UNIVERSIDADE DE SÃO PAULO
Programa de Pós-Graduação em Neurociências e Comportamento

Carlos López Noriega

Desenvolvimento de um programa computacional para avaliação postural de código aberto e gratuito

São Paulo
2012

Carlos López Noriega

Desenvolvimento de um programa computacional para avaliação postural de código aberto e gratuito

(Exemplar Revisado)

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre em Neurociências e Comportamento do Programa de Pós-Graduação do Instituto de Psicologia da Universidade de São Paulo

Área de Concentração: Neurociências e Comportamento

Orientador: Prof. Dr. Marcos Duarte

São Paulo

2012

Catalogação na publicação
Biblioteca Dante Moreira Leite
Instituto de Psicologia da Universidade de São Paulo

Noriega, Carlos López.
Desenvolvimento de um programa computacional para avaliação postural de código aberto e gratuito / Carlos López Noriega; orientador Marcos Duarte. -- São Paulo, 2012.
83 f.
Dissertação (Mestrado – Programa de Pós-Graduação em Psicologia. Área de Concentração: Neurociências e Comportamento) – Instituto de Psicologia da Universidade de São Paulo.

1. Linguagens de Programação para Computador 2. Postura 3. Aplicações do Computador 4. Diagnostico Computarizado I. Título.

QA76.7

Carlos López Noriega

Desenvolvimento de um programa computacional para avaliação postural de código aberto e gratuito

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre em Neurociências e Comportamento do Programa de Pós-Graduação do Instituto de Psicologia da Universidade de São Paulo sob a orientação do Prof. Dr. Marcos Duarte.

Aprovado em:

_____
Prof. Dr. Marcos Duarte
Universidade de São Paulo

_____
Prof. Dr. Elizabeth Alves Ferreira
Universidade de São Paulo

_____
Prof. Dr. Edison Puig Maldonado
Fundação Armando Álvares Penteado

São Paulo, _____/ _____/ _____

Para aquela bela mulher de cabelo castanho, olhos pardos e alegre sorriso: Maria, minha mãe.

Para Rô, minha amada esposa, melhor amiga e eterno amor: pelo seu constante apoio, maior incentivo e inabalável confiança (obrigado, Preciosa).

Para o Galo, meu irmão, o meu melhor amigo e companheiro em todas as etapas da minha vida (Gracias, Bro).

Agradecimentos

Primeiro e antes de todo a Deus e a Virgem Maria: diante de todas as dificuldades trouxeram pessoas ou criaram situações que de alguma forma me apoiaram e auxiliaram nesses momentos.

Ao Marcos, meu orientador, pelo apoio e confiança para um estrangeiro que apareceu e bateu na porta do seu laboratório: obrigado de todo coração, Marcos.

À todo o pessoal do LoB, em especial e por ordem de tamanho: Raquel, Cris, Andrea, Lúcia, Alethea, Bel e Janina, obrigado meninas por todo este tempo compartido.

Aos amigos que deixei no meu país, mas que me acompanharam virtual e espiritualmente todo esse tempo: Dorita, Ale, Christian (Peto) e em especial Stef, levo todos vocês no meu coração (“algún dia nos encontraremos para comer um cevichito juntos”).

Aos meus pais (Galo e Maru), que me incentivaram a tentar “esta aventura”, obrigado pelo apoio e a amizade.

A COSEAS, que me deu um lugar onde ficar e dormir no CRUSP durante quase dois anos.

A CAPES e a FAPESP, sem o apoio econômico destas instituições seria impossível realizar esta pesquisa.

Ao Galo e a Lelê: que me alojaram na sua casa quando recém cheguei ao Brasil, obrigado pelo apoio, dicas, mapas, comida: vocês foram demais, pessoal.

A Carla e a Bete , ambas do ProMexico, amigas “do bem” que acompanharam e torceram por mim no inicio deste processo: meu agradecimento infinito.

Ao pessoal do CRUSP que foram minha família durante os fins de semana, obrigado pelos magníficos momentos: Rô, Marielle, Gal.

Ao pessoal da halterofilia: Maira, Maria, Cora, Dê, Marcos, Salomão e em especial ao Renê e o Leo: tod@s vocês me fizeram sentir que sempre estava em casa, obrigado pela amizade, pessoal.

Aut viam inveniam aut faciam
*Hannibal (*247 a.C. — 183 a.C.*)*

Resumo

O uso de ferramentas computacionais para avaliação postural tem sido de grande valia na detecção das alterações posturais, porém a utilização destes programas exige estruturas de hardware complexas e implica em custos elevados para pesquisadores da fisioterapia, educação física e da comunidade científica. No ano 2005 foi criado o Software de Avaliação Postural (SAPO) que é uma opção gratuita para os mesmos fins, amplamente utilizada pela comunidade científica e profissional com ótimos resultados documentados. Apesar do sucesso do SAPO na comunidade científica este programa possui limitações. Neste âmbito a proposta do presente trabalho é desenvolver um software denominado ApLoB (Avaliação Postural do Laboratório de Biofísica) para avaliação postural, tendo como parâmetro de desenvolvimento o SAPO, mas tentando colaborar em relação à superação de suas limitações. Para isso, seu desenvolvimento é baseado nas estruturas e metodologias estabelecidas pela engenharia de software que permitam a continuidade do trabalho e melhorias de suas funcionalidades. O software foi desenvolvido utilizando a linguagem de programação Python, suas extensões científicas como NumPy, a biblioteca de processamento de imagem (PIL), a aplicação para interfaces gráficas (PyQt), além da biblioteca de plotagem de dados em 2D e desenvolvimento de aplicações de processamento de sinais (Guiqwt), dentre outros. O protótipo obtido foi testado e comparado em relação às suas funcionalidades com o software SAPO e foram considerados aceitáveis.

Palavras-chave: Linguagens de Programação para Computador, Postura, Aplicações do Computador, Diagnostico Computarizado.

Abstract

The use of computational tools for postural evaluation has been very valuable in the detection of postural changes, however the use of these programs requires complex hardware structures and involves high costs for researchers in physiotherapy, physical education and the scientific community. In 2005, the Postural Assessment Software (SAPO) became to be a free option for the same purpose, widely used by the scientific community and professional with excellent documented results. Despite the success of SAPO in the scientific community, this software has limitations. So, the purpose of this study is to develop a software called ApLoB (Postural Assessment Laboratory of Biophysics) for postural assessment, having as parameter the development SAPO, but trying to collaborate on the overcome of its limitations. For this reason, its development is based on the structures and methods established by the software engineering that allow continuity of work and improved functionality. The software was developed using the Python programming language, scientific and NumPy extensions, the library of image processing (PIL), the application for graphical interfaces (PyQt), as well as data plotting library of 2D and application development signal processing (Guiqwt), among others. The prototype obtained was tested and its functionality was found to be acceptable, compared to SAPO.

Keywords: Computer Programming Languages, Posture, Computer Applications, Computer Assisted Diagnosis.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

ABNT - Associação Brasileira de Normas Técnicas
ApLoB - Avaliação Postural do Laboratório de Biofísica
APPID - Avaliação Postural a partir de Imagem Digital
AVE - Acidente Vascular Encefálico
CASE - Computer-Aided Software Engineering
CAPES - Coordenação de aperfeiçoamento de pessoal de nível superior
CBSE - Component-Based Software Engineering
CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico
ESBC - Engenharia de Software Baseada em Componentes
FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo
GUI - Graphical User Interface
GPL - General Public License
ISO - International Organization for Standardization
IEC - International Electrotechnical Commission
IEEE - Institute of Electrical and Electronics Engineers
LoB - Laboratory of Biophysics
OOS - Sistemas Orientados a Objetos
PIL - Python Imaging Library
Pixel - Picture Element
ROSE - Rational Object Oriented Software Engineering
SAPO - Software de Avaliação Postural
UML - Unified Modeling Language
USO - Universidade de São Paulo
VHLL - Very High Level Language

Sumário

1. Introdução

A avaliação postural é um procedimento fundamental no diagnóstico do alinhamento dos segmentos corporais de um indivíduo e é amplamente utilizada pelos profissionais de fisioterapia e educação física, constituindo-se como um passo inicial e de acompanhamento para a avaliação e tratamento fisioterapêutico e prescrição de atividade física (Day, Smidt et al. 1984; Danis, Krebs et al. 1998; Nault, Allard et al. 2002).

Existem programas de computador disponíveis comercialmente para avaliação postural, mas estes programas carecem de uma fundamentação científica adequada ou apresentam custos elevados que impossibilitam seu uso pela ampla gama de profissionais interessados em sua utilização no Brasil e no exterior. A despeito de sua importância no diagnóstico, infelizmente poucos são os profissionais no Brasil que dispõem de tal ferramenta para uso profissional.

Em 2003, o Prof. Dr. Marcos Duarte coordenou um projeto para desenvolvimento de um software de avaliação postural financiado pela FAPESP e pelo CNPq. Desde então este software chamado de SAPO (Software de Avaliação Postural), tem sido amplamente utilizado pela comunidade científica e profissional. O programa se encontrava hospedado na Incubadora Virtual da FAPESP, e de acordo com a informação obtida neste site em 22 de fevereiro de 2010 o endereço do programa possuía em média mais de 300 visitas mensais e era o 13º mais acessado dentre os mais de 500 projetos hospedados[1]. Estes números demonstram o grande impacto e receptividade de tal ferramenta em nossa sociedade.

O Laboratório de Biofísica (LoB) da USP, inserido nos Programas de Pós-Graduação em Neurociências, na Física e na Educação Física, vem desenvolvendo trabalhos na área de Biomecânica e Controle Motor. Dentro deste contexto e dando continuidade a estas linhas de pesquisa em relação à avaliação postural, desenvolveu-se através desta pesquisa um programa denominado ApLoB. Deste modo o presente trabalho deu continuidade às possibilidades de pesquisas em avaliação postural capaz de possibilitar o uso de métodos não invasivos nesta área. Neste âmbito propõe-se um software que faça medições de ângulos e cálculos de distâncias entre pontos distantes nos planos anterior, posterior, lateral esquerdo e lateral direito, usando como método a biofotogrametria, tendo na sua estrutura de

[1] O programa pode ser descarregado na internet de http://puig.pro.br/sapo/.

desenvolvimento uma linguagem de programação orientada a objetos como Python, e com a idéia de ser de distribuição livre e com código aberto sob os parâmetros da engenharia de software.

1.1 Objetivo

O objetivo deste trabalho é o desenvolvimento de um programa computacional para avaliação postural, que sirva como ferramenta complementar para os profissionais da área, e cuja aquisição seja gratuita (descarregado via internet) e de código aberto (software que mostra suas fontes de programação com o propósito de possibilitar uma melhoria contínua mediante a colaboração de terceiros) usando como parâmetro de desenvolvimento o SAPO devido ao sucesso do mesmo nas comunidades científica e profissional.

Neste âmbito é importante salientar que nosso objetivo no desenvolvimento deste programa não é ultrapassar o SAPO, mas sim abrir o caminho para a criação de um programa cujo melhoramento que possa ser facilitado posteriormente. Para isso, a proposta de desenvolvimento deste trabalho se insere sob os lineamentos, metodologias e conceitos estabelecidos na engenharia de software, facilitando assim a inclusão de contribuições por parte de terceiros interessados no mesmo.

1.2 Justificativa

Diversas razões levam o software SAPO a ter boa recepção e reconhecimento pelas comunidades científica e profissional: o mesmo consiste em um software livre, que necessita de baixos requerimentos de hardware, apresentando várias pesquisas que comprovaram sua acurácia e confiabilidade, bem como sua conseqüente relevância nas áreas ligadas à avaliação postural.

Apesar destes importantes atributos, o SAPO possui características que podem ser consideradas como limitações: o programa está desenvolvido em Java, que apesar se de consistir numa excelente linguagem de programação, possui complexidade estrutural de código e requer alto nível de experiência de seus desenvolvedores. Além disso, essa linguagem não é frequentemente usada na comunidade científica. Outra questão importante é o fato do SAPO não apresentar uma modelagem propriamente dita do seu programa, mas apenas diagramas de hierarquia para a construção de classes. Como conseqüência, a ausência da

modelagem do programa dificulta a continuidade do trabalho e as melhorias de suas funcionalidades.

Diante deste contexto, a criação do ApLoB busca tornar pública a modelagem de um programa de avaliação postural e, deste modo, abrir caminhos para a melhoria contínua de suas funcionalidades. Para isso foi utilizado o Python como linguagem de programação: a escolha se deve ao fato de sua estrutura de desenvolvimento ser muito similar à do Matlab, programa comumente usado pela comunidade científica e, por isso, mais acessível em relação à sua programação. Com uma metodologia de desenvolvimento estandardizada na engenharia de software que permita uma evolução e estudo contínuo do programa, mantendo a ideia original do software livre onde o código desenvolvido para a aplicação é "aberto", ou seja, terceiros podem acessar a fonte do programa podendo reprogramar, reestruturar ou personalizar o mesmo conforme suas próprias necessidades.

2. Revisão bibliográfica

2.1. O que é postura?

Gangnet, Pomero et al. definem postura como a composição do posicionamento de todos os segmentos corporais num determinado momento (Gangnet, Pomero et al. 2003). Já Winter (1999) afirma que postura é o termo que descreve a orientação de qualquer segmento corporal relativo ao vetor aceleração da gravidade, enquanto o Comitê de Postura da *American Academy of Orthopaedic Surgeon* a define como o arranjo relativo das partes do corpo, onde o equilíbrio muscular e esquelético é responsável pelo posicionamento adequado e eficiência muscular.

Tradicionalmente, análises posturais buscam caracterizar a postura dos indivíduos utilizando como referencial a chamada "postura padrão" (Comerlato 2007). Magee (2010) define postura correta como sendo a "posição na qual um estresse mínimo é imposto sobre cada articulação".

Na postura padrão, a coluna apresenta curvaturas fisiológicas, os membros inferiores estão em alinhamento ideal para sustentação de peso, (Kendall 2007) e a cabeça está ereta, o que minimiza o estresse sobre a musculatura do pescoço.

Assim, uma boa postura não seria apenas uma questão estética, mas também uma utilização biomecânica ótima das estruturas do nosso corpo.

2.2 O que é Avaliação Postural?

A coluna vertebral sofre diversas influências como, por exemplo, sobrepeso, deficiências nutricionais, atividade física irregular ou insuficiente, perturbações respiratórias, padrões de encurtamentos musculares e vícios posturais inadequados, podendo causar desvios ou alterações na postura. Para determinar essas assimetrias ou desvios posturais é realizada a avaliação postural.

Podemos citar como uma das primeiras avaliações posturais o método empregado por Kendall no qual era usado um fio de prumo para representar uma linha de referência (intersecção do plano sagital com o plano coronal que forma uma linha que corresponde à linha de gravidade), determinando os desvios da postura dos sujeitos analisados em relação a essa linha (Kendall 2007).

Para os fisioterapeutas a avaliação postural é um procedimento imprescindível e parte fundamental de uma avaliação músculo esquelética, pois ela auxilia no diagnóstico precoce de alterações no posicionamento das estruturas articulares, ósseas e musculares, podendo dessa forma detectar desvios com mais facilidade e rapidez. Para essa avaliação, além do conhecimento profissional, é necessário experiência para a localização correta dos pontos anatômicos, o que é um pré-requisito importante para garantir a reprodutibilidade e confiabilidade da análise postural. Algumas regiões, como a coluna vertebral, por exemplo, oferecem maior dificuldade ao avaliador (Ferreira 2006). Assim a avaliação postural é importante para o diagnóstico, planejamento e acompanhamento da evolução e dos resultados de um tratamento fisioterapêutico (Batista LH 2006).

2.3 Métodos de Avaliação Postural

2.3.1 Avaliação Qualitativa

Simetrógrafo

Segundo Adams (2001) e Kendall (Kendall 2007), o simetrógrafo é um instrumento que permite avaliar a postura através das linhas horizontais e verticais pintadas em sua superfície formando quadrados. No centro uma linha é demarcada fortemente imitando o fio de prumo e essas linhas e quadrados auxiliam na visualização das deformidades quando o examinado posiciona-se ante o instrumento. O confronto dos segmentos corporais com as linhas e os quadrados são as referências para o exame postural no simetrógrafo.

2.3.2 Avaliação Quantitativa

- **Radiografia**

Os raios-X (radiografia) são os mais utilizados para a avaliação postural e sua confiabilidade está amplamente descrita na literatura. Gardocki, Watkins et al. (2002) utilizaram a radiografia para analisar e quantificar a lordose lombopélvica pelo Método de Cobb e concluíram que o uso da radiografia é satisfatório para quantificar as alterações da coluna.

Benson and Richmond (1997) indicam que esse é um método efetivo e fidedigno na avaliação das alterações posturais, porém apontam o problema da radiação ser prejudicial ao corpo humano, podendo ocorrer alterações em nível celular como morte da célula, interrupção ou desaceleração do seu processo de divisão, além de alterações no DNA que podem ser transmitidos geneticamente (Herscovici and Sanders 2000).

Assim, o raio-X (radiografia) é uma ferramenta quantitativa para avaliar a presença e o grau de alteração da coluna apesar dos riscos comprovadamente presentes na exposição sucessiva à radiação (Comerlato 2007; Giglio and Volpon 2007). Cabe ressaltar também que este método possui um custo maior e normalmente não está disponível para o fisioterapeuta durante a sua avaliação (Benson and Richmond 1997).

Neste âmbito ressalta-se a importância de novos estudos objetivando o desenvolvimento e aperfeiçoamento de técnicas não invasivas, de fácil manejo e

baixo custo que possam ser amplamente utilizadas para a avaliação postural, como por exemplo, acompanhamento da escoliose (desvio lateral da coluna vertebral) que requer grandes cuidados e acompanhamento por longo período (Comerlato 2007).

- **Fotogrametria**

A *American Society of Photogrammetry* define a fotogrametria como "*a arte, ciência e tecnologia de obtenção de informação confiável sobre objetos físicos e o meio ambiente através de processos de gravação, medição e interpretação de imagens fotográficas e padrões de energia eletromagnética radiante e outros fenômenos*" (American Society of Photogrametry and Remote Sensing 1980). A fotogrametria é utilizada em diversas áreas, como arquitetura, medicina, indústria, engenharia, entre outras. Quando aplicada à avaliação postural o sujeito é fotografado a uma distância pré-determinada pelo avaliador e a imagem é analisada em softwares específicos de avaliação postural (Tommaselli 1999).

2.3.3 Softwares de avaliação postural

A necessidade de quantificar as variáveis relacionadas à avaliação postural é antiga e o desenvolvimento tecnológico tem possibilitado o uso de ferramentas relativamente simples que oferecem boa resposta. Muitos pesquisadores baseados na precisão da fotogrametria associados aos avanços tecnológicos utilizam diversos programas computacionais de avaliação postural com ótimos resultados (Mercadante 2005; Furlaneto, Candotti et al. 2007; Nery 2009). As funcionalidades destes softwares assim como a capacidade de medição quantitativa permitem aos fisioterapeutas e pesquisadores não só uma avaliação precisa das alterações posturais, mas também avaliar os resultados obtidos com o tratamento.

Como a postura pode ser qualitativa e quantitativamente avaliada por meio da interpretação rigorosa de imagens fotográficas, as quais também podem ser utilizadas para monitorar os resultados do tratamento, varias empresas independentes desenvolveram aplicações de software para avaliação postural que geralmente consistem na localização de marcadores digitais em imagens fotográficas e ferramentas de medição de diversas variáveis (Ferreira, Duarte et al. 2010).

Algumas empresas com negócios orientados à quiropraxia, postura e saúde desenvolvem ou usam com êxito os softwares de avaliação postural, assim Dunk, Chung et al.(2004) relatam a utilização e aplicação destes softwares por algumas empresas como Biotonix, PosturePro e ChiroVision que desenvolveram diferentes sistemas de análise postural computadorizado envolvendo digitalização da imagem do sujeito comparando com uma postura padrão, o que permite avaliar qualquer assimetria e recomendar um programa de exercício e/ou uma intervenção clínica para corrigir aquela alteração. Os mesmos autores (Dunk, Chung et al. 2004) utilizaram uma ferramenta de diagnóstico clinico criada pela Universidade de Waterloo (Ontário, Canadá) chamada Gober, usada para digitalizar as imagens, calcular ângulos e quantificar distâncias.

O SAPO aparece no mercado como alternativa ante os eventos mencionados anteriormente.

2.3.4 SAPO

2.3.4.1 Histórico do SAPO

Em 2003, por meio de um projeto de pesquisa do Conselho Nacional de Pesquisa e Desenvolvimento (CNPq) e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), foi desenvolvido um software de livre acesso para avaliação da Postura. De acordo com SAPO(2005) e Sapo-Desktop (2012) Os objetivos do Software de Avaliação Postural (SAPO) se concentram em:

- Desenvolvimento de software livre para avaliação postural,
- Desenvolvimento de estudos metrológicos sobre avaliação postural computadorizada,
- Criação de tutoriais científicos sobre avaliação postural, e
- Criação de banco de dados com resultados de avaliação feitos por centros colaboradores.

O software é um programa de computador que faz uso de fotografias digitalizadas – biofotogrametria dos indivíduos, possibilitando a mensuração dos desvios posturais (SAPO 2011).

O SAPO foi desenvolvido na linguagem de programação JAVA (JRE6), portanto pode ser instalado tanto em Windows quanto em Linux. Foi desenvolvido em língua portuguesa (PT-BR), é de código aberto e livre. O código-fonte do SAPO

pode ser descarregado do seguinte site: http://puig.pro.br/sapo/ ou http://code.google.com/p/sapo-desktop/.

2.3.4.2 Funcionalidade do SAPO

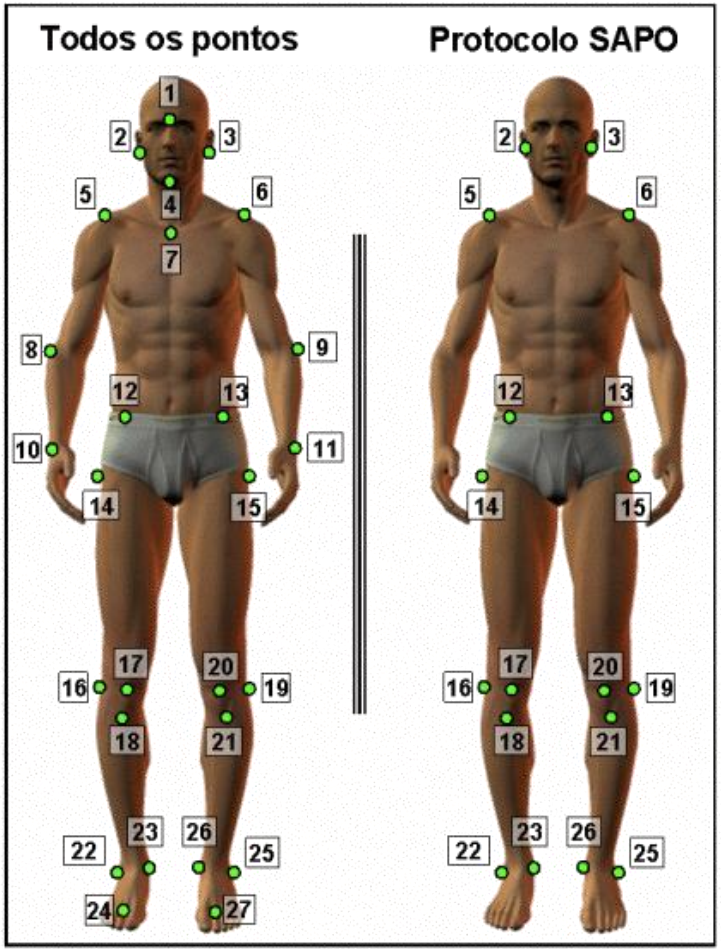
O SAPO fundamenta-se na digitalização de pontos anatômicos espacialmente definidos, possibilitando assim funções diversas tais como a calibração da imagem, utilização de zoom, marcação livre de pontos, medição de distâncias e de ângulos corporais. O método usado pelos softwares de avaliação postural (palpação dos pontos anatômicos de referência, colocação dos marcadores reflexivos e digitalização) é o mesmo método sugerido por (Mercadante 2005) e utilizado pelo Software de Avaliação Postural (SAPO) chamado o protocolo SAPO de marcação de pontos.

2.3.4.3 O Protocolo SAPO de marcação de pontos

O Protocolo SAPO de marcação de pontos é uma sugestão de pontos de marcação e medidas para avaliação postural. A escolha desses pontos foi baseada na relevância clínica, base científica, viabilidade metodológica e aplicabilidade (SAPO 2005). Apesar da existência deste protocolo previamente estabelecido o SAPO permite que o usuário defina seu próprio protocolo de marcação de pontos. O protocolo SAPO existe para cada uma das vistas a serem avaliadas: anterior, posterior, lateral esquerdo e lateral direito e é utilizado no desenvolvimento da presente dissertação.

Figura 1 - Vista anterior da marcação dos pontos anatômicos do protocolo SAPO

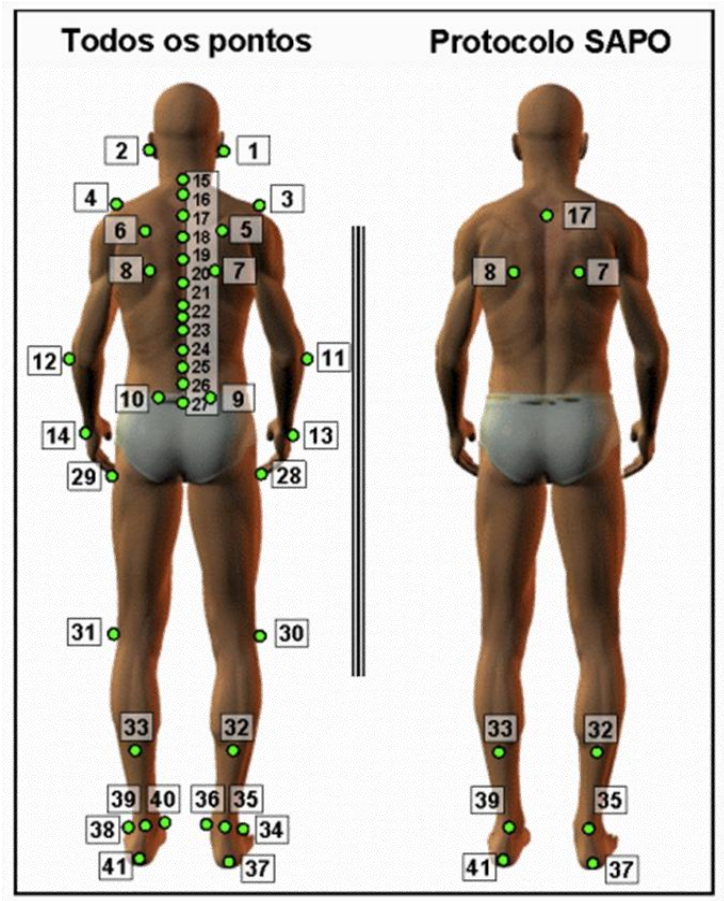
1. Glabela
2. Trago direito
3. Trago esquerdo
4. Mento
5. Acrômio direito
6. Acrômio esquerdo
7. Manúbrio do esterno
8. Epicôndilo lateral direito
9. Epicôndilo lateral esquerdo
10. Ponto médio entre o processo estilóide do rádio e a cabeça da ulna direita
11. Ponto médio entre o processo estilóide do rádio e a cabeça da ulna esquerda
12. Espinha ilíaca ântero-superior direita
13. Espinha ilíaca ântero-superior esquerda
14. Trocânter maior do fêmur direito
15. Trocânter maior do fêmur esquerdo
16. Linha articular do joelho direito
17. Ponto medial da patela direita
18. Tuberosidade da tíbia direita
19. Linha articular do joelho esquerdo
20. Ponto medial da patela esquerda
21. Tuberosidade da tíbia esquerda
22. Maléolo lateral direito
23. Maléolo medial direito
24. Ponto entre a cabeça do 2° e 3° metatarso direito
25. Maléolo lateral esquerdo
26. Maléolo medial esquerdo
27. Ponto entre a cabeça do 2° e 3° metatarso esquerdo



Fonte: www.demotu.org/sapo

Figura 2 - Vista posterior da marcação dos pontos anatômicos do protocolo SAPO

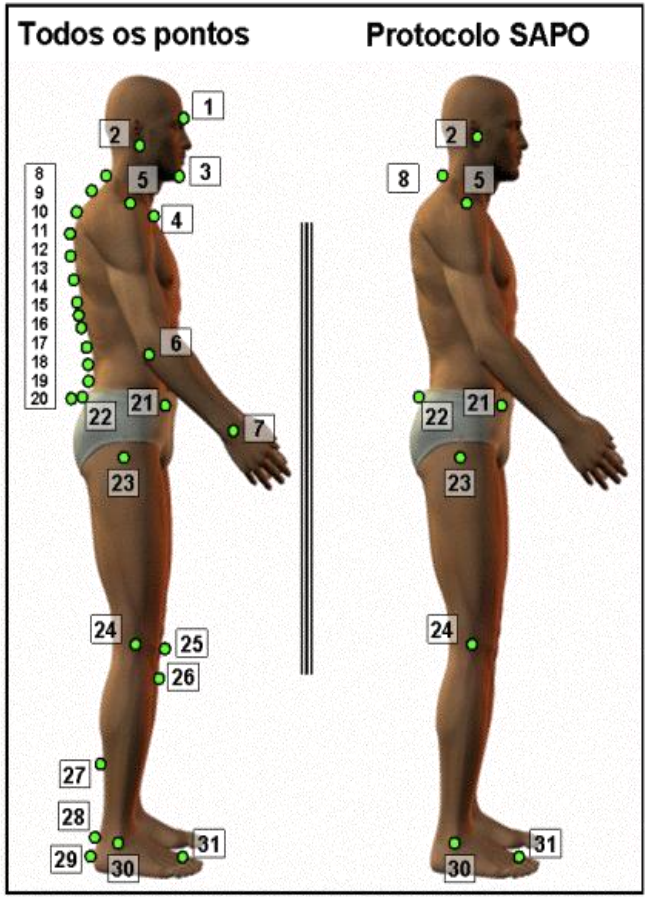
1. Trago direito
2. Trago esquerdo
3. Acrômio direito
4. Acrômio esquerdo
5. Ponto de transição entre a margem medial e a espinha da escápula direita
6. Ponto de transição entre a margem medial e a espinha da escápula esquerda
7. Ângulo inferior da escápula direito
8. Ângulo inferior da escápula esquerdo
9. Espinha ilíaca póstero-superior direita
10. Espinha ilíaca póstero-superior esquerda
11. Epicôndilo lateral direito
12. Epicôndilo lateral esquerdo
13. Ponto médio entre o processo estilóide do rádio e a cabeça da ulna direita
14. Ponto médio entre o processo estilóide do rádio e a cabeça da ulna esquerda
15. Processo espinhoso C7
16. Processo espinhoso T1
17. Processo espinhoso T3
18. Processo espinhoso T5
19. Processo espinhoso T7
20. Processo espinhoso T9
21. Processo espinhoso T11
22. Processo espinhoso T12
23. Processo espinhoso L1
24. Processo espinhoso L3
25. Processo espinhoso L4
26. Processo espinhoso L5
27. Processo espinhoso S1
28. Trocânter maior do fêmur direito
29. Trocânter maior do fêmur esquerdo
30. Linha articular do joelho direito
31. Linha articular do joelho esquerdo
32. Ponto sobre a linha média da perna direita
33. Ponto sobre a linha média da perna esquerda
34. Maléolo lateral direito
35. Ponto sobre o tendão do calcâneo direito na altura média dos dois maléolos
36. Maléolo medial direito
37. Calcâneo direito
38. Maléolo lateral esquerdo
39. Ponto sobre o tendão do calcâneo esquerdo na altura média dos dois maléolos
40. Maléolo medial esquerdo
41. Calcâneo esquerdo



Fonte: www.demotu.org/sapo

Figura 3 - Vista lateral direita da marcação dos pontos anatômicos do protocolo SAPO

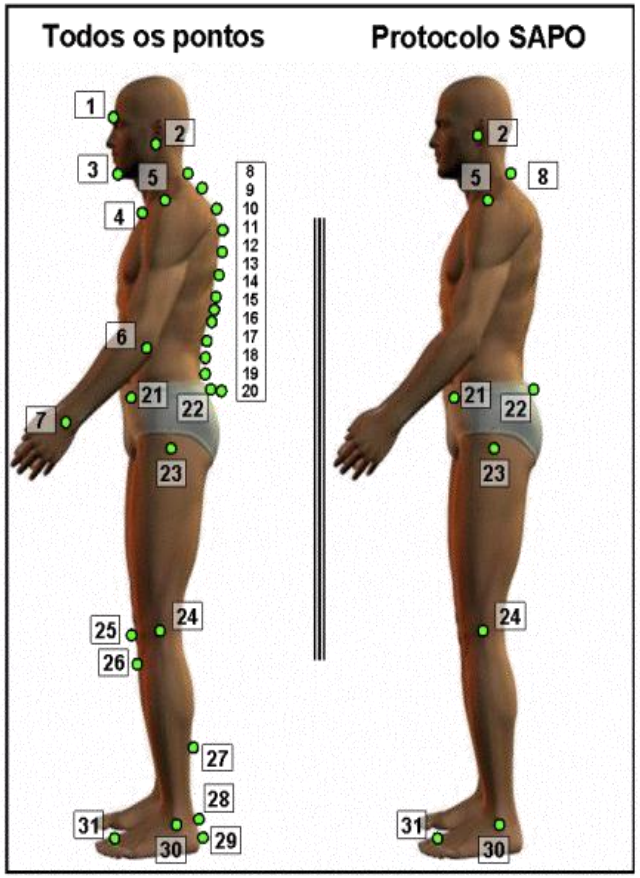
1. Glabela
2. Trago direito
3. Mento
4. Manúbrio do esterno
5. Acrômio direito
6. Epicôndilo lateral direito
7. Ponto médio entre o processo estilóide do rádio e a cabeça da ulna direita
8. Processo espinhoso C7
9. Processo espinhoso T1
10. Processo espinhoso T3
11. Processo espinhoso T5
12. Processo espinhoso T7
13. Processo espinhoso T9
14. Processo espinhoso T11
15. Processo espinhoso T12
16. Processo espinhoso L1
17. Processo espinhoso L3
18. Processo espinhoso L4
19. Processo espinhoso L5
20. Processo espinhoso S1
21. Espinha ilíaca ântero-superior direita
22. Espinha ilíaca póstero-superior direita
23. Trocânter maior do fêmur direito
24. Linha articular do joelho direito
25. Ponto medial da patela direita
26. Tuberosidade da tíbia
27. Ponto sobre a linha média da perna direita
28. Ponto sobre o tendão do calcâneo direito na altura média dos dois maléolos
29. Calcâneo direito
30. Maléolo lateral direito
31. Ponto entre a cabeça do 2° e 3° metatarso direito



Fonte: www.demotu.org/sapo

Figura 4 - Vista lateral esquerda da marcação dos pontos anatômicos do protocolo SAPO

1. Glabela
2. Trago esquerdo
3. Mento
4. Manúbrio do esterno
5. Acrômio esquerdo
6. Epicôndilo lateral esquerdo
7. Ponto médio entre o processo estilóide do rádio e a cabeça da ulna esquerda
8. Processo espinhoso C7
9. Processo espinhoso T1
10. Processo espinhoso T3
11. Processo espinhoso T5
12. Processo espinhoso T7
13. Processo espinhoso T9
14. Processo espinhoso T11
15. Processo espinhoso T12
16. Processo espinhoso L1
17. Processo espinhoso L3
18. Processo espinhoso L4
19. Processo espinhoso L5
20. Processo espinhoso S1
21. Espinha ilíaca ântero-superior esquerda
22. Espinha ilíaca póstero-superior esquerda
23. Trocânter maior do fêmur esquerdo
24. Linha articular do joelho esquerdo
25. Ponto medial da patela esquerda
26. Tuberosidade da tíbia
27. Ponto sobre a linha média da perna esquerda
28. Ponto sobre o tendão do calcâneo esquerdo na altura média dos dois maléolos
29. Calcâneo esquerdo
30. Maléolo lateral esquerdo
31. Ponto entre a cabeça do 2° e 3° metatarso esquerdo



Fonte: www.demotu.org/sapo

2.4 Validade dos métodos de avaliação postural

Os instrumentos para avaliação postural anteriormente mencionados requerem confiabilidade e acurácia na sua utilização, já que sua validação é imprescindível para possibilitar o estabelecimento de uma linguagem comum entre os profissionais das diferentes áreas, prover bases científicas para a compreensão e estudo dos problemas observados, propiciar a comparação de dados ao longo do tempo e permitir o confronto de técnicas e abordagens terapêuticas (Giglio and Volpon 2007).

Alguns ortopedistas utilizam a análise observacional qualitativa como método de avaliação postural na prática clínica, baseado nessa questão Fedorak, Ashworth et al. (2003) realizaram um estudo para verificar a confiabilidade da análise observacional das lordoses cervical e lombar; para isso recrutaram 28 profissionais (com experiência profissional entre 01 e 42 anos), com formação em quiropraxia, fisiatria, fisioterapia, reumatologia e cirurgiões ortopédicos para avaliarem as fotos de 36 indivíduos (17 com dor lombar e 19 sem dor). Os autores concluíram que não houve diferença estatisticamente significativa entre as análises observacionais dos cinco grupos de profissionais, mas que estas eram imprecisas e contrariamente à hipótese dos autores o estudo mostrou que a avaliação visual de lordoses cervical e lombar é pouco confiável.

A fotogrametria mostra maior nível de confiabilidade em seus resultados, (Sacco ICN 2007) indicando que a utilização dessa avaliação pode facilitar a quantificação das variáveis morfológicas relacionadas à postura, trazendo dados mais confiáveis do que aqueles obtidos pela observação visual. Watson and Mac Donncha (2000) mencionam que a reprodutibilidade deste processo de avaliação o torna adequado para investigar as relações entre a postura e as variáveis de saúde tais como afecções músculo-esqueléticas. Já pesquisadores como Zonnenberg, VanMaanen et al. (1996) ressaltam que o uso das fotografias permite mais objetividade na avaliação.

Apesar da utilização das fotografias na avaliação postural ser um procedimento relativamente comum é importante tomar alguns cuidados. Normalmente são observadas pequenas assimetrias que podem ser mal interpretadas se não houver alguns cuidados na aquisição e interpretação das imagens (Ferreira 2006). Vegter e Hage (2000) já indicavam a necessidade de padronizar as medições feitas com fotos, não se baseando somente na utilização do simetrógrafo.

Em relação aos softwares de avaliação postural e sua confiabilidade, autores como Dunk, Chung et al (2004) indicam que ainda são necessários maiores estudos para validar e estimar a confiabilidade de cada um desses softwares. Validações parciais foram realizadas para diversas dessas ferramentas computacionais, porém a maioria destes estudos tem avaliado somente algumas regiões específicas do corpo (não avaliações posturais globais).

Os autores Iunes, Cecilio et al. (2010) realizaram um estudo sobre análise postural quantitativa empregando a fotogrametria no qual foi utilizado um programa computacional chamado ALCimagem 2000 software em que foram obtidos resultados muito confiáveis. Já Furlaneto, Candotti et al.(2007) utilizaram o software APPID (Avaliação Postural a partir de Imagem Digital) concluindo que este forneceu informações consistentes sobre a postura dos indivíduos podendo ser utilizado como método de medição.

Com respeito ao SAPO foram encontrados diversos estudos sobre a sua utilização. Braz (2008) indicou que o SAPO é um programa de uso relativamente simples e gratuito que fornece além das medidas lineares, valores angulares. Ferreira, Duarte et al.(2010) analisaram a sua acurácia indicando que o software é preciso na medição de ângulos e distâncias inter e intra observador, devendo ser considerado como uma ferramenta útil e confiável para mensurar a postura. Alguns estudos sobre a sua aplicabilidade como em Pereira e Medalha (2008) que baseados na confiabilidade desta ferramenta a utilizaram para realizar uma avaliação postural. Farias, Rech et al. (2009) analisaram a metodologia de avaliação postural do SAPO, implementando-o com sucesso no projeto de extensão fisioterapia orientada à tarefa em portadores de sequela de Acidente Vascular Encefálico (AVE), onde os autores concluíram que o instrumento utilizado é capaz de avaliar o paciente eficientemente.

Baseado nas informações anteriores decidimos continuar com o projeto de pesquisa criando o programa de avaliação postural ApLoB tendo como parâmetro o SAPO.

2.5 Funcionalidade do sistema

Revisando na literatura as experiências de diferentes pesquisadores com software de avaliação postural (Dunk, Chung et al. 2004; Ferreira 2006; Furlaneto, Candotti et al. 2007; Iunes, Cecilio et al. 2010), assim como a experiência adquirida baseada no desenvolvimento do SAPO (software de Avaliação Postural), podemos indicar que um software para avaliação da postura humana com relevância comparável ao software SAPO deve apresentar as seguintes funcionalidades:

a) **Carregar imagens**: parte do programa onde uma ou mais imagens a serem investigadas são carregadas no programa. Opcionalmente o programa permite selecionar quadros específicos de vídeos para a análise;

b) **Processamento de imagem**: parte do programa onde a imagem pode ser tratada para melhor avaliação. O processamento inclui selecionar apenas parte da imagem, rotação e aplicação de filtro para mudar as características da imagem;

c) **Calibração da imagem**: parte do programa em que é determinada uma função de transformação entre imagem (fotografia) e objeto real (corpo fotografado). Isto é, estabelece-se uma relação matemática entre as dimensões do espaço real (em metros, por exemplo) e as dimensões da imagem (em pixels, por exemplo).

d) **Medição livre de posição, distâncias, ângulos e área na imagem**: parte do programa onde podem ser medidas livremente na imagem grandezas geométricas como posição, distância, ângulos.

e) **Digitalização de pontos na imagem**: parte do programa onde após especificação da representação do que se pretende avaliar (por exemplo, quais pontos representativos dos segmentos do corpo serão avaliados) é feita a digitalização destes pontos;

f) **Cálculo de grandezas físico-matemáticas a partir dos pontos reconstruídos**: parte do programa onde podem ser calculados posição, distância, ângulos, área e centro de massa do corpo humano a partir da representação e de modelo das características inerciais do corpo sob estudo.

g) **Visualização dos pontos reconstruídos e das grandezas calculadas**: parte do programa onde os pontos reconstruídos e as grandezas calculadas podem ser visualizados a partir de planilhas de dados e gráficos.
h) **Geração de relatórios**: parte do programa onde os resultados principais podem ser apresentados junto com a análise destes dados pelo usuário para efeitos de laudo da avaliação.
i) **Documentação**: parte do programa onde suas funcionalidades são explicadas.

2.6 Método de Desenvolvimento de Software

O presente trabalho alinha suas atividades, tarefas e processos aos modelos e padrões estabelecidos na Engenharia de Software:

2.6.1 Engenharia de Software

A engenharia de software é uma disciplina da engenharia relacionada com todos os aspectos da produção de software desde os estágios iniciais de especificação do sistema até sua manutenção depois que este entra em operação (Sommerville 2007). Já Pressman (2005) a define como os métodos, ferramentas e procedimentos que possibilitam o controle do processo de desenvolvimento do software e oferece uma base para a construção produtiva de alta qualidade do mesmo.

Tendo em conta os fundamentos científicos na engenharia de software devemos ter presente os processos e modelos de desenvolvimento computacional.

2.6.2 Processo de desenvolvimento de software

O processo de desenvolvimento de software é definido por Sommerville (2007) como o conjunto de atividades que leva a produção de um produto de software. Essas atividades envolvem o desenvolvimento de software propriamente dito usando uma linguagem de programação.

Justamente um dos problemas encontrados é o vasto conteúdo sobre os procedimentos de desenvolvimento de software estabelecidos por instituições como a ISO (*International Organization for Standardization*), a IEC (*International*

Electrotechnical Comission), a Associação Brasileira de Normas Técnicas (ABNT) e a IEEE (*Institute of Electrical and Electronics Engineers*) para mencionar as mais conhecidas, cada uma com suas correspondentes versões para desenvolvimento e posteriores melhoras (versões) dos mesmos. Podemos observar que com o decorrer do tempo, seja por pequenas falhas técnicas, por melhorias de usabilidade ou por prolongar a aplicabilidade das suas especificações a ISO e a IEC realizaram uma parceria e geraram as Normas ISO/IEC, dentre as quais destaca-se a ISO/IEC 9126 referente à qualidade do produto de software e enquadrada no modelo de qualidade das normas da família 9000.

A Associação Brasileira de Normas Técnicas (ABNT) lançou depois da união das Normas ISO/IEC sua norma correspondente, a NBR ISO/IEC 9126-1, que se refere à Engenharia de Software – Qualidade de Produto onde o processo de desenvolvimento de software é também mencionado.

Finalmente temos a IEEE (*Institute of Electrical and Electronics Engineers*) que tem a norma STD 1471–2000*, Recommended Practice for Architectural Description of Software-intensive Systems* que seria a “prática recomendada para descrição da arquitetura de sistemas intensivos de software”. A ISO adotou esse procedimento e em Julho 2007 lançou sua norma equivalente ISO/IEC 42010:2007 (International Organization for Standardization 2001), sendo que o texto deste standard é idêntico ao IEEE 1471:2000 (Institute of Electrical and Electronics Engineers 2011).

A existência de instituições reconhecidas na criação de normas e processos preocupadas a respeito dos procedimentos de desenvolvimento de software mostra não só a complexidade, mas também a importância dos mesmos.

Sommerville (2007) indica que os processos de software são complexos e, como todos os processos intelectuais e criativos, dependem do julgamento humano. Deste modo, podemos considerar que não existe um processo melhor que outro e sim, um mais adequado para nossa realidade, que seja baseada em nossa infra-estrutura de hardware, na experiência em desenvolvimento, profundeza de conhecimentos, e no alcance de nossas ferramentas. Justamente Berzins (1991) indica que existem diferentes pontos de vista nos processos de desenvolvimento de software, enquanto os nomes, limites e alcances nos diferentes estágios do processo diferem de autor para autor, todos concordam que o desenvolvimento de software consiste em muitos e variados, mas altamente qualitativos tipos de atividades.

2.6.3 Modelos de Processos (de desenvolvimento) de software

À medida que a nossa capacidade de produzir software aumentou, também cresceu a complexidade dos sistemas requeridos e apareceram novas tecnologias. Estas características que resultaram da convergência da evolução do hardware com as tendências tecnológicas do mercado resultaram na necessidade de modelos que nos sirvam de guia e que nos provejam o controle e a estabilidade do nosso projeto de desenvolvimento: os modelos de processos.

Pressman (2005) define estes como a distinta série de atividades, tarefas, etapas e produtos que são requeridos pelo engenheiro no desenvolvimento de software de alta qualidade. Sommerville (2007), por sua vez, indica que é uma representação sob determinada perspectiva e, dessa forma, fornece somente informações sobre esse processo. Sommerville (2007) indica que não são descrições definitivas, ao contrário, são abstrações do processo que podem ser usadas para explicar diferentes abordagens para o desenvolvimento de software. Eles podem ser considerados como frameworks de processo que podem ser ampliadas e adaptadas para criar processos mais específicos de engenharia de software. Já Pressman (2005) menciona que esses processos não são perfeitos, mas eles proveem um útil mapa de rota para o trabalho da engenharia de software.

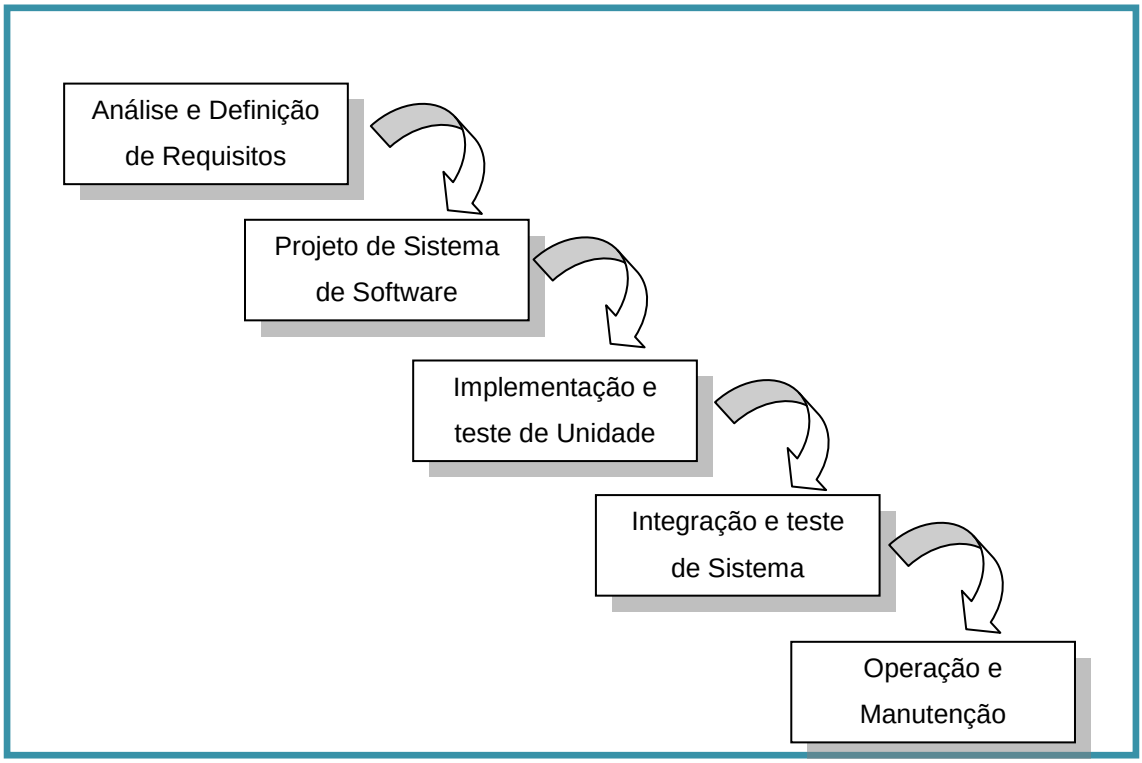
Assim, cada modelo de processo deve ser adaptado para que seja utilizado de forma eficaz para um determinado projeto de software. Cada modelo tem suas vantagens e desvantagens, eles não são mutuamente exclusivos e na prática frequentemente são combinados, fato que acontece no nosso projeto. Entre os mais conhecidos modelos de processo temos:

- Modelo em cascata
- Desenvolvimento evolucionário
- Engenharia de software baseada em componentes
- Iteração de processo:
    - -Entrega Incremental
    - -Desenvolvimento Espiral

Os modelos acima citados serão explicados a seguir.

2.6.3.1 Modelo em cascata, linear o clássico.

Figura 5 - Modelo Cascata baseado em Royce, 1970



Fonte: Royce, 1970

O modelo cascata descreve um método de desenvolvimento que é linear e sequencial, uma vez que uma fase de desenvolvimento é completada o desenvolvimento prossegue para a seguinte fase, sem retorno. Pressman (2005) indica que este modelo considera as atividades fundamentais do processo e assim, o desenvolvimento move do conceito, através do projeto (design), implementação, teste, instalação, descoberta de defeitos e termina com a operação e manutenção.

Os autores Pressman (2005) e Sommerville (2007) mencionam como desvantagens o fato deste modelo ser o que mais gasta tempo produtivo, e que os projetos raramente seguem o fluxo sequencial que o modelo propõe. Pressman (2005), também se percebe que este modelo é muito rígido e que as atividades de requisitos, análise e desenho têm que ser muito bem dominadas, uma vez que não são permitidos erros (Wilson de Pádua Filho 2003).

2.6.3.2 Desenvolvimento evolucionário

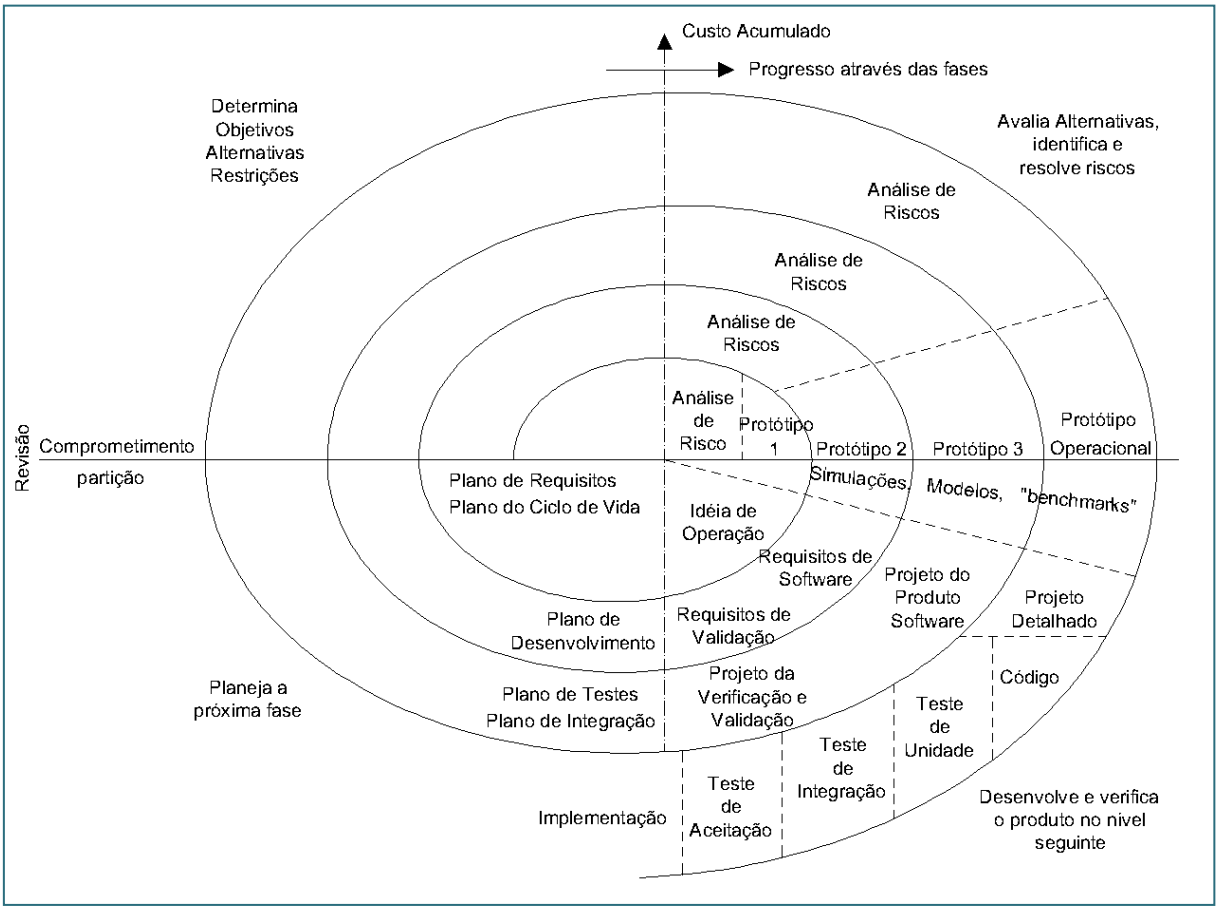
O desenvolvimento evolucionário baseia-se na idéia de desenvolvimento de uma implementação inicial, expondo o resultado aos comentários do usuário e refinando esse resultado por meio de várias versões até que seja desenvolvido um sistema adequado (Sommerville 2007). A produção destas versões iniciais chamadas protótipos permite verificações e experimentações para se avaliar as características do sistema antes de ser realmente construído. Devido ao fato de se gerar de um protótipo que é entregue ao final de cada versão, é que este modelo é conhecido também como Prototipagem.

2.6.3.3 Espiral

O modelo espiral apresentado por Boehm em 1988, em vez de representar o processo de software como uma sequencia de atividades com algum retorno entre uma atividade e outra é representada como uma espiral. Cada loop na espiral representa uma fase do processo de software (Sommerville 2007). Este modelo foi desenvolvido para abranger as melhores características tanto do ciclo de vida clássico como da Prototipagem, acrescentando ao mesmo tempo um novo elemento, que é a análise de riscos que falta a esses paradigmas. O modelo define quatro importantes atividades representadas por quatro quadrantes:

1. Definição de objetivos: Determinação dos objetivos, alternativas e restrições.
2. Análise de riscos: Avaliação e redução de riscos
3. Desenvolvimento e validação (Engenharia)
4. Planejamento: o projeto é revisado para prosseguimento ao próximo loop.

Figura 6 - Modelo em espiral do processo de software de Boehm



Fonte: IEEE (Institute of Electrical and Electronics Engineers), 1988

2.6.3.4 Engenharia de software baseada em componentes

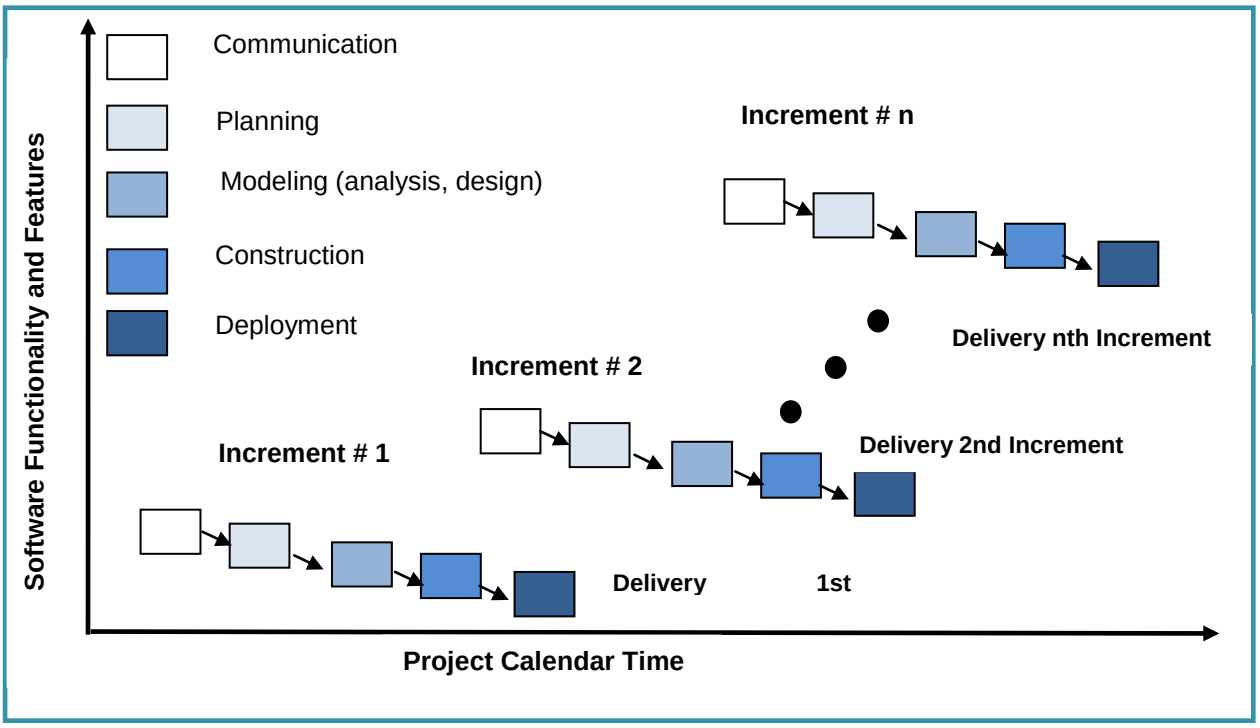
Um componente é uma unidade de composição que pode ser desenvolvida independentemente. Assim temos como exemplos de componentes: código compilado, código fonte, classes e métodos, interfaces de usuário, Plug-In, planos, bibliotecas, estratégias e regras arquiteturais, diagramas, etc.

A engenharia de software baseada em componentes (*component-based software engineering* - CBSE) é um modelo onde as aplicações são construídas a partir de componentes de software pré-encapotados chamados de classes (Pressman 2005). A abordagem deste modelo é orientada ao reuso de uma grande base de componentes de software e algum framework de integração deles (Sommerville 2007). Essa reutilização é importante por que permite um aumento na produtividade ao possibilitar adicionar funcionalidades pré-existentes na produção do novo software.

2.6.3.5 Entrega incremental

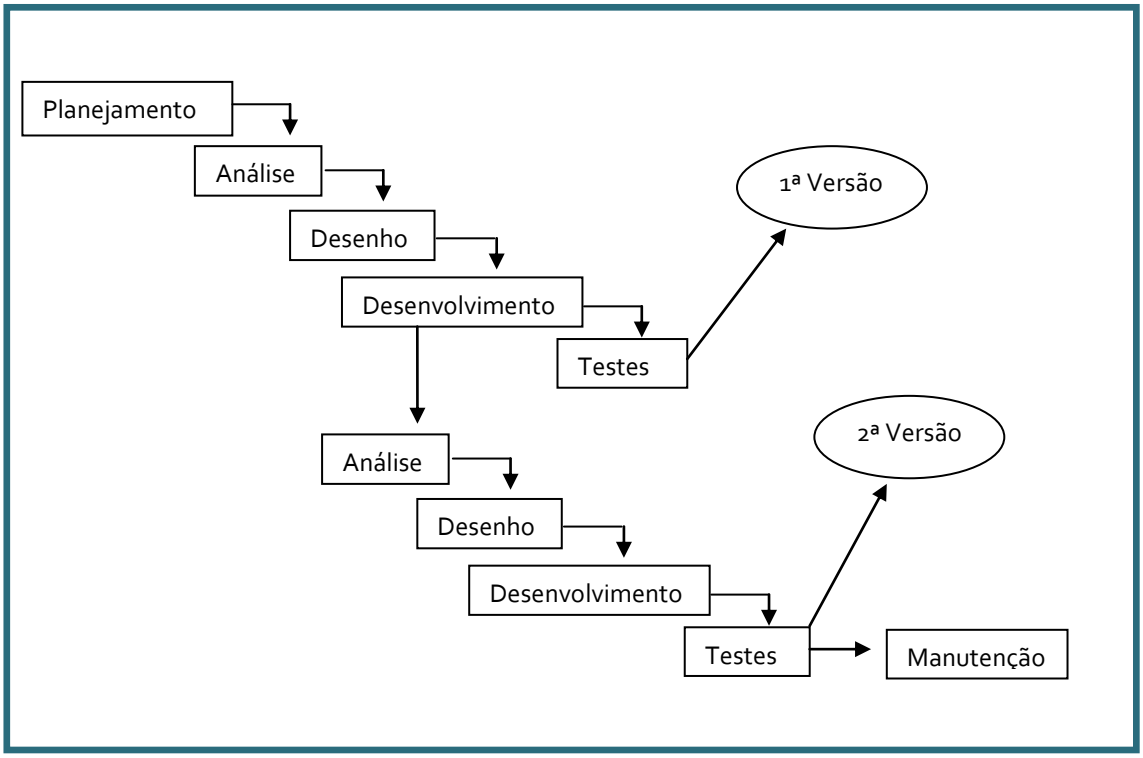
O modelo incremental combina elementos do modelo cascata sendo aplicado de maneira interativa (Pressman 2005). Sommerville (2007) indica que o modelo de processo incremental é uma abordagem que combina as vantagens do modelo cascata e o evolucionário. A proposta é dividir o trabalho em partes menores ou iterações, sendo que cada iteração resultará num incremento. Iterações são passos em fluxo de trabalho e incrementos são crescimentos do produto (Macoratti 2005). As figuras 7 e 8 são exemplos gráficos do modelo incremental.

Figura 7 – Modelo Incremental de Pressman



Fonte: Pressman, 2005

Figura 8 – Modelo Incremental de Macoratti



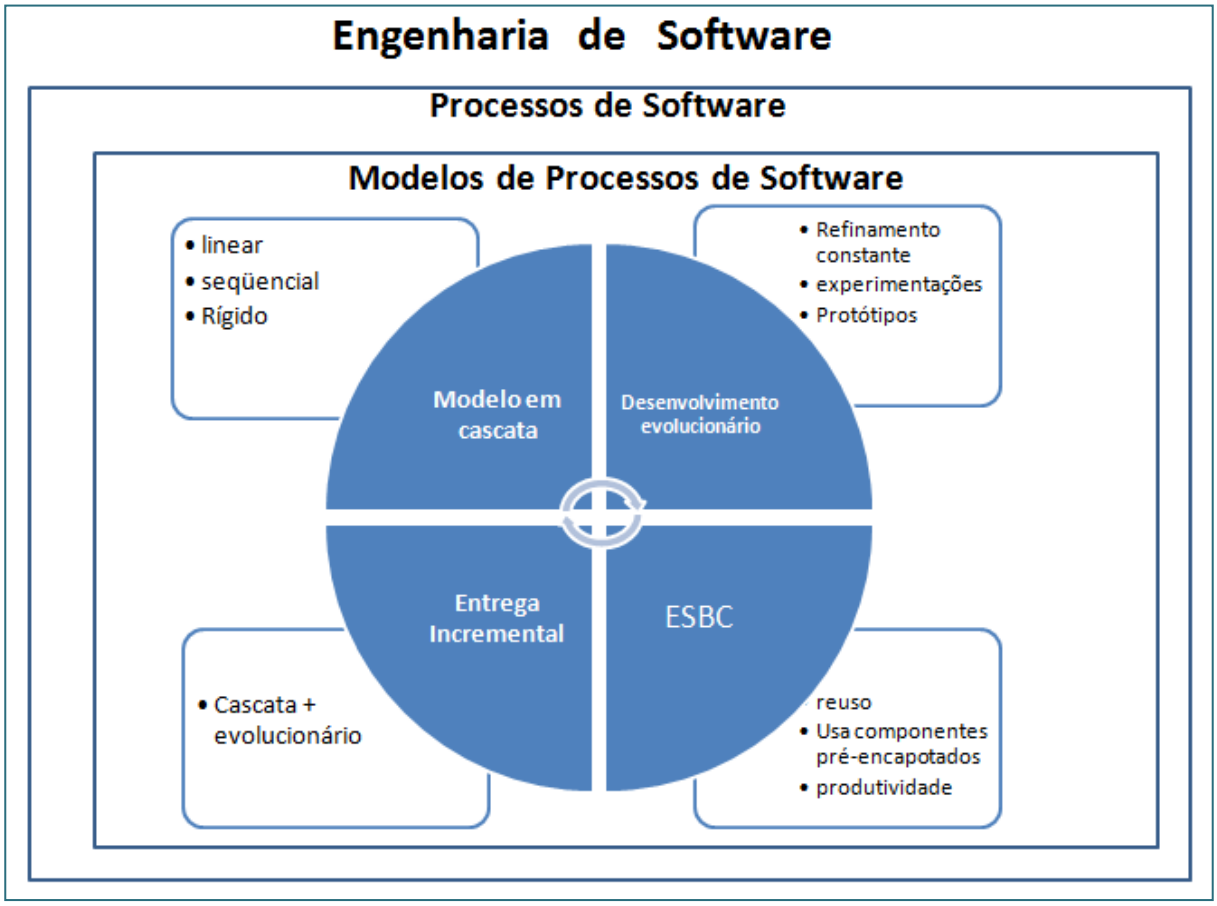
Fonte: Macoratti 2005

2.7 Resumo dos modelos de Processo de Software

Na Figura 9 mostram-se, a título de resumo, os modelos expostos anteriormente com as suas principais características e a sua correspondência na engenharia de software. Indicamos que existem diversos modelos de processos de software, mas são estes aqui expostos os que foram escolhidos para gerar o nosso modelo.

No capítulo 3-Metodologia, correspondente ao desenvolvimento de nosso projeto, indica-se o modelo escolhido explicando-se o mesmo e as razões da escolha.

Figura 9 - Resumo dos modelos de Processos de Software



Fonte: Elaboração Própria

2.8 Linguagem de programação

Uma linguagem de programação serve como meio de comunicação entre o individuo que deseja resolver um determinado problema e o computador escolhido para ajudá-lo na solução (Price and Toscani 2001). Segundo Silva e Assis (1998) as Linguagens de Programação podem ser divididas em duas categorias:

- Linguagem de Baixo Nível: é uma linguagem mais próxima da Linguagem de Máquina, usando código hexadecimal, ou seja, mais próxima do hardware.

- Linguagem de Alto Nível: são linguagens mais afastadas da Linguagem de Máquina, através do uso de abstrações cada vez mais complexas.

As linguagens de programação mais utilizadas hoje são as de Alto Nível, consideradas mais próximas às linguagens naturais ou ao domínio da aplicação em questão. Para que o computador as compreenda e se tornem operacionais devem ser traduzidas para linguagem de máquina. Essa conversão é realizada através de

sistemas especializados: os compiladores ou interpretadores (Price and Toscani 2001)

Assim, nas linguagens interpretadas um programa é executado instrução a instrução, ou seja, cada comando é primeiro traduzido para a linguagem de máquina, para somente em seguida ser executado. Nas linguagens compiladas um programa é executado somente quando toda a tradução foi completada. A compilação de um programa fonte (texto escrito diretamente na linguagem de alto ou médio nível) prevê que o mesmo seja traduzido para a linguagem de máquina correspondente antes da execução (Silva and Assis 1998).

2.8.1 Python

É uma linguagem de programação de Alto Nível (VHLL – *Very High Level Language*). A proposta de programar em Python é de gerar software livre (gratuito, código aberto), multiplataforma (Python roda em Windows, Linux, Mac, Palm, celulares e outros sistemas) e principalmente com clareza de sintaxe, que hoje é responsável pela alta produtividade obtida em Python (Labaki 2008). Justamente sobre a sua sintaxe, facilidade de entendimento e percepção Sanders e Langford (2008) fizeram comparações com outras linguagens de programação e mencionam a capacidade de poder formular algoritmos com um estilo funcional. A respeito do seu desempenho Cai, Langtangen et al.(2005) fizeram avaliações do código feito em Python para aplicações computacionais científicas paralelas e seriais, recomendando finalmente sua utilização para essas aplicações. Outra pesquisa, já de tipo comparativo sobre estratégias de otimização de velocidade de código (Li and Li 2010) indicou que os scripts desenvolvidos em linguagem Python oferecem alto desenvolvimento e eficiência, assim como uma versatilidade de bibliotecas.

As características do programa são as seguintes (Python 2011):

- É um software de distribuição gratuita (Free software), com código fonte disponível (Open Source software). Sua linguagem é simples e de fácil aprendizado;
- É uma linguagem altamente modular, isto significa que partes do programa a ser desenvolvido já podem ter sido criadas por terceiros sendo possível a incorporação das mesmas através da programação por componentes,

fato que possibilita a economia do tempo de trabalho. Fornece um ambiente de interatividade para programação (similar ao que o programa Matlab oferece);

- É um ambiente poderoso para programação numérica (incluindo manipulação de matrizes) e geração de gráficos;
- Permite fácil interatividade com rotinas escritas em outras linguagens;
- Suporta o desenvolvimento de conjuntos de funções voltadas para aplicações especificas (os chamados toolboxes);
- Permite o desenvolvimento de interfaces gráficas mantendo o ambiente de interatividade para programação.

A popularidade do Python é crescente, a qual pode se apreciar nas estatísticas a respeito da notoriedade do programa ou das quantidades de postagens em sites especializados em estatísticas na internet, onde figura no primeiro lugar em crescimento no mercado de desenvolvimento de software e quinto em tendência de uso (Ertl 2011). O programa possui diversos materiais on-line assim como cursos e bibliotecas referenciadas ao tema (Python Software Foundation, 2010).

Por estas características apresentadas e com o objetivo de criar um software livre com metodologia orientada a objetos, é que o Python é a linguagem escolhida para o desenvolvimento de nosso programa computacional de avaliação postural com código aberto e gratuito.

2.8.2 PyQt

Qt é um sistema multiplataforma para desenvolvimento de programas de interface gráfica. A empresa responsável pela criação é a Trolltech, da Noruega, depois foi adquirida pela Nokia (Qt 2011), já a versão correspondente para Python, PyQt, é desenvolvida pela Riverbank, que lançou uma versão da biblioteca sob as licenças GPL em todas a as plataformas suportadas incluindo a plataforma Windows (Riverbank 2011)

2.9 Modelagem de software

A Modelagem visual é o processo de descrever o sistema a ser desenvolvido graficamente. Isto nos permite apresentar detalhes essenciais de um problema complexo e filtrar fora os detalhes dispensáveis, alguns autores como (Gennari and

Reddy 2000) indicam que a maior parte dos produtos de software desenvolvidos para a área médica-informática tem insuficientes esforços gastos em quanto ao desenho na fase de produção, e que a má concepção muitas vezes leva a sistemas que ou não são bem aceitos ou muito menos eficazes do que poderiam ser. Isto se deve ao fato do desenho de aplicações de software ser a maior, mais complicada e mais custosa atividade que envolve um projeto informático (Moran and Carroll 1996).

Assim, a modelagem é uma parte central de todas as atividades que levam à implantação de um bom software. Construímos modelos para comunicar a estrutura e o comportamento desejado do nosso sistema, para visualizar e controlar a arquitetura do sistema, para compreender melhor o sistema que estamos construindo muitas vezes expondo oportunidades de simplificação e reutilização e para ministrar o risco. (Booch, Rumbaugh et al. 1998)

(Pressman 2005) indica que para o desenvolvimento de uma aplicação usamos os procedimentos usuais da engenharia de software que abrange um conjunto de três elementos fundamentais: métodos (o “como fazer” o software,), as ferramentas (que oferecem apoio aos métodos nomeados anteriormente), e os procedimentos (elo de ligação entre os métodos e as ferramentas, como documentos, formulários, controles). Para ter ótimo controle sobre todos esses procedimentos é necessário realizar a modelagem do software.

2.9.1 UML

A *Unified Modeling Language* (UML) é uma linguagem proposta inicialmente como a unificação das severas diferenças entre as notações visuais e técnicas de modelagem usadas para o desenho de sistemas, que se tornou um padrão industrial para aplicações de modelagem de dados e comunidades de engenharia de software (Booch, Rumbaugh et al. 1998).

Na literatura procurada observamos a importância do UML na engenharia de software, assim (Chen, Wang et al. 2009) mencionam que é a ferramenta mais freqüentemente usada para a análise e modelagem de requerimentos, e (Taylor 2010) indica que é o próprio coração da arquitetura de software, já que define o *como* os sistemas são desenhados e construídos, e (Z.M.Ma, Zhang et al. 2011) indicam que o UML é o formalismo de sistemas computacional mais amplamente aceito para a análise e desenho de software.

Assim a linguagem de modelagem unificada (UML) é nossa notação de arquitetura de software.

2.10 Ferramentas CASE

CASE (*Computer-Aided Software Engineering*) que significa “Engenharia de Software Auxiliada por Computador” são ferramentas auxiliadoras de desenvolvimento e manutenção nas atividades na engenharia de software (Gane 1990).

(Garcia-Magarino, Fuentes-Fernandez et al. 2010) indicam sobre as ferramentas CASE, que com a crescente complexidade dos projetos de software faz-se necessário o uso de ferramentas de apoio generalizado, já que elas ajudam a gerenciar os artefatos envolvidos no desenvolvimento e do próprio processo. Sobre o beneficio de integrar as ferramentas CASE com sistemas orientados a objetos (OOS) já há quase 15 anos (Holsing and Yen 1997) indicaram que podem prover uma maior efetividade no modelo de desenvolvimento de software, resultando assim em um incremento na produtividade, reduzindo custos no desenvolvimento e melhorando a qualidade do produto (confiabilidade, usabilidade e performance) . A abrangência das ferramentas CASE é extensa, assim (Fisher 1991) indica que inclui muitas ferramentas de desenvolvimento de software, como compiladores, depuradores, análises de desempenho, sistemas de controle de código, análise de requerimentos, especificações de desenho e engenharia reversa. No presente, referente ao seu uso no desenvolvimento de software (Garcia-Magarino, Fuentes-Fernandez et al. 2010) mencionaram que as equipes de desenvolvimento costumam trabalhar com diversas destas ferramentas, mesmo em um mesmo projeto, pois apresentam características de uso diferentes que respondem a necessidades específicas.

2.10.1 Rational Rose

Rational Rose é uma ferramenta CASE orientada à modelagem visual e ao desenvolvimento de software usando o *Unified Modeling Language* (UML) permitindo assim desenvolvimento de aplicações de software, modelagem de informação, estabelecimento de modelo de negócios, desenho de serviços web e

componentes de aplicações (IBM developerWorks Resource Center 2004), a palavra ROSE quer dizer Rational Object Oriented Software Engineering.

O Rational Rose é projetado para fornecer ao desenvolvedor de software um conjunto completo de ferramentas de modelagem visual para o desenvolvimento de soluções robustas, eficientes em aplicações de ambientes cliente/servidor, redes distribuídas e ambientes de sistemas em tempo real (Quatrani 1999). Rational Rose é nossa ferramenta CASE para a modelagem do nosso programa.

3. Metodologia

Neste capítulo se discute o desenvolvimento do programa de avaliação postural denominado ApLoB (Avaliação Postural do Laboratório de Biofísica) o qual foi desenvolvido como parte desta pesquisa.

3.1 Requisitos levantados para o desenvolvimento do ApLoB

A seguir são descritos os requisitos funcionais levantados para a implementação do trabalho:

a) Carregar imagens.
b) Processar imagem (rotar, filtros de imagem).
c) Calibração da imagem.
d) Medição livre de posição, distâncias, ângulos e área na imagem.
e) Digitalização de pontos na imagem
f) Cálculo de grandezas físico-matemáticas a partir dos pontos reconstruídos
g) Visualização dos pontos reconstruídos e das grandezas calculadas
h) Geração de relatórios.
i) Documentação

3.2 Proposta do projeto de ApLoB

A proposta do ApLob consiste em construir uma aplicação que cumpra com os requisitos levantados anteriormente, apresentando como características genéricas:

- Utilizar procedimentos físico-matemáticos reconhecidamente aceitos pela comunidade científica para análise do movimento;
- Ser aberto e gratuito;
- Ser baseado em soluções gratuitas e de fácil acesso;
- Possibilitar desenvolvimento em colaboração;
- Permitir programação de forma relativamente simples;
- Permitir ao usuário fácil customização para atender suas necessidades específicas;

O aplicativo ApLoB, resultado deste trabalho, deverá servir de base para desenvolvimentos futuros, assim como permitir acrescentar extensões e funcionalidades no mesmo.

3.3 Materiais e Métodos

Esta subseção descreve as ferramentas utilizadas no projeto de desenvolvimento do ApLoB:

Utilizou-se o Microsoft Project 2007 como ferramenta de gerenciamento das atividades do projeto, assim como gestor dos diagramas de Gantt para controlar os processos e tarefas; UML (*Unified Modeling Language*) linguagem standard no ambiente de desenvolvimento de software para modelagem de dados; Rational Rose Enterprise Edition 2002 como ferramenta CASE (*Computer-Aided Software Engineering*) no auxilio para o trabalho da modelagem de software da aplicação, esta ferramenta foi escolhida pela grande facilidade de integração que possui com o UML; Python como linguagem de programação base para a criação do programa e as suas bibliotecas de computação científica Guidata, Pil, Guiqwt, Numpy, o Guiqwt é uma biblioteca de Python para plotagem de dados 2D (visualização de curva/imagem e ferramentas relacionadas) para computação interativa e de desenvolvimento de aplicações de processamento; PyQt conjunto de ferramentas para a criação de interfaces gráficas; Enterprise Architech Ver. 9.2 como ferramenta para fazer engenharia reversa no Guiqwt e obter as classes em formato UML da aplicação; e finalmente o Spyder, que é o ambiente de desenvolvimento de computação científica em Python e que foi escolhido pela capacidade de gestão gráfica e de código de todas a ferramentas anteriormente mencionadas.

3.4 Descrição da metodologia empregada para elaboração ApLoB

Na revisão literária se expõem os conceitos de Engenharia de Software que dão um marco de trabalho para o desenvolvimento do ApLoB, usando como ferramenta o Microsoft Project 2007 no qual se obtém as tarefas mais relevantes e suas principais atividades que o conformam mostradas no Diagrama de Gantt (Tabela 1).

O presente trabalho está composto por 12 etapas maiores, sendo que cada uma delas está subdividida em várias atividades.

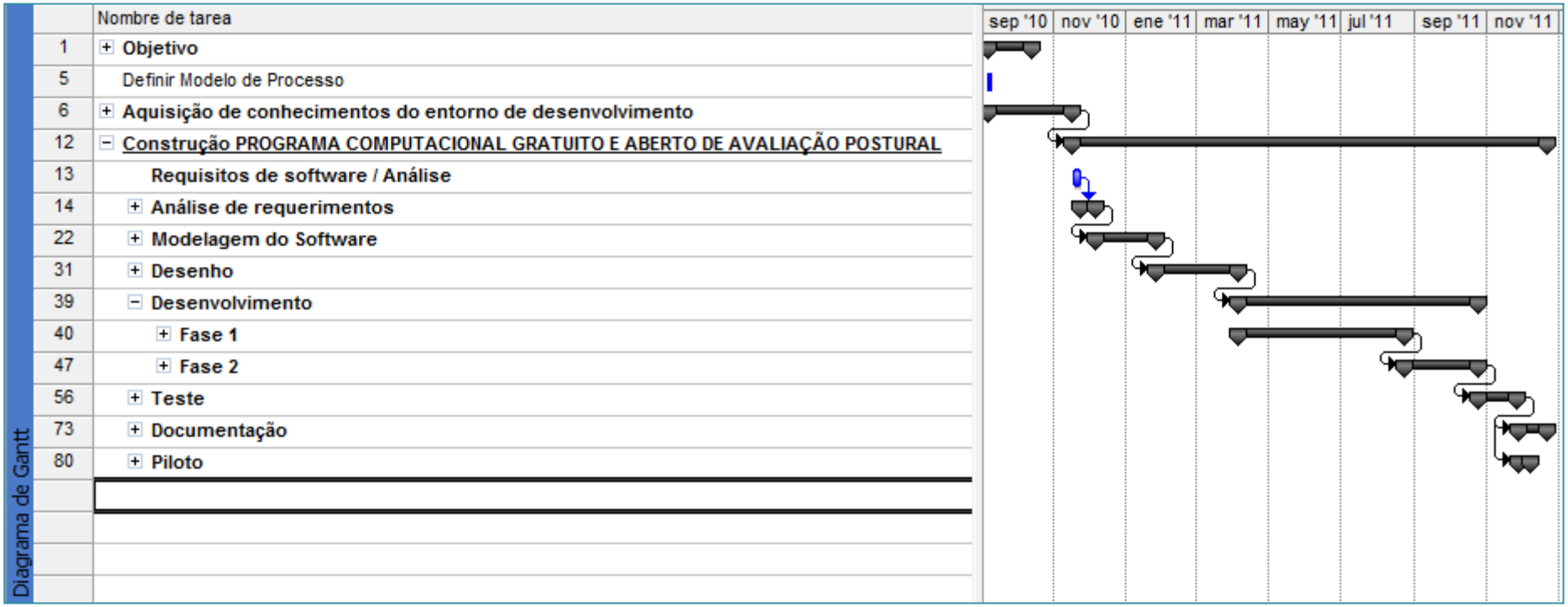
As etapas são as seguintes:

- Determinação dos requisitos de software
- Objetivo

- Aquisição de conhecimentos do entorno de desenvolvimento
- Análise de requerimentos
- Modelagem do Software
- Desenho
- Desenvolvimento
- Teste
- Documentação
- Piloto

Identificar as tarefas e sub-tarefas correspondentes para posteriormente escrever elas no Project 2007 permitiu ter uma visão dos objetivos do projeto assim como as demandas necessárias, identificar os “stakeholders” que são as pessoas envolvidas no projeto e determinar a equipe envolvida no mesmo.

Tabela 1 Diagrama de Gantt Resumido do Projeto



Fonte: Elaboração Própria em Microsoft Project 2007

3.5 O modelo de Processo de Software do ApLoB

Sommerville (2007) e Pressman (2005) indicam que os modelos são uma representação sob determinada perspectiva e não descrições definitivas, ao contrário são abstrações do processo, por isso não são perfeitos mas proveem um útil mapa para a rota de trabalho.

Na atualidade temos diferentes modelos de processo, cada uma possui diferentes características tanto na sua função como na sua aplicação, assim eles escolhidos por diferentes motivos, como a quantidade de desenvolvedores, gestão do risco, atendimento do cliente, qualidade de software entre muitos outros, por isso indicar as características, utilização e funcionalidades formariam parte de outra pesquisa e tema que não correspondem ao presente trabalho.

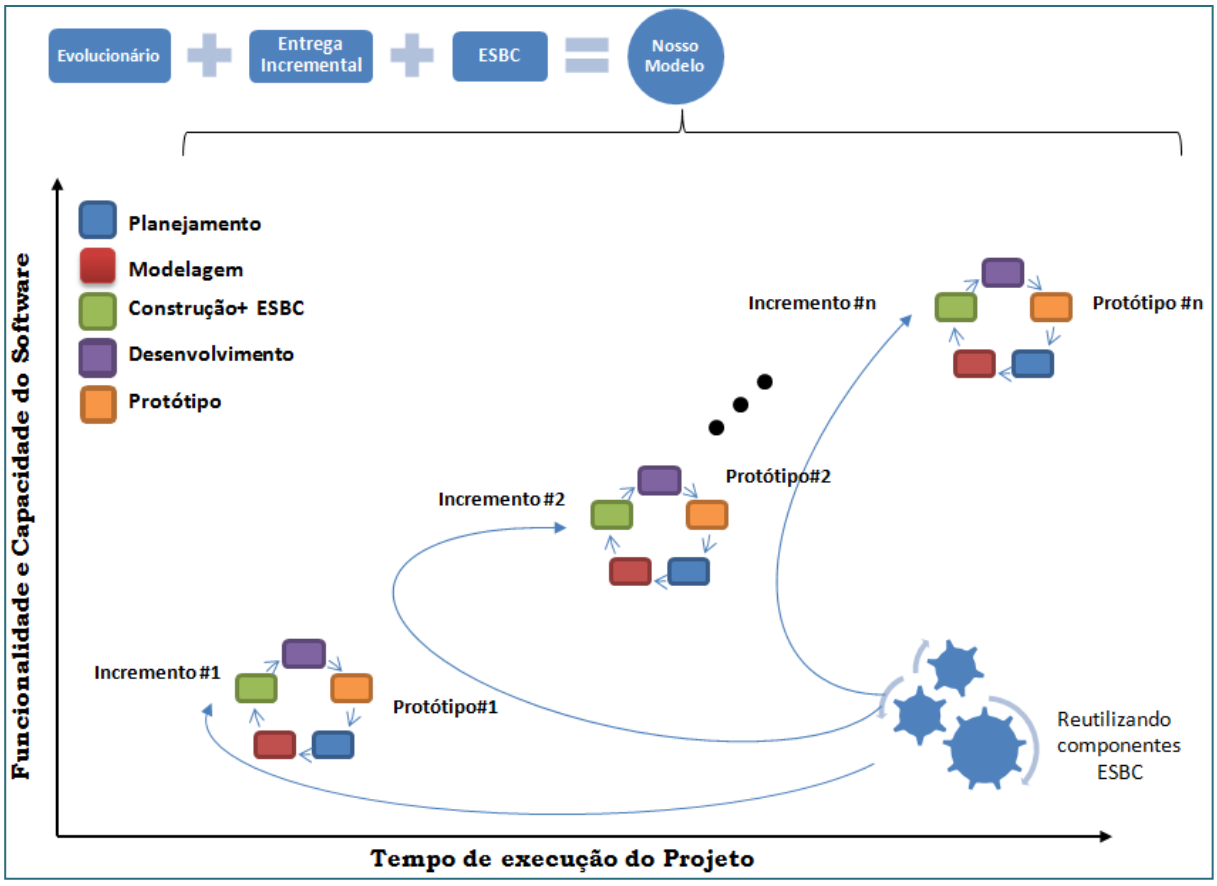
Para construir o nosso projeto foram estabelecidas as etapas cíclicas pelas que deve passar e que são: Planejamento, Modelagem, Construção, Desenvolvimento e Protótipo, tendo como resultado um Protótipo do ApLoB. Na figura 10- Modelo Utilizado em nosso projeto podem-se observar estas Etapas graficamente. Deve-se indicar que estas etapas são estabelecidas mais por uma adequação dos processos ao tempo de trabalho assim como a disponibilidade, experiência e conhecimentos da equipe.

Nosso modelo de desenvolvimento é uma combinação dos modelos: evolucionário, entrega incremental e baseado em componentes (ESBC).

Iniciamos a construção com o planejamento de atividades e objetivos, fazendo a modelagem e escrevendo o código necessário em função dos requerimentos obtidos nas duas etapas anteriores, assim do Modelo evolucionário tomamos o refinamento constante e as experimentações contínuas que serviram para testar o código escrito.

Uma vez obtida uma primeira produção do programa passamos a aplicar o modelo de entrega incremental, onde identificadas as novas necessidades foram adicionados os incrementos (obtidos mediante a ESBC) que permitiu gerar o Protótipo 1 do ApLoB.

Figura 10 - Modelo Utilizado em nosso projeto



Fonte: Elaboração Própria

Devemos salientar que a ESBC (Engenharia de Software Baseada em Componentes) como toda metodologia em engenharia requer da utilização de processos e abordagens que correspondam a sua ciência, mas que não fazem parte do presente trabalho, mas sim a utilização e aplicação das técnicas conceptuais no mesmo. Assim, da ESBC tomamos a reutilização de componentes que foram adquiridos no processo de engenharia reversa na biblioteca Guiqwt, adicionando estes na etapa de construção.

Uma vez obtido o protótipo 1 do ApLoB este passava para uma nova etapa de análise e planejamento, que permitisse obter novos requerimentos e funcionalidades que posteriormente seriam incrementadas. Algumas novas funcionalidades foram obtidas mediante a reutilização de componentes e adicionadas ao protótipo (isto se observa na figura10, na etapa de Construção). Como o programa foi modelado os componentes encaixam muito melhor, sem ter incidência no desempenho ou

compatibilidade. Assim, cada versão do ApLob teve seus incrementos, que foram permitindo obter seus posteriores protótipos com mais funcionalidades ou melhorando elas.

A utilização e aplicação das etapas estabelecidas em nosso modelo, alinhadas aos processos de software mencionados anteriormente, geraram Protótipos que finalmente permitiram a criação de nosso projeto de software ApLoB

3.6 Aquisição de conhecimentos do entorno de desenvolvimento do ApLoB

Foi necessário o aprendizado de diferentes ferramentas para a construção do programa, citados a seguir:

- Python: a linguagem de programação base
- NumPy: é o pacote que permite trabalhar computação científica com Python (arranjos, vetores e matrizes em N dimensões, álgebra linear)
- PyQt: ferramenta para o desenvolvimento da interfaces gráficas de usuário (GUI).
- Guiqwt: Biblioteca de Python que fornece plotagem de dados em 2D (visualização de curvas e imagens e ferramentas relacionadas) para computação interativa e desenvolvimento de aplicações de processamento de sinais.
- PIL (*Python Imaging Library*): Essa livraria suporta diferentes formatos de arquivos de imagens e adiciona capacidades de processamento de imagens ao Python
- Enterprise Architect versão 9.2: Ferramenta usada fazer a Engenharia Reversa do código feito em Guiqwt.
- Spyder: é o ambiente de desenvolvimento (programação) científica em Python

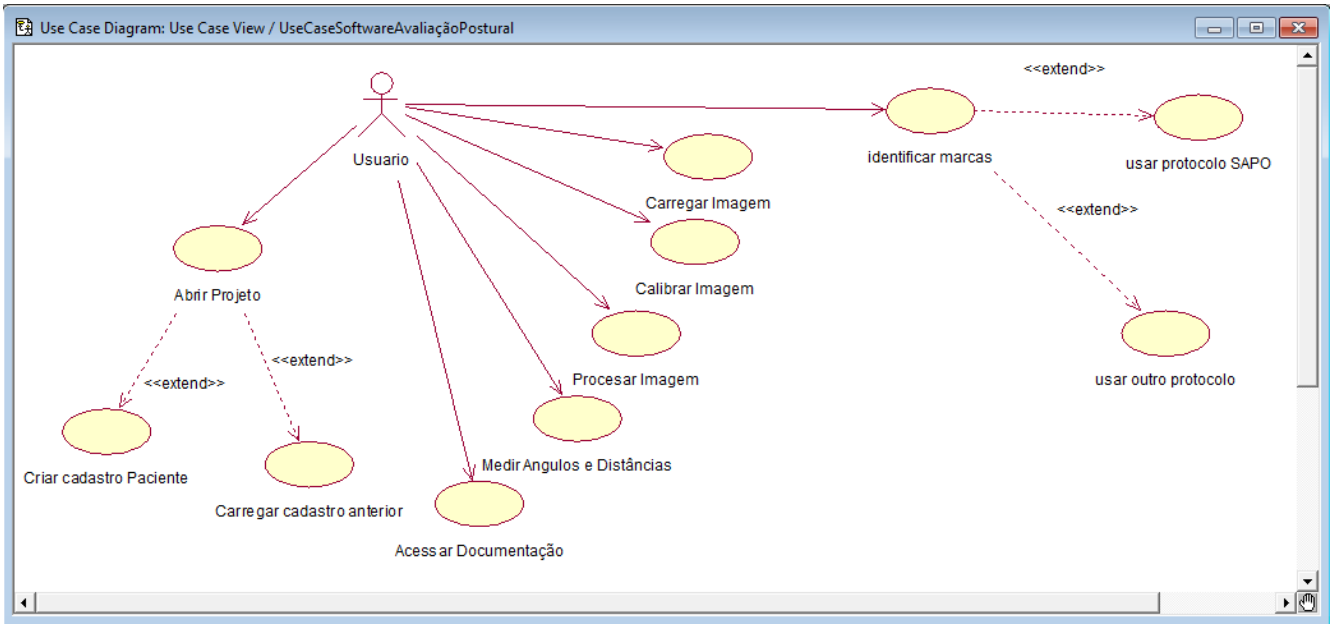
3.7 Modelagem do ApLoB

Nesta seção começa a construção específica do programa e serão expostas as principais atividades que foram realizadas. A seguir a modelagem do programa de avaliação postural.

3.7.1 Diagrama de Casos de uso

Apresentam-se os casos de uso a seguir, explicando de maneira mais detalhada os casos de maior influencia dentro do sistema.

Figura 11 – Diagrama de Casos de Uso do ApLob - Desenvolvido pelo Rational Rose



Estes casos de uso, mostrados na figura 11, representam as funcionalidades propostas para o programa de avaliação postural ApLoB desenvolvido em Python. Na continuação será detalhado cada caso de uso para um melhor entendimento dos processos.

3.7.2 Caso de Uso: Acessar Programa

Ao executar o programa o usuário carrega a aplicação junto com as bibliotecas do Python já predefinidas.

Tabela 2 – Caso de Uso: Acessar Programa

Caso de Uso – Acessar Programa	
Ator Primário	Usuário (pesquisador, fisioterapeuta)
Ator Secundário	Não há
Pré-Condições	O usuário deve ter o software instalado
Fluxo Principal	- Clicar sobre o ícone que abre a aplicação - Abre-se a aplicação - a aplicação carrega os processos definidos e as bibliotecas numpy, guiqwt, guidata
Fluxo alternativo	- Caso não encontre a aplicação a versão correspondente de software mostrará mensagem de erro correspondente
Pós Condições	Aplicação operativa
Regras de Negócio	O programa deve estar instalado no equipamento do usuário

3.7.3 Caso de uso: Carregar Imagem

O usuário tem a opção dentro do programa de carregar a imagem digital tanto ao clicar no ícone de “Open an Image” ou a partir do menu “File, Open”. O arquivo pode ter diferentes formatos de imagem já conhecidos (gif, tif, tiff, jpeg, jpg, png).

Tabela 3 – Caso de Uso: Carregar Imagem

Caso de Uso – Carregar Imagem	
Ator Primário	Usuário (pesquisador, fisioterapeuta)
Ator Secundário	Não há
Pré-Condições	- deve existir uma imagem armazenada em uma unidade conhecida
Fluxo Principal	- o usuário acessa ao sistema -seleciona a opção do menu “File, Open” -será apresentada uma janela padrão do sistema para localizar o arquivo -seleciona a unidade, a pasta, o arquivo e clica em “Open”
Fluxo alternativo	- caso o arquivo não seja do formato padrão de imagem o programa mostra uma mensagem de erro correspondente
Pós Condições	Imagem carregada e pronta para trabalhar
Regras de Negócio	Sem importar o formato da imagem, enquanto seja o padrão (tif, tiff, jpg, png) o programa conseguirá ler o arquivo.

3.7.4 Caso de Uso: Processar Imagem

O usuário pode trabalhar a imagem para melhorar a avaliação: esse processamento inclui opções como aplicações de filtros para mudar as características, rotação da imagem, obter diferentes aspetos (“zoom in”, “zoom out”).

Tabela 4 – Caso de Uso: Processar Imagem

Caso de Uso – Processar Imagem	
Ator Primário	Usuário (pesquisador, fisioterapeuta)
Ator Secundário	Não há
Pré-Condições	Imagem aceita pelo programa e carregada na tela.
Fluxo Principal	- o usuário seleciona a imagem. - Ativam-se as formatações correspondentes a formatos de imagem. - o usuário pode usar filtros de imagem para trocar a cor da imagem. - a imagem pode ser rotacionada em diferentes ângulos para obter alinhamento vertical. - pode se ativar o “zoom in” ou “zoom out” sobre a imagem.
Fluxo alternativo	- caso a imagem se encontre estragada o programa exibe mensagem de voltar a carregar a mesma. - se o usuário cancelar a operação o sistema retorna à tela inicial.
Pós Condições	- a imagem fica com zero grau de alinhamento - a imagem se encontra com as características de trabalho escolhidas pelo usuário.
Regras de Negócio	A imagem deve ser adquirida seguindo as recomendações estabelecidas no manual de usuário.

3.7.5 Caso de Uso: Calibrar Imagem

O usuário tem a opção de determinar, mediante o comando “Calibration”, as dimensões do espaço real (em centímetros) e as dimensões da imagem (em pixels).

Tabela 5 – Caso de Uso: Calibrar Imagem

Caso de Uso – Calibrar Imagem	
Ator Primário	Usuário (pesquisador, fisioterapeuta)
Ator Secundário	Não há
Pré-Condições	Imagem alinhada
Fluxo Principal	- O usuário usa a ferramenta “Segment” para obter a quantidade de pixels que mede o fio de prumo com o metro marcado nele. - com a quantidade de pixels obtidos o programa faz a transformação correspondente de pixels a centímetros, o resultado é armazenado numa variável interna. - cada segmento medido na imagem é transformado de pixels a centímetros, usando para isso a variável armazenada com o resultado da conversão.
Fluxo alternativo	Se o sistema não recebe o número inicial de pixels para fazer a conversão mostra como resultado uma solicitação do número.
Pós Condições	A variável da conversão de pixels a centímetros é armazenada na memória do computador e pronta para usos posteriores
Regras de Negócio	Tem que ser ingressado o número de pixels que mede o fio de prumo

3.7.6 Caso de Uso: Medir ângulos e distâncias

O usuário consegue obter as distâncias em centímetros de diferentes segmentos e medir os ângulos entre diferentes pontos marcados.

Tabela 6 – Caso de Uso: Medir Ângulos e Distancias

Caso de Uso – Medir Ângulos e Distancias	
Ator Primário	Usuário (pesquisador, fisioterapeuta)
Ator Secundário	Não há
Pré-Condições	- A imagem deve estar alinhada verticalmente tendo como referencia o fio de prumo. - a imagem deve estar carregada na tela principal - deve de haver sido efetuada a operação de calibração.
Fluxo Principal	- o usuário clica na ferramenta “Segment” e traça uma reta, obtendo a distância em centímetros. - o usuário clica na ferramenta “Angle” e mede o ângulo entre dois ou três pontos escolhidos. - pode se armazenar a imagem mostrando os resultados.
Fluxo alternativo	Se o usuário cancelar a operação o sistema retorna à tela inicial.
Pós Condições	Obtêm-se os resultados das grandezas geométricas (distância, ângulos).
Regras de Negócio	O programa deve carregar as funções correspondentes de medição de grandezas físico-matemáticas

3.7.7 Caso de Uso: Acessar Documentação

Mostra informação referente ao programa; requerimentos do sistema, configuração do programa, informação de utilização, sites de informação.

Tabela 7 – Caso de Uso: Acessar Documentação

Caso de Uso – Acessar Documentação	
Ator Primário	Usuário (pesquisador, fisioterapeuta)
Ator Secundário	Não há
Pré-Condições	Ter o programa instalado no computador
Fluxo Principal	O usuário acessa a documentação do sistema para obter requerimentos, configurações e informação sobre a utilização do mesmo.
Fluxo alternativo	Em caso de procurar informação e não ser encontrada o usuário pode entrar em contato via email ou via web ao site do programa.
Pós Condições	- Melhor utilização do programa. - Melhora continua das capacidades do software em função das experiências e requerimentos obtidos pelo usuário.
Regras de Negócio	O documento deve ser constantemente atualizado.

3.7.8 Caso de Uso: Identificar Marcas

O usuário pode escolher entre usar um protocolo conhecido como o do SAPO ou qualquer outro, incluso pode personalizar o seu protocolo indicando as marcas para serem avaliadas.

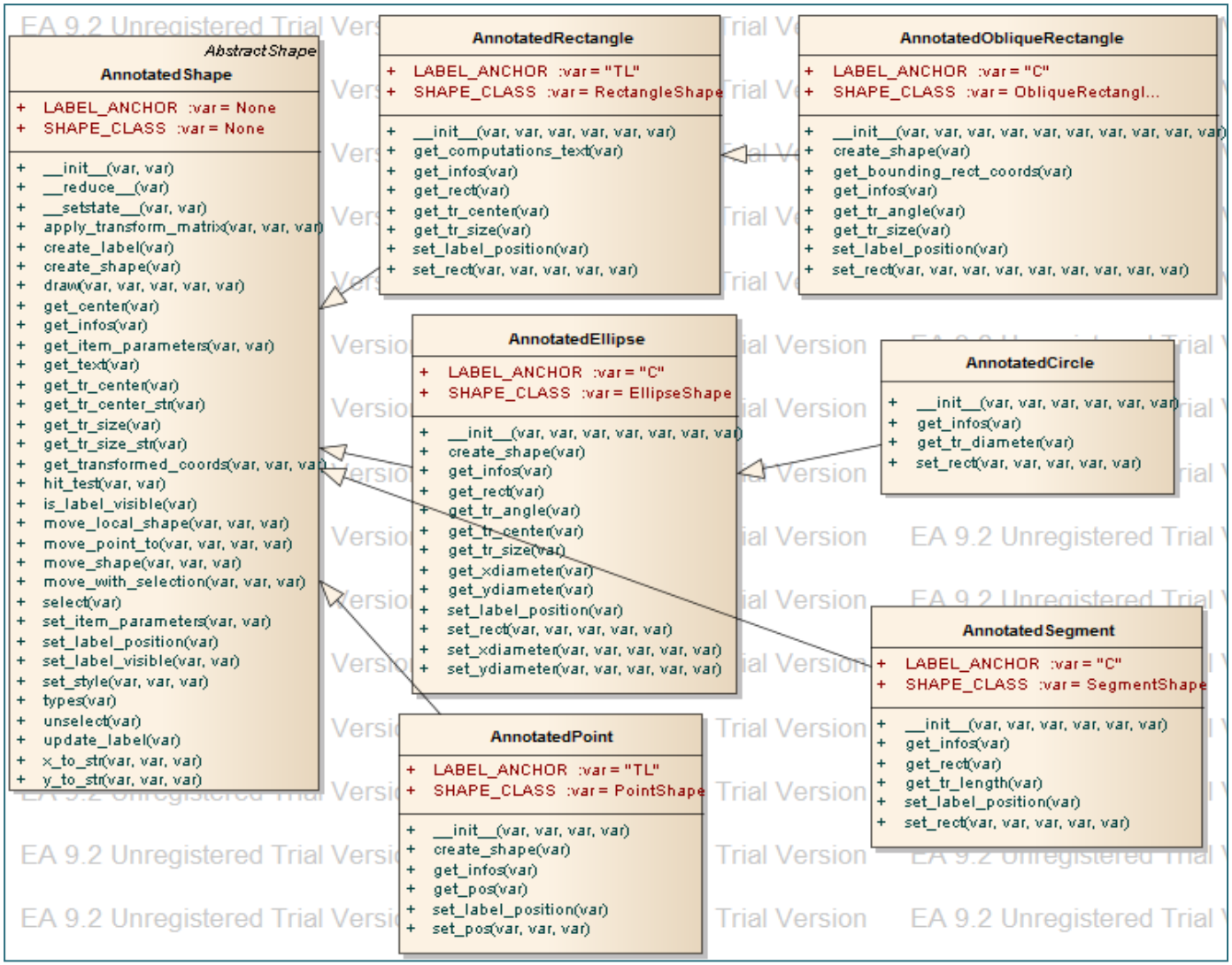
Tabela 8- Caso de uso: Identificar marcas

Caso de Uso – Identificar Marcas	
Ator Primário	Usuário (pesquisador, fisioterapeuta).
Ator Secundário	Não há.
Pré-Condições	- imagem carregada na tela principal - o alinhamento vertical da imagem com o fio de prumo já deve ter sido feito.
Fluxo Principal	- o usuário escolhe o protocolo a usar - o usuário clica na opção de “Mark” e começa a clicar nas áreas onde quer localizar os marcadores - Procede-se ao reconhecimento das marcas estabelecidas
Fluxo alternativo	- se o usuário não escolher nenhuma marca o sistema indica que deve escolher algum protocolo
Pós Condições	- O novo protocolo de marcas estabelecidas foi reconhecido
Regras de Negócio	- deve se indicar as marcas a reconhecer e isto deve ser feito com o ícone de Marcas

3.8 Modelo de Classes

Considerando-se os casos de uso, as classes necessárias para o desenvolvimento do programa ApLoB foram identificadas e, para cada classe, foram relacionadas as operações necessárias para executar os procedimentos descritos nos casos de uso.

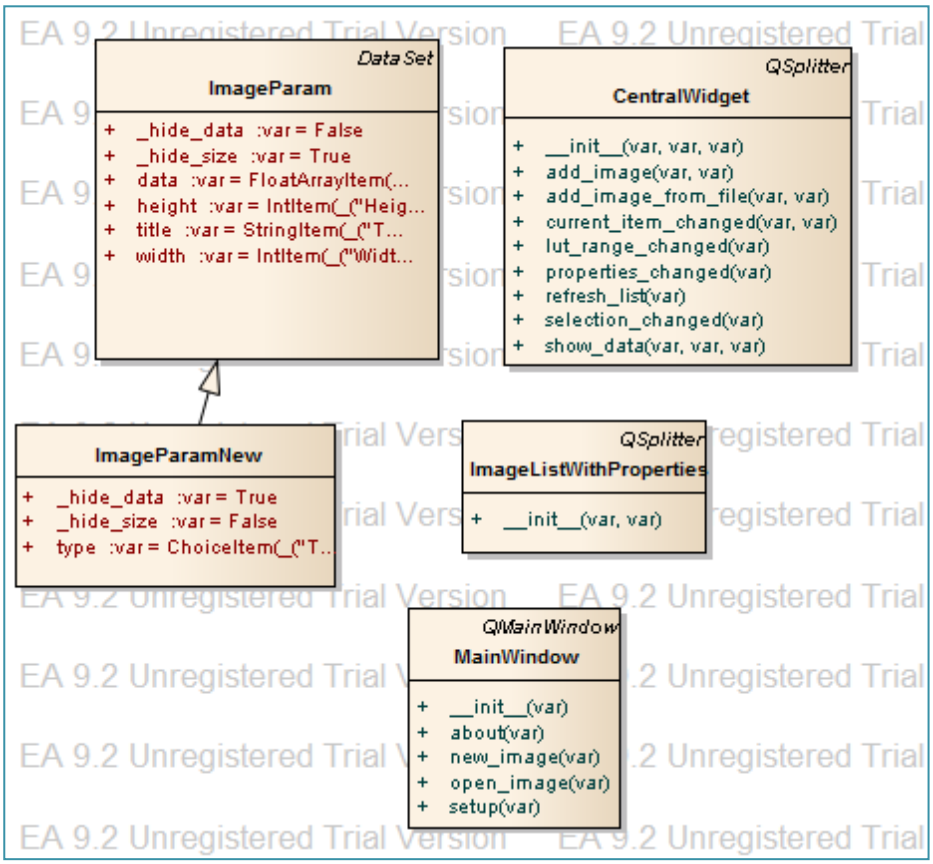
Figura 12 – Diagrama de Classes da Biblioteca Gráfica



Fonte: Elaboração Própria

O programa é composto por vários objetos de negócio nos quais cada um representa uma necessidade básica que deverá ser comportada pelo programa, controlando todas as ações que poderão ser tomadas em cada situação. O ApLoB carrega as classes das bibliotecas gráficas do Python e mostradas na Figura-12, sendo que o programa central criado conta com 5 classes mostradas na figura 13, nas quais: 2 estarão focadas diretamente no negócio, 1 delas gerencia a estrutura da aplicação e as outras 2 estão dispostas especialmente para prover integração dos dados (integra as classes das bibliotecas com as de nosso programa). Cabe ressaltar que o ApLoB carrega outras classes e que as anteriormente mostradas são as mais importantes.

Figura 13 - Diagrama de Classes do Programa



Fonte: Elaboração Própria desenvolvida no programa Enterprise Architect

A seguir encontra-se o detalhamento das classes do ApLoB:

- Classe ImageParam: é responsável por carregar a imagem, as propriedades da mesma como tamanho em pixels, filtros da imagem e ministrar os parâmetros que serão executados.

- Classe ImageParamNew: carrega uma nova imagem e herda as propriedades do ImageParam.
- Classe CentralWidget: é a classe que ministra a estrutura de trabalho, gerencia o carregamento das bibliotecas gráficas e de aplicação do Python, como o “zoom in”, “zoom out”, rotação da imagem.
- Classe MainWindow: todas as demais classes estão carregadas nesta estrutura, ou seja, ministra as funções que foram criadas na aplicação, assim como as variáveis do programa, carrega as partes correspondentes à medição de ângulos e distâncias.

Entre os principais métodos temos:

- Get_tr_angle: obtém e guarda o resultado das medições do ângulo.
- Get_transformed_coords: transforma a posição dos pixels em coordenadas dentro da imagem.
- Get_pos: obtém a posição de um ponto marcado sobre a imagem.
- Add_image_from_file: carrega na memória do computador uma imagem para depois posicionar a mesma sobre a tela principal do programa, faz uma gestão da memória da aplicação.
- Properties_changed: armazena as características feitas sobre a imagem numa estrutura matricial para depois transformar a imagem.

3.9 Desenvolvimento e Implementação

Seguindo a lógica de programação do Python e de suas bibliotecas, o projeto do ApLoB carrega vários arquivos (scripts) para seu funcionamento onde cada um faz uma série de tarefas. Assim tem-se os seguintes:

- geometry.py: onde se encontram as funções básicas geométricas (matrizes, seno, coseno, arco tangente, arco coseno, etc)
- annotations.py: esse arquivo carrega do geometry.py os cálculos obtidos e os armazena em funções mais complexas, como compute_center, compute_rect_size, compute_distance, compute_angle que posteriormente serão utilizadas.
- tools.py: possui uma coleção de ferramentas diversas para serem utilizadas (zoom, help, separador, contrastes de imagem, etc)

- plot.py: trabalha com o arquivo tools.py, o qual é usado para plotagem de algumas ferramentas
- ApLoB.py: arquivo onde se encontra o código, ele carrega dos outros arquivos as funções necessárias para sua utilização.

Nos tópicos a seguir são apresentadas as implementações efetuadas e a operacionalização do programa.

3.9.1 Carregar imagem

A ação de carregar a imagem consiste abrir a imagem e obter as características dela para a sua posterior utilização, carregar os filtros e as ferramentas que estão habilitadas para trabalhar com os formatos visuais. Também são carregadas as variáveis do sistema que controlam as mensagens de erro durante a operação. A figura 14 mostra algumas funções que tratam sobre esse tema.

Figura 14 – Parte do Código fonte para abrir a Imagem

```
243      def add_image_from_file(self, filename):
244          image = ImageParam()
245          image.title = unicode(filename)
246          image.data = imagefile_to_array(filename, to_grayscale=True)
247          image.height, image.width = image.data.shape
248          self.add_image(image)
249
250
251      def new_image(self):
252          """Create a new image"""
253          imagenew = ImageParamNew(title=_("Create a new image"))
254          if not imagenew.edit(self):
255              return
256          image = ImageParam()
257          image.title = imagenew.title
258          if imagenew.type == 'zeros':
259              image.data = np.zeros((imagenew.width, imagenew.height))
260          elif imagenew.type == 'rand':
261              image.data = np.random.randn(imagenew.width, imagenew.height)
262          self.mainwidget.add_image(image)
263
264      def open_image(self):
265          """Open image file"""
266          saved_in, saved_out, saved_err = sys.stdin, sys.stdout, sys.stderr
267          sys.stdout = None
268          filename, _filter = getopenfilename(self, _("Open"), "",
269                                              IMAGE_LOAD_FILTERS)
270          sys.stdin, sys.stdout, sys.stderr = saved_in, saved_out, saved_err
271          if filename:
272              self.mainwidget.add_image_from_file(filename)
```

Fonte:elaboração própria

3.9.2 Medição de distâncias e ângulos

Para poder obter as medições dos ângulos, a classe executada é AnnotatedEllipse, ela obtém os resultados do arquivo geometry.py e depois os carrega numa função, neste caso get_tr_angle. O trecho de código mostra a rotina de obtenção do ângulo e distância. Os resultados depois são carregados em novas funções para sua posterior utilização.

Figura 15 – Parte do código fonte que mede ângulos e distâncias

annotations.py* | geometry.py | plot.py | tools.py

```
521
522
523 class AnnotatedEllipse(AnnotatedShape):
524
525     SHAPE_CLASS = EllipseShape
526     LABEL_ANCHOR = "C"
527     def __init__(self, x1=0, y1=0, x2=0, y2=0, ratio=1., annotationparam=None):
528         self.ratio = ratio
529         AnnotatedShape.__init__(self, annotationparam)
530         self.set_xdiameter(x1, y1, x2, y2)
531     #---Public API-----------------------------------------------------------
532     def set_xdiameter(self, x0, y0, x1, y1):
533         self.shape.set_xdiameter(x0, y0, x1, y1)
534         self.set_label_position()
535
536     def get_xdiameter(self):
537         return self.shape.get_xdiameter()
538
539     def set_ydiameter(self, x2, y2, x3, y3):
540         self.shape.set_ydiameter(x2, y2, x3, y3)
541         self.set_label_position()
542
543     def get_ydiameter(self):
544         return self.shape.get_ydiameter()
545
546     def get_tr_angle(self):
547         xcoords = self.get_transformed_coords(0, 1)
548         _x, yr1 = self.apply_transform_matrix(1., 1.)
549         _x, yr2 = self.apply_transform_matrix(1., 2.)
550         return (compute_angle(reverse=yr1 > yr2, *xcoords)+90)%180-90
```

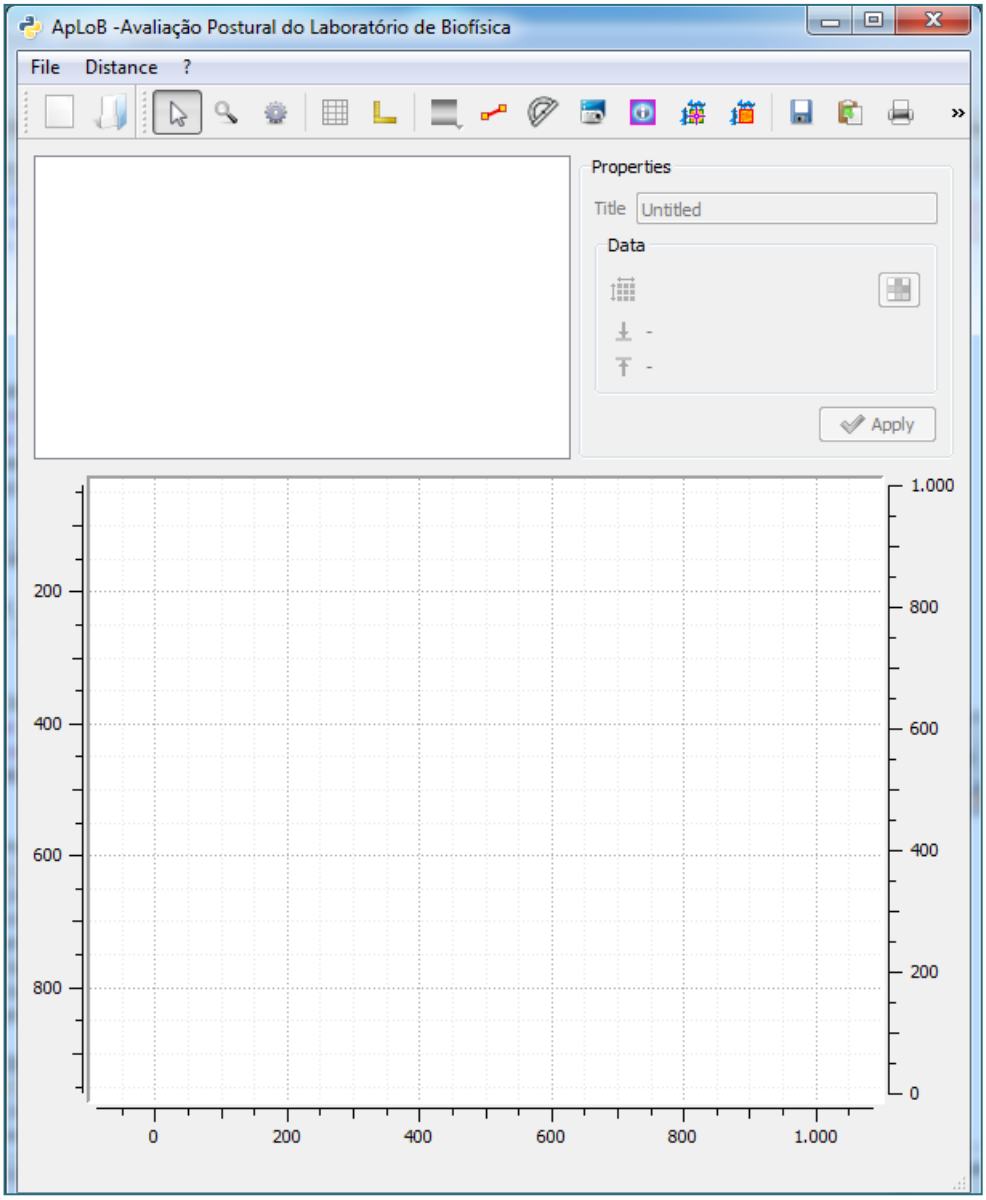
Fonte: elaboração própria

3.10 Telas do Protótipo ApLoB

3.10.1 Tela principal

Mostra a estrutura principal do programa, a tela onde se carregam as funções de cálculo e ferramentas necessárias para o trabalho. Uma barra de menu com as opções “File” e “Distance”, sendo o primeiro para trabalhar com imagem (abrir, guardar, rotar) e o segundo para fazer os cálculos de distancia. Uma barra de ferramentas com os ícones e suas respectivas funcionalidades (ângulo, distancia, configurar, zoom, filtros de imagem, imprimir, guardar), uma área que mostra as características da imagem e outra maior onde se carregará a imagem.

Figura 16 – Tela principal com a imagem carregada

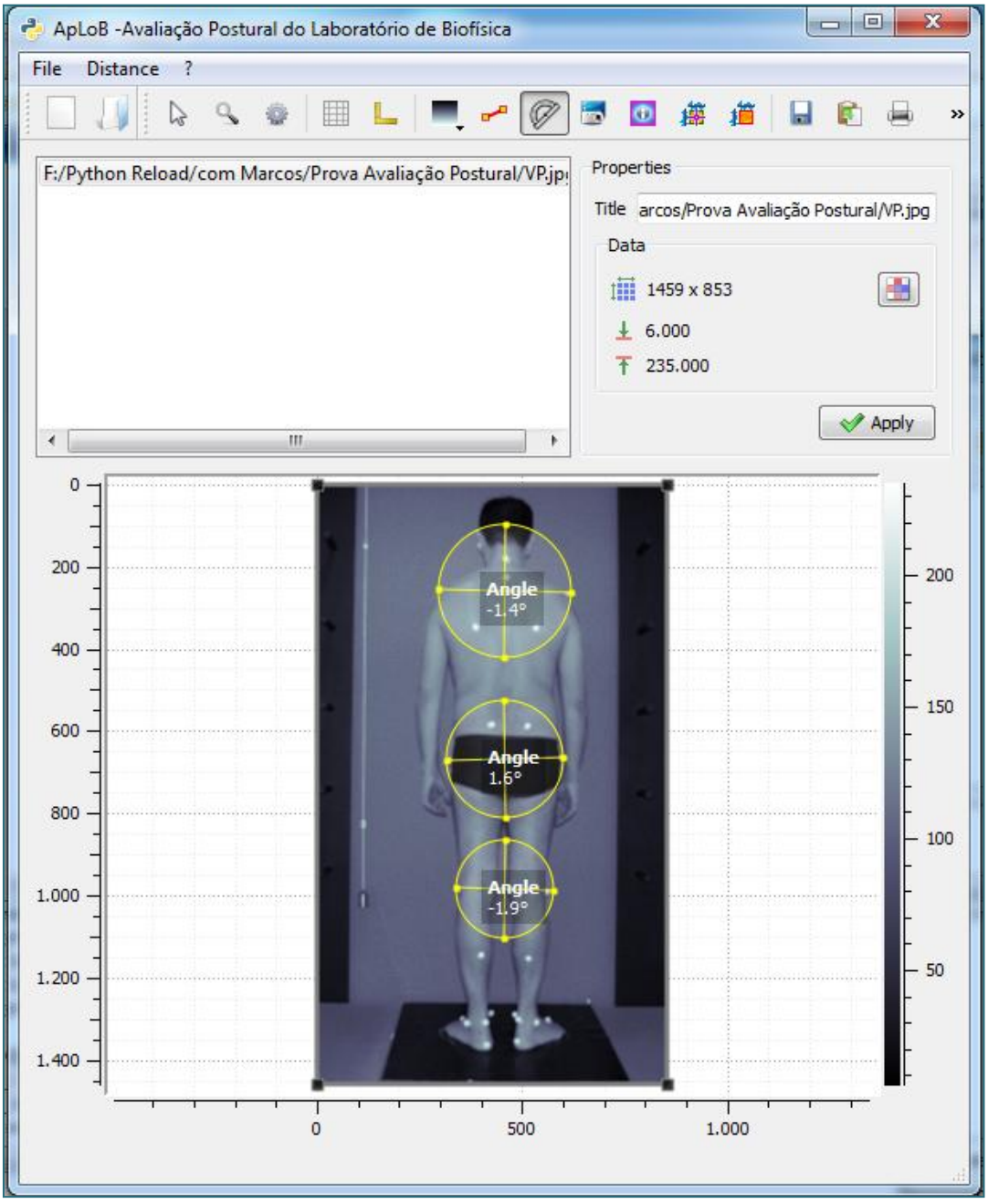


Fonte: elaboração própria

3.10.2 Tela de Imagem com medição de ângulos

Na figura 17 mostra-se uma imagem da vista posterior já carregada na área correspondente. O programa carrega e ativa as ferramentas correspondentes, neste caso a ferramenta “Angle”, representada pela figura do transferidor e é utilizada para fazer a medição de ângulos. O usuário clica sobre uma marca e sem soltar une com outra marca; a partir desta operação aparece um indicador da quantidade de graus do ângulo formado.

Figura 17 – Imagem Vista Posterior com medição de ângulos

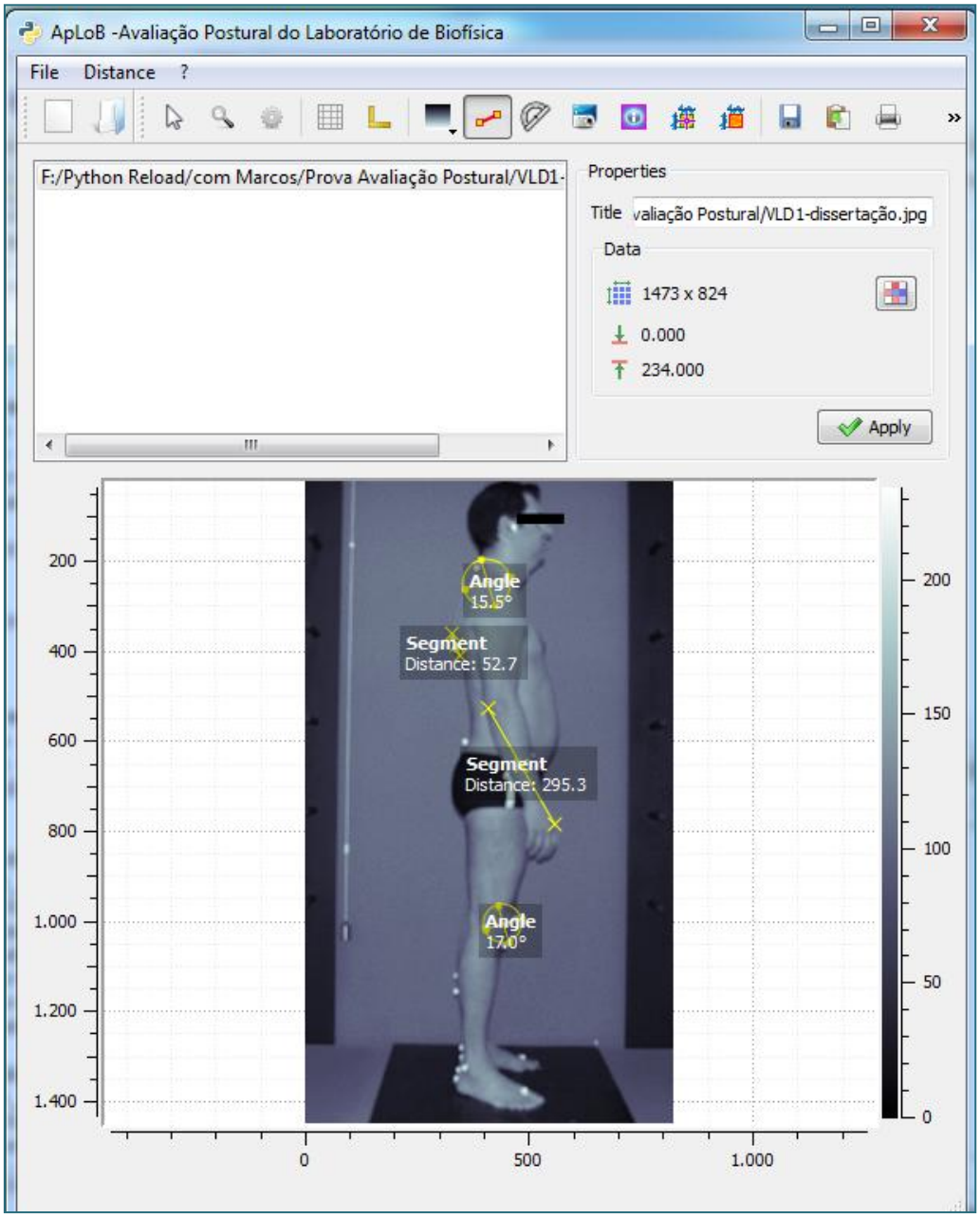


Fonte: elaboração própria (interface ApLoB)

3.10.3 Distancias e ângulos

Na figura 18, vista lateral direita, mostra-se a imagem carregada com aplicações diretas tanto da ferramenta de distancia de segmento (Segment) como a de medição de ângulo (Angle). Seguindo a metodologia anterior, a ferramenta de medição de segmento é “drag and drop”, arrastar e soltar. Para obter uma distancia correta em centímetros previamente deve-se fazer a conversão de pixels a centímetros.

Figura 18 – Imagem vista lateral direita com ângulos e distancias

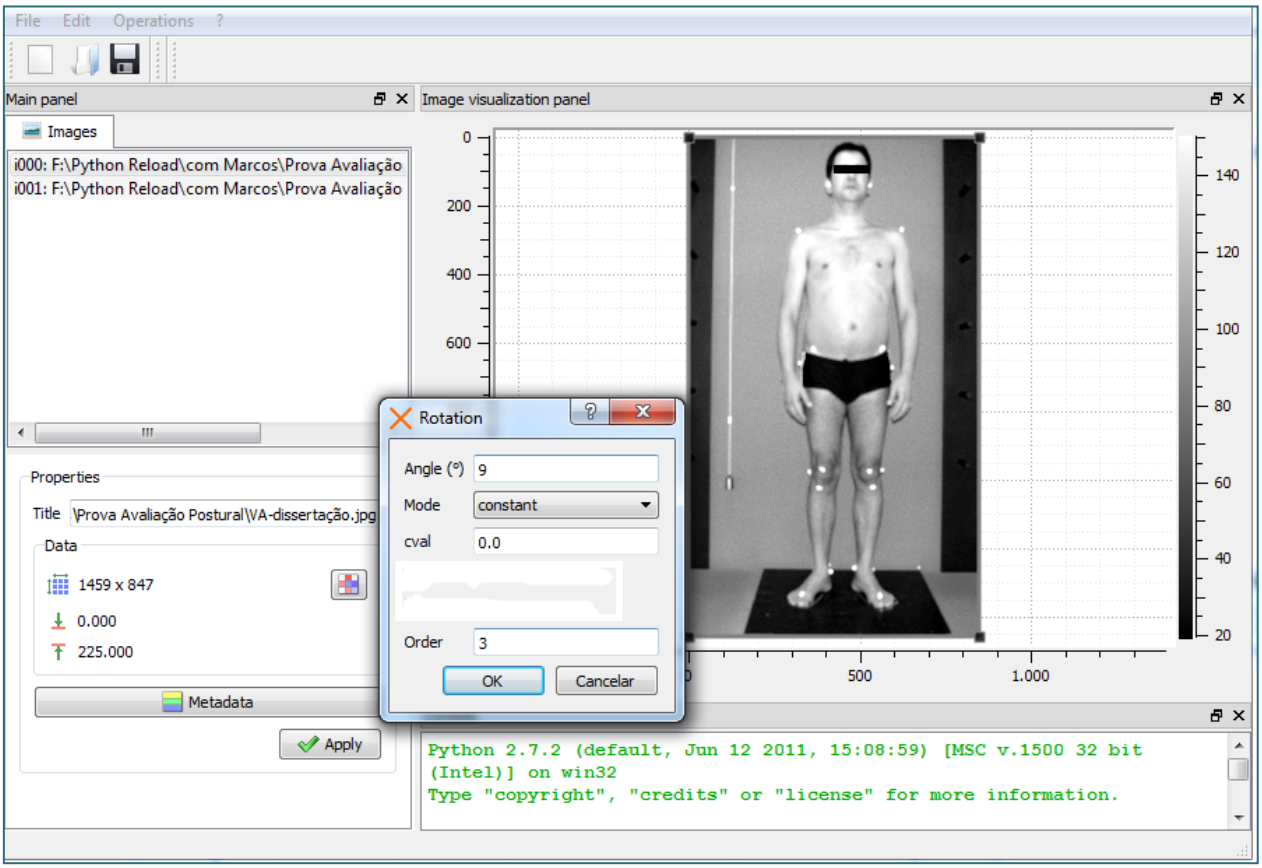


Fonte: elaboração própria (interface ApLoB)

3.10.4 Tela com a opção de Rotar Imagem

A figura 19 mostra a opção de rotar a imagem na qual o fio de prumo serve de referencia vertical. A ferramenta “Angle” serve para saber a diferença da inclinação da imagem capturada com respeito à vertical indicada pelo fio de prumo. Conhecendo esse valor procede-se a rotação da imagem com os graus desejados.

Figura 19 – Vista anterior com a opção de rotar imagem



Fonte: elaboração própria (interface ApLoB)

4 Resultados e discussões do ApLob

4.1 Testes

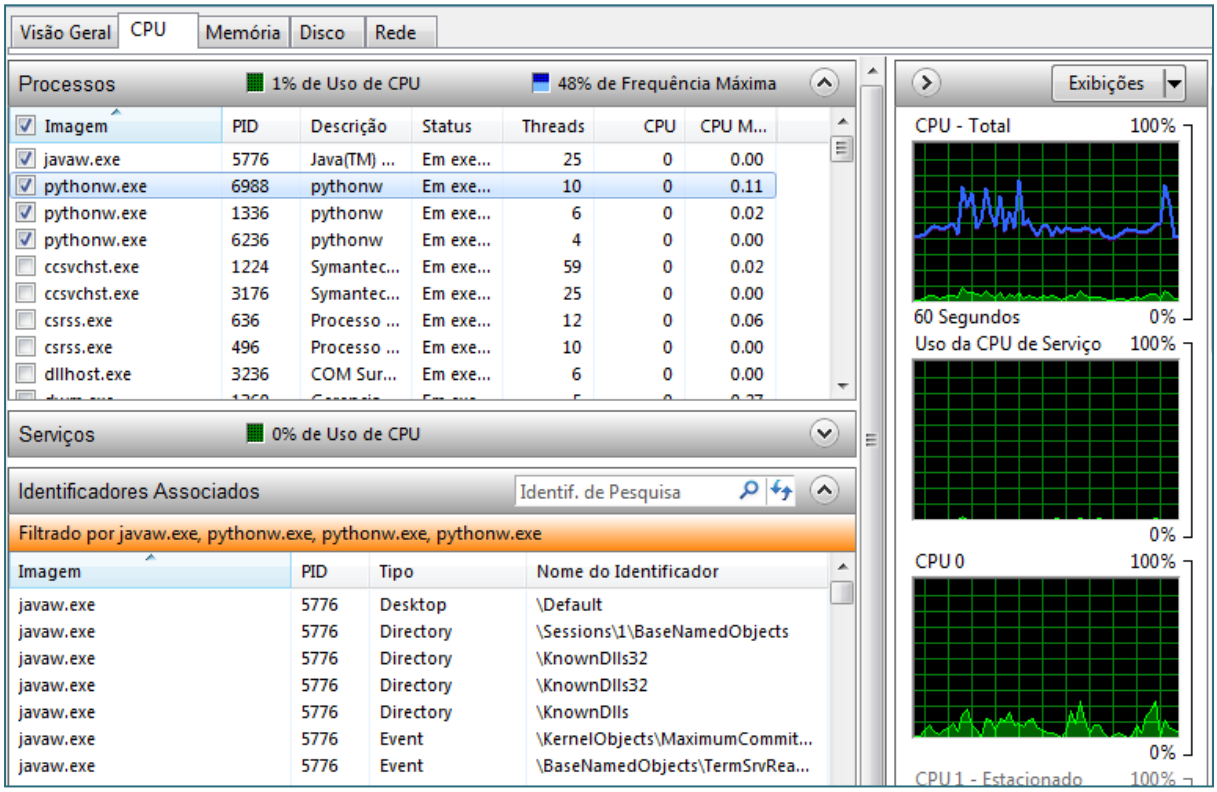
O ApLoB foi testado e validado nas dependências do Laboratório de Biofísica da Escola de Educação Física e Esporte da Universidade de São Paulo. Realizaram-se os seguintes testes:

- Medição de objetos inanimados com dimensões conhecidas: Nesta etapa, foi testada a acurácia e precisão do programa. Procedeu-se à captura de imagens de diferentes objetos e com a ajuda de uma régua de um metro (que serviu para calibrar o tamanho proporcional) realizou-se a conversão correspondente de pixels a centímetros, obtendo-se o tamanho correto dos objetos medidos. Em total foram medidos 15 objetos de diferente tamanho e volumem cada um. Fez-se a medição com a ajuda de uma régua ou fita métrica, dependendo da facilidade que cada um apresentava em função de seu volumem. Os resultados obtidos das medições digitais foram iguais aos feitos manualmente.
- Medição da postura de seres humanos: para testar a aplicabilidade do programa em seres humanos realizou-se as medições correspondentes nas mesmas imagens que foram avaliadas através do programa SAPO e foram comparados os resultados do ApLoB com os relatórios gerados pelo SAPO. Foram utilizadas as imagens de 20 pessoas (foto frontal, lateral direito, lateral esquerdo e posterior, resultando em 80 fotos em total) cujos relatórios já tinham sido gerados pelo SAPO, assim os resultados obtidos poderiam ser comparados com os gerados anteriormente. As medições de distâncias feitas pelo ApLob tiveram variação entre 0,2 cm até 1,0 centímetro, sendo consideradas como aceitáveis. Já as medições dos ângulos tiveram variação de 0 graus até 5 graus.
- Deve-se indicar que existe uma margem de erro na medição tanto das distâncias assim como os ângulos que é considerada aceitável, isso não se deve ao momento de fazer a transformação de pixels a centímetros ou erro no cálculo, mas sim ao lugar onde o usuário começa a clicar para fazer a medição: escolher o centro da marca pode ser 1 ou 2 pixels

diferente da escolha de outro usuário por razões de percepção, podendo gerar um resultado de décimos de grau de diferença.

- Medição de recursos: Referente à utilização de recursos de hardware de cada aplicativo. Foi realizado um teste (figura 20) onde cada aplicativo fez medições de ângulos com a mesma imagem carregada. Observa-se que o SAPO (aplicativo em Java) apesar de ter maior quantidade de funções carregadas em paralelo (25 versus 20 do ApLoB) tem um melhor tempo de resposta por parte do microprocessador em décimos de segundo, mas por se tratar de aplicações gráficas ambos demandam a mesma quantidade de memória. Em resumo ambas as aplicações tem o mesmo desempenho na utilização de hardware.

Figura 20 – Medição de Recursos entre o SAPO e o ApLoB



Fonte: elaboração própria

4.2 Comparações entre os softwares SAPO e ApLoB

Tendo como parâmetro de desenvolvimento o software SAPO elaborou-se uma tabela comparativa (tabela 9) das funcionalidades do mesmo e das funcionalidades alcançadas pelo ApLoB. Cabe ressaltar que as funcionalidades apontadas nesta tabela são as mesmas indicadas no item 2.5 deste trabalho referente às funcionalidades que um software de avaliação postural com relevância comparável ao SAPO deve apresentar.

Tabela 9 – Comparação de funcionalidades entre o SAPO e o ApLoB

SAPO	ApLoB
1. Carregar imagens.	✓
2. Processar imagem (rotar, filtros de imagem).	✓
3. Calibração da imagem.	✓
4. Medição livre de posição, distâncias, ângulos e área na imagem.	✓
5. Digitalização de pontos na imagem	✓
6. Cálculo de grandezas físico-matemáticas a partir dos pontos reconstruídos	Este ponto é a continuação do processo anterior (5). O programa não consegue medir ângulos entre 3 pontos.
7. Visualização dos pontos reconstruídos e das grandezas calculadas	O ApLoB visualiza os pontos, mas como indicado no ponto anterior (6) não calcula 3 pontos marcados.
8. Geração de relatórios.	Abre uma planilha predefinida com parâmetros de avaliação postural em Excel para o usuário preencher manualmente os resultados.
9. Documentação	✓

A seguir encontra-se o detalhamento das funcionalidades descritas na tabela anterior:

- Referente aos pontos 1e 2 de Carregar e Processar Imagem, o ApLoB consegue carregar a imagem (uma ou várias) e com elas ativar os filtros de imagem correspondentes. O processo de rotar é conseguido manualmente depois de obter os graus necessários e não automaticamente como o SAPO.
- O ponto 3 correspondente a Calibração da imagem é obtido manualmente mediante a utilização dos menus.
- O pontos 4 e 5, Medição e Digitalização são obtidos com a mesma acurácia do software SAPO, contudo sem serem igualmente automatizados.

- Os pontos 6 e 7 não são obtidos na sua totalidade por que ainda não se consegue medir ângulos entre três pontos marcados.
- A geração de relatórios correspondente ao ponto 8 é obtida mediante o acesso a uma planilha de Excel. O ApLoB não gera uma planilha automática de resultados mas da acesso a uma planilha pré-estabelecida.
- O ponto 9, Documentação do programa (em anexo) foi realizado com sucesso.

4.3 Potencialidades do ApLoB

Podemos indicar como potencialidades do ApLoB:

- Sua acurácia comprovada mediante os testes comparativos com o SAPO.
- A utilização, no seu desenvolvimento, de um programa computacional científico de fácil entendimento e constante evolução como é o Python assim como a disponibilização das fontes para trabalhos futuros.
- Possuir modelagem de software que identifica os Casos de Uso, Classes e funcionalidades que norteariam a continuidade do projeto.

Cabe ressaltar que no mercado existem softwares de avaliação postural que conseguem realizar as tarefas de avaliação com a mesma acurácia do ApLoB mas com menos funcionalidades que o mesmo. Neste âmbito o ApLob apesar de não conseguir realizar todas as funcionalidades do SAPO apresenta mais funcionalidades que boa parte do softwares de avaliação postural disponíveis no mercado e possibilita um melhoramento contínuo do mesmo, baseado na documentação gerada pelo presente trabalho.

Acrescenta-se às características acima o fato do ApLob ser fácil de usar e consumir poucos recursos de hardware, fatores que acessibilizam enquanto a tecnologia o programa para usuários da fisioterapia, educação física, além da comunidade científica.

5 Considerações Finais

A proposta do presente trabalho foi desenvolver o Software ApLoB que consiste em um software livre capaz de cumprir com as funcionalidades próprias de um programa para avaliação postural, tendo como parâmetro de funcionalidade e desenvolvimento o software SAPO.

5.1 Conclusões

Para o desenvolvimento do software ApLoB analisaram-se as características que um programa de avaliação postural deve apresentar e, a partir disso, elaborou-se um projeto no qual foram utilizadas as metodologias e processos estabelecidos na engenharia de software.

Para efeito de validação das ferramentas criadas no programa foram realizadas provas comparando os resultados obtidos pelo ApLoB com os relatórios gerados pelo SAPO. Com relação aos resultados dos testes e às comparações deste programa estabelecidas em relação ao SAPO verificou-se que o protótipo atende as funcionalidades dos softwares de avaliação postural existentes no mercado e a maior parte das funcionalidades estabelecidas pelo SAPO.

Mais do que isso, ressalta-se a característica do ApLoB apresentar interface gráfica amigável graças ao seu desenvolvimento gráfico em PyQt. Esse atributo aliado à facilidade de uso do programa, ao baixo consumo de recursos de Hardware e à caracterização do mesmo como um software livre são fatores que abrem possibilidades de ampla utilização do programa por profissionais das áreas de fisioterapia, educação física bem como da comunidade científica.

Este trabalho ainda poderá ser explorado por outros pesquisadores tornando-se base para novas implementações ou extensões de programas de avaliação postural, uma vez que toda a documentação, modelagem e código fonte desenvolvidos estarão disponíveis para este fim.

5.2 Sugestões de trabalhos futuros

Utilizando o protótipo do ApLoB desenvolvido neste trabalho é possível identificar novas funcionalidades num programa de avaliação postural, como:

- ✓ Implementação de uma opção no menu que carregue um simetrógrafo que permitirá ter uma ferramenta a mais para a avaliação postural
- ✓ Desenvolver um banco de dados especializado, que permita um seguimento cronológico dos pacientes e gere relatórios comparativos.

Referencias Bibliográficas

Adams, R. C. (2001). Jogos, Esportes e Exercícios para o Deficiente Físico. são paulo.

American Society of Photogrametry and Remote Sensing, A. (1980). Manual of photogrammetry. Virginia.

Batista LH, C. P., Aiello GV, Oishi J, Salvini TF. (2006). Avaliação da amplitude articular do joelho:correlação entre as medidas realizadas com o goniômetro universal e no dinamômetro isocinético. Revista Fisioterapia - Universidade de São Paulo. São Paulo, Universidade de São Paulo. **10**.

Benson, P. E. and S. Richmond (1997). "A critical appraisal of measurement of the soft tissue outline using photographs and video." Eur J Orthod **19**(4): 397-409.

Berzins, V. (1991). Software Engineering with Abstractions. California, Adison Wesley Publishing Company.

Booch, G., J. Rumbaugh, et al. (1998). The unified modeling language user guide, Addison-Welsley

Braz, R. G., F.; Carvalho, G. (2008). "Confiabilidade e validade de medidas angulares por meio do software para avaliação postural (SAPO)." Fisioterapia e Movimento **21**(3): 117-126.

Cai, X., H. P. Langtangen, et al. (2005). "On the performance of the Python programming language for serial and parallel scientific computations." Scientific Programming **13**(1): 31-56.

Chen, J., B. Wang, et al. (2009). Requirements analysis of real-time systems by rational-rose UML, Wuhan.

Comerlato, T. (2007). Avaliação da postura corporal estática no plano frontal a partir de imagem digital. Ciências do Movimento Humano. Porto Alegre**:** 73.

Danis, C. G., D. E. Krebs, et al. (1998). "Relationship between standing posture and stability." Physical Therapy **78**(5): 502-517.

Day, J. W., G. L. Smidt, et al. (1984). "Effect of pelvic tilt on standing posture." Physical Therapy **64**(4): 510-516.

Dunk, N. M., Y. Y. Chung, et al. (2004). "The reliability of quantifying upright standing postures as a baseline diagnostic clinical tool." Journal of Manipulative and Physiological Therapeutics **27**(2): 91-96.

Ertl, M. A. (2011) How popular are various programming languages?

Farias, N., I. Rech, et al. (2009). "Avaliação postural em hemiparéticos por meio do software SAPO - Relato de caso." ConScientiae Saúde **8**(4): 649-654.

Fedorak, C., N. Ashworth, et al. (2003). "Reliability of the visual assessment of cervical and lumbar lordosis: how good are we?" Spine (Phila Pa 1976) **28**(16): 1857-1859.

Ferreira, E. A., M. Duarte, et al. (2010). "Postural assessment software (PAS/SAPO): Validation and reliabiliy." Clinics (Sao Paulo) **65**(7): 675-681.

Ferreira, E. A. G. (2006). Postura e controle postural: desenvolvimento e aplicação de método quantitativo de avaliação postural. São Paulo, Universidade de São Paulo**:** 77.

Fisher, A. S. (1991). CASE: Using Software Development Tools. Toronto.

Furlaneto, T., C. Candotti, et al. (2007). DESENVOLVIMENTO DE UMA METODOLOGIA DIGITAL PARA AVALIAÇÃO POSTURAL NO PLANO SAGITAL. Biomecânica 2007. Universidade do Vale do Rio dos Sinos – São Leopoldo – RS, Universidade do Vale do Rio dos Sinos – São Leopoldo – RS**:** 6.

Gane, C. (1990). CASE O relatório Gane. Rio de Janiro.

Gangnet, N., V. Pomero, et al. (2003). "Variability of the spine and pelvis location with respect to the gravity line: a three-dimensional stereoradiographic study using a force platform." Surg Radiol Anat **25**(5-6): 424-433.

Garcia-Magarino, I., R. Fuentes-Fernandez, et al. (2010). "A framework for the definition of metamodels for Computer-Aided Software Engineering tools." Information and Software Technology **52**(4): 422-435.

Gardocki, R. J., R. G. Watkins, et al. (2002). "Measurements of lumbopelvic lordosis using the pelvic radius technique as it correlates with sagittal spinal balance and sacral translation." Spine J **2**(6): 421-429.

Gennari, J. H. and M. Reddy (2000). "Participatory design and an eligibility screening tool." Proc AMIA Symp: 290-294.

Giglio, C. A. and J. B. Volpon (2007). "Development and evaluation of thoracic kyphosis and lumbar lordosis during growth." J Child Orthop **1**(3): 187-193.

Herscovici, D., Jr. and R. W. Sanders (2000). "The effects, risks, and guidelines for radiation use in orthopaedic surgery." Clin Orthop Relat Res(375): 126-132.

Holsing, N. F. and D. C. Yen (1997). "Integrating computer-aided software engineering and Object-Oriented systems: A preliminary analysis." International Journal of Information Management **17**(2): 95-113.

IBM developerWorks Resource Center (2004). "IBM developerWorks Resource Center." Retrieved 26/04/11, 2011, from http://www.developers.net/ibmshowcase/product/Rational_Rose.

Institute of Electrical and Electronics Engineers, I. (2011). Retrieved 05/05/2011, 2011, from http://www.iso-architecture.org/ieee-1471/.

International Organization for Standardization, I. (2001). "International Organization for Standardization." Retrieved 05/05/2011, 2011, from http://www.iso.org/iso/home.html.

Iunes, D. H., M. B. Cecilio, et al. (2010). "Quantitative photogrammetric analysis of the Klapp method for treating idiopathic scoliosis." Rev Bras Fisioter **14**(2): 133-140.

Kendall, M. E. G., ProvancePatricia ; Peterson, Kendall Florence ; McIntyre, RodgersMary ; Romani, William Anthony; (2007). MMúsculos: Provas e Funções, manole.

Labaki, J. (2008). Grupo Python. São Paulo, UNESP - Campus de ilha solteira.

Li, J. and L. Li (2010). "Comparative Research on Python Speed Optimization Strategies." IEEE: 57,58,59.

Macoratti, J. C. (2005). Retrieved 10/05/2010, 2010, from http://www.macoratti.net/proc_sw1.htm.

Mercadante, F. O., L.A.; Duarte, M. (2005). "Avaliação postural quantitativa através de imagens bidimensionais." Anais do XI Congresso Brasileiro de Biomecânica em meio digital: Página final.

Moran, T. P. and J. M. Carroll (1996). Design rationale : concepts, techniques, and use. Mahwah, N.J., L. Erlbaum Associates.

Nault, M. L., P. Allard, et al. (2002). "Relations between standing stability and body posture parameters in adolescent idiopathic scoliosis." Spine **27**(17): 1911-1917.

Nery, P. B. (2009). Análise da confiabilidade intra e interexaminador do software de avaliação postural - SAPO em escolares do município de Riberão Preto-SP. Escola de Enfermagem de Riberão Preto-SP. São Paulo, Universidade de São Paulo**:** 108.

Pereira, B. and C. Medalha (2008). "Avaliação postural por fotometria em pacientes hemiplégicos." ConScientiae Saúde **7**(1): 35-42.

Pressman, R. (2005). Software Engineering: A Practitioner´s Approach. São Paulo, Makron Books.

Price, A. M. d. A. and S. Toscani, Eds. (2001). Implementação de Linguagens de Programação:Compiladores. Porto Alegre - RS.

Python (2011). "Quotes about Python." Retrieved abril 2011, 2011, from http://www.python.org/about/quotes/.

Qt (2011). Retrieved maio 2011, 2011, from http://qt.nokia.com/.

Quatrani, T., Ed. (1999). Visual Modeling with Rational Rose and UML, Pearson Education Corporation Sales Division.

Riverbank (2011). Retrieved janeiro 2011, 2011, from http://www.riverbankcomputing.co.uk/news.

Sacco ICN, A. S., Queiroz BWC, Pripas D, Kieling I, Kimura AA, Sellmer AE, Malvestio RA, Sera MT (2007). "Confiabilidade da fotogrametria em relação a goniometria para avaliação postural de membros inferiores." Revista Brasileira de Fisioterapia **11**(n.5): 411-417.

Sanders, I. D. and S. Langford (2008). Students' perceptions of python as a first programming language at wits. ITiCSE - The 13th Annual Conference on Innovation and Technology in Computer Science Education, Madrid.

Sapo-Desktop (2012).

SAPO (2005). from http://puig.pro.br/sapo/.

SAPO, S. d. A. P.-. (2011). Retrieved novembro, 2011, from http://demotu.org/sapo/.

Silva, J. and S. G. Assis (1998). Linguagens de Programação: Conceitos e Avaliações. São Paulo, McGraw-Hill.

Sommerville, I. (2007). Engenharia de Software. São Paulo.

Taylor, R. M., N.; Dashofy, E. (2010). Software Architecture: Foundations, Theory, and Practice.

Tommaselli, A. M. G. S., J. F. C da; Hasegawa, J. K.; Galo, M.; DalPoz, A. P (1999). "Fotogrametria: Aplicações a curta distância." FCT 40 anos. Perfil Científico Educacional. Unesp: Presidente Prudente SP: 147-159.

Vegter, F. and J. J. Hage (2000). "Standardized facial photography of cleft patients: just fit the grid?" Cleft Palate Craniofac J **37**(5): 435-440.

Watson, A. W. and C. Mac Donncha (2000). "A reliable technique for the assessment of posture: assessment criteria for aspects of posture." J Sports Med Phys Fitness **40**(3): 260-270.

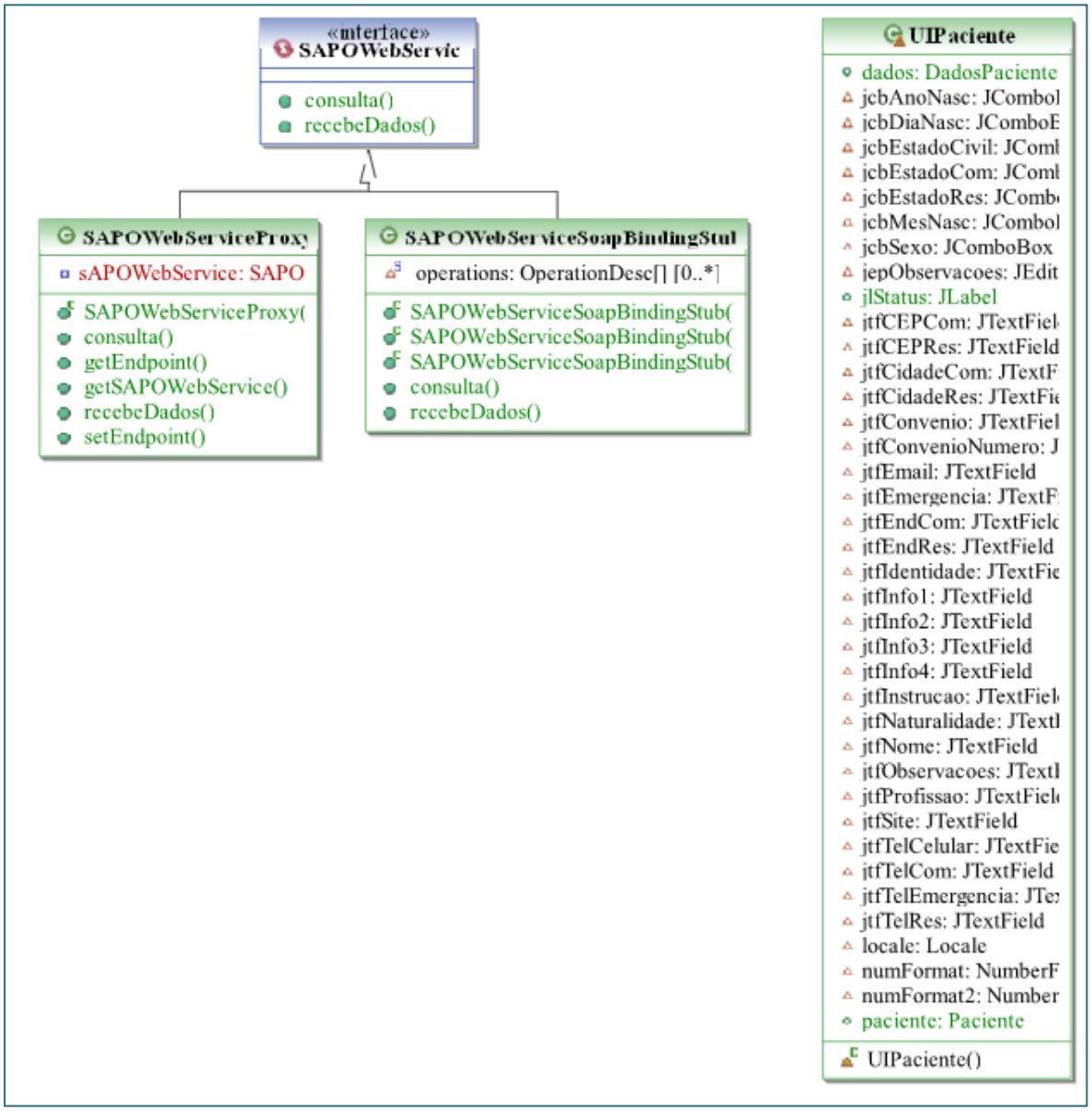
Wilson de Pádua Filho (2003). Engenharia de Software: Fundamentos, Métodos e Padrões. Rio de Janeiro, Livros Técnicos e Científicos Editora S.A.

Z.M.Ma, F. Zhang, et al. (2011). "Representing and reasoning on fuzzy UML models: A description logic approach." Elsevier Expert Systems with Applications **38**: 2536–2549.

Zonnenberg, A. J. J., C. J. VanMaanen, et al. (1996). "Body posture photographs as a diagnostic aid for musculoskeletal disorders related to temporomandibular disorders (TMD)." Cranio-the Journal of Craniomandibular Practice **14**(3): 225-232.

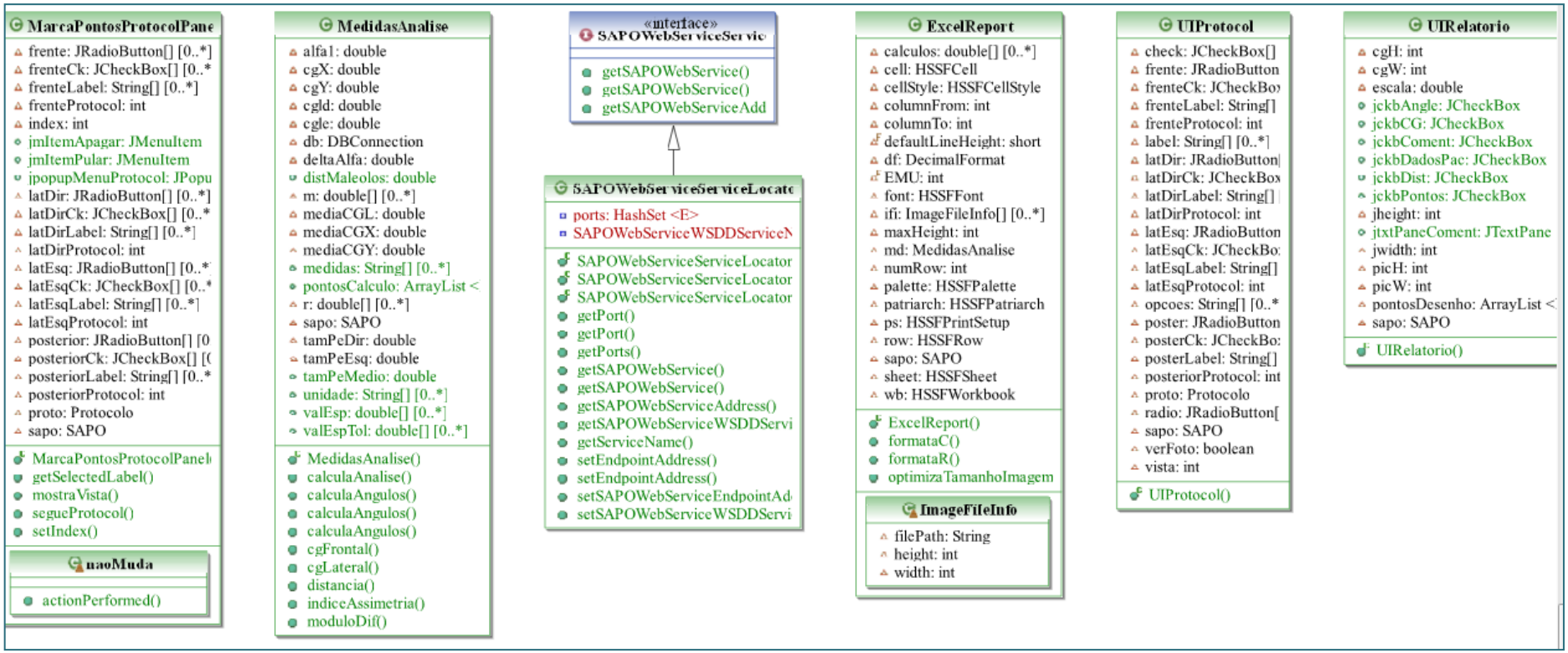
Anexos

Anexo A- Diagramas de Hierarquias do SAPO

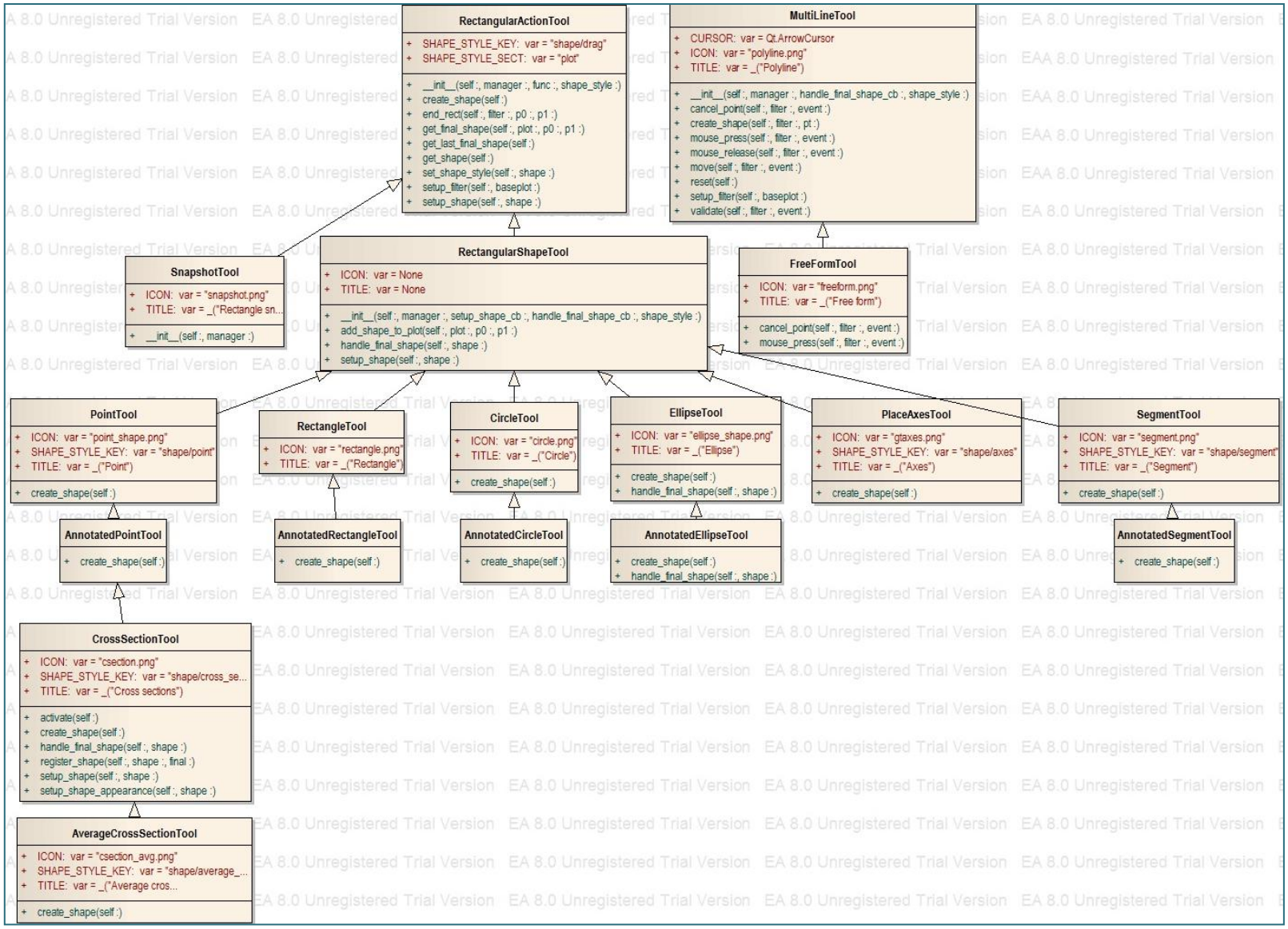


Anexo B- Diagramas de Hierarquias do SAPO

Anexo C- Diagramas de Hierarquias do SAPO



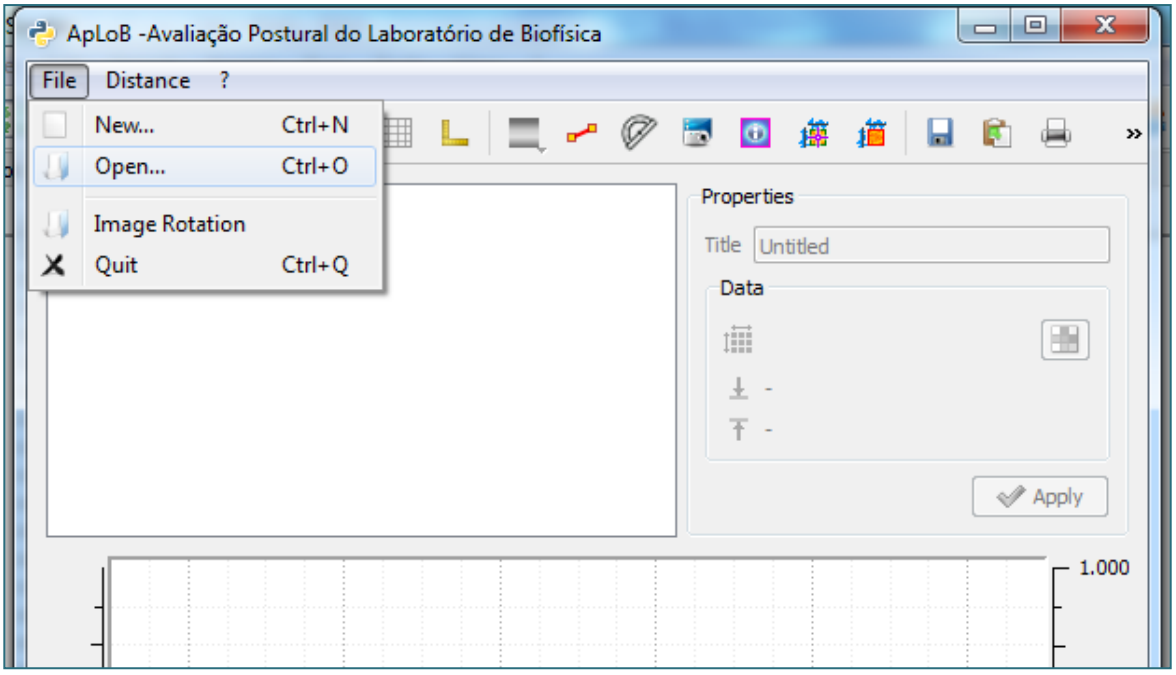
Anexo D- Diagramas de Classes do arquivo Tools.py do ApLoB



Anexo E – Manual de Uso

Abrir Arquivo: Pode abrir uma imagem (png, jpg, jpeg, gif) seguindo o menu File, Open (fig. 21). Aparece uma janela do explorador de Windows que permite procurar em unidades de rede, dispositivos de armazenamento ou em outro local a imagem a carregar. Depois de escolher a imagem se procede a Abrir.

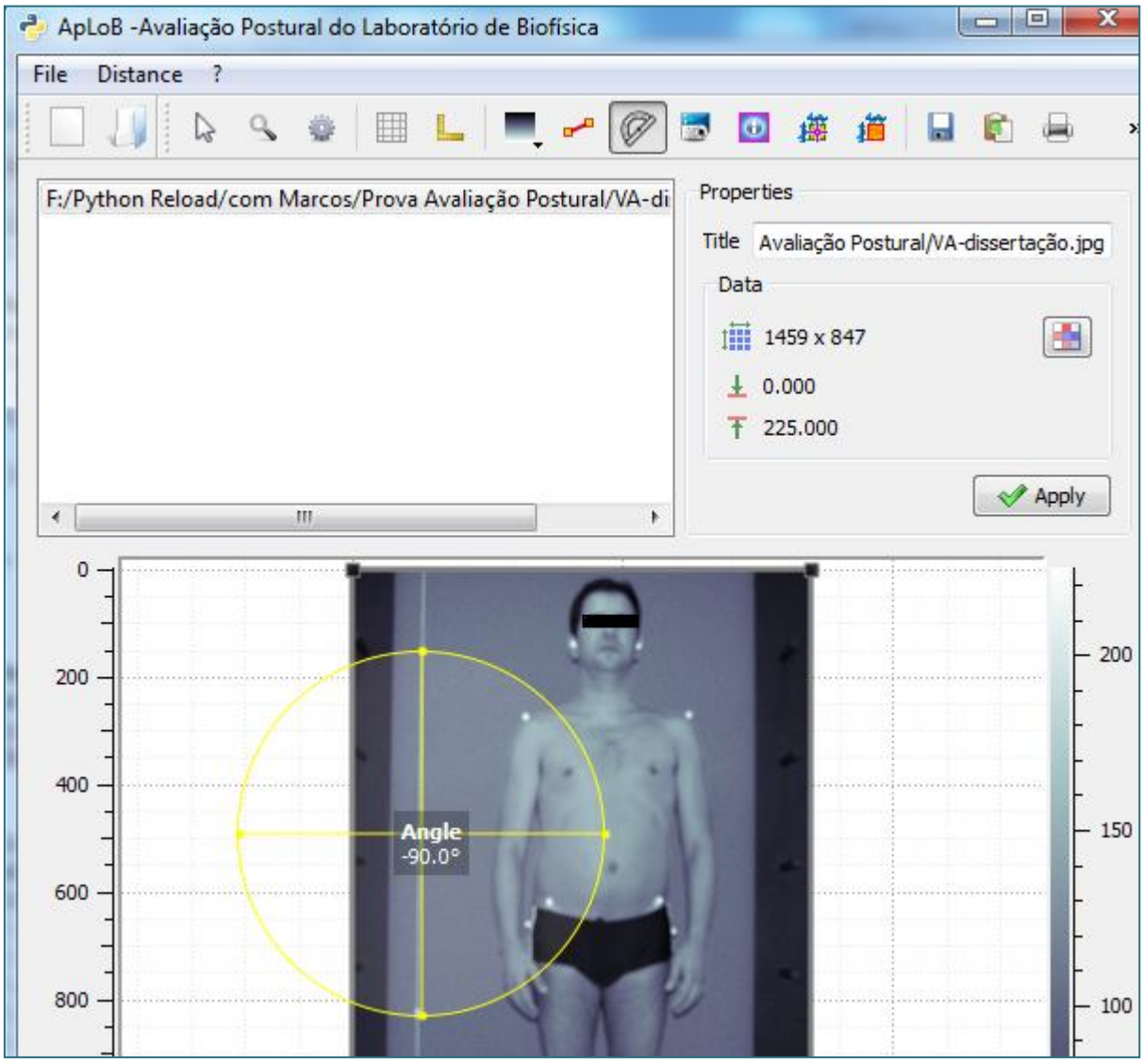
Figura 21 – Abrir arquivo



Fonte: elaboração própria

Calibrar imagem: com a imagem aberta se seleciona a ferramenta ângulo e se desenha uma linha entre os dois pontos do fio de prumo (que conformam o metro), se pode obter a medida em graus que representa a inclinação da imagem.

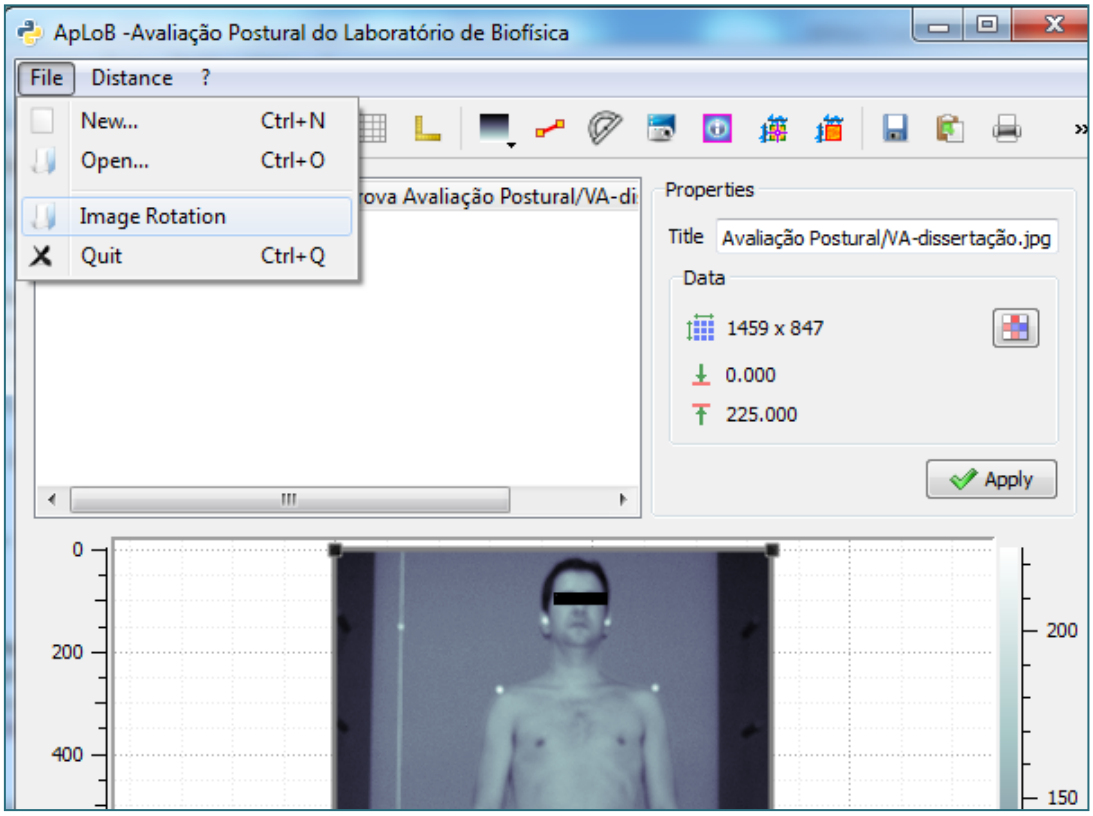
Figura 22 – Medir inclinação



Fonte: elaboração própria

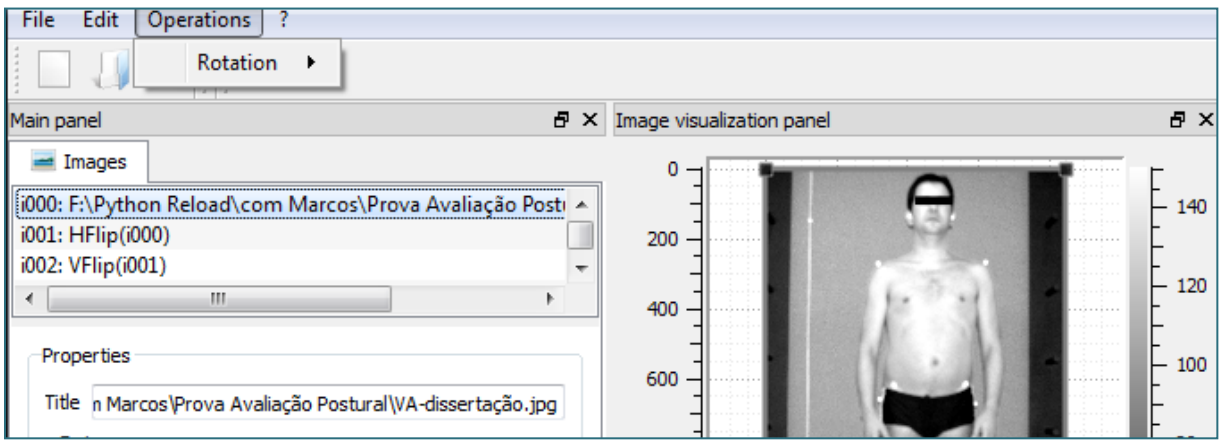
Rotar Imagem: No menu File, Image Rotation(fig.23), nos leva a outra janela onde abriremos o arquivo ao que acabamos de medir a inclinação. Vamos ao menu Operations, Rotation (fig.24) e se abre um quadro onde pode-se inserir a quantidade de graus que se desejam para rotar. Procede-se a armazenar a imagem (pode se usar um novo nome ou indicar um novo local de armazenamento).

Figura 23 – Abrir janela de Rotação de Imagem



Fonte: elaboração própria

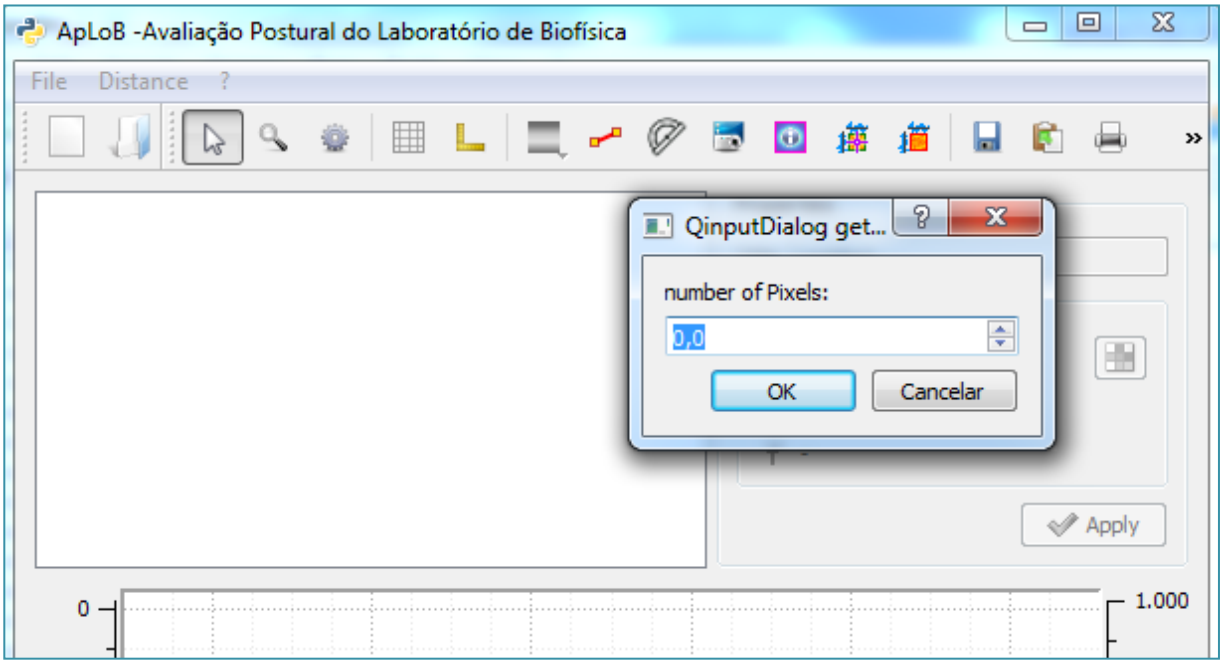
Figura 24 – Abrir caixa de rotação da imagem



Fonte: elaboração própria

Medir Segmentos: A ferramenta Segment permite desenhar uma linha sobre o fio de prumo (que tem a medida do metro) obtendo-se assim a quantidade de pixels que correspondem a essa medida. Depois, clicando em Distance, Calibration aparece uma caixa de diálogo onde se deve ingressar a quantidade de pixels obtidos. Por exemplo, se foram 600 pixels obtidos na medição do metro, se escreve essa quantidade na caixa, ele faz a conversão e armazena o resultado. Posteriormente se pode obter as medidas de cada segmento do corpo no menu Distance, Segment, escrevendo manualmente o número de pixels e o programa converte usando o resultado anteriormente armazenado (em Distance, Calibration). O processo ainda é mecânico, mas o resultado é muito confiável.

Figura 25 – Conversão pixels em centímetros



Fonte: elaboração própria