UFSM

TRABALHO DE GRADUAÇÃO

Modelagem de um Software de Monitoramento de Sinais Cardíacos

Acadêmico:
Cristiano Mazzutti

Orientador:
Raul Ceretta Nunes

Santa Maria, RS, Brasil

2006

Modelagem de um Software de Monitoramento de Sinais Cardíacos

Por

Cristiano Mazzutti

Trabalho de Graduação apresentado ao Curso de Graduação em Ciência da Computação – Bacharelado, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para
obtenção do grau de
Bacharel em Ciência da Computação

Curso de Ciência da Computação

Trabalho de Graduação nº 201
Santa Maria, RS, Brasil
2006

Universidade Federal de Santa Maria

Centro de Tecnologia

Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada, aprova o

Trabalho de Graduação

Modelagem de um Software de Monitoramento de Sinais Cardíacos

elaborado por

Cristiano Mazzutti

como requisito parcial para obtenção do grau de

Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA

_____

Raul Ceretta Nunes - UFSM
(Orientador)

_____

Oni Reasilvia Sichonany – UFSM

_____

Giovani Baratto – UFSM

Santa Maria 2006

Agradecimentos

Existem momentos na vida em que chegamos a uma espécie de encruzilhada, aí temos que decidir qual caminho a seguir, foi assim na escolha do curso, na troca de cidade, na escolha de um lugar para morar, e não é diferente agora que chegamos ao final do curso e partimos para o mercado de trabalho ou continuamos na carreira acadêmica. Não obstante, são também nestes momentos que percebemos a importância da ajuda daqueles que por muitas vezes apostam em nosso sucesso. Dessa forma agradeço primeiramente a Deus, pela oportunidade da escolha, a toda a minha família, especialmente a meus pais Roberto e Salete pelo carinho, confiança, pela logística e apoio irrestrito. A meu irmão Fabiano, que mesmo estando longe sempre me apoiou, em todos os momentos e a minhas "maninhas" Ketty e Kelly e ao meu irmão emprestado Marcos, pelo companheirismo, dedicação, amizade e pela convivência. Aos amores e desamores que de uma forma ou de outra, acabam por marcar e até mesmo alterar nossos caminhos.

Agradeço ainda, aos muitos colegas e amigos que fiz aqui, pelas festas, cumplicidade e pelo apoio. Aos professores e funcionários que tanto se empenham para garantir a qualidade do ensino, mesmo em tempos difíceis, e em especial ao meu orientador Raul pelas contribuições neste e nos muitos trabalhos desenvolvidos sob sua orientação.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Centro de Tecnologia
Universidade Federal de Santa Maria

Modelagem de um Software de Monitoramento de Sinais Cardíacos

Acadêmico:
Cristiano Mazzutti

Orientador:
Raul Ceretta Nunes

Este trabalho apresenta a modelagem de um sistema de monitoramento de sinais cardíacos. A modelagem se apóia na utilização de módulos funcionais independentes e em padrões de projetos para tornar flexível, tanto a inclusão de novos *drivers* para integração de diferentes tipos de dispositivos (ex: marca-passos e eletrocardiógrafos), como para acrescentar algoritmos analisadores de sinais. A flexibilidade também possibilita substituir componentes funcionais como, por exemplo, o responsável pela geração da onda cardíaca. Como resultado, têm-se o projeto de um software que possibilita visualizar e analisar curvas cardíacas a partir de sinais captados por dispositivos de monitoramento, viabilizando que o profissional de saúde realize acompanhamento e diagnóstico e que os profissionais de computação realizem sua expansão de forma facilitada.

1 INTRODUÇÃO

A necessidade do desenvolvimento de novas tecnologias constitui-se num constante desafio no meio acadêmico. Novas tecnologias alavancam o progresso, geram riquezas e ajudam a melhorar a qualidade de vida da população. Contudo, para que isso aconteça é preciso que essas tecnologias alcancem o maior número de pessoas possíveis, daí a importância da pesquisa ser disseminada por todas as regiões do país.

Empenhado neste sentido, o grupo de microeletrônica – GMICRO da Universidade Federal de Santa Maria – UFSM, vem pesquisando tecnologias ligadas à microeletrônica e, mais especificamente, ligadas aos equipamentos médicos cardiológicos. Exemplo disto é o desenvolvimento de *chips* para amplificação e controle de um marca-passo externo. Visando um sistema de instrumentação remota para o protótipo do marca-passo, o grupo, por meio do projeto TECIR – apoiado pelo CNPq, começou a investir no desenvolvimento de um sistema computadorizado de visualização de sinais cardíacos, o qual foi intitulado CardioMonitor (MAZZUTTI & NUNES, 2003).

A primeira versão do **CardioMonitor** é um sistema que atende as necessidades básicas de captação de sinais e geração da curva de batimentos cardíacos. No desenvolvimento do software, para maior flexibilidade e agilidade adotou-se a metodologia de programação extrema (SOARES, 2003), a qual baseia-se em características ligadas ao cliente e ao produto, tais como: simplicidade, flexibilidade e objetividade. Naquele trabalho foi adotada uma metodologia orientada ao cliente e à programação, sendo a prática de desenvolvimento dirigida por testes e por submissão dos resultados à análise do cliente. A aplicação da metodologia funcionou, no que diz respeito a velocidade de implementação do protótipo, mas infelizmente o projeto do CardioMonitor não atendeu adequadamente fases importantes de análise e projeto de software. Esta lacuna revelou problemas na manutenção do sistema, na inclusão de novas funcionalidades, na documentação (insuficiente) e no projeto das estruturas internas (projetos não padronizados).

Visando dotar o software de uma modelagem e implementação mais profissional, o presente trabalho re-visita os conceitos utilizados no CardioMonitor, aplicando agora os preceitos de engenharia de software na busca de um projeto flexível e modular que facilite a inclusão de novas funcionalidades e que permita a substituição de algoritmos.

Neste novo trabalho, assumi-se como mais importante a modelagem do sistema, tendo como meta um projeto modular, de fácil manutenção e ampliação (inclusão de novos módulos) e que reutilize padrões de projeto consagrados.

O restante do texto está organizado como segue. No capítulo 2 são apresentados os conceitos básicos ligados ao entendimento do projeto. O capítulo 3 mostra todos os aspectos relacionados a modelagem do aplicativo, trazendo os casos de uso, as especificações e o diagrama de classes. No capítulo 4 são abordados aspectos da implementação e testes do sistema. Por fim, o capítulo 5 apresenta as conclusões e aponta alguns possíveis e desejáveis trabalhos futuros.

2 CONCEITOS BÁSICOS

Este capítulo revisa conceitos que são utilizados no projeto da aplicação de monitoramento. Eles ajudam no entendimento das funcionalidades e compreensão do próprio projeto. A seção 2.1 apresenta um estudo do eletrocardiograma; a seção 2.2 demonstra o projeto da primeira versão do **CardioMonitor**; a seção 2.3 ajuda a compreender a nova forma de modelagem, utilizando orientação à objetos e UML; a seção 2.4 traz uma análise acerca dos padrões de projetos e por fim a seção 2.5 conclui o capítulo.

2.1 Estudo do Eletrocardiograma

O estudo dos princípios básicos do eletrocardiograma permite uma melhor compreensão dos conceitos relacionados à geração de curvas de batimentos cardíacos. Nele são apresentados os tipos de eletrocardiogramas, a origem do fenômeno elétrico do coração e o conceito de **Derivação**.

2.1.1 Eletrocardiograma

O Eletrocardiograma - ECG é um registro gráfico da atividade elétrica do coração, sendo um dos exames mais comuns da prática cardiológica. É poderoso auxiliar em diagnósticos de diferentes cardiopatias e distúrbios cardiovasculares, imprescindíveis para a Medicina moderna. Se um paciente queixa-se de que seu coração está descompassado, ao ser submetido ao ECG será mostrado em tempo real e graficamente este descompasso, sua natureza e freqüência (RUIZ, 2002).

Existem três tipos de exames de ECG, a saber:

- **Simples ou de Repouso:** O mais utilizado na prática médica: consiste na aplicação de eletrodos (placas capazes de captar o fluxo elétrico) sobre o paciente, que deve encontrar-se deitado. É importante que o paciente esteja desprovido de objetos metálicos (como pulseiras, brincos, correntes, relógios, etc.) que podem causar eventuais interferências na captação do fluxo elétrico a ser medido.
- **ECG Dinâmico (HOLTER):** Consiste na utilização de um pequeno gravador conectado aos eletrodos. O paciente deverá portá-lo por vinte e quatro horas. Nesta modalidade de ECG toda a atividade elétrica do coração do paciente é monitorada e gravada para posterior análise. O paciente deve anotar em um pequeno diário suas atividades durante o período de gravação, para que seja feita a confrontação de seus

eventuais sintomas e/ou atividades com o desempenho de seu coração. É um exame muito útil na detecção de arritmias e processos obstrutivos em artérias coronárias.

- **ECG de Esforço (Testes Ergométricos):** Segue os mesmos princípios de aplicação de eletrodos, porém é realizado com o paciente em atividade física. Antigamente era realizado com o paciente subindo e descendo degraus de escada. Atualmente é realizado em uma esteira ergométrica ou em bicicleta ergométrica. O objetivo é detectar possíveis alterações provocadas pelo aumento da solicitação cardíaca. Este exame destacou-se também por ser capaz de avaliar a pressão arterial durante o exercício, tornando esta modalidade de ECG uma das mais completas avaliações do desempenho cardiovascular (RUIZ, 2002)

2.1.2 As derivações

Segundo Enéas F. Carneiro (CARNEIRO, 1997), o eletrocardiógrafo é, em última análise, um galvanômetro que recolhe, a partir de dois eletrodos dispostos em determinados pontos do corpo humano, as diferenças de potencial aí existentes decorrentes da atividade cardíaca e chama-se **derivação** à linha que une esses dois eletrodos.

Derivações do Plano Frontal:

A utilização de eletrodos no ombro esquerdo, no ombro direito e na perna esquerda representa o chamado triângulo de Einthoven e as linhas imaginárias, as derivações clássicas D1, D2 e D3.

O terminal central de Wilson utiliza um eletródio dito explorador, nos mesmos pontos citados e um eletródio dito indiferente em um ponto distante daqueles, originando assim, as derivações clássicas aVR, aVL e aVF. A figura 2.1 traz um exemplo das derivações clássicas.

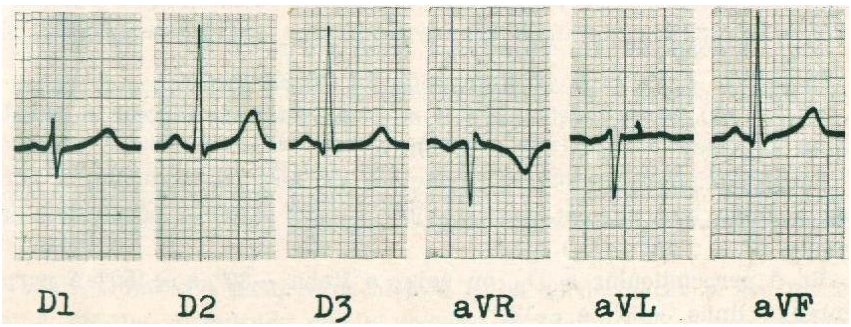

Figura 2.1 – Derivações Clássicas no Plano Frontal

Derivações do Plano Horizontal:

Além das seis derivações anteriormente citadas, na prática são registradas outras seis derivações unipolares que exploram o fenômeno elétrico a partir da face anterior do tórax e são denominadas "precordiais". São elas V1, V2, V3, V4, V5 e V6. Veja na figura 2.2 os pontos de aplicação dos eletrodos.

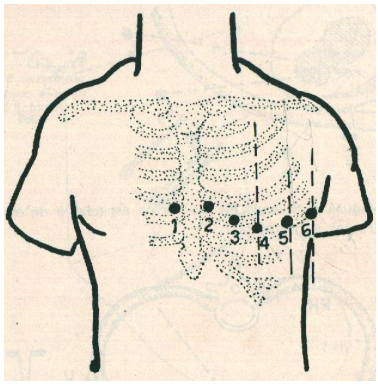

Figura 2.2 – Pontos de Aplicação dos Eletrodos: V1 a V6.

O estudo do eletrocardiograma e suas derivações, e analogamente o marca-passo, que pode ser entendido como uma derivação, para efeitos da geração da curva de sinais cardíacos, permite observar que um monitor cardíaco adequado, deve no mínimo, ser capaz de gerar um gráfico nítido o suficiente para que o profissional de saúde possa analisar o grau de inclinação do complexo QRS, como o da figura 2.3. Também permite perceber a necessidade da geração de mais de uma curva concomitantemente, para o caso das várias derivações do ECG.

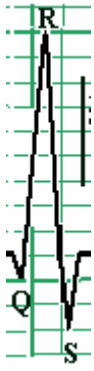

Figura 2.3 – Complexo de Ativação Ventricular

2.2 O CardioMonitor

O **CardioMonitor** (MAZZUTTI & NUNES, 2003) é uma ferramenta de software cujo objetivo principal é oferecer um mecanismo de visualização em tempo real de sinais cardíacos captados por um marca-passo externo. Sua arquitetura consiste dos seguintes blocos:

- **Conexão Serial:** neste bloco são tratadas as funções ligadas a comunicação com o meio físico, através da interface serial RS-232 ou das interfaces seriais emuladas pelo *driver* do dispositivo *Bluetooth* – tecnologia criada no início dos anos 90, pela Ericsson para transmissão de dados baseada em rádio freqüência, operando na faixa de 2.4GHz à 2.483GHz.
- **Configuração de Dispositivo:** providencia uma interface para alterar os parâmetros do dispositivo.
- **Captura dos Dados:** responsável pela captação dos sinais.
- **Tratamento do Sinal:** centraliza o módulo operacional da aplicação, formatando e distribuindo o sinal para os módulos de visualização, armazenamento, envio de emails e geração de alarmes.
- **Armazenamento dos Dados:** providencia uma forma simples de armazenamento dos sinais coletados utilizando objetos `java.lang.StringBuffer` e provê meios de acesso aos mesmos.
- **Geração de Sons e Alarmes:** utiliza o pacote `javax.sound.midi` para gerar um som de alarme que é utilizado pela aplicação em mensagens de alerta.
- **Envio de E-mails: e**sta unidade providencia uma interface de comunicação capaz de mandar mensagens de e-mails utilizando qualquer serviço SMTP (Simple Mail Transfer Protocol) habilitado.
- **Visualização da Curva de Batimentos:** este módulo implementa uma interface de visualização dos sinais cardíacos sob a forma de um gráfico de linhas contínua.
- **Gerenciamento de Paciente:** trata da inclusão, edição e exclusão de dados dos pacientes. Por motivos de simplicidade foram utilizadas apenas informações básicas, tais como, o nome do paciente, o número do prontuário, a data do último exame, o leito, o médico responsável e um e-mail para envio de alerta remoto.

A figura 2.4 apresenta o diagrama de blocos do CardioMonitor versão inicial, salientando as inter-relações entre eles.

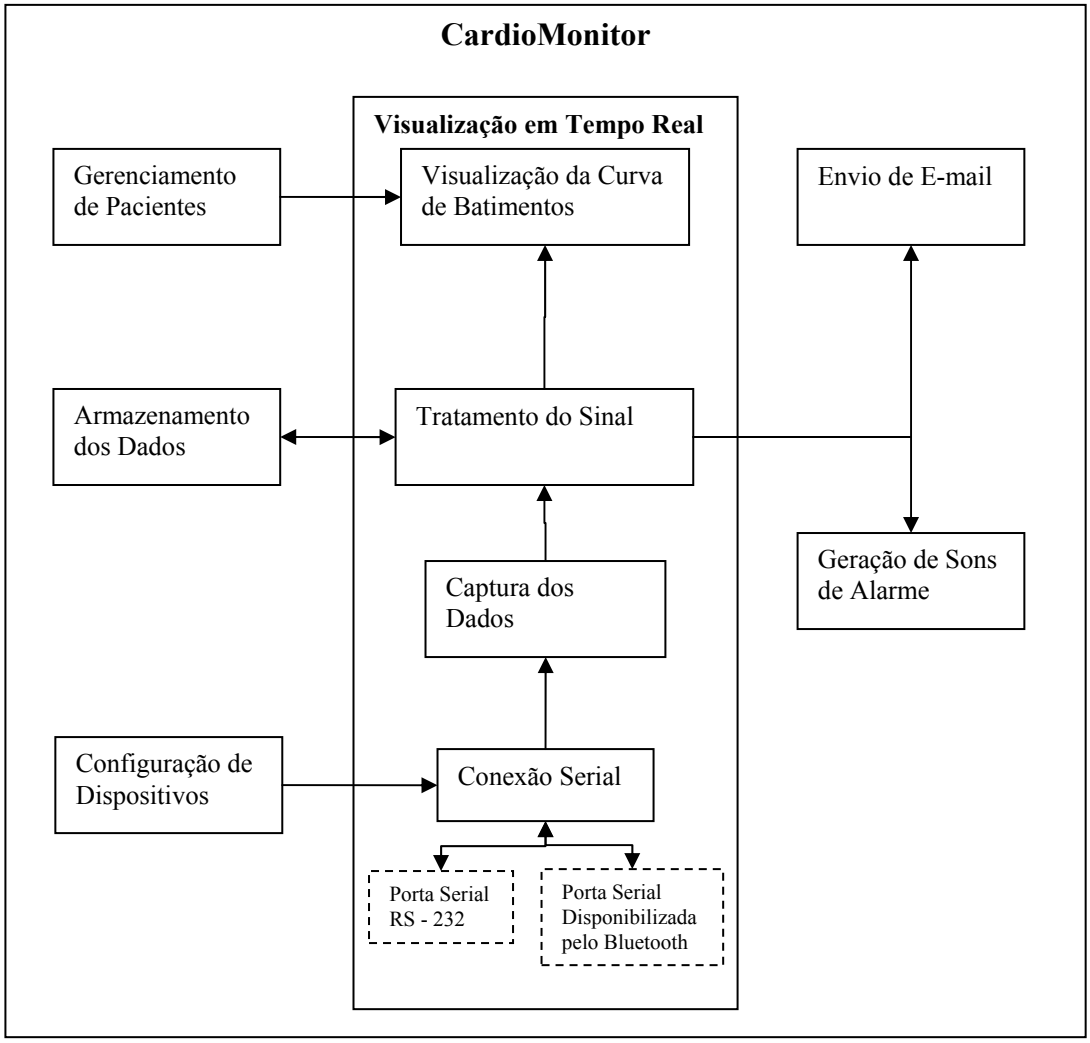

Figura 2.4 – Diagrama de Blocos do CardioMonitor

Nesta abordagem os módulos "fins", que recebem os sinais já formatados, não são considerados blocos funcionais, uma vez que não realizam operação sobre os dados (sinais). Eles simplesmente, dão vazão a dados já formatados, cada um desempenhando um serviço especializado. Esta visão facilita implementação das atividades fins, mas torna extremamente complexo o desenvolvimento e a manutenção do bloco responsável pelo tratamento dos sinais.

No projeto do CardioMonitor utilizou-se a metodologia de **Programação Extrema**, a qual baseou-se nos resultados e na submissão dos mesmos a análise do grupo de pesquisa, possibilitando a imediata correção, caso os rumos do desenvolvimento se afastassem do ideal. Esta abordagem permitiu uma evolução paralela entre o desenvolvimento do hardware e do software do projeto.

Buscando independência do sistema e um bom suporte de componentes gráficos, escolheu-se a linguagem de programação Java da *Sun Microsystems, Inc*. (JAVA HOME

PAGE, 2006) para implementar os softwares do sistema. Foram utilizados principalmente os pacotes gráficos AWT (*Abstract Window Toolkit*) e Swing na implementação da interface gráfica, a qual pode ser conferida na figura 2.5.

Figura 2.5 – Interface Principal do CardioMonitor

Os resultados do CardioMonitor foram satisfatórios, pois atenderam o propósito de ser um protótipo de um monitor cardíaco que possibilita a visualização local e remota da curva de sinais cardíacos captados e amplificados pelo marca-passo externo. Contudo, o projeto não previa uma interface modular que permitisse a fácil integração de novos módulos, ou mesmo a substituição de alguns componentes por uma versão melhorada. Como todas as operações estão centralizadas no bloco "Tratamento de Sinal" (vide figura 2.4), a alteração ou inclusão de componentes (módulos) ficava inviabilizada, pois isto só seria possível através de alterações profundas na estrutura do mesmo. Mesmo que as alterações fossem realizadas com cuidado, há sempre o risco das modificações interferirem em outras operações, já que todas são tratadas no mesmo bloco funcional. A concepção da primeira versão do software também não contempla a criação de *drivers* de dispositivos, o que força apenas uma forma de acesso aos dados/sinais. Tais problemas, dificultam a manutenção e ampliação do software, motivando a re-visita ao projeto do CardioMonitor.

2.3 Modelagem Orientada a Objetos e UML

Esta seção tem por objetivo revisar conceitos utilizados na modelagem da aplicação, apresentada no capítulo 3.

2.3.1 Modelagem Orientada a Objetos

Conforme (LARMAN, 1997), a essência da modelagem orientada a objetos está em considerar o domínio do problema e a solução lógica sob a perspectiva dos objetos (coisas, conceitos, ou entidades). A entidade fundamental deste paradigma é o objeto, que é projetado de forma a representar as características particulares de algum elemento do sistema em estudo.

O modelo de objetos possui várias vantagens para o desenvolvimento de software, incluindo:

- O mesmo modelo pode ser usado desde o início, quanto os conceitos são muito abstratos, até a implementação, quando são mais concretos.
- O modelo permite dar mais importância à estrutura de um sistema do que às funções. Isso é importante porque as funções que um sistema deve realizar mudam com o tempo (mesmo durante o desenvolvimento do sistema) enquanto são muito raras as mudanças da estrutura.

2.3.2 Definições do Modelo de Objetos

Seguem as definições formais das principais noções do modelo de objetos (ANQUETIL, 2002):

- **Classe:** A modelagem de um conceito do mundo real. As propriedades associadas ao conceito são representadas por atributos (variáveis) e operações (i.e. funções). Uma classe descreve os atributos que seus objetos vão ter e as funções que podem executar.
- **Objeto:** Qualquer entidade do mundo real pode ser representada por um objeto. Objetos são entidades independentes uns dos outros. Dois objetos com exatamente os mesmos atributos são dois objetos diferentes.
- **Instância:** Um objeto. A noção de instância é quase igual à de objeto. Instância se refere implicitamente a uma classe, é sempre uma instância de uma classe específica. Objetos podem ser considerados independentemente das suas classes. A noção de

instância é também mais geral. Qualquer conceito abstrato pode ter instâncias (aplicações concretas do conceito).

- **Atributo:** Uma propriedade importante de uma classe que pode ser representada como uma variável.
- **Método:** Uma propriedade importante de uma classe que pode ser representada com uma função.
- **Operação:** Um método mais abstrato. O método se refere implicitamente a uma função. A operação é mais abstrata, ela se refere à idéia do que a função faz, mas não se preocupa com os detalhes de implementação. Um método implementa uma operação.
- **Mensagem:** Se manda uma mensagem a um objeto para pedir a ele para executar uma operação particular.
- **Herança:** A modelagem da noção de especialização/generalização. No mundo real, conceitos são especializações uns dos outros. Os conceitos mais especiais têm todas as propriedades dos conceitos mais gerais, mais algumas propriedades em si próprio. No modelo a objetos, uma sub-classe possui todos os atributos e todas as operações da super-classe. A sub-classe pode acrescentar alguns atributos e métodos. Ela pode também redefinir alguns métodos da super-classe. Ela não pode tirar nenhuma propriedade da super-classe.
- **Hierarquia de herança:** Todas as relações de herança entre todas as classes formam uma árvore chamada de hierarquia de herança.
- **Polimorfismo:** Várias classes podem implementar a mesma operação de maneira diferente. A mesma operação (com o mesmo nome) tem várias ("poli") formas ("morfismo") em cada classe. Uma característica poderosa do modelo a objetos é que todas as formas de uma operação podem ter o mesmo nome. Assim se torna mais fácil de reconhecer qual operação um método particular implementa. Isso é possível porque cada objeto sabe qual é sua própria classe, e pode executar os métodos apropriados.
- **Interface:** Conjunto das propriedades da classe que pertencem ao conceito implementado pela classe. A interface permite proteger o exterior das mudanças dentro da classe. Enquanto a interface não muda, para o exterior, a classe não muda.

2.3.3 UML - *Unified Modeling Language*

A linguagem de modelagem unificada (UML) é um modelo de linguagem que derivou principalmente da unificação das linguagens de modelagem de três métodos (LARMAN, 1998): o "OMT"' de Rumbauch, o "método de Booch" e os "casos de uso" de Jacobson. Cada um foi bem sucedido como método de análise orientada a objeto, e o objetivo da UML é tirar vantagem das principais características de cada um deles.

Ela formaliza os elementos para se descrever o modelo de objetos. Utilizando-se de vários diagramas, fornece uma maneira clara de representar os aspectos do mundo real sob a forma de modelos. Segue-se os principais diagramas UML:

- **Diagrama de Casos de Uso:** É um diagrama usado para se identificar como o sistema se comporta em várias situações que podem ocorrer durante sua operação. Descreve o sistema, seu ambiente e a relação entre os dois. Os componentes deste diagrama são os **atores** e os **casos de usos**, os quais podem ser conferidos na figura 2.6.
    - **Ator:** Representa qualquer entidade que interage com o sistema. Pode ser uma pessoa, outro sistema, etc. Algumas características do ator:
        - Não é parte do sistema. Representa os papéis que o usuário do sistema pode desempenhar.
        - Pode interagir ativamente com o sistema.
        - Pode ser um receptor passivo de informação.
        - Pode representar um ser humano, uma máquina ou outro sistema.

    - **Casos de Uso:** é uma seqüência de ações que o sistema executa e produz um resultado de valor para o ator. Segue algumas características:
        - modela o diálogo entre atores e o sistema.
        - é iniciado por um ator para invocar uma certa funcionalidade do sistema.
        - é um fluxo de eventos completo e consistente.

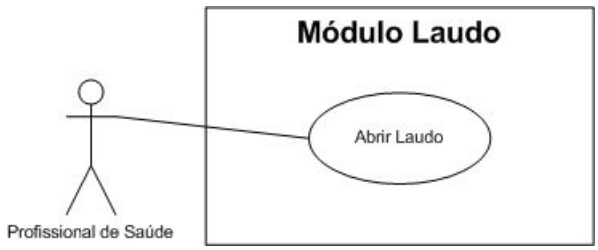

Figura 2.6 – Exemplo de Diagrama de Caso de Uso

- **Diagrama de Seqüência:** mostra a interação entre os Objetos ao longo do tempo. Apresentando os objetos que participam da interação e a seqüência de mensagens trocadas. A notação usada pela UML para representar o Diagrama de Seqüência, é a seguinte:
    - Objetos são representados por retângulo com seus nomes sublinhados.
    - As linhas de vida dos Objetos são representadas por linhas verticais tracejadas.
    - As interações entre Objetos são indicadas por flechas horizontais que são direcionadas da linha vertical que representa o Objeto cliente para a linha que representa o Objeto fornecedor.
    - As flechas horizontais são rotuladas com as mensagens.
    - A ordem das mensagens no tempo é indicada pela posição vertical, com a primeira mensagem aparecendo no topo.
    - A numeração é opcional e baseada na posição vertical.

- **Diagrama de Colaboração:** É um modo alternativo para representar a troca de mensagens entre um conjunto de Objetos. O Diagrama de Colaboração mostra a interação organizada em torno dos Objetos e suas ligações uns com os outros. A notação usada é a seguinte:
    - Objetos são representados por retângulo com seus nomes sublinhados.
    - As interações entre Objetos são indicadas por uma linha conectando-os.
    - As ligações indicam a existência de um caminho para comunicação entre os Objetos conectados.
    - As ligações no Diagrama de Colaboração podem ser apresentadas por:
        - Flechas apontando do Objeto cliente para o Objeto fornecedor.
        - O nome da mensagem.
        - Numeração seqüencial, mostrando a ordem relativa de envio das mensagens.

- **Diagrama de classes:** descreve as classes que formam a estrutura do sistema e suas relações. As relações entre as classes podem ser associações, agregações ou heranças.
    - Classes são representadas por retângulo com seus nomes.
    - Seus atributos e métodos podem ser listados em retângulos adjacentes e abaixo da classe.

2.4 Padrões de Projeto

Na busca pela eficiência na produção de softwares, programadores experientes fazem refinamentos sucessivos em seus códigos tornando-os cada vez mais legíveis, claros e eficientes. Neste empenho, eles acabam por chegar a soluções otimizadas que se parecem muito com soluções de outros colegas, em se tratando de um mesmo problema. Estas soluções similares, acabam se tornando a maneira mais aceita de se realizar a tarefa, constituindo daí por diante um novo padrão.

Os padrões resolvem problemas específicos e tornam os projetos orientados a objetos mais flexíveis e reutilizáveis. Eles ajudam os projetistas a reutilizar projetos bem-sucedidos ao basear os novos projetos na experiência anterior. Desta forma, um projeto orientado a objeto, não deve partir do zero, reinventando a roda, deve sim, exaurir as fontes bibliográficas na busca da experiência acumulada pelos desenvolvedores que o precederam. Este trabalho se detém no estudo dos padrões de projeto utilizados no livro *Padrões de Projetos em Java* (METSKER, 2004), o qual analisou e adaptou para a linguagem Java os padrões descritos na consagrada obra *Padrões de Projeto* (GAMMA *et al.* 1995).

2.4.1 Definição de Padrões

De acordo com (FERREIRA, 1977), um padrão é definido com sendo: 1. Modelo oficial. 2. Qualquer objeto que serve de modelo à feitura de outro. Em outras palavras, um padrão é a consagração de uma maneira de se fazer alguma coisa. Dentre os diferentes modos de se fazer algo, um geralmente se destaca, seja pela sua simplicidade, pela praticidade ou até mesmo pela simples repetição no seu uso, tornando-se um padrão, ou seja, o modo mais aceito de se realizar a tarefa pretendida.

2.4.2 Definição de Padrões de Projeto

"Cada padrão descreve um problema que ocorre outra e mais outra vez no nosso ambiente, e então descreve o âmago da solução deste problema, de forma que você possa usar esta solução um milhão de vezes, sem fazer o mesmo duas vezes." (ALEXANDER, et al, 1977)

"Um padrão de projeto sistematicamente nomeia, motiva e explica um projeto genérico, que endereça um problema de projeto recorrente em sistemas orientados a objetos. Ele descreve o problema, a solução, quando é aplicável e quais as conseqüências de seu uso." (GAMMA, et al, 1995)

"Um padrão de projeto é um projeto – uma maneira de alcançar um objetivo – o que utiliza classes e seus métodos em uma linguagem orientada a objetos." (METSKER, 2004)

2.4.3 Organização dos Padrões de Projeto

Existem diversas maneiras de se classificar padrões, por semelhança de estrutura, seguindo a ordem de bibliografias consagradas como *Padrões de Projeto* ou agrupando-os de acordo com seus objetivos. A classificação adotada em (METSKER, 2004) agrupa-os de acordo com seus objetivos em cinco categorias, a saber:

- **Interfaces:** declaram o conjunto de métodos que uma classe implementa, implicando desta forma, que a mesma deverá fornecer os serviços que os nomes dos métodos sugerem.
- **Responsabilidade:** padrões que levam o desenvolvedor a garantir que os métodos executem os serviços sugeridos pelos seus nomes e utilizam adequadamente a visibilidade para limitar a responsabilidade dos mesmos.
- **Construção:** padrões orientados para construção são projetos que permitem ao cliente construir um novo objeto mediante outros meios que não impliquem chamar um construtor de classes.
- **Operações:** tratam de contextos em que são necessários mais de um método, geralmente com a mesma assinatura, para participar de um projeto.
- **Extensões:** tratam de contextos nos quais é preciso acrescentar comportamento específico para uma coleção de objetos ou acrescentar novos comportamentos a um objeto sem alterar a sua classe.

2.4.4 Descrição dos Padrões

Os padrões devem, idealmente, ter nomes sugestivos que contribuam para o seu entendimento e memorização, tornando-se assim, jargões que facilitam a comunicação entre os especialistas da área. Todo o projeto de software que pretenda ser bem sucedido deve fazer uso de padrões de projeto consolidados, pois eles trazem a experiência acumulada de outros desenvolvedores que os definiram através do uso de soluções recorrentes para problemas semelhantes. A seguir são descritos os principais padrões utilizados neste projeto.

Adapter: este padrão pretende fornecer a interface desejada pelo cliente, usando os serviços de uma classe com uma interface diferente, convertendo a interface de uma classe em outra interface, esperada pelos que venham a utilizar o código. No projeto pretende-se utilizá-

lo para construir a interface de um **analisador genérico**, conforme ilustra a figura 2.7 visando fornecer um modo uniforme de acesso para a implementação do algoritmo de análise dos sinais cardíacos.

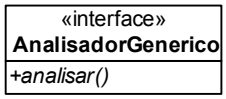

Figura 2.7 – Exemplo do Padrão Adapter

Bridge: separa a abstração, da implementação das operações em uma classe abstrata, permitindo que ambas variem independentemente. Evita que mudanças na implementação de uma abstração tenham impacto sobre o código dos clientes. Pretende-se utilizá-lo para facilitar a criação de *drivers* de dispositivos e também para facilitar a substituição dos algoritmos de plotagem dos gráficos. A figura 2.8 traz um exemplo deste padrão.

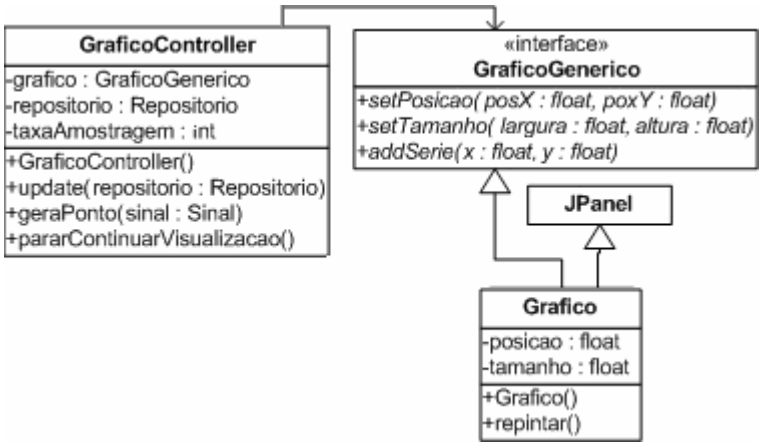

Figura 2.8 – Exemplo do Padrão Bridge

Observer: define uma dependência, de modo que quando um objeto mudar de estado, todos os seus dependentes sejam notificados e atualizados automaticamente. Pode ser usado para impulsionar a geração dos gráficos, gravar sinais e todas as demais atividades dinâmicas do sistema.

Command: permite o encapsulamento de uma solicitação como um objeto de modo que este possa ser parametrizado. Usado em menus e botões de comando.

Iterator: fornece uma maneira de acessar os elementos de uma coleção sequencialmente.

2.5 Conclusões Parciais

Construir um sistema que auxilie a tarefa de monitoramento de sinais cardíacos foi o objetivo inicial do GMicro. Isto motivou estudos sobre eletrocardiogramas e o desenvolvimento de uma primeira versão de um sistema de monitoramento cardíaco.

A orientação a objetos é um paradigma que pode auxiliar a construção de softwares, assim como a programação extrema é uma importante metodologia de desenvolvimento, mas observou-se que isto não é suficiente para garantir um projeto flexível e versátil. É necessário uma boa análise e projeto.

Uma boa interface deve trazer elementos consistentes e integrados, dispostos de maneira agradável, sem componentes como botões e caixas de escolha, em demasia. Deve ser simples porém completa. No projeto inicial do CardioMonitor pode-se facilmente perceber o excesso de componentes e sua completa heterogeneidade, devendo estes aspectos ser re-trabalhados. Contudo, deve-se tomar cuidado para manter a homogeneidade dos componentes gráficos, utilizando por exemplo painéis de tabulação para integrar as diferentes formas de construção e análise das curvas de sinais cardíacos geradas.

Acredita-se que o uso de padrões de projeto confere ao trabalho a garantia de re-usabilidade dos componentes desenvolvidos. Os padrões de projetos representam um importante ferramental que, apoiado na experiência acumulada de outros desenvolvedores, garantem uma solução profissional para o problema, conferindo também clareza de implementação, eficiência na geração de código e legibilidade ao sistema.

3 MODELAGEM DO SISTEMA

Tendo como meta o re-projeto de um sistema de monitoramento cardíaco modular, expansível e flexível, este capítulo descreve a modelagem dos elementos constantes no sistema e suas relações estáticas, através do diagrama de classes, e dinâmicas, através dos diagramas de seqüência. Desta forma, o capítulo garante uma ampla cobertura das características do sistema, além de demonstrar o fluxo de informações no mesmo.

O uso da modelagem orientada à objetos confere credibilidade a modelagem da aplicação, visto que seus modelos já são consagrados e bem aceitos nos meios acadêmico e profissional. Já a linguagem UML empresta os diagramas necessários à geração e compreensão do modelo formal criado a partir da parcela do mundo real em estudo. Cabe salientar, que os diagramas deste capítulo foram desenhados utilizando-se a ferramenta de diagramação Microsoft Office Visio Professional 2003 (MS VISIO, 2003).

O capítulo está organizado como segue. A seção 3.1 mostra uma breve descrição dos conceitos básicos e apresenta os requisitos do sistema na busca de uma modelagem precisa. A seção 3.2 mostra a modelagem arquitetural. Na seção 3.3 são apresentados os casos de uso. Já a seção 3.4 traz a especificação dos principais componentes, enquanto a seção 3.5 apresenta o diagrama de classes. Por fim a seção 3.6 conclui o capítulo.

3.1 Requisitos do Sistema

Para descrever os requisitos do sistema de monitoramento cardíaco necessita-se definir as principais entidades externas do sistema. A saber:

- **Paciente:** que representa a entidade da qual os dados (sinais) são coletados e a qual se pretende associar o diagnóstico ou acompanhamento;
- **Dispositivo:** aparelho que captura os sinais, gerando os dados de entrada no sistema. Pode ser um marca-passo ou um eletrocardiógrafo; e
- **Sinal:** pulso elétrico proveniente da atividade cardíaca, o qual pode ser capturado, amplificado e digitalizado. Participa do sistema de forma ativa sendo a análise e visualização de seu conjunto o objetivo maior do sistema de monitoramento.

Neste trabalho, tais entidades devem ser manipuladas de forma que:

- o sistema seja modular, possibilitando a inclusão de novas funcionalidades sem afetar o comportamento daquelas já incorporadas;
- o sistema contemple os casos de uso levantados;
- o sistema contenha uma interface para o armazenamento dos dados cadastrais dos pacientes;
- a interface de visualização suporte a troca dos algoritmos de geração das curvas e de desenho;
- a sistema seja independente de plataforma;
- os sinais sejam armazenados para visualização e análise futura;.
- o sistema suporte visualizações de diversos dispositivos; e
- a implementação use padrões de projeto.

Fluxo da Entrada de Sinais

O sistema de monitoramento deverá ser capaz de implementar o fluxo de sinais mostrado na figura 3.1 e descrito a seguir:

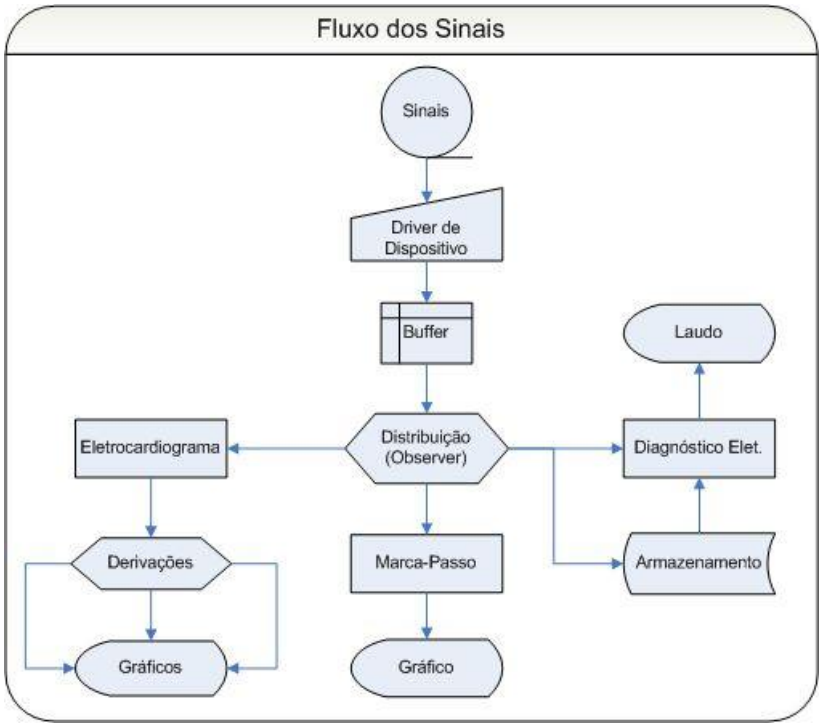

Figura 3.1 – Fluxograma dos Sinais

Os sinais cardíacos são obtidos através do uso de eletrodos colocados nos pacientes. Uma vez captados, os mesmos são digitalizados, filtrados e amplificados dando origem a um sinal digital mensurável.

Eles chegam ao sistema já digitalizados e podem ser captados pelas mais diferentes formas. Assume-se que é responsabilidade do projetista de hardware a implementação do *driver* do dispositivo capaz de entender os modos de transmissão do meio físico utilizado para a comunicação. Por exemplo, se o dispositivo for um marca-passo externo ligado via porta RS-232 (serial padrão), o projetista de hardware deverá criar um *driver* de dispositivo que consiga ler dados recebidos por ela e entregar ao CardioMonitor no formato pré-determinado. Cabe salientar que o uso do padrão de projeto Bridge pode facilitar a realização desta tarefa por parte do projetista.

Após serem obtidos, os sinais são armazenados em um buffer temporário, onde são acumulados até que se atinja certo número, quando então é disparado um evento avisando ao restante da aplicação que os dados já estão disponíveis e podem ser utilizados. A partir do aviso de dados disponíveis, cada parte da aplicação lê os sinais do buffer e realiza a operação sobre os mesmo. No caso da parte de geração do gráfico, a curva cardíaca é então desenhada, já no caso do gerador de diagnóstico os sinais são analisados para gerarem um laudo e assim por diante.

Outra característica importante proveniente do conceito de sinais, é a taxa de amostragem. Como cada dispositivo possui uma taxa de transmissão diferente, a mesma deverá ser associada a cada dispositivo especifico e gravada junto com ele, pois só desta forma pode-se calcular o tempo de cada sinal, pulso ou mesmo pontos da curva cardíaca, a qualquer tempo.

3.2 Modularização Arquitetural

O novo projeto do software de monitoramento CardioMonitor é composto por módulos específicos para cada funcionalidade, sendo que cada um é responsável pela operação a que se propõe realizar. Esta forma de modelagem, utilizando módulo funcionais, favorece a inclusão de novas funcionalidades ou mesmo a alteração das já existentes. O diagrama da figura 3.2 apresenta estes módulos, sendo os mesmos descritos a seguir:

- **Módulo Seleção:** trata operações de seleção do paciente a monitorar e de seleção do módulo de visualização pretendido. Além disto também trata funções relacionadas ao estilo da interface gráfica utilizada.
- **Módulo Marca-Passo:** é o responsável pela geração da curva de batimentos a partir de sinais provenientes de um dispositivo marca-passo.

- **Módulo Eletrocardiograma:** trata de funções ligadas aos sinais coletados de um dispositivo eletrocardiograma, levando em consideração as derivações dele provenientes.
- **Módulo Diagnóstico Eletrônico:** funções de apoio à geração de diagnósticos eletrônicos.
- **Módulo Laudo:** providencia uma forma de visualização de laudos.
- **Módulo Gerenciamento de Paciente e Dispositivo:** responsável pelas operações de cadastro prévios, tanto dos pacientes como dispositivos de monitoramento.

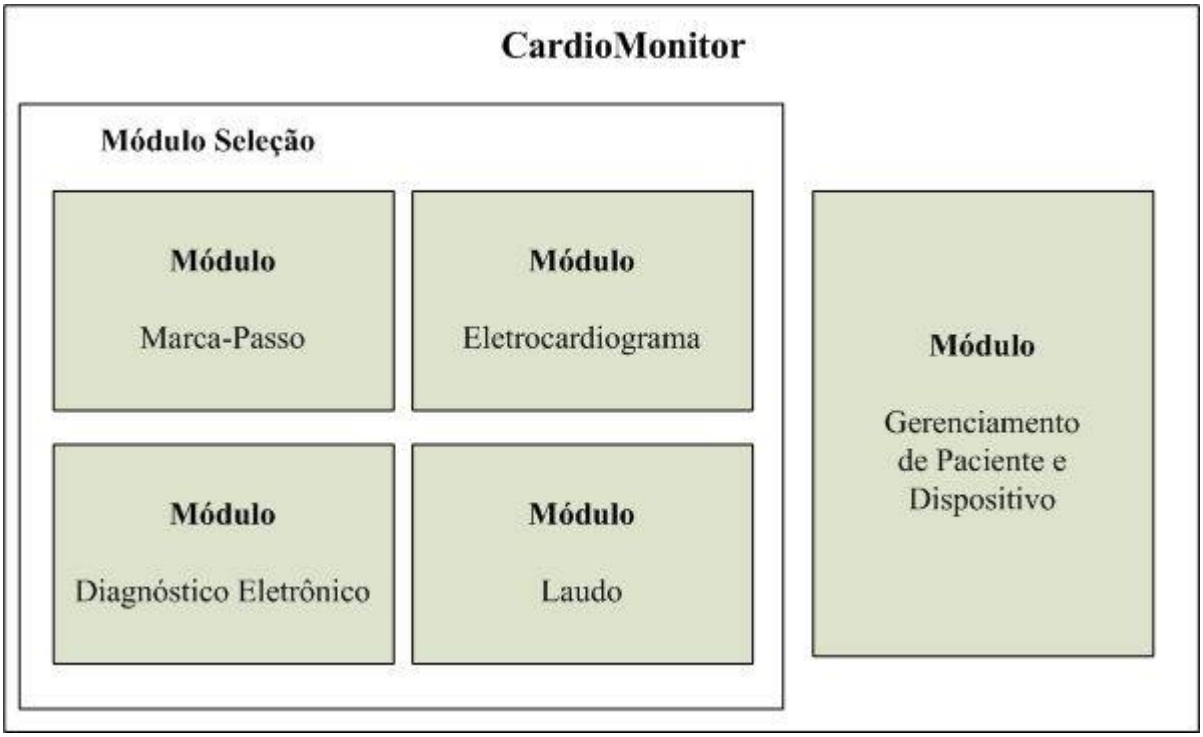

Figura 3.2 – Módulos do CardioMonitor

Observe que os módulos operacionais mostrados na figura 3.2, são independentes entre si. A única exceção é o **módulo seleção**, do qual dependem todos os módulos ligados a funções fins, ou seja, aquelas cujas operações dependam da prévia escolha do paciente.

Os blocos funcionais mostrados na figura 3.3 caracterizam a arquitetura do sistema e ajudam na compreensão da nova forma de modelagem. Eles são descritos a seguir:

- **Dispositivo:** trata as funções de cadastro, modificação, exclusão e parametrização de dispositivos.
- **Driver de Dispositivo:** implementa as funções ligadas à recepção de sinais.
- **Buffer e Distribuição:** acumula e distribui os sinais para os módulos operacionais.
- **Blocos Operacionais:** responsáveis pelo tratamento dos sinais e realização de operações específicas, tais como: geração de curvas de cardíacas, análise de sinais, etc.
- **Paciente:** trata as funções de manutenção de paciente.

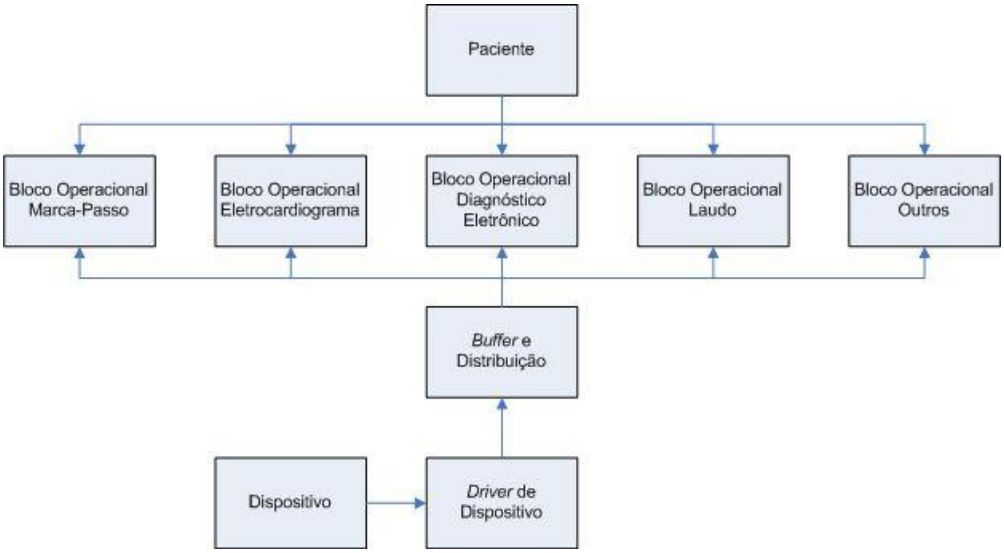

Figura 3.3 – Arquitetura do Sistema

Uma análise mais detalhada dos blocos funcionais, listados em paralelo no diagrama da figura 3.3, permite constatar a facilidade de inclusão de novas funcionalidades. Uma vez que cada bloco implementa sua operação independentemente dos demais, basta adicionar um novo bloco operacional para incluir uma nova operação. A modificação de funcionalidades segue o mesmo princípio de independência.

3.3 Casos de Uso

Nesta seção, os casos de uso do sistema são identificados, descritos e mostrados na forma de caso de uso estendido e diagramas de casos de uso.

3.3.1 Casos de Uso – Módulo Seleção

No módulo de seleção, são tratadas as funções que irão dar suporte à implementação dos módulos específicos para a visualização da curva do marca-passo, do ECG, diagnóstico e laudo. Tais funções correspondem a casos de uso gerais do sistema, a saber:

- Selecionar Paciente;
- Selecionar Estilo da Interface Gráfica; e
- Selecionar Módulo de Visualização.

A seguir, cada caso de uso é descrito na forma de caso de uso expandido. A figura 3.4 presenta os casos de uso do módulo na forma de diagrama de caso de uso. Observa-se que os casos de uso deste módulo são independentes entre si.

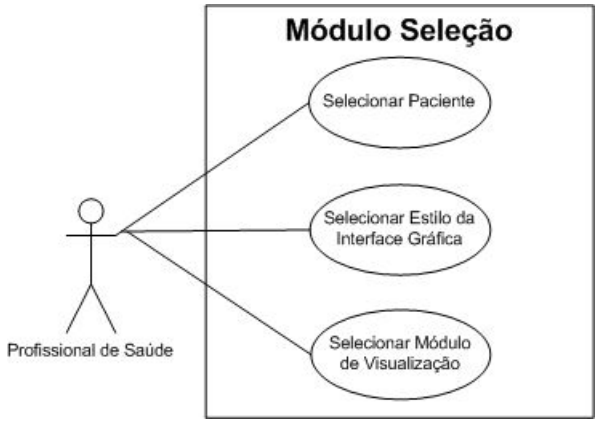

Figura 3.4 – Diagrama de caso de uso – Módulo Seleção

3.3.1.1 Selecionar Paciente

A base das funcionalidades do sistema está ligada a escolha do paciente, pois dele serão coletados os sinais cardíacos que formarão os gráficos do marca-passo e do ECG, bem como a análise dos sinais que darão origem ao diagnóstico eletrônico. O profissional de saúde deve selecionar um paciente na lista de pacientes, antes de proceder qualquer atividade de monitoramento.

Atores:	Profissional de saúde
Propósito:	Selecionar o paciente a ser monitorado.
Descrição:	O profissional de saúde seleciona o paciente a ser monitorado, por seu nome ou código. Vários pacientes podem ser acompanhados simultaneamente.
Tipo:	Principal e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde chega para monitorar um paciente.	
2. O profissional de saúde inicia a aplicação.	3. Mostra lista com nome e código de pacientes cadastrados.
4. O profissional de saúde seleciona o paciente.	5. Mostra a tela de monitoramento.

3.3.1.2 Selecionar Estilo da Interface Gráfica

O profissional de saúde deve poder selecionar o estilo da interface gráfica que deseja visualizar na aplicação.

Atores:	Profissional de saúde.
Propósito:	Permitir que o profissional de saúde configure o estilo de interface gráfica com o qual está mais acostumado a trabalhar.
Descrição:	O profissional de saúde escolhe se quer trabalhar com uma interface estilo Windows, Motif ou estilo padrão Java Swing.
Tipo:	Secundário e opcional.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde seleciona o menu **Aparência**.	2. Mostra os sub-menus com os estilos disponíveis.
3. O profissional de saúde seleciona o estilo desejado.	4. Re-configura a interface

3.3.1.3 Selecionar Módulo de Operação

O profissional de saúde deve poder selecionar o módulo de operação que deseja trabalhar (Marca-Passo, ECG, Diagnóstico ou Laudo).

Atores:	Profissional de saúde.
Propósito:	Escolher o tipo de operação.
Descrição:	O profissional de saúde escolhe se deseja operar marca-passo, ECG, diagnóstico ou laudo.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde seleciona o paciente a monitorar.	2. Mostra o módulo marca-passo, por *default*.

3. O profissional de saúde seleciona o módulo desejado.	4. Re-configura a interface para refletir o novo módulo.

3.3.2 Casos de Uso – Módulo Marca-Passo

Este módulo tem a finalidade de interagir com um dispositivo tipo marca-passo e gerar gráficos de batimentos cardíacos adequadamente. Suas principais funções são: visualização simples dos sinais cardíacos; parar/continuar a visualização; gravar/parar gravação dos sinais; e conectar/desconectar de dispositivos marca-passo.

O diagrama dos casos de uso do módulo Marca-Passo é mostrado na figura 3.5. Nele podem ser observados os casos de usos e os atores de cada iteração.

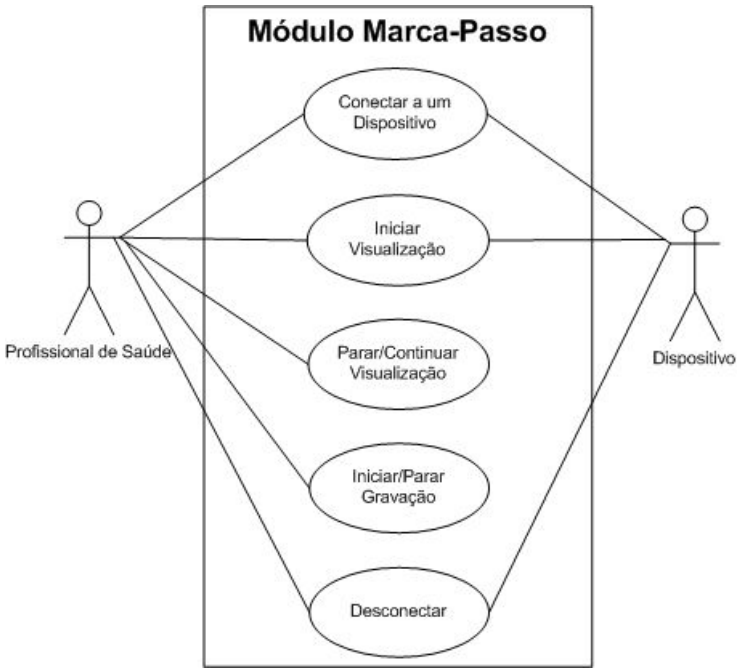

Figura 3.5 – Diagrama de caso do uso – Módulo Marca-Passo

3.3.2.1 Conectar a um Dispositivo

O profissional de saúde abre uma conexão com o dispositivo ao qual o paciente está ligado. Para isso ele deve escolher entre os dispositivos preparados, qual deseja conectar.

Atores:	Profissional de saúde, Dispositivo.
Propósito:	Abrir uma conexão entre o objeto paciente e um dispositivo previamente preparado.

Descrição:	O profissional de saúde escolhe o dispositivo ao qual o paciente está fisicamente ligado e do qual serão captados os dados.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **conectar**.	2. Apresenta uma lista com os dispositivos já preparados e associados ao paciente selecionado.
3. O profissional de saúde seleciona um dispositivo.	4. Requisita uma conexão ao *driver* do dispositivo.
5. O dispositivo abre uma conexão com o hardware.	6. Mostra mensagem de sucesso ou erro.

3.3.2.2 Iniciar Visualização

O profissional de saúde deve habilitar a captação dos sinais provenientes do dispositivo, ativando o fluxo de sinais no sistema e iniciando assim a geração do gráfico.

Atores:	Profissional de saúde, Dispositivo.
Propósito:	Visualizar gráfico de batimentos cardíacos.
Descrição:	O profissional de saúde inicia o procedimento para a visualização da curva de batimentos cardíacos.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **iniciar**.	2. Providencia a captação dos dados.
3. O dispositivo envia os dados.	4. Gera gráfico de batimentos cardíacos.
5. O profissional de saúde visualiza o gráfico.	

3.3.2.3 Parar Visualização

O fluxo dos sinais no sistema deve ser interrompido pelo profissional de saúde, quando este deseja fazer uma avaliação mais precisa de um determinado ponto do gráfico.

Atores:	Profissional de saúde.
Propósito:	Parar a geração do gráfico para uma análise mais detalhada.
Descrição:	O profissional de saúde pára a geração do gráfico em um ponto, afim de realizar uma análise mais detalhada.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **parar**.	2. Interrompe a captação dos dados e mantém o gráfico estático.
3. O profissional de saúde realiza análise.	

3.3.2.4 Continuar Visualização

De modo inverso ao caso de uso anterior, uma vez interrompido, o fluxo de sinais deve poder ser reativado a qualquer tempo pelo profissional de saúde.

Atores:	Profissional de saúde.
Propósito:	Retomar a geração dinâmica do gráfico.
Descrição:	O profissional de saúde quer retomar a geração da curva de batimentos.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **continuar**.	2. Retoma a captação dos dados e gera o gráfico dinamicamente.
3. O profissional de saúde faz novo acompanhamento.	

3.3.2.5 Gravar Sinais

O profissional de saúde deve poder gravar os sinais em meio de armazenamento permanente, visando uma futura análise no módulo de diagnóstico eletrônico.

Atores:	Profissional de saúde.
Propósito:	Gravar sinais para a geração posterior do diagnóstico eletrônico.
Descrição:	Os sinais são armazenados em disco para a posterior geração do diagnóstico eletrônico.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **gravar**.	2. Gera um arquivo de saída e passa a armazenar nele os sinais coletados do marca-passo.

3.3.2.6 Parar Gravação dos Sinais

O profissional de saúde deve também poder interromper uma gravação de sinais em meio secundário (permanente).

Atores:	Profissional de saúde.
Propósito:	Interromper armazenamento secundário dos sinais cardíacos.
Descrição:	O profissional de saúde interrompe o armazenamento dos sinais coletados.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **interromper gravação**.	2. Fecha o arquivo de saída e continua apenas com a geração do gráfico.

3.3.2.7 Desconectar

O profissional de saúde fecha a conexão com o dispositivo interrompendo a captação dos sinais e finalizando o exame corrente.

Atores:	Profissional de saúde, Dispositivo.
Propósito:	Interromper a captação dos sinais e desconectar o dispositivo associado.
Descrição:	O profissional de saúde interrompe a captação dos dados, encerrando o monitoramento do paciente.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **desconectar**.	2. Interrompe a captação dos dados e desconecta o dispositivo associado.
3. O dispositivo encerra a conexão.	

3.3.3 Casos de Uso – Módulo Eletrocardiograma

O módulo eletrocardiograma, difere-se do módulo marca-passo por levar em conta a derivação escolhida para a captação dos sinais e por mostrar até três derivações simultaneamente. Por este motivo, além dos casos de uso responsáveis pela geração dos gráficos dos batimentos cardíacos, faz-se necessário identificar o caso de uso "Escolher Derivação". Segue os casos de uso do módulo eletrocardiograma, lembrando que os casos de uso do módulo marca-passo também são próprios do módulo eletrocardiograma e por isso não serão descritos novamente. No diagrama da figura 3.6 são demonstrados os casos de usos do módulo eletrocardiograma.

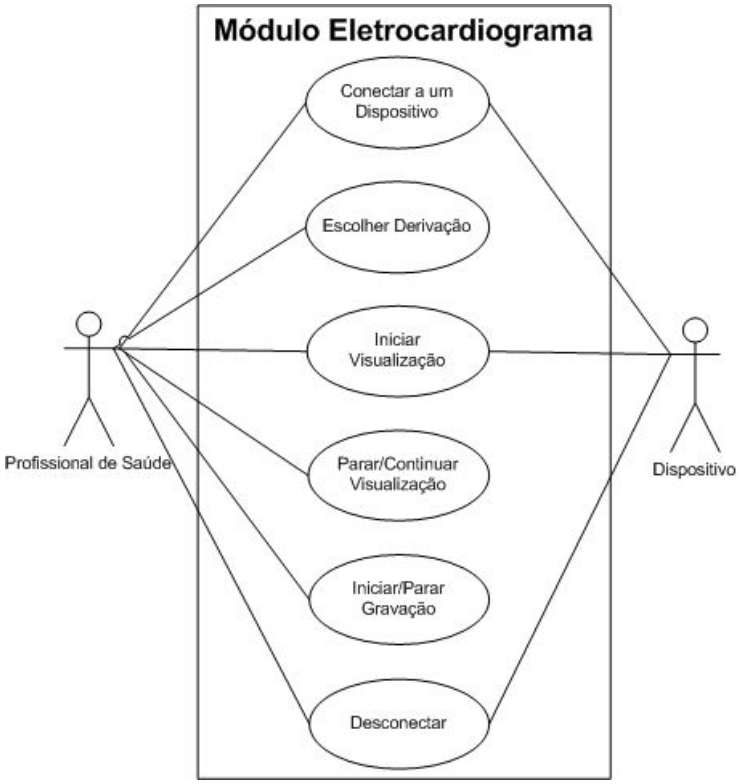

Figura 3.6 – Diagrama de caso do uso – Módulo ECG

3.3.3.1 Escolher Derivação

O profissional de saúde deve poder escolher até três derivações para monitorar simultaneamente, desta forma o sistema permite uma comparação visual entre diferentes curvas, de maneira clara e direta.

Atores:	Profissional de saúde.
Propósito:	Escolher até três derivações para visualizar.
Descrição:	O profissional de saúde escolhe dentre as derivações *Onlines*, até três para visualizar simultaneamente.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde deseja visualizar uma nova derivação.	2. Mostra o conjunto das derivações *Onlines*.
3. O profissional de saúde escolhe até três delas.	4. Gera gráfico para cada derivação e mostra na tela.

3.3.4 Casos de Uso – Módulo Diagnóstico Eletrônico

No módulo diagnóstico eletrônico são levantados os casos de uso referentes à geração de diagnósticos e laudos, contudo novamente não serão redefinidos os casos de uso dos módulos marca-passo e eletrocardiograma, pois estes também se aplicam aqui, a exceção dos casos de uso *parar visualização, continuar visualização, gravar sinais e para gravação dos sinais*.

A figura 3.7 traz o diagrama de caso de uso do módulo de diagnóstico eletrônico e a seguir cada novo caso de uso é explicado em detalhes.

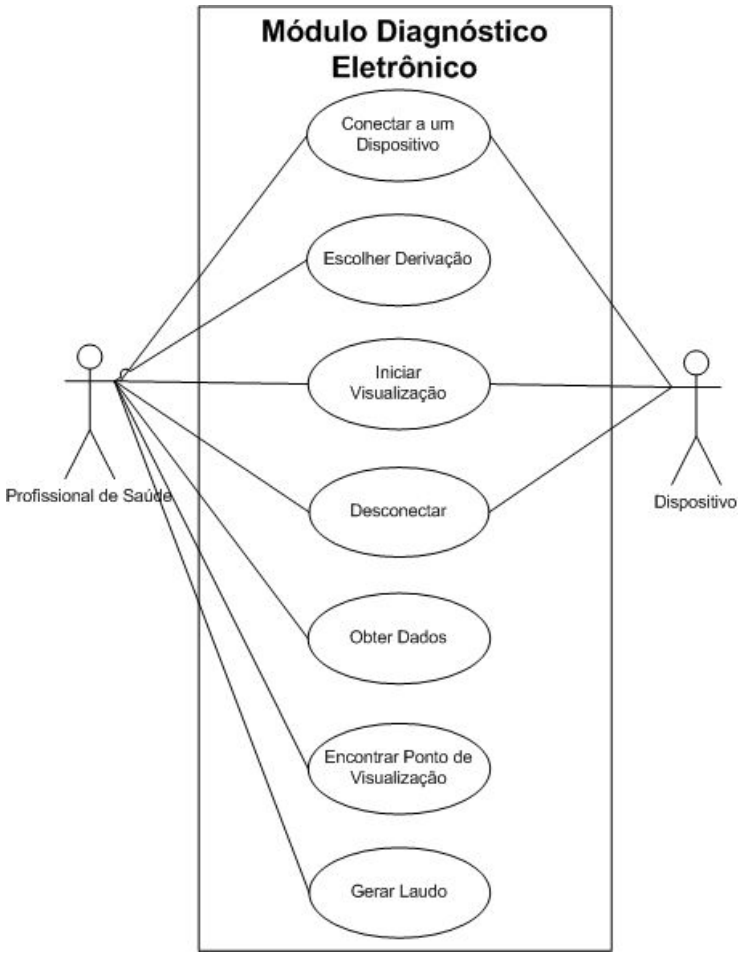

Figura 3.7 – Diagrama de caso do uso – Módulo Diagnóstico Eletrônico

3.3.4.1 Obter Dados

Quando o profissional de saúde desejar fazer uma análise minuciosa de um exame, ele primeiro deve gravar os sinais, e só então poderá abrir no modo de diagnóstico, obter os dados e depois se deslocar até a posição desejada.

Atores:	Profissional de saúde.
Propósito:	Abrir o arquivo com os sinais gravados previamente.
Descrição:	O profissional de saúde escolhe um arquivo com os sinais do paciente para analisar.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **abrir arquivo de dados**.	2. Abre caixa de dialogo para a escolha do arquivo.
3. O profissional de saúde seleciona um arquivo.	4. Monta o gráfico de batimentos.

3.3.4.2 Encontrar Ponto de Visualização

Uma vez carregado os dados, o profissional de saúde pode informar uma posição em unidades de tempo, para que o sistema refaça os gráficos a partir desta posição. O tempo é calculado pegando por base a posição inicial do arquivo e a quantidade de dados em função da taxa de amostragem.

Atores:	Profissional de saúde.
Propósito:	Mostrar o gráfico no ponto de visualização informado.
Descrição:	O profissional de saúde informa uma posição e o sistema procura os dados e monta o gráfico a partir da mesma.
Tipo:	Secundário e opcional.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o	2. Mostra o gráfico de batimentos a partir da

profissional de saúde informa a posição desejada.	posição desejada.
3. O profissional de saúde analisa a curva.	

3.3.4.3 Gerar Laudo

O profissional de saúde informa um intervalo de tempo para análise, ou deixa o sistema analisar todo o arquivo de sinais. O sistema utiliza algoritmos de análise e gera um laudo, o qual pode ser gravado em disco para futuras análises.

Atores:	Profissional de saúde.
Propósito:	Gerar um laudo automático a partir da aplicação de um analisador de sinais.
Descrição:	O profissional de saúde deseja utilizar um algoritmo para gerar um laudo dos sinais captados ou que estavam armazenados.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **gerar laudo**.	
2. Informa o intervalo de tempo para análise ou deixa o sistema analisar todo o arquivo, no caso de dados gravados.	2. Aplica o algoritmo de análise de sinais a fim de gerar um laudo com o diagnóstico eletrônico.
	3. Apresenta o resultado.
4. O profissional de saúde analisa o resultado.	

3.3.5 Casos de Uso – Módulo Laudo

O módulo laudo é responsável por mostrar o laudo profissional de saúde gerado pelo módulo de diagnóstico. O diagrama do caso de uso do módulo Laudo é mostrado na figura 3.8, onde se tem o caso de uso abrir laudo.

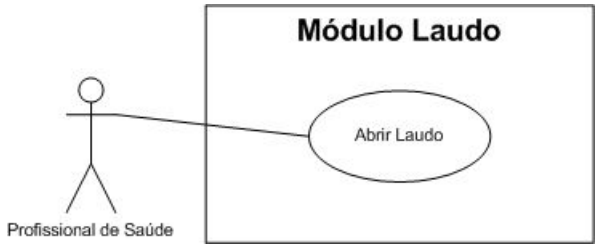

Figura 3.8 – Diagrama de caso do uso – Módulo Laudo

3.3.5.1 Abrir Laudo

O profissional de saúde deve estar apto a abrir um laudo para fazer uma análise mais detalhada de um exame.

Atores:	Profissional de saúde.
Propósito:	Abrir o laudo gerado no módulo de diagnóstico eletrônico.
Descrição:	O profissional de saúde escolhe um laudo com o diagnóstico do paciente para analisar.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde solicita a ação **abrir laudo**.	2. Abre caixa de dialogo para a escolha do laudo.
3. O profissional de saúde seleciona um arquivo.	4. Mostra o laudo.
5. O profissional de saúde analisa.	

3.3.6 Casos de Uso – Módulo Gerenciamento de Paciente e Dispositivo

O módulo gerenciamento de paciente e dispositivo é independente em relação aos módulos vistos até agora, pois nele são tratadas as questões relacionadas à manutenção tanto de pacientes, quanto de dispositivos, ou seja, questões como cadastro e exclusão, configuração e cadastro de parâmetros, além da preparação para o monitoramento.

O diagrama dos casos de uso desse módulo está ilustrado na figura 3.9. Em seguida cada caso de uso é detalhado.

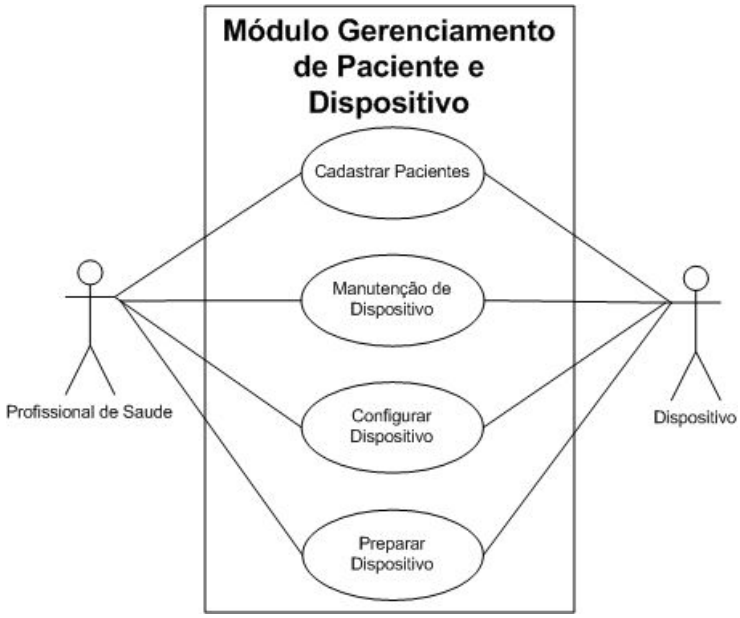

Figura 3.9 – Diagrama de caso do uso – Módulo Gerenciamento de Paciente e Dispositivo

3.3.6.1 Cadastrar Pacientes

Um sistema de monitoramento de sinais cardíacos, pressupõe a existência de um agente passivo - o paciente, do qual os sinais são coletados. Baseado nisto, o caso levantado a seguir, apresenta uma maneira de vincular este agente ao sistema.

O profissional de saúde deve poder cadastrar, alterar e excluir os pacientes no sistema. Pois os dados informados irão identificá-los frente às funcionalidades da aplicação.

Atores:	Profissional de saúde.
Propósito:	Cadastrar os pacientes.
Descrição:	O profissional de saúde cadastra os dados básicos dos pacientes.
Tipo:	Principal e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde chega para cadastrar pacientes.	
2. O profissional de saúde seleciona o menu, ou ícone de cadastro de pacientes.	3. Mostra a tela de cadastro.
4. O profissional de saúde informa os dados.	5. Verifica se o paciente já está cadastrado.
	6. Se o paciente já estiver no cadastro, emite

	mensagem de erro, senão emite mensagem de sucesso.
7. O profissional de saúde sai do formulário.	8. Fecha a tela de cadastro

3.3.6.2 Manutenção de Dispositivo

O profissional de saúde deve cadastrar os dispositivos no sistema para realizar o monitoramento. Nele são informados os dados básicos e também os parâmetros utilizados pelos mesmos. Ele também pode modificar os dados, bem como excluir um determinado dispositivo.

Atores:	Profissional de saúde, Dispositivo.
Propósito:	Realizar as atividades de inserção, modificação e exclusão de dispositivos.
Descrição:	O profissional de saúde cria, modifica ou apaga um dispositivo.
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde seleciona um dispositivo ou cria um novo.	2. Apresenta tela com as propriedades do dispositivo.
3. O profissional de saúde informa os dados e ação (inserção, modificação ou exclusão).	4. Realiza a ação.

3.3.6.3 Configurar Dispositivo

O profissional de saúde pode re-configurar um dispositivo a qualquer tempo, apenas informando novos parâmetros.

Atores:	Profissional de saúde, Dispositivo.
Propósito:	Configurar o dispositivo cardíaco, seja marca-passo ou ECG.
Descrição:	O profissional de saúde informa os parâmetros de configuração do dispositivo de captação.

Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde seleciona um dispositivo para reconfigurar.	
2. O profissional de saúde seleciona entre modo Marca-passo e ECG.	3. Mostra as propriedades que podem ser alteradas, de acordo com o tipo escolhido.
4. O profissional de saúde informa os novos parâmetros.	5. Verifica se os parâmetros estão dentro da faixa de valores válidos.
	6. Se estão, manda um sinal de alteração ao dispositivo e uma mensagem de sucesso a tela, caso contrário mostra uma mensagem de erro.

3.3.6.4 Preparar Dispositivo

Após fazer a ligação física do dispositivo ao computador, o profissional de saúde deve selecioná-lo na lista de dispositivos cadastrados previamente e associá-lo ao paciente em questão, colocando-o então no estado "Preparado".

Atores:	Profissional de saúde, Dispositivo.
Propósito:	Preparar dispositivo para receber os sinais.
Descrição:	O profissional de saúde muda o estado do dispositivo associado fisicamente ao paciente, para "preparado".
Tipo:	Primário e essencial.
Seqüência de Eventos	
Ação do Ator	**Resposta do Sistema**
1. Este caso de uso inicia quando o profissional de saúde seleciona o paciente a monitorar.	2. Mostra os dispositivos já cadastrados e qual porta está usando.
3. O profissional de saúde escolhe aquele, cuja porta está conectada ao paciente e o marca como preparado.	4. Associa o paciente ao dispositivo e muda seu estado para "preparado".

3.4 Especificação do Sistema

Nesta seção serão especificados os conceitos utilizados na aplicação, bem como o relacionamento entre os mesmos. Também serão identificados alguns padrões de projeto em uso.

3.4.1 Cadastros Necessários

Para que o sistema funcione adequadamente é necessário o cadastro prévio dos pacientes. Neste trabalho acadêmico são utilizados apenas atributos essenciais, mas para projetos maiores podem ser incorporadas muitas outras características dos pacientes de forma similar. Visando evitar o recadastramento dos pacientes cada vez que o sistema é reinicializado, o cadastro dos mesmos deve ser gravado em meio persistente.

Da mesma forma que os pacientes, os dispositivos também devem ser cadastrados previamente, armazenados em meio persistente e recuperados por ocasião do uso. Como principais atributos do dispositivo, destacam-se:

- **NumSerie:** representa o número de série do equipamento, o qual deve ser um identificador único.
- **TaxaAmostragem:** é o número de amostras por segundo enviadas pelo dispositivo.
- **Porta:** é a porta a qual o dispositivo está fisicamente conectado.
- **Estado:** representa os diversos estados que o dispositivo pode apresentar, sendo: 0 – Normal; 1 – Preparado e 2 – Ativo.

3.4.2 Sinais

Para representar os sinais cardíacos, provenientes do dispositivo, deverá ser criada a classe **Sinal**. Conforme mostrado na figura 3.10, esta classe possui: os atributos *derivação* do tipo Byte (representação de número inteiro de oito Bits), que armazena a informação sobre qual derivação o sinal pertence, no caso do dispositivo ser um eletrocardiógrafo; e o atributo *valor* também do tipo Byte, que efetivamente significa o valor do sinal coletado. Também possui apenas dois métodos, além do construtor, que servem para recuperar as informações de derivação e valor, comentados anteriormente.

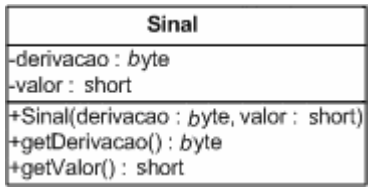

Figura 3.10 – Classe Sinal

3.4.3 Eletrocardiograma

O modo eletrocardiograma deve tratar de diversas derivações simultaneamente. Para tal a classe eletrocardiógrafo, que representa os dispositivos, traz dois atributos especiais: *derivaçõesOnLine*, um vetor que armazena as derivações fisicamente ligadas ao sistema; e *derivaçõesAtivas*, que armazena as derivações efetivamente monitoradas, que podem ser no máximo três.

Multiplexação e Demultiplexação do Sinal

No caso eletrocardiograma, o *driver* de dispositivo deve ser capaz de multiplexar, ou seja intercalar corretamente os sinais provenientes dos diferentes canais, pois cada um representa uma derivação e o sistema só é capaz de reconhecer três delas simultaneamente.

3.4.4 *Drivers* de Dispositivo

Visando facilitar o acoplamento dos dispositivos ao sistema, utilizou-se o padrão Bridge que permite a fácil construção de *drivers*, deixando assim a responsabilidade pelo recebimento e tratamento do sinal para o projetista do hardware, ou seja, quem conhece o dispositivo é que terá de tratar as especificidades do mesmo. Questões como a multiplexação do sinal, parâmetros do canal de comunicação, devem ser tratadas por este *drive*. Para ter um *driver* compatível com o sistema CardioMonitor, um projetista de hardware deve obedecer as interfaces apresentadas na figura 3.11. Para tal basta estender a interface **DispositivoDriver**, de acordo com o padrão de projeto Bridge.

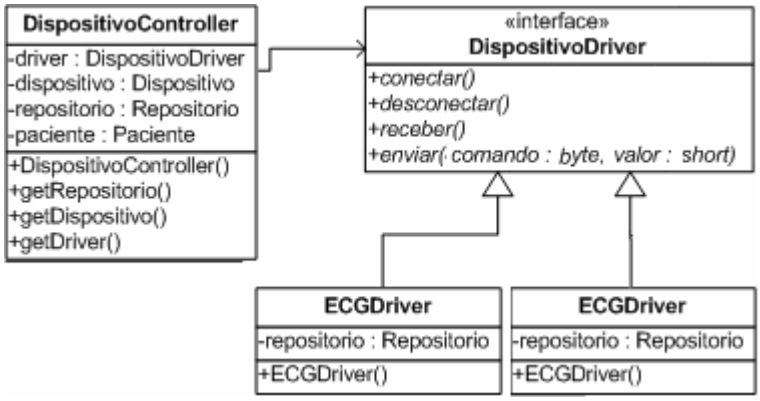

Figura 3.11 – Interfaces para *Driver* de Dispositivo Usando o Padrão Bridge

3.4.5 Repositório de Sinais

Uma vez coletados e tratados, os sinais são armazenados em um repositório. Este repositório utiliza o padrão de projeto Observer para, juntamente com as classes responsáveis pela geração do gráfico, criarem um esquema de atualização dos dados baseado na idéia de mudança de estado. Usando uma estrutura de classes como mostra a figura 3.12, quando o *preBuffer* estiver cheio, o objeto observável (da classe **Repositório)** avisa aqueles que o observam de que seu estado mudou. Então os objetos observadores atualizam seus iteradores coletando os novos dados e gerando novos gráficos e/ou diagnósticos.

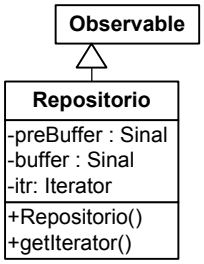

Figura 3.12 – Classe Repositório

3.4.6 Gráfico de Sinais

Da mesma forma que na construção dos *drivers* de dispositivo, a geração dos gráficos também se vale do padrão de projeto Bridge, pois desta forma fica garantida a fácil substituição do algoritmo de plotagem. Conforme mostrado na figura 3.13, basta substituir a classe **Gráfico**, implementando a interface **GraficoGenerico** e um novo algoritmo poderá fazer o trabalho de desenhar o gráfico de batimentos.

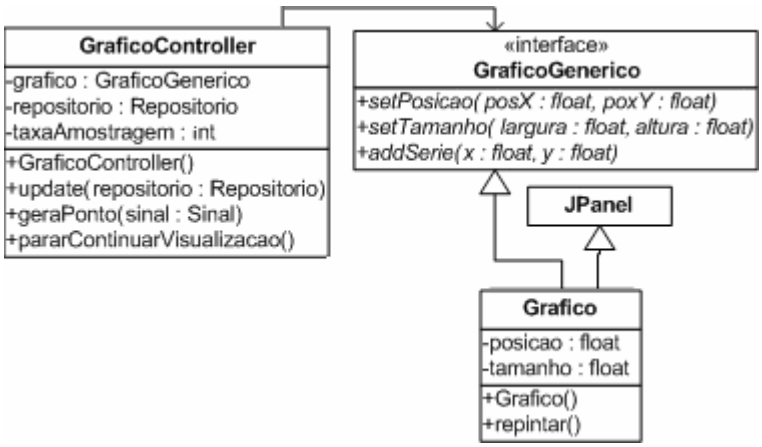

Figura 3.13 – Gráfico Usando o Padrão Bridge

Visando uma maior flexibilidade no projeto, permitindo que até mesmo aplicações escritas em outras linguagens possam ser utilizadas para gerar o gráfico, a interface **GraficoGenerico** apresenta os requisitos mínimos que as mesmas devem cumprir para que haja uma perfeita integração com os outros módulos da aplicação.

3.4.7 Laudo

Inicialmente, os laudos deverão ser gerados em formato *Rich Text*, contudo existem pacotes, como o **PDFBox** que é uma biblioteca Java *open source* para trabalhar com documentos PDF, e permite a criação de novos documentos PDF, manipulação de documentos existentes e extração de conteúdo dos documentos (PDFBox, 2006).

3.4.8 Armazenamento dos Sinais

A gravação dos sinais também se valerá da utilização do padrão Observer para obter o fluxo de dados a ser armazenado. Os sinais serão acumulados e gravados juntamente com uma referência ao paciente do qual eles provêem, e também da taxa de amostragem utilizada pelo dispositivo, necessária a correta temporização do fluxo de dados, quando dá recuperação do arquivo. A classe **SinaisArmazenados** mostrada na figura 3.14 é responsável pelo armazenamento dos sinais.

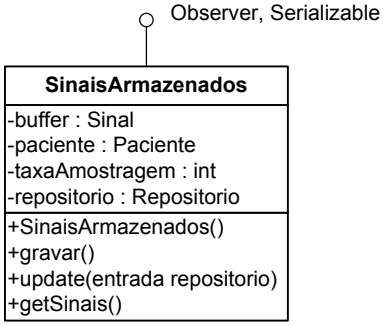

Figura 3.14 – Classe SinaisArmazenados

3.5 Diagrama de Classes

O diagrama de classes do sistema está ilustrado na figura 3.15. Nele a estrutura estática das classes da aplicação, seus atributos e operações, são demonstrados utilizando a notação UML. Os relacionamentos entre as classes, bem como as noções de agregação, generalização e especialização, também podem ser observados. Cabe ressaltar que, por economia de espaço, nem todas as classes utilizadas na aplicação estão retratadas neste diagrama, ficando omitidas classes acessórias como as derivadas do próprio Tool Kit Java. Exemplo disto, são as classes de interface, como barras de ferramentas, caixas de texto, etc. Também, para evitar a aglomeração de elementos textuais no diagrama, somente os métodos essenciais ao entendimento da solução foram retratados no mesmo.

No diagrama de classes podem-se perceber as interações estáticas que formam os módulos identificados no projeto. A classe **Dispositivo** representa uma generalização dos diferentes tipos de dispositivos que podem ser acoplados ao sistema. Ela juntamente com a classe **DispositivoController**, a interface **DispositivoDrivers** e suas especializações formam a base do módulo *gerenciamento de dispositivos*. Os módulos operacionais, tais como o *Marca-passo* e o *eletrocardiograma* seguem um padrão de classes utilizadas. Todos usam as classes **DispositivoController**, **Repositório**, **Sinal**, bem como as relacionadas a interface e a formação do gráfico, para realizar suas operações.

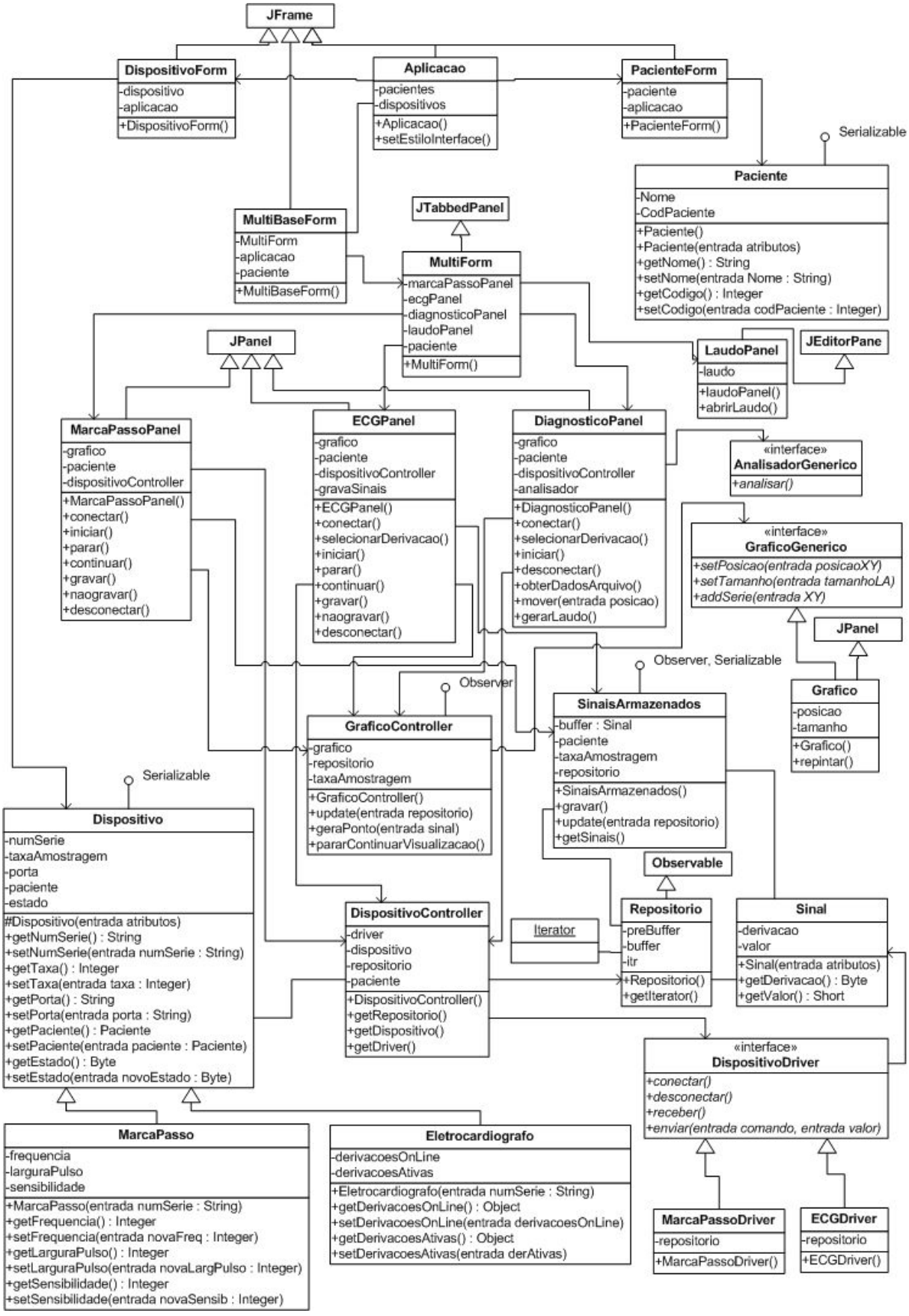

Figura 3.15 – Diagrama de Classes

3.6 Conclusões Parciais

Na busca da concepção de um projeto flexível, versátil e modular, destaca-se o uso do modelo orientado a objetos, da linguagem UML e dos padrões de projetos.

O modelo orientado a objetos confere o poder da abstração ao projeto, trazendo noções de operação e interface permitindo a concepção de classes de maneira abstrata sem se preocupar com suas implementações; traz a noção de encapsulamento, onde a implementação pode ser alterada, mas enquanto a interface da classe não mudar, ela não muda para o exterior; a noção de polimorfismo, que permite que uma mesma operação possa ter várias implementações em classes diferentes; e a característica da reutilização de código através da herança de funcionalidades da classe superior.

Entretanto, o simples uso da orientação a objetos não garante a obtenção de sistemas confiáveis, robustos, extensíveis e reutilizáveis. Para tal, na modelagem apresentada neste capítulo foi utilizado extensivamente padrões de projeto, os quais descrevem as relações entre conceitos, estruturas e mecanismos sedimentados por uma grande gama de projetos.

A descrição detalhada dos principais casos de uso e requisitos, possibilitaram a especificação e a diagramação usando componentes UML, o que confere clareza e objetividade à modelagem.

4 IMPLEMENTAÇÃO E TESTE DO SISTEMA

Neste capítulo são tratados aspectos referentes a implementação e testes do sistema. Em linhas gerais, o leiaute da aplicação segue a utilizada no desenvolvimento inicial do CardioMonitor, pois esta mostrou-se eficiente. Porém a base será toda alterada para espelhar a filosofia utilizada neste projeto, a qual prima por uma formalização bem maior do que a do antecessor. A seção 4.1 analisa a implementação do sistema, enquanto a seção 4.2 traz uma explanação acerca dos testes e resultados. A seção 4.3 apresenta uma breve conclusão do capítulo.

4.1 Implementação

Da mesma forma que no projeto inicial, nesta nova abordagem também foi utilizada a linguagem de programação Java, pois esta oferece recursos de orientação a objetos necessários ao desenvolvimento do sistema a partir da modelagem proposta. Para facilitar a programação, sobre tudo dos componentes gráficos, utilizou-se o ambiente de desenvolvimento integrado Java NetBeans 4.0 (NetBeans, 2006).

As classes sugeridas no diagrama de classes foram agrupadas em pacotes para tornar a programação mais clara e legível. Segue a relação dos pacotes criados:

- **CardioMonitor:** possui as classes **Paciente**, **Main** e **SaveFile**. As classe **Main** e **SaveFile** não estão relacionadas no diagrama de classes, contudo a primeira traz o método estático *Main*, que é utilizado para inicializar a aplicação, enquanto a segunda providencia métodos para a gravação e recuperação de objetos em disco.
- **CardioMonitor_ui:** agrupa as classes relacionadas a geração da interface com o usuário (ui – user interface).
- **CardioDispositivo:** traz classes que permitem modelar os dispositivos de monitoramento cardíaco.
- **CardioDrivers:** contém as classes relacionadas à criação de *drivers* de dispositivos.
- **CardioGrafico:** traz as classes relacionadas à construção dos gráficos de batimentos.
- **CardioSinais:** neste pacote são agrupadas as classes que modelam os sinais cardíacos, bem como os *buffers* utilizados para armazená-los.

Embora a modelagem tenha especificado os módulos de eletrocardiograma, diagnóstico eletrônico e laudo, foram implementadas somente as classes relacionadas aos módulos de seleção, controle de dispositivo e marca-passo. A implementação dos outros

módulos foi preterida em função da modelagem. Os módulos eletrocardiograma e diagnóstico eletrônico devem ter sua implementação similar ao módulo marca-passo, enquanto a do módulo laudo deve ser uma extensão da classe Java **RTFEditorKit,** que pode ser conferida na documentação contida em (JAVA HOME PAGE, 2006).

A seguir são demonstradas algumas interfaces resultantes dos casos de uso implementados. O caso seleção do paciente a monitorar aparece ilustrado na figura 4.1. O sistema mostra a lista com os pacientes e o médico escolhe aquele que pretende monitorar.

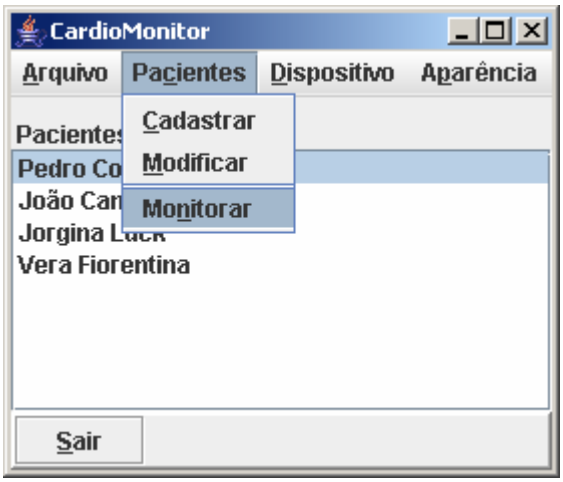

Figura 4.1 – Seleção de Paciente

Também na seleção do estilo da interface gráfica as opções disponíveis são apresentas diretamente no menu, facilitando a escolha. A figura 4.2 retrata este caso de uso apresentando uma tela no estilo "Windows", contrastando com as demais telas mostradas neste trabalho que são apresentadas no estilo padrão "Java Swing". Nela pode-se conferir as opções disponíveis através do menu "Aparência".

Figura 4.2 – Seleção de Estilo da Interface Gráfica

A figura 4.3 mostra a implementação dos casos de uso do módulo marca-passo, além de ilustrar a formação de uma curva cardíaca produzida a partir de uma onda simulada. Nela ainda pode ser observada a utilização de painéis de tabulação demonstrando o caso de uso escolha do módulo de visualização.

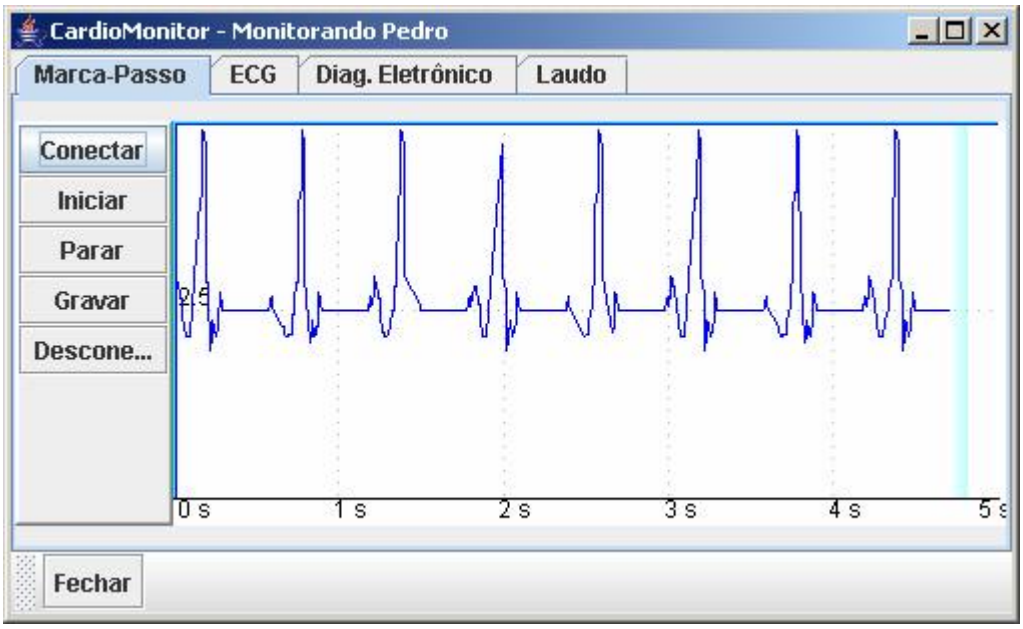

Figura 4.3 – Módulo Marca-passo

O cadastro de pacientes, onde o médico pode inserir, alterar e excluir pacientes, deve ser realizado através da interface mostrada na figura 4.4. Para realizar o cadastro o médico abre a tela para informar os dados do novo paciente. O sistema então verifica a existência de um paciente com o mesmo código informado, e em existindo emite uma mensagem de alerta ao médico, não permitindo a inserção. Caso contrário o sistema adiciona o paciente a base de dados e a lista de pacientes aptos ao monitoramento.

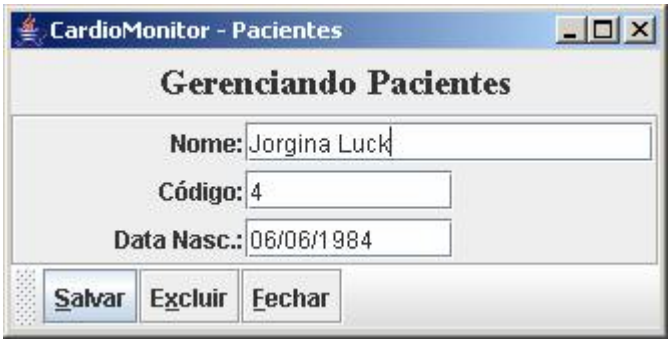

Figura 4.4 – Gerenciamento de Pacientes

Cabe ressaltar ainda a utilização da API de comunicação serial Java (COMMAPI) no desenvolvimento do *driver* para o dispositivo marca-passo capaz de acessar a porta serial. A COMMAPI, permite desenvolver aplicações Java baseadas em comunicação serial via portas

RS-232 (MAZZUTTI & NUNES, 2004). Ela possui basicamente duas versões: a 2.0.3 para Solaris/SPARC; e a 2.0 for Windows e Solaris x86. Sua classes permitem aos desenvolvedores:

- Enumerar as portas disponíveis do sistema;
- Abrir e identificar o proprietário das portas;
- Resolver conflitos entre múltiplas aplicações;
- Tratar modos síncronos e assíncronos de transmissão;
- Receber eventos que descrevem mudanças de estado nas portas.

4.2 Testes e Resultados

Foram realizados testes tão somente para analisar a validade do fluxo de sinais na aplicação.

O primeiro passo foi testar a **recepção dos sinais através da porta serial**. Para isto, utilizou-se um dispositivo gerador de sinais, não necessariamente sinais cardíacos, na porta serial. Em seguida, foi testada a implementação do padrão Observer, o qual ficou com a responsabilidade de notificar os demais componentes sobre uma mudança de estado no buffer de sinais, possibilitando o repasse dos sinais aos mesmos. Por último foram realizados testes de geração do gráfico do módulo marca-passo, sendo utilizada uma classe geradora de ondas cardíacas desenvolvida e testada na primeira versão do CardioMonitor.

O resultado da geração de um gráfico de sinais para uma visualização de cinco segundos a uma taxa de amostragem de 500 amostras por segundo pode ser conferida na figura 4.1. Os testes de captação de dados na porta serial também foram considerados satisfatórios, assim como a utilização do padrão Observer mostrou-se de grande valia, no que tange a flexibilidade do projeto.

Já a intenção inicial de utilizar o padrão Iterator para acessar a coleção de sinais armazenados no buffer da aplicação, mostrou-se ineficaz, pois conforme (METSKER, 2004) os iteradores fornecidos pelo pacote *java.util* possuem reconhecimento *fail-fast* de modificação concorrente a uma coleção, ou seja, não permitem a modificação da coleção entre duas iterações. Logo não podem ser utilizados em coleções que sofrem constantes mudanças como no caso do *buffer* de sinais. A solução encontrada para contornar este problema, foi utilizar um recurso oferecido pela própria implementação do padrão Observer em Java, o qual oferece a possibilidade de se passar um objeto como parâmetro em uma chamada de notificação. Desta forma, a cada chamada o *buffer* informa seu índice atual,

permitindo aos objetos notificados saberem a partir de qual posição os dados estão disponíveis.

4.3 Conclusões Parciais

Este capítulo tratou da implementação, testes e resultados dos aspectos apresentados na modelagem do projeto. Nele ficou demonstrada a divisão das classes em diferentes pacotes, visando trazer legibilidade e organização ao código. Também expõe o modelo de interface adotado, o qual irá utilizar painéis de tabulação para alternar entre as diferentes operações. Ainda traz uma maneira de acessar os dados através de portas seriais, a qual pode ser utilizada na construção dos *drivers* de dispositivo. A seção de teste recebeu pouca ênfase, limitando-se a testar estruturas essências, como as que compõem os padrões de projeto utilizados.

Mesmo não contemplando todos os módulos levantados na mesma, este capítulo demonstrou clara indicação para uma implementação definitiva da solução descrita na modelagem, ficando desta forma ressaltada a validade da mesma.

5 CONCLUSÃO

A informática em saúde tem apresentado grandes avanços, graças à evolução da tecnologia da informação. Gradual e imperceptivelmente, ela vai se tornando cada vez mais presente no cotidiano do profissional de saúde. Novas tecnologias e metodologias, como a telemedicina, os sistemas de apoio à decisão médica, o prontuário eletrônico, entre outros conhecimentos, prometem revolucionar a forma de se praticar a medicina. Daí, a importância deste trabalho, o qual apresentou a modelagem de um sistema de monitoramento cardíaco capaz de gerar as curvas de batimentos cardíacos a partir de sinais coletados de marca-passos e eletrocardiogramas, também servindo de base para o acoplamento de algoritmos de diagnóstico eletrônico e geração de laudos.

Nele foram abordados tópicos sobre eletrocardiograma, visando trazer a luz da compreensão, termos e conceitos que envolvem está área da medicina. Também foi revisitado o trabalho inicial do CardioMonitor, o qual serviu de inspiração para o desenvolvimento do projeto. Ainda foi contemplado o estudo da modelagem orientada a objetos na busca pela formalização e dos padrões de projeto, visando garantir a utilização de soluções largamente testadas na solução do problema. Na modelagem da solução foram levantados os casos de uso, determinados os requisitos, feita a especificação de componentes, bem como foram elaborados diagramas de classes e seqüência procurando demonstrar o comportamento, tanto estático como dinâmico, dos objetos. Já no capítulo de implementação e testes, as entidades ligadas ao módulo **MarcaPasso** foram implementadas e testadas, servido desta forma, de modelo a implementação dos demais módulos, validando o fluxo de dados da aplicação e demonstrando a validade da utilização de padrões de projeto.

As principais diferenças entre o projeto inicial do **CardioMonitor** e a nova abordagem proposta neste trabalho, encontram-se na modelagem modular desta e no uso de padrões de projeto, pois naquele os sinais eram tratados no bloco “Tratamento do Sinal” e entregue já formatados aos blocos funcionais, conforme visto no diagrama da figura 2.5. Isso praticamente inviabilizava a inclusão de novas funcionalidades, porque haveria a necessidade de se alterar o módulo de tratamento do sinal, correndo-se o risco de prejudicar o funcionamento das funções já implementadas. Por outro lado, neste novo projeto, cada módulo funcional é responsável pelo tratamento do sinal, permitindo desta forma a inclusão sem interferência nos demais módulos.

A implementação dos módulos apresentados na modelagem, mas não contemplados neste trabalho, bem como a substituição do algoritmo gerador do gráfico, por um com desempenho superior, ficam como sugestões para trabalhos futuros. Fica ressaltada também a importância da continuidade das pesquisas nesta área tão promissora, que une os avanços tecnológicos da informática, com as mais modernas técnicas da medicina em prol daquilo que de mais caro há para as pessoas: a saúde.

6 BIBLIOGRAFIA

[ALEXANDER, 1977] ALEXANDER, C., ISHIKAWA, S., SILVERSTEIN, M., JACOBSON, M., FIKSDAHL-KING, I., ANGEL, S. **A Pattern Language**. New York, NY (USA): Oxford University Press, 1977.

[LARMAN, 1997] ANQUETIL, Nicolas. **Desenvolvimento de software orientado a objetos**, 2002. Acesso em setembro de 2005. http://www.ucb.br/ucbtic/mgcti/paginapessoalprof/Nicolas/Disciplinas/UML/node1.html

[CARNEIRO, 1997] CARNEIRO, Enéas F. **O eletrocardiograma – 10 anos depois**. 1ª ed. 5ª reimpressão. Rio de Janeiro: Livraria Editora Enéas Ferreira Carneiro, 1997.

[FERREIRA, 1977] FERREIRA, Aurélio B. H. **Minidicionário da Língua Portuguesa**. Rio de Janeiro, RJ: Nova Fronteira, 1977.

[GAMMA *et al.* 1995] GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Reading, MA: Addison Wesley, 1995.

[JAVA HOME PAGE, 2006] **Java Technology Home Page**. Acesso em janeiro de 2006. http://java.sun.com.

[LARMAN, 1997] LARMAN, Craig. **Applying UML and Patterns:** Na Introduction to Object-Oriented Analysis and Design. Editora Prentice-Hall do Brasil: Rio de Janeiro, 1997.

[MAZZUTTI & NUNES, 2003] MAZZUTTI, Cristiano. NUNES, Raul Ceretta. **CardioMonitor: Ferramenta para Visualização de Curvas de Batimentos Cardíacos**. CRICTE, Univali**,** Itajaí – SC. Outubro de 2003.

[MAZZUTTI & NUNES, 2004] MAZZUTTI, Cristiano. NUNES, Raul Ceretta. **Acesso às Portas Seriais Utilizando Java Commapi**. SEMINFO, Torres – RS. Outubro de 2004.

[METSKER, 2004] METSKER, Steven J. **Padrões de Projeto em Java.** Trad. Werner Loeffler. Porto Alegre: Bookman, 2004.

[MS VISIO, 2003] Microsoft Office Online. **MS Visio**, 2003. Acesso em outubro de 2005. http://office.microsoft.com/pt-br/FX010857981046.aspx

[NetBeans, 2006] **NetBeans Home Page**. Acesso em janeiro de 2006. http://www.netbeans .org/community/releases/41/index.html

[PDFBox, 2006] **PDFBox - Java PDF Library**. Acesso em janeiro de 2006. http://www.pdfbox.org/.

[RUIZ, 2002] RUIZ, Rogério M., Dr., Cardiologista de Medicina Preventiva. **Eletrocardiograma: o que é e para que serve?,** 2002. Acesso em outubro de 2005 http://www.medial.com.br/ noticia.asp?cod=240.

[SOARES, 2003] SOARES, Humberto. **Extreme Programming** – Supere o Medo. Anais do IV Workshop sobre Software Livre. Porto Alegre: Sociedade Brasileira de Computação, 2003. p. 13.

[TRANCHESI, 1983] TRANCHESI, João. **Eletrocardiograma normal e patológico:** noções de vectorcardiografia. 6ª ed. Ver. E atualizada/ por Paulo Jorge Moffa. São Paulo: Atheneu, 1983.