MedSquare: Plataforma modular para exploração de imagens tomográficas

Marcos Bonci Cavalca
Novembro de 2009

Instituto de Matemática e Estatística – Universidade de São Paulo
MAC0499 – Trabalho de Formatura Supervisionado

Orientador: Prof. Marcel Parolin Jackowski

Índice

1. Introdução

Próximo ao fim do primeiro semestre de 2008, procurei o professor Marcel na intenção de desenvolver uma atividade que me permitisse ganhar experiência prática e obter resultados concretos. À época, o professor estava à procura de um aluno que o auxiliasse na criação de um software intuitivo e padronizado de interface com o usuário, idéia que muito me animou. O software seria destinado a substituir a interface do programa de exploração de imagens tomográficas de cujo desenvolvimento o professor participava, criando uma nova vertente do projeto, de forma e aparência completamente novas.

Em pouco tempo, a idéia inicial se expandiu, e a integração de outros projetos do grupo de pesquisa a essa interface passou a integrar o horizonte. Freqüentando as reuniões quinzenais do grupo, pude notar uma evidente heterogeneidade nas características pessoais e na abordagem do desenvolvimento. Um traço, porém, era comum: a aversão ao desenvolvimento de interfaces com o usuário. Alguns meses mais tarde, analisando e comparando programas de código aberto para visualização de imagens tomográficas, tive essa noção de certa forma universalizada. Enquanto vários programas apresentavam conceitos interessantes, com implementações provavelmente complexas, apenas um apresentava uma clara preocupação com a praticidade da interface. Por outro lado, mesmo esse programa, que se destacava quanto à interface com o usuário, parecia pouco reutilizável do ponto de vista de desenvolvimento de outras aplicações.

Ao final do segundo semestre de 2009, após quase um ano participando das discussões do grupo de pesquisa (então chamado "Grupo ANIME", de Análise de Imagens Médicas do IME), e passando por muitas frustrações na tentativa de desenvolver o programa segundo os requisitos definidos pelo professor, tive uma conversa com o professor Alfredo Goldman, que naquele semestre ministrava a disciplina de Laboratório de Programação Extrema. A conclusão da conversa foi simples e direta: não havia chance de finalizar o projeto antes do fim do ano e ao mesmo tempo garantir sua longevidade. A tarefa era realmente mais complexa do que aparentava inicialmente, e o projeto parecia fadado a falhar.

Apesar de todos os contratempos, vimos esse contexto como uma oportunidade para aprendizado e inovação. Se formou então a proposta ambiciosa de uma plataforma modular, que substituiria a proposta inicial, solucionaria nossos principais problemas e auxiliaria também na criação de mais software por outros grupos. Separando o projeto em blocos e padronizando a interação entre eles, cada desenvolvedor poderia se concentrar mais em sua área de interesse. Ao mesmo tempo, poderíamos também buscar mais colaboradores, mesmo antes de obter uma versão completa do programa. Por fim, as partes do programa seriam reutilizáveis por pesquisadores e desenvolvedores em outros grupos, permitindo maior eficiência na pesquisa e criação de novas ferramentas na área.

Em outras palavras, este trabalho tratava-se inicialmente da criação de um programa, de forma tradicional e isolada, com foco apenas no produto final. Com o tempo e as frustrações que vieram, percebemos que o produto final exigia um foco no processo de criação em si, e a partir daí esse processo passou a ser central no projeto. Apesar de o trabalho desenvolvido não ter ocorrido todo de forma linear e seqüencial no tempo, este texto, a fim de simplificar a exposição de detalhes, é divido em seções relativas a certos passos, ligados por uma ordem cronológica natural. Essa estrutura de seções, por sua vez, se repete em duas partes principais, uma relativa à proposta inicial e outra relativa à proposta reformulada.

2. Conceito inicial e pesquisa decorrente

2.1. Motivação e proposta inicial

2.1.1. BioImage Suite

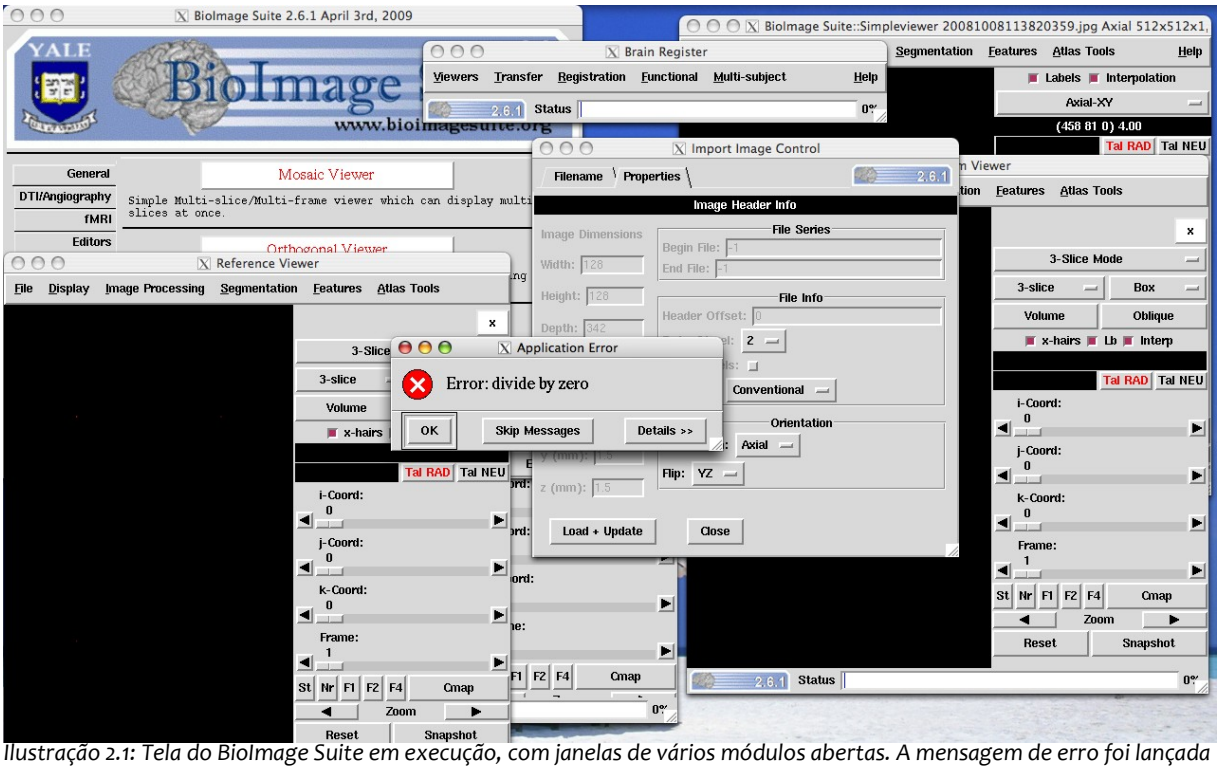

Ilustração 2.1: Tela do BioImage Suite em execução, com janelas de vários módulos abertas. A mensagem de erro foi lançada durante uma tentativa de abertura de um volume de imagens.

A motivação inicial para a criação do software, que mais tarde evoluiria para o projeto MedSquare, tratava-se exclusivamente do melhoramento do pacote BioImage Suite. O BioImage Suite é uma plataforma de software livre que reúne diversas ferramentas de visualização, manipulação e análise de imagens tomográficas, desenvolvida por uma equipe de acadêmicos na Universidade de Yale, nos Estados Unidos, entre eles o professor Marcel. Apesar de englobar uma grande quantidade de ferramentas e algoritmos avançados, o BioImage Suite sofre de uma deficiência evidente, no quesito de interação com o usuário.

Entre os fatores que contribuíram para essa deficiência, estão a escolha da biblioteca Tk e uma aparente falta de planejamento da interface. Como ponto positivo, a biblioteca Tk representa uma ponte direta até a saída na tela desde a biblioteca VTK, usada como base na manipulação de dados tridimensionais, e ao mesmo tempo permite a execução do programa em vários sistemas operacionais. Apesar disso, ela não se adequa ao desenvolvimento de interfaces complexas, devido ao seu caráter extremamente minimalista, que não aborda aspectos essenciais da organização de controles e janelas. Como conseqüência da interface desorganizada resultante, há uma grande dificuldade na localização de funcionalidades do programa, ao mesmo tempo em que a poluição visual diminui a eficiência na visualização e na interação software-usuário.

2.1.2. Proposta

Em junho de 2008, quando procurei o professor Marcel, o BioImage Suite já era há algum tempo utilizado no InRad, Hospital das Clínicas de São Paulo, onde o professor até hoje provê consultoria relativa à sua utilização. Partindo da própria experiência e da experiência dos colegas no InRad, a tarefa proposta por ele a mim foi a de criar uma interface gráfica completamente nova, capaz de reunir as ferramentas presentes no BioImage Suite, dando início a um "*fork*" do projeto inicial, encabeçado por acadêmicos da Universidade de São Paulo. Nos meses seguintes, o escopo da proposta foi gradualmente se estendendo, e passou a incluir também a integração de projetos desenvolvidos no grupo de pesquisa liderado pelo professor (Grupo ANIME).

Inicialmente, a previsão era de que essa nova interface estaria pronta dentro de um ou no máximo dois semestres. Gradualmente, à medida em que a proposta foi sendo atualizada e refinada, cada vez mais aumentando seu escopo de curto e de longo prazo, foi possível notar inconsistências e paradoxos que mais tarde levaram à sua completa reformulação. Esse processo de formulação da proposta, pesquisa e reflexão, levando mais tarde à reformulação, é descrito nesta primeira parte do texto.

2.2. Requisitos iniciais

2.2.1. Requisitos primários

Dentro do contexto descrito na seção anterior, isolamos os seguintes requisitos primários para o programa a ser criado:

- **Capacidade de acoplamento ao BioImage Suite:**
  O software deveria ser desenvolvido como um bloco auto-suficiente. Mesmo assim, esse bloco teria como objetivo principal atuar como peça de um bloco maior (composto por elementos do BioImage Suite), e por isso a consideração prioritária em qualquer decisão seria em função da capacidade do programa atuar dessa forma.

- **Intuitividade:**
  O segundo requisito imprescindível era a intuitividade de operação do software, que na prática se traduzia em uma interface bem planejada, simples e padronizada. Seria essencial que o usuário não fosse bombardeado por informações visuais que não interessassem a ele em um determinado instante.

- **Flexibilidade de aplicação e uso:**
  Assim como a plataforma original, esse novo programa deveria dar acesso a funcionalidades de diferentes aplicações e em diferentes contextos. Como recurso adicional, o programa deveria permitir várias formas de utilizar cada funcionalidade, se adaptando ao gosto e à necessidade do usuário.

- **Grande alcance de público:**
  O programa produzido deveria ser acessível, tanto para obtenção quanto em seu funcionamento, a profissionais em diversas áreas de atuação e com diferentes níveis de formação e experiência em relação à medicina e à informática.

2.2.2. Requisitos indiretos

A partir dos requisitos primários, identificamos os seguintes requisitos adicionais:

- **Componentes padronizados e reutilizáveis:**
  Para garantir ao mesmo tempo a interface padronizada e a capacidade de integração com os blocos de software já existentes, seria necessária uma padronização da própria estrutura interna relativa aos blocos da interface. Essa padronização permitiria também uma maior facilidade e confiabilidade no processo de testes de consistência do programa.

- **Multi-sistema:**
  Para garantir o acesso ao programa pelo maior número possível de usuários, o programa deveria ser capaz de ser executado em diferentes sistemas operacionais. Entre eles, os principais seriam Microsoft Windows, Mac OS X e Linux.

- **Licença aberta:**
  Também servindo à abrangência do universo de usuários em potencial, o programa deveria ser disponibilizado sob uma licença *open source*. Por seu caráter universal e sua popularidade, a licença GPL seria utilizada.

- **Internacionalização:**
  Ainda no intuito de garantir o acesso ao maior número de usuários, o programa deveria apresentar a possibilidade de sua tradução para vários idiomas.

- **Processamento paralelizado:**
  Para garantir a flexibilidade de aplicação, o programa deveria permitir que seus blocos executassem operações de processamento paralelizado. O principal uso dessa propriedade, imaginado inicialmente, seria em funcionalidades de simulação física de processos biológicos e de aquisição de imagens.

- **Unidade nas escolhas:**
  Como conseqüência do requisito de acoplamento a blocos do BioImage Suite (e possivelmente outros blocos futuros, em formato semelhante), o projeto não teria liberdade para escolha própria de tecnologias a serem utilizadas. Não só as tecnologias utilizadas no BioImage Suite restringiam as escolhas possíveis, como também era necessário que as escolhas feitas a partir destas fossem aprovadas e compartilhadas pelos desenvolvedores dos projetos associados.

2.3. Tecnologias e conceitos pesquisados

2.3.1. Linguagens de programação

- **Python:**
  Logo no início, a linguagem Python foi considerada para a construção do novo programa. Do ponto de vista pessoal, seria algo positivo, considerando meu conhecimento prévio da linguagem e minha motivação para o aprofundamento. Além disso, por ser uma linguagem interpretada, seria automaticamente multi-plataforma. No entanto, temendo que a ausência de rigidez dessa linguagem pudesse se refletir em uma falta de consistência no projeto, decidimos procurar outra alternativa. Além disso, o uso dos *wrappers* em Python para acesso à biblioteca VTK (criada em C++, e base do BioImage Suite), representava um risco para a estabilidade, já que aparentemente esses wrappers não recebiam tanto suporte e atenção geral dos desenvolvedores quanto os outros elementos da biblioteca.

- **Java:**
  Apesar de Java também não ser a linguagem de implementação da biblioteca VTK, decidimos tentar utilizá-la, por seu conhecido suporte a paralelismo, sua integração mais direta com a linguagem C++, e também sua popularidade. Assim como Python, Java também permitiria um programa automaticamente multi-plataforma. Em pouco tempo, porém, descobrimos que o suporte a Java para vários componentes do VTK era sabidamente instável, pouco confiável ou de origem incerta. Decidimos então mais uma vez partir para outra alternativa.

- **C++:**
  Finalmente, após as duas tentativas frustradas, decidimos utilizar C++, única alternativa restante. Por um lado, operaríamos diretamente com a biblioteca VTK, sem *wrappers* envolvidos. Por outro lado, teríamos que passar pelo processo de compilação a cada teste do programa, diminuindo drasticamente a eficiência do desenvolvimento. Além disso, por se tratar de uma linguagem com vários aspectos de baixo nível, herdados integralmente da linguagem C, C++ não era a linguagem mais adequada para o desenvolvimento de uma interface.

2.3.2. Toolkits de interface com o usuário

- **Tk:**
  Toolkit utilizado no próprio BioImage Suite. Descartado logo no início por seu minimalismo extremo e sua completa falta de integração a padrões visuais de sistemas operacionais. Basicamente, foi utilizado como referência para as características das quais devíamos nos afastar.

- **Swing:**
  Exclusivo para Java, o Swing foi considerado por se tratar do *toolkit* padrão para essa linguagem. Foi descartado assim que descartamos Java.

- **SWT (Standard Widget Toolkit):**
  Também exclusivo para Java, o SWT foi considerado por se tratar do *toolkit* padrão na plataforma Eclipse de desenvolvimento. Assim como o Swing, foi descartado logo que descartamos Java.

- **FLTK (Fast, Light Toolkit):**
  Simples, leve e de boa aparência, o FLTK era uma das alternativas reais. No entanto, é um *toolkit* pouco popular, o que dificulta seu aprendizado. Sua simplicidade também era potencialmente um problema, pela falta da capacidade de configuração de detalhes possivelmente importantes.

- **wxWidgets:**
  Terceiro *toolkit* multi-plataforma mais popular, o wxWidgets também era uma opção real. Porém, não apresentava vantagens sobre o GTK+, largamente mais popular. Por isso, foi descartado.

- **GTK+ (GIMP Toolkit):**
  Ao fim da pesquisa sobre *toolkits,* as duas alternativas "finalistas" eram o GTK+ (base do ambiente gráfico Gnome) e o Qt (base do ambiente gráfico KDE). Inicialmente, não havia vantagens ou desvantagens aparentes na comparação entre os dois, e a escolha parecia uma questão de gosto pessoal. Um ponto que parecia favorecer o GTK+ em discussões era sua licença (LGPL desde o início), mas o próprio Qt já estava universalmente disponível sob a GPL havia vários anos. Mais tarde, foi possível constatar diversas vantagens apresentadas pelo Qt, que nos levaram a descartar o GTK+.

- **Qt:**
  Com uma ativa comunidade de desenvolvedores e uma documentação atualizada e bem detalhada, o Qt já se apresentava como a melhor alternativa. Como pontos favoráveis adicionais, o Qt oferecia um fino controle sobre detalhes e ainda apresentava uma integração visual mais profunda com os sistemas operacionais. Finalmente, descobrimos que o Qt tinha suporte "oficial" do VTK, e então não tivemos mais dúvidas de que era a melhor escolha. Mais tarde, já durante a segunda encarnação do projeto, vieram mais duas propriedades para o Qt, possivelmente vantajosas: adoção da licença LGPL e suporte ao sistema Symbian (utilizado em aparelhos móveis).

2.3.3. Bibliotecas para computação gráfica tridimensional

Por ser a base de todo o BioImage Suite (além de vários programas da mesma área), vimos o VTK como única alternativa real no quesito de computação gráfica. Outra alternativa seria o uso direto de OpenGL, mas isso aparentemente representaria apenas uma dificuldade a mais, já que o que realmente nos interessava eram as funcionalidades já implementadas no VTK, baseadas sobre o próprio OpenGL.

Durante a primeira fase do projeto (antes da reformulação), essa escolha veio a revelar muito mais implicações negativas do que o previsto, provavelmente constituindo um dos principais fatores para o fracasso prematuro da primeira proposta. Essas implicações negativas são descritas mais adiante, nas seções sobre dificuldades e conclusões, ao fim desta parte.

2.3.4. Plataformas de desenvolvimento de software

À medida em que foi se cristalizando a intenção de integrar também projetos de outros membros do grupo ANIME à nova interface, em um contexto em que o programa seria praticamente um grande bloco, lapidado de vários ângulos ao mesmo tempo, sentimos a necessidade de padronizar nossas escolhas não só com relação às bibliotecas utilizadas, mas também com relação às ferramentas que iriam compor nosso ambiente de desenvolvimento. Seguem algumas considerações sobre as alternativas cogitadas:

- **Editores de texto:**
  A alternativa mais simples seria o uso de um editor de texto avançado, como o Gedit (editor padrão do Gnome), associado a ferramentas específicas, como o Qt Designer (criador de janelas e *widgets* em Qt) e o Cmake (sistema de compilação multi-plataforma). Enquanto esse conjunto poderia funcionar bem em regime temporário, provavelmente seria inadequado a longo prazo, pois não permitiria de forma prática uma organização hierárquica de elementos do projeto, e também não integraria as diferentes atividades de desenvolvimento.

- **Netbeans:**
  O IDE (ambiente integrado de desenvolvimento) Netbeans possuía uma interface simples e bem organizada, uma estrutura modular que permitia integração de ferramentas independentes, e aparentemente permitiria um bom gerenciamento de projeto. Por outro lado, seu foco claramente era a linguagem Java, e sua popularidade era baixa em relação ao seu concorrente mais direto, o Eclipse. Por esse motivo, não só seu futuro parecia mais incerto, como também sua disponibilidade de *plug-ins* era muito menor. Em resumo, ele não apresentava uma vantagem clara sobre o concorrente mais óbvio.

- **Eclipse:**
  No contexto do universo de todas as linguagens de programação, o Eclipse era o IDE mais popular. Para o iniciante, alguns de seus aspectos (ou aspectos de seus *plug-ins*) podiam ser um pouco confusos, mas o aprendizado em geral parecia rápido. Para quase todo tipo de atividade de desenvolvimento, encontrava-se um *plug-in* disponível. Porém, o *plug-in* para Qt se mostrou inutilizável da forma como se encontrava, pois era totalmente instável, travando o programa freqüentemente. Como queríamos um programa que integrasse toda a atividade de desenvolvimento, não ficamos satisfeitos.

- **KDevelop:**
  IDE oficial do KDE, o Kdevelop se mostrava como uma boa alternativa, pois focava em Qt, além de oferecer suporte a várias linguagens. No entanto, sua versão mais recente na época, foco de seus desenvolvedores, encontrava-se em um estado questionável, com a ausência de várias funcionalidades importantes, além de uma aparente instabilidade. Assim, seguimos para a outra alternativa.

- **Qt Creator:**
  Recém-criado, porém promissor pela proposta e pelo histórico de perfeccionismo de seus criadores, o Qt Creator se apresentava como o IDE oficial para Qt. Sua proposta se baseava em uma interface totalmente simplificada e, mesmo usando uma arquitetura de *plug-ins*, seu foco explícito era em programação em Qt na linguagem C++. Assim, o Qt Creator foi o IDE escolhido para o desenvolvimento.

2.3.5. Métodos e dispositivos de entrada alternativos

Ao fim do segundo semestre de 2009, com a os elementos principais – linguagem de programação, *toolkit* de interface, biblioteca gráfica 3D e ambiente de desenvolvimento – todos definidos, o projeto parecia bem encaminhado (conclusão que mais tarde mostrou-se equivocada). A criação da interface em si parecia então uma tarefa simples, e por isso decidimos adicionar um elemento de inovação.

Trocando idéias sobre os elementos que deveriam ser encontrados em uma interface ideal, surgiu a idéia de incorporar no programa novas formas de interação através do uso de tecnologias novas e radicalmente diferentes das tradicionais (mouse e teclado), porém acessíveis. Havia na época um aparelho que reunia essas qualidades: o Wii Remote (controle remoto para o aparelho de *video-game* Wii, da Nintendo). Durante os meses seguintes, o uso do Wii Remote permaneceu como um objetivo de curto prazo. Na seção seguinte são descritas as atividades realizadas com relação a esse objetivo.

2.4. Atividades realizadas

2.4.1. Visita ao Hospital das Clínicas

Ainda no segundo semestre de 2009, fizemos uma visita em grupo ao InRad, no Hospital das Clínicas de São Paulo, guiada pelo professor Marcel. Durante essa visita, pudemos observar em funcionamento o mais potente aparelho de ressonância magnética do país. Os aparelhos de tomografia, e o processo de aquisição de imagens tomográficas em si, fogem do escopo deste projeto; porém, foi também possível observar em funcionamento as interfaces dos programas que controlavam esses aparelhos e acompanhavam o processo de aquisição de imagens.

Foi possível notar que, assim como no BioImage Suite, a interface com o usuário não era uma prioridade, e que provavelmente era necessário um treinamento para o usuário/operador se familiarizar com a ferramenta, e mesmo assim a eficiência no controle e na operação (em ações como processamento instantâneo de imagens) não seria realmente otimizada.

2.4.2. Pesquisa e experimentação com o Wii Remote

O Wii Remote é um dispositivo de aparência semelhante a um controle remoto comum, com seis botões pequenos – quatro de ação (“1”, “2”, “+”, “-”) e dois de operação (“Home” e “Power”) – um botão grande (“A”, que se encaixa ao polegar do usuário) e um botão em forma de gatilho (“B”, que se encaixa ao indicador do usuário). Além dos botões, o controle tem um pequeno alto-falante embutido.

Porém, o verdadeiro poder do controle se encontra em seus acelerômetros e em sua câmera infravermelha. Enquanto os acelerômetros são capazes de detectar a movimentação no espaço e a rotação do controle, com boa precisão, a câmera permite aprimorar no tempo essa precisão utilizando pontos de emissão de luz infravermelha como referência, evitando que movimentos bruscos desestabilizem seu sistema de referência espacial.

Utilizando esse poder de orientação espacial, associado a uma boa interface de software com o usuário, poderíamos estudar novos meios para interação com imagens tridimensionais, úteis em situações como apresentações, demonstrações, ou outras tarefas em que o conjunto mouse-e-teclado é pouco intuitivo para navegação tridimensional.

Logo após o surgimento da idéia da utilização do Wii Remote no projeto, seguida por uma breve pesquisa, foi possível constatar que a interação entre esse controle e o computador não só era possível, como também era relativamente simples. A característica mais importante do controle para que isso fosse possível era sua utilização do padrão Bluetooth para comunicação. Outro ponto interessante era a presença, já naquela época, de programas para testes e até uso do Wii Remote para o controle de computadores pessoais comuns.

Explorando essa possibilidade de conexão com computadores comuns, outras pessoas já haviam concebido formas bastante interessantes de uso do Wii Remote. Talvez a mais criativa dessas pessoas, o pesquisador Johnny C. Lee apresentou grandes idéias de utilização simples, que poderiam ser integradas no projeto. Entre elas, duas se destacavam: uma lousa virtual sobre uma superfície projetada, e um par de óculos que permitia a visão em perspectiva real em monitores comuns. Apesar de revolucionários para usuários comuns de computador, essas idéias eram relativamente simples de serem implementadas.

Para a lousa virtual, os requisitos adicionais eram apenas uma superfície plana e lisa, ou pouco rugosa (onde seria feita a projeção de imagens da tela do sistema, ou mesmo em um monitor plano), e de um a quatro LEDs infravermelhos alternáveis (montados nas pontas de canetas com botões para acionamento e com pilhas embutidas, por exemplo).

Bastaria uma calibração inicial, definindo para o sistema a posição e o tamanho da lousa virtual através de pontos de referência, e a partir daí a câmera do Wii Remote, fixado em um ponto de boa perspectiva, registraria a posição em que os pontos de luz infravermelha surgissem (acionando-se as canetas). O Wii Remote não transmite as imagens captadas, e sim um conjunto de um até quatro pontos já extraídos a partir das regiões mais luminosas da imagem. Assim, o software teria apenas que fazer cálculos trigonométricos simples para relacionar os dados de entrada ao seu significado real.

Para a implementação dos óculos de perspectiva, bastaria um par de óculos qualquer (ou mesmo apenas a armação), adicionado de um emissor de luz infravermelha em cada uma das duas extremidades laterais. Assim como no caso da lousa virtual, a câmera infravermelha do Wii Remote ficaria fixa, e funcionaria como um observador informando ao software sobre os movimentos da cabeça do usuário. Dessa forma, também com cálculos simples, seria possível ajustar a projeção bidimensional de acordo com o ponto de vista do usuário, fazendo com que a tela do monitor aparentasse ser uma janela real. Como conseqüência principal, o usuário teria uma impressão muito mais realista a partir de qualquer imagem tridimensional, facilitando a visualização.

Essas idéias poderiam ser aplicadas também usando-se outros aparelhos, como *webcams* comuns, assim como também outras idéias que utilizassem apontadores de laser visível comum. Mas, através do Wii Remote, nós não só poderíamos evitar a inconveniência da interferência da luz visível na interação, como também poderíamos integrar todos esses métodos de interação em apenas um aparelho de alta precisão e custo relativamente baixo.

Durante o tempo em que a utilização do Wii Remote esteve entre nossos objetivos de curto prazo, foi comprado um controle (pois não havia um *video-game* Wii disponível para testes), e foram testados sua conexão nos sistemas Linux e Windows, e seu sistema interno de localização por pontos de referência infravermelha (utilizando velas e luzes de Natal). Também forma montadas quatro canetas com LEDs infravermelhos, pilhas embutidas e botões de acionamento.

Infelizmente, a aplicação dessas idéias dependia do programa de visualização em já funcionamento, no qual pudessem ser feitas extensões desse tipo. Assim, logo que percebemos a dimensão real da dificuldade da criação desse programa, as idéias relativas ao Wii Remote foram adiadas por tempo indefinido. Atualmente, elas se encontram incluídas no nosso escopo de longo prazo, sem previsão para início de implementação.

2.4.3. Análise de programas de proposta semelhante

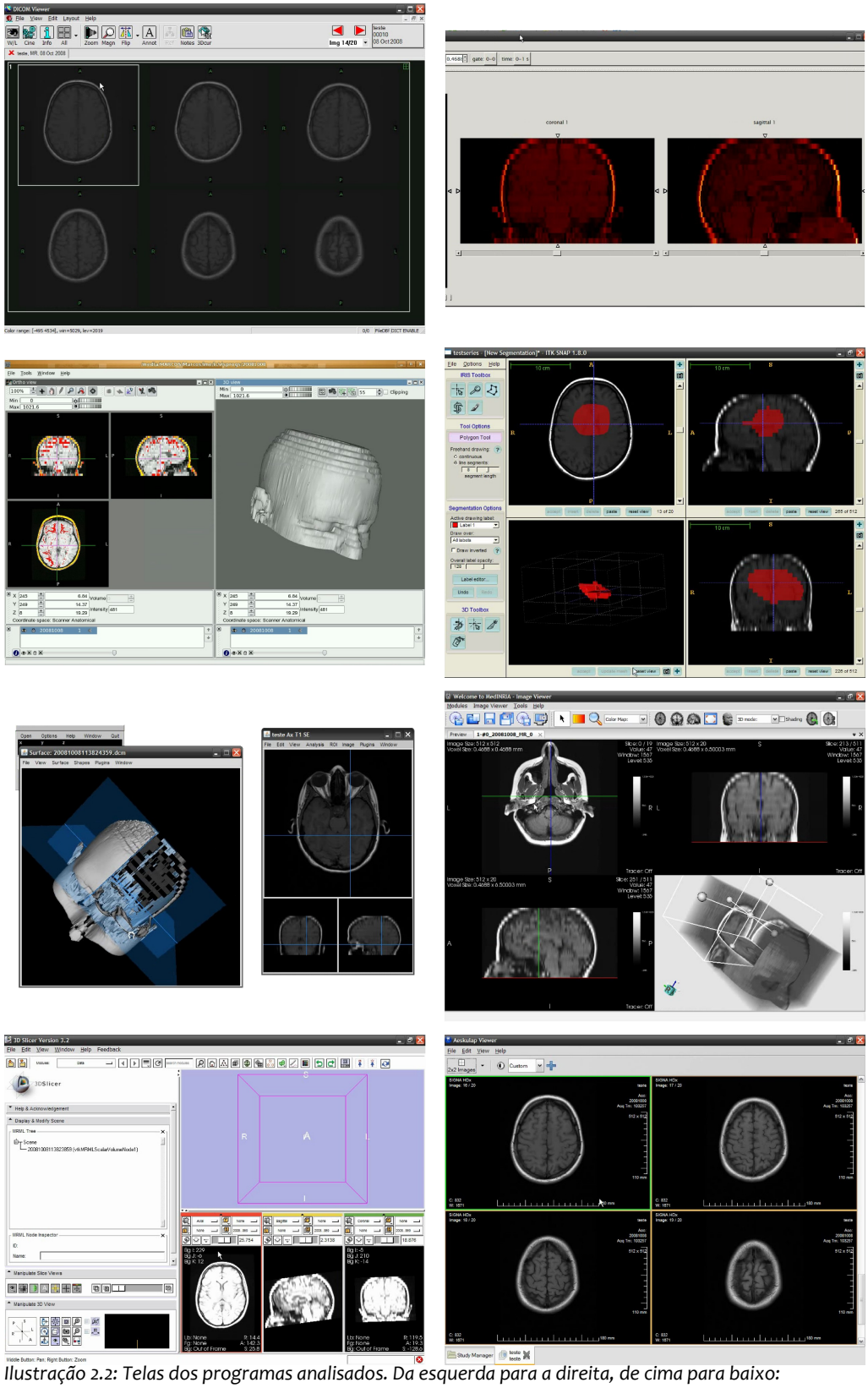

Ilustração 2.2: Telas dos programas analisados. Da esquerda para a direita, de cima para baixo: Aeskulap, Amide, FSL, ITK-SNAP, Mango, MedINRIA, Slicer, UniPACS Viewer.

Com o tempo, passamos a notar que, mesmo com as decisões sobre tecnologias para o projeto já tomadas, o projeto não prosseguia como previsto. A tarefa não parecia mais tão simples quanto antes, e por isso decidimos dedicar tempo especificamente para a análise de programas já existentes, tentando identificar pontos positivos, falhas e possíveis causas para essas falhas.

Inicialmente, nossa intenção era analisar os programas que realmente estão entre os mais utilizados, que na prática são, em sua maioria, provenientes das próprias empresas fabricantes dos aparelhos de tomografia. Infelizmente, não possível encontrar nenhum desses programas que pudesse ser utilizado sem custos adicionais, em regime temporário ou limitado, ou sob condição alguma. Assim, prosseguimos para os programas de licença *open source*. No intuito de reunir um conjunto que representasse de forma realista essa classe de programas, selecionamos alguns nomes a partir da lista de programas mais usados, no site idoimaging.com. A seguir são enunciadas as principais características observadas em cada um desses programas.

- **Aeskulap:**
  Simplicidade na interface, porém simplicidade também nas funcionalidades disponíveis. Aparentemente trata-se de um programa antigo, para uso em computadores comuns em uma época em que o poder de processamento típico não era suficiente para processamento tridimensional em tempo real. Realiza razoavelmente bem as operações que oferece, que focam apenas na visualização corte a corte (ou seja, a seqüência de cortes dada pelo aparelho de tomografia).

- **Amide:**
  Exibe cortes ortogonais reconstruídos a partir dos cortes originais (eixos axial, sagital e coronal). Logo ao abrir a imagem de testes, a janela do programa se expande além dos limites da tela, tornando impossível a sua utilização. Sem qualquer flexibilidade na interface, o programa provavelmente foi criado com imagens menores ou telas maiores em mente.

- **FSL:**
  Interface medianamente simples e eficiente. Trabalha com cortes em seqüência, cortes ortogonais e reconstrução tridimensional. Apresenta um certo desperdício de espaço e não distribui suas sub-janelas automaticamente. A julgar pelo conjunto de programas analisados, representa um programa típico.

- **ITK-SNAP:**
  O único nesse conjunto que, mesmo incluindo funcionalidades complexas, apresenta uma clara preocupação com a simplicidade da interface. Sua tela principal é constituída por duas partes: o painel de ferramentas à esquerda e a área de visualização (cortes ortogonais) à direita. Seu foco principal é a segmentação de regiões cerebrais a partir de imagens tomográficas. A área de visualização inclui também um quarto quadro, onde é exibido o resultado geométrico tridimensional da segmentação. Exclui eficientemente da visão do usuário os elementos desnecessários a cada momento, e explora muito bem as ações do mouse sobre a visualização. Infelizmente, não é modular e não é flexível em sua interface, provavelmente por ter foco fechado em suas funcionalidades atuais.

- **Mango:**
  Características semelhantes às do FSL, com a adição de algumas funcionalidades sobre a reconstrução tridimensional. Sua divisão em janelas flutuantes é pouco prática. A janela de cortes ortogonais é pouco flexível. A reconstrução tridimensional é extremamente lenta em alguns casos.

- **MedINRIA:**
  Também semelhante ao FSL, porém reúne vários módulos além do de visualização. Cada módulo opera como um programa isolado. Permite vários modos no quadro de exibição tridimensional, além de apresentar controles presentes dentro dessa imagem tridimensional, claramente implementados através da biblioteca VTK.

- **Slicer:**
  Provavelmente o mais completo de todos nesse conjunto. Apesar de aparentemente organizada, sua interface é inicialmente confusa e exige treinamento. Reúne uma grande quantidade de módulos, que se apresentam na região esquerda da tela quando invocados.

- **UniPACS:**
  Semelhante ao Aeskulap, porém com capacidade de organização de operar em sistemas de armazenamento de dados tipos PACS. Aparentemente tem seu foco principal nessa capacidade, e por isso oferece apenas visualização simples (quadro a quadro). Porém, cumpre bem essa tarefa.

2.4.4. Implementação do software

Inicialmente, imaginamos que a implementação representaria a maior parte do trabalho a ser realizado. No entanto, a implementação em si acabou sendo um mero detalhe, que por outro lado motivou todo o processo de planejamento não previsto no início, terminando por não gerar frutos concretos de curto prazo.

Em primeiro lugar, o trabalho na implementação se iniciou apenas meses após a formulação da primeira proposta, devido aos vários pontos importantes que julgamos necessário pesquisar antes da programação em si. Depois de iniciado, esse trabalho foi lento e intermitente, devido às dificuldades que surgiram no caminho. Essas dificuldades são descritas na seção mais ao fim desta parte.

2.5. Resultados obtidos

2.5.1. Resultados práticos

Ao final dessa primeira fase do projeto, não havia resultados práticos concretos.

2.5.2. Resultados indiretos

Apesar da falta de resultados concretos, esses dois semestres de pesquisas, reflexões e dificuldades práticas levaram a uma compreensão muito mais profunda não só dos problemas a serem solucionados no desenvolvimento do projeto, mas também das próprias motivações que levaram à sua concepção.

Nosso objetivo era criar uma ferramenta que cumprisse seu papel de uma determinada forma. Seguindo o caminho da inércia inicial, corríamos um sério risco de construir uma ferramenta que cumprisse seu papel da mesma forma que aquela a qual ela deveria substituir (BioImage Suite). Procurando entender o processo de criação, identificar o fracasso e entender os motivos que levaram até ele, foi possível evitar que esse fracasso ocorresse muito mais tarde, evitando assim o desperdício de muito tempo de trabalho.

Mas o fruto mais importante dessa primeira fase foi, indo além da identificação do fracasso, a identificação da oportunidade. Analisando os programas disponíveis, notamos que nenhum deles satisfazia o que nós mesmos esperávamos, e vimos que o nosso projeto poderia preencher essa lacuna, se passássemos a considerá-lo como uma iniciativa de longo prazo.

2.6. Dificuldades e encontradas

Entre as dificuldades encontradas, duas delas se destacaram: a ineficiência da programação em C++ e a dificuldade de uso do VTK. Por um lado, a criação do programa em C++ exigia a compilação do programa a cada tentativa de teste manual, e cada compilação, usando VTK, podia durar até dois minutos. Como o desenvolvimento inicial tratava apenas de uma janela com componentes básicos (como a área para cortes ortogonais), a verificação manual do funcionamento era essencial para a continuação do processo.

Por outro lado, por sua estrutura em *pipelines* e divisão convoluta de atribuições entre classes, o funcionamento do VTK parecia ir contra noções pessoais de intuitividade. Complicando ainda mais o problema, a documentação oficial (tanto a documentação gerada automaticamente quanto os livros sobre seu uso) pouco ajudavam na orientação de iniciantes. Somando essa dificuldade à impossibilidade de compilações freqüentes (que permitiriam aprendizado por tentativa-e-erro) o desenvolvimento chegou a um estado de quase estagnação.

Outras dificuldades, pouco relacionadas entre si, atrapalharam bastante, de forma pontual no tempo, o progresso do projeto. Uma delas era a freqüente necessidade de recompilação do VTK em vários sistemas operacionais e sob várias condições (por exemplo, incluindo ou não suporte a uma determinada linguagem ou *toolkit*). Essa compilação demorava horas, e muitas vezes travava próximo à metade ou até o fim do processo, e tinha que ser reiniciada "do zero". Várias vezes não ficava claro o motivo do erro ou sua solução.

Quanto aos sistemas operacionais, cada um apresentou suas próprias dificuldades. No sistema Linux, os *drivers* de saída de vídeo e de suporte a Bluetooth eram freqüentemente problemáticos. No sistema Windows, a compilação utilizando MinGW (Minimalist GNU for Windows) muitas vezes falhava por motivos obscuros, enquanto a compilação no Visual Studio exigia a recompilação de outras bibliotecas pelo compilador Microsoft, exigindo mais horas de processamento, também sujeitas a interrupções inesperadas durante o processo.

2.7. Conclusões alcançadas

2.7.1. Redundância parcial da proposta

Por um lado, a proposta buscava solucionar um problema claramente presente. Por outro lado, essa solução buscada tinha várias características em comum com as soluções buscadas por outros projetos, sendo que nenhum deles reutilizava a solução de outro, e também o nosso projeto não permitiria (ao menos inicialmente) essa reutilização. Na soma de todos os projetos dessa área, havia uma repetida "reinvenção da roda", na qual nenhuma das "rodas" ficava perfeita.

Enquanto a possibilidade da reutilização de código-fonte existente relativo a interfaces não parecia realista para o nosso projeto, tínhamos a oportunidade de tomar uma iniciativa no sentido de quebrar essa reinvenção, oferecendo a "roda" em questão pronta para ser usada em algo maior, ou para ela própria ser melhorada.

2.7.2. Tecnologias escolhidas e dinâmica de trabalho em grupo

Fechando o círculo que se iniciou com o BioImage Suite, passou por todas as tecnologias, problemas e análises vistos, finalmente retornamos ao início, identificando a raiz do problema através do conhecimento adquirido. Apesar de se manifestar de formas diferentes, descritas nas seções acima, o problema que deu origem não só aos nossos problemas principais como provavelmente também aos defeitos do próprio BioImage Suite, é a falha na divisão clara e efetiva do trabalho. Essa falha foi iconificada em grande parte pela biblioteca VTK nessa primeira fase do projeto.

É possível notar na evolução dessa primeira fase que o VTK dirigiu, direta ou indiretamente, grande parte das escolhas. Essa biblioteca é criada por profissionais de alta qualificação, preocupados com a visualização, mas provavelmente pouco preocupados com a interface ao redor dessa visualização. Levando isso em consideração, podemos prever que a direção de decisões relativas à interface em função dessa biblioteca pode levar a incongruências. Sob a interpretação pessoal, foi exatamente o que ocorreu. Enquanto o objetivo era o de introduzir um elemento diferencial na fórmula do programa, seus elementos iniciais mantinham a tendência original.

2.7.3. Necessidade de reformulação

Ao final do segundo semestre de 2009, sob forte influência dos conceitos introduzidos na disciplina de Laboratório de Programação Extrema, além da análise dos outros programas e das idéias adicionais introduzidas pelo professor Marcel no decorrer projeto, sentimos uma necessidade de reformulação na abordagem. Em uma conversa decisiva com o professor Alfredo Goldman, ele apresentou sua conclusão de que era necessário diminuir o escopo do projeto, ou então estar ciente de que ele não seria terminado no tempo previsto inicialmente.

A princípio, a segunda opção não parecia promissora, mas também a primeira opção não era realmente aceitável, já que levava à redundância da maior parte do esforço desenvolvido e a uma longevidade improvável. Nos dias que se seguiram, porém, foi tomando forma uma abordagem em que a segunda opção (de não completar o projeto em curto prazo) parecia totalmente aceitável. Propus então ao professor Marcel essa nova abordagem, que foi aceita, e daí surgiu o que hoje é o projeto MedSquare. Essa nova abordagem é o tema da segunda parte deste texto.

3. Conceito reformulado e estado atual

3.1. Redefinição da proposta

Tendo em nossas mãos o conhecimento adquirido durante a primeira fase do projeto, não só concluímos que poderíamos alcançar resultados se cortássemos o mal pela raiz, como também sentimos confiança suficiente para nos empenharmos em uma iniciativa muito maior, tratada daí em diante como "plataforma". A característica principal dessa nova iniciativa seria exatamente a resposta contra o problema-raiz identificado: a modularidade da plataforma, contra a obscuridade nas fronteiras.

Através dessa arquitetura modular, poderíamos ao mesmo tempo solucionar nossos próprios problemas, e também apresentar uma solução para problemas de outros, de forma que o usuário final seria também beneficiado por um resultado mais completo e integrado. "Dividindo e conquistando", seria possível tratar problemas de forma mais isolada, para ao fim obter um resultado mais abrangente.

Em termos mais concretos, a plataforma seria composta por blocos capazes de execução independente, mesmo em casos em que os blocos não tivessem utilidade em forma isolada. Entre esses blocos, estaria uma interface de programação, composta por uma biblioteca própria, encarregada de unificar a forma de comunicação interna. Dessa forma, cada bloco teria uma independência muito maior em suas escolhas e cada programador ou equipe teria uma liberdade muito maior no processo de desenvolvimento. Também cada bloco poderia ser testado independentemente de outras partes, adicionando dinamicidade ao processo e garantindo a consistência desde o nível local.

Grande parte da inspiração na lapidação dessa nova abordagem veio da plataforma Eclipse de programação, que integra uma grande quantidade de ferramentas produzidas de forma independente, mas que ao fim trabalham em conjunto. Assim como na plataforma Eclipse, nossa meta agora era buscar um estado em que outros programas, produzidos de forma independente, poderiam se acoplar à plataforma, e permitir a produção de resultados de difícil obtenção (ou mesmo impossíveis) sem essa integração.

Com esse horizonte mais ambicioso à frente, a iniciativa tomou vida própria e deixou de se tratar apenas do trabalho sob a responsabilidade pessoal do trabalho de conclusão de curso. A partir daí, o projeto englobaria também os outros projetos desenvolvidos no grupo, e o próprio grupo passaria a se identificar pelo objetivo comum dessa plataforma unificada. Também nesse ponto o professor Choukri Mekkaoui, do Massachusetts General Hospital, na Universidade de Harvard, convidado pelo professor Marcel, passou a participar do planejamento e do gerenciamento da nova plataforma.

3.2. Requisitos atuais

O processo de detalhamento e definição precisa dos requisitos atuais ocorreu de forma gradual ao longo do semestre, mas, para fins expositivos, ele será resumido aqui em uma seção.

A partir das expectativas sobre os prováveis futuros usuários da plataforma, três perfis básicos foram definidos: clínicos, pesquisadores e desenvolvedores. O grupo dos clínicos se refere aos usuários finais, que não adicionam funcionalidades à plataforma. O grupo dos pesquisadores se refere aos usuários intermediários, que impulsionam a plataforma para novas direções, possivelmente adicionando funcionalidades. Por fim, o grupo dos desenvolvedores, por definição encarregado da adição e manutenção de funcionalidades, é tratado também como usuário, pois utiliza blocos produzidos por outros.

Baseando o planejamento no conceito de modularidade, eliminando assim o requisito de unidade de escolhas, e orientados por esse modelo de três perfis, detalhamos os requisitos da forma como se vê na ilustração que se segue.

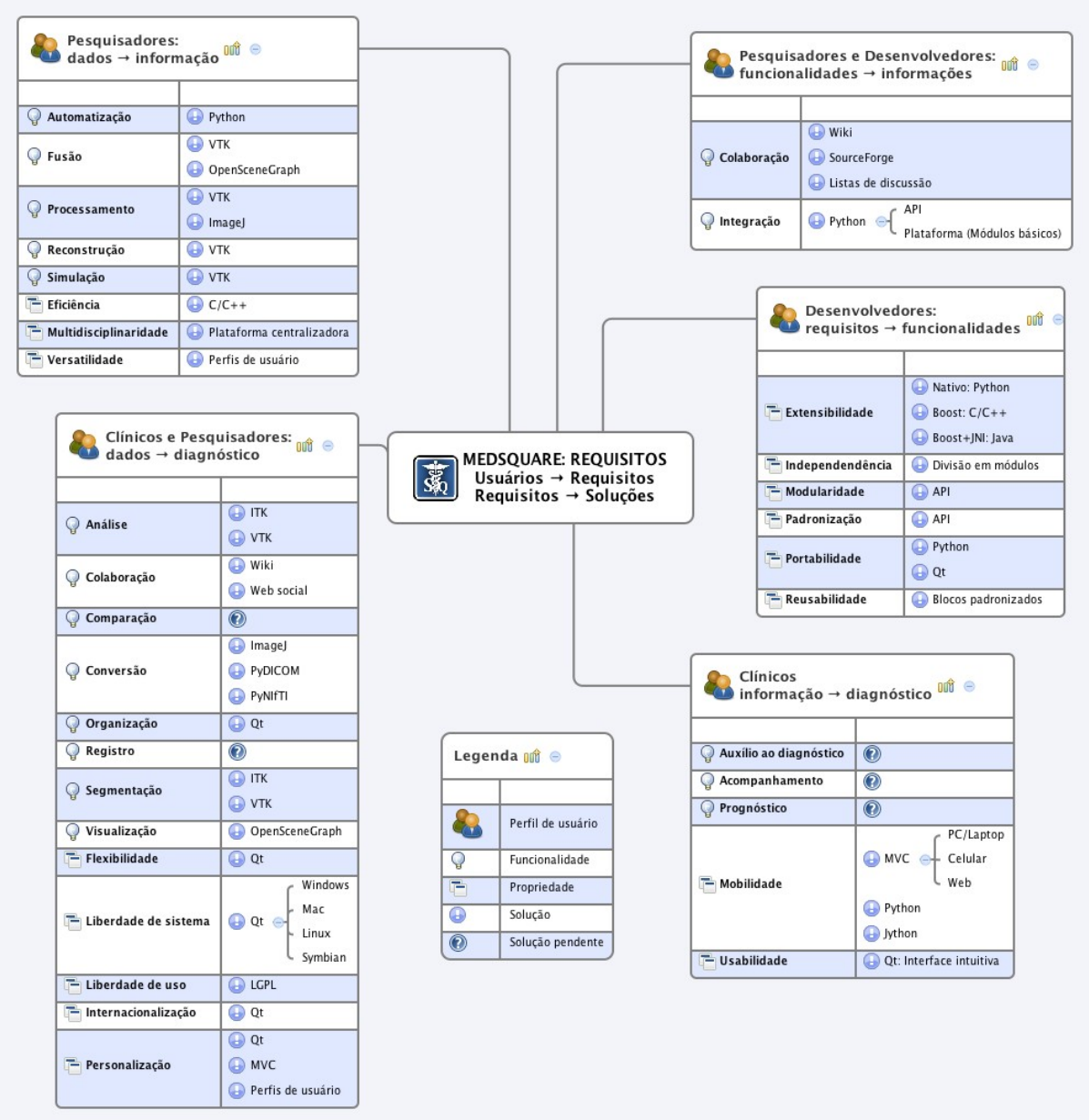

Ilustração 3.1: Esquema dos requisitos de funcionalidades e propriedades dentro da proposta atual, orientado a partir dos três perfis de usuário (Clínico, Pesquisador e Desenvolvedor)

Dentro desse contexto, foram encaixados os trabalhos em andamento, desenvolvidos dentro do grupo de pesquisa, como enumerados na tabela seguinte.

Atribuições pessoais, consideradas no contexto da plataforma	
Prof. Choukri Mekkaoui	Visualização e análise de imagens DT-MRI
Jihan Zoghbi	Segmentação de lesões cerebrais
Luiz Corte Real	Simulação de dinâmica molecular
Prof. Marcel Jackowski	Visualização e análise de imagens DT-MRI Supervisão do projeto
Maysa Macedo	Análise de morfologia vascular
Marcos Bonci	Interface entre módulos (API) Interface com o usuário Organização de volumes/imagens
Renato Borges	Simulação de imagens de difusão
William Mizuta	Registro de volumes/imagens

Ilustração 3.2: Atribuições pessoais.

3.3. Tecnologias e conceitos pesquisados

3.3.1. Gerenciamento de projetos de código aberto

Nessa nova fase, foi adicionado o objetivo de atrair e permanecer atraindo novos colaboradores, próximos e distantes geograficamente, para permitir a longevidade do projeto independente da composição de seu grupo de desenvolvedores. Para isso, foram observados conceitos de planejamento, gerenciamento e manutenção de comunidades de desenvolvimento em projetos *open-source*.

Seguindo esses conceitos, no intuito de adequar o projeto desde o início ao modelo de comunidade aberta, realizamos a definição da identidade do projeto (incluindo nome, missão e público-alvo), estabelecimento do modelo de colaboração, e a definição de uma estratégia de divulgação segundo a identidade e o modelo de colaboração estabelecidos.

3.3.2. Bibliotecas de computação gráfica tridimensional

Para a exibição de imagens tridimensionais geradas pelos módulos (todos eles inicialmente baseados VTK), foram pesquisadas outras bibliotecas de computação gráfica tridimensional que pudessem evitar seus pontos fracos e permitir o uso de outras alternativas em módulos futuros. Foram levados em conta, principalmente, a possibilidade de integração com VTK, a praticidade de uso e a maturidade da biblioteca.

- **Panda3D:**
  Nessa nova fase o projeto faria uso intensivo da linguagem Python, baseando a própria interface entre módulos nessa linguagem. Por isso, a primeira alternativa estudada foi a biblioteca Panda3D, desenvolvida pela Disney, em conjunto com uma comunidade voluntária, e planejada especificamente para uso em Python. Inicialmente, por sua proposta, parecia uma boa alternativa. Porém, devido à popularidade relativamente limitada, ao foco em entretenimento, e ao seu estado ainda relativamente pouco maduro, foi descartada.

- **Coin3D:**
  Clone *open-source* do Open Inventor (iniciativa pioneira de programação de alto nível sobre OpenGL, criada em 1989). Baseada no conceito de modelagem de cenas a partir de grafos de elementos, também parecia uma boa alternativa. Foi descartada (ou mais precisamente, deixada de lado) por não apresentar vantagens sobre a OpenSceneGraph.

- **OpenSG:**
  Assim como a Coind3D, também utiliza o conceito de grafos de elementos. Da mesma forma, também não parecia oferecer vantagens sobre a OpenSceneGraph.

- **OpenSceneGraph:**
  Com uma base de usuários aparente maior que a do próprio VTK, e com exemplos concretos de utilização em visualização científica, parecia a alternativa ideal. Porém, na época da pesquisa sobre esse assunto, seu suporte à linguagem Python encontrava-se com alguns problemas introduzidos na versão mais recente, por isso sua adoção no projeto ainda depende da verificação do estado desse suporte. Como o processamento 3D ainda não chegou a ser introduzido na nova implementação, esta escolha encontra-se em aberto.

3.3.3. Serviços de hospedagem de projetos

- **Google Code:**
  Utilizado por um breve período, antes da reformulação do projeto. Muito simples e prático, porém adequado apenas a projetos de pequenas proporções.

- **GitHub:**
  Serviço de popularidade crescente, baseado no sistema distribuído de versões Git. Freqüentemente utilizado por desenvolvedores em Python, e por isso uma alternativa óbvia para consideração. Porém, não parecia apresentar vantagens sobre o Launchpad, que utilizava um sistema de versões semelhante.

- **Launchpad:**
  Serviço mantido pela Canonical, onde se hospedam todos os projetos incorporados ao Ubuntu (sistema operacional criado pela mesma empresa). Apresentava várias características interessantes, como o acompanhamento de árvores de versões (semelhante ao GitHub, porém baseado no sistema Bazaar), o relacionamento de projetos e subprojetos, a ênfase no componente social do desenvolvimento e, por fim, facilidade para integração ao sistema Ubuntu. Foi escolhido inicialmente. No entanto, foi deixado de lado em favor do SourceForge, pela popularidade e pelos sistemas de versões utilizados neste.

- **SourceForge:**
  Apesar de nos últimos tempos ter diminuído sua participação em termos percentuais, é ainda sem nenhuma dúvida o serviço de hospedagem de projetos mais utilizado. É baseado no sistema de versões Subversion, também o mais popular em sua categoria, mas interage também com outros sistemas. Pela popularidade e, principalmente, por essa maior liberdade de escolha quanto ao sistema de versões, tornou-se o serviço escolhido.

3.3.4. Linguagem Python

Durante os dois semestre que se passaram, foi possível alcançar um aprofundamento no conhecimento da linguagem Python, que levou a uma percepção de que essa linguagem poderia não só ser utilizada no projeto, como também poderia torná-lo muito mais dinâmico através de seu uso em partes fundamentais. Atualmente, a linguagem Python é uma das principais escolhas técnicas no projeto, encontrada como solução para vários problemas, além de permitir características positivas adicionais não consideradas inicialmente.

Através do uso de suas características diferenciais (como a ausência da necessidade de compilção, a natureza dinâmica, a possibilidade de introspecção e a alta legibilidade de código por olhos humanos), torna-se muito mais simples e eficiente a prototipação de partes da plataforma e a transformação desses protótipos em módulos reais. Enquanto seu papel exato na implementação da interface entre módulos (API) permanece ainda indefinido, a exploração de seus recursos é uma prioridade na definição dessa interface.

3.4. Atividades realizadas

3.4.1. Definição da identidade do projeto

Ilustração 3.3: Logotipo do projeto.

Durante um período de aproximadamente dois meses, discutimos o formato do nome a ser dado para o projeto, as características e conceitos desejavelmente encapsulados nesse nome e, por fim, o nome em si. Foram levadas em consideração idéias de todos os participantes envolvidos, culminando em uma sessão de *brainstorming* em que, após uma rápida evolução de idéias partidas das sugestões coletadas, foi sugerido o nome MedSquare, instantaneamente aprovado por todos.

O nome reunia a idéia de "aplicação médica" (através do prefixo "Med") junto à idéia de "ponto de encontro" e, implicitamente, "colaboração" (através da palavra "Square"), em uma forma de fácil pronúncia, e também se encontrava disponível como domínio na internet. Assim, adotamos o nome, e no mesmo instante também já registramos o domínio medsquare.org, onde hoje se encontra nosso conteúdo na web.

Associado ao processo de escolha do nome, correu o processo de definição da missão do projeto, que hoje se encontra enunciada da seguinte forma: *"MedSquare é uma iniciativa ambiciosa, focada em prover meios para o desenvolvimento e a integração eficiente de ferramentas de software para a exploração de imagens médicas. Com essa iniciativa, esperamos não apenas criar um poderoso ambiente de software capaz de alcançar os altos padrões da comunidade médica, mas também promover o intercâmbio ativo de conhecimento em uma rede colaborativa de clínicos, pesquisadores e desenvolvedores por todo o mundo."*

3.4.2. Criação do website

Partindo de pouca experiência na criação de websites, e nenhuma experiência no gerenciamento de websites dinâmicos, o desenvolvimento da face do projeto na web foi também um aprendizado. Inicialmente, foi escolhido um software para gerenciamento dinâmico e colaborativo (MoinMoin, gerenciador de *wiki*) e um serviço de hospedagem web, através do qual seria utilizado esse software e através do qual teríamos também a oportunidade de hospedar o domínio relativo ao nome escolhido para o projeto.

Após essas escolhas feitas, foi instalado e configurado o MoinMoin, e foi escolhido e personalizado um modelo de *layout* (fontes, cores e distribuição na tela). Sobre esse modelo, foram definidas as linhas gerais das seções do site e do conteúdo de cada uma delas.

Mais tarde, na intenção de tornar a apresentação inicial do projeto mais direta, decidimos pela criação de um novo site, que incorporaria o anterior como uma de suas seções, e que adicionaria basicamente um pequeno conjunto de páginas criadas individualmente, com ênfase na característica gráfica da sua apresentação. Os resultados são apresentados nas duas ilustrações seguintes.

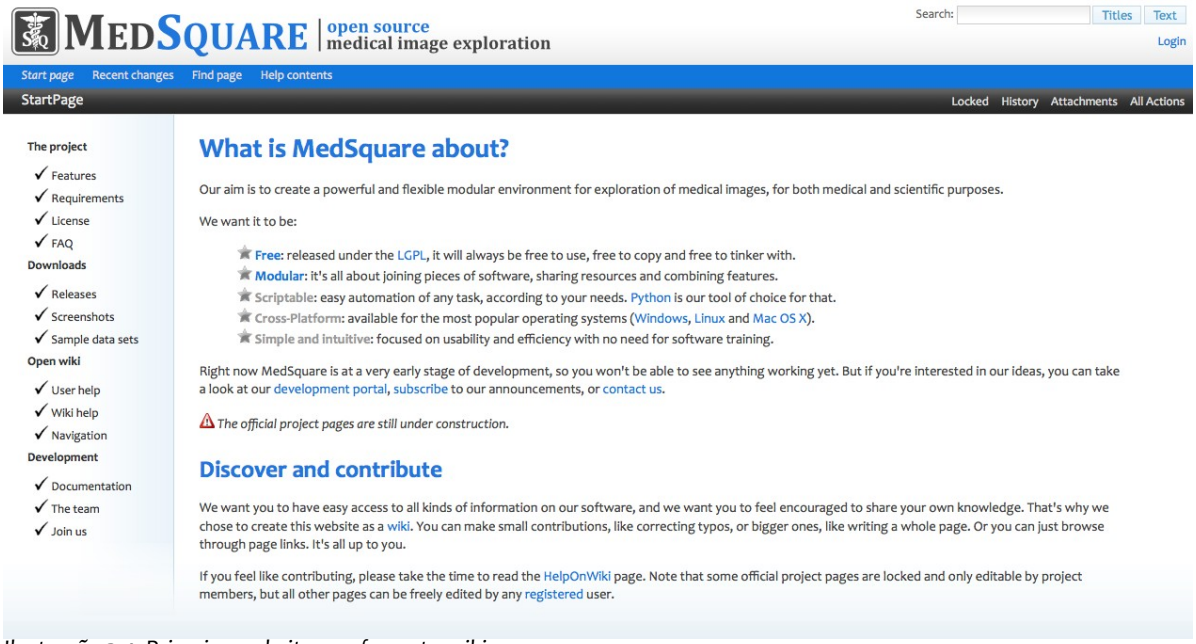

Ilustração 3.4: Primeiro website, em formato wiki.

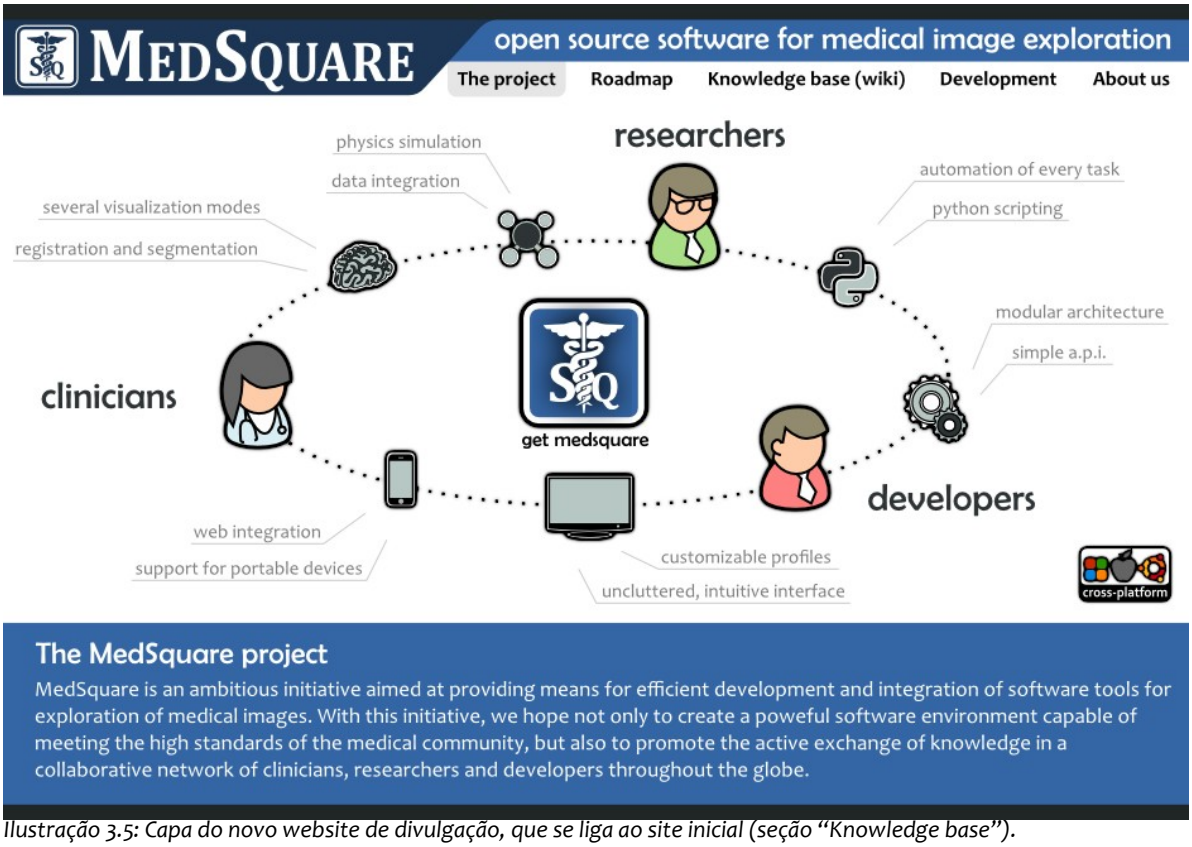

Ilustração 3.5: Capa do novo website de divulgação, que se liga ao site inicial (seção "Knowledge base").

3.4.3. Definição da licença e do modelo de colaboração

Seguindo na definição da estratégia de divulgação e colaboração, partimos para a decisão do modelo de licenciamento de software a ser adotado. Inicialmente, na primeira fase projeto, a licença GPL havia sido adotada, por seu caráter simples e universal. No entanto, no novo contexto havia novos aspectos a serem considerados, sendo a modularidade e a independência de desenvolvimento os principais.

Para garantir a maior liberdade possível para introdução de novos módulos e funcionalidades, através do desenvolvimento independente, mas sem comprometer a natureza aberta de toda a parte central, optamos por adotar uma licença que reunisse as seguintes qualidades: reconhecimento pela entidade competente (Open Source Initiative), exigências de *copyleft* (proibição de uso de licenças fechadas para trabalhos derivados), compatibilidade com a GPL (licença mais popular), possibilidade de *linkagem* com software disponível sob outras licenças e, por fim, popularidade. Seguindo a linha de comparação ilustrada a seguir, decidimos pela adoção da licença LGPL 2.1 (que inclui indiretamente sua versão mais recente, a LGPL 3.0).

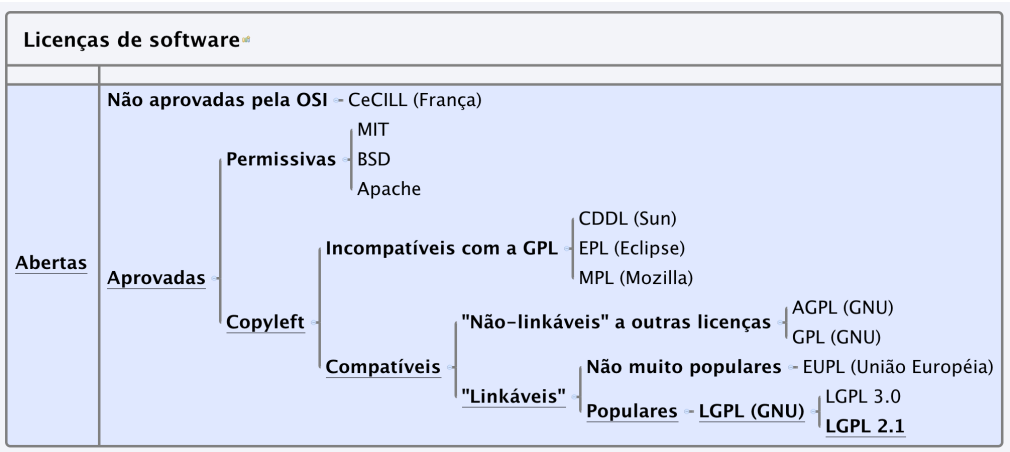

Ilustração 3.6: Classificação das licenças pesquisadas. Os itens sublinhados representam o caminho até a LGPL 2.1, escolhida para a biblioteca central e os módulos básicos do projeto.

Através do uso dessa licença, o software central à plataforma manterá sua característica *open-source*, ao mesmo tempo em que permitirá a integração com software produzido sob qualquer tipo de licença. Dentro desse modelo, torna-se possível o agrupamento de módulos em diferentes distribuições da plataforma, cada uma gerenciada por um desenvolvedor ou uma equipe independente, e possivelmente com características técnicas e legais diferenciadas. Um aspecto importante para a garantia dessa possibilidade é que toda comunicação entre módulos será feita através da API da plataforma, e dessa forma nenhum módulo tem acesso direto a outro módulo.

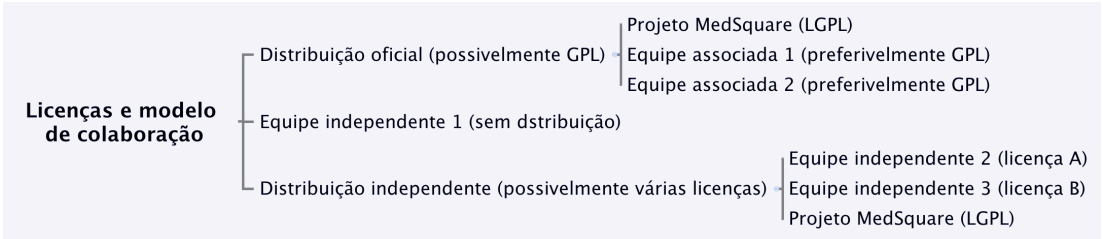

Ilustração 3.7: Exemplo hipotético seguindo o modelo de colaboração adotado.

3.4.4. Implementação do software

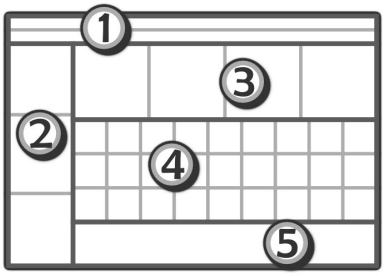

Ilustração 3.8: Elementos da interface em desenvolvimento atualmente

A implementação da tela inicial em Qt e a implementação experimental da API estão atualmente seguindo em paralelo. Com a conclusão do site e a definição detalhada da API, esperamos alcançar a curto prazo uma versão (instável) em funcionamento, com as funcionalidades básicas representadas pelos seguintes elementos da interface:

- **(1) Barra de menu e de ferramentas:**
  As barras de ferramentas serão opcionais, personalizáveis e reposicionáveis. Usando o modelo do Qt de abstração de comandos do usuário em *"actions"*, os elementos de menu e os botões na barra de ferramentas são apenas os mesmos objetos em manifestações distintas.

- **(2) Área para ferramentas dos módulos:**
  Também opcional, composta por uma ou mais sub-janelas reposicionáveis e automaticamente ajustáveis à janela principal. Por convenção, cada sub-janela representará um módulo (de segmentação, simulação, etc), mas a plataforma não restringirá a área de atuação de cada módulo na interface, permitindo, através da API central, a adição de elementos visíveis ou invisíveis a qualquer das áreas da janela.

- **(3) Área de visualização volumétrica:**
  Composta, por padrão, pelos quatro quadros vistos em vários programas: corte axial, corte sagital, corte coronal, e por fim o quadro de visualização tridimensional. Este último poderá exibir vários tipos de imagens, como reconstruções, montagens dos cortes no espaço, ou combinações de vários modos. Cada quadro dessa área poderá ser reposicionado ou excluído da tela, e a organização poderá ser alterada (por exemplo, 4x1 ou 2x2).

- **(4) Área de visualização seqüencial:**
  Exibição simples dos cortes originais do volume, em disposição e tamanho definidos pelo usuário.

- **(5) Linha de comando de Python:**
  Sub-janela contendo uma linha de comando de Python integrada, com acesso ao ambiente de execução do programa. Poderá ser usada em fases de teste (para alterações em tempo real ou exibição de mensagens de erro), ou para execução de tarefas a partir de instruções em texto.

Cada uma dessas áreas poderá ser modificada (redimensionada, reposicionada, etc) por ações de fácil alcance, porém sem sobrecarga de informação visual, preferivelmente através de ações do mouse.

Após alcançar essa versão, com a interface descrita já pleno em funcionamento, o passo seguinte será a integração com módulos desenvolvidos anteriormente como parte de outros projetos.

3.5. Resultados obtidos

3.5.1. Resultados práticos

Seguindo uma linha de desenvolvimento oposta à inicial (na qual não havia planejamento previsto e na qual os resultados finais em curto prazo eram o objetivo), pudemos chegar a um estado no qual, apesar do resultado final ainda não se manifestar de forma concreta, uma base sólida foi estabelecida para que esse resultado seja alcançado.

A partir dessa base, manifestada na documentação apresentada neste texto, a implementação do resultado final desejado não depende mais de uma composição específica da equipe ou de uma distribuição específica de tarefas, e assim eventuais falhas futuras no processo de desenvolvimento podem ser isoladas localmente, evitando que o trabalho desenvolvido em outros pontos seja perdido.

Outro resultado prático alcançado foi o modelo de divulgação bem definido, tornado em forma concreta no website, que em poucos dias estará totalmente concluído e em funcionamento. A partir da inclusão do projeto em listas de interesse e, principalmente, através da comunicação pessoal, esperamos atrair colaboradores e futuros usuários, que terão a oportunidade de se informar em nosso site. Um fato interessante e relacionado foi a manifestação de interesse em colaboração por parte do desenvolvedor do software MedINRIA (citado na primeira parte do texto), que demonstra aprovação da proposta e de sua validade no contexto de pacotes de software de exploração de imagens médicas.

3.5.2. Resultados indiretos

Graças aos resultados citados logo acima, agora temos a possibilidade concreta de inclusão de novos colaboradores, independente da forma como eles cheguem ao projeto. No curto prazo, eventuais novos colaboradores poderão auxiliar no processo de definição da API, compartilhando suas necessidades como desenvolvedores ou como pesquisadores. No médio prazo, após concluída a API, esses colaboradores poderão iniciar a criação ou a integração de novas ferramentas na plataforma, e auxiliar na expansão e manutenção da hierarquia de tomada de decisões.

Apesar de não ter caminhado exatamente segundo a proposta apresentada para este trabalho (sob a disciplina MAC0499), o projeto tem evoluído de forma sólida e satisfatória, e por isso pode ser considerado um sucesso no escopo de tempo previsto, tendo efetivamente aproveitado o conhecimento adquirido a partir dos erros passados e preparado o terreno de forma consistente.

3.6. Dificuldades encontradas

3.6.1. Sobrecarga de informações

Em momentos de pesquisa e busca por decisões que alcançassem aprovação geral do grupo, houve algumas vezes uma certa sobrecarga de informações, por conta da concorrência entre diferentes pontos de vista e também da divisão da atenção entre diferentes tipos de tarefas. Mesmo assim, esse tipo de congestionamento mental foi passageiro e provavelmente apenas um efeito natural e inevitável nesse estágio no planejamento de um projeto de grande porte.

3.6.2. Conceito recente

O conceito atual, apesar de ter surgido como conseqüência de um processo gradual, foi adotado de forma relativamente repentina, o que em vários momentos causou confusão entre os participantes, pois nem mesmo aqueles que o propuseram tinham uma idéia completa formada. Provavelmente mais um efeito natural e inevitável.

3.7. Conclusões alcançadas

3.7.1. Dinâmica de trabalho em equipe

Incluindo todo o grupo de pesquisas sob a mesma denominação (o projeto MedSquare), a dinâmica de certa forma se alterou de uma interação apenas "em grupo" para uma interação "em equipe". Enquanto o objetivo final se mantém em grande parte o mesmo, a participação de cada um se estabelece de forma mais clara, facilitando a obtenção de boas idéias e resultados a partir da comunicação entre os membros.

Mesmo estando geograficamente próximos, os membros do grupo têm rotinas diferentes e acabam tendo relativamente pouca interação direta. Além disso, cada um tem seu próprio perfil e se concentra em determinada área de atuação. Estabelecendo de forma mais explícita a relação de cada um com o projeto, oferecendo meios técnicos para que não ocorra uma sobreposição desnecessária de preocupações e tarefas, e tendo um modelo bem definido para a integração de esforços, permite-se uma maior eficiência. Da mesma forma, esses pontos se aplicam também a equipes distribuídas geograficamente.

3.7.2. Escopo do projeto e expectativas de prazo

É essencial reconhecer a natureza genérica ou específica de um projeto logo cedo no seu planejamento. Conceitos específicos exigem menos tempo, menos pessoas envolvidas e menos preocupação com aspectos imprevistos. Conceitos genéricos exigem mais pessoas envolvidas e, independentemente do tamanho da equipe, exigem tempo para a consideração de aspectos inicialmente imprevistos. Além disso, no processo de inclusão de participantes, é necessário considerar as necessidades específicas de cada grupo para uma interação bem-sucedida.

3.8. Previsões e expectativas

3.8.1. Expansão do grupo de colaboradores

Para o médio e longo prazo, prevemos alcançar a colaboração com grandes projetos atuais (como MedINRIA, Slicer e OsiriX) na questão de interoperabilidade. Para o curto e médio prazo, esperamos delegar tarefas a novos colaboradores, apresentados ao projeto através de contatos na Universidade de São Paulo, no Hospital das Clínicas, na Universidade de Yale e na Universidade de Harvard. Esperamos também que a primeira versão "completa" do programa possa atrair mais participantes. (Não haverá um estado em que a plataforma poderá ser considerada realmente completa, já que ela dará sempre margem para expansões e viradas de rumo.)

3.8.2. Evolução do software

Esperamos alcançar uma versão *"alpha"* dentro dos próximos dois meses, e uma versão *"beta"* antes do próximo semestre. Desde o início, pretendemos manter pacotes de instalação padronizados e automatizados para os sistemas operacionais mais populares. Em médio prazo, possivelmente serão incluídos métodos inovadores de interação com o usuário, como descritos na primeira parte deste texto.

Para o longo prazo, temos idéias relativas à inclusão de serviços específicos para web, como processamento de dados a partir de imagens em modelo "*cloud computing*" e a criação de um website de interação social entre usuários da plataforma, possivelmente com ferramentas de colaboração entre profissionais (utilizando perfis pessoais introduzidos no site) incluídas no próprio software.

4. Referências

4.1. Referências bibliográficas

- Kitware Inc., 2006
  VTK User's Guide Version 5

- Will Schroeder/Ken Martin/Bill Lorensen, 2006
  Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition

- Karl Fogel, 2005
  Producing Open Source Software: How to Run a Successful Free Software Project

- Kent Beck/Cynthia Andres, 2004
  Extreme Programming Explained: Embrace Change (2nd Edition)

- Erich Gamma/Richard Helm/Ralph Johnson/John M. Vlissides, 1994
  Design Patterns: Elements of Reusable Object-Oriented Software

4.2. Referências na Web

4.2.1. Projeto MedSquare

http://www.medsquare.org/

http://www.medsquare.org/wiki/

http://sourceforge.net/projects/medsquare/

https://launchpad.net/medsquare/

4.2.2. Programas analisados

http://www.bioimagesuite.org/

http://www.idoimaging.com/

http://aeskulap.nongnu.org/

http://amide.sourceforge.net/

http://www.fmrib.ox.ac.uk/fsl/

http://www.itksnap.org/

http://ric.uthscsa.edu/mango/

http://www-sop.inria.fr/asclepios/software/MedINRIA/

http://www.slicer.org/

http://www.unipacs.com/

4.2.3. Documentação de tecnologias de software

http://www.vtk.org/

http://docs.python.org/

4.2.4. Wii Remote

http://johnnylee.net/

5. Experiência pessoal

5.1. Desafios e frustrações

Acredito que foram dois os principais desafios que por várias vezes geraram frustrações: a dificuldade no entendimento do escopo do trabalho pessoal, e a perseguição do alvo móvel que é o desenvolvimento de um novo software. A questão do entendimento do escopo certamente foi a que mais gerou frustrações.

Vejo a primeira questão (presente na "primeira fase" do projeto) como um problema que poderia ser evitado por conhecimento prévio de contexto semelhante (que foi o que ocorreu na "segunda fase"). Já a segunda questão, do caráter "furtivo" do planejamento e desenvolvimento de software, eu vejo como algo inevitável nesse tipo de atividade.

5.2. Disciplinas mais relevantes e relação com o trabalho

- **MAC0332: Engenharia de Software e MAC0342: Laboratório de Programação Extrema:**
  Sem dúvida essas foram as duas disciplinas que mais tiveram influência direta no trabalho. Enquanto Engenharia de Software mostrou abordagens tradicionais no gerenciamento do desenvolvimento, Programação Extrema mostrou uma abordagem oposta, menos formal e mais dinâmica. Além disso, Programação Extrema me serviu também como uma ótima experiência prática no trabalho em equipe. Através da reflexão provocada pelo contraste entre as duas abordagens, aplicada ao projeto, foi possível chegar a conclusões que alteraram completamente seu rumo.

- **MAC0420: Introdução à Computação Gráfica e MAC0417: Visão e Processamento de Imagens:**
  Assim como as duas disciplinas citadas acima, essas duas também apresentaram um contraste. Porém, nesse caso o contraste ocorria entre abordagens complementares (e não concorrentes) no tratamento de dados, fluindo em sentidos opostos. Além de abrirem os olhos para os tipos de processos envolvidos, essas disciplinas serviram também para despertar o interesse nas duas áreas e me levar a procurar o que veio a se tornar este trabalho. Ainda como ganho adicional, fui apresentado à linguagem Python na disciplina de Visão.

- **MAC0422: Sistemas Operacionais e MAC0412: Organização de Computadores:**
  Apesar de não terem gerado um impacto diretamente identificável no desenvolvimento do projeto, essas duas disciplinas foram muito úteis para ampliar a percepção de sistemas computacionais em sua totalidade, ajudando a enxergar os dois lados da moeda (o alto e baixo nível de abstração) em constante interação, e a isolá-los melhor quando necessário em um momento de análise.

- **MAC0211: Laboratório de Programação I:**
  Por se tratar de uma primeira experiência relativamente realista de trabalho prático, a disciplina de Laboratório de Programação teve um enorme impacto no meu aprendizado geral na área de computação. Mais que isso, foi importantíssima para manter minha motivação direcionada para a área da computação em uma época em que disciplinas pouco ou nada relacionadas com meus interesses profissionais minavam minha motivação geral.

- **MAC0441: Programação Orientada a Objetos:**
  Essa disciplina foi de grande utilidade em aumentar a consciência sobre a identificação de padrões naturais e sobre o quanto essa naturalidade pode ser subjetiva. Foi muito útil também em mostrar o quanto é importante, em certos momentos, não só desenvolver ou identificar padrões, mas também dar nome a eles para permitir uma comunicação eficiente.

- **MAC0339: Informação, Comunicação e a Sociedade do Conhecimento:**
  Apesar de se distanciar bastante do perfil típico das disciplinas do curso (ou talvez por esse exato motivo), essa disciplina foi muito interessante ao expor o poder e a complexidade do trabalho distribuído e voluntário, como observado atualmente no contexto da Internet.

5.3. Aprendizado e continuidade

O conhecimento prático adquirido pela participação neste projeto e na condução de atividades relativas a ele foi certamente de dimensões comparáveis à soma de todo o conhecimento adquirido nas disciplinas do curso. Considero que foi uma atividade de fundamental importância para minha formação profissional.

Enquanto o curso em geral enfatiza o conhecimento teórico, geralmente incluindo atividades de valor apenas didático, essa experiência deu a oportunidade de vivenciar um contexto realista, da perseguição de resultados diretamente úteis à sociedade, no qual a iniciativa pessoal é um fator essencial. Outro aprendizado importante foi a lição de reconhecer o fracasso o mais cedo possível e, nesse fracasso, enxergar a verdadeira oportunidade de sucesso.

Minha expectativa é de utilizar diretamente na minha vida profissional a experiência adquirida, possivelmente seguindo no próprio projeto ainda por vários anos. De qualquer forma, tenho certeza de que essa experiência me será muito útil no futuro próximo e distante.

6. Agradecimentos

Ao professor Marcel,
por sua presença constante, seu otimismo e sua busca pela excelência.

À Jihan, ao Luiz, à Maysa e ao Renato,
por embarcar nessa iniciativa por algo ainda maior.

Ao professor Choukri,
por sua fé à distância.

Ao professor Alfredo,
por apontar o caminho de volta à estrada.

Aos meus pais, Diva e Pádua,
pelo apoio de sempre.

Aos meus irmãos, Eduardo e Lúcia,
por agüentar minha prosa delongada.