UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

DANIEL COSMO PIZETTA

Biblioteca, API e IDE para o desenvolvimento de projetos de
metodologias de Ressonância Magnética

São Carlos
2014

DANIEL COSMO PIZETTA

Biblioteca, API e IDE para o desenvolvimento de projetos de
metodologias de Ressonância Magnética

Dissertação apresentada ao Programa de
Pós-Graduação em Física do Instituto de
Física de São Carlos da Universidade de
São Paulo, para obtenção do título de
Mestre em Ciências.

Área de concentração: Física Aplicada
Opção: Física Computacional
Orientador: Prof. Dr. Alberto Tannús

Versão Corrigida

(versão original disponível na Unidade que aloja o Programa)

São Carlos

2014

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pelo Serviço de Biblioteca e Informação do IFSC,
com os dados fornecidos pelo(a) autor(a)

Pizetta, Daniel Cosmo

Biblioteca, API e IDE para o desenvolvimento de projetos de metodologias de Ressonância Magnética / Daniel Cosmo Pizetta; orientador Alberto Tannús - versão corrigida -- São Carlos, 2014.

93 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Física Aplicada Computacional) -- Instituto de Física de São Carlos, Universidade de São Paulo, 2014.

1. Ambiente integrado de desenvolvimento. 2. Ressonância Magnética. 3. Desenvolvimento de software. 4. Sequência de pulsos. I. Tannús, Alberto, orient. II. Título.

À minha mãe, Jandira Stênico Pizetta, por minha educação e por ter feito
com que eu chegasse até aqui.

À Geisiane Rosa da Silva, por estar ao meu lado em todos os momentos.

Aos professores do colégio João Geraldo Vieira e Gilsandro Cesar
Prudente que, além de incentivarem meus estudos, dedicaram finais de
semana e outras datas para aulas particulares, nos levar às olimpíadas de
física e matemática ou nos ajudar em algum projeto.

À todos os meus amigos, que fizeram parte deste caminho, que me
ajudaram e me deram forças para continuar.

AGRADECIMENTOS

Ao Prof. Dr. Alberto Tannús, pela paciência e liberdade com que me orientou, pelo desenvolvimento das sequências no novo sistema e seus incansáveis testes.

Ao Dr. Édson Luiz Géa Vidoto, pelo tempo e conhecimento disponíveis.

Ao Dr. Mateus José Martins, pelo tempo e conhecimento disponíveis e pelo ACQ-Server.

Ao Prof. Dr. Fernando Fernandes Paiva pela paciência nas reuniões de 4 horas.

À todos os amigos do CIERMag e ‘agregados’ pela amizade, ajuda, ensinamentos e piadas, em especial ao Gustavo V. Lourenço pelo desenvolvimento da interface de edição gráfica de sequências do ToRM-IDE, ao Danilo M. D. D. da Silva pelo desenvolvimento do ToRM-Console, ao Felipe B. Coelho e Guilherme M. Freire pelo desenvolvimento do ToRM-Compiler e da linguagem F, ao Erick Fonseca pelo *plug-in* para linguagem F do Eclipse e contribuições para o editor gráfico de sequências.

Ao Daniel M. Consalter, através da FIT – *Fine Instrument Technology* pela parceria através da Rede EMOH (Rede de Equipamentos Médicos Odontológicos e Hospitalares) do SIBRATEC e por ceder algumas das imagens deste trabalho.

À todos que contribuíram com a pesquisa de opinião sobre os equipamentos de ressonância magnética e que assim fizeram parte deste trabalho.

Aos amigos da Fiscomp07 e agregados, que fizeram e fazem a academia um lugar mais divertido e que mesmo nas piores horas as piadas sejam lembradas para fazer renascer a vontade de lutar e seguir em frente.

À todos que me ajudaram a organizar a CPWG (Ciclo de Palestras e Workshop de Graduação), que depois se tornou a SEMAFIS (Semana da Física), que se uniu a Pós-Graduação e segue como SIFSC (Semana Integrada do Instituto de Física de São Carlos). Agradeço a sabedoria compartilhada, o crescimento pessoal, amizades e contatos que surgiram.

À todos os demais que não se enquadram nas referências acima, mas que fizeram parte e que contribuíram com este trabalho, de forma direta ou indireta, o meu muito obrigado.

Ao Instituto de Física de São Carlos, pela oportunidade de realização do curso de mestrado.

À CAPES com o programa PROEX, pela concessão da bolsa de mestrado para a realização desta pesquisa.

As agências de fomento FAPESP:2005/5.6663-1 e CNPQ:565.047/2010-8, FINEP, FNS pela realização deste projeto.

"They did not know it was impossible, so they did it"

Mark Twain

RESUMO

PIZETTA, D.C. **Biblioteca, API e IDE para o desenvolvimento de projetos de metodologias de Ressonância Magnética**. 2014. 93 p. Dissertação (Mestrado em Ciências) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2014.

Neste trabalho serão discutidas novas ferramentas para a construção de um espectrômetro de Ressonância Magnética (RM) totalmente digital. A motivação parte das dificuldades encontradas pelos pesquisadores no momento de programar um equipamento de RM, incluindo a falta de ferramentas para desenvolvimento de metodologias, as quais não são oferecidas pelos softwares atuais. Em particular tratamos do desenvolvimento de uma biblioteca, a PyMR (*Python Magnetic Resonance*), de uma API (*Application Program Interface*) e de um IDE (*Integrated Development Environment*). Nesta estrutura, a biblioteca PyMR é o *front-end* para programação e *setup* dos equipamentos de RM enquanto a API constitui o *back-end*. O IDE, por sua vez, é uma ferramenta de auxílio especializado para criação e gerenciamento das metodologias e protocolos de RM de forma funcional e amigável. O desenvolvimento baseado no estado-da-arte das tecnologias de Computação e Ressonância Magnética garante a qualidade, robustez, adaptabilidade e ainda assim, a simplicidade para uso dos menos experientes. Para a validação do sistema, além de métricas de software, foi montada uma sequência de pulsos conhecida como CPMG (*Carr-Purcell-Meiboom-Gill*) executada no espectrômetro local sobre uma amostra de $CuSO_4$ em solução, o qual mostrou valores de T_2 compatíveis com os valores esperados. Os resultados do novo sistema mostram sua capacidade de atender as principais exigências dos usuários e desenvolvedores de metodologias de RM, oferecendo um amplo conjunto de ferramentas. Em suma, este projeto provê a estrutura básica e funcional de uma nova forma de se programar e utilizar equipamentos de RM, gerando um poderoso instrumento para a pesquisa na área.

Palavras-chave: Interface de programação de aplicação. Ambiente integrado de desenvolvimento. Ressonância Magnética. Desenvolvimento de software. Sequência de pulsos.

ABSTRACT

PIZETTA, D.C. **Library, API and IDE for the development of Magnetic Resonance methodologies**. 2014. 93 p. Dissertação (Mestrado em Ciências) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2014.

In this study we discuss new tools for the building of a fully digital Magnetic Resonance (MR) spectrometer. The research was motivated by several difficulties experienced by researchers in programming MR machines, which include the lack of tools for the development of methodologies that are not currently offered by companies. In particular, we treat the development of a library, PyMR (Python Magnetic Resonance), an API (Application Program Interface) and an IDE (Integrated Development Environment). In this structure, the PyMR library acts as a front-end for MR equipment programming and setup while the API is a back-end. Finally, the IDE is a user-friendly tool that helps the developer to create and manage methodologies and protocols. The state-of-the-art of Computer Sciences and Magnetic Resonance technologies adopted here has ensured the quality, robustness and adaptability keeping simplicity for non-experienced users. For the validation of the system, besides software metrics, a pulse sequence known as CPMG (Carr-Purcell-Meiboom-Gill) was assembled and performed on an onsite spectrometer, using a solution of $CuSO_4$ as a sample, which exhibited compatible T_2 values. The results show that the system can meet the main requirements of both users and developers and offer a large set of tools. This project provides a basic and functional structure of a new way to program and use the MR equipment and a powerful tool for researchers in this area.

Keywords: Application programming interface. Integrated development environment. Magnetic Resonance. Software development. Pulse sequence.

LISTA DE FIGURAS

Figura 1 - Exemplo de uma sequência de pulsos, <i>Spin Echo</i> , onde cada linha representa o dispositivo e as formas de onda que serão geradas através deles numa sequência temporal.	30
Figura 2 - Dispositivos e periféricos dos sistemas de gradientes e de transmissores.....	32
Figura 3 - Componentes da cadeia do dispositivo de gradientes.....	32
Figura 4 – Componentes da cadeia do dispositivo de transmissão.	33
Figura 5 - Componentes da cadeia do dispositivo de recepção.	33
Figura 6 - Periféricos comuns para um sistema básico de ressonância magnética.....	34
Figura 7 – a) Bobina de RF para imagem de punho, volumétrica. b) Bobina de RF de superfície.....	35
Figura 8 - Sistema da FIT. a) Magneto aberto de 0,3T Centauro e conjunto de bobinas, ao fundo, para MRI. b) Rack de amplificadores do sistema de MRI. c) Magneto permanente para espectroscopia e uso em bancada ao lado do espectrômetro e computador.	36
Figura 9 – Etapas do processo de desenvolvimento de software <i>Scrum</i>	38
Figura 10 - Relação entre bibliotecas, API's e o software especializado e alguns exemplos.	42
Figura 11 - Esquema básico de um IDE, com navegador de arquivos, editor de código, <i>debugger</i> entre outros.	44
Figura 12 - Interface de parâmetros da sequência do software ODIN.	47
Figura 13 - Visualização da interface gráfica do DAMARIS.....	48
Figura 14 - Linha do tempo da metodologia de RM dentro da nova visão de software: Desenvolvimento – onde são geradas e gerenciadas as sequências, testes e validações; Execução – Geração dos protocolos, refinamento e validação experimental; Processamento – Processamento dos dados e refinamento dos algoritmos de processamento.....	49
Figura 15 - Estrutura e relação entre a biblioteca PyMR, a ToRM-API e o ToRM-IDE juntamente com a metodologia gerada através delas. A PyMR representa um <i>front-end</i> para programação generalizada enquanto que a ToRM-API fornece o <i>back-end</i> para um equipamento específico. O ToRM-IDE é o ambiente para geração de metodologias e protocolos, que auxilia o usuário nesta tarefa.	50
Figura 16 - Projeto ToRM completo relacionado com os itens deste trabalho: PyMR, ToRM-API e ToRM-IDE na etapa de desenvolvimento da linha do tempo da metodologia. A sequência em linguagem F, até o momento, que é	

parte da metodologia é compilada pelo ToRM-Compiler para envio para a FPGA. Estes itens foram desenvolvidos por Felipe B. Coelho e Guilherme M. Freire. O ToRM-Console faz o controle do uso do espectrômetro, providenciando a execução das metodologias e ToRM-Processing realiza o processamento dos dados - desenvolvidos por Danilo M.D.D. da Silva. Por fim, o ToRM-Acq-Server que permite a comunicação com a FPGA foi desenvolvido por Mateus J. Martins..	51
Figura 17 – Número de pessoas que utilizam as diferentes ferramentas: a) Sistemas operacionais. b) Editores de texto. c) IDE's.	52
Figura 18 – Grau de semelhança no uso, praticidade e programação dos softwares de equipamentos de RM.	53
Figura 19 - Como o usuário lida com os diferentes tipos de erros na programação de sequências de pulsos quando comparadas a programação comum.	53
Figura 20 - De que forma o usuário aprende a utilizar e/ou programar o equipamento.	54
Figura 21 - Quais as principais razões para não programar uma sequência segundo os usuários.	54
Figura 22 - a) Linguagens de programação mais conhecidas entre os usuários de RM entrevistados. b) Conhecimento dos entrevistados sobre POO.	55
Figura 23 – a) Quem realiza modificações nas sequências quando são necessárias. b) Qual o grau de dificuldade empregado para realizar a modificação do item a.	55
Figura 24 - a) A familiaridade dos usuários entrevistados com os equipamentos disponíveis no mercado. b) Qualidade da edição das sequências de forma textual em cada equipamento.	56
Figura 25 - a) Formação acadêmica dos entrevistados. b) Faixa de idade dos entrevistados.	56
Figura 26 - Língua do país de origem das publicações com o tópico ' <i>magnetic resonance</i> ' em todas as bases da <i>Web of Knowledge</i> no período de 1940 a 2013.	60
Figura 27 - Equipamento de testes do projeto ToRM-15. a) Conjunto magnético e amplificadores (RF e gradientes). b) Conjunto magnético evidenciando as partes que o compõem. c) Núcleo do espectrômetro: os dispositivos em FPGA para geração de pulsos e recepção do sinal.	61
Figura 28 - Diagrama dos pacotes e módulos do <i>core</i> da PyMR.	62
Figura 29 - Estrutura geral da relação entre metodologia, dispositivos e periféricos que formam a base da PyMR, exemplificada com o subsistema de gradientes. As três camadas garantem simplicidade na descrição, criação e manutenção das funcionalidades do sistema.	63

Figura 30 - Exemplo de parâmetros visualizados na aba de controle..	65
Figura 31 - Grupos de usuários e desenvolvedor. Escala de maior para a menor permissão no sistema, de cima para baixo.	67
Figura 32 - Diagrama de classes do módulo <i>nucleus</i> da PyMR, onde algumas das classes foram omitidas por falta de espaço.	68
Figura 33 - Hierarquia de classes referentes às bobina, <i>Coil</i> e <i>CoilSet</i> , e classes auxiliares que são utilizadas como enumeradores como <i>GradientCoilType</i> , <i>OperationMode</i> , <i>PolarizationMode</i> e <i>RFProbeType</i> .	68
Figura 34 - Diagrama de classes para a classe de magneto permanente (<i>PermanentMagnet</i>).	68
Figura 35 – Estrutura de um projeto de metodologia no ToRM-IDE.	72
Figura 36 - ToRM-IDE em detalhes. Editor de parâmetros em foco com a identificação das principais funcionalidades.	73
Figura 37 – Editor de parâmetros do ToRM-IDE da metodologia CPMG.	74
Figura 38 - Editor de parâmetros do ToRM-IDE com os parâmetros do sistema aberto na aba de configuração dos gradientes.	74
Figura 39 - Editor gráfico de sequências do ToRM-IDE com exemplo de uma sequência Spin Echo Multi Slices.	76
Figura 40 - ToRM-Console executando a metodologia CPMG com TE=5ms, TR=4s, 256 ecos e 8 médias na amostra de <i>CuSO₄</i> em solução na concentração de 1g/l.	77
Figura 41 - ToRM-Processing executando a interpolação dos máximos dos ecos adquiridos para retornar a informação do valor de <i>T₂</i> =276ms para a amostra de <i>CuSO₄</i> em solução de 1g/l.	78

LISTA DE TABELAS

Tabela 1 - Estatísticas do código da biblioteca PyMR.	69
Tabela 2 - Estatística do código do ToRM-API.....	71
Tabela 3 - Estatística do código do ToRM-IDE.....	76

LISTA DE ABREVIATURAS

ADC	Analog to Digital Converter
AM	Amplitude Modulation
API	Application Programming Interface
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CD	Compact Disc
CIC	Cascade Integrator Comb
CIERMag	Centro de Imagens e Espectroscopia in vivo por Ressonância Magnética
CInAPCe	Cooperação Interinstitucional de Apoio à Pesquisa sobre Cérebro
CLI	Command Line Interface
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CPMG	Carr-Purcell-Meiboom-Gill
CRT	Cathod Ray Tube
DAC	Digital to Analog Converter
DAMARIS	Darmstadt Magnetic Resonance Instrument Software
DNA	Deoxyribonucleic acid
DNP	Dynamic Nuclear Polarisation
DVD	Digital Versatile Disc / Digital Video Disc
EDVAC	Electronic Discrete Variable Automatic Computer
ENIAC	Electronic Numerical Integrator And Computer
EPI	Echo Planar Imaging
EPR	Electronic Paramagnetic Resonance
FAPESP	Fundação de Amparo à Pesquisa do Estado de São Paulo
FIR	Finite Impulse Response
FM	Frequency Modulation
fMRI	Functional Magnetic Resonance Imaging
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HCI	Human-Computer Interaction
IDE	Integrated Development Environment
IP	Internet Protocol

JSON	JavaScript Object Notation
LCD	Liquid Cristal Display
LED	Light-Emitting Diode
MRI	Magnetic Resonance Imaging
NCO	Numerically Controlled Onscillator
NMR	Nuclear Magnetic Resonance
NUI	Natural User Interface
ODIN	Object Oriented Development Interface for NMR
OLED	Organic Light-Emitting Diode
PDP-1	Programmed Data Processor-1
PM	Phase Modulation
pMRI	Parallel Magnetic Resonance Imaging
POO	Programação Orientada a Objetos
PyMR	Python Magnetic Resonance
RF	Radiofrequency
RM	Ressonância Magnética
SE	Spin Echo
SEMS	Spin Echo Multi Slices
SUS	Sistema Único de Saúde
TCP	Transmission Control Protocol
TE	Tempo ao Eco
ToRM	Tomografo de Ressonância Magnética
TR	Tempo de Repetição
UML	Unified Modelling Language
USB	Universal Serial Bus
VHDL	Very High Speed Integrated Circuits hardware Description Language
WAN	Wide Area Network
WWW	World Wide Web
XML	Extensible Markup Language

SUMÁRIO

1	INTRODUÇÃO	25
1.1	CONTEXTUALIZAÇÃO	25
1.1.1	<i>Definição do problema.....</i>	<i>25</i>
1.1.2	<i>Crterios de sucesso.....</i>	<i>27</i>
1.1.3	<i>O projeto ToRM.....</i>	<i>27</i>
1.2	OBJETIVOS.....	28
1.3	RESSONÂNCIA MAGNÉTICA	29
1.3.1	<i>O fenômeno de ressonância magnética</i>	<i>30</i>
1.3.2	<i>Hardware de ressonância magnética</i>	<i>31</i>
1.4	COMPUTAÇÃO	36
1.4.1	<i>Métodos e ferramentas de desenvolvimento de software.....</i>	<i>37</i>
1.4.2	<i>Escolha da linguagem de programação</i>	<i>39</i>
1.4.3	<i>Bibliotecas.....</i>	<i>41</i>
1.4.4	<i>API.....</i>	<i>42</i>
1.4.5	<i>IDE.....</i>	<i>43</i>
1.5	COMPUTAÇÃO E RESSONÂNCIA MAGNÉTICA	45
2	DESENVOLVIMENTO.....	49
2.1	METODOLOGIA.....	49
2.1.1	<i>Pesquisa com usuários.....</i>	<i>51</i>
2.1.2	<i>Gerenciamento do desenvolvimento</i>	<i>58</i>
2.1.3	<i>Implementação.....</i>	<i>58</i>
2.1.4	<i>Equipamento para testes reais</i>	<i>60</i>
2.2	RESULTADOS E DISCUSSÃO.....	61
2.2.1	<i>A biblioteca PyMR.....</i>	<i>62</i>
2.2.2	<i>ToRM-API</i>	<i>69</i>
2.2.3	<i>ToRM-IDE</i>	<i>71</i>
2.2.4	<i>Execução da metodologia no Espectrômetro ToRM</i>	<i>77</i>
3	CONCLUSÃO.....	79
3.1	TRABALHOS FUTUROS.....	80
	REFERÊNCIAS	81
	APÊNDICE A – QUESTIONÁRIO APLICADO AOS USUÁRIOS E PROGRAMADORES DE SEQUÊNCIAS	87

1 INTRODUÇÃO

1.1 Contextualização

Os equipamentos de Ressonância Magnética (RM) têm sido um dos meios mais promissores para pesquisas nos últimos anos, principalmente no que se refere a química e física médica. Equipamentos que foram concebidos a mais de 40 anos e que, embora, tenham sido aprimorados ao longo do tempo, sofrem muito com suas interfaces e ambientes para gerar as sequências de pulsos, quando estas ainda existem. Assim, muitas vezes, o pesquisador permanece na espera pela empresa ou enfrenta as grandes dificuldades de gerar novas sequências. Em atenção à este problema, surgiu a iniciativa de se gerar, com tecnologia nacional, um espectrômetro que forneça meios mais rápidos aos pesquisadores envolvidos, podendo se adaptar facilmente as suas necessidades. Como parte do projeto ToRM-15, este trabalho tem foco principal na etapa de desenvolvimento das metodologias e suas ferramentas.

1.1.1 Definição do problema

O principal problema se situa na falta de uma interface mais amigável e generalizada para programação de sequências de pulsos e a falta de ferramentas que auxiliem esta tarefa de forma integrada e generalizada. A importância do estudo destes problemas se reflete na grande dificuldade de programação de sequências nos modos para sua concepção dos modelos atuais. Além disso, os principais atores para a concepção de tais sequências não são, em sua maioria, especialistas em programação, mas sim físicos e químicos em conjunto com outros profissionais, que enfrentam grandes dificuldades no entendimento e implementação destas. É sabido também que, a incorporação de várias áreas na programação de sequências se faz necessário para um desenvolvimento multidisciplinar, já que além das áreas de física e química, outras áreas como a medicina, engenharia, agricultura, farmacologia, entre outras,

fazem uso da tecnologia de RM para estudos, aperfeiçoamento e análises.

As possibilidades com o desenvolvimento de soluções para este problema são inúmeras. Entre elas podemos citar a criação de novas sequências onde dificilmente são criadas devido à limitação dos sistemas atuais ou a alta complexidade para sua implantação e incorporação de novas área e pesquisadores que, muitas vezes, ficaram distantes pelas dificuldades encontradas. Além disso, algumas empresas têm uma nova perspectiva do uso de seus equipamentos, não visando apenas a venda dos mesmos como um produto final, fechado, mas também para aqueles que desejam desenvolver algo específico para uma aplicação como parte de seu equipamento. Sendo assim, as interfaces que dão acesso a estes dispositivos (API's) são bem vindas, como forma de simplificar a comunicação e também proteger o código proprietário do sistema, excluindo a necessidade de se entender todo o projeto do equipamento original.

No caso de insucesso no tratamento deste problema, os envolvidos com programação de sequências continuarão a depender exclusivamente do desenvolvimento a partir das empresas que produziram o equipamento, dificultando o avanço mais rápido e limitando as diferentes vertentes que o estudo por demais profissionais poderia gerar.

Neste caminho, os principais desafios para sua realização são a contemplação das inúmeras possibilidades que incluem a adaptação para os diferentes hardwares existentes e a estruturação de forma que permita uma programação e interface fáceis para o usuário final. Por fim, a aceitação por parte de outras empresas no uso destas ferramentas pode ser incentivado pelo equipamento desenvolvido pelo CIERMag (Centro de Imagens e Espectroscopia por Ressonância Magnética) no projeto ToRM-15, que visa a versatilidade e grande adaptabilidade do equipamento, o qual é descrito brevemente na seção seguinte.

De certo, a concatenação de todas as informações úteis para fornecer uma biblioteca e um IDE completos e de fácil interação transformam este problema em um desafio maior. Isto implica em incluir, também, toda a gama de programação, suas definições e ferramentas de modo que mesmo aqueles que não são programadores, em sua essência, possam trabalhar sem grandes problemas.

1.1.2 Critérios de sucesso

A solução deste problema deve atender a todos os envolvidos, nos diferentes níveis, e promover a fácil utilização das ferramentas geradas, dos quais citamos os programadores de softwares que gerarão todo o código de mais baixo nível, as pessoas responsáveis pela instalação dos equipamentos e manutenção, os criadores de sequências, os especialistas que farão o uso mais específico e o devido aperfeiçoamento de determinada sequência e até mesmo o simples operador que fará uso da interface criada para refinamentos e estudos. Em relação às características dos softwares e códigos gerados, estes devem englobar as mais novas técnicas de programação e de ressonância magnética de forma a permitir uma fácil integração entre diferentes ferramentas, hardwares, sistemas operacionais e ser escalável, para que assim possa completar seu objetivo.

1.1.3 O projeto ToRM

O projeto ToRM teve seu início em 1987 com um papel importante no desenvolvimento da tecnologia de imagens por ressonância magnética nacional e na América Latina de forma geral. Um marco importante foi o desenvolvimento do ToRM-005 de 500 Gauss iniciado em 1989. Além deste sistema, outros indicadores importantes marcaram a trajetória do grupo desde 1983, com as primeiras imagens geradas no hemisfério sul (imagem de um quiabo), seguidos da definição do primeiro sistema pré-clínico baseado em um magneto supercondutor de 2.0 Tesla.^{1,2} Em 1995 iniciou-se a operação de um sistema clínico localizado em um hospital local, que operou por cerca de 8 anos e gerou mais de 8 mil exames de RM, em grande parte para o SUS. Outro sistema pré-clínico foi desenvolvido visando aplicações em Neurociências, parte pela integração do grupo à rede estadual CInAPCe, parte pelo programa homônimo da FAPESP, e que culminou com a criação do próprio do Centro de Imagens e Espectroscopia *in vivo* por Ressonância Magnética – CIERMag – que abriga atualmente este projeto.

Este trabalho faz parte do projeto ToRM-15, que permitiu a criação de um

Espectrômetro Digital de RM com completo domínio da tecnologia pelo Centro o qual ainda vem sendo aperfeiçoado. O sistema possui sintetizado todo o hardware de controle dos pulsos e aquisição em FPGA (*Field-Programmable Gate Array*), o que permite que o hardware seja gerado por síntese a partir de linguagens como VHDL (*Very High Speed Integrated Circuits hardware Description Language*).³ Um dos principais objetivos do uso de síntese de hardware é a possibilidade de atualização, manutenção e adaptação do equipamento apenas pela substituição do hardware sintetizado e, se necessário, novos periféricos. Desta forma o espectrômetro tem uma vida útil muito maior sem necessidade de troca drástica de componentes ou mesmo sua substituição completa. Em adição, tem a capacidade de atender uma área maior de possíveis experimentos que podem ser executados com o mesmo equipamento, proporcionado pela sua fácil adaptação. Além disso, as ferramentas desenvolvidas neste trabalho farão parte de sua principal característica, a fácil adaptabilidade, de forma que o projeto como um todo seja algo que impulsionará a tecnologia dos equipamentos de RM e que até o momento nunca foram oferecidos nesta completude. Isso permitirá o real uso destes equipamentos como ferramentas de pesquisa não necessitando da intervenção ou espera dos fornecedores dos equipamentos.

1.2 Objetivos

Os objetivos destacam-se pelo desenvolvimento de uma biblioteca que forneça acesso ao hardware de RM para programação de sequências de pulsos, disponível aos vários níveis de usuários existentes, de forma a permitir uma interface de programação moderna, amigável, escalável e tendo a capacidade de se adaptar aos diferentes tipos de hardware. Além dessa vertente, gerar uma API específica para o projeto ToRM (ToRM-API) juntamente com um IDE, que trate especificamente de programação para RM como, por exemplo, programação gráfica de sequências e ferramentas que auxiliem na criação e manutenção das inter-relações entre parâmetros. Nos objetivos citados anteriormente, espera-se integrar as necessidades e sugestões dos usuários e programadores de sequências de forma a facilitar e agilizar o desenvolvimento em todas as etapas. O objetivo geral visa a atualização da questão do software de ressonância magnética gerando uma estrutura uma fácil e extensível, que possa se

adaptar aos vários equipamentos, em especial ao hardware do projeto ToRM-15, onde a busca pela grande versatilidade é essencial.

1.3 Ressonância Magnética

A RM está dividida em dois grandes campos, não concorrentes, mas complementares, que se compõe por Espectroscopia por Ressonância Magnética (NMR - *Nuclear Magnetic Resonance*) e Imagens por Ressonância Magnética (MRI – *Magnetic Resonance Imaging*). A primeira delas é muito desenvolvida por cooperação entre químicos orgânicos e a indústria de equipamentos de RM, enquanto a segunda está mais próxima da medicina e observou um processo acelerado de evolução desde a sua concepção devido ao extremo sucesso desta interação. Observa-se um fenômeno atual que se repete com o advento da Imagem Funcional de RM, que encontrou abrigo como uma das principais ferramentas da Neurociência, mas isso não impede o uso de tais técnicas em outras áreas médicas.⁴ Por exemplo, podemos citar o uso da espectroscopia de *voxel* localizado em MRI para determinação de enzimas que podem ajudar a diagnosticar patologias. Atualmente, com a evolução dos equipamentos e melhoria das técnicas, a RM está sendo empregada em vários ramos da ciência e tecnologia como petróleo, solo, agricultura, engenharia de materiais e aeronáutica além das tradicionais áreas de química e medicina. Algumas de suas vantagens estão associadas a ser uma técnica não destrutiva, ou seja, não há modificações no objeto em estudo após a análise, a alta seletividade, que de acordo com cada técnica, permite serem observados fenômenos diferentes numa mesma amostra com muita facilidade, a radiação presente é não ionizante, pois a radio frequência, devido a sua faixa de frequência e consequentemente sua energia, não age diretamente em tecidos vivos, por exemplo, destruindo as cadeias de DNA como o efeito da radiação ionizante dos raios-X ou raios-gama. Além disso, é facilmente aplicada, em suas principais vertentes à sólidos ou líquidos e *in vivo* ou *in vitro*.⁵⁻⁷

1.3.1 O fenômeno de ressonância magnética

A técnica de RM se baseia na propriedade de *spin*, nuclear ou eletrônico, e sua interação com radiações da ordem da frequência de Larmor na faixa de RF (radiofrequência). Equipamentos específicos, os espectrômetros de RM, excitam os núcleos que estão em um campo magnético estático e captam os sinais com bobinas de RF de forma que se consegue mapear o espaço do objeto em estudo no caso MRI ou fazer o estudo das estruturas atômicas devido as suas interações em NMR. Gradientes de campo magnético podem ser utilizados para modificar ligeiramente o campo local, introduzindo uma codificação espacial da informação de forma que possam ser mapeadas setores diferentes de uma mesma amostra. Abordaremos aqui, superficialmente, o assunto de sequências de pulsos as quais são bem tratadas em outros trabalhos ⁽⁷⁻⁹⁾. Somente para que o leitor tenha uma ideia mais próxima do que se trata tais sequências e de como elas se relacionam com este trabalho, na figura 1, mostramos uma sequência conhecida como *Spin Echo (SE)*, onde cada linha representa um subsistema de hardware juntamente com as formas de onda dos pulsos que serão transmitidos, que estão ao longo destas linhas, as quais representam uma sequência temporal.

Figura 1 - Exemplo de uma sequência de pulsos, *Spin Echo*, onde cada linha representa o dispositivo e as formas de onda que serão geradas através deles numa sequência temporal. Fonte: Adaptada de *Spin Echoes*, E.L. Hahn, DOI: 10.1103/PhysRev.80.580.

Esta sequência, uma das possíveis modificações da sequência básica *Spin Echo*, pode ter ainda outras variantes que servem para aprimorar a técnica, as quais não foram indicadas

nesta figura, que mostra apenas o essencial para sua conceituação na técnica de imagens.¹⁰ Descrevemos a funcionalidade de cada linha da figura 1 como segue. A linha RF representa o sinal gerado pelo transmissor, englobando como veremos mais a frente às modulações em amplitude, fase e frequência do sinal. As linhas Seleção, Fase e Leitura representam os gradientes lógicos de campo magnético, também como veremos mais a adiante, estes estão relacionados com os gradientes físicos nas direções X, Y e Z. Na linha Fase há um detalhe onde os vários traços dentro da forma de onda representa sua variação em cada *loop*. Ao final, as linhas ADC e Sinal representam a habilitação do receptor e a captura do sinal (de forma ilustrativa), respectivamente. Tendo em mente que o controle do sistema feito pelo hardware do equipamento em conjunto com tais sequências de pulsos, isso gera a instrumentação de RM, portanto, trataremos nas próximas seções e capítulos dos principais hardwares e softwares, procedendo assim uma breve introdução ao assunto tratado neste trabalho.

1.3.2 Hardware de ressonância magnética

Nesta seção, trataremos do hardware básico dos equipamentos de RM que, de modo geral, são formados por partes em comum para equipamentos para NMR e MRI. Seguimos com a separação do hardware em dois grandes grupos que são os dispositivos e os periféricos. Dispositivos são aqueles que estão dentro do gerador/controlador de pulsos e que são representados pelos gradientes, receptores, transmissores e outros dispositivos auxiliares como controles de cama para MRI ou de rotação de amostra para NMR. Observe que estes dispositivos são os que fazem o controle dos possíveis periféricos. Por exemplo, o dispositivo de gradientes é quem controla todos os periféricos de gradiente, como os amplificadores e fornecem assim, os dados necessários para sua utilização. Já os periféricos são todos os equipamentos anexos a este sistema que permite, em conjunto com os dispositivos, que sejam gerados os subsistemas como, por exemplo, o sistema de gradientes que é composto pelos dispositivos de gradientes (geradores de pulsos, pré-ênfase, rotação) e pelos periféricos como cabos, amplificadores e bobinas. Na figura 2 pode ser visualizada uma figura que resume dois subsistemas, o de gradientes e o de transmissores.

Figura 2 - Dispositivos e periféricos dos sistemas de gradientes e de transmissores. Fonte: Elaborado pelo autor.

Nos dois itens que seguem tratamos de forma um pouco mais aprofundada, cada um destes dois grupos, os dispositivos e periféricos, que de modo básico compreendem os principais itens dos equipamentos de RM.

Dispositivos

Nas figuras a seguir podemos ver um resumo dos principais dispositivos, que compreendem os dispositivos de gradientes, transmissão e recepção. O primeiro componente para os dispositivos de gradientes são os geradores de pulsos, que geram as formas de onda, figura 3. Após ser recebida a forma de onda a ser gerada, a partir dos gradientes lógicos (Seleção/*Slice*, Codificação de Fase e Leitura/Frequência), os sinais são transformados nos canais físicos X, Y e Z através de uma operação de rotação.

Figura 3 - Componentes da cadeia do dispositivo de gradientes. Fonte: Elaborado pelo autor

Após esta operação o sinal segue para a pré-ênfase, negativa ou positiva, que se caracteriza por corrigir as possíveis correntes parasitas geradas pela atuação dos gradientes próximos a partes metálicas que compõem o sistema.¹¹⁻¹³ A partir da pré-ênfase o sinal segue para o conversor digital-analógico (DAC). Para os dispositivos de transmissão, figura 4, os moduladores trabalham basicamente da mesma forma que os geradores de gradiente, gerando formas de onda, se consideradas as três componentes de modulação: amplitude (AM –

Amplitude Modulation), fase (PM – *Phase Modulation*) e frequência (FM – *Frequency Modulation*) do pulso de RF onde o sinal é modulado com ajuda do NCO (*Numerically Controlled Oscillator*) que fornece as diferentes frequências (em fase, em quadratura, frequência base, etc). Finalmente a saída por um DAC leva o sinal para os amplificadores (periféricos).

Figura 4 – Componentes da cadeia do dispositivo de transmissão. Fonte: Elaborado pelo autor.

Na recepção, figura 5, como o sinal é recebido e não gerado, o conversor analógico-digital (ADC) participa da conversão de sinais no sentido inverso ao DAC, convertendo agora sinais analógicos para digitais.

Figura 5 - Componentes da cadeia do dispositivo de recepção. Fonte: Elaborado pelo autor.

Após a passagem pelo ADC, se inicia uma cadeia de filtros que são responsáveis por limitar a quantidade de dados que serão registrados. Dentre os principais tipos de filtros, citemos o CIC (*Cascade Integrator Comb*) e sua principal funcionalidade no circuito é de realizar a primeira redução de amostragem conhecido como *down sampler*. Um segundo tipo de filtro vastamente utilizado é o filtro FIR (*Finite Impulse Response*) o qual possui coeficientes, normalmente 64 ou 128 deles, que podem ser calculados ou gerados por vários métodos e normalmente são conhecidos como de janelas. Por fim, os dados são armazenados no *buffer* para o envio ao computador. Como parte complementar dos subsistemas, os dispositivos necessitam de componentes extras para a realização de sua tarefa, estes equipamentos complementares são os periféricos que serão vistos a seguir.

Periféricos

Como dito anteriormente, os periféricos dizem respeito a todo o hardware anexo aos dispositivos, como amplificadores, pré-amplificadores, cabos, bobinas entre outros. Assim, trataremos resumidamente de cada um destes itens, os quais são relacionados no esquema da figura 6.

Figura 6 - Periféricos comuns para um sistema básico de ressonância magnética. Fonte: Elaborado pelo autor.

A introdução de cabos no mapeamento do circuito pode parecer irrelevante, mas quando tratamos de sinais com altas frequências, RF, por exemplo, as características dos cabos como impedância e blindagem podem ser importantes para ajustes de ganho devido às perdas no trajeto do sinal. Os atenuadores são usados para que ajustes possam ser feitos de forma que o máximo de digitalização nos DAC's possa ser utilizado sem que, com isso, o sinal de entrada dos amplificadores esteja saturado.

Em um sistema de RM existem vários tipos de amplificadores, dedicados a cada subsistema, entre eles estão os amplificadores de RF que são os que atuarão sobre as bobinas de transmissão de forma que o sinal do gerador seja amplificado. De modo geral amplificadores de RF devem trabalhar com altas tensões ($\sim$ kV), baixas correntes ($\sim$ mA) e frequências altas da ordem de MHz.¹⁴

Já no caminho de volta, ou seja, na recepção, temos os pré-amplificadores que atuam sobre o sinal captado pelas bobinas receptoras. Como o sinal gerado pelo sistema de *spins* do objeto em estudo é muito baixo, e há perdas nos cabos que fazem as conexões entre a bobina receptora com o receptor em si, um dispositivo de pré-amplificação se faz necessário para que

o sinal chegue a níveis detectáveis, fazendo com que o máximo de informação possa ser recuperada quando convertido o sinal analógico para digital. Sua principal característica é ter baixa figura de ruído.¹⁴

Como podemos notar no esquema da figura 6, há somente um transdutor de RF (*probe* de transmissão e recepção), exigindo que seja necessário a introdução de um comutador entre os circuitos de alta e baixa potência. Desta forma, o comutador de RF atua como chave comutadora entre transmissão e recepção, permitindo a atuação de alta potência na transmissão e a recepção em baixa potência e baixo sinal-ruído. Este dispositivo comumente é automático, utilizando diodos de potência juntamente com circuitos de carga que impedem que alta potência seja levada para os receptores, o que danificaria o circuito.¹⁴

Por fim, os amplificadores de gradiente possuem características um pouco diferentes dos de RF. Devem trabalhar com altas correntes ($\sim 100\text{A}$), baixa tensão, de (~ 100)V e em baixa frequência (100-1000 Hz). Algumas de suas características compõem informações sobre a taxa de variação do gradiente (*slew rate*) e o seu ganho.¹⁴

Na maioria dos subsistemas existem bobinas, que de modo geral, são elementos capazes de produzir um campo eletromagnético, ou seja, elementos com a propriedade de indução magnética, capazes de gerar, a partir de uma força eletromotriz um campo eletromagnético e vice-versa. Dependendo de sua construção e/ou aplicação elas podem ser especializadas em campo estático ou campo variável, ter melhor uniformidade ou homogeneidade. Alguns exemplos podem ser vistos na figura 7. Bobinas de RF ou os chamados transdutores de RF são os responsáveis por transmitir e receber o sinal de RF. Suas principais características são de ter uma boa sensibilidade ($\sim \text{mT}/\text{cm}^2$).¹⁵

Figura 7 – a) Bobina de RF para imagem de punho, volumétrica. b) Bobina de RF de superfície. Fonte: Cedido pela empresa parceira FIT - *Fine Instrument Technology*.

Já as bobinas de gradiente são responsáveis por gerar os gradientes de campo nas direções X, Y e Z no centro do magneto e neste caso, o termo linearidade é atribuído à estas bobinas, relacionando o quanto o campo é linear em função da tensão ou corrente aplicada.

Por fim, as bobinas de *shimming* que são necessárias para realizar o ajuste fino no campo (*shimming* de ordem zero, um, dois,..., n-ésima), melhorando assim sua homogeneidade. Estas bobinas possuem características que são muito próximas as das bobinas de gradientes, porém trabalham em mais baixa potência e o campo produzido é constante, não sendo pulsados.

Por ultimo, o magneto que, para o sistema, é o elemento que gera o campo magnético estático B_0 o qual é da ordem de Tesla. O magneto deve fornecer um campo estável e ter boa homogeneidade espacial, além de ter as dimensões para o sistema em operação. Na figura 8 mostramos dois exemplos de sistemas, o primeiro deles de MRI e o segundo NMR de bancada, ambos com subsistemas desenvolvidos em parceria com a empresa FIT – *Fine Instrument Technology* com o mesmos dispositivos em FPGA.

Figura 8 - Sistema da FIT. a) Magneto aberto de 0,3T Centauro e conjunto de bobinas, ao fundo, para MRI. b) Rack de amplificadores do sistema de MRI. c) Magneto permanente para espectroscopia e uso em bancada ao lado do espectrômetro e computador. Fonte: Cedido pela empresa parceira FIT - *Fine Instrument Technology*.

Todas as principais características expostas nesta seção serão utilizadas para a caracterização das classes, métodos e funções presentes na biblioteca e API geradas. Dando sequência a exposição, entraremos nas características da computação e depois mais especificamente na computação envolvida com a RM.

1.4 Computação

Desde sua concepção os computadores executavam tarefas e na medida do desenvolvimento, linguagens, formas de armazenamento, arquitetura e componentes foram sendo atualizados para se ter, para a grande maioria dos usuários destes equipamentos, formas

padrões de se utilizar, seja os dispositivos de entrada e saída ou mesmo acesso a memória, disco e demais elementos. Isso inclui, também, a portabilidade de códigos entre diferentes máquinas ou um protocolo padrão de comunicação como, por exemplo, TCP/IP. Em um segundo momento, foram introduzidas às linguagens de programação paralela que trouxeram um novo conceito em programação e estas também aderiram a alguns formatos padrões como, por exemplo, algumas arquiteturas.¹⁶ A padronização facilita a comunicação entre desenvolvedores e entre usuários e entre os diferentes sistemas, apesar é claro, de nunca agradar a todos ou incluir todas as possibilidades. Mencionamos tal fato, pois, em meio aos vários softwares e sistemas existentes em RM, pouco há em padrões ou comunicação entre eles, dificultando a interação e portabilidade.^{17,18} Itens estes que foram levados em consideração no momento da construção da biblioteca, API e IDE que são os objetos deste trabalho. Nos tópicos que seguem tratamos de alguns pontos cruciais para o desenvolvimento de softwares que se farão necessários durante a execução do projeto.

1.4.1 Métodos e ferramentas de desenvolvimento de software

Várias áreas da computação tratam particularmente da interação humano-computador conhecida como HCI (*Human-Computer Interaction*). Tais métodos auxiliam na busca pela melhor interface que auxiliará a compreensão, usabilidade, robustez, entre outras características que podem estar no foco do projeto.^{19,20} Desta forma, métodos como aplicação de questionários, entrevistas e acompanhamento durante as tarefas foram realizados para se obter as especificações que atendessem a todos os interessados.²¹ A aplicação destes métodos proporcionou uma coleção de informações sobre os tipos de usuários, problemas envolvidos além do contato com os softwares já existentes. Além disso, outras questões como a localidade, que é a capacidade do software de se adaptar a cultura local (língua, notação decimal, entre outros) da região onde está sendo executado, garante que a comunicação com o usuário seja feita de forma precisa. Os itens mais importantes são a língua, formato de data e hora, formatação de número (vírgula ou ponto) e moeda, fusos horários.

Assim como as ferramentas já citadas, o desenvolvimento de novos softwares estão amparados por métodos que agilizam o processo e fazem dele um mecanismo iterativo, garantindo melhor produtividade. Métodos como *Scrum*, que são baseado no modelo iterativo

e incremental *Agile* de desenvolvimento de software, promovem etapas que garantem sempre a usabilidade do código.²² Na figura 9 podemos ver o diagrama de processos envolvidos no *Scrum*.

Figura 9 – Etapas do processo de desenvolvimento de software *Scrum*. Fonte: Adaptada de http://en.wikipedia.org/wiki/File:Scrum_process.svg.

Basicamente, o método *Scrum* se inicia no *Product Backlog*, que são as listas de requerimentos, onde a partir delas são escolhidas tarefas chaves para serem implementadas, que seguem para a *Sprint Backlog*. Cada tarefa é ainda subdividida para ser realizada em, no máximo, 24h dentro de uma *Sprint*, a qual pode durar de sete dias até um mês. Ao final de uma *Sprint* uma versão do software é liberada. Assim, o processo se repete até que se tenha o software todo implementado. O uso deste método permite fáceis modificações no trajeto, as quais seriam dificilmente implementadas nos métodos sequenciais tradicionais, onde depois de projetado, pouco pode ser alterado. Em adição, métricas de softwares podem ser utilizadas para que se mantenham os critérios de qualidade durante as implementações. Atualmente, existem ferramentas que realizam medidas sobre o código-fonte, fornecendo prontamente os valores para esta verificação. Algumas medidas importantes são: o índice de manutenibilidade, que indica o grau dificuldade de se fazer manutenção no código que inclui a facilidade, segurança e economia na execução das ações e o nível de complexidade ciclomática, o qual indica o quão complexo é um programa através do calculo do caminhos independentes gerados para cada ação - grupo indivisível de comandos²³. Na seção seguinte, mostramos algumas informações que são úteis na escolha da linguagem de programação para o projeto.

1.4.2 Escolha da linguagem de programação

Um dos principais itens a serem tratados quando no início da execução de um projeto de software é a escolha da linguagem de programação, por isso, trataremos de alguns itens importantes para a escolha da linguagem, do paradigma e das ferramentas.

Muitas vezes, devido à evolução dos computadores e seu poder de processamento, o desempenho é negligenciado, porém este é um quesito importante onde uma quantidade muito grande de dados é processada. Em contrapartida, o poder computacional oferecido pela linguagem, ou seja, a quantidade de otimizações e recursos fornecidos prontamente pela linguagem também deve ser levado em consideração, onde boa parte do trabalho pode estar prontamente disponível.

Em seguida, a consideração do nível da linguagem é um item a ser ponderado durante a escolha para o projeto. Sabe-se que quanto mais próximo da linguagem de máquina melhor pode ser o desempenho, porém mais exigente torna-se o conhecimento do programador e mais difícil é a implementação de código. Em oposição, uma linguagem de alto nível pode oferecer uma maior facilidade de programação, mas acarretar em uma menor eficiência. Contudo, quando tratamos de uma linguagem de alto nível, como Python ou C++, o desempenho do código está, muitas vezes, ligado ao algoritmo usado. Assim, no caso da possibilidade de escolha, o melhor caso pode ser aquele que permite fácil integração com outras linguagens de modo que, módulos que exigem características diferentes possam ser construídos com linguagens diferentes. Deste modo, reintegrados os módulos, podemos melhorar o desempenho total, na execução e no desenvolvimento do código.

Quando tratamos um problema que deve ser resolvido ou auxiliado pela construção de um programa, as linguagens de programação têm certos padrões ou formas de tratá-los utilizando estruturas semânticas, sintáticas ou léxicas, tais estruturas são chamadas paradigmas de programação. Um paradigma de programação é certo padrão na forma de resolver o problema proposto, em alguns casos uma mesma linguagem pode suportar vários paradigmas de programação, ficando a cargo do programador a escolha do mais indicado. De forma básica podemos tratar dois dos principais paradigmas: o funcional e o orientado a objetos. O primeiro, funcional, é mais voltado à matemática e à teoria da computação, relacionado diretamente com as áreas de inteligência artificial, tomada de decisões e análise e processamento de dados, onde a estrutura principal é baseada em funções. No que se refere à

programação orientada a objetos, esta é baseada em descrever os sistemas e objetos do mundo real existindo uma estrutura hierárquica, divisão de classes, métodos - que são ações destas classes e seus atributos. Softwares comerciais comumente utilizam este último paradigma tanto pela sua representatividade quanto pela sua facilidade de manutenção e gerenciamento.²⁴ Sendo assim, o paradigma reflete como será tratado o problema, auxiliando na estruturação do código.

Outro fator importante está relacionado à quantidade de documentos e ferramentas que podem ser encontrados relacionados à linguagem escolhida. Uma linguagem bem difundida, por exemplo, pode ter muito material e de fácil acesso para consulta. Em adição a documentação, quando lidamos com grandes projetos uma ferramenta muito útil são os ambientes integrados de desenvolvimento (IDE). Estas ferramentas trazem facilidades que garantem agilidade e diminuem a necessidade de memorizar métodos ou funções, além de facilitar o acesso a documentação durante a escrita do código. Além dos IDE's, a utilização de gerenciadores de versão, integrados ou não ao IDE, garante a documentação do desenvolvimento e, é claro, o gerenciamento correto da versão dos códigos-fonte. Somado a estes dois, tem-se também ferramentas que podem contribuir com a agilidade de produção de documentação interna do código, conhecidos como *docstrings*, importante especialmente para desenvolvedores que auxiliam no *parsing*, passagens de parâmetros, etc.

Do lado do usuário, podemos resumir a escolha da linguagem padrão como sendo a linguagem do sistema operacional base, ou da linguagem que forneça mais interação com o sistema operacional em uso. No caso do uso de sistemas operacionais diferentes temos que pensar na linguagem que mais oferece suporte a operação multiplataforma. Somado a isso, uma linguagem que provê uma grande quantidade de bibliotecas e *frameworks* e que também possua qualidade e grande respaldo da comunidade é de grande importância para o projeto. Criar seu próprio código é importante no momento do aprendizado, mas quando tratamos de produtividade e garantia de funcionamento, além de uma boa documentação, o uso de códigos de alta qualidade oferecidos pela própria linguagem é de grande ajuda. Problemas comuns que necessitam ser resolvidos, possivelmente já foram resolvidos e testados por grandes empresas, instituições ou mesmo a própria comunidade de usuários. O uso de recursos oferecidos pela linguagem garante o melhor caminho quando se trata da criação de uma ferramenta e não de um objeto de pesquisa. Outro motivo para o uso de recursos disponíveis na linguagem é que, se o assunto tratado não é o foco da biblioteca ou *framework* utilizado, então certamente alguém, outra instituição ou empresa já construiu algo de muito mais alta qualidade do que você mesmo poderia produzir.

Por fim, o uso de licenças adequadas para que satisfaçam as necessidades dos projetos é de grande ajuda para a melhoria da qualidade e agilidade no desenvolvimento. A inserção de bibliotecas de terceiros é algo de grande uso tanto na academia quando na indústria de software. Porém, isso deve ser visto com cautela no quesito de licença e de descontinuação do projeto da biblioteca utilizada, o que pode gerar problemas futuros para o projeto. Na questão do tipo de licença a ser aplicado ao produto final, convém perceber que ferramentas que possam ter seu código aberto permitem uma evolução muito mais rápida e com grande respaldo pela comunidade.²⁵

1.4.3 Bibliotecas

Bibliotecas oferecem funções, classes, métodos e tipos especializados para uma determinada tarefa ao que se propõe a dar suporte, comumente próprias de uma linguagem. Ao contrário da API, como veremos no tópico seguinte, ela não é especializada para um determinado caso, um produto, mas é especializada para fornecer solução para um problema, genérico, de uma determinada área. As bibliotecas em conjunto com a linguagem formam a base necessária para um desenvolvimento em camadas de forma mais controlada. De forma geral, a construção de softwares com base nos itens mencionados anteriormente pode ser resumida na figura 10, onde podemos ver a relação entre vários itens.

Figura 10 - Relação entre bibliotecas, API's e o software especializado e alguns exemplos. Fonte: Elaborado pelo autor.

1.4.4 API

Embora muitas das fontes não façam distinção entre biblioteca e API, consideramos que uma API está um nível mais próximo ao usuário, ou seja, a API proporciona ao usuário um estágio mais próximo ao que ele planeja no projeto final, onde são necessários apenas algum *setup*, com pouca necessidade de implementação extra. Mais recentemente uso de API's tem se generalizado para criação dos *plug-ins*, onde uma API específica é fornecida pelo proprietário do software para que outros autores possam ter acesso as principais e desejadas funcionalidades para estender o programa principal. Alguns exemplos de produtos que tem API's são aplicativos como Facebook, Google Maps, Microsoft Office, Open Office, Eclipse entre outros. A comunidade, além de fazer uso das API's, pode também contribuir para seu desenvolvimento, neste ponto os avanços são bem maiores do que produtos sem este tipo de interface. Cuidados devem ser tomados no momento sua criação e gerência, alguns deles como tratar do desenvolvimento da API com o mesmo esforço do seu produto principal, já que ela é quem irá ser a porta de comunicação com outros sistemas. Além disso, mesmo que toda a comunidade possa contribuir para seu desenvolvimento, o melhor conhecedor dela deverá ser o próprio criador, mantendo-se, em conjunto, um local como uma pagina web com informações e criar fóruns, além da documentação de uso e desenvolvimento. Outro fator importante é o uso de um *sandbox*, que é um local onde pessoas podem testar os produtos que usam sua API sem ter que pagar ou ter muita burocracia para isso. Muitas vezes a API pode se tornar o principal produto ao longo do tempo, gerando um efeito cascata que pode alavancar o produto principal.²⁶⁻³²

1.4.5 IDE

As chamadas IDE's são ambientes integrados de desenvolvimento que reúnem características e ferramentas como editores de texto, compiladores, interpretes, depuradores, modeladores de UML³³ (*Unified Modelling Language*), automação de testes e onde geralmente se tem uma interface gráfica avançada de modo a agilizar o processo de desenvolvimento.^{34,35} Atualmente está cada vez mais difícil conseguir manter grandes projetos somente com programação em CLI (*Command Line Interface*). O conceito de IDE surgiu de imediato com o início das linguagens de programação onde se usavam os terminais gráficos para tal tarefa. Em conjunto com o desenvolvimento das linguagens de programação, o conceito de IDE também evoluiu para o que conhecemos hoje, onde tais ambientes agregam inúmeras ferramentas. A primeira delas é datada de 1975, chamada Maestro I onde o conceito de ambiente, em sua forma mais completa, começou a ser aplicado.³⁴ A partir de 1995 as IDE's começaram a utilizar o conceito de plug-ins para adicionar novas ferramentas que seriam agregadas ao software principal.³⁴ Entre os IDE's mais conhecidos nos dias atuais podemos citar o Eclipse (The Eclipse Foundation), NetBeans (Oracle Corporation), Visual Studio (Microsoft) e o Dev C/C++ (Bloodshed Software). Na figura 11 podemos ver o esquema básico de um IDE com suas principais funcionalidades.

Figura 11 - Esquema básico de um IDE, com navegador de arquivos, editor de código, *debugger* entre outros.
 Fonte: Adaptada de <http://eric-ide.python-projects.org/eric-screenshots>.

Embora existam IDE's projetadas para linguagens específicas, a maior parte delas já comporta o tratamento de múltiplas linguagens e com multipropósito, o que representa uma tendência atualmente, incluindo também ser multiplataforma. Também com esta tendência de generalidade, em sua maioria, estas ferramentas são de código aberto e/ou gratuitas o que promove mais rapidamente seu desenvolvimento. Elas incluem em seu arcabouço API's, que facilitam a criação de plug-ins para inserção ou adaptação de novas funcionalidades.³⁵ De posse destes breves resumos dos principais assuntos da área de Computação que precisamos para nosso trabalho, seguimos com a integração entre a RM e a Computação, no que se refere aos softwares utilizados para controle dos equipamentos.

1.5 Computação e ressonância magnética

Os equipamentos de ressonância magnética sempre caminharam juntos com os avanços da computação, principalmente na questão de reconstrução de imagens e processamento onde, por muito tempo, foi o gargalo da técnica.^{18,36} Atualmente o poder de processamento das máquinas já garante com tranquilidade a viabilidade de todas as técnicas conhecidas, incluindo a integração com técnicas de processamento paralelo.³⁷ No quadro 1, como forma de ilustrar esta trajetória, podemos ver uma breve descrição da evolução das tecnologias de RM e de Computação desde sua concepção até os dias atuais.

Quadro 1 - Comparativo da evolução das técnicas e tecnologias de Ressonância Magnética e da Computação entre décadas de 1930 a 2010.

Década	Ressonância Magnética	Computação
1930	Descoberta do spin.	Maquina de Turing, Primeiro computador com válvulas, Cartão perfurado.
1940	Descoberta do fenômeno de ressonância magnética, Primeiro EPR, Primeiro NMR.	ENIAC, EDVAC, Mark I, Mark II, Transistor.
1950	Spin-Echo, Primeiro DNP.	Computadores transistorizados, Fita magnética, Primeiro Compilador, Linguagem Fortran, Modem, Cobol.
1960	Espectroscopia FT-NMR.	PDP-1, Fita cassete, ARPANet, WAN, Programação em linhas, Linguagem Basic.
1970	Uso de gradientes, Primeira imagem 1D, MRI para propósito médico, Primeiro equipamento MRI patenteado.	Microprocessadores, Computador comercial, VAX, Monitor CRT, Disquete, TCP/IP, Linguagem C.
1980	Equipamento de corpo inteiro, Imagem cardíaca, fMRI, Equipamentos comerciais, EPI, Supercondutores.	IBM-PC, PC-XT, Disco rígido, Mouse, GUI, Internet, Linguagem C++.
1990	Phased array, Magneto aberto.	Pentium, Monitor LCD, CD, DVD, WWW, Linguagem Python, Java.
2000	Compressive sensing, pMRI.	Processadores multi-core, Flash drive, USB, Programação em GPU, Monitor LED/OLED/3D, 10GB Ethernet, NUI
2010	30.000.000 de exames só nos EUA.	Mobile computing, Cloud computing, 100GB Ethernet, 1TB Ethernet, Tablets, Smartphones

Fonte: A historical overview of magnetic resonance imaging, focusing on technological innovations, PMID: 23070095; A brief history of nuclear magnetic resonance, DOI: 10.1021/ac00054a716, Early History of Magnetic Resonance, Physics in Perspective, 1999, Volume 1, Issue 2, pp 123-135; SMASH, SENSE, PILS, GRAPPA: How to Choose the Optimal Method, Martin Blaimer et al, Top Magn Reson Imaging 2004;15:223-236, Timeline of computing history, IEEE, 1996; http://en.wikipedia.org/wiki/History_of_computing_hardware, Magnetic Resonance Imaging: Historical Perspective, DOI: 10.1080/10976640600755302.

Um fato interessante pode ser destacado do quadro 1, sobre a técnica de EPI, que produz imagens ultrarrápidas, a qual precisou esperar cerca de 10 anos para que pudesse ser implementada comercialmente devido ao poder computacional necessário para reconstrução das imagens e a falta de engenharia adequada para se produzir gradientes ultrarrápidos. Isso mostra a grande interdependência entre as tecnologias de RM, Computação e Engenharia.

Algo que podemos notar na evolução dos equipamentos é que pouco se evoluiu nos softwares que geram as sequências de pulsos e na forma de programação de tais sequências. Muitas das empresas utilizam C como linguagem de descrição de sequências, além de terem estruturas altamente complexas e pouco flexíveis. Temos consciência de que grandes modificações são difíceis de serem implementadas nos sistemas, os quais devem garantir compatibilidade entre versões, porém muitas vezes são necessárias para atualizações e melhorias.

Uma das grandes críticas às empresas que produzem os equipamentos é em relação à distância em que foram postos os pesquisadores externos, quando se veem as dificuldades que são encontradas ao fazer modificações nas sequências e isso quando é possível. Com isso, muitas vezes, as pesquisas podem ficar dependentes das empresas pela dificuldade de geração de novas metodologias ou modificações que seriam necessárias para atender aos pesquisadores. Este ponto de vista foi fornecido pelos próprios desenvolvedores de metodologias, mas há também o ponto de vista do usuário, ou melhor, do operador. Lidar com tais equipamentos é algo difícil, existindo várias questões que poderiam ser implementadas para sua melhoria. As interfaces para uso dos equipamentos são complicadas, além do necessário para o uso, dificultando o trabalho dos operadores e até mesmo levando-o a cometer erros. Desta forma, parte deste trabalho também é oferecer algo mais simples e ainda funcional baseado no estado-da-arte da computação. Com base no texto anterior, comentaremos a seguir sobre três dos principais trabalhos que se relacionam de modo mais próximo com o que estamos tratando neste momento.

O ODIN³⁸ (*Object Oriented Development Interface for NMR*) é um software amplo, figura 12, que utiliza uma linguagem rápida e bem conceituada que é o C++. Integra ao

desenvolvimento de sequências, um desenvolvedor de pulsos e também um simulador, itens importantes para desenvolvimento de sequências. Possui um visualizador da sequência gerada, porém não editável. Apesar de usar uma linguagem orientada a objetos e rápida como o C++, a complexidade para a programação de sequências é relativamente grande e exige um aprendizado mais aprofundado na questão, principalmente, de programação. A estrutura de organização das sequências, ou seja, da forma que é concebida deixa um pouco a desejar, dificultando a leitura holística da sequência a partir do código fonte.

Figura 12 - Interface de parâmetros da sequência do software ODIN. Fonte: <http://odin.sourceforge.net/index.html>

Um segundo trabalho muito interessante é o DAMARIS³⁹ (*Darmstadt Magnetic Resonance Instrument Software*) tendo a linguagem nativa o Python aliada a uma API relativamente pequena e até mesmo restritiva, pois trata somente de experimentos em 1D. Usa a linguagem Python que, especialmente para aqueles que não trabalham exatamente com isso, é mais fácil do que outras como o C++. Sua interface possui poucos comandos os quais podem ser facilmente decorados, sem necessidade de consulta aos manuais a todo o momento. Porém, ainda une as etapas de desenvolvimento da sequência com a aquisição de dados e processamento e não fornece uma interface muito amigável. Sua interface gráfica pode ser visualizada na figura 13.

Figura 13 - Visualização da interface gráfica do DAMARIS. Fonte: http://element.fkp.physik.tu-darmstadt.de/damaris_cms/index.php?id=a-not-so-short-tutorial-on-damaris.

Por fim o trabalho intitulado *Novel Software Architecture for Rapid Development of Magnetic Resonance Applications*⁴⁰ provê a ideia mais próxima do nosso trabalho, porém sem levar em conta a inserção de pessoas que não necessariamente programadores no desenvolvimento de novas sequências. Sua linguagem nativa é o Java com integração com C++ para partes em que se exige rapidez na execução. O repertório gerado é voltado ao desenvolvimento interno de software, deixando a desejar o uso por desenvolvedores externos a empresa, por exemplo, a falta de editor gráfico de sequência. Mesmo assim, reúne grandes ideias, relações e configurações que contribuem muito para a base de nosso trabalho.

Embora todos estes softwares contribuam para a atualização de ideias, nenhum deles foi capaz de gerar ferramentas independentes para desenvolvimento de forma amigável e simples. Nosso trabalho, portanto, será reunir ideias de forma que o desenvolvimento seja uma das partes integrantes da linha do tempo de criação das metodologias de RM, promovendo a integração de uma biblioteca, uma API e um IDE. A biblioteca realizará um tratamento genérico do problema, fornecendo estruturas fundamentais para a programação de metodologias e acesso aos sistemas. A API será gerada para ser a interface com o hardware específico, de forma a ser sua porta de comunicação e acesso a seus atributos. Por fim, o IDE será a ferramenta de auxílio na concepção das sequências de pulsos que permitirá todo o controle do projeto. Os procedimentos para a execução do projeto são descritos a seguir no capítulo de metodologia.

2 DESENVOLVIMENTO

2.1 Metodologia

A metodologia de desenvolvimento seguiu métodos tradicionais, portanto, não serão tratados de modo aprofundado neste trabalho, mas trarão as referências e as informações necessárias nos tópicos que seguem. As informações necessárias para o atendimento dos objetivos propostos (biblioteca, API e IDE) foram extraídas de várias fontes, entre elas artigos, livros e documentos como manuais e *datasheets* que serviram para a composição das classes, propriedades e métodos as quais, algumas delas, foram mencionadas no capítulo de introdução. Demais fontes consultadas encontram-se junto a documentação do software. Além dessas fontes, consultas aos usuários dos equipamentos de RM fornecendo, em conjunto, as especificações para o desenvolvimento.

Um dos objetivos deste trabalho é dividir as fases da linha do tempo de concepção de uma metodologia de RM em três etapas - desenvolvimento, execução e processamento - e trabalhar na primeira delas. O diagrama da figura 14 mostra em resumo como uma metodologia é concebida, executada e os dados processados.

Figura 14 - Linha do tempo da metodologia de RM dentro da nova visão de software: Desenvolvimento – onde são geradas e gerenciadas as sequências, testes e validações; Execução – Geração dos protocolos, refinamento e validação experimental; Processamento – Processamento dos dados e refinamento dos algoritmos de processamento. Fonte: Elaborado pelo autor.

A fase de execução é realizada pela console de controle e após os dados serem adquiridos, são processados pelos módulos de processamento que também podem ser separados da execução.⁴¹ Tendo esta primeira divisão em mente, nosso trabalho segue por tratar do item de desenvolvimento onde farão parte as bibliotecas, API e IDE geradas. Como

mentionado no capítulo da introdução, a biblioteca fornece estruturas, funções e métodos generalizados que tratam da programação dos equipamentos de RM. A biblioteca gerada foi intitulada PyMR como sigla de *Python Magnetic Resonance*, e que este primeiro momento, foram gerados protótipos do formato geral da biblioteca, seguido de partes funcionais os quais entram para os testes no projeto ToRM. A API, que é uma interface específica para o projeto ToRM, nomeada ToRM-API e representa o hardware funcional até o presente momento. A principal ideia com esta estruturação é que a biblioteca PyMR seja o *front-end* para programação e *setup* de sistema, de forma que a API, sendo uma reimplementação desta, forneça o *back-end*, ambos de forma padronizada. Com isto, mais a frente, pode-se pensar em usar uma mesma metodologia, ou seja, uma mesma sequência de pulsos para qualquer equipamento, se programada com auxílio da PyMR. Por fim, um protótipo funcional do IDE, nomeado ToRM-IDE, específico para projetos de RM foi desenvolvido, seguindo alguns trabalhos anteriores do CIERMag, de forma que este ainda não foi totalmente substituído por *plug-ins* para IDE's existentes, como mencionado na introdução. Esta primeira versão serviu como um protótipo funcional, permitindo avaliar algumas características do uso de tal ferramenta como auxiliadora nos projetos de metodologias. Na figura 15 é descrita a inter-relação entre os materiais gerados.

Figura 15 - Estrutura e relação entre a biblioteca PyMR, a ToRM-API e o ToRM-IDE juntamente com a metodologia gerada através delas. A PyMR representa um *front-end* para programação generalizada enquanto que a ToRM-API fornece o *back-end* para um equipamento específico. O ToRM-IDE é o ambiente para geração de metodologias e protocolos, que auxilia o usuário nesta tarefa. Fonte: Elaborado pelo autor.

Mais detalhes de cada uma das estruturas poderão ser vistos na seção 2.2. Como forma de ilustrar o projeto ToRM como um todo, a figura 16 representa os elementos envolvidos, incluindo itens além dos que serão tratados neste trabalho, apenas como forma de situar o leitor.

Figura 16 - Projeto ToRM completo relacionado com os itens deste trabalho: PyMR, ToRM-API e ToRM-IDE na etapa de desenvolvimento da linha do tempo da metodologia. A sequência em linguagem F, até o momento, que é parte da metodologia é compilada pelo ToRM-Compiler para envio para a FPGA. Estes itens foram desenvolvidos por Felipe B. Coelho e Guilherme M. Freire. O ToRM-Console faz o controle do uso do espectrômetro, providenciando a execução das metodologias e ToRM-Processing realiza o processamento dos dados - desenvolvidos por Danilo M.D.D. da Silva. Por fim, o ToRM-Acq-Server que permite a comunicação com a FPGA foi desenvolvido por Mateus J. Martins. Fonte: Elaborado pelo autor com base na documentação interna do CIERMag.

A previsão é que a descrição da sequência, atualmente na linguagem F devido a trabalhos anteriores, possa ser substituída pela descrição em Python utilizando a biblioteca PyMR. Citaremos agora, brevemente, alguns dos métodos utilizados para o desenvolvimento do projeto além, é claro, das principais ferramentas de software.

2.1.1 Pesquisa com usuários

No que se refere à pesquisa com usuários foram providenciados questionários online, entrevistas e acompanhamento durante o uso dos equipamentos/software seguindo as recomendações das práticas de pesquisa de HCI, onde diferentes tipos de pesquisa são

realizados para colher as informações necessárias para o desenvolvimento do protótipo e/ou novas funcionalidades. A aplicação dos questionários *online* foi realizada por e-mail sendo cerca de 40 pessoas contatadas das quais 27 responderam. O questionário possuía cerca de 30 questões referentes ao uso dos equipamentos, programação destes, programação de forma geral e questões de cunho profissional, o qual pode ser visualizado no Apêndice A, página 87, e que foi disponibilizado em português e inglês. Dos 27 candidatos que responderam, quatro eram de fora do país. Os questionários foram direcionados, principalmente, para programadores de sequências nos quais estão inclusos físicos, físicos-médicos e químicos, em sua maioria. O principal objetivo era colher informações sobre como era a interação atual dos equipamentos do mercado e quais as principais dificuldades na programação de sequências.

Após aplicados os questionários e de posse dos dados foram providenciados os gráficos a seguir, os quais proporcionam várias informações sobre os usuários dos equipamentos. Na figura 17.a, temos uma informação pouco esperada quando vemos que os sistemas operacionais baseados em Linux estão quase com a mesma quantidade de usuários que os sistemas derivados do Windows. Porém, as figuras seguintes, figura 17.b e figura 17.c, mostram que apesar das pessoas terem conhecimento de ambos os sistemas, a tendência a usarem softwares que são específicos para o sistema Windows como o NotePad e o Visual Studio, são maiores que os demais. Estes dados forçam a ideia inicial, apresentado na introdução, de sistemas multiplataforma, considerando crucialmente os sistemas Windows e Linux em uma primeira etapa.

Figura 17 – Número de pessoas que utilizam as diferentes ferramentas: a) Sistemas operacionais. b) Editores de texto. c) IDE's. Fonte: Elaborado pelo autor.

Neste momento avaliamos as diferenças entre os equipamentos que os entrevistados têm conhecimento, de modo que na figura 18 temos uma clara tendência na opinião dos usuários à dizer que a praticidade no uso (vermelho) é quase idêntica entre os equipamentos, enquanto que a semelhança no uso (azul) e na programação (verde) é distinta. Esse é um dos dados dos quais nos chama a atenção para o fato da dificuldade de programação entre os diferentes equipamentos.

Figura 18 – Grau de semelhança no uso, praticidade e programação dos softwares de equipamentos de RM.
Fonte Elaborado pelo autor.

Estes dados podem ser reforçados pelas respostas da questão aberta sobre as principais dificuldades ou empecilhos na programação de sequências para os equipamentos, as quais são comentadas ao final desta seção. A justificativa muitas vezes apontada é que as empresas usam dessas dificuldades e diferenças para manter o usuário em seu sistema, sendo esta atitude considerada precária e ultrapassada. Talvez a melhor justificativa é de que há uma grande dificuldade em se modificar e atualizar estes softwares, mantendo ao mesmo tempo, a compatibilidade entre eles. Porém, há sempre a possibilidade de se iniciar um novo projeto que ao longo do tempo permitirá a transição entre as versões, da mesma forma como outras tecnologias fizeram ao longo do tempo.

Uma análise foi realizada sobre como o usuário lida com os diferentes tipos de erros na programação de sequências, indicada na figura 19, o que reflete que é tão difícil ou mais lidar com os erros do que outras formas que programação.

Figura 19 - Como o usuário lida com os diferentes tipos de erros na programação de sequências de pulsos quando comparadas a programação comum. Fonte: Elaborado pelo autor.

Quando tratamos de como o usuário aprende a usar o equipamento ou programar sequências, como observado na figura 20.a, vemos que quase a metade aprende pela documentação dos equipamentos e a outra metade por treinamentos da própria empresa. Isso

nos leva, ao criar o nosso software, que haja uma vasta documentação para que os usuários possam aprender a usá-lo de forma correta e eficiente.

Figura 20 - De que forma o usuário aprende a utilizar e/ou programar o equipamento. Fonte: Elaborado pelo autor.

Quanto às dificuldades encontradas por aqueles que não programam sequências, em sua maioria as dificuldades são, em ordem decrescente: o pouco conhecimento de programação, a dificuldade em lidar com o equipamento e o pouco conhecimento do mesmo como observados na figura 21. Isso valida a escolha da linguagem Python como linguagem base para o projeto, tida como uma das mais fáceis para o aprendizado. Na questão seguinte, podemos observar que há mais uma justificativa para tal escolha.

Figura 21 - Quais as principais razões para não programar uma sequência segundo os usuários. Fonte: Elaborado pelo autor.

Quando questionados do conhecimento de linguagens de programação de forma geral, figura 22.a, não se limitando a programação de sequências de pulsos, o maior número de pessoas tem conhecimento nas linguagens MatLab, C++, C e Fortran, fato que pode ser atribuído as várias das empresas que empregam estas linguagens além de serem muito bem difundidas e utilizadas para desenvolvimento de cálculos matemáticos e algoritmos de processamento de dados.⁴² Contudo, entre elas, ainda podemos ver a linguagem Python, mostrando que os usuários tem certo contato com ela. O uso da linguagem do MatLab foi descartado por ser parte de uma ferramenta proprietária. Na figura 22.b, o pouco conhecimento ou a pouca familiaridade com POO foi refletido em nosso trabalho com o uso de hierarquias pequenas e evitando o uso excessivo de polimorfismo, a reimplementação de métodos e outras características que pudessem necessitar de conhecimentos mais aprofundados.

Figura 22 - a) Linguagens de programação mais conhecidas entre os usuários de RM entrevistados*. b) Conhecimento dos entrevistados sobre POO. Fonte: Elaborado pelo autor.

Procuramos entender também quem faz as modificações quando há necessidade de modificações drásticas nas sequências e como a tarefa é realizada. A figura 23.a mostra que os 33% dos usuários recorrem aos especialistas do grupo e que independente de quem realiza essa tarefa, sempre encontra alguma dificuldade para resolver o problema, como podemos ver na figura 23.b. Essas informações reforçam que há dificuldade envolvida em se resolver os problemas nos equipamentos, mesmo pelos especialistas ou técnicos das empresas tarefa que, pode ser facilitada por nossa implementação.

Figura 23 – a) Quem realiza modificações nas sequências quando são necessárias. b) Qual o grau de dificuldade empregado para realizar a modificação do item a. Fonte: Elaborado pelo autor.

Sobre o conhecimento dos entrevistados nos diferentes equipamentos, podemos notar pela figura 24.a que houve um grande número de usuários com grande experiência, de modo que tivemos 8 equipamentos com, pelo menos, algum usuário com um pouco de familiaridade. Com isso, temos garantias de que as opiniões fornecidas são de usuários experientes. Na figura 24.b os entrevistados foram arguidos sobre como é a programação textual das sequências nos equipamentos por eles mencionado, de forma que notamos que em sua grande maioria todos tem dificuldades ou não gostam de como é a programação. As respostas destas questões evidenciam a insatisfação dos usuários com os atuais sistemas.

* O item *Outras* no gráfico mostra a soma de todas as demais linguagens mencionadas, as quais em nenhuma delas o valor fosse superior ao número do menor dado (Python).

Figura 24 - a) A familiaridade dos usuários entrevistados com os equipamentos disponíveis no mercado[†]. b) Qualidade da edição das sequências de forma textual em cada equipamento. Fonte: Elaborado pelo autor.

Uma segunda questão, sobre como é a edição gráfica de sequências (questão 2.1 apresentada no Anexo A) gerou, de certa forma, uma dubiedade de significado, pois somente um dos equipamentos realmente tinha a possibilidade de edição gráfica da sequência. Como houve resposta para os demais equipamentos, crê-se que os usuários responderam a questão pelo uso da interface gráfica de controle do equipamento. Desta forma, preferiu-se anular as repostas desta questão a qual deveria ter sido reformulada e reaplicada para se obter dados mais precisos.

Como questões finais, a formação dos usuários, mostrada na figura 25.a, foi ligeiramente inclinada para físicos e físicos médicos em relação a outros como, por exemplo, químicos, devido a lista de contatos que possuíamos. Sobre a faixa etária dos entrevistados, figura 25.b, nota-se uma tendência de pessoas com grande experiência, entre 31 e 50 anos de idade.

Figura 25 - a) Formação acadêmica dos entrevistados. b) Faixa de idade dos entrevistados. Fonte: Elaborado pelo autor.

Por fim, como um resumo das questões abertas realizadas e das sugestões que nos

[†] A série de dados referente a *Nada Familiarizado* foi omitida por não trazer informação extra nesta questão.

foram enviadas, segue por ordem de frequência, uma listagem dos principais assuntos:

- Principais problemas:
 - Dificuldade na compreensão/aprendizado no formato do código adotado pela empresa;
 - Falta de editores gráficos de sequência;
 - Falta de experiência em programação;
 - Pouca informação sobre as sequências ou documentação não compatível com a realidade do código ou funcionalidade do equipamento;
 - Curva de aprendizado longa e desestimulante;
 - Dificuldade em modificar os parâmetros, suas inter-relações e como são mostrados (aba de controle).
- Possíveis melhorias:
 - Melhor documentação;
 - Melhores treinamentos/cursos/tutoriais;
 - Criação de editor gráfico;
 - Uso de linguagem de programação mais fácil;
 - Simulação gráfica e programação das sequências de forma generalizada;
 - Modularização e estrutura mais lógica do código;
 - Possibilidade de leitura holística do código;
 - Ferramentas de auxílio;
 - Padronização.
- Itens que auxiliariam na flexibilidade das sequências/equipamentos:
 - Possibilidade de modificações nos *loops*;
 - Mudança de nomes de parâmetros e inter-relação entre eles;
 - Reutilização de trechos da sequência de forma eficiente (bibliotecas);
 - Montagem de sequência em blocos.

Em resumo, o questionário aplicado reflete a necessidade do desenvolvimento deste trabalho, pela insatisfação e dificuldades que os usuários e desenvolvedores de sequências têm com os sistemas atuais. Seguimos agora com a descrição das ferramentas, métodos e materiais usados na implementação do código.

2.1.2 Gerenciamento do desenvolvimento

Os sistemas desenvolvidos foram baseados no paradigma de Programação Orientada à Objetos.²⁴ Diagramas simplificados de classes, hierarquia e pacotes foram gerados para o controle do software de modo geral.³³ Alguns destes diagramas podem ser vistos no capítulo de resultados, exemplificando como é a estrutura da biblioteca PyMR. Do conjunto de métodos e ferramentas utilizados para os itens pretendidos por este trabalho, destacam-se:

- Scrum²²: método *Agile* utilizado para o desenvolvimento de software, empregado em sua forma simplificada, onde algumas etapas foram omitidas, como reuniões diárias, por exemplo.
- Mercurial⁴³ (HG): controlador de versão distribuído que promoveu o gerenciamento das versões de todos os códigos gerados.
- Redmine⁴⁴: gerenciador de projeto de plataforma web que auxilia na concentração de informações sobre *bugs*, *features* e requisições integrável ao Mercurial.
- The Eric Python IDE⁴⁵ /Eclipse⁴⁶: ambientes de desenvolvimento de software.
- Epydoc⁴⁷: gerador automático de documentação para códigos em Python a partir de campos *docstring* do código permitindo fácil manutenção e navegação.
- Radon²³: forneceu as principais estatísticas do software como a complexidade ciclomática, índice de manutenibilidade além de valores de número de linhas de código, comentários, documentação entre outras.

Na seção seguinte expomos os materiais utilizados para a implementação dos códigos, além de algumas características adotadas quanto a língua e sistemas de unidades.

2.1.3 Implementação

A implementação do código foi realizada na linguagem Python⁴⁸, motivado pela facilidade de codificação, vasta documentação, fácil integração com outras linguagens,

simplicidade, portabilidade, grande número de bibliotecas, API's e ferramentas, além de ser reforçada pelas respostas do questionário aplicado.⁴² Dentre as bibliotecas utilizadas podemos citar a NumPy⁴⁹, para cálculos numéricos, e JSON⁵⁰, para manipulação de arquivos de protocolos, entre outras básicas como *os*, *sys* e *shutils*.

Uma questão importante quando tratamos de quantidades físicas como medidas de distâncias, comprimentos, tempo, frequências, entre outras, é que há sempre a preocupação com a conversão dos valores, problemas de escalas e de unidades. Em um software como o deste trabalho, que opera o tempo todo com grandezas físicas, isso deve ser algo elementar. Para tanto, foi utilizado o módulo PhysicalQuantity da ScientificPython⁵¹, que proveu as conversões, tratamentos de erros de conversões e demais benefícios e facilidades aos cálculos. Deste modo, além da facilidade nos cálculos podem-se gerar muitas variações na interface gráfica que aceita, por exemplo, mudanças de sistema de unidades e/ou a escala multiplicativa, o que não é encontrado nas interfaces atuais.

Para a interface gráfica foi utilizada o PySide⁵² que é uma interface em Python da *framework* Qt⁵³, esta implementada em C++ e desenvolvido pela Nokia.⁵⁴ Qt é uma moderna e robusta *framework* para interfaces gráficas se comparada com Tk. Atualmente, sua versão é compatível com os sistemas Windows, Linux e MacOS, além de dispositivos móveis.⁵²

Como mencionado na introdução, a localidade reflete-se na língua e em outros pontos como unidades de medida e notação numérica. Sendo assim, uma breve busca sobre os países envolvidos na pesquisa com RM é mostrada a seguir, a qual determinou tais características, ou neste caso, reforçou o que já era esperado. A métrica utilizada constatou na busca pelo tópico “*magnetic resonance*” em todas as bases da *Web of Knowledge* no período de 1940 a 2013, que forneceu aproximadamente 530.000 resultados. Filtramos estes resultados pelos 20 primeiros países e territórios em número decrescente de publicações fornecendo, então, os dados da figura 26 onde podemos ver a língua Inglesa, de modo geral, sendo o principal idioma, como já era de esperado, pois, esta é a linguagem universal da ciência.

Figura 26 - Língua do país de origem das publicações com o tópico ‘*magnetic resonance*’ em todas as bases da *Web of Knowledge* no período de 1940 a 2013[‡]. Fonte: Elaborado pelo autor.

Considerando estes dados, promoveu-se todo o desenvolvimento com base no idioma Inglês, sendo as demais inseridas por prioridade na lista de traduções requeridas para documentações e interface, a pode ser traduzida utilizando o sistema do Qt. O sistema internacional (SI)^{55,56}, este foi adotado como base para as unidades e suas conversões. Em conjunto com os dois itens mencionados, procurou-se utilizar, por padrão, a notação numérica internacional, onde o ponto representa o início da casa decimal nas interfaces gráficas.

Por fim, todos os códigos foram implementados levando-se em consideração a utilização, principalmente, nos sistemas operacionais Windows e Linux. Tal adoção que é confirmada pelos resultados do questionário aplicado. Como os sistemas MacOS são baseados em Unix estes, também, dificilmente terão problemas de compatibilidade. Os códigos foram testados em Windows 7 e 8 e Linux 12.04, 12.10 e 13.04.

2.1.4 Equipamento para testes reais

Durante e após a geração dos itens propostos neste trabalho, sequencias foram sendo desenvolvidas para serem testadas no espectrômetro local, mostrado na figura 27, onde sequencias como CPMG (*Carr-Purcell-Meiboom-Gill*), SE (*Spin Echo*), entre outras, foram

[‡] Foram consideradas somente duas diferenças entre a uma mesma língua que pode ser visto no Inglês (Americano e Britânico) e Português (Brasileiro e de Portugal).

utilizadas.

Figura 27 - Equipamento de testes do projeto ToRM-15. a) Conjunto magnético e amplificadores (RF e gradientes). b) Conjunto magnético evidenciando as partes que o compõem. c) Núcleo do espectrômetro: os dispositivos em FPGA para geração de pulsos e recepção do sinal. Fonte: Elaborado pelo autor com base na documentação interna do CIERMag.

O equipamento é composto de um magneto permanente de $NdFeB$ (Neodímio, Ferro e Boro) com campo de $0.56T$ e distância entre peças polares de $30mm$. Tal equipamento foi utilizado para aplicação de técnicas de espectroscopia, pois imagens não puderam ser geradas neste momento devido a problemas com a calibração do sistema de gradientes ocasionadas por correntes parasitas (*eddy currents*) provocadas pelo núcleo metálico do magneto. Na recepção/transmissão conta com uma bobina tipo *Saddle*. A FPGA, núcleo do espectrômetro, possui dois canais de transmissão, dois de recepção e três gradientes. De posse dessas informações, seguimos para os resultados onde são expostos cada um dos itens produzidos, incluindo a execução da metodologia no ToRM-Console, que serviu como validação no sistema real.

2.2 Resultados e discussão

Procedendo com as técnicas mencionadas na seção anterior, foi iniciado o desenvolvimento do ToRM-IDE na forma de um IDE independente e não como *plug-ins* para os já existentes, isso devido a derivação de trabalhos anteriores no CIERMag. Em conjunto

como IDE, sendo as informações adquiridas ao longo do trajeto, foi criada e incrementada a ToRM-API. Isto pois, para a construção da biblioteca era necessário um grau de generalidade maior. Deste modo, gradativamente algumas das funcionalidades da ToRM-API foram repassadas para a PyMR. Este processo decorreu até o final deste trabalho, garantindo que as implementações da PyMR carregassem apenas informações genéricas dos equipamentos de RM. Nos tópicos que seguem, tratamos dos códigos gerados para a biblioteca PyMR, ToRM-API, ToRM-IDE e finalmente os testes práticos.

2.2.1 A biblioteca PyMR

A biblioteca foi dividida em dois grandes pacotes que representam o núcleo (*core*) e a interface gráfica (*GUI*) que fornecem as funcionalidades básicas e a interação gráfica com o usuário. A PyMR teve grande parte implementada e os demais itens são protótipos (nomes, estruturas, descrição) gerados a partir das especificações, que fornecem o formato e a estrutura básica para sua devida implementação. Na figura 28 apresentamos o diagrama de pacotes do *core*.

Figura 28 - Diagrama dos pacotes e módulos do *core* da PyMR. Fonte: Elaborado pelo autor com base na documentação da PyMR.

Dentro do *core* há ainda alguns outros pacotes como *peripherals*, *devices* e *methodology* que reúnem os principais módulos. Como mencionados na introdução, os periféricos representam os sistemas anexos, como amplificadores e bobinas, e os dispositivos

os itens internos ao controlador do espectrômetro, como geradores de pulsos de RF e gradientes. Esta divisão ajuda a simplificar o *setup* do sistema e a implementação do *back-end*, a API. O último pacote, *methodology*, descreve a sequência de forma genérica e seus diferentes níveis serão vistos mais a frente. Como forma de exemplificar o que foi exposto até o momento, a figura 29 resume como é a estrutura entre metodologia, dispositivos e periféricos.

Figura 29 - Estrutura geral da relação entre metodologia, dispositivos e periféricos que formam a base da PyMR, exemplificada com o subsistema de gradientes. As três camadas garantem simplicidade na descrição, criação e manutenção das funcionalidades do sistema. Fonte: Elaborado pelo autor.

Note que há uma clara divisão entre as partes envolvidas, onde a metodologia faz acesso aos dispositivos por seus nomes lógicos, exemplificados pelos gradientes de Seleção, Fase e Leitura e em contrapartida, os periféricos se referem aos dispositivos por seus nomes físicos, os gradientes X, Y e Z. Uma observação para a nomenclatura adotada neste trabalho é que quando nos referimos a dispositivo ‘lógico’, isso quer dizer no sentido de como ele é visto pelo programador de sequência e quando mencionamos dispositivo ‘físico’ estamos relacionando o significado ao que o hardware significa. Por exemplo, para um programador de sequência, a informação de qual é o gradiente X, Y e Z não é importante, a não ser para fins de posicionamento. O que é o importante para ele são os gradientes lógicos de Seleção, Fase e Leitura. A conexão entre o significado dos dispositivos lógicos e físicos é feita dentro da própria classe de dispositivos. Como dissemos anteriormente, o intuito da PyMR é fornecer uma interface genérica para os sistemas de RM e embora existam funcionalidades que possam depender de cada sistema de modo específico, podemos considerar essas funcionalidades como *features*, ou seja, algo que possa ser agregado, mas que não seja essencial para a descrição. A seguir trataremos de algumas outras estruturas que puderam ser

definidas e que descrevem cada um dos elementos apresentados na figura anterior.

Níveis de uma metodologia

Em nossa concepção a metodologia foi dividida em vários níveis, de forma que cada nível tenha uma característica diferente. Apresentamos cada um dos níveis de forma breve

Metodologia: consiste de basicamente de uma sequência e vários protocolos. A sequência é um método generalizado capaz de se adaptar a diferentes funcionalidades através dos protocolos. Exemplo: Metodologia *Spin Eco* com protocolos para imagens de cabeça, joelho, abdômen, etc.

Sequência: Uma sequência pode conter vários módulos os quais podem ou são ser executados todas às vezes. Existem módulos de uso geral, que uma vez desenhados para executar como uma sequência podem ser identificados e disparados sob controle do console quando necessários. Assim funcionam os módulos de ajuste de *shim*, ajuste de *flip angle*, ajuste da fase do espectrômetro.

Módulo: Módulos são conjuntos de eventos que são executados, um após o outro, e sem qualquer tipo de interferência do usuário. Porém, módulos podem ser condicionados a *triggers* dos dispositivos do equipamento de RM ou seus periféricos. Por exemplo, o *trigger* cardíaco e/ou o *gate* respiratório, que garantem que enquanto não são estabelecidos o sistema fica em espera.

Evento: Eventos são pequenas partes de um módulo onde todo o conteúdo dos dispositivos é executado em paralelo (pelo hardware). Os eventos, assim como módulos e sequências, são elementos reutilizáveis e comumente são nomeados por suas tarefas. Por exemplo, um evento de seleção pode ativar um gradiente e um transmissor ao mesmo tempo.

Os elementos apresentados acima fornecem um amplo formato para programação de sequências e que até o momento não mostraram nenhum empecilho. Somado a isto, o controle da metodologia, que é realizada pelo usuário no momento da execução, é feita através de uma aba de parâmetros fornecida para tal finalidade a qual é descrita na seção seguinte, juntamente com as funcionalidades dos protocolos e scripts.

Parâmetros, protocolos e *scripts*

Os parâmetros, de um modo geral, são variáveis que são comumente modificadas pelos usuários para uso específico de uma metodologia, por exemplo, o Tempo ao Eco (TE) e o Tempo de Repetição (TR). São estes parâmetros que aparecem na aba de controle da console de operação e que permite serem modificados pelos usuários no momento da utilização. Assim, os desenvolvedores de metodologias podem gerar tais abas de forma que o operador faça uso delas no momento da execução. A interface gráfica das abas são automaticamente geradas quando o desenvolvedor preenche os campos do parâmetro na seção de hierarquia que reflete na classificação primária entre *System*, *Sequence* ou *Real Time* na figura 30 e suas sub abas e caixas. Sua posição dentro do mesmo box pode ser modificada pelos itens de posicionamento do parâmetro. De forma padrão, os parâmetros são apresentados por uma legenda e um campo de valor, que depende do tipo associado. Na figura 30 mostramos alguns parâmetros com tipos diferentes, entre eles, ponto flutuante, inteiro, quantidade física e lista.

Figura 30 - Exemplo de parâmetros visualizados na aba de controle. Fonte: Elaborado pelo autor com base na documentação interna do CIERMag.

Os protocolos e scripts são associados em três formas distintas: protocolo de execução, protocolo de aquisição e protocolo de processamento. Um protocolo contém todos os atributos dos parâmetros necessários para uma determinada tarefa e este arquivo é escrito no formato JSON⁵⁰. Este formato de armazenamento estruturado é mais humanamente legível se comparado a formatos tradicionais como o XML⁵⁷. Por exemplo, para uma mesma sequência CPMG, podemos gerar um protocolo para execução para imagens ou somente para tempo de relaxação, ou mesmo protocolos especializados para diferentes tipos de amostras. Já nos scripts em Python, tem a informação do que é necessário ser feito com os parâmetros adquiridos da aba de controle para executar determinada tarefa ou validação de seus valores.

Neste caso, como temos acesso a todos os parâmetros podemos criar um script que valide o valor de TR que deve ser, no caso de uma sequência de eco de spin simples, no mínimo duas vezes o valor de TE. Deste modo, as três categorias de protocolos e scripts são explicadas nos itens abaixo:

Execução: contém todas as informações necessárias para a execução do experimento. Valores como, por exemplo, número de *slices*, número de codificação de fase ou núcleo de observação e suas inter-relações e validações escritas no script.

Aquisição/reconstrução: juntamente com seu *script*, especifica como deve ser feito a reestruturação dos dados, ou como devem ser lidos os dados do arquivo gerado pelo espectrômetro. Isso é necessário, pois cada sequência recebe dados de formas diferentes, de acordo como ela é executada. Assim, com um script que faça a reestruturação, sempre teremos um formato de dados padrão na saída, o qual pode ser lido facilmente por outras ferramentas.

Processamento: partindo dos dados já organizados de forma padrão, pode-se fazer o processamento sobre eles, por exemplo, uma transformada de Fourier, gerando outros conjuntos de dados que nunca se superpõem aos dados originais, de forma que possam ser sempre gerados, a partir deste protocolo e script, os dados finais.

Com os protocolos e scripts isolados podemos ter uma aba ou interface diferente para cada etapa desta na console de operação, sendo esta criada automaticamente a partir dos parâmetros referidos nos protocolos. Como citamos na introdução, existem vários níveis de usuários que engloba, por exemplo, os administradores, os gerenciadores de metodologias e especialistas. Na próxima seção apresentamos cada um deles exemplificando suas atribuições.

Grupos de usuários

Os equipamentos de RM possuem diferentes níveis de usuários que executam determinadas tarefas no sistema que são, do nível mais alto ao mais baixo em permissões: (1) o **proprietário** é quem irá gerar a API do sistema e gerar todas as configurações básicas para uso do equipamento e o software essencial; (2) o **administrador**, é o usuário que irá estabelecer o hardware do equipamento e as funcionalidades (instalação), além de ser o responsável pela manutenção do mesmo; (3) o **gerente** é o indivíduo que irá produzir as metodologias, ou seja, programar as sequências de pulsos e gerar os protocolos básicos para

seu funcionamento. Dentre seus deveres, deverá gerar metodologias que possam ser as mais genéricas possíveis de forma que os protocolos criem sua especificidade, como exemplo os físicos; (4) o **especialista** pode criar novos protocolos, mais especializados para a aplicação e que podem ser executados pelos operadores, como exemplo, os físicos médicos para imagens ou químicos em espectroscopia; (5) o **operador**, que apenas pode utilizar os protocolos e refiná-los, ou seja, ajustar os parâmetros dentro da faixa pré-determinada pelo mesmo. Vale ressaltar também que este último está apenas no contexto da Console de operação pois, suas modificações se apresentam como últimos refinamentos para a execução. A figura 31 representa de forma resumida as atribuições de cada um destes grupos de usuários.

Figura 31 - Grupos de usuários e desenvolvedor. Escala de maior para a menor permissão no sistema, de cima para baixo. Fonte: Elaborado pelo autor.

Observe que todos eles devem trabalhar em conjunto com programadores que podem maximizar e melhorar a qualidade do software gerado. Essa visão diferenciada dos sistemas permite com que uma equipe trabalhe em conjunto porém em camadas diferentes, sem necessariamente que cada camada tenha que entender todo o software. No sistema isso é implementado de forma que no código para cada função, método ou propriedade exista a chamada de um *decorator*. Este *decorator* é reimplementado pelo sistema (API especializada) de modo que se tenha controle das operações de cada chamada seguindo os grupos de usuários acima listados, onde aquele que não estiver habilitado não terá como executar tal tarefa. Para tal, um sistema de banco de dados de usuários deverá ser provido e integrado ao sistema.

As definições e estruturas apresentadas até o momento formam a base principal da PyMR, atualmente em conjunto com a linguagem F, das mais importantes atribuições de cada parte que compõe todo o sistema gerado por elas. Como forma de exemplificar algumas das estruturas de classes provida pela PyMR, na figura 32 mostramos a hierarquia de classes que representam os principais núcleos atômicos com informações sobre seu número de spin, fator

giromagnético e métodos que fornecem a frequência de Larmor, se fornecido o valor do campo magnético.

Figura 32 - Diagrama de classes do módulo *nucleus* da PyMR, onde algumas das classes foram omitidas por falta de espaço. Fonte: Elaborado pelo próprio autor com base da documentação da PyMR.

Mais um exemplo como forma de ilustrar as classes oferecidas pela PyMR, pode ser visualizado na figura 33, onde vemos a hierarquia de classes referentes às bobinas. Na Figura 34, é apresentado o diagrama de classe para o magneto permanente.

Figura 33 - Hierarquia de classes referentes às bobina, *Coil* e *CoilSet*, e classes auxiliares que são utilizadas como enumeradores como *GradientCoilType*, *OperationMode*, *PolarizationMode* e *RFProbeType*. Fonte: Elaborado pelo próprio autor com base da documentação da PyMR.

Figura 34 - Diagrama de classes para a classe de magneto permanente (*PermanentMagnet*). Fonte: Elaborado pelo próprio autor com base da documentação da PyMR.

Como uma informação mais generalizada sobre o código, na tabela 1 vemos algumas estatísticas referentes ao código da PyMR, evidenciando as métricas de software propostas na

metodologia como a complexidade ciclomática e manutenibilidade, onde os valores são os melhores possíveis. Nota-se também que a quantidade de linhas de documentação é relativamente alta se comparada com a quantidade de linhas de código, fazendo com que a biblioteca possa ser facilmente utilizada.

Tabela 1 - Estatísticas do código da biblioteca PyMR.

Métrica/Variável	Valor
Arquivos	26
Complexidade ciclomática média [§]	A (1,04)
Manutenibilidade por arquivo ^{**}	A-Todos
Linhas de código total	5242
Linhas de código lógico	1636
Linhas de código	4573
Linhas de documentação	3004
Comentários	431
Em branco	669

Fonte: Elaborado com base nos dados gerados pelo software Radon²³.

Vários outros exemplos de classes poderiam ser apresentados, mas os pontos mais importantes são os conceitos e estruturas geradas, que formam a base para o desenvolvimento e implementação da PyMR.

2.2.2 ToRM-API

A partir do uso e reimplementação de algumas funcionalidades da PyMR, foi gerada a ToRM-API, a qual fornece a interface de programação do hardware específico do projeto ToRM-15, sendo facilmente reconfigurado para novas funcionalidades. Embora a ToRM-API tenha sido concebida antes da PyMR, esta última fornece, agora, os modelos básicos de classes e métodos para serem reimplementados gerando assim a ToRM-API. Assim, não trataremos de modo aprofundado dos itens pertinentes a ToRM-API, pois muitos já foram tratados na seção anterior. Um pequeno exemplo de *setup* de hardware através desta interface pode ser visto abaixo, com o *setup* do pré-amplificador de RF (Miteq AU-1448) para o

[§] Escala de A à F, sendo A melhor e F pior.

^{**} Escala de A à C, sendo A fácil manutenção e C difícil.

sistema de recepção.

```
from pymr.core.system.peripherals import RFPreAmplifier
import scientific.physical.PhysicalQuantity as PhQtt
#////////////////////////////////////
#RF Pre Amplifier - Miteq AU-1448
#////////////////////////////////////

pre_rf = RFPreAmplifier()
pre_rf.label = "RF Pre Amplifier Miteq"
pre_rf.vendor = "Miteq"
pre_rf.model = "AU-1448"
pre_rf.serialNumber = 786215
#Electrical characteristics
pre_rf.inputImpedance = PhQtt(50,'Ohm')
pre_rf.outputImpedance = PhQtt(50,'Ohm')
pre_rf.gain = PhQtt(50,'dB')
pre_rf.frequencyRange(PhQtt(10,'MHz'), PhQtt(400,'MHz'))
pre_rf.noiseFigure = PhQtt(1.2,'dB')
pre_rf.maxInputPower = PhQtt(10,'dBm')
pre_rf.maxInputPower1dB = PhQtt(11,'dBm')
pre_rf.maxInputPowerSaturation = PhQtt(54,'dBm')
pre_rf.maxInputPowerCW = PhQtt(30,'dBm')
pre_rf.maxSWR = 2.0
pre_rf.maxInputVoltagePP = PhQtt(20,'V')
pre_rf.recoveryTime = PhQtt(3.5,'mus')
#Source characteristics
pre_rf.sourceVoltage = PhQtt(15,'V')
pre_rf.sourceCurrent = PhQtt(70,'mA')
```

Nota-se a simplicidade e clareza na descrição do código, além de já incorporar as unidades físicas pertinentes, evitando assim erros que poderiam aparecer por falta do tratamento de unidades. Na tabela 1, podemos verificar, da mesma forma da seção anterior, que os valores para as métricas de software utilizadas são bons. Vale ressaltar neste momento que, a quantidade de documentação é reduzida, pois como trata-se de uma reimplementação da PyMR, as informações básicas encontram-se nela restando, assim, apenas a necessidade de algumas linhas de comentários.

Tabela 2 - Estatística do código do ToRM-API.

Métrica/Variável	Valor
Arquivos	41
Complexidade Ciclomática Média ^{††}	A (1.57)
Manutenibilidade por arquivo ^{‡‡}	A-Todos
Linhas de código total	4664
Linhas de código lógico	2867
Linhas de código	3884
Linhas de documentação	130
Comentários	372
Em branco	780

Fonte: Elaborado com base nos dados gerados pelo software Radon²³.

2.2.3 ToRM-IDE

Assim como projetos de software, uma metodologia ou sequência de pulsos é agora, também, tratada como tal, sendo um projeto de um IDE que pode ser, após finalizado, utilizado para execução na console de controle do equipamento, neste caso específico o ToRM-Console^{§§}. A figura 35 mostra como é a estrutura de um projeto gerado para tal finalidade de forma que contenha todos os elementos necessários para sua execução como os arquivos de código, protocolos e outras informações auxiliares como imagem da sequência resumida.

^{††} Escala de A à F, sendo A melhor ou menos complexo e F pior ou mais complexo.

^{‡‡} Escala de A à C, sendo A fácil manutenção e C difícil.

^{§§} Autor Danilo D. D. M. da Silva, Dissertação de Mestrado, Desenvolvimento de console multiplataforma para aquisição, organização e visualização de dados do Espectrômetro Digital de RM do CIERMag: ToRM Console, IFSC-USP, 2014.

Figura 35 – Estrutura de um projeto de metodologia no ToRM-IDE. Fonte: Elaborado pelo autor com base na documentação interna do CIERMAG.

Como podemos ver pela figura 35, já estão incorporadas as bibliotecas de módulos, eventos e formas de onda para os pulsos. O arquivo *.xml* é uma saída do editor gráfico de sequências^{***} que é compatível com a entrada para o simulador JEMRIS. Os arquivos *parameters.py* e *validation.py* são, respectivamente, o parâmetros utilizados na metodologia e seu *script* de validação, que contém as inter-relações. Por fim, o item com extensão *.trp* é o arquivo gerado pelo ToRM-IDE com informações básicas do projeto. Na figura 36 mostramos o ToRM-IDE com o editor de parâmetros em destaque, onde podem ser visualizados a aba de controle de parâmetros e demais ferramentas. Nas figuras seguintes mostramos, em detalhes, as abas de parâmetros da metodologia (figura 37) e a de parâmetros do sistema (figura 38).

^{***} O editor gráfico de sequência foi produzido por Gustavo Lourenço Vilaça.

Figura 36 - ToRM-IDE em detalhes. Editor de parâmetros em foco com a identificação das principais funcionalidades. Fonte: Elaborado pelo autor com base na documentação interna do CIERMag.

Figura 37 – Editor de parâmetros do ToRM-IDE da metodologia CPMG. Fonte: Elaborado pelo autor com base na documentação interna do CIERMag.

Figura 38 - Editor de parâmetros do ToRM-IDE com os parâmetros do sistema aberto na aba de configuração dos gradientes. Fonte: Elaborado pelo autor com base na documentação interna do CIERMag.

No trecho de código seguinte, mostramos o código em Python gerado pelo ToRM-IDE para arquivo de parâmetros, realizando sua inicialização a partir das informações da interface gráfica. Neste arquivo estão os valores iniciais dos parâmetros que pode ser alterados pelos protocolos.

```

"""
Parameters in Python language

Bits: 32bit
Project: CPMG_E
Created on Tue 03 Dec 2013 11:22:29 AM BRST
Generated with ToRM IDE v.1.2
"""

from pymr.parameters import *

C_TE = ParameterPhQt()
C_TE.levelTwo = "Control"
C_TE.description = "Time to echo"
C_TE.colPos = 0
C_TE.editable = True
C_TE.visible = True
C_TE.levelOne = "Methodology"
C_TE.minimum = "0.0 mus"
C_TE.levelThree = ""
C_TE.rowPos = 0
C_TE.name = "C_TE"
C_TE.script = " "
C_TE.default = "1000.0 mus"
C_TE.realTime = False
C_TE.maximum = "10000000.0 mus"
C_TE.label = "TE"
C_TE.current = "6000.0 ms"

```

Abaixo seguimos com um exemplo de validação de parâmetros, no caso a validação do tempo de repetição (TR), nomeado C_TR, relacionado com o tempo ao eco (TE), nomeado C_TE. Após o teste, se inválido, é adicionado uma mensagem para o usuário.

```

if C_TR.current < 2*C_TE.current:
    error_msg = "TR is too short, try a value less than \
2*TE={}".format(2*C_TE.current)
    errors.append(error_msg)

```

Este modo de reportar erros através do *script* torna-se um fácil método de comunicação entre o programa e o usuário que faz o uso da interface.

Na figura 39, mostramos o editor gráfico de sequências^{†††} que é capaz de gerar os códigos a partir da programação gráfica. Atualmente, o sistema conta com a operação inversa, ou seja, a partir do código fonte, escrito na linguagem F, é gerado a visualização gráfica, a qual pode ser novamente editada e salva em XML. Este tipo de ferramenta não é encontrada em nenhum software comercial de equipamentos de RM, sendo exclusividade de nosso IDE.

^{†††} O editor gráfico de sequência foi produzido por Gustavo Lourenço Vilaça.

Figura 39 - Editor gráfico de seqüências do ToRM-IDE com exemplo de uma seqüência Spin Echo Multi Slices.
Fonte: Elaborado por Gustavo V. Lourenço baseado na documentação interna do CIERMag.

Como forma de finalizar a apresentação do conteúdo do ToRM-IDE, algumas métricas geradas sobre o código podem ser vistas na tabela 3.

Tabela 3 - Estatística do código do ToRM-IDE.

Métrica/Variável	Valor
Arquivos	110
Complexidade Ciclomática Média ⁺⁺⁺	A (2,98)
Manutenibilidade por arquivo ^{\$\$\$}	2-B, 3-C, Resto A
Linhas de código total	26854
Linhas de código lógico	17928
Linhas de código	23320
Linhas de documentação	1835
Comments	1646
Blank	3534

Fonte: Elaborado pelo autor com base nos dados gerados pelo software Radon.

Notamos que o código com cerca de 100 arquivos chega a quase 27.000 linhas de código, dos quais cinco módulos necessitam de algum aprimoramento para melhorar a manutenibilidade. Alguns deles tornaram-se grandes demais ou contém várias classes que

⁺⁺⁺ Escala de A à F, sendo A melhor e F pior.

^{\$\$\$} Escala de A à C, sendo A fácil manutenção e C difícil.

podem ser separadas, inserindo-os como trabalho para próximas versões.

Como informação geral sobre todos os códigos gerados, o que inclui todos os itens, é que são funcionais para as versões de Windows e Linux, testadas para as versões Windows 7, Windows 8 e Ubuntu 12.04, 12.10. Para outras versões do sistema operacional há necessidade de testes de compatibilidade. O mesmo é válido para a PyMR e a ToRM-API, embora nenhum dos códigos estejam com funcionalidades específicas destes sistemas, pois em todos os casos, utilizamos métodos e funções multiplataforma. Na seção seguinte mostramos a execução de uma metodologia no espectrômetro local.

2.2.4 Execução da metodologia no Espectrômetro ToRM

Como dissemos no início do capítulo, a execução e processamento dos dados são realizados em outro ambiente, neste caso o ToRM-Console. Como uma forma de validação em um equipamento real, que permitiria mostrar a funcionalidade do que foi proposto, foi montada uma sequência para execução no equipamento local. A figura 40 mostra a execução da sequência CPMG no ToRM-Console onde podemos ver o sinal adquirido com uma metodologia CPMG em uma amostra de CuSO_4 em solução na concentração de 1g/L.

Figura 40 - ToRM-Console executando a metodologia CPMG com TE=5ms, TR= 4s, 256 ecos e 8 médias na amostra de CuSO_4 em solução na concentração de 1g/l. Fonte: Autor Danilo D. D. M. da Silva, Dissertação de Mestrado, Desenvolvimento de console multiplataforma para aquisição, organização e

visualização de dados do Espectrômetro Digital de RM do CIERMag: ToRM Console⁶⁰.

Após adquirido o sinal, foi executado o *script* de processamento gerando a interpolação do sinal adquirido e retornando a informação do tempo de relaxação T_2 , como pode ser visto na figura 41.

Figura 41 - ToRM-Processing executando a interpolação dos máximos dos ecos adquiridos para retornar a informação do valor de $T_2=276ms$ para a amostra de $CuSO_4$ em solução de $1g/l$. Fonte: Autor Danilo D. D. M. da Silva, Dissertação de Mestrado, Desenvolvimento de console multiplataforma para aquisição, organização e visualização de dados do Espectrômetro Digital de RM do CIERMag: ToRM Console⁶⁰.

O valor encontrado, $T_2=276ms$ da amostra de $CuSO_4$ em solução, encontra-se dentro do valor esperado para esta concentração.⁵⁸ Esta última fase mostra a aplicabilidade em um sistema real, mostrando que os materiais gerados são capazes de controlar o espectrômetro de forma funcional. Mais detalhes do uso e execução de metodologias produzidas com este projeto podem ser visto no dissertação de mestrado de Danilo D. D. M. da Silva.⁶⁰

3 CONCLUSÃO

Tendo como base o estado-da-arte das tecnologias de Computação e da Ressonância Magnética, somado aos requerimentos e especificações dos próprios usuários dos equipamentos de RM, foi gerado um novo sistema, capaz de fornecer as ferramentas necessárias para o desenvolvimento de metodologias em RM. Tal sistema, que inclui a biblioteca PyMR, a ToRM-API e o ToRM-IDE, foi capaz de se adaptar a todas as necessidades previstas até o momento, sem necessidade de grandes modificações, o que condiz com o seu fundamento genérico o suficiente para atender as demandas de hardware existente para equipamentos de RM. A concepção deste trabalho mostrou que a divisão nas três etapas distintas, que se constitui em desenvolvimento, execução e processamento, fornecem dinamismo a forma como é concebida uma metodologia o que, até atualmente, eram unidas em uma única fase e aplicação. Em especial, na fase de desenvolvimento de metodologias, é promovida uma nova visão do desenvolvedor de metodologias para RM dando-lhe grande atenção e fornecendo ferramentas atuais facilitam seu trabalho de pesquisa. Além do próprio desenvolvedor de metodologias, a estrutura proposta fornece ferramentas e facilidades a todos os níveis de usuários, sejam eles administradores, gerenciadores, especialistas ou mesmo operários. Isso é promovido pelas facilidades de *setup* do sistema, criação de metodologias e protocolos, além do uso da interface parâmetros de controle. Em adição, as métricas de software mostraram a qualidade, facilidade de manutenção e utilização das classes, métodos e funções, evitando grandes estruturas hierárquicas, abreviaturas ou grandes inter-relações. Deste modo, o material desenvolvido é capaz de entregar ao pesquisador uma ferramenta que facilite seu trabalho, transformando o desenvolvimento de metodologias de RM em algo efetivo e autônomo. Devido à extensão do projeto, muito não pôde ser coberto, restando vários pontos a serem estudados e desenvolvidos. Contudo este trabalho forneceu as características básicas para algo que pode se tornar um conjunto fundamental de ferramentas para desenvolvedores de metodologias de RM.

3.1 Trabalhos futuros

Pela extensão do trabalho, alguns pontos não foram abordados por não serem essenciais neste momento, mas seguem como sugestão de trabalhos futuros: **Plug-ins para IDE's já existentes:** transformação total do ToRM-IDE em plug-ins para IDE como Eclipse⁴⁶ e Eric⁴⁵, por exemplo, contemplando assim os itens mencionados no capítulo da introdução na seção IDE, fazendo com que tais IDE's fiquem especializados para o desenvolvimento de metodologias de RM, reaproveitando seus inúmeros aperfeiçoamentos agregados durante sua evolução. **Incorporar simuladores:** incluir plug-ins que integrem mais funcionalidades do JEMRIS e mesmo outros simuladores de sistemas como SIMRI, GAMMA, VESPA além de simuladores de pulsos, bobinas, etc. **Visualização de inter-relação entre parâmetros:** implementar uma possível interface de gerenciamento de inter-relação entre parâmetros via visualização por grafos dinâmicos, onde cada nó representa um parâmetro e suas inter-relações são os vértices. Isso, somado com as possíveis informações trazidas pela teoria de grafos como a centralidade, por exemplo, trariam novas ferramentas que nunca foram utilizadas para tal fim. **Testes Beta:** realizar testes fora do ambiente de desenvolvimento para validação dos códigos gerados e pesquisa com desenvolvedores de sequências e demais usuários para seu aprimoramento. **Espectrômetro online:** nosso espectrômetro é capaz de se comunicar via Ethernet, isso traria grande vantagem para o uso no aprendizado por pessoas em outros locais, servindo como uma base de teste para os usuários experimentarem o novo sistema. **Incorporar unidades físicas no NumPy/SciPy:** embora existam inúmeras bibliotecas que fazem o tratamento de unidades físicas, nenhuma delas consegue ser completa, com todos os tipos de operações, tratamento de erros, conversões etc. Sugere-se a inserção de um novo tipo no NumPy⁴⁹ de forma que tais operações possam ser trabalhadas incluindo sua implementação interna em C que melhoraria o desempenho do código. **Garantia de integridade dos arquivos do sistema e metodologias:** todos os arquivos que fazem parte do sistema podem e devem estar sobre proteção do sistema operacional pelo gerenciamento de suas permissões. Além disso, para garantir a integridade da fonte dos arquivos é recomendado usar alguma função de *hash* como, por exemplo, o MD5⁵⁹ procedendo ao cálculo do mesmo toda vez que iniciado o sistema para verificar a integridade dos dados (MD5Sum). Desta forma, há garantias de que o que está sendo executado está vindo de fontes seguras e/ou conhecidas.

REFERÊNCIAS

- 1 TANNÚS, A. **Desenvolvimento da tecnologia de tomografia por ressonância magnética nuclear**. 1987. 229 p. Tese (Doutorado em Física Aplicada) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 1987.

- 2 MARTINS, M. J. **Desenvolvimento de um tomógrafo de ressonância magnética: integração e otimização**. 1995. 78 p. Tese (Doutorado em Ciências) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 1995.

- 3 BELMONTE, S. B. et al. A field-programmable gate-array-based high-resolution pulse programmer. **Measurement Science and Technology**, v. 14, n. 1, p. N1–N4, 2003. doi:10.1088/0957-0233/14/1/401.

- 4 MACDONALD-JANKOWSKI, D. S. Magnetic resonance imaging. Part 1: the basic principles. **Asian Journal of Oral and Maxillofacial Surgery**, v. 18, n. 3, p. 165–171. doi:10.1016/S0915-6992(06)80013-0, 2006.

- 5 BUXTON, R. B. **Introduction to functional magnetic resonance imaging: principles and techniques**. Cambridge: Cambridge University Press, 2002.

- 6 HAGE, M. C. F. N. S.; IWASAKI, M. Imagem por ressonância magnética: princípios básicos. **Ciência Rural**, v. 39, n. 4, p. 1275–1283, 2009. doi:10.1590/S0103-84782009005000041.

- 7 HAACKE, E. et al. **Magnetic resonance imaging: physical principles and sequence design**. New York: Wiley-Liss, 1999. 937 p.

- 8 WESTBROOK, C. **Handbook of MRI technique**. 2nd. ed. [s.l.]: Clackwell Science, 2012. p.464

- 9 MRI sequences: spin echo, gradient echo, echo planar. Disponível em: <<http://www.imaio.com/en/e-Courses/e-MRI/MRI-Sequences>>. Acesso em: 30 ago. 2013.

- 10 MR-TIP: MRI sequences. Disponível em: <<http://www.mr-tip.com/serv1.php?type=seq>>. Acesso em: 30 ago. 2012.

11 SCHMITHORST, V. J.; DARDZINSKI, B. J. Automatic gradient preemphasis adjustment: a 15-minute journey to improved diffusion-weighted echo-planar imaging. **Magnetic Resonance in Medicine**, v. 47, n.1, p. 208–212, 2002. doi:10.1002/mrm.10022.

12 LECHLITER, M. **Pre and de emphasis - explanation and assistance**: running the numbers. Disponível em: <<http://www.repeater-builder.com/tech-info/pdemph-post.html>>. Acesso em: 16 set. 2013.

13 LI, Y. et al. Image homogenization using pre-emphasis method for high field MRI. **Quantitative Imaging in Medicine and Surgery**, v. 3, n. 4, p. 217–223, 2013. doi:10.3978/j.issn.2223-4292.2013.07.01.

14 INTRODUCTION TO NMR/MRI RF AMPLIFIERS. Disponível em: <<http://cpcamps.com/introduction-to-nmr-mri-amplifiers.html>>. Acesso em: 25 set. 2013.

15 VAUGHAN, J. T. ; GRIFFITHS, J. R. (Ed.). **RF coils for MRI**. [s.l.]: Wiley, 2012. p. 468

16 VIANA, R. M. Programação em GPU: passado, presente e futuro. Disponível em: <<http://www.ufpi.br/subsiteFiles/ercemapi/arquivos/files/minicurso/mc1.pdf>>. Acesso em: 13 jan.2014.

17 STECKNER, M. C.; HITACHI Medical Systems America, Inc. Advances in MRI equipment design, software, and imaging procedures. 2006. Disponível em: <<http://www.aapm.org/meetings/amos2/pdf/26-5961-46702-744.pdf>>. Acesso em: 20 jun. 2012.

18 AI, T. et al. A historical overview of magnetic resonance imaging, focusing on technological innovations. **Investigative radiology**, v. 47, n. 12, p. 725–41, doi:10.1097/RLI.0b013e318272d29f, 2012.

19 NIELSEN, J. Finding usability problems through heuristic evaluation. In: PROCEEDINGS OF THE SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS (CHI'92), 1992. **Anais...** New York, NY, USA: ACM, 1992. Disponível em: <<http://portal.acm.org/citation.cfm?doid=142750.142834>>. Acesso em: 13 jan.2014. p. 373-380

20 NIELSEN, J. Usability inspection methods. In: CONFERENCE COMPANION ON HUMAN FACTORS IN COMPUTING SYSTEMS (CHI'94), 1994. **Anais...** New York, NY, USA: ACM, 1994. Disponível em: <<http://portal.acm.org/citation.cfm?doid=259963.260531>>. Acesso em: 13 jan 2014. p. 413-4.

21 DUMAS, J. S. The Human-computer Interaction Handbook. In: JACKO, J. A.; SEARS, A. (Eds.). Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 2003. p. 1093–1117.

22 SCRUM.ORG: the home of Scrum. Disponível em: <<https://www.scrum.org/Resources>>. Acesso em: 26 jun. 2012.

23 RADON. Disponível em: <<https://radon.readthedocs.org>>. Acesso em: 10 dez. 2013.

24 TRAVIESO, G. **Introdução à orientação a objetos em C++**. São Carlos: Universidade de São Paulo, 2007. 181 p.

25 THE OPEN source initiative. Disponível em: <<http://opensource.org/>>. Acesso em: 26 jul. 2012.

26 INSIDER, E. T. **7 erros mais comuns na hora de criar uma API**. Disponível em: <<http://techinsider.com.br/2012/06/22/7-erros-mais-comuns-na-hora-de-criar-uma-api/>>. Acesso em: 30 jul. 2012.

27 JACQUES, M. **API usability** : guidelines to improve your code ease of use. Disponível em: <<http://www.codeproject.com/Articles/8707/API-Usability-Guidelines-to-improve-your-code-ease>>. Acesso em: 8 out. 2012.

28 VENNERS, B. **Perfection and simplicity**: a conversation com Ken Arnold, part 1. 2002. Disponível em: <<http://www.artima.com/intv/perfectP.html>>. Acesso em: 8 out. 2012.

29 TULACH, J. **Practical API design**: confessions of a java framework architect. Berkeley: Apress, 2008. 417 p.

30 BLANCHETTE, J. **The little manual of API design**. [s.l.]: Trolltech, a Nokia company, 2008. p. 33

31 RONACHER, A. **Good API design**: study, improve & create. Ukraine: [s.n.]. 2010. Disponível em: <<http://dev.pocoo.org/~mitsuhiko/API.pdf>>. Acesso em: 13 jan. 2014.

32 DOURNAEE, B. **Essential elements of API management**. Disponível em: <http://blogs.intel.com/security-gateways/2011/09/13/essential_elements_of_api_mana/>. Acesso em: 8 out. 2012.

33 UML. Disponível em: <<http://www.uml-diagrams.org/>>. Acesso em: 13 set. 2012.

34 LEHRAUM, J.; WEINBERG, B. **EE times**: IDE evolution continues beyond eclipse. Disponível em: <http://www.eetimes.com/document.asp?doc_id=1150355>. Acesso em: 15 jun. 2013.

- 35 NAGL, M. **Building tightly integrated software development environments: the IPSEN approach**. Berlin/Heidelberg: Springer-Verlag, 1996. (Lectures notes in computer science, v. 1170)
- 36 FILLER, A. G. The history, development and impact of computed imaging in neurological diagnosis and neurosurgery: CT , MRI, and DTI. 1983. Disponível em: <<http://dx.doi.org/10.1038/npre.2009.3267.5>> . Acesso em: 13 jan. 2014. p. 1–76.
- 37 STONE, S. S. et al. Accelerating advanced MRI reconstructions on GPUs. **Journal of Parallel and Distributed Computing**, v. 68, n. 10, p. 1307–1318, 2008. doi:10.1016/j.jpdc.2008.05.013.
- 38 JOCHIMSEN, T. H.; MENGERSHAUSEN, M. von. ODIN-object-oriented development interface for NMR. **Journal of Magnetic Resonance**, v. 170, n. 1, p. 67–78, 2004. doi:10.1016/j.jmr.2004.05.021.
- 39 GÄDKE, A. et al. DAMARIS - a flexible and open software platform for NMR spectrometer control. **Magnetic Resonance Imaging**, v. 25, n. 4, p. 576, doi:10.1016/j.mri.2007.01.081, 2007.
- 40 DEBBINS, J. et al. Novel software architecture for rapid development of magnetic resonance applications. **Concepts in Magnetic Resonance**, v. 15, n. 3, p. 216–237, 2002. doi:10.1002/cmr.10039.
- 41 SILVA, D. M. D. D. et al. Subsistema multiplataforma para controle de aquisição, visualização e organização de dados do espectrômetro digital de RM ToRM console. In: CONGRESSO BRASILEIRO DE FÍSICA MÉDICA, 18., 2013, São Pedro. **Anais...** São Pedro: Associação Brasileira de Física Médica - ABFM, 2013. Disponível em: <<http://www.brasilrad.com.br/anais/XVIIIICBFM/PDF/COINFO4.pdf>>. Acesso em: 2 set. 2013
- 42 PROGRAMMING language popularity. Disponível em: <<http://www.langpop.com/>>. Acesso em: 26 nov. 2012.
- 43 MERCURIAL SCM. Disponível em: <<http://mercurial.selenic.com/>>. Acesso em: 15 mar. 2012.
- 44 REDMINE. Disponível em: <<http://www.redmine.org/>>. Acesso em: 28 maio. 2012.
- 45 THE ERIC Python IDE. Disponível em: <<http://eric-ide.python-projects.org/>>. Acesso em: 20 mar. 2013.

46 ECLIPSE - the eclipse foundation open source community website. Disponível em: <<http://www.eclipse.org/>>. Acesso em: 30 set. 2013.

47 EPYDOC. Disponível em: <<http://epydoc.sourceforge.net/>>. Acesso em: 2 dez. 2012

48 PYTHON programming language – official website. Disponível em: <<http://www.python.org/>>. Acesso em: 2 fev. 2012.

49 NUMPY. Disponível em: <<http://www.numpy.org/>>. Acesso em: 2 mar. 2013.

50 JSON. Disponível em: <<http://www.json.org/>>. Acesso em: 6 jun. 2012.

51 HINSEN, K. **Bitbucket:** scientificpython. Disponível em: <<https://bitbucket.org/khinsen/scientificpython>>. Acesso em: 26 jun. 2013.

52 QT project: PySide - python for Qt. Disponível em: <<http://qt-project.org/wiki/PySide>>. Acesso em: 6 mar. 2012.

53 QT - cross-platform application and UI development framework. Disponível em: <<http://qt.digia.com/>>. Acesso em: 5 mar. 2013.

54 GUI programming with PyQT. Disponível em: <<http://www.commandprompt.com/community/pyqt/>>. Acesso em: 14 fev. 2012.

55 MOHR, P. Fundamental Constants and the International System of Units. In: CODATA INTERNATIONAL CONFERENCE, 20., 2006, Beijing. **Anais...** Beijing: [s.n.], 2006. Disponível em: <<http://www.codata.org/06conf/presentations/KeySessions/I3/PeterMohr.pdf>>. Acesso em: 2 mar. 2013

56 TAYLOR, B. N.; THOMPSON, A. **The international system of units**. 8th. ed. Gaithersburg: National Institute of Standards and Technology, 2008. 96 p.

57 XML tutorial. Disponível em: <<http://www.w3schools.com/xml/>>.

58 PRICE, R. R. et al. Quality assurance methods and phantoms for magnetic resonance imaging. **Medical Physics**, v.17, n.2 p. 287-95, 1990.

59 RIVEST, R. L. **The MD5 message-digest algorithm**. Cambridge: MIT, 1992. Disponível em: <<http://tools.ietf.org/html/rfc1321>>. Acesso em: 6 jan. 2014.

60 DA SILVA, D.M.D.D. **Desenvolvimento de console multiplataforma para aquisição, organização e visualização de dados do Espectrômetro Digital de RM do CIERMag: ToRM Console**. 2014. Dissertação (Mestrado em Física Aplicada: Física Computacional) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 1987

APÊNDICE A – Questionário aplicado aos usuários e programadores de sequências

CIERMag IFSC USP - Questionário para desenvolvimento de API e IDE para Ressonância Magnética

Caro(s) convidado(s),

você(s) foram convidados a participar de uma pesquisa que visa fornecer dados sobre a utilização de equipamentos e softwares de Ressonância Magnética (RM), contribuindo para um projeto do CIERMag - IFSC - USP. Caso queiram participar, seguem abaixo algumas informações, bastando depois, somente clicar em continuar, onde você(s) serão redirecionados ao questionário. Sua ajuda é de grande importância para nossa pesquisa. Ela foi direcionada a usuários de RM e em especial a programadores de sequências. O link estará disponível por 15 dias a partir da data de recebimento do e-mail. O tempo estimado para responder é de cerca de 30min. Caso alguém tenha pessoas para indicar, por favor, repassem o link do questionário ou nos envie o contato.

Sobre o projeto

Nós do CIERMag, estamos a frente do desenvolvimento de uma API (*Application Programming Interface*) e de uma IDE (*Integrated Development Environment*) para o desenvolvimento de sequências de pulsos para Ressonância Magnética (RM) em Imagens e Espectroscopia, de forma generalizada. A que tudo indica, esta nova forma de abordagem pode se tornar uma ferramenta importante e facilitadora no desenvolvimento de sequências nos mais diversos equipamentos, sendo multiplataforma, livre e independente do hardware. Particularmente este meu projeto de mestrado se conecta com o desenvolvimento do Espectrometro Digital de RM desenvolvido pelo CIERMag no projeto atual ToRM-15 (Tomógrafo de Ressonância Magnética 1.5T), o qual foi concebido de forma a ser desde um simples relaxômetro a um equipamento analítico e de imagens num mesmo equipamento.

Sobre o questionário

As informações obtidas com este questionário terão como objetivo auxiliar o desenvolvimento da API e da IDE. Informamos que nenhuma das respostas da pesquisa serão divulgadas com informações que possam identificar os entrevistados ou que gerem algum tipo de relação com empresas do ramo citadas no texto. Sinta-se a vontade para não responder questões que não são de seu agrado. Informamos também que, em todas as questões, existem palavras chaves em caixa alta para facilitar a leitura e agilizar suas respostas. O tempo estimado para responder o todo o questionário é de cerca de 30 minutos. O questionário está subdividido em 4 páginas com 29 questões, no total, que representam os nichos de perguntas sobre:

- 1 - Uso dos equipamentos (software) de RM (5 questões);
- 2 - Programação de sequências de pulsos nos equipamentos de RM (10 questões);
- 3 - Sistema operacional e linguagens de programação (7 questões);
- 4 - Formação profissional e atuação (6 questões)

Desde já agradecemos a todos os convidados.

1 - Uso dos equipamentos (softwares) de RM (5 questões)

Os equipamentos serão identificados pelo fabricante/marca. Tratamos aqui somente sobre o uso, não quanto a programação de sequências. Ressaltamos que nenhuma das respostas da pesquisa serão divulgadas com informações que possam identificar os entrevistados ou que gerem algum tipo de relação com empresas do ramo citadas no texto

1.1 Qual sua FAMILIARIDADE COM O USO dos equipamentos(s)?

Nesta questão estamos tratando especificamente dos softwares de uso. No caso de escolher NÃO LISTADO, por favor, nos informe o nome nas sugestões na última página para que possamos incluir.

	Muito familiarizado	Pouco	Nada familiarizado	Não quero opinar
Bruker	☐	☐	☐	☐
GE	☐	☐	☐	☐
Phillips	☐	☐	☐	☐
Toshiba	☐	☐	☐	☐
Varian/Agilent	☐	☐	☐	☐
Siemens	☐	☐	☐	☐
TecMag	☐	☐	☐	☐
Maran	☐	☐	☐	☐
Magritek	☐	☐	☐	☐
NÃO LISTADO	☐	☐	☐	☐

1.2 No caso de ter conhecimento em mais de um equipamento, qual o nível de SEMELHANÇA NO USO entre eles?

Você pode avaliar quanto à quantidade de aprendizado necessário para realizar a mesma tarefa nos diferentes equipamentos.

1 2 3 4 5

Pouca semelhança ☐ ☐ ☐ ☐ ☐ Muita semelhança

1.3 Para o(s) equipamento(s) que tem mais familiaridade, qual é a PRATICIDADE do software para uso do equipamento?

Considere por exemplo, coisas que você precisaria fazer (ou gostaria, pois existe em outro equipamento) mas não existe esta funcionalidade no software.

1 2 3 4 5

Pouco prático ☐ ☐ ☐ ☐ ☐ Muito prático

1.4 Se você não programa sequências e apenas faz uso delas, você NÃO PROGRAMA PORQUÊ:

Caso você programe, desconsidere esta questão.

- ☐ Tenho pouco conhecimento na física da ressonância magnética
- ☐ Tenho pouco conhecimento no equipamento
- ☐ Minha área de atuação é outra
- ☐ Dificuldade em criar novas sequências
- ☐ Tenho pouco conhecimento de programação

1.5 Cite algumas características as quais poderiam MELHORAR a interação entre o equipamento de RM e o usuário.

Exemplos: Mais informações na tela de setup de parâmetros, a forma de atualizar ou validar os parâmetros, etc.

2 - Programação de seqüências nos equipamentos de RM (10 questões)

Caso você não programe, desconsidere estas questões e avance para a próxima página. Ressaltamos que nenhuma das respostas da pesquisa serão divulgadas com informações que possam identificar os entrevistados ou que gerem algum tipo de relação com empresas do ramo citadas no texto

2.1 Avalie quanto a PROGRAMAÇÃO GRÁFICA de seqüências, sua interação com os softwares/equipamentos abaixo.

Modo gráfico de programação se refere a possibilidade, por exemplo, de drag/drop de pulsos, tabelas, etc. Caso não exista essa opção, marque Não se aplica. No caso de escolher NÃO LISTADO, por favor, nos informe o nome nas sugestões na última página para que possamos incluir.

	Bom/Ótimo	Regular	Ruim/Péssimo	Não se aplica	Não quero opinar / Não sei
Bruker	☐	☐	☐	☐	☐
GE	☐	☐	☐	☐	☐
Phillips	☐	☐	☐	☐	☐
Toshiba	☐	☐	☐	☐	☐
Varian/Agilent	☐	☐	☐	☐	☐
Siemens	☐	☐	☐	☐	☐
TecMag	☐	☐	☐	☐	☐
Maran	☐	☐	☐	☐	☐
Magritek	☐	☐	☐	☐	☐
NÃO LISTADO	☐	☐	☐	☐	☐

2.2 Avalie quanto a PROGRAMAÇÃO TEXTUAL de seqüências, sua interação com os softwares/equipamentos abaixo.

Avalie a forma de descrição da seqüência em forma textual, o modo como deve ser realizado, a estrutura e o código do programa. Caso não exista essa opção, marque Não se aplica. No caso de escolher NÃO LISTADO, por favor, nos informe o nome nas sugestões na última página para que possamos incluir.

	Bom/Ótimo	Regular	Ruim/Péssimo	Não se aplica	Não quero opinar / Não sei
Bruker	☐	☐	☐	☐	☐
GE	☐	☐	☐	☐	☐
Phillips	☐	☐	☐	☐	☐
Toshiba	☐	☐	☐	☐	☐
Varian/Agilent	☐	☐	☐	☐	☐
Siemens	☐	☐	☐	☐	☐
TecMag	☐	☐	☐	☐	☐
Maran	☐	☐	☐	☐	☐
Magritek	☐	☐	☐	☐	☐
NÃO LISTADO	☐	☐	☐	☐	☐

2.3 Quando a sequência possui erros, dos mais variados tipos, como se dá a DEPURAÇÃO (caso geral)?

Avalie a facilidade com que se detecta e com que se resolve o problema. Exemplos: Erro de Sintaxe: classe no lugar de class (Python, C++). Erro de compilação: ('a'+3)/2 ao invés de (a+3)/2 Erro de execução: tempo = 'a', esperava-se um número.

	Com facilidade	Como outra linguagem qualquer	Com dificuldade	Não sei estimar
Erros de sintaxe	☐	☐	☐	☐
Erros de compilação	☐	☐	☐	☐
Erros de execução	☐	☐	☐	☐

2.4 No caso de ter conhecimento em mais de um equipamento, qual o nível de SEMELHANÇA NA PROGRAMAÇÃO de sequências entre eles?

Você pode avaliar quanto à quantidade de aprendizado necessário para realizar a mesma tarefa nos diferentes equipamentos.

1 2 3 4 5

Pouca semelhança ☐ ☐ ☐ ☐ ☐ Muita semelhança

2.5 Você recebeu TREINAMENTO PARA PROGRAMAR sequências de pulsos nos equipamentos citados?

- ☐ Sim, da empresa que vendeu/montou o equipamento
- ☐ Sim, mas de especialistas que não eram da empresa
- ☐ Não, apenas lendo os manuais e buscando informações por conta própria

2.6 Quando há necessidades de grandes modificações nas sequências QUEM realiza esta tarefa?

Relacionado ao software em si.

- ☐ Eu mesmo
- ☐ Alguém mais especializado do grupo de trabalho
- ☐ Técnico da empresa que faz o gerenciamento/manutenção
- ☐ Other:

2.7 Considerando o caso exposto acima e sua resposta, COMO é realizada a tarefa?

- ☐ Com facilidade
- ☐ Com pouca facilidade
- ☐ Com pouca dificuldade
- ☐ Com dificuldade

2.7 Considerando o caso exposto acima e sua resposta, COMO é realizada a tarefa?

- ☐ Com facilidade
- ☐ Com pouca facilidade
- ☐ Com pouca dificuldade
- ☐ Com dificuldade

2.8 Cite algumas características as quais poderiam MELHORAR a interação entre o equipamento de RM e o usuário desenvolvedor de sequências.

Exemplo: Criação de um editor gráfico de sequências, maior liberdade na criação, melhor documentação, etc.

2.9 Quais são suas maiores DIFICULDADES em programar as sequências de pulsos de modo geral?

Por exemplo: falta de editores gráficos, dificuldade no aprendizado da linguagem, dificuldade em programação, etc.

2.10 Quais os principais itens que julga serem responsáveis pela FLEXIBILIDADE das sequências criadas?

Por exemplo: possibilidade de mudanças em loops, os quais algumas empresas mantêm fixos.

3 - Sistema operacional e linguagens de programação (8 questões)

As questões a seguir foram elaboradas visando o ambiente onde são desenvolvidos os códigos e as linguagens de seu conhecimento. Ressaltamos que nenhuma das respostas da pesquisa serão divulgadas com informações que possam identificar os entrevistados ou que gerem algum tipo de relação com empresas do ramo citadas no texto

3.1 Quais SISTEMAS OPERACIONAIS costuma usar?

Considere qualquer distribuição ou derivados para os sistemas operacionais.

- ☐ Windows (WinXP, Win7, Win8, etc)
- ☐ Linux (Ubuntu, RedHat, Debian, Fedora, etc)
- ☐ MacOS (MacOS 5, MacOS 6, etc)
- ☐ Other:

3.2 Quais EDITORES DE TEXTO você usa para editar as sequências de pulsos?

Nesta questão tratamos apenas de EDITORES SIMPLES, ambientes de desenvolvimento são tratados na questão abaixo. Editores de texto que promovam SOMENTE o highlight das palavras reservadas são considerados editores simples.

- ☐ NotePad, NotePad++
- ☐ VI ou VIM
- ☐ GEdit
- ☐ Other:

3.3 Já utilizou algum tipo de AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE?

Ambientes de desenvolvimento são aqueles que promovem auto-completar, informam as variáveis criadas, métodos, funções, classes, estruturas, auxiliam no debug, etc.

- ☐ Eclipse
- ☐ NetBeans
- ☐ Eric
- ☐ Visual Studio
- ☐ Other:

3.3 Já utilizou algum tipo de AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE?

Ambientes de desenvolvimento são aqueles que promovem auto-completar, informam as variáveis criadas, métodos, funções, classes, estruturas, auxiliam no debug, etc.

- ☐ Eclipse
- ☐ NetBeans
- ☐ Eric
- ☐ Visual Studio
- ☐ Other:

3.4 Quais LINGUAGENS DE PROGRAMAÇÃO está familiarizado?

Nesta questão consideramos uma forma abrangente da definição de linguagem de programação, não especificamente para RM.

- ☐ Assembly
- ☐ C
- ☐ C++
- ☐ C#
- ☐ Fortran
- ☐ Java
- ☐ MatLab
- ☐ Python
- ☐ Visual Basic
- ☐ .NET
- ☐ Objective C
- ☐ Other:

3.5 Escolha uma LINGUAGEM DE PROGRAMAÇÃO FAVORITA e descreva o motivo da escolha abaixo.

Diga qual sua linguagem favorita e descreva resumidamente qual o motivo da escolha.

3.6 Qual seu conhecimento sobre PROGRAMAÇÃO ORIENTADA A OBJETOS (P.O.O)?

Conhecimentos relacionados as estruturas de classes, métodos, atributos, herança, herança multipla, polimorfismo, etc.

- ☐ Muito, conheço bem
- ☐ Razoável, mas somente o básico como classes, atributos, métodos e herança
- ☐ Pouco, mas ouvi falar
- ☐ Nenhum, não conheço

3.5 Escolha uma LINGUAGEM DE PROGRAMAÇÃO FAVORITA e descreva o motivo da escolha abaixo.

Diga qual sua linguagem favorita e descreva resumidamente qual o motivo da escolha.

3.6 Qual seu conhecimento sobre PROGRAMAÇÃO ORIENTADA A OBJETOS (P.O.O)?

Conhecimentos relacionados as estruturas de classes, métodos, atributos, herança, herança multipla, polimorfismo, etc.

- ☐ Muito, conheço bem
- ☐ Razoável, mas somente o básico como classes, atributos, métodos e herança
- ☐ Pouco, mas ouvi falar
- ☐ Nenhum, não conheço

3.7 Já fez uso de FERRAMENTAS DE SIMULAÇÃO para RM?

Considera qualquer tipo de ferramenta, por exemplo, criação de pulsos, desenho de bobinas, etc. Este pergunta reflete em nossa possibilidade de integrar simuladores de maior uso.

4 - Questões de nível profissional e de futuros contatos (6 questões)

Sinta-se à vontade para deixar de responder as questões e/ou não se comprometer com esta pesquisa.

4.1 Qual sua FORMAÇÃO?

Caso queira descrever de forma mais detalhista, você pode colocar o endereço de seu CV Lattes.

4.2 Qual sua ATUAÇÃO?**4.3 Qual sua IDADE?****4.4 Gostaria de ser AVALIADOR das versões teste do IDE e da API?**

SUA CONTRIBUIÇÃO SERIA DE MUITA IMPORTÂNCIA: Forneceremos os procedimento para utilização da IDE e da API e um questionário/entrevista durante a montagem de uma simples sequência de pulsos.

- ☐ Sim
- ☐ Me comunique sobre a possibilidade, mas não confirmo a participação
- ☐ Não

4.5 Caso deseje deixe seu e-mail abaixo.

Contato realizado como informado na questão anterior.

4.6 Espaço para sugestões, recomendações e críticas.