uma;
- Escolher as tarefas que deseja desenvolver na iteração corrente;
- Desenvolver os testes de unidade e escrever o código das tarefas. Teste de unidade é um teste automático que verifica se uma determinada classe (unidade básica em Programação Orientada a Objetos) está funcionando como deveria;
- Integrar o código constantemente a um repositório de versões.

O Gerente de projeto é responsável por:

- Conduzir as atividades de planejamento do projeto. As atividades de planejamento são i) escrita de *User Stories,* ii) levantamento de requisitos não funcionais, iii) planejamento de *releases* e planejamento de iteração;
- Avaliar os riscos do projeto. Um risco é um fator que pode levar ao não cumprimento do prometido;
- Manter o progresso do sistema através de métricas definidas tais como quantidade de classes produzidas, quantidade de componentes, etc.

3.2 Atividades do Processo

Há cinco tipos de atividades no processo XP1: i) Atividades de Planejamento, ii) Atividades de Projeto, iii) Atividades de Testes, iv) Atividades de Integração e v) Atividades de Gerência. Nesta seção descrevemos cada uma delas.

Cada atividade é realizada por uma ou várias pessoas. Ao realizar uma atividade uma pessoa desempenha um determinado papel, como citado em 3.1.

3.2.1 Atividades de Planejamento

O fluxo das atividades de planejamento pode ser visto na Figura 1.

Figura 1 - Atividade de Planejamento

Primeiro, o cliente descreve os requisitos funcionais do sistema através das *User Stories*. *User Stories* devem ter pouco detalhe, de forma que os desenvolvedores possam estimar o tempo de cada uma delas. Além disto, para cada *User Story*, o cliente deve descrever pelo menos um teste de aceitação.

A próxima atividade é descrever os requisitos não funcionais do sistema. Os requisitos não funcionais não necessitam de testes de aceitação. Eles descrevem o que o sistema deve ter quanto à segurança, interface, entre outros.

Após terem sido levantados os requisitos do sistema, é necessário que se planeje a disponibilização deste em ambiente externo. Isso é feito através da definição do plano de *release*s. No plano de *release*s, o cliente define o que se deve ter de funcionalidades no sistema e em quais datas. Após serem definidas as *User Stories* que comporão os *release*s, os desenvolvedores devem estimar os tempos de cada uma. O *release* é planejado por tempo, não por escopo. Essa é uma vantagem herdada de XP, pois é mais fácil controlar o escopo de um *release* do que o tempo em que ele será disponibilizado.

Por fim, faz-se o planejamento da iteração. Uma iteração em XP1 dura duas semanas. Aqui, assim como no plano de *release*s, o tempo é fixo e o escopo é variável. Os desenvolvedores devem escolher as *User Stories* que podem ser realizadas no seu tempo disponível durante a iteração e dividí-las em tarefas. Cada tarefa deve ter uma curta duração e seus tempos devem ser estimados pelos seus respectivos desenvolvedores.

3.2.2 Atividades de Projeto

O fluxo das atividades de projeto pode ser visto na Figura 2.

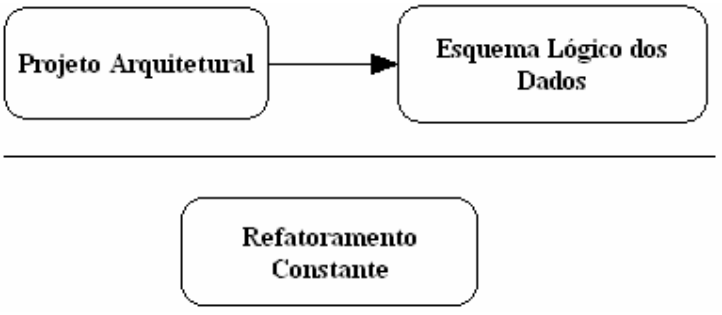

Figura 2 - Atividades de Projeto

Nela vemos que a primeira atividade a ser desenvolvida é a geração do projeto arquitetural, ou seja, a organização do sistema em módulos. Esses módulos descrevem as várias camadas do sistema. Logo após, se houver banco de dados no sistema, elabora-se o modelo lógico dos dados. Deve-se planejar esta atividade com muito cuidado, pois

uma simples alteração no modelo lógico pode acarretar numa mudança brusca no sistema.

A última atividade de projeto é a de refatoramento constante. Refatorar significa alterar código sem que se altere a sua funcionalidade. Veja que essa atividade ocorre em todo momento e em paralelo com as outras atividades.

3.2.3 Atividades de Testes

O fluxo das atividades de testes pode ser visto na Figura 3.

Figura 3 - Atividades de Testes

Há duas atividades que merecem destaque nessa figura. A primeira é a escrita de testes de unidade. Testes de unidade testa uma classe individual, ou um conjunto de classes interdependentes. Eles são completamente automatizados. Todo código que é desenvolvido deve estar com uma cobertura mínima de testes de unidade e estes devem estar funcionando 100%.

A segunda diz respeito com os testes de aceitação. Cada *User Story* deve corresponder no mínimo a um teste de aceitação. Isso garante ao cliente que o que ele requisitou foi cumprido.

Por fim, a atividade de *test review* é realizada por uma pessoa diferente daquela que desenvolveu o código e o teste. Ela corresponde a uma verificação no código e nos testes de unidade e de aceitação. Essa atividade é importante, pois ela tenta assegurar a qualidade do código e dos testes escritos.

3.2.4 Atividades de Integração

O fluxo das atividades de integração pode ser visto na Figura 4.

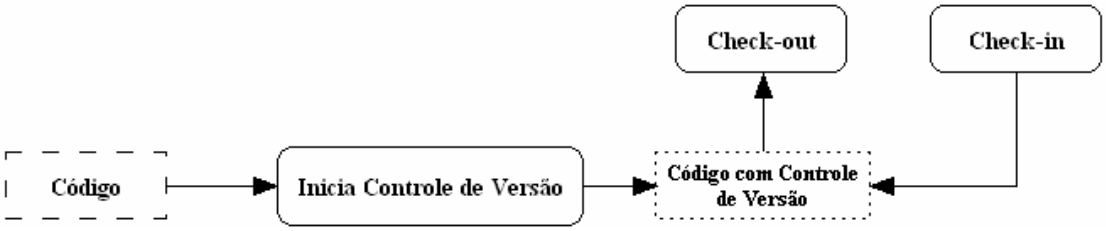

Figura 4 - Atividades de Integração

As atividades de integração são importantes principalmente quando há mais de uma pessoa trabalhando com o código do sistema. Há a possibilidade de dois ou mais desenvolvedores alterarem a mesma parte do código, caracterizando uma situação de conflito. Além disso, havendo um ponto único onde os artefatos são colocados reduz o problema de desenvolvedores estarem mexendo em versões antigas espalhadas por diretórios diferentes.

Primeiramente, os artefatos desenvolvidos devem ser colocados sobre controle de versão. Para isso é necessária uma ferramenta de controle de versões, tal como Concurrent Versions System (CVS) [CVS]. Quando houver necessidade de mexer em algum artefato, faz-se o check-out (tirar os artefatos do controle de versão). Esse artefato representa o último que sofreu alterações. Fazem-se as alterações necessárias e no fim das delas deve-se realizar o check-in (colocar os artefatos alterados no controle de versões). Nesse momento pode haver conflitos, que devem ser tratados, pois podem ser uma fonte de *bugs*. As melhores práticas de integração são integrar em seqüência e em curto espaço de tempo, pois reduz a possibilidade de conflito.

3.2.5 Atividades de Gerência

O fluxo das atividades de gerência pode ser visto na Figura 5.

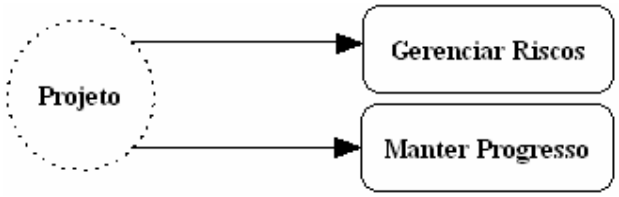

Figura 5 - Atividades de Gerência

As atividades de gerência do projeto são muito importantes, pois dão uma visão real do projeto ao gerente e ao cliente. Primeiramente, devem ser avaliados os riscos do projeto, ou seja, as possibilidades de não cumprir o que foi prometido. Isso deve ser feito numa reunião realizada uma vez por semana entre o gerente e os desenvolvedores e em conversas rápidas no dia a dia. Os riscos devem ser avaliados e soluções devem ser discutidas para que os riscos sejam eliminados do projeto. Deve-se montar uma tabela de riscos e essa precisa ser avaliada constantemente.

Por fim o gerente deve manter o progresso do projeto a partir da avaliação de métricas definidas. Tais métricas podem ser, por exemplo, quantidade de módulos do sistema, quantidade de *User Stories* realizadas, entre outras. Com os valores de tais métricas monta-se um gráfico chamado big-chart que representa o progresso do sistema.

3.3 Uso do processo

Quando entramos no projeto, grande parte do código já tinha sido desenvolvida pelo professor Dr. Jacques. Desta forma, algumas das atividades sugeridas pelo processo não foram seguidas por nós estagiários. Entre elas podemos destacar as atividades de projeto tais como a definição do projeto arquitetural e do esquema lógico de dados. Acreditamos que uma das experiências mais interessantes vivida no estágio foi o fato de termos pego um código em produção, ter que entender e estender com as funcionalidades requeridas pelo projeto. Sabemos que muitos dos trabalhos que iremos encontrar no mercado são projetos já em andamento que precisam de novas funcionalidades.

Precisamos definir bem os papéis que foram desempenhados no processo. O gerente de projeto era o professor Dr. Rosales. Mas em nível do projeto de software o gerente era o professor Dr. Jacques. O papel de cliente era desempenhado também pelo professor Jacques (a partir deste ponto o professor Jacques ora será chamado de cliente, ora será chamado de gerente), e o papel de desenvolvedor foi desempenhado por mim e por Hugo Ranieri (a partir deste ponto os estagiários Gustavo Wagner e Hugo Ranieri serão chamados de desenvolvedores).

As *User Stories* foram definidas pelo cliente, assim como o processo sugere. As atividades de integração foram realizadas pelos desenvolvedores. Inicialmente estas atividades não eram realizadas. Com o passar do tempo vimos que sem o trabalho de

integração contínua um projeto de software pode chegar ao ponto de *Integration Hell* (inferno da integração), como citado em [Continuous Integration].

As reuniões entre o cliente, gerente e desenvolvedores ocorria uma vez por semana, às quartas-feiras. Na primeira reunião da iteração o cliente definia quais *User Stories* iriam ser contempladas, e cada desenvolvedor escolhia quais *User Stories* iria desenvolver. Algumas delas já tinham suas tarefas definidas, em outras os desenvolvedores, juntamente com o professor Jacques, as definiam. Além disso, os desenvolvedores diziam quanto tempo levariam para cumprir cada tarefa alocada para eles. Na segunda reunião da iteração eram realizadas as atividades de gerência. Analisávamos os riscos e colocávamos para cada tarefa as horas trabalhadas e o percentual de completude. Dependendo do andamento da iteração, algumas tarefas alocadas para os desenvolvedores poderiam ser adiadas para a próxima iteração. Lembramos que as iterações e *release*s têm tempo fixo e escopo variável.

4. Descrição do problema

Atualmente as empresas concessionárias de energia elétrica CELB/SAELPA utilizam um sistema de medição convencional, onde leituristas coletam informações dos medidores dos clientes residenciais e empresariais de forma presencial. Uma parte dos consumos lidos pelas concessionárias é aferida dos próprios medidores dos clientes e outra parte através de médias de consumos anteriores. Veja que os dados não são precisos, haja vista que eles são coletados por seres humanos e que parte deles são uma média de valores. Além disso, os consumidores não têm acesso a informações detalhadas sobre seus consumos.

5. Proposta de Solução

O sistema SALAME é uma solução completa para os problemas descritos na seção 4. Um dos pontos mais importantes no sistema é a apresentação, na Internet, de gráficos de consumos para os clientes da concessionária. Dessa forma, eles têm a possibilidade de escolher horários de consumos (fora de picos) para que seus gastos com energia elétrica sejam reduzidos. O sistema SALAME utiliza *Power Line Communication* (PLC) para a leitura dos dados de consumo dos clientes utilizando a rede elétrica de baixa tensão.

5.1 Arquitetura do sistema SALAME

Nas próximas seções mostramos o sistema SALAME como um todo, bem como as soluções desenvolvidas em cada módulo, destacando a estação de gerência.

O sistema de medição SALAME tem como propósito geral agilizar o processo de obtenção de dados referentes à rede de distribuição de energia elétrica de forma a abrir um novo canal de fornecimento de informação para os clientes da empresa de distribuição; adicionalmente, o sistema auxilia na manutenção dos dispositivos da rede. Na Figura 6 mostramos a arquitetura do sistema SALAME.

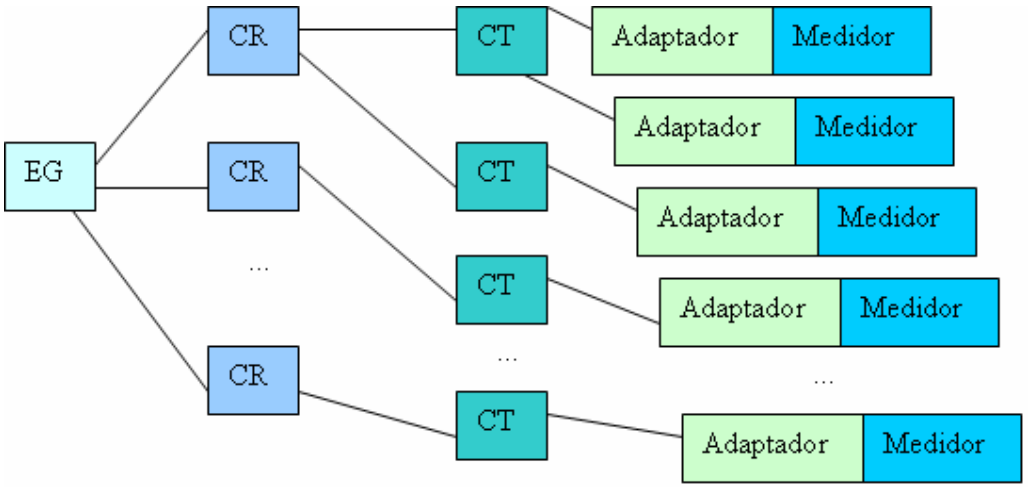

Figura 6 - Arquitetura do sistema SALAME

Toda comunicação efetuada entre os módulos do sistema é feita por demanda, significando que um módulo nunca pode responder a outro caso não tenha sido requisitado. A Estação de Gerência (EG) é um componente de software que opera em rede cuja principal função é a de gerenciar os dispositivos. O Concentrador Regional (CR) também é um componente de software que tem como principal função a obtenção de dados relativos ao estado dos medidores e aos concentradores. O Concentrador de Transformador (CT) tem a função de monitorar um conjunto reduzido de medidores. Para que esta arquitetura seja implantada no sistema de medição convencional, é preciso que se coloque um adaptador eletrônico para a comunicação do CT com os medidores e a inserção de CTs nos transformadores. A comunicação entre os componentes está detalhada nas subseções seguintes.

5.1.1 Estação de Gerência

A estação de gerência (EG) se comunica com o concentrador regional (CR) através do protocolo TCP/IP e utiliza o protocolo de aplicação SNMP (*Simple Network Management Protocol*) [SNMP] para realizar as leituras dos medidore
s e controle dos dispositivos do sistema.

5.1.2 Concentrador Regional

A comunicação entre Concentrador Regional (CR) e Concentrador de Transformador (CT) se dá via enlaces de rádio. O protocolo de aplicação utilizado é o ACS (Aneel/Celb-Saelpa).

5.1.3 Concentrador de Transformador

A comunicação entre o CT e os medidores se dá por meio de *Power Line Communication (PLC)* utilizando a rede elétrica de baixa tensão; o protocolo de aplicação utilizado é o SNAP1. Na realidade, a comunicação se dá entre CT e o Adaptador junto ao medidor.

5.1.4 Medidores

Os medidores, que podem ser monofásicos ou trifásicos, são instalados nas dependências dos consumidores. Cada medidor tem um adaptador conectado para que a comunicação seja efetuada.

5.2 Funções da Estação de Gerência

Sistemas de medição de energia convencionais não dão suporte à monitoração de seus dispositivos. O que ocorre são vistorias técnicas a medidores quando estes apresentam alguma falha. Isso acarreta dois problemas. O primeiro diz respeito aos gastos envolvidos na manutenção dos dispositivos e o segundo na diminuição da qualidade de serviço prestado aos consumidores. Havendo gerência e monitoração efetiva dos dispositivos tais problemas podem ser sanados. Isso é possível, pois uma gerência contínua possibilita uma identificação prévia de problemas em potencial, podendo ser tratados antes que eles ocorram.

Num sistema de medição remota a estação de gerência tem o papel fundamental de monitoração dos seus dispositivos. Mostramos nas próximas subseções as funções da estação de gerência do sistema SALAME.

5.2.1 Controle de acesso ao sistema

Dado que a EG controla o sistema de medição como um todo e está disponível em rede, é necessário que haja um controle de acesso às suas funções. Além disso, pessoas com papéis diferentes na distribuidora de energia devem ter níveis de acesso diferentes à EG. Desta forma, o controle de acesso dá-se por meio de autenticação de nome de *login* e senha. A Figura 7 mostra a página de logon.

Figura 7 - Logon da Estação de Gerência

5.2.2 Gerência de erros de comunicação

A comunicação entre os dispositivos da rede de energia elétrica está propensa a falhas, haja vista sua complexidade e abrangência. Dados tais fatores, falhas no sistema de medição convencional não são fáceis de serem identificados. Havendo uma monitoração remota a identificação de falhas entre dispositivos torna-se automática. Essa funcionalidade é muito importante, pois torna mais fácil e ágil o processo de correção de falhas no sistema. Dada a arquitetura da Figura 6, os erros possíveis seriam

entre EG-CR, CR-CT e CT-Medidor. A Figura 8 mostra os erros de comunicação ocorridos num determinado intervalo de tempo.

Figura 8 - Erros de comunicação.

5.2.3 Controle remoto dos dispositivos

Uma das funções mais importantes da EG é a de controlar remotamente os dispositivos da rede. Este controle é necessário para que o sistema se torne independente de informações humanas. Leitura de consumo e corte no abastecimento de energia são exemplos de controle remoto, como pode ser visto na Figura 9.

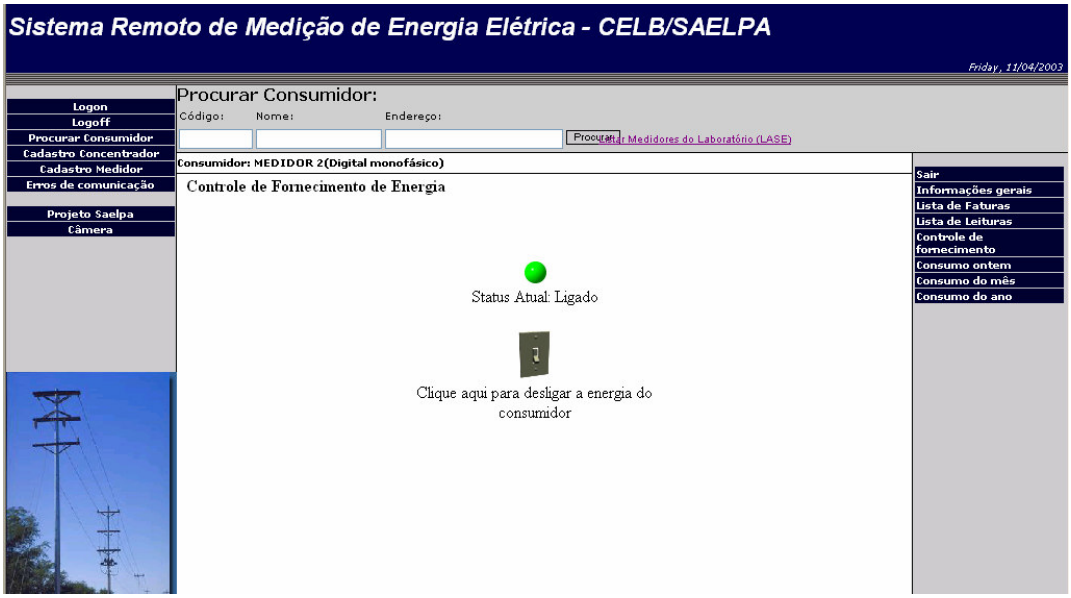

Figura 9 - Controle de fornecimento de energia.

5.2.4 Polling dos dispositivos

Para que a EG obtenha as informações das leituras dos medidores regularmente, é necessário fazer *polling* sobre todos os medidores cadastrados. Com o *polling* a EG se mantém atualizada com os dados de consumo e do status dos medidores continuamente. É possível, caso um medidor não esteja respondendo a requisições por um determinado tempo, que o *polling* não seja realizado mais sobre aquele medidor e um erro seja reportado para que o medidor seja reparado. Pode-se também, por demanda, desabilitar o *polling* nos medidores de um determinado CR.

5.2.5 Gerência de *logs* de execução

Em sistemas de computação há dois tipos bem definidos de usuários: os que utilizam o sistema como fim, ou seja, utilizam as funcionalidades que o sistema lhes oferece para o negócio e os que o utilizam para gerenciamento e aperfeiçoamento. Dada essa distinção, é necessário que *logs* para cada tipo de usuário sejam gerados, haja vista que o primeiro tipo necessita de informações referentes ao uso correto do sistema (informações não técnicas) e o segundo tipo necessita de informações técnicas para resolver eventuais problemas que ocorram na utilização do sistema. Desta forma, a EG foi projetada para que gere dois tipos de *logs*: os *logs* de usuários gerais, que são

reportados nas telas do sistema e os de gerenciamento do sistema, que são salvos em arquivo.

5.2.6 Controle dos dispositivos na rede (inserção, alteração e remoção de medidores e concentradores)

Para o sistema de medição é extremamente importante que se possa fazer controle (inserção, alteração e remoção) de todos os dispositivos na rede via web. Concentradores Regionais e Medidores podem ser controlados pela estação de gerência, bem como a listagem e identificação dos consumidores donos dos medidores. A Figura 10 mostra o cadastro de um concentrador.

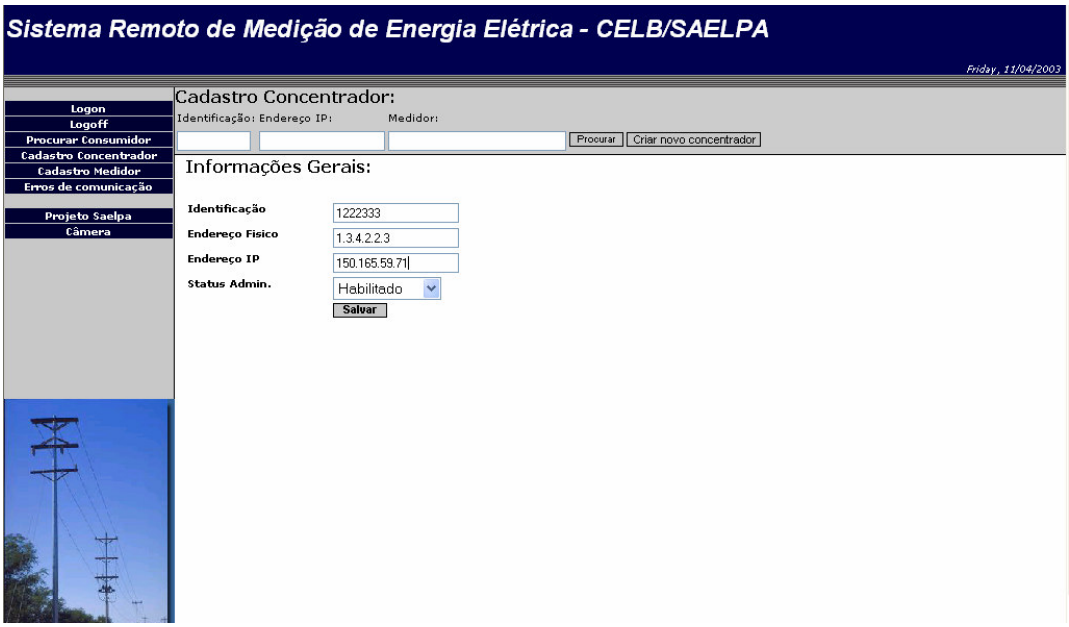

Figura 10 - Cadastro de um concentrador.

5.2.7 Relatórios aos clientes

Para que o cliente tenha disponibilizado 24x7 os dados referentes ao consumo de sua residência, a EG disponibiliza relatórios referentes a consumo diário, mensal e anual, bem como as últimas leituras de consumo realizadas. Com estes dados o cliente pode analisar os picos de consumo efetuados e tomar decisões de uso de aparelhos elétricos que antes não se podia fazer. Um gráfico do consumo do dia pode ser visto na Figura 11.

Figura 11 - Consumo do dia de um determinado consumidor.

5.3 User Stories

Nesta subseção citamos as *User Stories* desenvolvidas durante o período de estágio.

***User Story* 4 - Gustavo:** [leituras] Exibição das últimas leituras (sempre exibir leitura e consumo [classe Consumption] também). Exibição de consumo da última hora, último dia, último mês. Exibição em tabela. Últimas "n" leituras.

***User Story* 5 - Gustavo:** [dados medidor] Exibição e cadastro de dados do medidor/concentrador (ip, mac, ...).

***User Story* 6 – Gustavo e Hugo:** [erros] Exibição de detalhes completos sobre erros ocorrendo. Exibir últimos erros para cada medidor, com data. Exibir erros ocorrendo em operações online. Deixar o tratamento de erros organizado no software já feito, de forma a exibir mensagens adequadas e logar todas as atividades de erro.

- Task 6.1: Rever tratamento de erros para canalizar tudo em exceções bem organizadas (SalameException, SalameDBException, ...), diferenciar log de erros de software versus log de business;
- Task 6.2: Remover velho logger e usar log4j [log4j];

- Task 6.3: Remover todos os exits do programa que possam parar a JVM do tomcat;
- Task 6.4: Rever todas as atividades de log e logar coisas apropriadas nos lugares apropriados (log de software versus log de business);
- Task 6.5: Exibir últimos erros para cada medidor, com data.

***User Story* 7 - Gustavo:** [Erro de comunicação com concentrador regional] Qualquer erro de comunicação envolvendo o concentrador regional deve ser claramente indicado na página Web.

- Task 7.1: Combinar erros de comunicação com Peter e tratar estes erros. Na leitura, colocar erros numa tabela de erros indicando data/hora, fonte, causa;
- Task 7.2: Colocar na página Web. Pode ser tabela ou gráfico.

***User Story* 8 - Gustavo:** [wh] Colocar toda a informação em Watt-hora, tanto na página, quanto na forma de pegar medidas do concentrador.

***User Story* 12 - Gustavo:** [simultaneidade SNMP] Testar o que ocorre com acesso simultâneo ao agente de Peter. Se acesso simultâneo não for possível, inibir cliques na interface enquanto esperando resposta.

- Task 12.1: Testar a simultaneidade;
- Task 12.2: Implementar a solução (lançar exceção em acesso simultâneo);

***User Story* 14 - Gustavo:** [Web poll] Interface Web para *User Story* 1. Escolha do medidor por mac, ou por cliente, ou por concentrador, ou visualmente se deixar todos os medidores na mesma página.

- Estimativa:
  - T14.1: Servlet de boot do processo de poll;
  - T14.2: Fazer a interface gráfica, estaticamente;
  - T14.3: Dinamizar a interface gráfica, ligando com o business logic.

***User Story* 15 – Gustavo:** [Nova interface] Refazer a interface gráfica para ser mais ergonômica e mais bonita.

- Estimativa:
    - T15.1: Bolar a parte gráfica;
    - T15.2: Conectar a parte gráfica ao business logic.

***User Story* 17 – Gustavo e Hugo:** [Laptop] Instalar a estação de gerência no laptop.

***User Story* T2 – Gustavo e Hugo:** [JDBC] Abandonar castor [Castor], usar JDBC.

6. Detalhamento das Atividades Desenvolvidas

Na subseção 5.3 citamos as *User Stories* que desenvolvemos durante o estágio. Nesta seção detalhamos melhor o trabalho realizado em algumas dessas *User Stories*.

6.1 Atividades

6.1.1 Tratamento de exceções

O "design" de uma aplicação deve levar em consideração exceções que possam ocorrer nos seus módulos. É imprescindível que haja um bom "design" de exceções para que a aplicação possa se recuperar de erros e para que o usuário final tenha informações precisas sobre o estado da aplicação. Para isto, usamos uma forma de tratar exceções que separa informações para os administradores do sistema e informações para o usuário final.

Um problema comumente encontrado em diversos tipos de aplicações diz respeito a mensagens que são passadas nas exceções e que são tratadas muitas vezes de forma errônea. Pensemos num sistema com informações internacionalizadas. Quando uma exceção de qualquer tipo é lançada (ex. new `FileNotFoundException`("Arquivo não encontrado: " + nomeArquivo)), a mensagem que é passada está em uma língua específica, de forma que só é conveniente para usuários que falam esta língua.

Outro problema é que exceções têm semânticas diferentes em módulos diferentes de uma aplicação. Por exemplo, um módulo de leitura de arquivo pode lançar uma

exceção do tipo `IOException`, mas para um módulo de gerência de configuração, não é uma boa prática relançar este tipo de exceção pois pode não fazer nenhum sentido para o usuário final (talvez o melhor seria lançar um `ConfigurationException`).

Além disso, é necessário, para fins de "debug" e tomada de decisões da aplicação, que se saiba qual o nível de gravidade da exceção e qual a causa da mesma. Nível de gravidade é importante porque é possível tomar uma decisão de parar o programa, caso a exceção tenha um nível muito alto, ou continuar a execução após a aplicação se recuperar. Saber a causa da exceção é uma forma de achar qual a causa do erro cometido. Pensando nos problemas acima citados, foi desenvolvido o modelo mostrado na Figura 12.

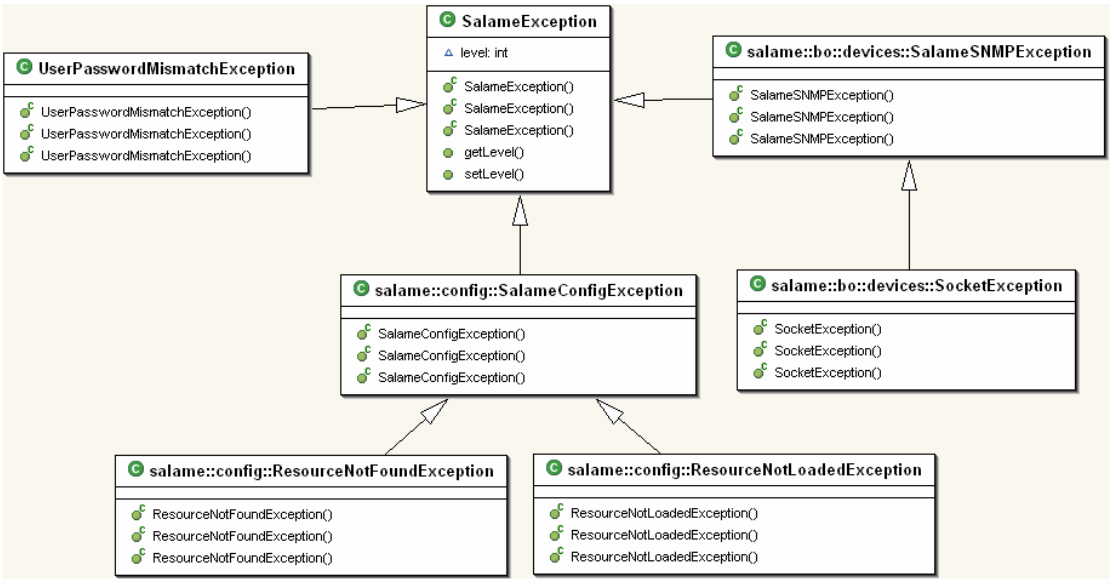

Figura 12 - Parte da hierarquia de exceção no SALAME

A partir deste modelo, a informação principal de uma exceção é a própria exceção, ou seja, seu nome. A mensagem que é passada para a mesma só deve ser utilizada para fins de "debug".

Percebe-se também, por este modelo, que um pacote tem uma exceção que representa a semântica do mesmo (ex. `SalameConfigException` do pacote `salame.config`). A partir daí, toda exceção de um determinado pacote deve herdar da exceção mais genérica do mesmo, que por sua vez herda de `SalameException`. Isto segue o modelo de exceções da Sun [Java], onde cada pacote tem uma exceção

principal, sendo que só esta é lançada. Por exemplo, o pacote IO de Java [Java] lança `IOException`, e as outras exceções, como `FileNotFoundException`, herdam de `IOException`. Com esta hierarquia de exceções, nunca é lançada uma exceção que não seja do tipo `SalameException`. Isto dá um maior controle à aplicação quanto ao tratamento dos seus erros.

Quanto à gravidade, cada exceção tem um atributo que a identifica. Os níveis possíveis são, do mais ameno ao mais grave, DEBUG, INFO, WARNING, ERROR e FATAL. Toda exceção, ao ser lançada, precisa ser identificada quanto ao seu nível de gravidade. Perceba que uma mesma exceção pode ter gravidades diferentes em diferentes partes da aplicação. Por fim, a causa da exceção deve ser preenchida com a exceção que a causou. Por exemplo, se uma exceção no código está sendo lançada, a causa da mesma é null. Mas caso uma exceção X tenha sido capturada no código e seja necessário lançar a exceção Y, coloca-se X como a causa de Y. Desta forma, fica fácil saber a causa de qualquer problema, haja vista que há uma cadeia de exceções que pode ser pesquisada.

6.1.2 Erro de Comunicação com Concentrador Regional

Todos os erros de comunicação entre os módulos são inseridos no banco de dados e reportados na web. Os possíveis erros entre os módulos da arquitetura são: EG-CR, CR-CT, CT-Medidor. No caso de erro de comunicação entre EG-CR, a EG é quem coleta essa informação. Os outros erros são reportados à EG pelo CR, através de uma variável de status de operação coletada via SNMP [SNMP]. A classe SNMPMeter, mostrada na Figura 13, é quem realiza essa tarefa. A cada comunicação SNMP entre a EG e o CR, SNMPMeter solicita também o status da operação, que é um código de erro ou o código de operação válida.

Figura 13 - Hierarquia SNMP no SALAME

6.1.3 Exibição das Últimas Leituras de Consumo

Um dos requisitos do sistema era exibir, para um dado consumidor, as últimas leituras de seu consumo. Podemos citar algumas dessas leituras: leitura da última hora, leitura do último dia e leitura do último mês. A classe `Consumption` modela um consumo e a classe `Consumer` modela um cliente. Quanto à exibição das últimas leituras, a classe `Consumer` é responsável por disponibilizar essas informações, como pode ser visto através das operações disponibilizadas por `Consumer`, na Figura 14.

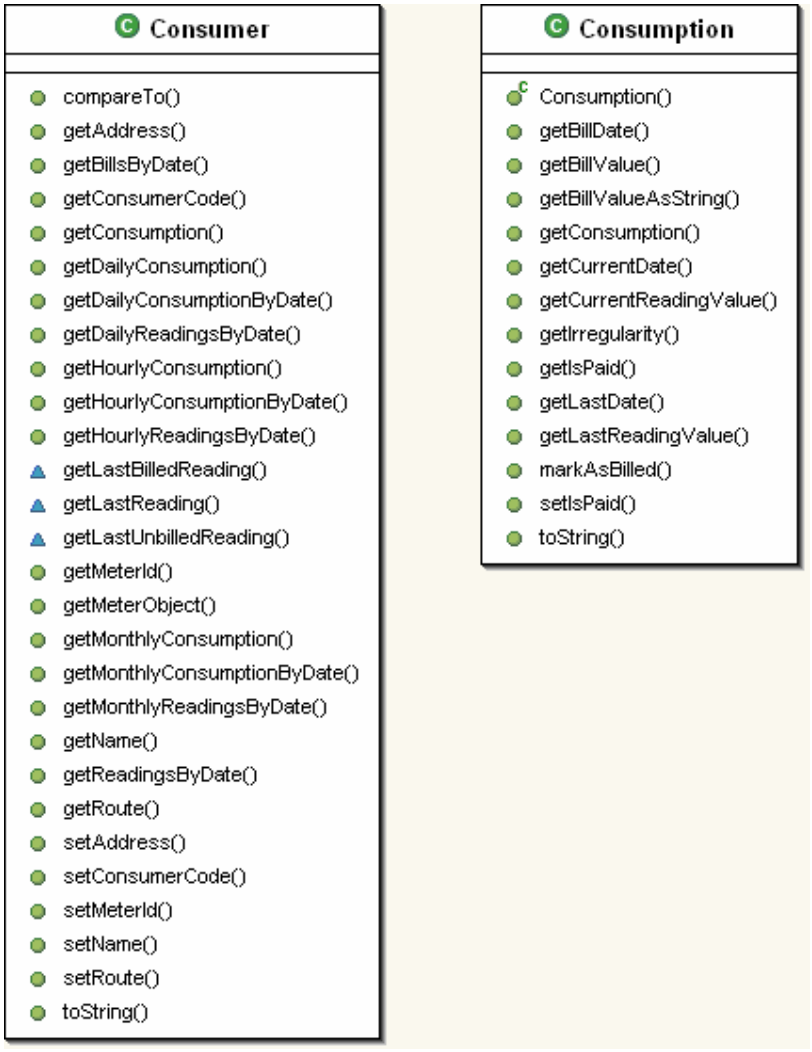

Figura 14 - Classes que contêm informações de leituras

6.2 Planos de Iteração

Nesta subseção mostramos os planos de iteração desenvolvidos durante o estágio. A notação Task X.Y denota o seguinte: Tarefa Y da *User Story* X.

Plano de iteração: 19/02/2003-12/03/2003

Task	Tempo	Responsável
Task 7.1	2	Gustavo
Task 4.1	3	Gustavo
Task 5.1	5	Gustavo
Task 12.1	4	Gustavo
Task 12.2	4	Gustavo
Task 15.1	10	Gustavo
Task 15.2	8	Gustavo

Task 11.1	8	Hugo
Task 14.3	7	Hugo

Plano de iteração: 05/02/2003-19/02/2003

Task	**Tempo**	**Responsável**
Task 7.1	2	Gustavo
Task 7.2	2	Gustavo
Task 4.1	8	Gustavo
Task 5.1	12	Gustavo
Task 12.1	6	Gustavo
Task 12.2	?	Gustavo
Task 8.1	3	Hugo
Task 9.1	3	Hugo
Task 2.1	2	Hugo
Task 11.1	8	Hugo
Task 14.3	8	Hugo

Plano de iteração: 22/01/2003-05/02/2003

Task	**Tempo**	**Responsável**
Task 6.4	8	Hugo
Task 7.1	16	Gustavo
Task 7.2	12	Gustavo
Task 1.1	4	Hugo
Task 1.2	(incluido acima)	Hugo
Task 1.3	(incluido acima)	Hugo
Task 2.1	4	Hugo
Task 14.1	2	Hugo
Task 14.2	4	Gustavo
Task 14.3	8	Hugo
T16.1	4	Hugo

Plano de iteração: 06/01/2003-22/01/2003

Task	**Tempo**	**Responsável**
Task 6.1	13	Gustavo
Task 6.2	4	Hugo
Task 6.3	4	Hugo
Task 6.4	13	Gustavo
Task 7.1	12	Gustavo
Task 1.1	20	Hugo
Task 1.2	5	Hugo
Task 1.3	5	Hugo

7. Considerações Finais

Vimos que o sistema SALAME é uma solução completa para os problemas de medição de energia elétrica das concessionárias CELB/SAELPA. Além disso, o fato das informações de consumo dos clientes poderem ser disponibilizadas na Internet é o diferencial do sistema.

Durante o estágio podemos colocar em prática várias teorias que foram contempladas em disciplinas da graduação. Em algumas disciplinas do curso de Ciência da Computação temos a obrigação de desenvolvermos sistemas para um cliente específico. Por termos desenvolvido um sistema para um cliente como a CELB/SAELPA tivemos uma experiência nova e muito positiva.

Dois itens são imprescindíveis em um trabalho de estágio: uso de uma metodologia e orientação de um professor experiente. O fato de não termos utilizado o processo XP1 com todas as suas atividades não invalidou a metodologia, mas trouxe alguns problemas durante o estágio. Entre eles podemos citar a falta de testes em alguns dos códigos que eram desenvolvidos, o que trazia medo aos desenvolvedores quando estes tentavam alterar o código. Outro problema que podemos citar foi quanto à não validação das *User Stories* via testes de aceitação junto ao cliente. Não tínhamos certeza se a funcionalidade acrescentada estava de acordo com o que o cliente queria.

Todos os trabalhos de pesquisa que desenvolvi durante o curso de graduação foram de grande importância para a minha formação profissional. Destaco principalmente os trabalhos desenvolvidos no Programa Especial de Treinamento (PET) [PET]. Mas o fato de ter continuado meus trabalhos no PET e ter feito o estágio ao mesmo tempo não foi uma boa experiência. Isso causou alguns transtornos durante o estágio, tais como entrega de funcionalidades do sistema fora do prazo estabelecido. Apesar de tudo isso, considero que este trabalho tenha sido de um grande aprendizado tanto em nível de pesquisa como em nível profissional.

8. Agradecimentos

Gostaria de agradecer especialmente ao professor Jacques por ter me dado a oportunidade de trabalhar neste projeto e por estar diretamente ligado à minha formação acadêmica. Vejo em Jacques a figura de um verdadeiro professor; Ao professor Peter por ter me ajudado em assuntos técnicos encontrados durante o estágio; Finalmente agradeço ao professor Rosales por ter proporcionado a nossa entrada no projeto e a apresentação dos nossos trabalhos na conferência [Metering, Billing and CRM/CIS].

Referências

Bibliográficas:

[GERREDCOM] – Lopes, Raquel V., Sauvé, Jacques P., Nicolleti, Pedro S. Melhores práticas para Gerência de Redes de Computadores. Campus, 2003

[SNMP] – Mauro, Douglas R., Schmidt, Kevin J. Essential SNMP. O´REILLY, 2001

Á sites:

[XP] - http://www.extremeprogramming.org (Acessada em 28/02/2004)

[XP1] - http://jacques.dsc.ufcg.edu.br/projetos/common/xp1/xp1.html (Acessada em 28/02/2004)

[Java] – http://java.sun.com (Acessada em 20/03/2004)

[Castor] - http://www.castor.org (Acessada em 12/03/2004)

[log4j] - http://jakarta.apache.org/log4j/docs (Acessada em 12/03/2004)

[CVS] - http://www.cvshome.org (Acessada em 12/03/2004)

[Continuous Integration] - http://www.martinfowler.com/articles/continuousIntegration.html (Acessada em 18/03/2004)

[Metering, Billing and CRM/CIS] - http://www.metering.com/events/mbr2003 (Acessada em 18/03/2004)

[PET] - http://www.dsc.ufcg.edu.br/~pet (Acessada em 18/03/2004)

Anexos

Anexo A : Artigo - Uso de SNMP para a Gerência de Sistemas de Medição de Energia Elétrica

Anexo B : Plano de Estágio

Anexo C : Manual de Instalação da Estação de Gerência